INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 05 MELBOURNE, AUGUST 15-18, 2005

An Architecture for Designers' Support Systems with Knowledge-embedded Documents

HIDEAKI TAKEDA, MASAHARU YOSHIOKA, YOSHIKI SHIMOMURA,
YUTAKA FUJIMOTO, KENGO MORIMOTO, WATARU ONIKI

Keywords: design knowledge management, annotation, XML, RDF, creative design support

1 Introduction

It is no doubt that engineering design is proceeded with a plenty of knowledge so that knowledge management in design is crucial to support design with computers. There are two aspects for knowledge management in design. One is how knowledge is used in design and the other is how knowledge is represented and maintained. With regard to the former, we have investigated how knowledge is used in design and proposed so-called Universal Abduction Studio in which abduction is used to integrate knowledge from different domains for creative design [1]. In the paper, we focus on the latter, i.e., how to represent and store knowledge in design. In particular, we aim to establish the practical method for design knowledge representation, i.e., knowledge is embedded into documents that are used in abduction in designers' support systems.

This paper is organized as follows. In the next sectoin, we dicuss knowledge-based design systems and introduce our project called UAS as a new approach for them. Since knowledge representation is a crucial matter for them, we briefly overview research on knowledge representation in Section 3, and then we investigated knowledge in design as a case study in Section 4. We picked up a book on know-how of mechanical design and extracted knowledge. Then we analyzed the extracted knowledge. With the characteristics on design knowledge taken by the case study, we propose formats for design knowledge in the following two sections. Firstly, in Section 5,we propose an XML-based document format that includes both human-readable texts and computer-understandable knowledge representation according to the result of the analysis of the extracted knowledge. XML-based format is easy to handle both for users and systems because of its simplicity, but lack of flexibility for various modeling. To overcome this problem, we introduce Semantic Web approach, i.e., RDF-based representation in Section 6. It can offer variety of vocabularies for modeling still without loosing simplicity of representation. We build an editor for this representaion interactively. In order to know how this representation is useful for design support systems, we show show reasoning mechanism with this representation. Finally we conclude the paper in Section 7.

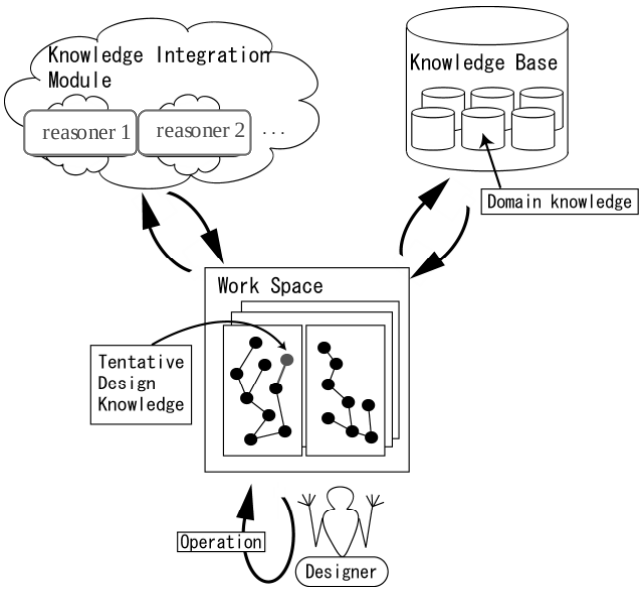

Figure 1. Fundamental concept of the Universal Abduction Studio

2 Background

2.1 Research on Knowledge-based Design Support Systems

Design support systems have been well developed for geometric and detail design stages. In contrast, those in the conceptual design stage are still far from success. In our opinion, the main difficulty comes from incomplete and insufficient understanding about design knowledge and its operations that play a crucial role in conceptual design. Recently, thanks to development of the Internet technologies, more and more knowledge is accumulated and available electronically. It then becomes an interesting research question how to apply such an enormous amount of diverse knowledge to conceptual design.

Design with large-scale knowledge bases has been studies from knowledge sharing point of view. Building ontologies is crusial to realize large scale knowledge sharing [2]. According to this approach, some projects relaeted to engineering design like SHADE[3] and PACT [4] were conducted. Another aproach for large-scale knowledge bases in engineering design is to use physical laws as the backbone to integrate various knowledge [5].

In our point of view, these studies still fail to solve variety of knowledg fully. The first approach tries to integrate various knowledge by logical relation and the second by physical relation. Both types of relation are important in knowledge integration but there are not all, because some of integration of knowledge in design is intentional, i.e., designers explore how knowledge can be integrated to achieve their design goal. Our project called Universal Abduction Studio (UAS)[1] aims to solve this problem in a unique way. The principle of UAS is abduction that leads design processes by integrating knowledge from various domains.

