Intelligent CAD, II
H. Yoshikawa, T. Holden (Editors)
Elsevier Science Publishers B.V. (North-Holland)

Logical Formalization of Design Processes for Intelligent CAD Systems

Hideaki Takeda *
Tetsuo Tomiyama †
Hiroyuki Yoshikawa ‡

Department of Precision Machinery Engineering
Faculty of Engineering
The University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113, JAPAN
Internet: takeda@pe.u-tokyo.ac.jp
Tel: 03-812-2111 ext. 6481
Telex: 272-2111 FEUT J
Fax: 03-812-8849

In this paper, we first describe the *design experiment,* an experimental method used to examine design processes. The results of design experiments were analyzed in two different approaches. One was to outline transitions of object descriptions and the designer's viewpoint. The other was to extract a *design cycle* which was a unit design process, and we found that design processes are composed of such design cycles. Secondly, we show a framework for logical formalization of design processes which is expected to serve as a basis for developing a design knowledge representation language for intelligent CAD systems. We introduce deductive formalization based on two kinds of non-standard logic, viz. modal logic and non-monotonic logic. This framework explains experimental data obtained in the design experiment and is considered appropriate to represent design processes.

1. Introduction

Recent development of the concept of intelligent CAD systems focuses on design knowledge representation that can be decomposed into two major issues yet to be solved. One is the representation of design objects, and the other is the representation of design processes [1]. We believe that the realization of intelligent CAD systems depends totally on these two issues. Advances in computer graphics and geometric modeling contributed to achievements in the representation of design objects. However, difficulties are found in the representation of design processes, because design is considered one of the most intelligent and complex human activities, and design processes reflect this fact.

In order to support designers with full knowledge about the intent and context of design, intelligent CAD systems are required to explicitly represent design processes. There are many approaches to representing design processes, but merely a *descriptive cognitive theory* might not be sufficient: We need to develop a *computational* (or *computable*) *theory* of design processes [2, 3]. Computational theories are not established without scientific observation of phenomena and a firm, sound mathematical foundation. As a first step towards such a theory, we need to have well-formed representations. For this purpose we lay the foundation on logic.

* Graduate Student, † Associate Professor, ‡ Professor

The remainder of this paper is organized as follows. Chapter 2 discusses an empirical method to analyze design processes. We made *design experiments* in which designers were asked to perform a design and the whole session was recorded on video tape. We analyzed protocols and extracted an empirical design process model that could be categorized as a descriptive cognitive theory of design. Chapter 3 proposes a method for logical formalization of design processes. This logical formalization employs two types of non-standard logic; viz. modal logic to represent necessity and possibility and non-monotonic logic to represent defaults and assumptions. The experimental data can be described in this framework, thereby, it is appropriate to represent design processes. Chapter 4 concludes the paper.

2. Design Experiment

2.1. A Method to Examine Design Processes

Design is an iterative process. The designer modifies and adds detail to design object using his design expertise (i.e. his experience of past designs, his engineering know-how, established design procedures). During the initial stages of the design the descriptions of design objects are poor and incomplete because they consist of only specifications of the design and the specifications themselves may be uncertain, incomplete and conflicting. As the designers perform the iterative design process, the uncertainty and incompleteness are progressively removed and the problem converges on a set of solutions.

There are some proposals to describe design generally and most of them can be classified as traditional design methodologies (for example see [4, 5]). But they are not completely successful because they have little generality, i.e., they cannot be adapted to various types of design in different domains. Although one reason of this is that design is too various and diverse and that it is not an easy task to discuss design generally, there is another reason i.e., traditional design methodologies are based on the researcher's experiences and textbook knowledge of the domain. Therefore they confront difficulty when they are applied to other domains.

Here, we try to establish a logical framework to represent design processes more generally and formally. It is clear that this cannot be achieved without scientifically extracting information about design processes. But it is difficult to perform this by simple observation of design activities, because they are diverse, personal, complex activities. One useful method to perform this is the experiment. Various kinds of scientific knowledge have been obtained by the experimental way. We have suggested an experimental approach to investigate design and called it *design experiment* [6, 7]. In the experimental way we can prepare desired conditions to reduce the difficulty of the observation.

