Chapter 10

# Requirements for knowledge-based systems in design

**John Lansdown**

## 10.1 Introduction

Even from the comparatively small amount of work that has been done in this area it is already clear that expert systems can be of value in many architectural applications. This is particularly so in those applications involving what broadly can be called, 'classification' (such as fault diagnosis, testing for conformity with regulations and so on). What we want to look at in this chapter are some of the developments in knowledge-based systems (KBS) which will be needed in order to make them more useful in a broader application area and, especially, in creative design.

At the heart of these developments will be two things: (1), more appropriate methods of representing knowledge which are as accessible to humans as they are to computers; and (2), better ways of ensuring that this knowledge can be brought to bear exactly where and when it is needed. Knowledge engineers usually call these elements, respectively, 'knowledge representation' and 'control'.

## 10.2 Design as an information-processing concept

It is possible to view design from a large number of different viewpoints (indeed, one is tempted to believe that there are as many viewpoints as there are designers). The word 'design' itself is used with at least five different meanings. First, 'design' is often taken to be synonymous with 'designed artifact', that is, the object itself. 'Design' can mean the set of instructions prepared by the designer (such as drawings, specifications, manufacturing instructions and so on) which are used to construct the object. Sometimes when we speak of 'design' we mean the overt activities which designers go through in order to produce the artifact (such as discussions with the client, preparing sketches and finished drawings, making planning applications and so on). Again, 'design' occasionally means the covert inner mental processes that designers use to envisage the concept. Finally, 'design' can mean all of these rather different things.

In this chapter we will distinguish the covert design *process* from the overt design activity. By 'design' we will mean the sets of manufacturing instructions used to specify the designed artifact. Indeed, we will say that an object (or system) has been 'designed' if it has undergone a process which transforms an initial, partial and incomplete description of it to a final, full and complete one. This transformation is brought about by the application of information and knowledge. In this sense, designing is an information-processing task *(Figure 10.1)*. We apply knowledge and

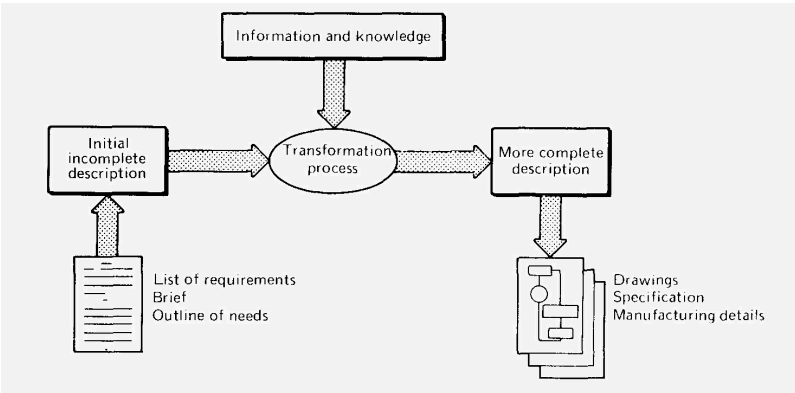


**Figure 10.1.**

information to an inadequate description in order to produce a design. The reader will note that, in this definition, designing is not seen as a problem-solving task. During both the activities and processes, problems certainly occur and have to be solved. But problem-solving is not the essence of designing any more than it is the essence of most other human activities. Problems arise: for example, when I am planning to go on holiday? Where should I go? What means of transport should I use? Will a visa be needed and how do I get one? How much foreign currency should be taken? These are problems and have to be solved but one would. not call planning a holiday a problem-solving activity, neither is the transformation process a continuous one. In a graph plotting the progress of a hypothetical design against time many discontinuities appear *(Figure 10.2)*. Especially in the early stages, new ideas arise which sometimes move us forward and, at other times, set us back. Furthermore, the initial set of clients' requirements, the brief, is modified as the developing proposals change both the clients' and the designers' perceptions of needs. Gradually (because throughout the process the clients and designers are gaining greater

