Tagging for Intelligent Processing of Design Information

Hideaki Takeda[1], Yutaka Fujimoto[2], Masaharu Yoshioka[3], Yoshiki Shimomura[2], Kengo Morimoto[2], and Wataru Oniki[2]

[1] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
takeda@nii.ac.jp
[2] Research into Artifacts, Center for Engineering (RACE), The University of Tokyo, Komaba 4-6-1, Meguro-ku, Tokyo 153-8904, Japan
{yutaka_f, simomura}@race.u-tokyo.ac.jp
[3] Graduate School of Information Science and Technology, Hokkaido University Kita 14 Nishi 9, Kita-ku, Sapporo, Hokkaido, 060-0814, Japan
yoshioka@ist.hokudai.ac.jp

Abstract. This paper describes how to add tags to design documents in order to extract knowledge from information for intelligent design support. Our project called Universal Abduction Studio (UAS) aims to build a new design support system that supports conceptual design by dynamically integrating knowledge in different design domains. This paper focuses on knowledge description form which can be used to capture knowledge from text-based information and then be used for inference for creative design. We propose so-called *design knowledge document* containing both human-readable texts and machine-readable knowledge such as propositions and rules.

1 Introduction

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.

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. In order to apply abduction, we need knowledge in logical or at least semi-logical form. How to describe such knowledge is the main topic of this paper.

In this paper, we first introduce UAS project briefly in Section 2. Then we discuss knowledge representation suitable for our purpose in Section 3. After we investigated knowledge in a book in Section 4, we propose our knowledge representation format in

2 Universal Abduction Studio (UAS)

In this section, we explain ideas behind UAS project and basic concepts in it briefly. More detail descriptions are found in [2].

2.1 Abduction in Design

The key issue to build a CAD system capable of supporting the early stages of design, in particular, its creativity aspects, is how to represent design processes. It is no doubt that a creative design process is one of the most intellectual thought processes and is difficult to model. This is not only because generating a creative product or idea itself is hardly inimitable by computers, but also because the knowledge used for creative design is generated, modified, and updated during the process. Once a designer achieves a new creative design after struggles, s/he is able to perform similar designs easily, which implies that her/his knowledge was expanded. We believe that the expansion of knowledge is a mandatory nature of creative design. Creative design therefore has two aspects, i.e., creating a new product and expanding knowledge, and the co-relation between these two is common to various creative activities.

How can we, then, model creative design processes with these two aspects? We have discussed formalization of design processes from the logical viewpoint [3][4][5]. In short, our theory models design as iteration of deduction and abduction (see also Coyne[6], Roozenburg and Eekels[7], while they did not offer any computing mechanisms). In our theory, the core part of design, i.e., creating a new idea or thing can be attributed to "abduction" while ensuring design to deduction. Abduction is thus the crucial part in design.

What is abduction and what can abduction offer as reasoning? Abduction proposed by C.S. Peirce is a logical process to find axiom from theorem [8]. The naïve interpretation is that abduction is an opposite process of deduction. Although this naïve interpretation is somewhat popular within computer science [9], abduction should be interpreted from wider viewpoints and therefore include more various types of reasoning. Schurz [10] collected various types of abductive reasoning and categorized them. In his work, abduction is firstly classified into three, i.e., factual abduction (first-order existential abduction), law abduction, and second-order existential abduction. However, since law abduction seems a sub-category of second-order existential abduction, we regard that distinction of factual and law abduction is the primary classification of abduction. The former concerns discovery of facts and the latter concerns discovery of new laws. "Abduction as inversed deduction" is merely one category of factual abduction.

As we mentioned above, the point of this paper is the dynamics of knowledge when formalizing design processes. Factual abduction infers new facts from given facts (observable facts) with fixed theory (rules). However, as long as we use reasoning with a fixed theory, the ability to create new facts is limited. In addition, although factual abduction can satisfy the primary requirement of abduction ("finding axioms from a theory"), this interpretation does not qualify another important feature of abduction mentioned by Peirce. He explained that abduction can find a "surprising" fact. "Inversed deduction" is insufficient to realize such a process and abduction is

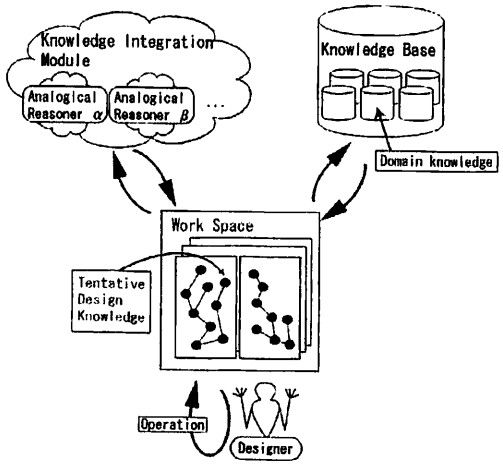

Fig. 1. Fundamental concept of the Universal Abduction Studio

necessarily accompanied by expansion or revision of knowledge [11]. In the domain such as design in which rich knowledge is available, a feasible expansion of knowledge is obtained by integrating existing knowledge [12]. Integration of knowledge here does not mean a simple addition of knowledge, but rather such operations as translation and modification. There seem to exist a number of possible ways to integrate knowledge. Abduction as a method to integrate knowledge can satisfy the two aspects of creative design, i.e., creating a new product and expanding knowledge. Therefore, we believe that abduction can be one model of creative design processes.

