Tutorial on ontological engineering

Riichiro Mizoguchi
The Institute of Scientific and Industrial Research, Osaka University
Email: miz@ei.sanken.osaka-u.ac.jp

PREFACE

This tutorial course describes the current state of the art of ontological engineering which is a successor of knowledge engineering. It covers theory, tools and applications and consists of three parts: Part 1 is an introduction to ontological engineering, Part 2 describes ontology development, languages and tools, and Part 3 is an advanced course dealing with philosophical issues of ontology design together with detailed guidelines of ontology development. Part 3 also presents a success story of ontological engineering with the deployment result in a company. The philosophy behind this tutorial is that ontological engineering is viewed as a challenge to enabling knowledge sharing and reuse which knowledge engineering failed to realize. Therefore, one of the major topics dealt with in this tutorial is to explain what an ontology should be while explaining how it is understood currently.

For readers' information, the following is the contents of Parts 2 and 3:

Part 1: Introduction to Ontological Engineering

1. INTRODUCTION

Ontology has been attracting a lot of attention recently. While ontology research has begun in the early 90's in the knowledge base community, the research activity has been accelerated and spread over the web technology community by the semantic web movement in the last few years. Many people talk about ontology nowadays. However, there seems to be some misunderstandings about what an ontology is and what ontological engineering is, how it is useful, etc. The purpose of this tutorial is to clarify issues on ontology and ontological engineering and to describe the author's personal views on how ontological engineering should contribute to the future knowledge processing.

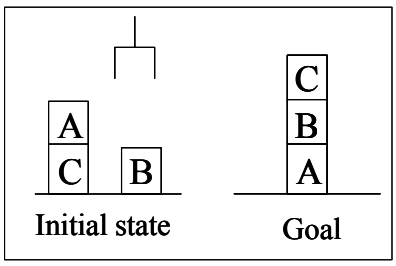
The first topic is what an ontology is followed by the explanation of the context where ontology issues need to be discussed. Ontological engineering is explained as the successor of knowledge engineering to give a first context. Another context to understand ontology is semantic web[SW] which requires semantic interoperability among metadata in which an ontology is expected to fill the semantic gap between metadata. The next topic is typology of ontology followed by FAQ and what is not an ontology to avoid possible misunderstanding of ontology. Roles of ontology are described in detail to explain the utility of an ontology. The following is the contents of this article.

2. WHATT IS AN ONTOLOGY?

2.1 A concrete example

Let me begin the discussion by presenting a small concrete example of ontology of the block world shown in Fig. 1. Although the example used is notorious as a toy problem, I intentionally

chose it because of the following two reasons: (1) It is enough to show what an ontology is. Practicality of an ontology is not an issue here which will be discussed in the next article. (2) It provides us with two convincing conceptualizations which are helpful to demonstrate the necessity of ontology as an explicit specification of conceptualization. One of the key concepts is *Conceptualization* which is an AI term and is defined as "a set of objects which an observer thinks exist in the world of interest and relations between them" [Genesereth 87]. *Conceptualization* plays a role of the backbone of the conceptual structure of the world of interest. The task of the block world is to stack blocks like the Goal configuration using a robot hand. No other objects or tasks exist in this tiny world. A conceptualization, called Conceptualization-1, of this tiny world would be one shown in Fig. 1b, that is, three blocks, a robot hand and a table together with four relations such as on(X,Y), clear(X), holding(X), and handEmpty. What is important here is there can be another conceptualization of this block world. Another conceptualization, called Conceptualization-2, is shown in Fig. 1c where neither table nor robot hand is recognized as an object and a relation **onTable**(X) is added instead. Because the task specified is very simple and the table is unique in the world, the table does not have to be recognized as an object. It only plays the role of anything which directly supports a block and hence the relation onTable(X) can replace on(X, table). The same applies to the robot hand. We can obtain an ontology from each of the two conceptualizations as shown in Fig. 2.



(1a)

Conceptualization-1

Object	**Relation**
block A	on(X, Y)
block B	clear(X)
block C	holding(X)
table A	handEmpty
hand A	

(1b)

Conceptualization-2

Object	**Relation**
block A	on(X, Y)
block B	clear(X)
block C	**onTable(X)**
	holding(X)
	handEmpty

(1c)

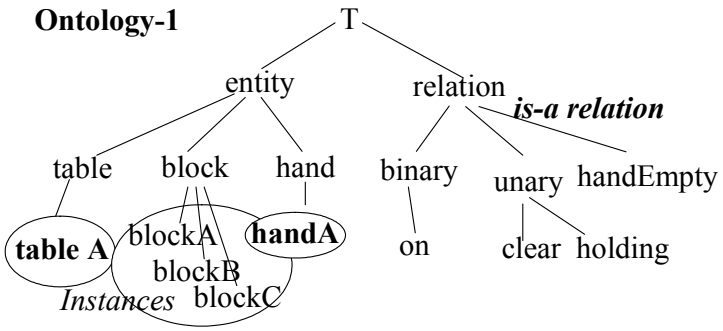
Fig. 1 Conceptualization of a block world

These two example ontologies roughly tell us that an ontology consists of a hierarchically organized concepts and relations between them which explain objects appearing in the target world as their instances. Concepts and relations are defined in terms of slots and axioms. A simple example of an axiom is shown in Fig. 2 in which a new predicate *above* is introduced and defined using *on* relation.