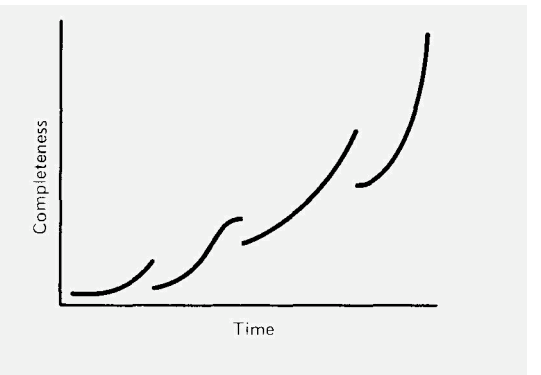


**Figure 10.2.**

insights into what has to be done) we move towards the complete description. Thus, creating a design is riot like solving a predefined arithmetical or logical puzzle. A design describes one possible set of proposals which properly fall within constraints, and we judge the quality of a designed artifact by how well and imaginatively its designers have responded to the limitations imposed by these constraints.

## 10.3 The logic of design

We can think of designing as partly a logical process arising from inductive and deductive reasoning and partly as an intuitive one arising from abduction (where something like guessing and testing is the norm). The exact balance between these two processes varies from task to task and designer to designer, but it would be rare to find anyone who believed that designing was solely logical, on the one hand, or solely intuitive, on the other. In both these processes knowledge is required. Indeed, one of the few things that one can say with certainty about creativity is that only those who know a great deal about a subject can be creative in it. In addition, this knowledge must be properly organized in order to allow it to be effectively brought to bear on the task at hand.

It is clear that expert designers have more effective mental representations of the tasks they are dealing with than have naive designers. They are also apparently able to generalize, specialize, formulate and reformulate their representations more readily. The approach they use seems to have two major constituents:

(1) A structural element which allows them quickly to identify pertinent factors and subproblems in the area of concern; and
(2) An inferential element assisting in the process of knowing, among other things, where, when and how to gain additional pertinent knowledge.

If we wish to model design processes and activities in computers these, too, must be equipped with similar constituents: properly represented and expert knowledge together with mechanisms for inductive, deductive and abductive reasoning.

## 10.4 Knowledge organization

It is not necessary here to go into the ways in which knowledge can be organized for the purpose of KBS. Here, too, we need good knowledge representations in company with proper control mechanisms to ensure that the knowledge is efficiently employed. Thus, a good KBS needs to incorporate not only explicit knowledge about its domain but also explicit knowledge about possible strategies of control.

The ways of representation that might be appropriate to design are introduced in outline in Lansdown (1982) and, for more general applications, in Barr and Feigenbaum (1981), Gevarter (1984) and Friedman

(1985). In summary, we can say that five generic types of representation are current: procedural, rule based, frame based, data based and object based. Each of these has inherent advantages and disadvantages and, despite the competing claims of the protagonists for each sort, no one representation is superior in all respects to the others. Strategies of control can also be represented in any of the five generic ways: indeed it is important that the similarities of representation are exploited. In addition, though, many normal programming language constructs are needed. These include sequencing, iteration, recursion and conditionals.

Much work has been carried out on rule-based representations. As pointed out in Lansdown (1982), these are powerful methods of encapsulating, in an explicit fashion, a good deal of the empirical, experiential knowledge that designers possess. In their usual manifestation, such representations allow us to express knowledge in an easilyunderstood IF - THEN format. For example,

IF the staircase rises more than 600 mm
AND the flight is more than 1000 mm wide
THEN at least one handrail is needed.

This type of representation has the technical name 'production-rule form' and is available in any situation which can be thought of in terms of substituting one set of symbols for another. Thus the staircase example can be interpreted to mean: if the symbol-string 'the staircase rises more than 600 mm' occurs with the symbol-string, 'the flight is more than 1000 mm wide', then we can substitute for these two a new symbol-string, 'at least one handrail is needed'.

Importantly, there is no restriction on the type of symbols that can be employed. Coyne and Gero (1984), extending the work of Mitchell (1979) and Stiny and Gips (1978), show that production rules can be used for the generation of building plans. (See also John Gero's contribution to this book, Chapter 9.) Lansdown (1970) illustrated the ways in which rulebased productions (together with frame-like elements) could be used for creating such diverse items as conversations, dances, theatrical swordfights and even custard-pie routines.

