Available online at www.sciencedirect.com
SCIENCE DIRECT
Computer-Aided Design 38 (2006) 713–725
COMPUTER-AIDED DESIGN
www.elsevier.com/locate/cad

Meta-ontology for automated information integration of parts libraries

Joonmyun Cho[a,*], Soonhung Han[a], Hyun Kim[b]

[a] *Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology (KAIST), 373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, South Korea*

[b] *Software Robot Research Team, Electronics and Telecommunications Research Institute (ETRI), 161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350, South Korea*

Received 20 May 2005; accepted 18 March 2006

Abstract

Seamless integration of digital parts libraries or electronic parts catalogs for e-procurement is impeded by semantic heterogeneity. The utilization of ontologies as metadata descriptions of the information sources is a possible approach to providing an integrated view of multiple parts libraries. However, in order to integrate ontologies, the mismatches between them should be resolved. In this paper, we propose meta-concepts with which the ontology developers describe the domain concepts of parts libraries. The meta-concepts have explicit ontological semantics, so that they help to identify domain concepts consistently and structure them systematically. Consequently, our method ensures that the mismatches between parts library ontologies are confined to manageable mismatches which a software program can resolve automatically. Modeling ontologies of real mold and die parts libraries is taken as an example task to show how to use the meta-concepts. We also demonstrate how easily a computer system can merge the resultant well-established ontologies.

Keywords: Parts library; Electronic parts catalog; Ontology-based information integration; Knowledge modeling; B2B e-procurement

1. Introduction

Automated integration of parts information in B2B e-procurements is a complex task. The following subsections give an overview about the problem domain and our considerations about the problem and solution.

1.1. The problem domain

Generally, a product is designed and manufactured using ready-made components or parts from multiple suppliers. So, the product manufacturers want to locate suppliers easily and effectively evaluate the complementary parts.

A number of recent developments in information technology have opened up a vast potential for new electronic forms of procurement. In particular, developments in information technology provide hitherto unknown opportunities for buyers. They have the possibility for searching for up-to-date parts all over the world and for picking the most favorable offer. Electronic parts catalogs or digital parts libraries are basic means for achieving this. They are the reference for part selection and supplier selection, describing the parts information which corresponds to an expertise on the criteria for selecting a part and on the condition of part usage [1–4]. However, in many cases the potential of the digital parts library remains unharnessed. The worldwide search for parts is impeded by the heterogeneity of the parts information descriptions and different search strategies required by the parts libraries [3]. Thus, the buyer has to access different parts libraries from multiple suppliers and navigate through their different search procedures. At the end, he or she has to transform the parts information found into his/her own language and compare them manually to pick the best part.

In order to utilize the potential of parts libraries, an intermediary which is capable of providing an integrated view of them is needed [4]. Providing an integrated view of multiple information sources can be achieved by metadata integration and interoperation [2,5]. The metadata of parts is usually described as follows. Parts are gathered in parts categories that are represented by classes. This set of classes is organized according to a simple hierarchy (to which inheritance applies) of classes. Each class is associated with a set of technical

* Corresponding author. Tel.: +82 42 860 1873; fax: +82 42 860 6790.
E-mail address: jmcho@etri.re.kr (J. Cho).

0010-4485/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2006.03.002

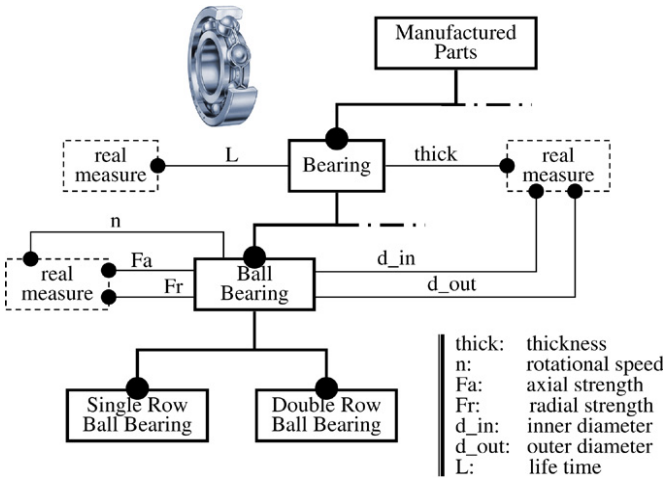

Fig. 1. A simple data model of a parts library for bearings [2].

properties. Figure 1 from [2] shows such knowledge description of parts for mechanical ball bearings. The graphical notation used is EXPRESS-G [6] where labeled solid boxes represent parts classes, dashed boxes represent data types, thick lines represent inheritance, and labeled solid lines ending with a circle (that emphasizes the relationship direction) represent properties.

Recently, several researchers recommended utilizing ontologies as the metadata descriptions of information sources [5,7]. Because ontologies are explicit and formal specifications of the knowledge, especially implicit or hidden knowledge, of information sources they help us with part of the integration problem by disambiguating information items. However, ontologies themselves could be heterogeneous. It is possible that ontologies would describe the knowledge of similar information sources in different ways. For example, one supplier may classify *Ball Bearing* parts into *Single Row Ball Bearing* class and *Double Row Ball Bearing* class as depicted in Fig. 1 according to the type of raceway for balls, whereas the other may classify them into *Small Bore Ball Bearing* class and *Large Bore Ball Bearing* class according to the size of the bore diameter. These differences, known as 'mismatches', are obstacles to integrating and interoperating independently developed ontologies.

1.2. Ontology mismatches

The ontology mismatches can be subdivided into two categories, *conceptualization mismatch* and *explication mismatch*, according to phases of ontology building [8,9]. The creation of an ontology involves two sub-processes; conceptualizing a domain, and explicating the conceptualization.

During the conceptualization process decisions are made about ontology concepts (classes and attributes), instances, relations, functions and axioms that are distinguished in the domain. Usually, the process also involves ordering the classes in a hierarchical fashion and assigning attributes to them. Thus, the *conceptualization mismatch* includes such types as (1) *categorization mismatch*, (2) *aggregation-level mismatch*, (3) *structure mismatch*, (4) *attribute-assignment mismatch*, and (5) *attribute-type mismatch* [8]. A categorization mismatch occurs when two conceptualizations distinguish the same ontology concept but divide this concept into different subconcepts. For example, this occurs when one conceptualization structures its knowledge about animals around classes *mammals* and *birds*, whereas the second structures around classes *carnivores* and *herbivores*. An aggregation-level mismatch occurs if two conceptualizations both recognize the existence of an ontology concept, but define concepts at different levels of abstraction. For example, it occurs when one conceptualization distinguishes the *persons* class, whereas the second distinguishes *males* and *females* but does not have *persons* as their superclass. A structure mismatch occurs when two conceptualizations distinguish the same set of ontology concepts but differ in the way these concepts are structured by means of relations. An attribute-assignment mismatch occurs when two conceptualizations differ in the way they assign an attribute to classes. An attribute-type mismatch occurs when two conceptualizations distinguish the same attribute but differ in their assumed instantiations.

In the end, the conceptualization of a domain is explicated into an ontology. Explicating the conceptualization requires an ontology language [10–12]. If we confine ourselves to a generic form of logical expressions instead of a specific ontology language, we can consider the definitions (which constitute the ontologies) to be composed of a *definiendum*, a *definiens*, and a *concept* of the domain. (In the following, the word 'term' is used instead of *definiendum* for clarity.) Thus, the mismatches pertaining to explication occur when two ontologies have different definitions where their terms, their definiens, or their ontological concepts are identical. The *explication mismatch* includes such types as (1) *concept and term mismatch*, (2) *concept and definiens mismatch*, (3) *concept mismatch*, (4) *term and definiens mismatch*, (5) *term mismatch*, and (6) *definiens mismatch* [8]. For example, a concept and term mismatch occurs when one ontology contains the definition (in PROLOG format) `vessel(x)` ← `seagoing(x)` ∧ `large(x)` (to define the concept of a vessel) and the other ontology contains the definition `whale(x)` ← `seagoing(x)` ∧ `large(x)` (to define the concept of a whale). In this case, the two ontologies use the same definiens (i.e., `seagoing(x)` ∧ `large(x)`) but differ in both the concept they define and the term (i.e., `vessel` and `whale`) linked to the definiens.