2.2 The Architecture of Universal Abduction Studio Systems

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

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 [13]. 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"[14] 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 [15].

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 [3]. 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 cut, it is sharp" using the knowledge "if glass is cut, it is

ontology. In like manner, we make an assumption like "if knife is grooved, it is easy to cut" from the knowledge "if chocolate is grooved, it is easy to cut" and similarity between knife and chocolate. The solution is then "grooved knife". In this example, two fragments of knowledge "if glass is cut, glass is sharp" and "if chocolate is grooved, it is easy to cut" 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 [12].

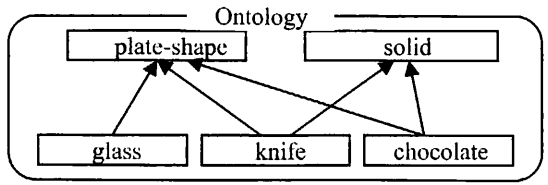

Fig. 2. A sample of ontology for a scenario

4 A Case Study for Knowledge Representation for UAS

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 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 [16] 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 criterion, multiple interpretations are observed. For example, "cut glass becomes sharp" can be either interpreted as "if there is glass, cutting it makes it sharp" or "if glass is cut, the glass 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

needs care for ...". The third category is those of which include state or situation o 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 cas study. The first one is that texts have naturally multiple interpretations. Objects ar easily identifiable but fragments of knowledge like rules are not. We need th different levels of flexibility for annotation. The second is variety of knowledge. Eve though we restricted our investigation to rule-style knowledge, meanings of rules ar various. The variety probably comes from situations or contexts when we want to us such knowledge. We listed fourteen categories but they are taken from a single boo and we should investigate such categories more systematically.

In the next section, we discuss format of tagging for knowledge. We mainl concern the first point in this paper and let the second left for future research.

5 Design Knowledge Document and Its Format

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

```
<knowledge>
  <head>
    <!-- Properties of this documents are described. -->
  </head>
  <document>
    <!-- Texts with annotations are described. -->
  </document>
  <model>
    <propositions>
      <!-- Facts corresponding annotations are declared. -->
    </propositions>
    <rules>
      <!-- Relations among facts are declared. -->
    </rules>
  </model>
</knowledge>
```

Fig. 3. The abstract structure of design knowledge document

The approach in this paper is knowledge as annotation to texts [17]. 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 focus on the first point in this paper and let the second point left as a future task.

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="cut"/>
<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

```
cut(knife1)
```

where knife1 is the occurrence of knife in the document..


```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE knowledge (View Source for full
doctype...)>
- <knowledge
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:fc="http://www.race.u-tokyo.ac.jp/fc/"
xmlns:rl="http://www.race.u-tokyo.ac.jp/rl/">
- <head>
<knowledgeid>003</knowledgeid>
<title>Knowledge on machining setup</title>
<creator />
<date>2004-08-13</date>
<version>1.0</version>
- <origin>
<dc:title>Hits for mechanical design,
A second series</dc:title>
<dc:creator>Hidenori Watanabe</dc:creator>
<dc:publisher>Nikkan Kogo Shinbunsha
</dc:publisher>
<dc:date>1990-08-29</dc:date>
</origin>
</head>
- <document>
<word base="fewer"
concept="predicate:fewer">Fewer</word>
<word base="number" concept="upper:number">
numbers</word> of
<word base="setup" concept="upper:setup"
id="setup">
setup</word> and
<word base="change" concept="predicate:change"
id="change">
changes</word> of
<word base="position"
concept="upper:machiing-position"</word>
id="position">position</word> and
<word base="attitude" concept="upper:attitude"
id="attitude">
attitude</word> in
<word base="machining" concept="upper:machining"
id="machining">machining</word>
<word base="cause" concept="predicate:cause"
id="cause">
cause</word>
<word base="decrease" concept="predicate:decrease"
id="decrease1">decrease</word> of
<word base="man-hour"
concept="upper:time,engineering:man-hour"
id="man-hour">
man-hour</word>
<word base="cost-reduction" concept="upper:cost,
engineering:cost-reduction" id="cost-reduction">
cost reduction</word>
</document>
```

```
- <model>
- <propositions>
- <proposition id="p1">
<predicate
concept="upper:machining" />
<arg idref="machining" />
</proposition>
- <proposition id="p2">
<predicate
concept="upper:setup" />
<arg idref="setup" />
</proposition>
- <proposition id="p3">
<predicate
concept="predicate:own" />
<arg idref="machining" />
<arg idref="setup" />
</proposition>
...(several lines are omitted)
</propositions>
- <rl:rules>
- <rl:rule>
- <rl:if>
- <rl:and>
<rl:atom propositionid="p1" />
<rl:atom propositionid="p2" />
<rl:atom propositionid="p3" />
- <rl:not>
<rl:atom propositionid="p4" />
</rl:not>
<rl:atom propositionid="p5" />
- <rl:not>
<rl:atom propositionid="p6" />
</rl:not>
<rl:atom propositionid="p7" />
- <rl:not>
<rl:atom propositionid="p8" />
</rl:not>
</rl:and>
</rl:if>
- <rl:then>
- <rl:and>
<rl:atom propositionid="p9" />
<rl:atom propositionid="p10" />
<rl:atom propositionid="p11" />
<rl:atom propositionid="p12" />
<rl:atom propositionid="p13" />
<rl:atom propositionid="p14" />
</rl:and>
</rl:then>
</rl:rule>
...(several lines are omitted)
</rules>
</model>
</knowledge>
```