2.2. Design Experiment

A typical *design experiment* is carried out as follows: We present designers with a design problem and ask them to talk about everything they think about while designing. This is the method which extracts human thought and it is called *protocol analysis* in psychology [8]. We use a video tape recorder to record their conversations and their actions such as pointing a part of figures. Furthermore we make copies of the figures, drawings and sketches once in every few minutes. Thus we obtain various types of information including gestures, conversations and drawings, and we arrange them to obtain protocol data. We have carried out several experiments in this way so far.

The design problem we provided in one experiment was to design a part of the conveying mechanism for an automatic vending machine of cigarettes. The carrier was a motor-driven mechanism to take out one cigarette packet at a time from the stack. The purpose of this experiment is to extract the flow of designers' thought and thy way they proceed their design. For this purpose we choose the design problem on the following conditions;

(1) The function of the machine must be specified clearly.

(2) The machine must be realizable.

(3) The mechanism of the machine is not known to the designers.

(4) Conceptual design must be included in the design process.

(5) The design process must be finished in several hours (i.e., in a day).

We provided the first two conditions, because designers should accomplish their tasks successfully. Conditions (3) and (4) are needed to avoid routine or regular design processes, because We think that designers' thought should not always appear explicitly in routine design processes. We must avoid long suspensions, because designers may think about the problem during the break and we cannot know the thought. This is one reason for (5), and the other reason is that too long protocol is difficult to analyze.

We made three pairs of designers composed of an engineer and five students and we presented each pair with the same problem (see Table 1). It took three to four hours for each pair to complete the design. The average protocol data counts approximately 500 statements and fifty meaningful chunks of drawings. Figure 1 shows an example of the protocol data which is the beginning of the design process performed in experiment No. 2.

Table 1: Designers of the Design Experiments

Experiment	Designers
No. 1	Two students
No. 2	Two students
No. 3	An engineer and a student

2.3. Results of the Design Experiments

The protocol data was analyzed in two ways. In the first one, we focused on the transitions of object descriptions and the designer's viewpoint. We picked up statements and pictures in which attributes of objects were mentioned explicitly and we constructed a description of the design object at a moment. From this, we obtained a series of object descriptions that illustrated the evolutionary transitions of object descriptions.

We extracted all the transitions from the design processes. Figure 2 depicts a partial example of transitions of object descriptions. This is a part of the design process in which the mechanism to push out the package was considered. This design process begins in the middle of the protocol data in Figure 1.

A description at a certain moment consists of all the attributes already determined and is represented by boxes connected by full lines in Figure 2. When a statement or a picture that shows a new attribute appears, it is combined with the old attributes and the next description of

"The problem is to decide how to take a package of cigarettes out."
"It is not necessary to pick up a package with a hand-like mechanism."
"How about pushing it out with something like a forklift."
"If the storage part is similar to the figure of the specifications, it is sufficient to push it out one by one."
"If it only has to drop a package out of the storage part, that is a good idea."
"If you don't like dropping, it can be pushed out and then carried by something like a conveyor belt."
"If the storage part is like in the figure, it is impossible to push out like in Figure A(a)."
"It can be taken out from the bottom like Figure A(b)."
"D parameter in Figure A(b) must be greater than the thickness of the package."
"OK. Then, let's think about a mechanism pushing it out." [A]
"How about a rubber conveyor belt (Figure B)?" [S]
"Packages can be pushed out with the friction between packages and the rubber belt. But it is better to attach pins on the belt to be sure to push them out." [D]
"It depends on the speed of the belt whether the next package is set on the right position or not." [E]
"It looks like this (Figure C). It returns to the stand-by state as soon as it pushes out a package."
"We can use a mechanism like this (Figure D)."
"I don't think it is suitable for the purpose."
"How about a crank mechanism?"
"How about a cam mechanism?"