## 10.5 The problems of production systems

An important feature of production systems is that they assume that knowledge in the form of rules can be added incrementally to the KBS in a piecemeal fashion. This, in turn, requires that every rule expresses a truth which is independent of the truth of any other rule and, further, that no contradictions in the knowledge gained arise either when rules are assembled or when a chain of inferences is made from them. This is an assumption of *monotonicity*. It is valid for many domains, particularly when they are 'closed' in the sense that all there is to know about domain is already known.

This situation pertains in, say, diagnosing the cause of breakdowns in some piece of equipment. Here, all the possible causes and symptoms of

breakdown are already known. In such cases it is not too difficult to build up monotonic, uncontradictory rules - and even to maintain the consistency and truth of the system automatically by computer program. Doyle (1979), for example, describes one such truth-maintenance system.

Designing, however, is not one of these closed domains. Whilst designing, one cannot guarantee consistency and truthfulness between one stage of the transformation process and another. We must therefore seek reasoning methods which operate *non-monotonically* and allow new facts and inferences to arise whether they contradict existing 'facts' or not. Moore (1985) and Bobrow (1980) cover some of the implications of nonmonotonic logics.

What is needed in the early stages of designing is not so much truth maintenance as belief maintenance. It would be extremely inconvenient, to say the least, if whilst using a KBS to assist in the conceptual design of an artifact one was continually interrupted by the system's drawing attention to inconsistencies in the inevitably incomplete proposals. Indeed, it is appropriate in this regard to paraphrase the Scottish proverb: 'Fools, children and computers shouldn't judge things half-finished.' In the conceptual stages of design all designers tend to make assumptions about the artifact which are not borne out by current reality. Often designers put a few sketchy lines on a drawing to indicate some aspect of a proposal which they assume to be feasible though, as yet, not thought out.

What frequently distinguishes an experienced designer from a naive one is the confidence that can be placed on this assumption. Having a greater depth of resources on which to draw (a range of previous designs, a knowledge that the incompleted aspect resembles something previously designed and so on), experienced designers rarely fool themselves when they make these enabling assumptions. They use, essentially, a process of *deferral*. They defer consideration of some aspects while they concentrate on others but - and this is the important point - they temporarily believe that the deferred elements are already designed. Intelligent KBS should be able to accept similar assumptions and maintain a belief structure that allows progress to be made whilst still assisting designers to make broadly consistent decisions.

There are other problems, too. In the description of their system for placing rooms and other elements in building plans Coyne and Gero (1985) point out that rules about objects competing for placement tend to produce different layouts, depending on the order in which they are used. This is because such rule sets are highly interdependent. They distinguish between two types of rule sets which arise in design KBS: interacting sets (where the rule ordering is important) and commutative sets (where ordering is unimportant). However, as they stress, 'it happens that any production system can be reformulated as a commutative system. There is normally no advantage in such reformulation except that it provides a structure which will accommodate explicit control knowledge if it is available.' They go on to describe methods of dealing with such knowledge, in particular, using an artificial-intelligence technique which is now generally known as *planning*. Tate (1985) gives an informative review of various planning techniques.

## 10.6 Meta-rules

In any comprehensive design KBS the rule sets are likely to be so large, complex and interrelated that it is inevitable that inconsistencies will arise. Certainly, as we have to build up knowledge incrementally it is inconceivable for us to check each added item against all the others. Furthermore, at any given time a large number of rules and chunks of knowledge might be appropriate to a particular situation. Thus, as is made clear by Davis (1980), we will need special strategies to invoke the correct rules in these cases. In order that the same explicit mechanisms can be used both for knowledge representation and control, these strategies need to be in the form of meta-rules: rules about rules.

Davis goes on to explain a technique for utilizing meta-rules which he calls *content reference*. In this technique, the meta-rules contain general, high-level knowledge about the domain, the best strategy for dealing with given circumstances and the content of rules. Thus, in a room-placement KBS there may be a meta-rule:

> IF almost all the elements have been placed
> AND the arrangement is only partly symmetrical
> AND there are rules which say that symmetrical arrangements are preferred
> THEN it is unlikely (0.3) that these rules will be currently useful.

