Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

P.J. Veerkamp

Multiple worlds in an intelligent CAD system

Computer Science/Department of Interactive Systems Report CS-R9057 October

---

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

Multiple Worlds in an Intelligent CAD System

Paul Veerkamp
Department of Interactive Systems
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

In this paper we introduce three concepts: (i), a general design model based on stepwise refinement, (ii), a design object model evaluation scheme to interpret the design object model and to evaluate it, and (iii), the multi-world mechanism. The multi-world mechanism stems from (i) and (ii). It enables the designer to create worlds. Worlds are alternative descriptions of the design object model. These alternatives provide a way to regard the design object from different view points. We distinguish between dependent and independent worlds.

CR Categories and Subject Descriptors: D.3, F.4.1, J.6.
Key Words & Phrases: intelligent CAD, design theory, design process model, meta-model, multiple worlds.

1. Introduction

Nowadays computer aided design (CAD) is widely accepted as a useful tool for designers. However, the early CAD systems are actually nothing more than sophisticated drafting systems. Accordingly, there arose a growing need for CAD systems that adequately assist the designer in performing the design process. Since designing is an intellectual activity which involves a great deal of problem solving and reasoning, such a CAD system can only be effective if it understands the designers demands, i.e. if it has some intelligence. Therefore, we aim at developing an intelligent CAD system which incorporates knowledge about the design object and about the design process [6].

Using knowledge engineering techniques to create more sophisticated, more user-friendly and more 'intelligent' software products is quite popular. The number of expert systems is growing rapidly. However, the majority of those so called ''narrow domain expert systems'' do not contribute at all to solving the software crisis. On the contrary those rule based spaghetti-like systems are hard to maintain and difficult to extend. For this reason we choose for a more general approach and decided to develop a general framework on which an intelligent CAD system for a certain domain may be built. This framework is based upon research on the theory of the design object description, the design process, and design knowledge [1, 7].

The *IIICAD*[1] (Intelligent Integrated Interactive CAD) system is a general system which can be applied to any domain (e.g. architecture, mechanical engineering, VLSI design etc.). One of our major concerns was to develop a general framework which enables the designer to encode design knowledge and the design process in a flexible manner. We decided to spend the first

[1] The IIICAD project is supported by NFI project NF-51/62-514

year of research on theoretical aspects of CAD, theory of design, theory of knowledge and theory of design objects. One of the outcomes of this research is IDDL (Integrated Data Description Language). This language is used in the IIICAD system to describe the design process and the design knowledge. It is also used as a data (knowledge) representation language and as an interface language between the kernel of the system and various modules, such as the user interface, an application module etc. [9].

IDDL is based both on the object-oriented programming and on the logic programming paradigms. The object-oriented programming paradigm is used for the technical description of the artifact. In IDDL prototypes are used to denote the 'class' of an object. Note that IDDL differs in this sense from Smalltalk-80† [3, 4]. So, if we want to create a certain object, the most appropriate prototype is selected. The advantage of such an approach is that the newly created object can easily be modified because it does not belong to a certain class; it is an totally independent object. The prototypes are templates containing attributes and functions. The 'is-a' and 'part-of' hierarchy is specified in the logic part of IDDL and not in the object-oriented part. The logic programming paradigm is used to describe the knowledge about the design object and the design process. The former has been achieved by definite program clauses [5], the latter by so called if-then rules [2]. For a more detailed description of the part of IDDL concerning the design object description, see [9].

In this paper we will concentrate on design process. We will give a model according to which the design takes place. This *general design model* resulted in some specific language constructs. One of these constructs is the *multi-world* mechanism, enabling the designer to model the design process. The multi-world mechanism allows the designer to create alternatives. A world can be seen as the state of the design object model at a certain stage of the design process. Using the multi-world mechanism, the designer creates multiple worlds which are active at the same time. Concerning these multiple worlds we distinguish between *dependent* and *independent* worlds.

In this paper we will discuss the multi-world mechanism in detail and we will show how it evolved from the general design model. In the following section, we will present a general design model. In the third section a framework which evaluates the design object model will be shown (DOMES, Design Object Model Evolution Scheme), in the fourth section an explanation of the ideas behind the multi-world mechanism will be found and in the fifth section we will show how we actually incorporated the multi-world mechanism in IDDL using modal logic.