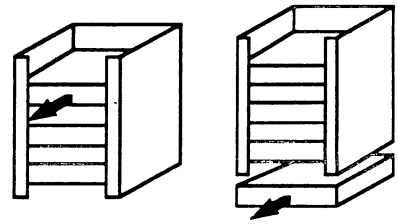
Figure A(a)

Figure A(b)

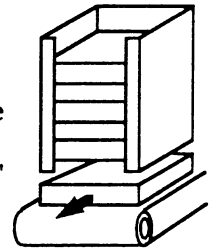
Figure B

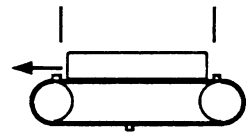
Figure C

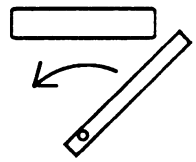
Figure D

Figure 1: Examples of the Protocol Data (Originally Spoken in Japanese)

the object is created. A white arrow indicates this adding operation of the new descriptions to the old one, while a black arrow shows a transition of the designer's viewpoint from one description to another. The designer proposes alternative candidates for one subproblem and this situation corresponds to the set of alternative solutions in the figure.

Some observations can be made.

(i) A design process is not linear but has many branches.

(ii) There are two transitions, viz. evolutionary transitions of object descriptions and transitions of the designer's viewpoint, and they are closely related to each other.

(iii) Having alternative solutions allows the designer to change his viewpoint from one to another, when he is confronted with difficulties. The designer's thought has such mutability.

In the second way of the analysis, a design process is viewed as a mental process, i.e., as a decision-making process. The designer decides to modify his descriptions of the object and this decision-making process is denoted by the transition from one object description to another, i.e., a white arrow in Figure 2. We extracted five subprocesses that are rough breakdowns of these decision-making processes and include the following:

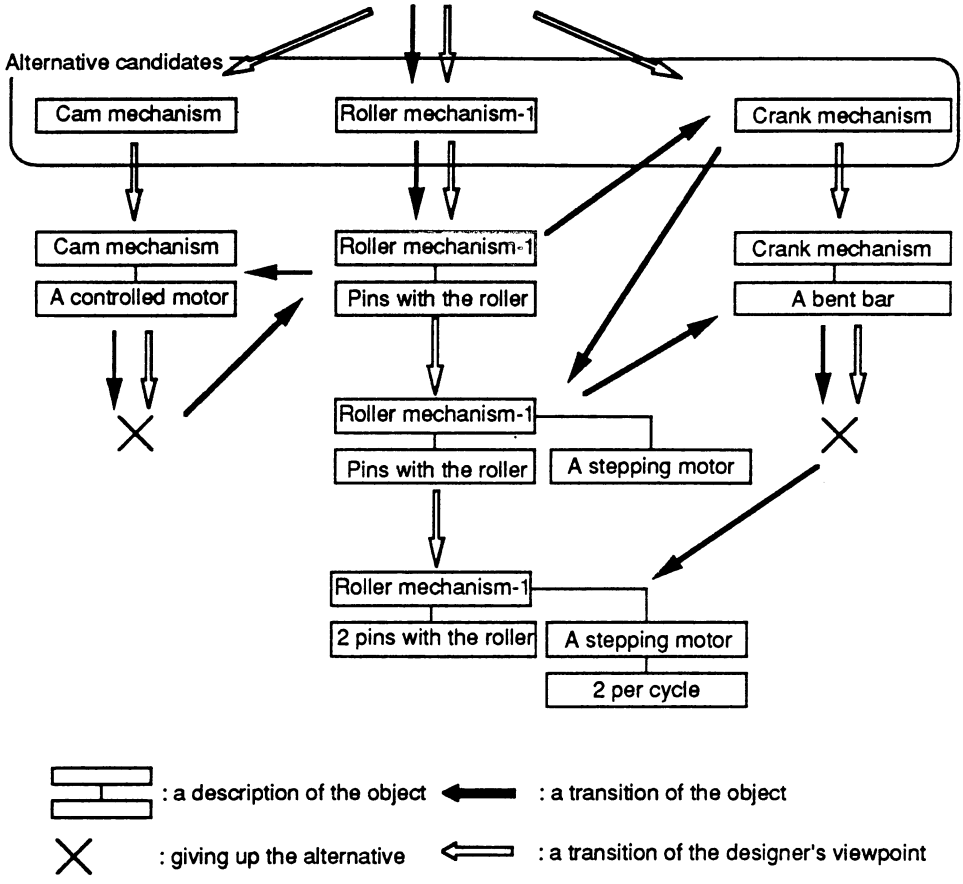

Figure 2: Examples of Transitions of Object Descriptions and the Designer's Viewpoint