2.2 Universal Abduction Studio

A Universal Abduction Studio (UAS) system [1] is a computer environment to support integration of theories (that contain knowledge) from various knowledge domains for creative

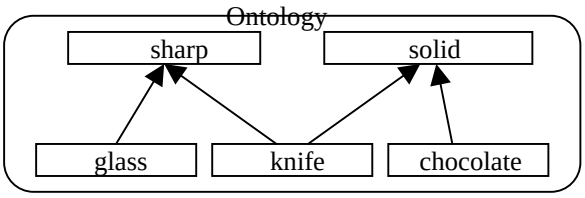

Figure 2**.** A sample of ontology for a scenario

design. UAS is not a design automation system but a cooperation system that can solve design problems by helping dynamic interaction between a designer and the system. UAS provides a toolbox consisting of a variety of domain knowledge as well as a variety of abductive reasoning mechanisms for knowledge integration. When the designer cannot solve a design problem with knowledge of one domain, the designer chooses a knowledge operation to make correspondences between that domain knowledge and another domain knowledge that the UAS system proposes. Then, the designer estimates and judges whether or not the proposed knowledge should be used. Finally, the designer generates design solutions based on the tentative design knowledge chosen by her / him. The basic feature of the system as an inference system is abduction that can integrate knowledge to proceed design processes. Integration realized by abduction is intentional, i.e., while other integration methods like ontological integration are objective. The detail dicsussion of this difference is found in [6].

Figure 1 shows the fundamental concept of UAS. In Figure 1, the designer operates design information and knowledge on the workspace. The knowledge integration module consists of multiple abductive reasoning mechanisms, and the designer chooses one or some of them depending on each design problem. The knowledge base consists of multiple domain knowledge bases and the designer first chooses one to solve a design problem. When the designer cannot solve the design problem, the system reasons about another domain knowledge base that can possibly be integrated with the first domain knowledge. The abductive reasoning system then performs knowledge integration. This fundamental concept requires unified knowledge description among various domain knowledge bases.

3 Knowledge Representation in Design

In this section, we overview knowledge representation in design, and show our basic approach for it.

Knowledge in design is mainly classified into two categories, i.e., knowledge on objects and knowledge on design processes or design procedures [7]. The former is knowledge on how objects are represented and operated, and the latter is knowledge on how designers proceed and complete design.

Many studies focus on object modeling. Typical examples of object modeling are 2D/3D geometric modeling and kinematic modeling. Each object modeling method provides a way of representation based on its aspect. Since any design requires two or more aspects to complete, we should manage multiple object modeling methods so that ontology should be introduced.

Ontology in information systems is introduced in knowledge sharing context. The popular definition of ontology is “an explicit specification of conceptualization”[2]. It provides basic concepts when one wants to represent the target world in some specific context. Each

modeling method assumes some basic concepts that are introduced by the theory that the modeling is based on. These concepts can be components of ontology. Some of these concepts are sharable with other modeling methods, and the others are not. Providing an ontology that consists of such sharable concepts helps managing multiple modeling methods. More concrete discussions on ontologies for engineering design are found in [8].

On the other hand, knowledge on design processes has not been investigated well. In object representation, we can assume some background theory that the object representation is based on. Then what kind of theory can we assume as background theory of design process modeling? We have proposed a logical framework for design processes and shown abduction can be the principle for design process [9]. In this framework, abduction corresponds to the process when a new design candidate is created, while deduction corresponds to the process when it is analyzed and validated.

Abduction for design should not closed in a single domain or modeling but should include knowledge from various domains and modeling methods.

Suppose that we are designing “knife which is always sharp”, while the following information is provided in the ontology shown in Figure 2. Concepts from different domains and modeling methods are connected in this ontology. Glass and knife may be included in an engineering domain knowledge, while chocolate in a cooking domain knowledge. A possible scenario to design it is as follows. First we make an assumption like “if knife is broken, its cross section is sharp” using the knowledge “if glass is broken, its cross section is sharp” and similarity between glass and knife which is come from relations in the ontology. In like manner, we make an assumption like “if knife is grooved, it is easy to break” from the knowledge “if chocolate is grooved, it is easy to break” and similarity between knife and chocolate. The solution is then “grooved knife”. In this example, two fragments of knowledge “if glass is broken, its cross section is sharp” and “if chocolate is grooved, it is easy to break” are jointly used in abduction by relating concepts in different domains and modeling methods with the ontology. It should be noted that each modeling method is based on the specific theory so that ontology covered completely can not expected. While ontology provides universally valid relationship among different modeling methods and domains, abduction is expected to support teleological relationship among them as assumptions [6].

4 A Case Study for Knowledge Representation

As we discussed in the previous section, we expect knowledge for UAS systems as logical or at least semi-logical form because abduction we suppose requires such forms. On the other hand, most of knowledge in real design activities is represented in text and figure. In order to know how and what knowledge can be captured from such information sources, we picked up a book on know-how of mechanical design [10] and tried to extract knowledge. Then we analyzed the extracted knowledge.

We extracted pieces of texts that describe information on design processes as candidates of knowledge from the book. The number of pieces extracted is 350. Then we transformed these pieces of texts into if-then rules.