1.3. Knowledge systematization

We should note one point from the discussion on the ontology mismatches. The point is that all the conceptualization mismatches must be present in some form in the explication (i.e., ontology) [8]. For example, the categorization mismatch occurs in the explication as a concept and definiens mismatch or definiens mismatch. This means that the origin of the ontology mismatches is the differences in the way the given domain is *interpreted* (*conceptualized*), rather than the differences in the way the conceptualization is *specified* (*explicated*) [9]. It is the differences in the way the given domain is interpreted that cause different ontology concepts and different relations. The explication mismatches are defined on *form*, rather than on *content*.

Returning to the parts library domain, it is easily postulated that, in order to integrate automatically heterogeneous parts libraries, the knowledge of each parts library must be interpreted in a consistent way, so that similar concepts are distinguished and they are structured with similar relations. Otherwise, the mismatches among the parts library ontologies become hard to resolve by a computer system. For example, in the case where one ontology classifies *Ball Bearing* parts into such sub-categories as *Small Bore Ball Bearing* category and *Large Bore Ball Bearing* category, the categories of other ontologies that are similar to the two sub-categories cannot be easily determined. This is because the classification criterion, i.e. 'the size of the bore diameter', is vague, so that it could be applied differently or would not be applied at all in other ontologies.

Using an ontology language alone is not enough to consistently conceptualize a given domain. For example, when describing the knowledge "a red ball exists", with a KL-ONE-like language, a good choice may be to consider ball as a *class* and red as a filler of color *role* such that $\texttt{ball} \subseteq \exists \texttt{color.red}$. However, nothing prevents another modeler from adopting a different choice: for instance, both ball and red may be considered as classes, with no role at all.

In AI/KR and database communities, "form-oriented researches" have been dominant as basic research to date. In other words, researches concerning *vessels* have been done paying little consideration to the *contents*. Research on "theory of contents", that is, research on "what and how to put it in the vessel" has rarely been done. Although knowledge representation formalisms such as predicate logic tell us how to *represent* knowledge, they cannot tell us how to *prepare* the knowledge [13–15]. Although the model-theoretic semantics of a knowledge representation formalism provides us with a powerful tool for sound reasoning by relating a model to mathematical structure (i.e., abstract set-theoretic structure), it offers no help in making the connection between the model and the real world [16].

What is necessary is something we need before the stage of *knowledge representation*, that is, *knowledge systematization*: the way that we consistently distinguish the domain concepts and that we systematically structure them. The knowledge systematization should be principled in the sense that it can answer such questions as "what are *classes*?", "what are *roles*?", and "how should they be related?". Explicit conceptualization of ontological assumptions on concepts can contribute to this by providing well-established what might be termed "human-perceived meaning" to the meta-concepts such as *class* and *role* in terms of which people describe phenomena in a given domain. It is just as formal theory provides model-theoretic semantics to the ontology language constructs.

This paper is organized as follows. The next section reviews related works and compares our work with them. Section 3 presents (meta-)ontologies of meta-concepts which have explicit ontological semantics. Section 3.1 introduces Guarino's upper ontology as a foundational meta-ontology. Section 3.2 discusses the main meta-ontology, *Parts Library Concept Ontology*, for our knowledge systematization of parts libraries. Section 4 discusses application of the *Parts Library Concept Ontology* to modeling ontologies of real mold and die parts libraries for B2B e-commerce. This section also discusses how easily a computer system can merge the well-established ontologies and provide an integrated interface.

2. Related works

Various ontology-based information integration methods have been developed. There are different ways to employ the ontologies, to represent knowledge, and to generate inter-ontology mapping [7]. For ways of ontology employment, most methods including our method follow the hybrid approach. In the single ontology approach each information source is related to the same global domain ontology. As a result, a new information source cannot bring in any new or specific concept without requiring change in the global ontology. In the multiple ontologies approach, each information source has its own ontology developed irrespective of other information sources. In this case the inter-ontology mapping is very difficult to define as the different ontologies may use different aggregations and granularities of the ontology concepts. In the hybrid approach, to overcome the drawback of single or multiple ontology approaches, each information source has its own ontology and all source ontologies are connected by some means to a common shared ontology.

This section reviews three representative researches using a hybrid approach particularly in terms of knowledge modeling and inter-ontology mapping.

COIN [10] uses its own language, COINL (COIN Language) as a knowledge representation formalism. COINL was developed directly from Frame Logic [17]. Thus, the ontological semantics, i.e. human-perceived meaning of language construct is general or content-independent [14,15]. The language alone is not enough for identifying domain concepts consistently and structuring them systematically; the ontology modeler must decide what is a concept (i.e., class) and what is a slot (i.e., attribute). As a result, ontologies have different conceptualizations, and mismatches may occur between arbitrary expressions. For example, mappings between class expressions and attribute expressions or even between a class expression and a compound expression that specifies some instances of several classes may be needed. In the COIN method, the mappings are manually specified and limited to data value conversion between class attributes, which are the instances of common types of a shared ontology, because mappings between arbitrary expressions are hard to handle.

BUSTER [11] uses the general-purpose Web ontology language OIL [18] which is developed based on Description Logic [19], so that it also has the drawbacks of the general or content-independent language. The BUSTER method overcomes the drawbacks by relying heavily on a shared ontology. In the BUSTER method, the shared ontology is a common global vocabulary that defines in advance all the terms (i.e., concepts) which can be used as class attributes. Source ontologies are built by defining domain classes that select and restrict only the concepts defined in the shared ontology as

their attributes, so that the shared ontology itself guarantees the automated integration and interoperation of source ontologies. The semantic correspondence between the attributes can be determined easily and, based on such attribute correspondence, classes of different ontologies can be mapped to each other by automated subsumption reasoning. However, the BUSTER method has a high cost for developing the shared ontology because domain experts must find all the necessary concepts and deeply consider the usage conditions in advance.

The PLIB approach [20], unlike COIN and BUSTER, does not use a logic-based knowledge representation language. Instead, it defines a proprietary data meta-model and an ontology developer represents domain knowledge with only the data meta-model. However, because the data meta-model defines only a small number of primitives i.e., entity types, the domain structure is too coarse for an ontology developer to model the domain knowledge in a consistent way. Moreover, the primitives have subjective meanings agreed upon by only the developers of the meta-model, independently of an explicit account of the underlying ontological assumptions. The problems of deciding what is a class, what is an attribute, etc. still remain as an unguided task. So, mismatches occur among arbitrary concepts and inter-ontology mapping is complex. In the PLIB approach, the mappings are manually specified as with the COIN method.

Our method uses meta-concepts which have explicit ontological semantics. The explicit ontological semantics help ontology developers to identify parts library concepts consistently and systematically structure them by reducing the conceptualization differences. Although our method cannot remove all the possible mismatches the remaining mismatches, referring to the results of existing research on automated ontology merging, are small enough to enable a computer system to resolve automatically. The existing research showed a good result of automation by using similarity measuring techniques such as syntactic analysis, lexical analysis, structural analysis, and extensional analysis [21]. Those techniques assume that the ontologies to be merged have similar form and structure so that the target ontology elements the similarity of which is to be assessed can be legitimately determined. Our method complements this with consistent structure of ontologies. In our method, mismatches emerge only among the concepts that belong to the same meta-concept and have the same parent concepts.

3. Meta-ontology framework for knowledge systematization

The goal of this paper is to provide a system of meta-concepts in terms of which knowledge modelers consistently distinguish domain concepts of parts libraries and systematically structure them. The system is organized into layered ontologies. Fig. 2 shows the hierarchical organization of ontologies. Ontologies are arranged upside-down to stress the analogy of physical construction.

An ontology in a layer plays the role of a meta-ontology of those in an upper layer. That is, an ontology gives a

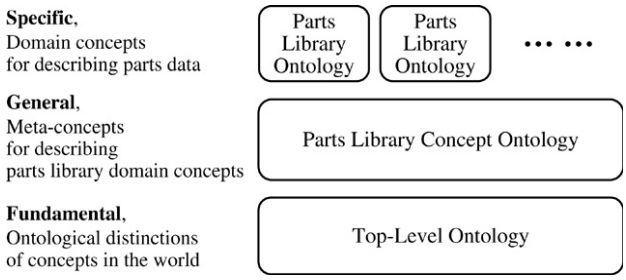

Fig. 2. Hierarchical organization of ontologies.

controlled vocabulary and structure for modeling consistently and systematically ontologies of the upper layers. The *Top-Level Ontology* stays at the bottom because it governs all ontologies in the upper layers with the most fundamental meta-concepts. Next is the *Parts Library Concept Ontology* which is the basis of our knowledge systematization for parts libraries because it provides effective meta-concepts for modeling domain ontologies i.e., the *Parts Library Ontologies* of the figure.