(1) *Awareness of the problem:* To pick up a problem by observing the object and the specification and to determine this problem to be solved next.

(2) *Suggestion:* To suggest key concepts needed to solve the problem.

(3) *Development:* To construct candidates for the problem from the key concepts using various types of design knowledge.

(4) *Evaluation:* To evaluate the candidates in various ways, such as structural computation, simulation of behavior, cost evaluation, etc.

(5) *Decision:* To decide which candidate to adopt so as to modify the descriptions of the object.

These five subprocesses repeatedly appear in design processes. Thus we call the sequence of these subprocesses a unit *design cycle,* and we consider that a design process is built up by unit design cycles (see Figure 3). Figure 4 illustrates an example of this cycle obtained in the protocol data. We found it from the part of the protocol data shown in Figure 1. **[A]**, **[S]**, **[D]** and **[E]** in Figure 1 indicate the sentences used as *Awareness-of-problem, Suggestion, Development* and *Evaluation* subprocesses in this example respectively.

There are two types of connections between design cycles (See Figure 5). If a design cycle terminates successfully, the *conclusion* subprocess is followed by an *awareness-of-the-problem* subprocess of the next cycle. This connection forms a *sequence* of design cycles and often

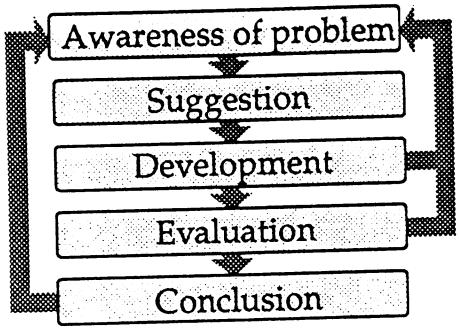

Figure 3: A Design Cycle

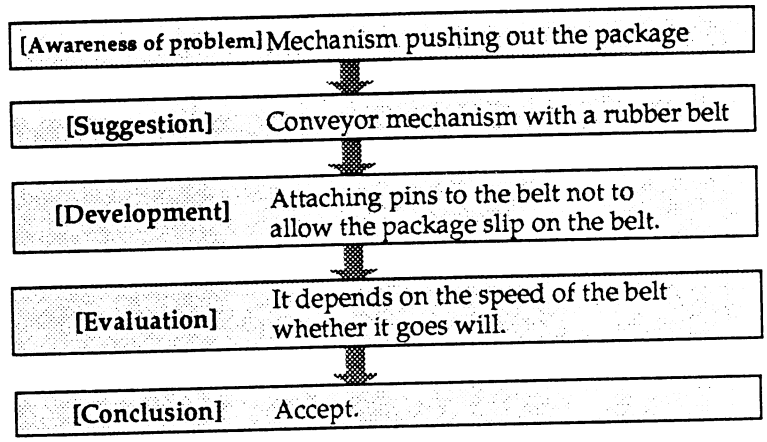

Figure 4: An Example of a Design Cycle

occurs in the routine design. When the designer wants to develop his design, he may notice the lack of important information, or while he evaluates his design object, he may find some problems. In these cases, the next step of the *development* or the *evaluation* subprocess is an *awareness-of-the-problem* subprocess of the next cycle. He must solve the new problems before he can come back to the previous problem. This connection composes a *hierarchy* structure of design cycles and often occurs in the conceptual design.