Imagine two persons develop a program to solve this block stacking task with unconsciously different conceptualizations of the target world. These conceptualizations are kept implicit as usual. When they start to communicate each other on how their programs work, they must have a trouble in the communication because of the difference of their understanding of the world. The situation would be worse if the collaboration is required between the two programs, since one believes there exists a table but the other does not. In most cases, such a conceptualization is left implicit and hence it is very difficult to diagnose the influent communication to identify the cause and to recover from the difficulty. However, if such ontologies are available, then it is very helpful to pinpoint the cause of the misunderstanding and how to realize the fluent communication during the collaboration between the two programs. Thus, an ontology is a declarative description of fundamental understanding of an agent on the world of interest and is useful to come to a mutual understanding through explication of what is left implicit.

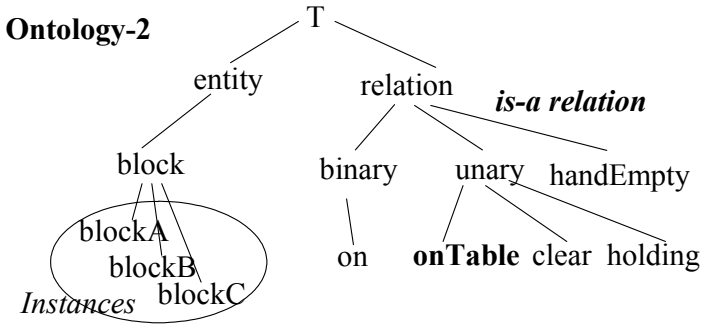
2.2 Definitions

There are many interpretations about what an ontology is. In fact, hot discussions are often done in many meetings on ontology. However, the ontology community has come to an agreement on giving



(a) Ontology based on the conceptualization-1

Axiom : above(X,Z):-on(X,Y), on(Y,Z).



(b) Ontology based on the conceptualization-2

Fig. 2 An ontology of the block world.

up its definition. The following are some of the definitions of an ontology.

(1) In philosophy, it means theory of existence. It tries to explain what is being and how the world is configured by introducing a system of critical categories to account things and their intrinsic relations.
(2) From AI point of view, an ontology is defined as "explicit specification of conceptualization" [Gruber] which is widely accepted in AI community. "Conceptualization" here should be interpreted as "intensional" rather than "extensional" conceptualization contrary to that defined in [Genesereth 87].
(3) From knowledge-based systems point of view, it is defined as "a theory(system) of concepts/ vocabulary used as building blocks of an information processing system" by R. Mizoguchi[Mizoguchi 95]. In a context of problem solving, ontologies are divided into two types: Task ontology for problem solving process and domain ontology for the domain where the task is performed.
(4) Another given by Gruber[Gruber]
Ontologies are agreements about shared conceptualizations. Shared conceptualizations include conceptual frameworks for modeling domain knowledge; content-specific protocols for communication among inter-operating agents; and agreements about the representation of particular domain theories. In the knowledge sharing context, ontologies are specified in the form of definitions of representational vocabulary. A very simple case would be a type hierarchy, specifying classes and their subsumption relationships. Relational database schemata also serve as ontologies by specifying the relations that can exist in some shared database and the integrity constraints that must hold for them.
(5) A compositional definition is given as follows: An ontology consists of concepts, hierarchical

(*is-a*) organization of them, relations among them(in addition to *is-a* and *part-of*), axioms to formalize the definitions and relations.

3. WHY ONTOLOGY?

3.1 A short history of knowledge processing research

Expert systems have demonstrated the importance of a knowledge base which stores expertise of human experts in solving real-world problems. It was striking because it has succeeded to show the power of knowledge base technology in real-world problem solving which the other AI techniques have failed to realize. The key technology at that time was mainly the rule base technology by which expertise is encoded in the form of *if-then* rule to develop a large rule base. The concept of knowledge engineering, which is mainly utilization of rule base technology, has been proposed by Feigenbaum[Feigenbaum 77]

Although rule base technology contributed to the initial success of expert systems, people noticed the difficulty in the maintenance of a rule base and that in sharing and reusing the knowledge in the knowledge base so that all the knowledge bases had to build from scratch. Knowledge engineering has started to evolve from rule base technology to knowledge modeling since then to overcome these difficulties. Generic tasks[Chandra, 86], role-limiting method[Kahn 85], KADS methodology [Breuker 94], task ontology[Mizoguchi 92][Mizoguchi 95] are typical achievements. Characteristics common to them are focusing on modeling the problem solving structure independently of the domains to come up with a fundamental concept structure of each task such as diagnosis, design, planning/scheduling, etc. to model and explain real-world tasks, and hence to make it easier to build and maintain knowledge bases.

In order to better understand knowledge engineering, let us investigate AI research from another perspective. In AI research history, we can identify two types of research. One is "Form-oriented research" and the other is "Content-oriented research". The former investigates formal topics like logic, knowledge representation, search, etc. and the latter content of knowledge. Apparently, the former has dominated AI research to date. Recently, however, "Content-oriented research" has attracted considerable attention because a lot of real-world problems to solve such as knowledge systematization, knowledge sharing, facilitation of agent communication, media integration through understanding, large-scale knowledge bases, etc. require not only advanced formalisms but also sophisticated treatment of the content of knowledge before it is put into a formalism.