2. General Design Model

In developing intelligent CAD systems there are currently two approaches. The automated design system approach and the apprentice-like approach. The former is used to develop a CAD system which is able to design artifacts without human intervention and the latter to develop a CAD system that assists human designers in performing their design activities. We do not aim at a system that automates the design process. The IIICAD system is intended to be an apprentice for the human designer, rather than having it take control over the design process. However, a

† Smalltalk-80 is a trademark of Xerox Corp.

system can only assist a designer if it has knowledge about the actual design process. Therefore we need a theory concerning the design process. And from this theory we may derive a model according to which the design activities take place in order to understand the designers demands.

In other words the designer decides how to perform the design and the IIICAD system is an intelligent aid to achieve this goal. In this section we will give the model which guides the design process and we will show what kind of conceptual models should be provided in the system to assist the designer during the design process.

We use General Design Theory [8] as a basis to give a formalization of design processes and design knowledge. The theory is based on axiomatic set theory and models design as a mapping from the function space where the specifications are described in terms of functions, onto the attribute space where the design solutions are described in terms of attributes (see Fig. 1). Roughly speaking, one starts with a functional specification of the design object and ends with a manufacturable description.

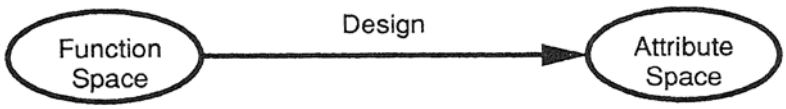

Fig. 1. Design process model

The basic ideas behind a logical formalization of design processes are as follows:

- From the given functional specifications a candidate is selected and refined in a stepwise manner until the solution is reached, rather than trying to get the solution directly from the specifications.
- Hence the design process can be regarded as an evolutionary process which transfers the model of the design object from one state to another.
- To evaluate the current state of the design object model, various interpretations of the design object model need to be derived in order to see whether the object satisfies the specifications or not. We call those interpretations of the design object model *worlds* and they can be regarded as interpretations of the design object observed from different view points.

Considering the general design model, the system starts from the specification S of the design object and continues the design process until the goal G is reached (see Fig. 2 below).

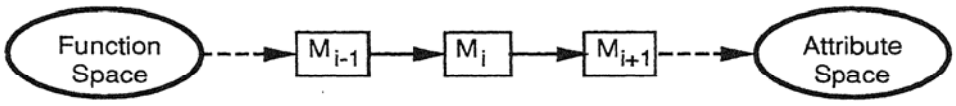

Fig. 2. Stepwise refinement of the meta-model

At a certain stage of the design process the design object model M_{i-1} is the current, incomplete description of the artifact. In order to get a more detailed description some

information is added to the design object model. After this refinement the design object model M_{i-1} is transferred to M_i if it is evaluated and approved. This process is continued, obtaining M_{i+1}, etc., until the design object model is a complete, satisfactory description of the artifact. How this refinement and evaluation process is performed will be shown in the next section.

The process as described above, deals with the ideal situation in which the stepwise refinement process is a linear process. It can be regarded as a sketch of the design process in retrospect. In practice, it is rather a process of trial and error, than such a straightforward process as shown in Fig. 2. During the design process an achieved subgoal might not be satisfactory and the designer might want to restart from a previous state of the design object model M_i. In that case, the current design object model M_j is discarded and the design process is continued from M_i taking a different direction (see Fig. 3).

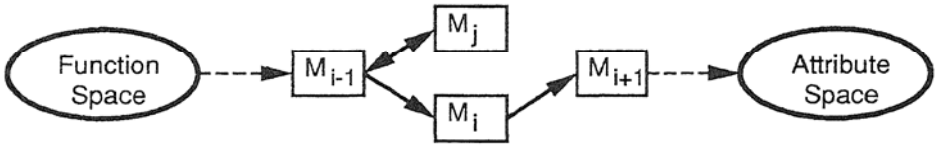

Fig. 3. Backtracking to a previous meta-model

From the above we can conclude that at a certain state M_i more than one possible way to model the design object existed. An alternative was chosen and refined further, resulting in an unsatisfactory solution. Instead of forcing a designer to take a decision at an early stage of the design process, the system should enable him to postpone such a decision and let him model more than one version of the design object simultaneously. At a later stage of the design process some of the alternative models may be discarded resulting in one solution. Or some alternative models may be merged into one design object description. In practice the design process model might look like Fig. 4 and Fig. 5.