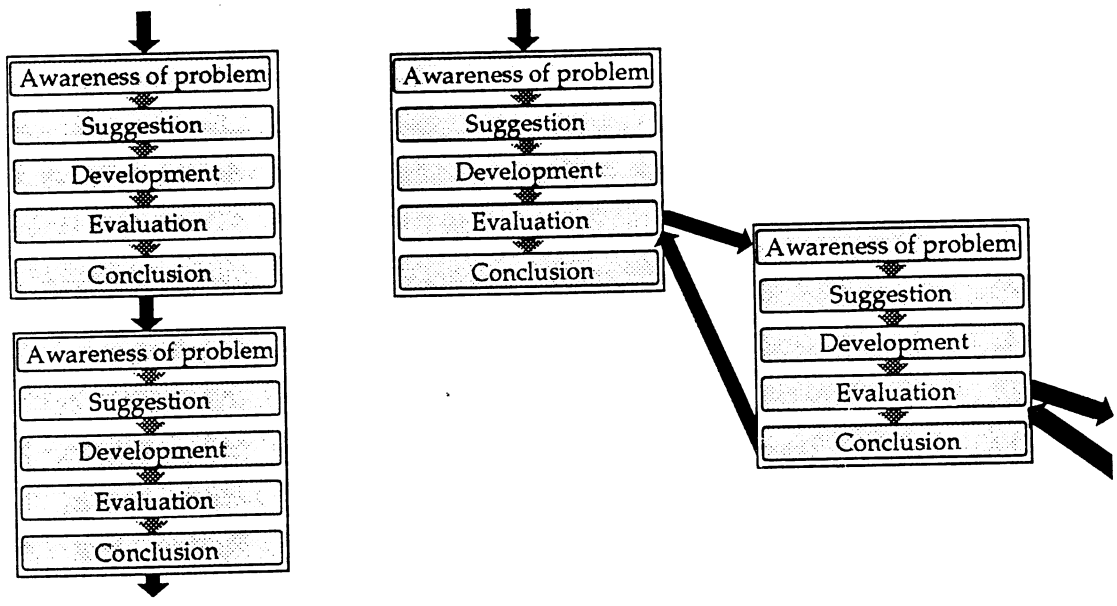

Figure 5: Two Types of Connections between Design Cycles

In this model, a design process is a set of design cycles, and a sequence of design cycles is related to the sequence of transitions of object descriptions shown in Figure 2.

Ullman *et al.* [9, 10] proposed a similar design process model by an empirical method. They used ten *design operators*, namely *select, create, simulate, calculate, compare, accept, reject, suspend, patch,* and *refine*, to represent design activities. These operators were classified into three types, i.e., *generation, evaluation,* and *decision*. Furthermore they introduced an *episode* as a sequence of several design operators to represent a certain activity, such as *verify* and *plan*. This might be a good way to empirically analyze and classify design activities, but they seem to be little interested in representing dynamically design processes as logical reasoning.

Thus, their design process model lacks a method to explore the mechanism of the reasoning in design processes, while we made our model in order to clarify and express it.

3. Logical Formalization of Design Processes

3.1. A Design Process as a Logical Process

In this section, we discuss how design processes can be logically formalized. We view a design process as a deductive process that can derive descriptions about the design object from the design knowledge and the specifications. This situation corresponds to logical deduction that derives theorems from axioms. Advantages of pursuing logical formalization are multifold. For instance, the results of this formalization can contribute to extracting language constructs to represent design knowledge that consists of knowledge about both design objects and processes.

When we look at a design process as changes of object descriptions, the results of design experiments make clear that design has three issues to be considered: The first is the *stepwise* nature of design processes, i.e., the design is performed repeatedly by similar processes. The second is *branching* in design processes. The designer may have some alternative solutions he will pursue later; they can be done either one after another in succession or randomly. The third is *retractability* of design processes. The designer may retract his previous decisions, because he often carries out his design process by the trial-and-error method.

Although these points are crucial, a simple deduction system is not capable of expressing them. Therefore, we introduce two non-standard logics to represent concepts that are particular to design: One is modal logic, and the other is non-monotonic logic.