This transformation is not simple. We set a rough criterion to separate if-part and then-part. If-part represents some observation, and then-part represents some action. Even under this

Table 1. Categorization of extracted knowledge

Focus	If-part	Then-part
Focus on object	If there is (object)	Should … Should not … Is … Is not … Has merits of … Has demerits of ..
Focus on operation on objects	If we (operate) (object)	It is a good design It is a bad design It needs care It has merits of … It has demerits of … It needs care of …
Focus on situation on objects	If (object) is in (situation)	We should … We should not …

criterion, multiple interpretations are observed. For example, “broken glass becomes sharp” can be either interpreted as “if there is glass, breaking it makes it sharp” or “if glass is broken, it becomes sharp”. The latter may seem more natural interpretation but it depends on situation.

For example, when looking for information of glass as candidate of material, the former rule may be useful.

The other issue is categorization. We investigated the collected rules closely and classified into three. The first category is a collection of rules that have objects as if-part. The rules are furthermore categorized into six sub-categories depending on then-part. Each category includes either “should”, “should not”, “is (are)”, is (are) not”, “there are merits that”, and “there are demerits that”. The second category is those of which if-then have operations to objects. This category is also divided into six sub-categories depending on then-part. Each includes either “it is a good design”, “it is a bad design”, “it needs care”, “it has merits of …”, “it has demerits of …” and “it needs care for …”. The third category is those of which include state or situation of object. It is also categorized into two sub-categories. One includes “should” in then-part and the other includes “should not”.

We can find some remarks on tagging for design knowledge through this case study. The first one is that texts have naturally multiple interpretations. Objects are easily identifiable but fragments of knowledge like rules are not. We need the different levels of flexibility for annotation. The second is variety of knowledge. Even though we restricted our investigation to rule-style knowledge, meanings of rules are various. The variety probably comes from situations or contexts when we want to use such knowledge. We listed fourteen categories but they are taken from a single book and we should investigate such categories more systematically.

In the following section, we discuss format of tagging for knowledge, in particular the first point in next section.

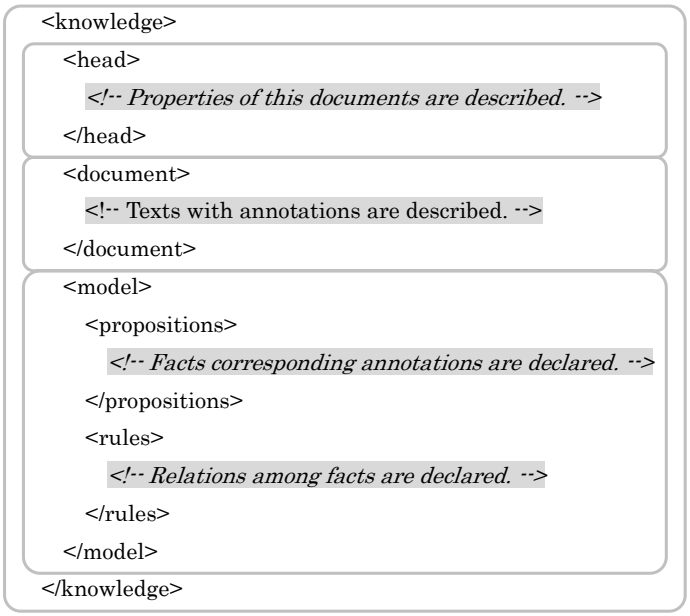

Figure 3. The abstract structure of design knowledge document

5 XML Representation for Design Knowledge Document: A Shallow Approach

Knowledge for design, in particular, knowledge for design processes is often included in documents written in natural languages. Forming knowledge bases by extracting such knowledge from documents is a possible approach but it requires cost for acquisition and maintenance of knowledge. The latter is especially serious because this approach hardly enables to track changes of documents.

The approach in this paper is knowledge as annotation to texts [11]. We call design knowledge document that contains texts and knowledgeable annotations to them. The former is just for human and the latter is mainly for computers but still understandable for human. The benefits of this approach are twofold. One is readability of knowledge. One can easily understand meaning of knowledge since texts can be used as comments for knowledge. This leads to productivity and ease of maintenance of knowledge. We can produce knowledge from existing documents and update knowledge when the corresponding documents are changed. The other is possibility of automatic extraction of knowledge. Because design knowledge documents can be seen as instances of mapping between texts and knowledge, they can be sources to learn this mapping function.

We adopt XML for scheme to represent knowledge in document because XML is popular scheme for documents and has flexibility by specifying own structure by DTD or XML Schema.

Figure 3 shows the abstract structure of design knowledge document. The overall structure is formatted as XML. It consists of three major parts, i.e., “head” part, “document” part and “model” part. “Head” part describes general properties of the document. “Document” part

describes natural language texts with the specific annotation. Without the annotation, they are just texts in documents. “Model” part describes knowledge related to these texts.