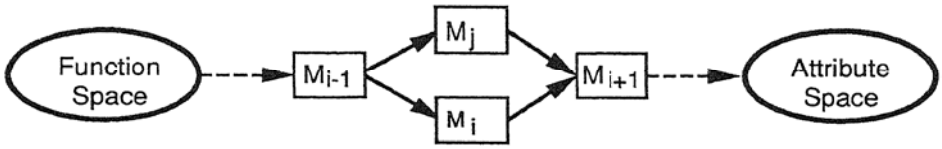

Fig. 4. Multiple views of the same meta-model

From this point of view the system should provide the designer with tools to model the design object in various ways. It must allow him to retract some of the obtained results and continue designing from a previous state. The designer might even want to restart from scratch, although the information obtained during the previous designing should remain (i.e. how **not** to do it). Concurrently modelling the design object following two distinct paths should also be possible. Again the decision in which way the design process should be directed is totally the responsibility of the designer. The IICAD system provides the designer with a framework which assists him in his design activities.

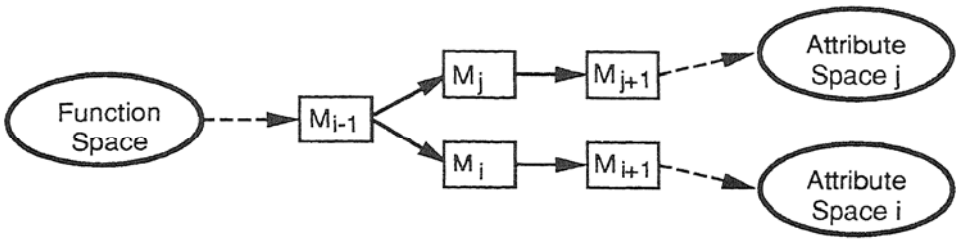

Fig. 5. Alternative meta-models

3. Design Object Model Evolution Scheme (DOMES)

The design process may be regarded as a constant manipulation of the design object model. A certain aspect of the artifact is highlighted and some properties about the design object is changed. This highlighting is done by worlds. A world is an interpretation of the design object model in a certain context. It is used to derive new properties or update uncertain or unknown properties about the design object model in order to get more detailed information. After the world being processed it is evaluated to check whether the newly derived information is consistent with the old design object model.

A world consists of (a part of) the design object description together with knowledge about the design object being valid in that particular world. A simple example of such a world is a graphical representation of the design object. A part of the design object description is taken and is processed in a world with graphical knowledge in order to generate an image of that part. Such a world needs to know how to map the general design object description to a description suitable for generating such an image. The designer can now interact with this world and change its contents. After the session the contents of the world being evaluated will be mapped back to a new design object description.

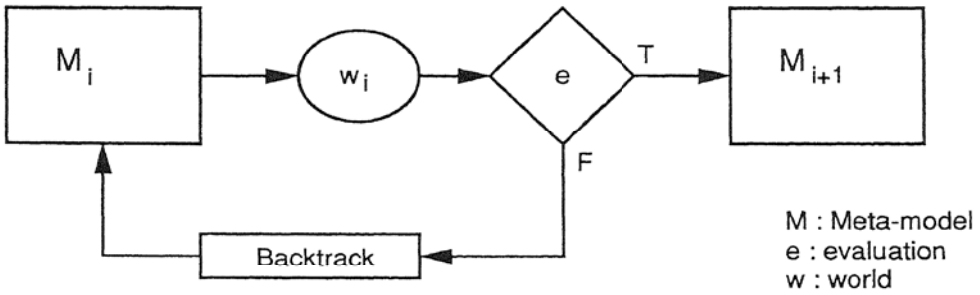

Fig. 6. Meta-model evolution

We call this mechanism the Design Object Model Evolution Scheme (DOMES, see Fig. 6). From the current design object model, M_i, a world, w^j, is taken and some action is performed in it. The world is evaluated after its termination. This evaluation checks the w^j on consistency with M_i, i.e. there are no facts that contradict each other and all constraints over the design object model are still met. The design object model M_i is transferred to M_{i+1}, if the evaluation succeeds. In case of failure, all results of w^j are discarded and the process will restart from M_i. This

backtracking is performed in dialogue with the users, so that the next attempt can be more successful.

As a conclusion we may say that the DOMES consists of the following sequence of steps:

1. A world on the design object model is created.
2. From this world new information about the design object is generated.
3. At a certain time the world terminates and the newly derived information is evaluated.
4. This either results in a new design object model, or it results in backtracking to a previous model.

There are two possibilities: either the current state of knowledge is complete and consistent or there is some incompleteness or inconsistency. In the first case the goal has been reached and we have finished the design process. In the latter case the design object model M_i needs to be processed further in order to resolve the incompleteness or inconsistency.