Fig. 4. An example of design knowledge document (translated from descriptions in Japanese

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 cut(knife1) then sharp(knife1)
```

Figure 4 shows an example of description with this syntax.

By providing design knowledge documents, human can understand and use these documents as usual on one hand, the system can extract their knowledge part and use them in its inference. An example of inference is shown in [1].

6 Summary

In this paper, we discussed knowledge representation in design with a case study and proposed so-called design knowledge document that has an XML-based format including both human-readable texts and computer-understandable knowledge.

There are many issues to be done. The current format for design knowledge document is tentative to need more discussion. Especially Semantic Web approach [18] rather than XML is more suitable for our purpose. We are going to re-write our format to RDF/RDFS/OWL and rule MLs like SWRL[1].

References

[1] Takeda, H., Sakai, H., Nomaguchi, Y., Yoshioka, M., Shimomura, Y., Tomiyama, T.: Universal abduction studio – proposal of a design support envrionment 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] Takeda, H., Yoshioka, M., Tomiyama, T.: A general framework for modeling of synthesis – integration of theories of synthesis –. In: 13th International Conference on Engineering Design – ICED 01, Design Research – Theories, Methodologies, and Product Modelling, Glasgow (2001) 307–314

[3] Takeda, H., Tomiyama, T., Yoshikawa, H.: A Logical and Computerable framework fc 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).

[4] Takeda, H., Veerkamp, P., Tomiyama, T., Yoshikawa, H.: Modeling design processes, A Magazine, Vol. 11, No. 4, pp. 37–48 (1990).

[5] Hayashi, K., Takeda, H., Tomiyama, T., Yoshikawa, H.: Analysis and logic: formalization of design processes (the third report) -- modeling with circumscription an abduction --, The proceedings of the annual conference of the Japanese Society fc Precision Engineering, pp.7-8 (1989) (In Japanese).

[6] Coyne, R.: Logic Models of Design. Pitman Publishing, London (1988)

[7] Roozenburg, N., Eekels, J.: Product Design: Fundamentals and Methods. John Wiley · Sons, Chichester, MA. (1995)

[8] Peirce, C.: Collected Papers of Charles Sanders Peirce. Volume 5. Harvard Universit Press, Cambridge, MA (1935)

[9] Flach, P., Kakas, A., eds.: Abductive and Inductive Reasoning: Essays on their Relatic and Integration. Applied Logic Series. Kluwer Academic Press (2000)

[10] Schurz, G.: Models of abductive reasoning. In Schurz, G., Werning, M., eds.: TP Preprints Annual. Number 1 in Philosophical Prepublication Series of the Chair · Theoretical Philosophy. The University of Dusseldorf (2002)

[11] Aliseda, A.: Abduction as epistemic change: A peircean model in artificial intelligence. I Flach, P., Kakas, A., eds.: Abductive and Inductive Reasoning: Essays on their Relatic and Integration. Applied Logic Series. Kluwer Academic Press (2000)

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

[13] Tomiyama, T.: From general design theory to knowledge-intensive engineering. Artifici Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM) 8 (199· 319–333

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

[15] Yoshioka, M., Umeda, Y., Takeda, H., Shimomura, Y., Nomaguchi, Y., Tomiyama, T Physical concept ontology for the knowledge intensive engineering framewor (Advanced Engineering Informatics) (Accepted for publication).

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

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

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