We provide “<word>” tag for text annotation. This tag relates the specific part of texts to concepts in knowledge. The form is

<**word base**=”*fundamental-form*” **concept**=”*concept-name*” **id**=”*ID*”> *string*</**word**>

String is related to concept concept-name, word fundamental-form, and ID. Concept-name is associated with concept of this name. Concept is declared either in model part of the same document or in some ontology. Fundamental-form is provided just for natural language processing and search. ID is used in model part to refer the specific occurrence of the concept. For example,

```
<word concept=knife id=knife1>this knife</word>
```

declares that there is an occurrence of concept knife named knife1.

Model part consists of two parts, i.e., proposition part and rule part. In proposition part, facts that are believed true in the document are listed. For example,

```
<proposition id="p1">
<predicate concept="break"/>
<arg idref="knife1"/>
</proposition>
```

<proposition> tag declares a proposition and should include a single <predicate> tag and one or more <arg> tags. Attribute concept for <predicate> and <arg> tags is used to specify the corresponding concept, while attribute idref is used to occurrence of the concept in the document. The example declares

```
break(knife1)
```

where knife1 is the occurrence of knife in the document..Rule part is used to declare rule-style knowledge. The example is as follows;

```
<rule>
<if>
  <atom propositionid="p1"/>
 </if>
 <then>
  <proposition>
     <predicate concept=”sharp”/>
     <arg refid=”knife1”>
   </proposition>
 </then>
</rule>
```

This description is a simple if-then rule. <atom> tag is used to specify a proposition declared in proposition part with propositionid attribute. This example is then declaration of the following rule.

```
If break(knife1) then sharp(knife1)
```

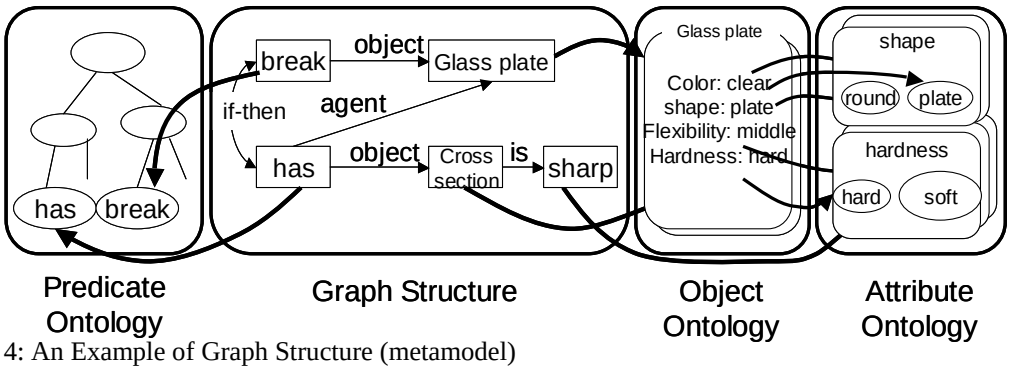

Figure 4: An Example of Graph Structure (metamodel)

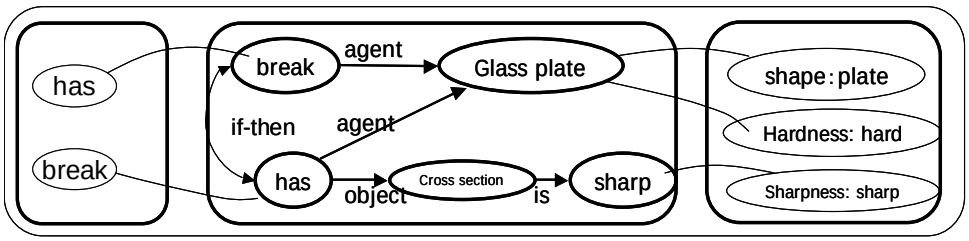

Figure 5: An Example of Graph Structure (aspect-specific model)

The example represented in this format is shown in [12].

This represenation is relatively simple and easy to handle both for users and systems. But it has disadvangates because of its simplicity. The most important disadvantage is lack of multiplicity for variety of knowledge domains. XML is a single layer structure so that we should provide both syntax and semantics of knowledge structure in a single scehme. To overcome this isssue, we step forwards to Semantic Web [13] where semantics is treated independently from XML syntax.

6 Format for knowledge representation for design: A deep approach

In this section, we show our representation scheme for design knowledge with Semantic Web approach. Our policy is to provide a rich structure to describe situations of design. So we introduce objects, attributes, predicates, and case as basic components of knowledge. We introduce knowledge repositories for these components so that we can describe knowledge with multiplicity by choosing and combining elements in these repositories. Furthermore representation is provided as RDF that enables knowledge representation to be published as Semantic Web documents [14].

6.1 The basic structure

As we discussed above, we need to represent rule-style knowledge for design process knowledge. Each hand of a rule is a situation that describes how objects exist. We represent a situation as a set of actions each of which is composed of objects with predicates. More precisely speaking, an action is composed of a predicate with some objects associated by deep case [15], which is used to specify roles of objects. Objects are also associated to some own attributes to specify conditions of objects. Figure 4 is an example of representation in this scheme.