3.2. Modal Logic

Modal logic can be viewed as a logic with necessity and possibility [11] and is defined as an extension of normal proposition or predicate logic. It has two modal symbols, i.e., **L** for necessity and **M** for possibility. In typical modal logic systems,

$$\mathbf{L}p \rightarrow p$$

and

$$p \rightarrow \mathbf{M}p$$

are always true because they are axioms or theorems. For example, suppose p is a sentence "The part A is connected to the part B." Then Mp represents "The part A must be connected to the part B." and p can be derived from it. Lp represents "The part A can be connected to the part B." and it can be derived from p.

Modal logic is interpreted in multi-worlds, while standard logic is interpreted in a single world. The fact **L***p* is valid in a certain world, if and only if *p* is valid in all the accessible worlds from that world. The fact **M***p* is valid in a certain world, if and only if *p* is valid in at least one accessible world from the original world. There are many different systems of modal logic in which the properties of accessibility are different, and we chose the S4 system in which accessibility is reflective and transitive. In the S4 system, the accessibility relation R must satisfy following two formula;

$$uRu$$

and

$$uRv \cap vRw \rightarrow uRw$$

where u, v and w are worlds.

We use this multi-world mechanism for representing design processes; a world represents one state of the design process and its accessibility, i.e., the connectivity of the worlds represents the flow of design processes. A formula in a certain world represents a description of the object at a certain moment. If a world is accessible from another, the description in the former world is more detailed than in the latter. Since we use the S4 system of modal logic, Lp in a world represents what is always necessary after the moment that the world was generated. For example, Lp in the initial world indicates what is always necessary through the entire design process, such as the required specifications, because it must be valid in all the worlds. On the other hand, for instance, Mp represents what is true in one or more worlds. Using such a modal logic system, we can represent designers' thought more naturally, because they seem to use modality such as "necessity" and "possibility", in their thought. For example, the designer may think "The length of this part must be 100mm," or "The part A can be replaced by the part B." These statements can be naturally formalized by the following logical formulae;

$$\mathbf{L}\mathit{equal}(\mathit{length}(p), 100), \qquad \mathbf{M}\{p(A) \rightarrow p(B)\}.$$

3.3. Non-Monotonic Logic

The second type of non-standard logic we use is non-monotonic logic (for example, see [12-15]). It was developed to express reasoning with incomplete knowledge such as human thought. What we believe at a certain moment is based on what we know at that moment. Therefore, we revise our belief, if our knowledge is changed. In the revision we may retract our belief, and such retractions cannot be expressed in standard logic. There are various types of non-monotonic logics. For example, Reiter's default logic [12] has a new symbol M and it is used like

$$\alpha : M\beta \,/\, \omega,$$

which reads "if there is α and it is consistent to assume β, then infer ω." This is useful to represent defaults and assumptions.

We express this non-monotonicity with a symbol A, which means a limited use of M. For example,

$$p \rightarrow \mathbf{A}q$$

means "if p is true and it is consistent to believe q, then q is true." We use a non-monotonic expression to express weak statements in design processes that may be retracted in later processes. Thus, statements in the *suggestion* subprocess, as well as temporary decisions, can be represented as non-monotonic expressions.

3.4. The Results of the Formalization

We translated the protocol data obtained in design experiments into the logical form discussed in the previous sections. Figure 6 shows some results that are extracted from the protocol data and formalized in the following way.

First, we extracted logical relations among the statements in the protocol data. We defined some facts as atom formulae appeared in the protocol data and changed the statements into the logical formulae. Suppose we have a statement "There is A" and a statement "if A then B". We consider the second statement as a logical implication from fact A to derive a new fact B, and we view this process as a deductive process. Then we introduce modal and non-monotonic symbols. When there are alternative candidates for the problem, we generate worlds and put statements about each candidate in one world. The notion of "(world n)" in Figure 6

identifies the name of the world in which the statement is contained. Statements independent from candidates are preceded by the **L** symbol, because these statements must be true in every world. Furthermore the **A** symbol are used for statements that the speaker made with uncertainty. Of course, there are some statements that cannot be viewed as deductive logical formulae. In Figure 6, we identify these statements in italic typeface. There are mainly three reasons why we cannot include them in the logical framework;

- They are not within the framework of first-order predicate logic.
- They are captured within the framework of deduction.
- They are not considered to be even within the framework of logic.