The implication of this technique is that the meta-rules must be able to consult the content of the rules. Future design KBS are likely to make use of planning methods as well as content reference.

## 10.7 IF-THEN-BECAUSE

In order to assist in this matter we have devised a minor development of the production-rule formalism which requires the justification of the rule to be incorporated in the rule itself. Thus the staircase production given previously might have an extra clause to make it read:

> IF the staircase rises more than 600 mm AND the flight is more than 1000 mm wide THEN at least one handrail is needed BECAUSE this is a mandatory requirements of the regulations (Clause 999.99) based on empirical safety requirements.

The objectives of these BECAUSE elements (which may contain multiple AND clauses), are three-fold:

(1) They help give more force to explanations arising from the use of the KBS, particularly when, as is this case, the reasons for the rule are essentially arbitrary.
(2) By requiring the human expert to give explicit justification for a rule as well as the rule itself, knowledge acquisition becomes more rational and checkable. Clarification of the reasoning behind the rule then becomes as important as the knowledge the rule contains. Thus better

distinctions can be made between objective and subjective knowledge; between those elements that arise from the laws of nature and those deriving from human laws; and between guesswork and more fundamental reasoning.

(3) The process of content reference becomes more powerful. Meta-rules can consult the BECAUSE part in the same way that people can, thus allowing more informed control of rule application. In addition, the automatic processes of generalization and specialization, essential to the development of intelligent techniques, have more information on which to work. This leaves the way open to better learning methods and, hence, to more refined knowledge.

## 10.8 Conclusions

We can conclude this brief review of design KBS by summarizing their essential requirements (not all of which have been explicitly referred to earlier):

(1) Design KBS (perhaps more than any others) require appropriate and multiple representation methods for both domain and control knowledge. Because of the special role drawing plays in designing, analogical and graphical representations must not be ignored,

(2) Truth-maintenance systems based on non-monotonic logics must be supplemented with design-belief systems.

(3) More efficient techniques of control must be developed, probably using planning and meta-rules methods. These are to help minimize combinatorial explosion and the sort of 'fan-out' problems that occur in situations where more and more rules apply at any given moment. The IF - THEN - BECAUSE formalism is suggested as one means of assisting in this by improving knowledge refinement.

(4) Designing requires both knowledge and access to vast quantities of information. Thus linking of design KBS to conventional databases and calculation procedures is essential.

Armed with such tools and techniques there is little doubt that designers will find future KBS powerful allies in improving the quality and performance of their products.

### References

BARR, A. and FEIGENBAUM, E. A. (eds) (1981). *The Handbook of Artificial Intelligence,* Los Altos: William Kaufman

BOBROW, D. G. (ed.) (1980). Special Issue on Non-monotonic Logic. *Artificial Intelligence (13),* 1,2

COYNE, R. and GERO, J. S. (1984). *Design Knowledge and Context,* Computer Applications Research Unit Working Paper, December

DAVIS, R. (1980). *Meta-rules: Reasoning about Control,* MIT Artificial Intelligence Laboratory AI Memo 576, March

DOYLE, J. (1979). A Truth Maintenance System. *Artificial Intelligence (12),* 231-272

FRIEDLAND, P. (ed.) (1985). Special Section on Architectures for Knowledge-Based Systems. CACM (28), 9, September

GEVARTER, W. B. (1984). *Artificial Intellience, Expert Systems, Computer Vision and Natural Language Processing,* Park Ridge: Noyes Publications

LANSDOWN, J. (1970). Computer Art for Theatrical Performance. *Proceedings ACM International Computer Symposium,* Bonn

LANSDOWN, J. (1982). *Expert Systems: Their Impact on the Construction Industry,* RIBA Conference Fund, London

MITCHELL, W. . (1979). Synthesis with Style. *Proceedings PArc 79 Conference.* Pinner: Online Publications

MOORE, R. C. (1985). Semantical Considerations on Non-Monotonic Logic. *Artificial Intelligence (25),* 1, 75-94

STINY, G. and GIPS, J. (1978). *Algorithmic Aesthetics,* Berkeley, CA: University of California Press

TAIT, A. (1985). A Review of Knowledge-based Planning Techniques. *The Knowledge Engineering Review (1),* 2, 4-17, June