A world is a set of data, the current state of the design object model, which can be manipulated by the user. It can either be the full artifact description or just a part of it on which the attention is focused. By creating a world the user is actually modelling a copy instead of the design object model itself. As long as a world remains active it does not care about inconsistencies with the outside environment. The only concern is its own internal consistency. The implication is that the user can model a specific part of the artifact independently from the whole. When the world is being closed its contents is evaluated and if consistent merged with the design object description. When the world is inconsistent with the design object model, it is either being improved in dialogue with the user until consistency is achieved, or the world is rejected. In the former case the design process is continued with M_{i+1}. For the latter it is restarted from M_i.

This means that we need language constructs to evaluate a design object model by creating worlds and to derive new properties or to update uncertain or unknown properties in such a world in order to get more detailed knowledge about the design object. We define q_j^i as the set of propositions at state i of design object model M_i with an interpretation in world w^j. The crucial point is: how do we proceed from M_i to M_{i+1}? In other words, how do we define q_j^i and how do we derive new information from this world? We may define this kind of progress by:

$$M_{i+1} \equiv M_i \cup \text{eval}(w^j)$$

Multiple worlds may be active at the same time in order to interpret the current state of the design object in different ways. From this point of view it seems natural to choose modal logic as representation language, since modal logic deals with interpretations of a model in several worlds.

4. Multiple Worlds

The multi-world mechanism enables the system to create alternative descriptions of the design object model. This leads to two or more worlds being active at the same time. Concerning this mechanism we distinguish between two types of alternatives: *dependent* and *independent* worlds. The first option offers the possibility to regard the design object model from different view-points to model the design object in various ways. However, in this case the worlds refer to

the same design object model, and hence the worlds depend on each other. The second one allows the designer to model the design object in different directions by following distinct paths. In this case the worlds refer to different design object models and hence the worlds are independent. In the following two subsections we will expand on both concepts.

4.1. Dependent worlds.

We call two or more worlds dependent if they refer to the same design object description. The same set of data is concerned, but seen in a different context. They can be regarded as different views on the same model. The user can interact with these worlds separately. The dependency of the worlds comes into being when the worlds are closed. After the closure of dependent worlds these worlds are compared with each other on consistency. Note that all dependent worlds need to be closed at that time. If so, the contents of the worlds will be merged into one containing all the changes. This resulting world is then checked for consistency with the design object model, resulting in a next design object model M_{i+1} (see Fig. 7).

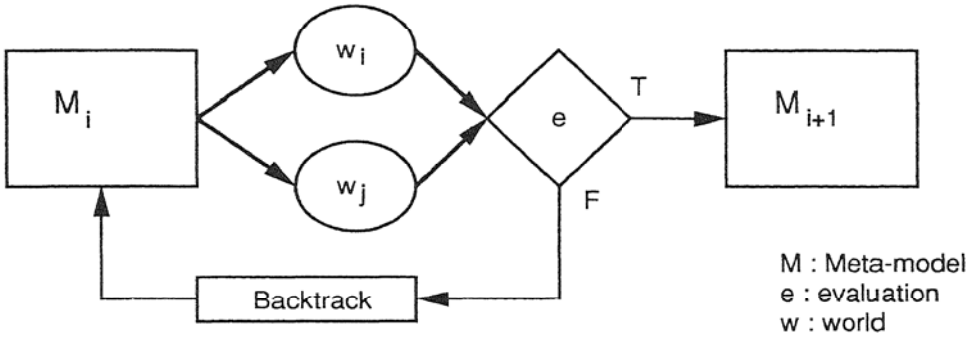

Fig. 7. Dependent worlds

Sometimes there is a need for a intermediate check on consistency. This can be done with an explicit update call from a world. When such a call occurs, all dependent worlds, including the one that called, are checked for consistency. A permanent change in one of the worlds is transferred to the others if appropriate. A change is appropriate if it fits in the context of a world.

In IDDL there exists a construct to generate dependent worlds. We call it the *and-mechanism*. With this and-mechanism the user is able to create multiple worlds on the same design object description to obtain an unambiguous result. To give an example: suppose a designer is designing an aeroplane. At a certain stage the designer wants to concentrate on the wing. By using the and-mechanism three worlds on the wing are generated; a graphical image, a finite element method analysis, and a wind tunnel simulator. These are different views on the same data from which new results are obtained. These results lead to new description of the wing which again leads to a more complete description of the aeroplane.