Table 2.Abbreviation for namespaces

Prefix	URI	Meaning
Rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	RDF
Rdfs	http://www.w3.org/2000/01/rdf-schema#	RDF Schema
Kd	http://samurai.race.u-tokyo.ac.jp/UAS/KnowledgeDocument#	Vocabulary for knowledge document
Dec	http://samurai.race.u-tokyo.ac.jp/UAS/DeepCase#	Vocabulary for deep case
Undef	http://samurai.race.u-tokyo.ac.jp/UAS/Undef#	Undefined ontology
Object	http://samurai.race.u-tokyo.ac.jp/UAS/Object#	Object ontology
predicate	http://samurai.race.u-tokyo.ac.jp/UAS/Predicate#	Predicate ontology
IiVerb	http://samurai.race.u-tokyo.ac.jp/UAS/IIVerb#	Vocabulary for object modifiers
Adverb	http://samurai.race.u-tokyo.ac.jp/UAS/Adverb	Ontology for predicate modifiers
shapness etc.	http://samurai.race.u-tokyo.ac.jp/UAS/Attribute/Sharpness	Attribute ontologies

In order to represent an instance of knowledge (a rule), we should provide knowledge repositories. We provide object ontology, attribute ontology, and predicate ontology for this purpose. The object ontology contains information on objects in a structured manner, i.e., as a hierarchy. Each object has a set of attribute names and values that are provided by the attribute ontology. On the other hand, the predicate ontology provides a set of possible actions also as a hierarchy. A predicate has a set of deep cases that can associate objects in specific roles.

A top-down composition of a rule is as follows. Suppose to describe a rule “If a glass plate is broken, its cross-section is sharp”. First we should identify pre-condition situation. It means that we should identify actions in the situation. In this case, it is “A glass plate is broken”. In this action, “glass plate” is agent case for predicate “break”. Then the same procedure is performed for post-condition part. We identify “the glass plate has cross-section and cross-section is sharp”. In this case, the first action has “has” as predicate, “glass plate” as its agent case, and “cross-section” as object case. The overall result of this process is shown in Figure 4. We call this representation “metamodel”.

It is merely a literal translation of the above sense, and we usually add some attributes by specifying aspects because some common sense knowledge is often missed in such a sentence. In this case, we assume that the sentence is uttered under the material aspect, i.e., how material behaves in various situations. We pick up shape and hardness for example. The final description is shown in Figure 5. We call this representation as “aspect-specific model” that corresponds to representation in a model in traditional engineering domain.

By separating metamodel and aspect-specific model, variety of representation is realized without loosing integrity. The basic structure taken from documents is represented as metamodel, while detailed information specific to aspects is represented as aspect-specific model.

Deep case

We adopt deep case to specify relationship between predicates and other entities. A deep case is a concept relation label indicating the deep-level relation between the verb and other words [16]. We use twelve cases taken from EDR [17], namely, agent, object, source, goal, place, scene, implement, material, purpose, cause, quantity, and beneficiary.

Predicate ontology

We provide vocabulary for actions as predicate ontology also taken from EDR concept

dictionary. Here predicate concepts are hierarchically organized. When composing knowledge, this hierarchy works as guide for users to find appropriate concepts. When the system calculates similarity of knowledge for analogical reasoning, it works as a structure to determine similarity of concepts.

Object ontology
Object ontology stores information on objects. Each object information contains attributes and their default values.

Attribute ontology
Attribute ontology contains attribute names and their possible values. Values are organized hierarchically, e.g., shape attribute is classified into plate-like and solid, and solid is furthermore classified into polyhedron and sphere. This hierarchy is used for guide and for measuring similarity.

6.2 Representation by RDF

Both metamodel and aspect-specific models are represented as graph so that we can easily translate them into RDF graph model. An example of aspect-specific model is shown in Figure 6. It corresponds to the aspect-specific model shown in Figure 5.

In this example, knowledge "If glass is broken, its cross section is sharp" is decomposed in the following way.
1. Object "glass " is an instance of "Object:glass".
2. Object "cross section" is an instance of "Object:cross_section".
3. Predicate "break" is an instance of "Predicate:break".
4. Attribute "sharp" is an instance of "Shapness:sharp".
5. "Its cross section is sharp" is interpreted as "glass has a cross section and the cross section is sharp".
6. "Glass" is focused by its sharpness and shape.

In Figure 6, relationships to elements in ontologies are omitted due to avoiding complexity. Namespaces used in this example are shown in Table 2[1]. According to RDF serialization to XML, we can obtain XML descriptions from RDF graph models.