Formal theories such as predicate logic provide us with a powerful tool to guarantee sound reasoning and thinking. It even enables us to discuss the limit of our reasoning in a principled way. However, it cannot answer any of the questions such as how to systematize knowledge, what knowledge we should prepare for solving the problems given, how to scale up the knowledge bases, how to reuse/share the knowledge, how to manage knowledge and so on. In other words, we cannot say it has provided us with something valuable to solve real-world problems.

Feigenbaum's knowledge principle has been widely accepted and a lot of development has been carried out with a deep appreciation of the principle, since it makes the point in the sense that he stressed the importance of accumulation of knowledge rather than formal reasoning or logic. This has been proved by the success of the expert system development and a lot of research activities have been done under the flag of "knowledge engineering" which is apparently a key research of content-oriented research. However, the author is not claiming the so-called rule-base technology is what we need for future knowledge processing. Rather, treatment of knowledge should be in-depth analyzed further to make it sharable and reusable among computer and human agents. Advanced knowledge processing technology should cope with various knowledge sources and elicit, transform, organize, and translate knowledge to enable the agents to utilize it.

3.2 From knowledge engineering to ontology engineering

Although the importance of knowledge engineering as "Content-oriented research" has been gradually recognized these days and much progress has been attained, we do not have satisfactory methodologies yet. We still have problems in building an intelligent systems and in utilizing knowledge base technology with its full power. We could identify the reasons for this as follows:

1) Basic assumptions of a knowledge-based system are left implicit, which prohibits us from sharing and reusing knowledge
2) There exists no common knowledge base on top of which we can build another knowledge base.
3) There is no sophisticated way to *accumulate* knowledge.

The next generation knowledge processing technology is expected to solve these difficulties.

In addition to this requirement, it would be informative to mention the change of the surrounding social circumstances. In the expert system era, people tried to build a stand-alone problem solver using rule base technology. In the WWW era, however, requirements to information processing technology have rapidly changed from stand-alone problem solvers to powerful search engines for WWW, standard protocol and format for data exchange for e-business/e-commerce/EDI, knowledge management for strengthen corporations and semantic web movement, etc. The trends require different things, that is, what is required is knowledge which is objective, with high commonality and shallow in the sense of domain specificity. Such knowledge is opposite to what has been required for expert systems. This trend is summarized from the knowledge processing perspective as follows:

From AI to IA

where AI stands, of course, for artificial intelligence and IA for Intelligence Amplifier, Information Access or Intelligent Assistant. This clearly shows what IA requires is not a stand-alone problem solver which solves your problem for you but an *intelligent partner* who invisibly stays with you all the time and give you an effective help when necessary. Realization of such an intelligent partner, refinement and augmentation of the conventional knowledge engineering has to be done. In order to fulfill the goal Ontological Engineering has been proposed.

Ontological Engineering is a research methodology which gives us the design rationale of a knowledge base, kernel conceptualization of the world of interest, semantic constraints of concepts together with sophisticated theories and technologies enabling accumulation of knowledge which is dispensable for knowledge processing in the real world. Taking knowledge management as an example, it essentially needs content-oriented research. It should be more than information retrieval with powerful retrieval functions. We should go deeper to realize the true knowledge management.

An ontology, which is a system of fundamental concepts, that is, a system of background knowledge of any knowledge base, explicates the conceptualization of the target world and provides us with a solid foundation on which we can build sharable knowledge bases for wider usability than that of a conventional knowledge base. Knowledge engineering has thus developed into ontological engineering.

3.3 Semantic interoperability in semantic web[SW]

The above explanation is done from the knowledge base research perspective and might include a bit rigid interpretation of an ontology and ontological engineering. In the line of IA research, there is another context to understand what an ontology is. Semantic web which is the next-generation of web requires semantic interoperability among metadata associated with the web information. The main use of metadata is to make it easier for a search engine to find web pages. It consists of pairs of attribute and value to characterize web pages. The role of an ontology is to provide vocabulary for metadata description with computer-understandable semantics. The issue here is how to make the