3.1. The Top-Level Ontology

Although the major objective of this paper is to establish the *Parts Library Concept Ontology* for parts knowledge systematization, we cannot avoid mentioning another ontology which provides foundations for it. It is one of the so-called upper ontologies or theories of ontology. Upper ontologies give fundamental distinctions of concepts in the world and theories of them based on formal consideration of such meta-questions as "what is a proper taxonomy?", "what is a class/attribute?", "what are *is-a* and *part-of* relations?", "how are an object and a process different?", "what is identity?", "how is identity inherited?", etc. [13–15]. They help us to understand systematically the ontological characteristics of domain concepts.

We use Guarino's upper ontology [22] as the *Top-Level Ontology* of our system of meta-ontologies. Guarino classified concepts into the ontological distinctions such as *TYPE*, *QUASI-TYPE*, *MATERIAL ROLE*, *PHASED SORTAL*, *CATEGORY*, *ATTRIBUTION* and explicitly characterized them with combination of the ontological natures such as *rigidity*, *identity*, and *dependence*. Table 1 summarizes those ontological distinctions and their ontological natures.

Identity is related to the notion of whether or not a concept provides an identity condition (IC) [22]. The ICs enable identification of an entity (or object) as an instance of a concept, re-identification of the instance in all possible forms, and to count it individually. For example, the *person* concept provides an IC such as *fingerprint*. It is necessary to distinguish concepts which supply ICs (in Table 1, denoted by $+O$) from concepts which simply carry ICs (denoted by $+I$). Concepts such as *student* only carry ICs such as *fingerprint* which are inherited from their parent concept *person*. An IC allegedly provided by the *student* such as *registration-number* is a local IC which has unambiguous meaning within a limited context and situation, whereas *fingerprint* is a global IC which does not change across time or context.

Table 1
Top-level ontological distinctions and their ontological natures [22]

Ontological distinctions	Example	Ontological natures			
		Identity		Rigidity (R)	Dependence (D)
		Supplying IC (O)	Carrying IC (I)		
CATEGORY	*concrete entity*, *abstract entity*	−	−	+	+ / −
QUASI-TYPE	*invertebrate animal*, *herbivore*	−	+	+	+ / −
TYPE	*person*, *cat*	+	+	+	+ / −
PHASED SORTAL	*caterpillar*, *butterfly*	−	+	∼	−
MATERIAL ROLE	*student*, *food*	−	+	∼	+
ATTRIBUTION	values of qualities like *color*, *shape*	−	−	∼ / ¬	− / + / −

Rigidity is related to the notion of whether a concept is essential to all the instances of it. The essentiality is strictly related to the notion of necessity [22], so that, for example, a rigid property is a property that is necessary for all instances that are true in every possible world. We normally think of *person* as rigid (denoted by $+R$) because, if x is an instance of *person*, it must be an instance of *person* in every possible world. The *student* concept, on the other hand, is normally not rigid because we can imagine an entity moving in and out of *student* while being the same individual. Concepts that are not rigid can be anti-rigid or semi-rigid. Anti-rigid concepts (denoted by $\sim R$) are those such as *student* and *food*, all the instances of which are true at least in one possible world but false in a different possible world. Semi-rigid concepts (denoted by $\neg R$) are concepts that are neither rigid nor anti-rigid. They can be true or false even in one possible world.

Dependence is related to the notion of whether or not the instances of a concept require the instances of other concepts in order to exist [22]. For instance, to be a student, a student needs a school.

Concepts belonging to *CATEGORY* are characterized as being rigid but neither supplying nor carrying ICs. They divide the domain into useful segments. Since they cannot have definite membership conditions like ICs, they are normally the highest level concepts in an ontology. *QUASI-TYPE* concepts are rigid and carry ICs. They often represent high level organizations by grouping entities based on combinations of properties that do not affect the ICs of entities. *TYPE* concepts are rigid and supply the global ICs. They play an important organizational role in a taxonomy (subsumption hierarchy). Assuming that each *TYPE* concept has a distinct set of global ICs, a taxonomy of them is always a tree. When a *TYPE* concept specializes another *TYPE* concept, it adds further global ICs to those inherited from the subsuming concept. *PHASED SORTAL* concepts are anti-rigid and independent. They do not supply global ICs but they supply local ICs. They account for entities which naturally change some of their identity criteria over discrete phases. For example, an entity may at one time be a *caterpillar* and at another time be a *butterfly*. *PHASED SORTAL* concepts must be subsumed by a *TYPE* concept to inherit the global ICs, because it must be possible to determine that they are the same entity at these two phases. *MATERIAL ROLE* concepts are anti-rigid and always dependent. They represent roles that are constrained to concepts. *ATTRIBUTION* concepts are either semi-rigid, or anti-rigid and independent. We assume they are anti-rigid and independent. They represent values of qualities of a concept itself such as color and shape.

3.2. The parts library concept ontology

In order to describe domain knowledge we need meta-concepts that play the role of a vocabulary. For example, in order to describe parts knowledge we need the "parts class" meta-concept and "parts property" meta-concept just as in Fig. 1. For consistent knowledge systematization, the meta-concepts should have a well-established human-perceived meaning to inform the knowledge modeler what exactly is the parts class and what are the appropriate parts properties.

To achieve this, we introduce seven meta-concepts and relate them to the ontological distinctions of Guarino's upper ontology. They inherit the ontological semantics from the related ontological distinctions. We call the system of meta-concepts *Parts Library Concept Ontology* and we call the meta-concept the *knowledge modeling primitive*. Fig. 3 shows the knowledge modeling primitives and their structure. The figure follows the UML notation rule. The domain concepts that are modeled using the *PARTS FAMILY CATEGORY*, *PARTS FAMILY*, and *PARTS MODEL* primitives constitute a main taxonomy. The *PARTS FAMILY* and *PARTS MODEL* concepts are composed of the *ATTRIBUTE* and *META-ATTRIBUTE* concepts. The *ATTRIBUTE* concepts of a subsuming concept are inherited by the subsumed concepts through the subsumption relation.

To model the domain concepts that represent classes of parts, we introduce the knowledge modeling primitive, *PARTS FAMILY*, and relate it to the *TYPE* ontological distinction. Therefore, the *PARTS FAMILY* concept should be rigid and able

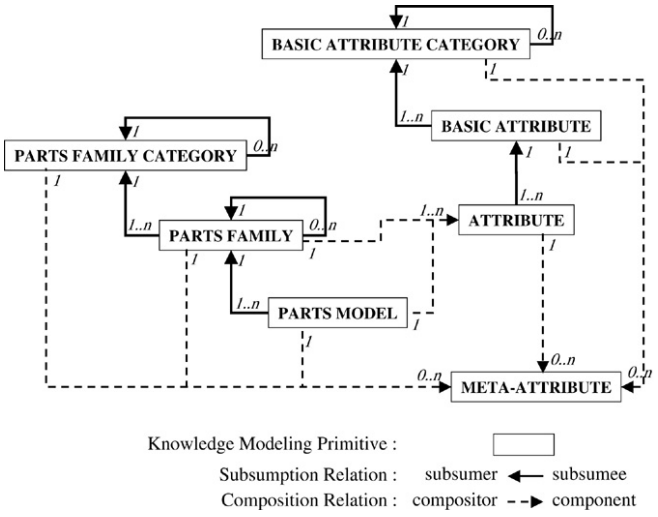

Fig. 3. Knowledge modeling primitives and their structure.

to supply its own global ICs. In other words, the meaning of the *PARTS FAMILY* concept must not change across suppliers and time. It should also be able to define properties to identify entities as instances of it and to distinguish them individually (see below for defining such properties). This ontological semantics of *PARTS FAMILY* coincides well with the human-perceived meaning in the sense that in order to pick a part from different parts libraries a buyer first finds similar parts classes to search not by their names but by their classification criteria and properties. Such ontological semantics of the *PARTS FAMILY* primitive help a knowledge modeler to consistently identify appropriate parts classes and to structure them systematically.