6.3 Editor tool

We built editor that can help users to compose knowledge with this schema, since searching and combining elements in different knowledge repositories is complex work. The editor enables users to operate graphs graphically, e.g., adding or removing nodes and links. Nodes can be labelled by specifying type of node like object, attribute, and predicate. Links can also labelled like deep case. It also helps users to identify nodes added by users with existing ontologies. It can translate graph models into RDF/XML data. The editor is built with JGraph[2] graphic library and Jena[3] RDF library on Sun Microsystems J2SE v 1.4.2_07 SDK. Figure 7 shows a snapshot of the editor. Different colours represent different roles of nodes. A user can

[1] We omit explanation on predicate and object modifiers because of page limitation. They basically correspond to adverb and adjective in grammer respectively.
[2] http://www.jgraph.com/
[3] http://jena.sourceforge.net/

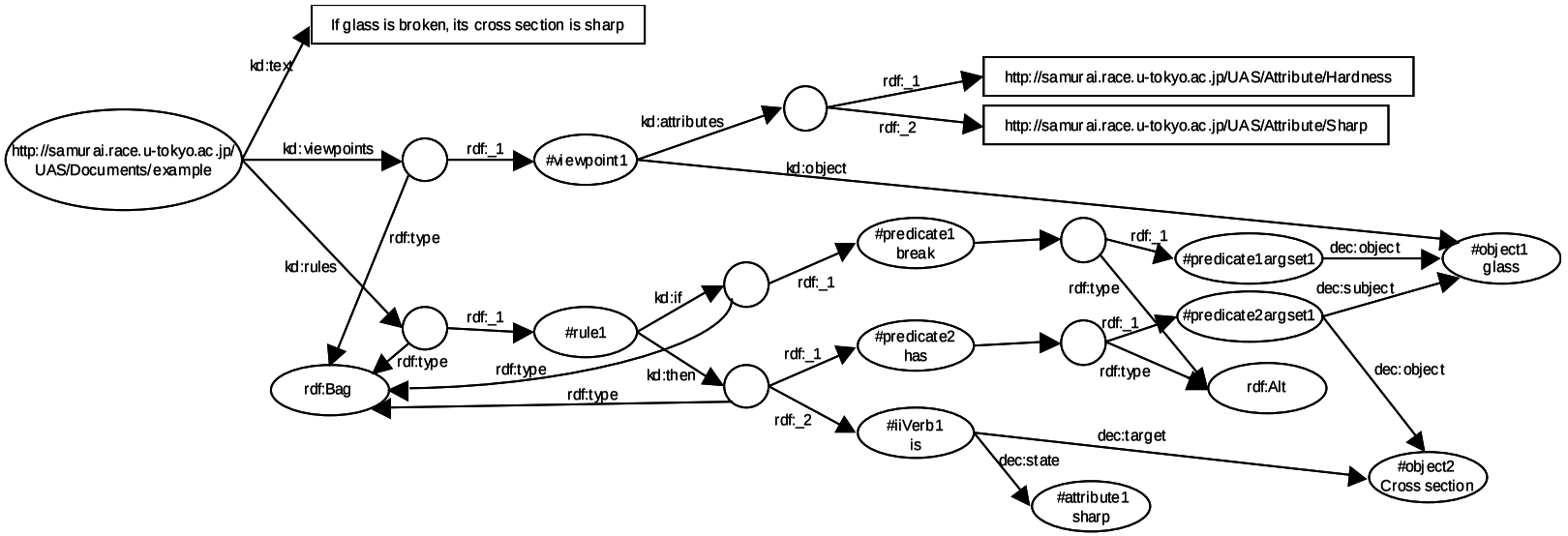

Figure 6. An example of RDF graph

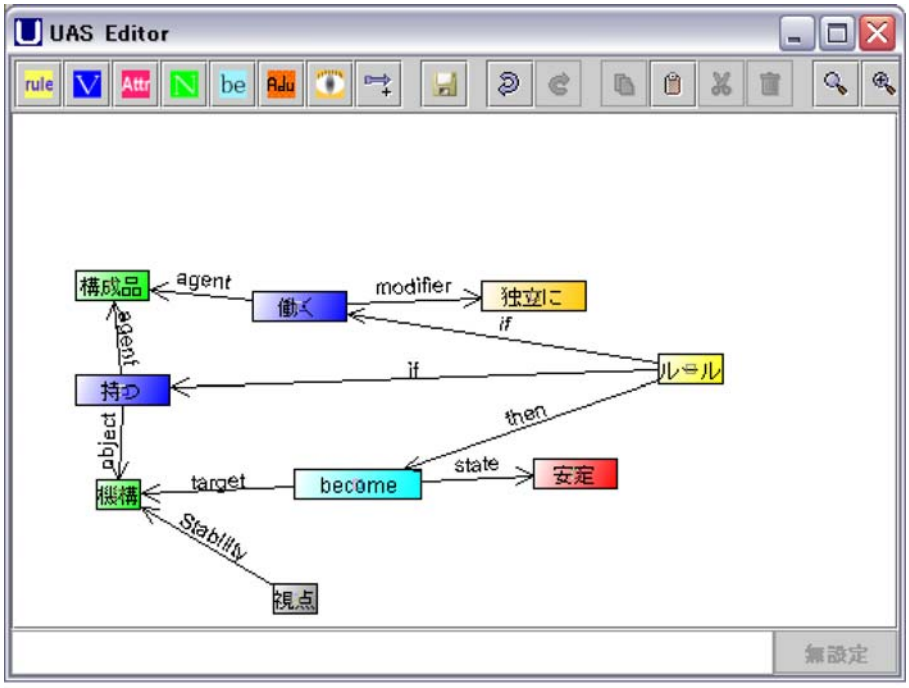

Figure 7. A snapshot of UAS editor

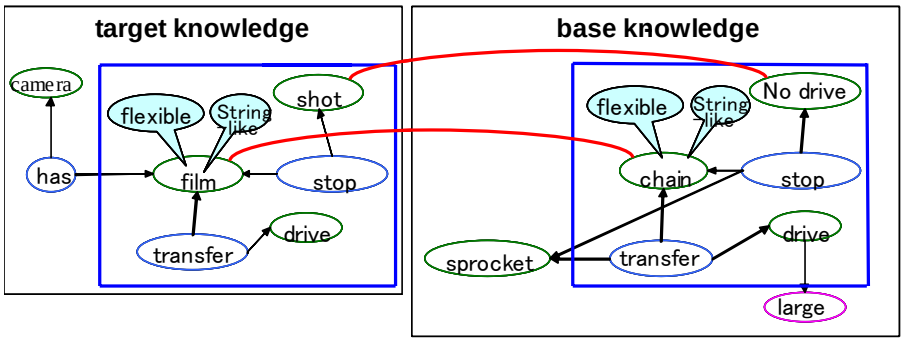

Figure 8. An example of mapping between target and base knowledge

add these nodes and links easily by clicking bottoms in menu shown in the upper part of the window.

6.4 Reasoning

As we dicussed in [1], in order to realize abduction to integrate different knowledge, it is necessary to find correspondence between elements in difference knowledge. In the paper, we proposed use of analogy to find new abduction candidates. The crucial issue in analogy is how to measure similarity between nodes in different knowledge.

The goal of analogy here is to find mapping between objects in the target graph and those in the base graph. Criteria for mapping are similarity of structure, i.e., similarity of links including labels, and similarity of objects connected by these links.

Focality

In our approach, we specify a predicate as “key predicate” that works as focal point over the entire graph. Mapping is to be found only for predicates surrounding the key predicate. We

take distance from the key predicate into account when calculating importance of nodes. We call it “focality (FC)”. Focality of an object is inverse of distance from the key predicate. Focality of a predicate is measured by average of focality of objects directly connected to the predicate.

Focality of predicates in the basic graph is defined in a similar way. Difference is how to specify the key predicate. We set the most similar predicate to the key predicate in the target graph by semantic similarity as key predicate in the basic graph.