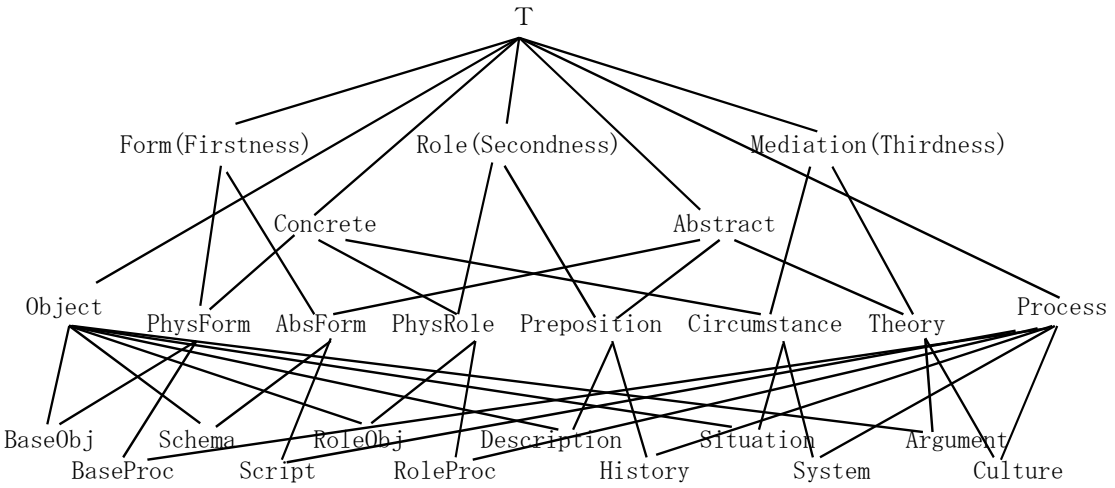


Fig. 3 Sowa's upper ontology[Sowa 95].

metadata interoperable. Because there would be no centralized control of metadata writing and ontology development for semantic web, it is highly possible that there appear a lot of metadata describing the same web page in different ways. This is the reason why we need semantic interoperability among metadata. An ontology is used to make it possible by ontology alignment which will be discussed in Part 2.

There are two large differences between the roles of an ontology for knowledge bases and those for metadata: One is philosophical and the other is practical. The philosophical one is that while an ontology for knowledge bases is a specification of the conceptualization of the target world, that for metadata is a set of computer-understandable vocabulary. The practical one is that an ontology for metadata does not have to consider the instance problem which is one of the most serious issues of an ontology for knowledge bases. In metadata cases, instances are usually very small portion of and independent pieces of a web document such as Mr. ABC:author, AI:topic, 15/04/03:date, etc. which are nothing to do with real models which might be treated seriously with strict consistency like those required for the knowledge base cases. This is the major reason why light-weight ontology, described in 4.3, is currently accepted in the semantic web community. Because the major use of metadata is intelligent search for web information, the utility of an ontology is to generalize the keywords in the query using an *is-a* hierarchy of concepts. Semantic interoperability does not require serious instance modeling neither, since it mainly needs identification of the correspondence between concepts included in the two different metadata under consideration.

4. TYPES OF ONTOLOGY

There are a few types of ontologies which have different roles. In some cases, discussion goes to a mess because of the ignorance of what type of ontology is under consideration. Some say "ontology is domain-specific like a knowledge base which was a failure". Others say "No, it isn't. An ontology is very generic and hence it is widely applicable and sharable". Both are correct because they are talking about different types of ontology which are the topic of this section.

4.1 Upper ontology

Philosophers have tackled what is being for about two thousands years. The portion of their work includes higher level categories which explain what exist in the world, which is called *upper ontology*. Aristotle's ten categories such as *matter, quantity, quality, relation, location, time*, etc. and C. S. Peirce's[Sowa 95] *firstness* which can be defined without assuming any other things like

human, iron, etc., the *secondness* which has to be defined depending necessarily on other things like mother, teacher, etc., and *thirdness* which provides an environment or context where the *secondness* works like motherhood, school, etc. are typical examples. Especially, the latter has been advocated by J. Sowa and has influenced AI researchers a lot.

Sowa introduced four important concepts, *continuant, occurrent, concrete and abstract* in addition to Peirce's three basic categories and obtains 12 top level categories by combining the seven primitive properties[Sowa 95](See Fig. 3). Guarino's upper ontology more extensively incorporates philosophical consideration. It is designed based on mereology(theory of parts), theory of identity, and theory of dependency. His ontology consists of two world: An ontology of Particulars such as things which exist in the world and Universals which include concepts we need when we describe Particulars[Guarino 97, 98].

Some of the practitioners show negative attitude to generic ontologies such as an upper ontology because they believe no use-independent ontology is useful. In the case of building an ontology for a large scale knowledge base, however, the validity of the knowledge base necessarily be justified in terms of its stability and applicability to wider rage of tasks, that is, it needs to show its generality rather than task-specific utility. Compliance of their ontologies with a principled upper ontology provides a good justification and an upper ontology provides useful guidelines on how to organize domain knowledge.

A standard upper ontology(SUO) has been being developed at IEEE P1600.1[SUO]. This is one of the most comprehensive ontologies among those in AI community. The working group has discussed what an identity is, how are 3D(3D space with time) and 4D(including time as the 4^{th} dimension) modeling are different/compatible, multiple ontologies or monolithic ontology, etc. The problem is that it is very hard to come to an agreement. Recently, SUMO(Suggested Upper Merged Ontology) [SUMO] and Cyc upper ontology[OpenCyc] have been proposed as a candidate of SUO. The committee is considering two possibilities: To have one SUO recommendation or to have a lattice of multiple upper ontologies.

4.2 Task ontology and Domain ontology

Roughly speaking, ontology consists of **task ontology**[Mizoguchi 95] which characterizes the computational architecture of a knowledge-based system which performs a task and **domain ontology** which characterizes knowledge of the domain where the task is performed. By a task, we mean a problem solving process like diagnosis, monitoring, scheduling, design, and so on. The idea of task ontology which serves as a system of the vocabulary/concepts used as building blocks for knowledge-based systems might provide us with an effective methodology and vocabulary for both analyzing and synthesizing knowledge-based systems.