4.2. Independent worlds

Another mechanism, the *or-mechanism,* is used to generate independent worlds. Two worlds are called independent if they refer to different design object descriptions. Independent worlds allow the user to model multiple design solutions concurrently. The user can then compare those worlds and choose the best solution. After the closure of independent worlds each world is checked for consistency separately. This may result in multiple distinct design object models, say M_{i+1} and M'_{i+1} (see Fig. 8). Each of them can then be processed further as a candidate for the design solution.

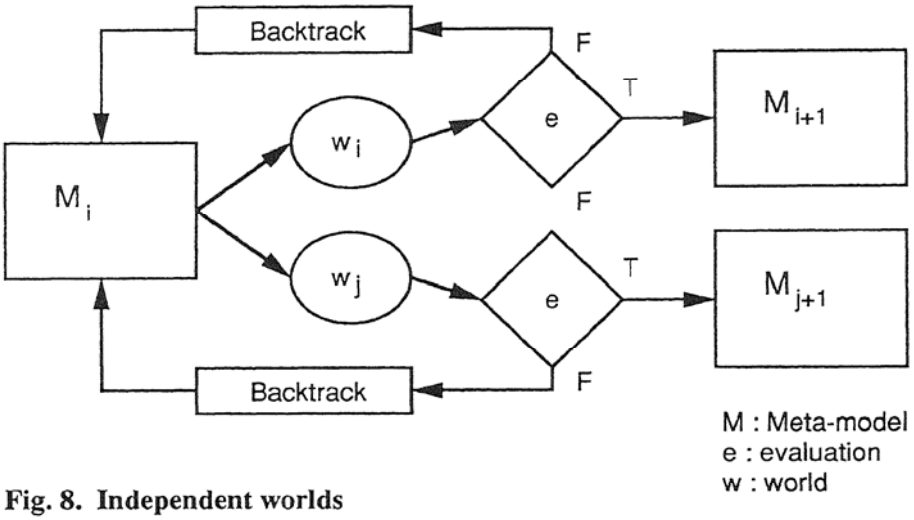

Fig. 8. Independent worlds

By using the or-mechanism the user is able to create multiple worlds on the design object description, that differ slightly in contents. To use the same example as mentioned above: suppose the designer is still designing the same aeroplane. In order to design a wing there are more possibilities for a certain parameter. The designer may now choose to generate multiple worlds by using the or-mechanism. Each of these worlds has a different value attached to this particular parameter. The worlds will all lead to a different result, and the designer may choose to continue with the most promising result. The other worlds may either be discarded or suspended. The designer might want to suspend a world when it is not sure whether or not the world that has been chosen will lead to the desired solution. The chosen world may be discarded after all and the suspended world may be favoured.

The or-mechanism differs from the and-mechanism in this sense that the or-mechanism may lead to more than one design object model being active at the same time. And hence a lot of extra administration has to be kept since more copies of the design object description has to be maintained. We are very well aware of the combinatorial explosion which might be caused by applying the or-mechanism. However, we consider it to be the responsibility of the user to use the or-mechanism with care. The decision to continue with multiple design object models in parallel is always taken by the user. There will be an intelligent user interface as part of the system, that allows the user to take such a decision in dialogue with the system.

5. Modal logic

In the above section we have introduced the multi-world mechanism. The language constructs which are used to control the mechanism are not yet shown, though. As stated before, all active worlds are independent entities during their existence. However, at certain points we want to evaluate different worlds and compare them. We have enriched IDDL with modal operators for facilitating the comparison of different worlds. We will show the basic notions that a system of modal logic is intended to express and show its application in IDDL afterwards.

Modal logic can be seen as the logic of *necessity* and *possibility*. Among true propositions we can distinguish between those which merely happen to be true and which are bound to be true. We call a proposition which is bound to be true a *known* proposition (Kp, it is known that p); a proposition that happens to be true is called an *believed* proposition (Bp, it is believed that p). In other words: a believed proposition is a proposition that is not known to be not true ($\neg K \neg P \equiv Bp$).

We have informally introduced the monadic proposition forming operators K and B. These operators are not truth functional, i.e. the truth value of the proposition cannot be deduced, not even when the truth-value of the argument is given. However there exist a strategy to determine the validity of a known or believed proposition. We shall not give the exact definition of this validity checking but show it informally. A known proposition, Kp, is valid in a certain world iff p is valid in all worlds *accessible* to that world. A believed proposition, Bp, is valid in a certain world iff p is valid in a certain world accessible to that world. Since we are using the system T, the accessibility relation encloses all the worlds that are active at that moment.