Semantic similarity

On the other hand, semantic similarity between two predicates is measured by distance in predicate ontology. In predicate ontology, all predicates are organized as a tree structure. We call depth of a node as distance from root node to it. When two nodes are specified, part of paths from the root node can be shared. We call depth of the deepest node in the shared path as common depth between two nodes. Then we can specify semantic similarity (SeS) between two predicates is defined as follows:

$$SeS(p_1, p_2) = \frac{2 \times common\_depth(p_1, p_2)}{depth(p_1) + depth(p_2)}$$

The same method is applied to similarity between attributes by using attribute ontology.

Structural similarity between objects

Finally we define structural similarity between an object in the target graph and an object in the base graph. Structural similarity is defined by similarity of links and similarity of predicates connected by these links. First we compare types of deep case that are connected to the objects. If there exists the same case, then we calculate similarity of two predicates that are connected by links labelled as the case.

$$StS(o_t, o_b) = \sum FC(p_t) FC(p_b) SeS(p_t, p_b)$$

$$where \;\; case\_link(o_t, p_t, c_1) \wedge case\_link(o_b, p_b, c_2) \wedge c_1 = c_2$$

Attribute-based similarity between objects

If both graphs are aspect-specific models, we can furthermore take similarity of aspects into count.

$$AS(o_t, o_b) = \sum_{a \in A(o_t)} \frac{SeS(a_t, a_b)}{|A(o_t)|} \quad where\ A(o) = \{a \mid attribute\_link(o, a)\}$$

We can measure how mapping between objects in the target and base graphs is valid with these two types of similarity measurement.

An Example

We show an example to illustrate how similarity is found by this procedure and is used to create new knowledge. Support the following sentence as the target knowledge: “A mechanism which transfers drive to film and stop film precisely during shot.” and the following sentence as the base knowledge: “If sprocket transfers large drive to chain, sprocket can stop chain when there are no drive”. We can build both target and base graph shown in Figure 8 (case labels are omitted). Then the above procedure indicates that two mapping are

feasible (two arcs in Figure 8). We can interpret this mapping as knowledge like “If sprocket transfers drive to film, sprocket can stop film during shot”. Of course, this mapping is just hypothesis but to be proven. But generating such a hypothesis by integrating different knowledge is one of the necessary features for creative design.