Task ontology is useful for describing inherent problem solving structure of the existing tasks domain-independently. It is obtained by analyzing task structures of real world problems. Proposal of task ontology has been done in order to overcome the shortcomings of generic tasks[Chandra 86] while preserving their basic philosophies. It does not cover the control structure but do components or primitives of unit inferences taking place during performing tasks. The ultimate goal of task ontology research includes to provide a theory of all the vocabulary/concepts necessary for building a model of human problem solving processes.

The determination of the abstraction level of task ontology requires a close consideration on granularity and generality of the unit of problem solving action. These observations suggest task ontology consists of the following four kinds of concepts:

1. **Task roles** reflecting the roles played by the domain objects in the problem solving process
2. **Task actions** representing unit activities appearing in the problem solving process,

3. **States** of the objects, and
4. **Other** concepts specific to the task.

Task ontology for a scheduling task, for example, includes:

> **Task roles:**
> "Scheduling recipient", "Scheduling resource", "Due date", "Schedule", "Constraints", "Goal", "Priority", etc.
> **Task actions:**
> "Assign", "Classify", "Pick up", "Select", "Relax", "Neglect", etc.
> **States:**
> "Unassigned", "The last", "Idle" etc.
> **Others:**
> "Strong constraint", "Constraint satisfaction", "Constraint predicates", "Attribute", etc.
> Actions are defined as a set of procedures representing its operational meaning. So, they collectively serve as a set of reusable components for building a scheduling engine.

Before task ontology has been invented, people tended to understand an ontology is often use-dependent and it has the same shortcoming of the knowledge bases of expert systems, that is, little reusability because of its task-specificity. The idea of task ontology contributes to the resolution of such problems. The reason why an ontology looks task-specific is the ontology mixes up task ontology and domain ontology. Task ontology specifies the roles which are played by the domain objects. Therefore, if a domain ontology is designed after task ontology has been developed, one can succeed to come up with a domain ontology independent at least of the particular task because all the task-specific concepts are detached from the domain concepts to form task-specific roles in the task ontology. A task ontology thus helps develop a use-neutral domain ontology.

4.3 Heavy-weight ontology and light-weight ontology

Another viewpoint suggests us another type of ontology: Light-weight ontology and heavy-weight ontology. The former includes ontologies for web search engines like Yahoo ontology which consists of a topic hierarchy with little consideration of rigorous definition of a concept, principle of concept organization, distinction between word and concept, etc. The main purpose of such a hierarchy is to power up the search engine and hence it is very use-dependent. The latter is different. It includes ontologies developed with much attention paid to rigorous meaning of each concept, organizing principles developed in philosophy, semantically rigorous relations between concepts, etc. Instance models are usually built based on those ontologies to model a target world, which requires careful conceptualization of the world to guarantee of the consistency and fidelity of the model. Upper ontology is a typical ontology of heavy-weight ontology.

5. FURTHER EXPLANATION

5.1 FAQ

(1) How is an ontology different from a knowledge base?
Let me cite a phrase found in the email archive of ontology:

Date: Wed, 26 Feb 1997 12:49:09 -0800 (PST)
From: Adam Farquhar axf@HPP.Stanford.EDU
……

① Does it express the consensus knowledge of a community of people?
② Do people use it as a reference of precisely defined terms?
③ Does it express the consensus knowledge of a community of agents?
④ Is the language used expressive enough for people to say what they want to say?

⑤ Can it be reused for multiple problem solving episodes?
⑥ Is it stable?
⑦ Can it be used to solve a variety of different sorts of problems?
⑧ Can it be used as a starting point to construct multiple (sorts of) applications including: a new knowledge base, a database schema, an object-oriented program?

The stronger the 'yes' answer is to these questions, the more 'ontological' it is.

The above opinion is based on that there is no clear boundary between ontology and knowledge. It is a reasonable understanding when we think of Cyc[OpenCyc] whose upper part is definitely an ontology and the whole seems to be a knowledge base. The above opinion is somewhat misleading, though many of AI researchers accept it, since it does not try to capture an essential property of an ontology which is something related to concepts rather than vocabulary and is something related to what exists in the target world of interest. Another answer to the question is that we need to introduce a concept of relativity when we understand an ontology. I mean, a clear differentiation of an ontology from a knowledge base should come from its role, that is, an ontology gives you a system of concepts which underlie the knowledge base and hence an ontology can be a specification of the KB builder's conceptualization of the target world and is a *meta-thing* of a conventional knowledge base.

(2) How an ontology is different from the class hierarchy in the object-oriented(OO) paradigm?
They are similar. The developmental methodology of an ontology and that of an object hierarchy is also similar to each other in the upper stream. In the lower stream, however, the former concentrates on declarative aspects and the latter on performance-related aspects. Thus, the essential difference between the two lies in that the ontology research exploits declarative representation, while the OO paradigm is intrinsically procedural. In OO paradigm, the meaning of class, relations among classes, and methods are procedurally embedded and they are implicit. The ontology paradigm, on the other hand, descriptions are made declaratively in most cases to maintain formality and explicitness.

(3) What's new? How is it different from a taxonomy of concepts?
An ontology contains a taxonomy as its component. So, it partially implies a taxonomy. In general, a new term is rarely totally new. Rather, it is usually coined by extending existing terms. The term "ontology" is not an exceptional case. It is a new term and concept including existing concepts such as "taxonomy", "common vocabulary", "upper model", etc. by adding formality, richer relations, explicit representation of things usually left implicit.