(world 0) *awareness-of-the-problem(mechanism-to-check-sold-out(main-body,?))*
  (world 1) **MA**mechanism-to-check-sold-out(main-body, photo-sensor)
(world 0) **LA**high(cost(photo-sensor))
  (world 2) **MA**mechanism-to-check-sold-out(main-body, spring)
  (world 2) mechanism-to-check-sold-out(main-body, spring) → be-stressed(package)
  (world 2) **M**be-stressed(package)
(world 0) **L**(weak(x) → ¬be-stressed(x))
(world 0) **L**weak(package)
(world 0) **L**(¬be-stressed(x))
  (world 2) ***M**be-stressed(package) and **L**(¬ be-stressed(x)) is contradiction*
  (world 2) **M**(¬mechanism-to-check-sold-out(main-body, spring))
(world 0) **L**(¬high(cost(photo-sensor)))
(world 0) *evaluation*
(world 0) *conclusion(mechanism-to-check-sold-out(main-body,?),?=photo-sensor))*
(world 0) **L**(mechanism-to-check-sold-out(main-body, photo-sensor))

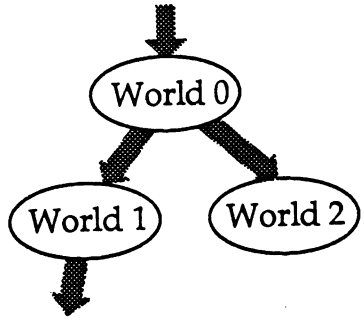

Figure 6: Examples of the Logical Representation

Though this example is just a draft of formulation, we verified the appropriateness of the formulation through translating the protocol data into this form and we can evaluate the results as follows;

(1) Most of the *development* subprocesses and certain numbers of the *evaluation* subprocesses in the empirical process model are represented as deduction. The non-italic statements in Figure 6 are included in the *development* and the *evaluation* subprocesses in the model. In these subprocesses new information is derived from the descriptions of the object and knowledge the designer has. There are differences in types of knowledge between these two subprocesses. For example, knowledge about how to propagate the alternation of the

size is used in the *development* subprocess and knowledge about how to estimate cost and life of the machine is used in the *evaluation* subprocesses. Because objects are detailed mainly in these subprocesses, deduction plays an important role in design processes. On the other hand, we cannot include the rest of the subprocesses, i.e., *awareness-of-the-problem*, *suggestion* and *conclusion* in the logical framework.

(2) Modal expressions are useful to represent stepwise, branching design processes and mutability of the designer's thought. The designer considers multiple candidates and furthermore multiple states of candidates at the same time. He sometimes focuses on one candidate and sometimes glances over. These situation can be expressed easily by using the multi-world mechanism in modal logic. For example, we generate a new world, when a new decision is made or when an alternative candidate is made. We change the current world, when he changes his viewpoint for instance.

(3) Backtracking and avoidance of inconsistency in design processes are similar to those in non-monotonic reasoning. When the designer meets an inconsistent situation, he tries a backtracking which causes the least modification, not chronological backtracking. He seems to check the dependency relations in his design process.

Some problems are left unsolved. A major problem is whether only deduction is appropriate to represent design processes. We assumed design processes were deductive, which turned out to be partly correct. However, there are some non-deductive processes in design, and deductive and non-deductive processes appear alternatively. Therefore, another inference mechanism is yet needed to capture the rest of the design processes. For instance, the *suggestion* subprocess can be viewed as a process to select a certain fact from design knowledge and as a process to produce new *axioms* from the known axioms. Such limitations of deduction must be investigated, and we must find out a most appropriate method (within the framework of logic) to represent design processes. As an answer to this, we are now trying to use abduction ([16]; also mentioned in [17]) and circumscription [18].