7 Conclusion

In this paper, we show a novel knowledge representation to support creative design process. Our RDF-based representation consists of modules for knowledge repositories like predicate and object ontologies. A user can compose an individual model for knowledge by selecting and combining suitable elements in these knowledge repositories. This representation solves two characteristics observed in our case study analysis, i.e., multiplicity of interpretation and variety of knowledge. The graph structure can generate multiple interpretations with minimum modification such as redirecting “if” and “then” links. Furthermore, since we separate metamodel and aspect-specific models, we can obtain various aspect-specific models from a single metamodel. The problem to handle with variety of knowledge is balance of expansivity and consistency. Since we provide categories of knowledge like object and predicate ontologies in order to clarify roles of knowledge, we can add easily new knowledge like domain-specific knowledge according to its roles without avoiding messy

Our representation is also useful for design support systems. A composed model can be published as RDF/XML that is a standard for WWW publishing. It is easy to use not only by our systems but also other systems. We also show how this representation is used in reasoning. Similarity between objects is calculated by using structural information of models and ontologies, which is an important process to realize creative abduction.

Reference

[1] Takeda, H., Sakai, H., Nomaguchi, Y., Yoshioka, M., Shimomura, Y., Tomiyama, T.: Universal abduction studio – proposal of a design support environment for creative thinking in design –. In Folkman, A., Gralen, K., Norell, M., Sellgren, U., eds.: The Fourteenth International Conference on Engineering Design (ICED 03), Stockholm (2003)

[2] Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. Technical Report KSL 93-4, Knowledge Systems Laboratory, Stanford University (1993)

[3] McGuire, J. G., Kuokka, D. R., Weber, J. C., Tenenbaum, , M., J., Gruber, T. R. and Olsen, G. R.: SHADE: Technology for knowledge-based collaborative engineering, Journal of Concurrent Engineering: Applications and Research (CERA), Vol. 1, p. 2 (1993).

[4] Cutkosky, M. R., Engelmore, R. S., Fikes, R. E., Genesereth, M. R., Gruber, T. R., Mark, W. S., Tenenbaum, J. M. and Weber, J. C.: PACT: An Experiment in Integrating Concurrent Engineering Systems, IEEE Computer, Vol. January 1993, pp. 28–38 (1993).

[5] Ishii, M., Sekiya, T. and Tomiyama, T.: A very large-scale knowledge base for the knowledge intensive engineering framework, in Mars, N. ed., KB&KS’95, the Second International Conference on Building and Sharing of Very Large-Scale Knowledge Bases, pp. 123–131, IOS Press, Ohmsha (1995).

[6] Tomiyama, T.: From general design theory to knowledge-intensive engineering. Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM) 8 (1994) 319–333

[7] Yoshioka, M., Umeda, Y., Takeda, H., Shimomura, Y., Nomaguchi, Y., Tomiyama, T.: Physical concept ontology for the knowledge intensive engineering framework. (Advanced Engineering Informatics), Vol. 18, No. 2, pp. 95–113 (2004).

[8] Takeda, H., Tomiyama, T., Yoshikawa, H.: A Logical and Computerable framework for reasoning in design, in D. Taylor and L. Stauffer eds., Design Theory and Methodology -- DTM '92 --, pp. 167–174, The American Society of Mechanical Engineers (ASME) (1992).

[9] Takeda, H.: Abduction for design. In Gero, J., Sudweeks, F., eds.: Proceedings of the IFIP WG5.2 International Workshop on Formal Design Method for CAD, Tallinn, Elsevier Science Publishers B.V. (1993)

[10] Watanabe, H.: Hits for mechanical design, A second series. Nikkan Kogyo Shinbun (1998) (In Japanese).

[11] Yoshioka, M. and Shamoto, Y.: Knowledge Management System for Problem Solving -- Integration of Document Information and Formalized Knowledge --. Proceedings of the 2003 ASME Design Engineering Technical Conference & Computers and Information in Engineering Conference, The American Society of Mechanical Engineers (ASME), New York, DETC2003/CIE-48217 (CD-ROM), (2003)

[12] Takeda, H., Fujimoto, Y., Yoshioka, M., Shimomura, Y., Morimoto, K., Oniki, W.: Tagging for intelligent processing of design information, in New Frontiers in Artificial Intelligence: Joint Proceeding of the 17th and 18th Annual Conferences of the Japanese Society for Artificial Intelligence, Lecture Notes for Computer Science, Springer (2005), (To appear).

[13] Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American (2001)

[14] Manola, F., Miller, E.: RDF Primer, W3C Recommendation 10 February 2004, http://www.w3.org/TR/rdf-primer/, 2004

[15] Fillmore, C. J.: The case for case. In Bach, E. et al., editors, Universals in Linguistic Theory. Holt, Rinehart, and Winston, 1968.

[16] Yokoi, T.: The EDR Electronic Dictionary. Communications of ACM Vol. 38, No. 11, pp. 42-44 (1995)

Corresponding authors name: Hideaki Takeda
Institution/University: National Institute of Informatics
Address: 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, Japan
Phone: +81-3-4212-2543, Fax: +81-3-3556-1916, E-mail: takeda@nii.ac.jp