(4) What is the computational semantics of an ontology? Is it just a set of labels?
This is one of the most crucial points of the roles an ontology plays. Contrary to that an ontology sometimes looks just a set of labels, it has deeper computational semantics. The author has proposed the following three levels of an ontology.

Level 1: A structured collection of terms. The most fundamental task in ontology development is articulation of the world of interest, that is, elicitation of concepts, then, organizing them in a hierarchy. These are indispensable to something to be an ontology. Typical examples of ontologies at this level include topic hierarchies found in internet search engines and tags used for metadata description. Little definitions of the concepts are made.

Level 2: In addition to that at the level 1 ontology, we can add formal definitions to prevent unexpected interpretation of the concepts and necessary relations and constraints also formally defined as a set of axioms. Relations are much richer than those at the level 1. Definitions are declarative and formal to enable computers to interpret. The interpretability of an ontology at this level enables computers to answer questions about the models built based on the ontology. Many of

the ontology building efforts aim at those at this level.

Level 3: The ontology at this level is executable in the sense that models built based on the ontology run using modules provided by some of the abstract codes associated with concepts in the ontology. Thus, it can answer questions about runtime performance of the models. Typical examples of this type are found in task ontologies[Mizoguchi 95][Breuker 94][Chandra 98].

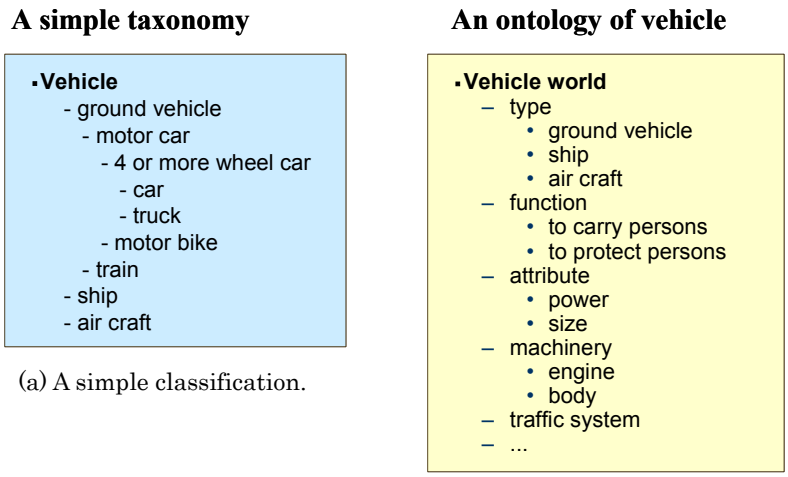
(5) Do you force people to accept your ontology?
An ontology should be shared by many people in nature. If it is not shared by a community, it loses its utility. However, this does not mean it is developed by one person who urges you to accept it. An ontology should be designed collaboratively with a happy agreement on its development in a community. It is neither true that there exists the only ontology for each domain nor that there exist as many ontologies as people in the community. The truth lies in between, hopefully, somewhere close to the former to enable an ontology to play its role.

One of the reasons for this concern is an ontology is understood to be dependent on the perspective which differs according to each developer. At first glance, this concern seems to make sense. However, it is also true that the fact that an ontology is something which reflects the fundamental conceptual structure of a target world cannot allow so many varieties. Many of the diversities come from ignorance of what an ontology is and of how to design an ontology. Another cause of the variety is that a domain ontology can be purpose-dependent. Because purposes are often dependent on task at hand, we can at least partially cope with such a case by employing the idea of task ontology which enables us to design a domain ontology independently of the problem solving tasks performed in the domain.
(6) Is it possible for you to come up with a stable and agreed ontology in this rapidly changing society?

What rapidly changing are not ontologies but models built by instantiating concepts defined in the ontology. Models include objects and relations between them as well as some rules which do change as time goes. An ontology, however, is a long-lasting fundamental conceptual structure on top of which knowledge bases are built. Especially, an upper ontology rarely changes.

5.2 What is not an ontology

If an ontology were nothing other than a concept hierarchy, then ontological engineering would not deserve to discuss seriously. Of course, all conceptual hierarchies are not an ontology. In order to characterize ontological engineering properly, let us discuss what is not an ontology, especially what is not a heavy ontology.



(a) A simple classification.

(b) Vehicle world ontology.

Fig. 4 Vehicle ontology.

(a) An ontology is not just a set of terms

While an ontology provides us with a common vocabulary, a vocabulary, that is, a set of terms, itself cannot be said to be an ontology as it is. An ontology needs to consists of an *is-a* hierarchy of concepts. This is partly because it reveals the proper classification of concepts to show inherent structure of the target world. The reason why an *is-a* hierarchy of concepts is indispensable is discussed later. Furthermore, there should be a clear distinction between term/word and concepts. The former are "names" of the latter and an ontology is a theory of concepts rather than terms/word. It does not care about how the concept is called. It puts an appropriate name on a concept for making it human understandable. This suggests the synonym is not an ontological issue.