According to Guarino's theory of upper ontology, a *TYPE* concept supplies a global IC that is defined as a relation Γ satisfying the following formula: $\Phi(x) \wedge \Phi(y) \wedge x = y \leftrightarrow \Gamma(x, y)$, where Φ denotes a *TYPE* concept [22]. Thus, a *PARTS FAMILY* concept, by some means, should supply the global IC because it is a *TYPE* concept. For the case of a parts library, the identifying relation Γ of a *PARTS FAMILY* concept can be formulated by using technical properties [23]. For example, the identifying relation of the *Ball Bearing* concept can be formulated as follows:

$$\begin{aligned}\Gamma(x, y) \equiv\ & \textit{axial strength}(x) = \textit{axial strength}(y)\ \textit{and}\\ & \textit{radial strength}(x) = \textit{radial strength}(y)\ \textit{and}\\ & \textit{thickness}(x) = \textit{thickness}(y)\ \textit{and}\\ & \textit{inner diameter}(x) = \textit{inner diameter}(y)\ \textit{and}\\ & \textit{outer diameter}(x) = \textit{outer diameter}(y)\end{aligned}$$

However, not every property can be used as the basis for the identifying relation. The properties that can be used as the basis for the identifying relation must be anti-rigid and dependent concepts. In order to identify and distinguish the instances by using these properties regardless of suppliers and time the properties should be essential at least in that *PARTS FAMILY* concept. If the properties are not necessary to all instances one parts library may define them whereas another parts library may not define them. For example, such properties as *color* and *weight* of the bearing parts are not essential, so that we cannot expect to identify and distinguish an individual part by using these properties. Also, the property should be dependent on its field of application, i.e. the *PARTS FAMILY* concept. If a property is independent of its *PARTS FAMILY* concept the instances of the concept can have any value of the property. For example, *measuring unit* property is independent of bearing concept, so that the property cannot help to identify an entity as an instance of the *PARTS FAMILY* concept. In our method, the properties that can be used as the basis of the identifying relation are modeled using the *ATTRIBUTE* primitive. The *ATTRIBUTE* primitive corresponds to the *MATERIAL ROLE* ontological distinction.

According to Guarino's theory, the *MATERIAL ROLE* concepts carry ICs by inheriting them from a subsuming concept. Thus, an *ATTRIBUTE* concept should have a subsuming concept to inherit the ICs because it is a *MATERIAL ROLE* concept. This coincides well with the fact as follows. Many of the attributes people think of are role attributes. For example, the *height*, *depth*, *width*, and *thickness* attributes are role concepts which are played by a genuine concept, *length*.

In our method, these genuine attributes (i.e., subsumers of *ATTRIBUTE* concepts) are modeled as a *BASIC ATTRIBUTE* concept. The *BASIC ATTRIBUTE* concept supplies the ICs and an *ATTRIBUTE* concept carries them by inheriting them. However, in real situations, especially in the parts library domain, it is not only difficult but also unnecessary to explicitly define the ICs of a *BASIC ATTRIBUTE* concept. This is because those concepts like *length* usually have intuitive meaning definable only with a simple name and some meta-characteristics such as value format and unit. The *BASIC ATTRIBUTE* concepts are rigid because they must be essential to every concept in which they are used as subsumers of the concepts' *ATTRIBUTE*s. So, the *BASIC ATTRIBUTE* primitive can be thought to correspond to the *QUASI-TYPE* ontological distinction.

The *ATTRIBUTE* concepts which are used to define the global ICs of a *PARTS FAMILY* concept are insufficient. Suppose different suppliers sell the same parts, where the word "same" means that the parts are interchangeable. This implies that they have the same values for the global *ATTRIBUTE*s. However, a buyer wants to know more about other properties in addition to the global *ATTRIBUTE*s to finally pick a part and the suppliers want to differentiate their products using these properties. For example, some suppliers may assign the *ball diameter* property or *number of ball* property to the ball bearing class in order to stress superiority of their parts in resistibility to axial load. However, those properties are only supplementary to the *axial strength* property which is already used as a global *ATTRIBUTE*.

In order to model such additional property knowledge, we introduce the *PARTS MODEL* knowledge modeling primitive. A concept that is modeled using the *PARTS MODEL* primitive defines the additional properties as its local *ATTRIBUTE*s. Although the *PARTS MODEL* concept represents a parts class like a *PARTS FAMILY* concept, the concept should be discriminated from the *PARTS FAMILY* concept. This is because some properties (i.e., the local *ATTRIBUTE*s)

have unambiguous meaning only within that *PARTS MODEL* concept (i.e., within the supplier). The *PARTS MODEL* primitive, therefore, corresponds to the *PHASED SORTAL* ontological distinction. A *PARTS MODEL* concept should be a subconcept of a *PARTS FAMILY* concept because it must inherit the global ICs to determine that two part instances from different parts libraries are the "same" entity in spite of different local *ATTRIBUTE*s.

We also need to organize the parts knowledge into segments in the same way as people confine themselves to a clear context to communicate some concepts to each other. The segments should be explicitly modeled as ontology concepts in parts library ontologies because without them the description of parts knowledge would be verbose. For example, the *fastener* concept that denotes such parts as bolt, nut, and washer provides a useful segment knowledge of the given domain. The *manufactured parts* concept of Fig. 1 is another example. Although we cannot define any explicit membership conditions for the segment concepts, they are rigid in the sense that we can determine whether an entity belongs to the segment or not. Therefore, they correspond to the *CATEGORY* ontological distinction.

Since these concepts cannot be subsumed by other rigid concepts such as the *PARTS FAMILY* concept and *PARTS MODEL* concept (otherwise they would have an IC), they only appear in the uppermost levels of the taxonomy of ontologies. In our method, these segment concepts clarify the meaning of the *PARTS FAMILY* concepts and the *BASIC ATTRIBUTE* concepts. The segment concepts for the *PARTS FAMILY* concepts are modeled using the *PARTS FAMILY CATEGORY* primitive, and the segmenting concepts for *BASIC ATTRIBUTE* concepts are modeled using the *BASIC ATTRIBUTE CATEGORY* primitive.

Finally, we need a primitive to represent meta-characteristics of concepts. Examples of the meta-characteristics may be unique code, textual description, value range and measuring unit. The values of meta-characteristics are qualities of a concept which are present in every instance of the concept and have the same value. In our method, these meta-characteristics are modeled using the *META-ATTRIBUTE* primitive. The *META-ATTRIBUTE* primitive corresponds to the *ATTRIBUTION* ontological distinction.

4. Application to mold and die parts library

We can now go into the detailed discussion on systematization of parts knowledge and its usefulness in automated information integration. Modeling ontologies of real mold and die parts libraries [24,25] for B2B e-commerce is taken as an example task to show how to use the *Parts Library Concept Ontology* for knowledge systematization of parts knowledge. We also analyze briefly an existing ontology of a commercial mold and die parts library to discuss the problems of the ad hoc parts knowledge descriptions. A Web-based parts library mediation system is implemented to show how easily a computer system can merge the well-established ontologies which are modeled using the *Parts Library Concept Ontology*.

4.1. Modeling parts library ontologies

We have built a set of ontologies to integrate automatically different mold and die parts libraries [26]. Fig. 4 schematically shows an excerpt from the resultant ontologies. Before modeling the ontologies of each supplier's parts library (source ontologies), a shared ontology is modeled; our method also follows the hybrid approach like most related work.

In some methods using a hybrid approach like BUSTER the shared ontology is a common global vocabulary that defines in advance all the terms (i.e., concepts) which are necessary to represent domain knowledge. Source ontologies are built by using only the terms defined in the shared ontology. The shared ontology plays the role of a rigid standard. However, in our method, the shared ontology provides only high-level segment concepts such as *manufactured parts* of Fig. 1 and basic parts class and attribute concepts such as *Bearing* and *inner diameter* as a starting point for modeling source ontologies. Those high-level segment concepts and basic parts class concepts guide the source ontologies to have similar aggregation and granularity. Our method assumes that each source ontology may add any new parts classes or attributes and even any new segment concepts. Our method does not assume that source ontologies are only restrictions of the common shared ontology.

In the shared ontology, the *Guide Component for Inner Die* concept is modeled as a *PARTS FAMILY CATEGORY* concept at the uppermost level of the taxonomy. Although we cannot define explicit membership conditions like ICs, the *Guide Component for Inner Die* concept provides a clear boundary in which entities legitimately belong. Also, the *Guide Post Unit* concept is modeled as a *PARTS FAMILY* concept. Makers of press die sets purchase a module which is composed of a guide post, guide post holders, guide bushing. Although many suppliers supply several variations of the module in which the dimensions and some features differ, we can identify a concept that denotes such a module with a general definition, 'a module which supports and guides the die set'. The meaning of the *Guide Post Unit* concept does not change across suppliers or time (i.e., is rigid), and the properties corresponding to such rigid meaning and essential to all the instances can be defined (i.e., supplying global ICs).