If we restrict ourselves to the internal state of a world, the modal operators do *not* influence the consistency of that particular world. Whether a proposition is a known or a believed proposition does not matter, it is always considered to be true within the world in which it appears. The modal operators are used when worlds are compared with each other.

We use a different operator M to express *default* values. Mp means p is consistent with the theory. A proposition is consistent if its negation cannot be derived. A default proposition, Mp, is considered to be valid if $\neg p$ cannot be found. With this mechanism we have the possibility to deal with non-monotonicity. During the design process some properties about the design object may not be known yet, so we can assume some default values. But as soon as some contradictory information is derived we remove the default property and assume the newly obtained information. On the other hand, when a proposition is derived, which is a tautology with a default proposition, the default proposition becomes a normal proposition. A proposition, which is derived from a default fact is asserted as an believed proposition.

The modal operators only appear in the design object model and not in the worlds derived from that model. They are used when the worlds are evaluated and checked for consistency. If several worlds are created by using the multi-world mechanism, these worlds only consist of non-modal propositions. Known propositions serve as so-called 'guard dogs'. These are the propositions which must hold at any time and any place. A known proposition must be true in all worlds being evaluated. A believed proposition is a proposition which may be true in one world but false in another. A world will never be discarded because of an believed proposition. If a believed proposition causes inconsistency, it is simply neglected, and will therefore not be

asserted to the design object model. Believed propositions can be seen as assumptions which are used as long as they seem to be true.

Another modal operator % is used to denote uncertainty. A proposition is *unknown* if neither its truth nor its falsity can be derived within the theory. An unknown proposition, %p, is considered to be valid if neither p nor ¬p can be found. Note that we now actually have introduced a third truth value (i.e. unknown). The reason we do not explicitly introduce a third truth value is that we want to keep our logic as simple as possible. So we have the open world assumption, but if we request p the knowledge base will return false if it finds ¬p or it cannot find p. Therefore, to be sure about the uncertainty of p one has to request %p explicitly.

6. Conclusions

Once we extend first order predicate logic with these operators we have a powerful tool to describe design knowledge in a flexible manner. Since the design object is constantly updated during the design process, we need to describe it in a dynamic way. IDDL is provided with a multi-world mechanism which is realised with modal logic. This mechanism enables the user to describe a design object seen from various viewpoints and to model several version of the design object concurrently. The modal logic can further be used to express default and uncertain information about a design object. The design process is an evolutionary process which modifies the design object description. In our view, IDDL equipped with modal logic enables the IIICAD system to describe design knowledge and to control the design process in an elegant and robust manner.

Acknowledgements

Among the other members of the IIICAD group I would especially like to thank Varol Akman, Paul ten Hagen and Tetsuo Tomiyama for their invaluable help in writing this paper.

References

1. V. Akman, P.J.W. ten Hagen, and T. Tomiyama, "Design as a Formal, Knowledge Engineering Activity," CWI-Report CS-R8744, September 1987.
2. R. Davis and J. King, "An Overview of Production Systems," in *Machine Intelligence 8*, ed. E.W. Elcock and D. Michie, pp. 300-332, Ellis Horwood Ltd., Chichester, 1977.
3. A. Goldberg and D. Robson, *Smalltalk-80: The Language and its Implementation,* Addison-Wesley, Reading, Mass., 1983.
4. A. Goldberg, *Smalltalk-80: The Interactive Programming Environment,* Addison-Wesley, Reading, Mass., 1984.
5. J. W. Lloyd, *Foundations of Logic Programming,* Second, Extended edition, Springer-Verlag, Berlin, 1987.
6. T. Tomiyama and P.J.W. ten Hagen, "The Concept of Intelligent Integrated Interactive CAD Systems," CWI-Report CS-R8717, April 1987.
7. T. Tomiyama and P.J.W. ten Hagen, "Representing Knowledge in two Distinct Descriptions: Extensional vs. Intensional," CWI-Report CS-R8728, June 1987.

8. T. Tomiyama and P.J.W. ten Hagen, "Organization of Design Knowledge in an Intelligent CAD Environment," CWI Report CS-R8720, Amsterdam, April 1987.

9. B. Veth, "An Integrated Data Description Language for Coding Design Knowledge," in *Intelligent CAD Systems 1: Theoretical and Methodological Aspects*, ed. Paul ten Hagen and Tetsuo Tomiyama, pp. 295-313, Springer-Verlag, Heidelberg, 1988.