(b) Heavy-weight ontology is not just a simple hierarchy of concepts

A heavy-weight ontology includes a taxonomy of concepts, but not all taxonomies are a heavy ontology. Let us take an example. Fig. 4(a) shows a simple classification of vehicles which includes ground vehicle, car, motor bike, ship and air craft, etc. It is nothing special. It is not enough to explain or to understand what a vehicle is. In order to clearly understand what a vehicle is, you need to know more concepts such as what function it has, what attribute it has, what machinery it has, how it works in what social environment, etc. Without these concepts, you cannot build a vehicle world. Fig. 4(b) shows a hierarchy of such concepts in which the former hierarchy is positioned one of the categories as *type of vehicle*. To make such a hierarchically organized concepts a heavy-weight ontology, axiomatic definitions of concepts and relations are necessary.

(c) An ontology is not a knowledge representation

An ontology is neither a semantic network nor a frame. Semantic network and frame are a formalism of the way of knowledge representation. Although an ontology inherently includes *is-a* hierarchy of concepts, it does not matter whether it is represented by semantic network or frame. In other words, whether a certain thing is an ontology or not is independent of how it is represented. An ontology provides us with a guideline for modeling the world. To do this, it consists of carefully chosen top-level categories which are reliable enough to explain lower concepts. The ontology problem is thus totally a content issue.

5.3 What are the roles of an ontology?

This question is also very important. The following is an enumeration of the merits we can enjoy from an ontology:

(a) A common vocabulary.
   The description of the target world needs a vocabulary agreed by people involved. The fundamental role of an ontology is to provide a common vocabulary.

(b) Data structure
   An ontology in a database is the conceptual schema. In this sense, an ontology provides us with a data structure appropriate for information description and exchange.

(c) Explication of what is left implicit.
   In all of the human activities, we find presuppositions/assumptions which are left implicit. Typical examples include definitions of common and basic terms, relations and constraints among them, and viewpoints for interpreting the phenomena and target structure common to the tasks they are usually engaged in. Any knowledge base built is based on a conceptualization possessed by the builder and is usually implicit. An ontology is an explication of such implicit knowledge. An explicit representation of such assumptions and conceptualization is more than a simple explication. Although it might be hard to be properly appreciated by people who have no experience in such representation, its contribution to knowledge reuse and sharing is more than expectation considering that the implicitness has been one of the crucial causes of preventing knowledge sharing and reuse.

(d) Semantic interoperability

Metadata used in semantic web is built on the basis of an ontology which constrains and partially defines the meaning of each tags and values. Interpretation and translation of the metadata can be done via ontologies. Ontologies thus play the role of glue which guarantees semantic interoperability among metadata.

(e) Explication of design rationale
An ontology contributes to explication of assumptions, implicit preconditions required by the problems to solve as well as the conceptualization of the target object which reflects those assumptions. In diagnostic systems, for instance, fault classes diagnosed and range of the diagnostic inference are typical examples.

(f) Systematization of knowledge
Knowledge systematization requires well-established vocabulary/concepts in terms of which people describe phenomena, theories and target things under consideration. An ontology thus contributes to providing backbone of systematization of knowledge.

(g) Meta-model function
A model is usually built in the computer as an abstraction of the real-world target. And, an ontology provides us with concepts and relations among them which are used as building blocks of the model. Thus, an ontology specifies the models to build by giving guidelines and constraints which should be satisfied.

(h) Theory of content
In summary, an ontology provides us with "a theory of content" to enable research results to accumulate like form-oriented research avoiding ad-hoc methodologies which the conventional content-centered activities have been suffering from.

6. Concluding remarks

We have discussed ontology as an introduction to ontological engineering. Before concluding this article, let us discuss a bit more fundamental issues of ontology:

(1) What is an *is-a* hierarchy
(2) How *is-a* and *part-of* relations are different from each other.

A more fundamental issue than the above two is why we think the two relations, *is-a* and *part-of,* are important for designing an ontology. To investigate things, one of the fundamental conceptual tools is *counting* or *identity*. That is, it is crucial to know how many things are there. This gives us two principles: one is the proper categorization of things and the other is the proper way of concept organization. The former suggests us, say, blue thing, is inappropriate for a category, since it does not allow us to properly count how many blue things are there. You can understand it if you are given a blue table. A blue table has many blue things other than itself. The latter is as follows. In counting, we also want to have many ways of counting in terms of specific to generic concepts. For example, "How many tables are there in this room?", "How many lights are there?" and "How many furniture are there" This requires us to count a thing exactly once and number of things included in the parent concept is equal to the summation of those of its children concepts, and hence a class hierarchy has to be strictly in a tree structure with the condition that the set of members of a child class has to be a subset of the member set of its unique parent, which is a common property of an *is-a* hierarchy. In the ontology research, we add another constraint for the sake of *identity*: An *is-a* hierarchy should represent inheritance of essential property of each concept rather than ad-hoc property. The importance of this additional constraint is discussed in (1) below.

Things are composed of its parts. Some are essential to the whole, the thing itself. The concept of *whole* is crucial to understand things properly, since it is also related to counting or identity of things. Why a thing is recognized as the whole not as a different whole with less or more parts. This is a mysterious issue of philosophical ontology. Although we computer scientists do not have to investigate such a deep issue, parts of a thing is really important to properly capture a concept. This is why *part-of* relation is considered as important as well as *is-a* relation.