Usually, suppliers' parts libraries are implemented with very specific and detailed parts classes such as the *LBGMP*, *LBGFP*, *MYAK*, and *MYCK* (bottom boxes in Fig. 4) without a rigorous upper-level taxonomic structure, because they are convenient for receiving the purchase order or for internally managing the production and delivery. According to our method, those specific parts class concepts are modeled as the *PARTS MODEL* concepts at the lowermost levels of taxonomy in the source ontologies. In those specific parts classes, there are properties defined only by the supplier and only in accordance with the supplier's wishes. Such properties usually correspond to non-principal geometric appearances or non-principal functionalities, so that they are not essential to the instances of the parts class. The properties, therefore, play the role of the local *ATTRIBUTE*s. *Dowel Pin Hole Diameter*, *Stopper Fastening Bolt Diameter*, *Post End Rounding Radius*,

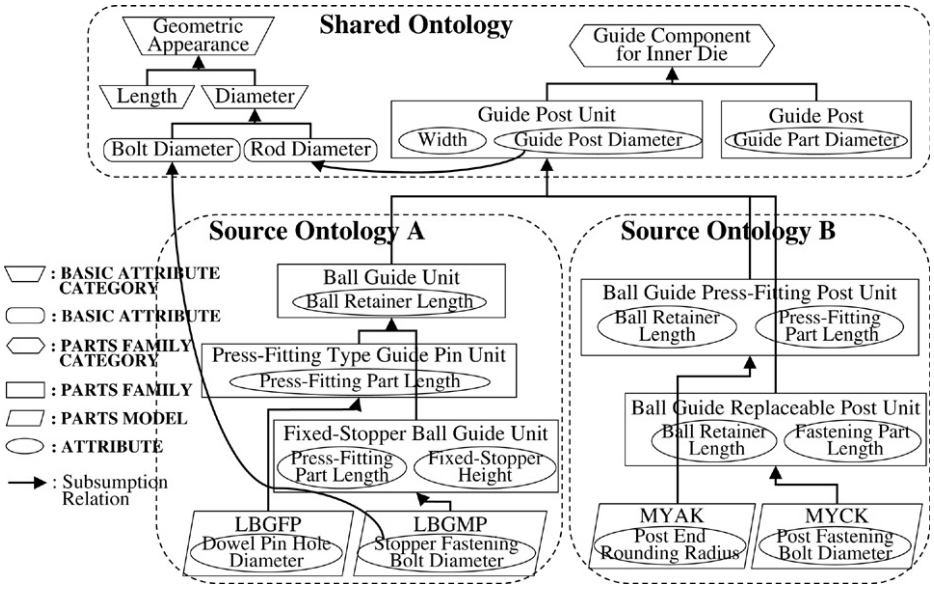

Fig. 4. Parts library ontologies modeled using the meta-ontology (portion).

and *Post Fastening Bolt Diameter* are modeled as local *ATTRIBUTE*s.

The absence of rigorous taxonomic structure usual suppliers' parts libraries means much parts knowledge is implicit. In other words, such knowledge as what the classification criteria of parts classes is, how the attributes are assigned, what the meaning of each attribute is, whether a subsumption relation exists between the parts classes, etc. is hidden. However, in order to integrate and interoperate the source ontologies, such implicit or hidden knowledge should be interpreted and explicated.

The implicit or hidden knowledge is explicated by specializing the basic concepts of the shared ontology into sub-concepts in each source ontology. The sub-concepts are further specialized and finally subsume the specific parts classes. For example, in source ontology A, the *Guide Post Unit* concept of the shared ontology is specialized into the *Ball Guide Unit* concept. Because the basic concept, the *Guide Post Unit* concept, is a *PARTS FAMILY* concept and the specific parts class concepts, *LBGMP* and *LBGFP,* are *PARTS MODEL* concepts, the newly defined concept *Ball Guide Unit* is modeled as a *PARTS FAMILY* concept. Therefore, it must be rigid, and must have global *ATTRIBUTE*s. The specialization is conducted according to the method of guiding the die set in order that the newly defined concepts are rigid and supply global *ATTRIBUTE*s. Since the specialization criterion is concerned with the principal functionalities and usage conditions, the meaning of it does not change across suppliers and time. The *Ball Retainer Length* property is modeled as a global *ATTRIBUTE* because it is necessary to all instances regardless of suppliers in the sense that the length of the ball retainer exposes the ability of the *Ball Guide Unit* such as the guiding range.

On the other hand, source ontology B explicates the hidden knowledge in a different manner from source ontology A. However, conforming to the ontological semantics of the *PARTS FAMILY* primitive leads to comparable results. For example, although source ontology B specializes the *Guide Post Unit* concept of the shared ontology into the *Ball Guide Press-Fitting Post Unit* concept and the *Ball Guide Replaceable Post Unit* concept, it also uses the method of guiding the die set as the primary specialization criterion. No other specialization criteria for the *Guide Post Unit* concept could satisfy the ontological semantics of the *PARTS FAMILY* concept. Consequently, the *Ball Guide Press-Fitting Post Unit* concept is comparable with the *Press-Fitting Type Guide Pin Unit* concept of ontology A, and the *Ball Guide Replaceable Post Unit* concept is legitimately determined as a subconcept of the *Ball Guide Unit* concept.

Without the sophisticated treatment of parts knowledge as suggested by our method we cannot have well-established parts library ontologies. We can easily find ad hoc descriptions of parts knowledge which hinder the automated integration of different parts libraries.

For example, an e-marketplace company in Korea defines an ontology of a mold and die parts library [27] for B2B e-commerce. The e-marketplace company categorizes the die set parts into two parts classes according to the material of the parts: *Castiron Die Set* and *Steel Die Set*. This classification is not appropriate for automated integration. Because the material concept itself does not supply its own ICs, the two parts classes eventually have the same global attributes, which in turn makes it difficult for a computer system to determine of which parts class a die set part is an instance. As a similar example, the company defines a parts class, *Precision Ejector Pin*. This is also incorrect because this parts class cannot add further global ICs to those inherited from its parent class, *Ejector Pin*. In these cases, it is difficult for a computer system to know which supplier's parts classes exactly correspond to the e-marketplace company's parts classes by using ICs, i.e. *ATTRIBUTE*s. As another example, the company defines the property *Bushing Fastening Method* in a parts class *Ball Guide Post Set For Die*

Set. According to our ontological method, this is also incorrect. Based on the ordinary meaning that a ball guide post set part is one in which the principal functionality is to guide the outer die set, the method of assembling its components is not essential.

The above discussion can be summarized as follows. Although our method based on the meta-concepts, i.e., knowledge modeling primitives having explicit ontological semantics, cannot eliminate all possible mismatches among source ontologies, it confines the mismatches to manageable mismatches by reducing the differences in the way a domain is interpreted. Since the meta-concepts provide rigorous constraints on identifying and structuring domain concepts, the "significant" mistakes such as the *Castiron Die Set* and *Steel Die Set* can be avoided in advance. Even when source ontologies are developed independently, comparable domain concepts are distinguished and represented using the same knowledge modeling primitives. These distinguished concepts are structured in a consistent way: similar parts classes are subsumed by the same parent concept; similar attributes are assigned to similar parts classes. On the contrary, when ontologies are developed arbitrarily, as the e-marketplace company did in the above example, the inter-ontology mappings become complicated because mismatches emerge among arbitrary expressions.

4.2. Integrating parts libraries

We have developed a Web-based parts library mediation system [28] that automatically merges the source ontologies and generates an integrated interface for the distributed parts libraries. In the following subsections, we discuss the ontology merging algorithm based on an abstract set-theoretic structure of the parts library ontologies. We also present details of the system implementation.

4.2.1. Ontology merging algorithm

According to our knowledge systematization of parts libraries, the main taxonomic structure of the shared ontology is constituted by the rigid concepts such as *PARTS FAMILY* concepts. The shared ontology, therefore, may be defined as a simplified 4-tuple:

$O^R \equiv \{F^R, A^R, Sub^R, Applic^R\}$, where

- F^R is a set of *PARTS FAMILY* concepts defined in the shared ontology.
- A^R is a set of *ATTRIBUTE* concepts defined in the shared ontology.
- $Sub^R = F^R \rightarrow 2^{F^R}$ is the subsumption function, which associates each *PARTS FAMILY* concept with directly subsumed *PARTS FAMILY* concepts. 2^{F^R} denotes the power set of F^R.
- $ApplicF^R = F^R \rightarrow 2^{A^R}$ is the function that associates each *PARTS FAMILY* concept with its global *ATTRIBUTE* concepts.