4. Conclusion

The purpose of this paper is to formalize design processes in order to develop a computational theory of design processes. First, we have presented the design experiment method which is an experimental method to obtain various useful information about design processes. From the results of design experiments, we were able to obtain characteristics of design processes, such as transitions of the designer's viewpoint and the evolutionary nature of design, and we proposed an empirical design process model.

Second, we suggested a framework for formalization of design processes based on deductive logic that includes modal logic and non-monotonic logic. We examined how empirical data could be represented in this framework, in which deduction plays a crucial role. We discussed that these two types of non-standard logic are useful to express the characteristics of design processes, which might be out of the scope of standard logic. Thus, the proposed framework is considered appropriate to logically represent design processes and this implies that the logical formalization will contribute to develop a design knowledge representation language for intelligent CAD systems. However, we also found that there are processes that cannot be represented in the framework and, thus, our next target is to extend our formalization to include such types of design processes.

Acknowledgements

We would like to thank Mr. Kazuto Hayashi and Mr. Hiroyuki Segawa, who participated the project of the design experiments and discussed with whom were stimulating in trying to formulate design processes.

References

1. ten Hagen, P. J. W. and Tomiyama, T., eds., *Intelligent CAD systems 1: Theoretical and Methodological Aspects*, Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1987.

2. Dixon, J. R., On research methodology towards a scientific theory of engineering design, *AIEDAM (Artificial Intelligence for Engineering Design, Analysis and Manufacturing) 1*, 3 (1987), 145-157.

3. Kalay, Y. E., ed., *Computability of Design*, John Wiley & Sons, New York, Chichester, Brisbane, Toronto, Singapore, 1987.

4. Hubka, V., *Theorie der Konstruktionsprozesse*, Springer-Verlag, Berlin, Heidelberg, New York, 1977.

5. Roth, K., *Konstruieren mit Konstruktionskatalogen*, Springer-Verlag, Berlin, Heidelberg, New York, 1982.

6. Yoshikawa, H., CAD Framework Guided by General Design Theory, in *CAD Systems Framework, Proceedings of IFIP Working Group 5.2*, Bo, K. and Lillehagen, E. M. (editor), North-Holland, Amsterdam, 1983, 241-253.

7. Yoshikawa, H., Arai, E. and Goto, T., Theory of Design Experiment, *Journal of JSPE 47*, 7 (1981), 830-835. (In Japanese).

8. Ericsson, K. A. and Simon, H. A., Verbal reports as data, *Psychological Review 87*, 3 (1980), 215-251.

9. Ullman, D. G., Dietterich, T. G. and Stauffer, L. A., A model of the mechanical design process based on empirical data: a summary, in *Artificial Intelligence in Engineering Design*, Gero, J. S. (editor), Elsevier, Amsterdam, Oxford, New York, Tokyo, 1988, 193-215.

10. Ullman, D. G., Stauffer, L. A. and Dietterich, T. G., Preliminary results on an experimental study of mechanical design, in *Proceedings of the NSF Workshop on Design Theory man Methodology*, 1987, 145-188.

11. Hughes, G. E. and Cresswell, M. J., *An Introduction to Modal Logic*, Hethuen, London, 1968.

12. Reiter, R., A logic for default reasoning, *Artificial Intelligence 13* (1980), 81-132.

13. McDermott, D. and Doyle, J., Non-monotonic logic I, *Artificial Intelligence 13* (1985), 41-72.

14. Poole, D., A logical framework for default reasoning, *Artificial Intelligence 36* (1988), 27-47.

15. Ginsberg, M. L., ed., *Readings in Nonmonotonic Reasoning*, Morgan Kaufmann Publishers, Inc., California, 1987.

16. Fann, K. T., *Peirce's Theory of Abduction*, Martinus Nijhoff, The hague, Holland, 1970.

17. Gero, J. S., Coyne, R. D., Radford, A. D. and Roseman, M. A., A unifying model of knowledge-based design processes, in *Preprint of the First IFIP W.G. 5.2 Workshop on Intelligent CAD*, 1987.

18. McCarthy, J., Circumscription - A form of non-monotonic reasoning, *Artificial Intelligence 13* (1980), 27-39.