(1) As mentioned above, an *is-a* hierarchy is not a simple classification of concepts. Among possible classifications, it should be one formed by concentrating on the inherent property of each concept. One of the most important utility of designing an ontology of a domain is that intermediate concepts(classes) in an *is-a* hierarchy reveals the intrinsic structure of the target world which cannot be attained by a flat classification. To realize this, the *is-a* hierarchy should not be an ad-hoc classification, instead, it has to realize an inheritance of an *essential* property of the higher concepts down to the leaf concepts. This assures that intermediate concepts in an *is-a* hierarchy are meaningful and help people in-depth understand the target world. Ad-hoc classifications (taxonomies) are usually purpose-dependent, which is why they are not appropriate for a component of an ontology which has to be sharable by many people and stable as a long-lasting backbone of the knowledge structure. As mentioned earlier, however, a purpose-independent ontology seems weak because it reminds us that general theories of AI are not powerful enough to cope with the real-world problem. While such an observation makes sense at first glance, it turns out that an ontology is not a knowledge base for problem solving but one for a foundation of knowledge bases for various purposes which has to be built on top of it. This justifies that an ontology as a generic and fundamental system of concepts is of value.

(2) How *is-a* and *part-of* relations are different.
At first glance, *is-a* and *part-of* relations look very different from each other. However, one often comes across a difficulty when he/she tries to use one of them to model a target world. Let us take the following examples:

(a)<dog *is-a* mammal >
(b)<dog *is-a* mammal living in Japan>
(c)<dog *is-a* mammal living on the earth>

In the context of plant operation:

(d)<normal operation *is-a* operation>, <restoration operation *is-a* operation>
(e)<normal operation *part-of* operation>, <restoration operation *part-of* operation>

While there would be no problem with (a), (b) might be problematic because “mammal living in Japan” looks representing *a species* rather than a class and hence one tends to think

<“dog species” *part-of* “mammal species” living in Japan>.

However, in spite of (c) is essentially the same as (a), it also suggests us to use *part-of* relation. We need a theory to resolve this issue.

Concerning the last two examples, (d) looks absolutely correct. However, when you are given (e), you might be convinced by it as well, since operation of any plant is composed of normal operation and restoration operation. This issue is of a different type from that of the former, though it has the same characteristic. This is very problematic because it is impossible to hold *is-a* and *part-of* relations between the same concepts at the same time. A short explanation of a solution to this issue is the following. “Operation” in (e) is an abstraction of *event* or *process* which has time interval to specify from when to when the operation has been performed. On the other hand, “Operation” in (d) is an abstraction of an *action* which has no explicit specification when it occurs. Therefore, there is no conflict between (d) and (e).

What we have discussed in this article includes only basic topics of an ontology and many important topics such as ontology building methodology, ontology representation languages, ontologies built, applications of ontology, more fundamental issues which are necessary to in-depth understanding of ontological engineering, etc. All these will be discussed in the subsequent articles.

Reference

[Breuker94] J.Breuker and W.V.de Velde: The Common KADS Library for Expertise Modelling,

IOS Press, Amsterdam, 1994.
[Chandra, 86] Chandrasekaran, B.: Generic tasks in knowledge-based reasoning: High-level building blocks for expert system design, IEEE Expert, 1, No.3, pp.23-30, 1986.
[Chandra 98] B. Chandrasekaran, J. R. Josephson and Richard Benjamins, Ontology of tasks and methods, ECAI98 Workshop Notes on Applications of Ontologies and Problem-Solving Methods, 1998.
[Feigenbaum 77] FEIGENBAUM, E.A. (1977) "The art of artificial intelligence – Themes and case studies of knowledge engineering" Proc. of 5th IJCAI, pp.1014-1029, 1977.
[Genesereth, 87] Genesereth, M. and Nilsson, N. (1987) Foundation of Artificial Intelligence, 1987.
[Gruver] Gruber, T. http://www-ksl.stanford.edu/kst/ what-is-an-ontology.html.
[Guarino 97] Guarino N.,Some Organizing Principles for a Unified Top-Level Ontology. Revised version of a paper appeared at AAAI 1997 Spring Symposium on Ontological Engineering (LADSEB-CNR Int. Rep. 02/97)
[Guarino 98] Guarino, Nicola: 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.
[Kahn 85] Kahn, G. and J. McDermott: MORE: An intelligent knowledge acquisition tool, Proc. of IJCAI-85, 1, pp.581-584, 1985.

[OpenCyc] http://www.opencyc.org/
[Mizoguchi 92] Mizoguchi, R. et al.: Proc. of JKAW92.
[Mizoguchi 95] Mizoguchi, R. et al. (1995) Task ontology for reuse of problem solving knowledge, Proc. KB&KS95, Enshede, The Netherland.
Smith, B. & Welty, C. (2001) Proc. of the Second International Conference on Formal ontology and Information Systems: FOIS2001, ACM ISBN 1-58113-377-4 ACM Press
[Sowa 95] Sowa, J. (1995) Distinction, combination, and constraints, Proc. of IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing.
[SUMO] http://ontology.teknowledge.com/
[SUO] Standard Upper Ontology, http://suo.ieee.org/
[SW] http://www.semanticweb.org/