The concepts specific to a supplier are modeled in a source ontology by specializing the most similar concept, i.e. the smallest subsumer, which is defined in the shared ontology or in the source ontology itself. The source ontology, therefore, is defined as a simplified 9-tuple:

$O^S \equiv \{F^S, A^S, M^S, Sub^S, Sub\_R^S, SubM^S, SubM\_R^S, ApplicF^S, ApplicM^S\}$, where

- F^S is a set of *PARTS FAMILY* concepts defined in the source ontology.
- A^S is a set of *ATTRIBUTE* concepts defined in the source ontology.
- M^S is a set of *PARTS MODEL* concepts defined in the source ontology.
- $Sub^S = F^S \rightarrow 2^{F^S}$ is the subsumption function, which associates each *PARTS FAMILY* concept of the source ontology with directly subsumed *PARTS FAMILY* concepts of the source ontology.
- $Sub\_R^S = F^R \rightarrow 2^{F^S}$ is the subsumption function, which associates each *PARTS FAMILY* concept of the shared ontology with directly subsumed *PARTS FAMILY* concepts of the source ontology.
- $SubM^S = F^S \rightarrow 2^{M^S}$ is the subsumption function that associates each *PARTS FAMILY* concept of the source ontology with directly subsumed *PARTS MODEL* concepts of the source ontology.
- $SubM\_R^S = F^R \rightarrow 2^{M^S}$ is the subsumption function that associates each *PARTS FAMILY* concept of the shared ontology with directly subsumed *PARTS MODEL* concepts of the source ontology.
- $ApplicF^S = F^S \rightarrow 2^{A^S \cup A^R}$ is the function that associates each *PARTS FAMILY* concept of the source ontology with its global *ATTRIBUTE* concepts.
- $ApplicM^S = M^S \rightarrow 2^{A^S}$ is the function that associates each *PARTS MODEL* concept of the source ontology with its local *ATTRIBUTE* concepts.

The parts library mediation system (see Section 4.2.2 for implementation details) starts the ontology merging process with the source ontologies which are individually connected to the shared ontology. The connections are made through two subsumption relations, Sub_R^S and $SubM\_R^S$. Fig. 4 shows well such connection of ontologies. We call such ontology connection *initially connected ontology*, and this can be defined as a simplified 7-tuple:

$O^{PI} \equiv \{F^{PI}, A^{PI}, M^{PI}, Sub^{PI}, SubM^{PI}, ApplicF^{PI}, ApplicM^{PI}\}$, where

- $F^{PI} = F^R \cup_{(i|1\le i\le n)} F^{S_i}$.
- $A^{PI} = A^R \cup_{(i|1\le i\le n)} A^{S_i}$.
- $M^{PI} = \cup_{(i|1\le i\le n)} M^{S_i}$.
- $Sub^{PI}(\mathtt{f}) = \begin{cases} Sub^R(\mathtt{f}) \cup_{(i|1\le i\le n)} Sub\_R^{S_i}(\mathtt{f}), \\ \quad if\ \mathtt{f} \in F^R \wedge_{(i|1\le i\le n)} \mathtt{f} \notin F^{S_i} \\ Sub^{S_i}(\mathtt{f}), \quad if\ \mathtt{f} \notin F^R \wedge \mathtt{f} \in F^{S_i}. \end{cases}$
- $SubM^{PI}(\mathtt{f}) = \begin{cases} \cup_{(i|1\le i\le n)} SubM\_R^{S_i}(\mathtt{f}), \\ \quad if\ \mathtt{f} \in F^R \wedge_{(i|1\le i\le n)} \mathtt{f} \notin F^{S_i} \\ SubM^{S_i}(\mathtt{f}), \ if\ \mathtt{f} \notin F^R \wedge \mathtt{f} \in F^{S_i}. \end{cases}$
- $ApplicF^{PI}(\mathtt{f}) = \begin{cases} ApplicF^R(\mathtt{f}), if\ \mathtt{f} \in F^R \\ ApplicF^{S_i}(\mathtt{f}), if\ \mathtt{f} \notin F^R \wedge \mathtt{f} \in F^{S_i}. \end{cases}$
- $ApplicM^{PI}(\mathtt{m}) = ApplicM^{S_i}(\mathtt{m})$, *if* $\mathtt{m} \in M^{S_i}$.

Therefore, the initially connected ontology has a single tree structure. However, in the set F^{PI} and M^{PI}, the parts classes that represent the same class can exist at several levels of the tree because the parts classes come from different source ontologies. Also, some subsumption relations between parts classes may not be explicitly defined in the relation specifications, Sub^{PI} (f) and $SubM^{PI}$ (f). So, an algorithm is needed to join the same parts classes into a single parts class and to establish the missing subsumption relations.

The same parts classes and the missing subsumption relations are easily identified because semantically similar parts classes are distinguished and represented using the same knowledge modeling primitives, they are subsumed by the same parent concept, and similar attributes are assigned to them. In the example of Fig. 4, *Press-Fitting Type Guide Pin Unit* of ontology A and *Ball Guide Press-Fitting Post Unit* of ontology B are the same parts classes because they belong to the same knowledge modeling primitive, they have the same parent parts class, and they have the same *ATTRIBUTE*s. The *Ball Guide Press-Fitting Post Unit* of ontology B is a subsumee of the *Ball Guide Unit* of ontology A because the former has all *ATTRIBUTE*s of the latter and has other *ATTRIBUTE*s that the latter does not have.

The ontology merging algorithm is implemented by iteratively applying the following two steps to a parent node and its direct child nodes, in which the parent node is replaced with one of its child nodes in the next iteration. (1) If there exist the same parts classes in the child nodes, delete all the same parts class nodes except one, and move the deleted child nodes' child nodes under the undeleted node. And then, (2) if there exist subsumption relations between the remaining child nodes, create new subsumption relations between them, and delete the current subsumption relation between the parent node and the child nodes. The two steps are applied iteratively because the same parts classes and the missing subsumption relations could exist at several levels of the tree. The iteration process is the same as the well-known pre-order tree search algorithm. The required computing resources such as memory size and CPU power for the algorithm are already known as sufficiently small to be implementable on a personal computer.

The above discussion can be summarized as follows. The ontology merging algorithm is based on the assessment of semantic correspondence between parts classes, and the assessment for parts classes, in turn, can be simplified to an assessment of the semantic correspondence between the *ATTRIBUTE*s defined in each parts class. Although the prototype system (see Section 4.2.2 for implementation details) assumes that the source ontologies to be merged have the same name, i.e., symbol, for the same *ATTRIBUTE*s, this is not a serious defect because the aim of the prototype implementation is to show how easily a computer system can merge the well-established ontologies which are modeled using our ontological method. Once the target ontology elements are legitimately identified with the help of similar conceptualization and consistent structure of the ontologies, the correspondence between them can be easily determined by using the syntactic, lexical, and structural analysis techniques for similarity measuring [21].

4.2.2. Web-based parts library mediation system

Fig. 5 shows the architecture of the Web-based parts library mediation system. This system consists of a mediator, wrappers for each parts library, and a registry. The mediator implements the above described ontology merging algorithm. It remotely accesses the source ontologies by using the access path information that is registered in the registry, and merges them into a single ontology whenever a user starts a new Web server session of the mediator. Using the dynamically merged ontology, the mediator generates an integrated interface to the distributed parts libraries. The source ontologies were encoded using the XML schema definition language, XML Schema 1.1 [29]. They are stored in each local machine of suppliers and exposed as a regular XML file on the Web. When a user asks the mediator to search for parts, the mediator requests the parts search to each wrapper. The search results from each parts library are conveyed to the mediator and gathered in the integrated interface. The wrapper implements the parts search functionality of each parts library as Web Services [30].

The mediation system has been implemented on a personal computer with Microsoft® Windows XP as the operating system, Microsoft® Internet Information Services (IIS) v5.1 as the Web server, and Microsoft®.NET Framework SDK v1.1 as the programming tool kit.

The mediator, called the *Parts Library Mediator*, consists of two tabbed pages. One page is for viewing the ontology, and the other is for viewing the retrieved data.

Fig. 6 is the ontology view page, which shows the dynamically merged ontology. In this figure, we can see that the two *PARTS FAMILY* concepts, *Ball Guide Replaceable Post Unit* and *Ball Guide Press Fitting Post Unit*, of source ontology B are joined with corresponding ontology A's *PARTS FAMILY* concepts and re-structured. The ontology view page also provides the means to specify the parts search conditions. The search conditions are made of specific values of each *ATTRIBUTE*. When the user requests the mediator to search for parts by clicking the *Search in the Selected PF* button, the mediator searches for parts from the selected *PARTS FAMILY* concept to all subsumed *PARTS FAMILY* concepts and *PARTS MODEL* concepts. In this case, only the global *ATTRIBUTE* values are used as the search conditions. When the *Search in the Selected PM* button is clicked, the mediator searches for parts only in the selected *PARTS MODEL* concept. Both the global *ATTRIBUTE* values and the local *ATTRIBUTE* values are used as the search conditions.

Fig. 7 shows the retrieved data view page, which displays the instances that belong to the *Ball Guide Unit* parts class and its sub-classes and satisfy the search conditions depicted in Fig. 6. In the *Parts Family Instance Data Table*, only the global *ATTRIBUTE*s' values of *PARTS FAMILY* concepts' instances are displayed. When the user selects a row from this table (i.e., when an instance of the *PARTS FAMILY* concept is selected), the retrieved data view page displays the instance data of the *PARTS MODEL* concepts which are subsumed under the *PARTS FAMILY* concept in the separated *Parts Model Instance Data Tables*. The *Parts Model Instance Data Tables* display the values of local *ATTRIBUTE*s. Since a *PARTS FAMILY* concept

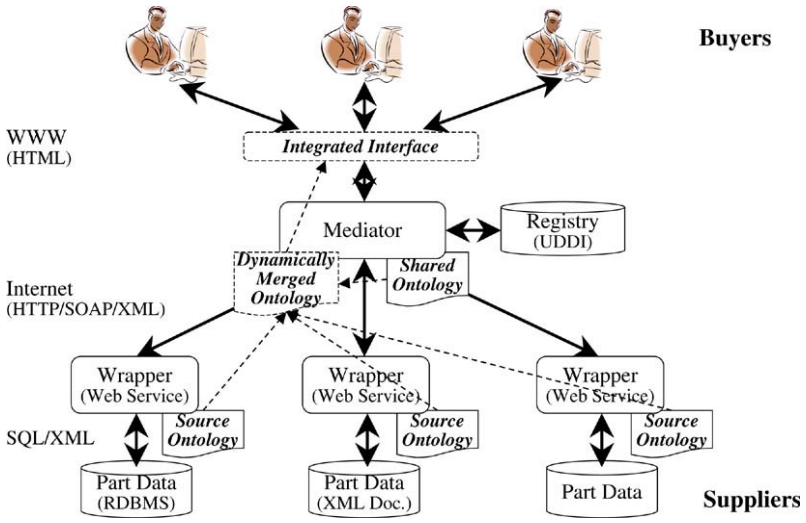

Fig. 5. Architecture of the parts library mediation system.

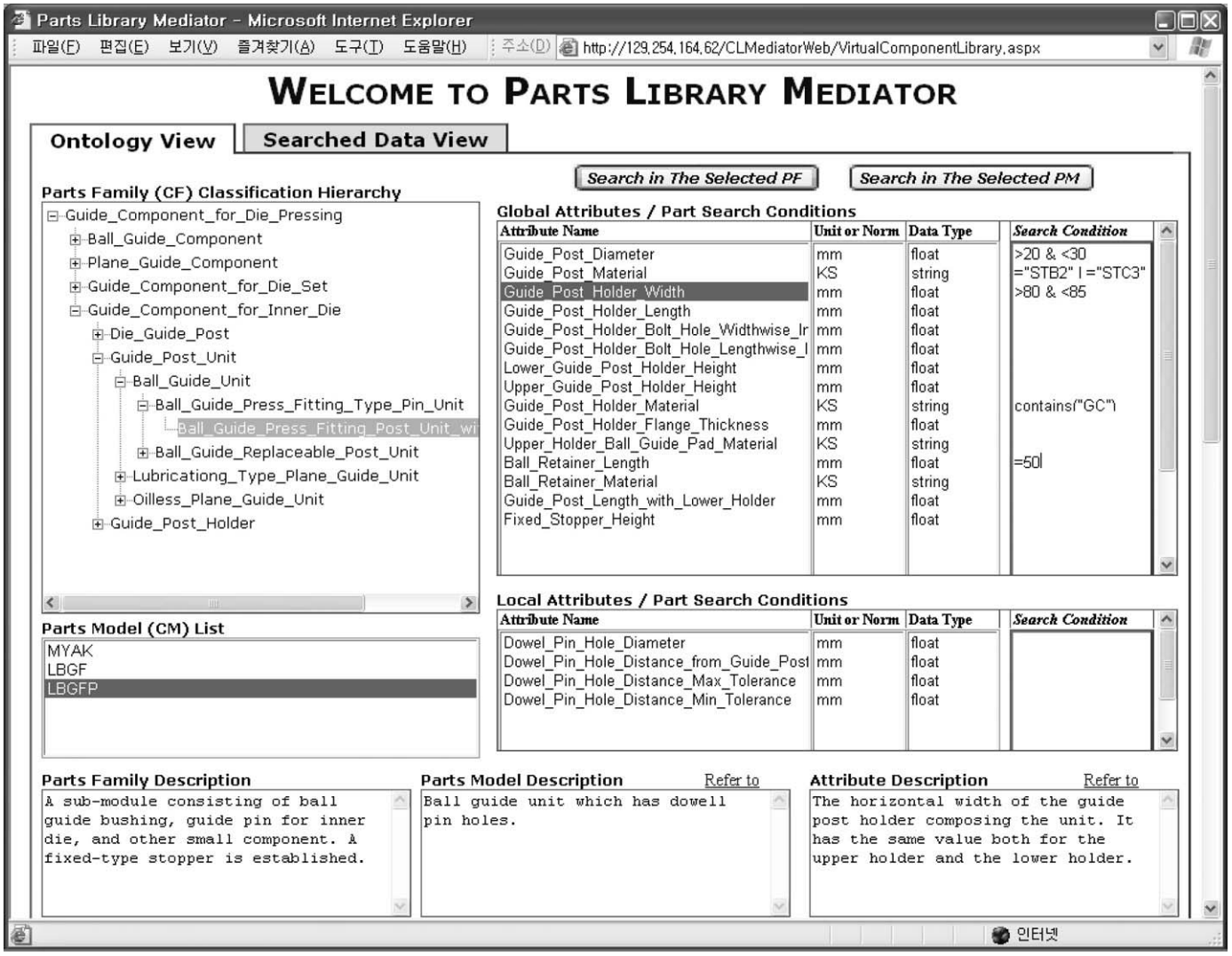

Fig. 6. Ontology view page of *Parts Library Mediator*.

can subsume many different *PARTS MODEL* concepts several *Parts Model Instance Data Tables* may be created. Also, since an instance of a *PARTS FAMILY* concept can appear as many different forms in each *PARTS MODEL* concept, several *PARTS*

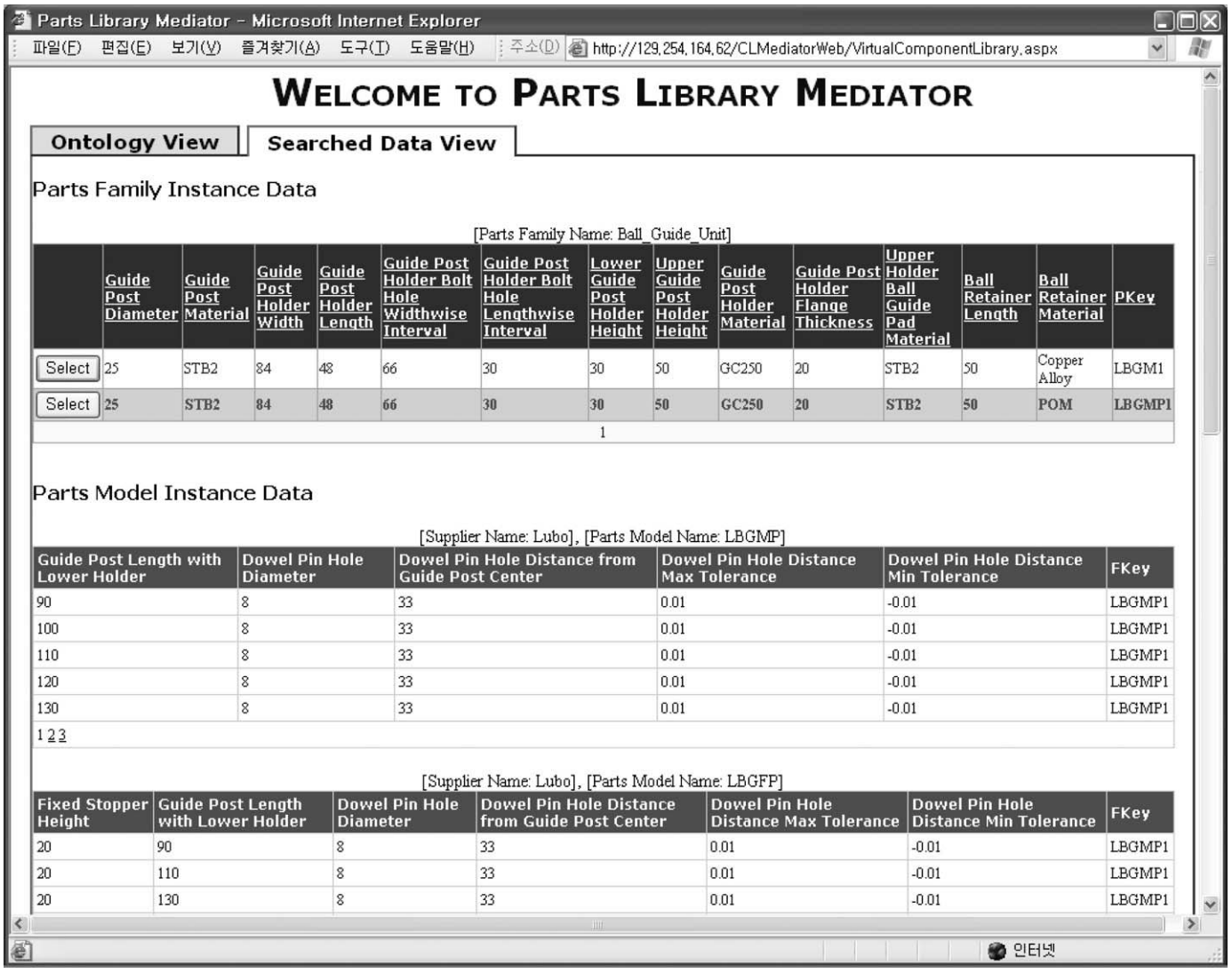

Fig. 7. Retrieved data view page of *Parts Library Mediator*.

MODEL instances can be listed in each *Parts Model Instance Data Table*. Consequently, all properties of a real part can be inspected using the values of the global *ATTRIBUTE*s displayed in the *Parts Family Instance Data Table* together with the values of the local *ATTRIBUTE*s displayed in the *Parts Model Instance Data Table*.

5. Conclusion

We have discussed knowledge systematization based on content-oriented meta-ontologies for automated integration of parts libraries. The *Parts Library Concept Ontology* is introduced to reduce the differences in the way the parts knowledge is interpreted by providing an appropriate structure with appropriate vocabulary, i.e. knowledge modeling primitives. Guarino's theory of upper ontology contributes by giving explicit conceptualization of ontological assumptions to the knowledge modeling primitives. The *Parts Library Concept Ontology* helps ontology modelers to distinguish parts library concepts consistently and structure them systematically, so that ontology mismatches are confined to manageable mismatches. We applied the method for knowledge systematization to modeling the ontologies of real mold and die parts libraries, and demonstrated the usefulness of the method in automated integration of parts libraries using a prototype parts library mediation system.

References

[1] Baron JP, Shaw MJ, Bailey Jr AD. Web-based e-catalog systems in B2B procurement. Communications of the ACM 2000;43(5):93–100.

[2] Satdet E, Pierra G, Murayama H, Ait-Ameur Y. Simplified representation of parts library: Model, practice and implementation. In: Proceedings of the 10th symposium on product data technology Europe. 2001. p. 163–74.

[3] Cui Z, Shepherdson JW, Li Y. An ontology-based approach to eCatalogue management. BT Technology Journal 2003;21(4):76–83.

[4] Handschuh S, Schmid BF, Stanoevska-Slabeva K. The concept of a mediating electronic product catalog. EM-Electronic Markets 1997;7(3): 32–5.

[5] Hakimpour F, Geppert A. Resolving semantic heterogeneity in schema integration: an ontology based approach. In: Proceedings of the international conference on formal ontology in information systems. ACM Press; 2001. p. 297–308.

[6] ISO 10303-11:1994 Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual. ISO/TC 184/SC4, 1994.
[7] Wache H, Vogele T, Visser U, Stuckenschmidt H, Schuster G, Neumann H et al. Ontology-based integration of information — A survey of existing approaches. In: IJCAI-01 workshop: ontologies and information sharing. 2001. p. 108–17.
[8] Visser PRS, Jones DM, Bench-Capon TJM, Shave MRJ. An analysis of ontology mismatches: heterogeneity versus interoperability. In: AAAI 1997 spring symposium on ontological engineering. California (USA): Stanford University; 1997. p. 164–72.
[9] Klein M. Combining and relating ontologies: an analysis of problems and solutions. In: IJCAI-01 workshop: ontologies and information sharing. 2001. p. 53–62.
[10] Goh CH, Bressan S, Madnick S, Siegel M. Context interchange: New features and formalisms for the intelligent integration of information. ACM Transactions on Information Systems 1999;17(3):270–93.
[11] Stuckenschmidt H, Vogele T, Visser U, Meyer R. Intelligent brokering of environmental information with the BUSTER system. In: Proceedings of the 5th international conference 'Wirtschaftsinformatik'. 2001. p. 15–20.
[12] Kopena JB, Regli WC. Design repositories on the semantic web with description-logic enabled services. In: Proceedings of VLDB semantic web and databases workshop. 2003. p. 349–56.
[13] Jurisica I, Mylopoulos J, Yu E. Ontologies for knowledge management: an information systems perspective. Knowledge and Information Systems 2004;6(4):380–401.
[14] Mizoguchi R. A step towards ontological engineering. In: Proceedings of the 12th national conference on AI of JSAI. 1998. p. 24–31.
[15] Guarino N. Formal ontology, conceptual analysis and knowledge representation. International Journal of Human–Computer Studies 1995; 43(5–6):625–40.
[16] Farrugia J. Model-theoretic semantics for the web. In: Proceedings of the 12th international conference on World Wide Web. ACM Press; 2003. p. 29–38.
[17] Kifer M, Lausen F, Wu J. Logical foundations of object-oriented and frame-based languages. Journal of ACM 1995;42(4):741–843.
[18] Fensel D, van Harmelen F, Horrocks I, McGuinness DL, Patel-Schneider PF. OIL: An ontology infrastructure for the semantic web. IEEE Intelligent Systems 2001;16(2):38–45.
[19] Baader F, Calvanese D, McGuinness DL, Nardi D, Patel-Schneider PF, editors. The description logic handbook. Cambridge University Press; 2003.
[20] Pierra G. Context-explication in conceptual ontologies: the PLIB approach. In: Proceedings of CE'2003, special track on data integration in engineering. 2003. p. 243–54.
[21] Silva N, Rocha J. Merging ontologies using a bottom-up lexical and structural approach. In: Seventh international society for knowledge organization conference. 2002.
[22] Guarino N, Welty CA. A formal ontology of properties. In: Proceedings of the 12th European workshop on knowledge acquisition, modeling and management. Lecture Notes in Computer Science (LNCS), vol. 1937. Springer Verlag; 2000. p. 97–112.
[23] Colomb RM. Use of upper ontologies for interoperation of information systems: a tutorial. Technical report 20/02. ISIB-CNR, Padova, Italy, 2002.
[24] Lubo Co. Ltd. Electronic parts catalog, http://www.luboinc.co.kr/english/product/product02_01.htm.
[25] Sinweon Co. Ltd. Electronic parts catalog, http://www.shinweon.com/Eng/product/product_menu.htm.
[26] Cho J, Kim H, Han S. Content-oriented knowledge modeling for automated parts library ontology merging. In: Computer supported cooperative work in design II. Lecture Notes in Computer Science (LNCS), vol. 3865. Springer Verlag; 2006. p. 324–33.
[27] Hub-m.com Co. Ltd. Electronic parts catalog, http://www.hub-m.com/default.asp.
[28] Cho J, Han S, Kim H. Web-based parts library mediation system:, http://129.254.164.61/CLMediatorWeb/default.htm.
[29] W3C. XML Schema Portal, http://www.w3.org/XML/Schema.
[30] W3C. Web Service Portal, http://www.w3.org/2002/ws.