A conceptual framework for knowledge-based design research at Sydney University's design computing unit

John S. Gero and Michael A. Rosenman

Design Computing Unit, University of Sydney, NSW 2006, Australia

This paper presents the conceptual framework behind the Design Computing Unit's knowledge-based design research. It commences with a brief overview before introducing the role of experience in design. The conceptual schema 'prototypes' is introduced and described within a framework of design as transforming required or expected functions to structure descriptions. Current projects related to this conceptual framework are briefly described.

THE DESIGN COMPUTING UNIT

The Design Computing Unit was established in 1967 under the name of the Computer Applications Research Unit within the Department of Architectural Science. It is a centre of research and teaching related to design and computing. It is concerned with 'design science' and focuses on computational models of design as a process. Its early work focused on simulation models of design[5,15,17]. This was then expanded to utilize optimization approaches and techniques derived from operations research to construct a goal-seeking paradigm of design[6,7,8,9,18,19,22,24]. The design paradigm based on optimization as the process culminated in three major publications. The first[10] was the result of the IFIP International Working Conference on Optimization in Computer-Aided Design. The second[11] was an edited selection of invited contributions. The third[23] is a textbook which distills many of the ideas based on this paradigm.

In 1981, members of the Unit began to extend their interest to design variables with non-numeric values and to the articulation and representation of the knowledge implicit in formulating design as a process. This led to work on knowledge modelling, knowledge processing and expert systems. Much of this work is reported in three major publications. The first two[12,13] are the results of two IFIP International Working Conferences; one on Knowledge Engineering in Computer-Aided Design and the other on Expert Systems in Computer-Aided Design. The third is a textbook[1] which brings together many of the ideas of design using these approaches.

Since 1987 there has been a growing interest in developing a conceptual framework for knowledge-based design activity. The remainder of this paper elaborates some of our results.

INTRODUCTION

Design is a creative human activity similar to language (oral or written), composition of music and art. Whereas in language the aim is to produce an arrangement of words to convey some meaning and in music the aim is to produce an arrangement of sounds in response to certain emotions, in design the aim is to produce a description of an artifact which will exhibit the necessary attributes to carry out a given function. Design is, thus, the process of providing form from a formless description. A formless description is given in function terms whereas form is given in structure terms. In design, as in other domains, knowledge is used to map between the meaning (semantics) and the configuration of the vocabulary (syntax).

The analytical processes involved in human creative activities are not known. We cannot simply deduce the 'correct design', or even any design, given a set of function requirements. Design is not a logical process in the sense that, given any function requirement, it follows inevitably that a certain structure description must be deduced. Thus, design is not merely a deductive process, although this is not to say that deductive analysis does not play a part in the design process. Given a particular design we can, given the appropriate functions or knowledge, analyse (deduce) the behaviour and the performance of this design with precision. However, given a required performance we cannot specifically deduce a required behaviour or a description. Similarly, in creative writing, given some theme, we cannot say what the composition should be. On a given topic, Shakespeare, say, would produce a particular composition, whereas Chaucer would produce another. We cannot say that one is more correct than another. Both are acceptable and, by some general consensus, we agree that both are of a high quality. The same holds in design, although some types of design may be evaluated more easily than others (e.g. does it work or not, does it sell). Design is an abductive process. In that sense, it is unlike science where the aim is to generalize, whereas the aim in design is to specifize. Design starts with the results required and uses available knowledge to arrive at a description of an artifact which will produce those results. Design does not pretend that the artifact so produced is unique in that sense. Many different artifacts can satisfy the given performance requirements. The number of such artifacts which may

© 1990 Computational Mechanics Publications

satisfy a given set of requirements depends upon the 'exactness' of those requirements. Planning is similar to design although the products are not objects in the sense of artifacts but sequences of actions.

THE ROLE OF EXPERIENCE

There is a leap in going from a set of performance requirements to an actual artifact which exhibits characteristics capable of satisfying those requirements. This 'creative' leap, using the word 'creative' in the sense that something is created, at present, defies analytical description. Nevertheless, it is a process carried out by designers. Schank[25] argues that humans are being creative whenever they make an utterance in response to some intended desire for communication. Hillier, Musgrove and O'Sullivan[20] and Darke[2] argue that designers begin designing by postulating some general concept, e.g. the 'Primary Generator', but do not specify from where this 'Primary Generator' comes.

Since it is not possible that new ideas and concepts are formed from absolutely nothing, it follows that there must exist some store of previous ideas and concepts from which to draw. Similarly, designers must have some prior knowledge of the objects and concepts which they will manipulate. In other words, designers, in common with other creative persons, such as writers, composers and artists, must have a good knowledge of the vocabulary with which they are dealing as well as of the knowledge on how to manipulate this vocabulary. A particular problem, then, triggers the relevant vocabulary elements and the knowledge necessary to compose these vocabulary elements into a coherent structure. Schank argues that all creative acts are based on finding the closest relevant experience and 'tweaking' it as necessary to fit the particular situation. This suggests that experience plays a major part in design.

Designers accumulate knowledge and experience over time. They thus accumulate a library of vocabulary elements within their domain (as well as more general knowledge) each such element having with it associations to other elements and to their relevance to various situations. However, more than just accumulating a number of separate instances of particular experiences, designers generalize their experiences into concepts. These concepts form a framework from which particular cases may be formed. This 'chunking' of knowledge provides an efficient storage medium enabling designers to use their experience in recalling cases or concepts which are relevant in providing solutions to the problem at hand. Designers tend to recall the most specific experience (or experiences) which is relevant. This experience may be general and may need to be made specific to the given problem. When no close experience is found then analogical reasoning may be used, that is relating the situation to some other situations where concepts may be found to be similar at some other level of abstraction. The mark of experts is their ability to efficiently arrive at a relevant area for solution.

The acquisition of experience – learning

Experiences have to be acquired and stored either as separate cases or generalizations have to be made and stored with associations to the cases as necessary. Dreyfus and Dreyfus[4] argue that humans store many individual cases and the work of Stanfill and Waltz[26] is based on this premise. However, this seems to suggest not only an enormous memory capacity but also a very efficient retrieval mechanism. Humans seem capable of abstracting features from similar experiences and structuring these into a framework of more general concepts. It is argued that individual cases are then stored with only those values which are specific to that case. In this way, information is only stored once.

Generalizations are formed by the process of induction. In that sense, generalization is akin to learning. Human beings generalize over even a single experience. The stronger the experience the better humans learn, either in the positive sense or in the negative sense to avoid repeating that experience. Learning is considered a measure of intelligence. The intelligence of species is related to how quickly they learn the lessons undertaken and apply the concepts to particular situations. In design, generalizations are made after each experience and concepts are formed or altered to suit. When faced with a design problem these concepts are brought to bear. For example, in transporting heavy objects, finding that the use of logs under a load makes it easier to move the load may lead to the generalization that providing rotating elements (e.g. wheels) under an object makes it easier to move it. This generalization may lead to the design of objects such as shopping cars or suitcases with wheels. A further abstraction may lead to the generalization that reducing friction between two elements makes it easier to move one element over the other. The application of this principle may lead us to the design of objects such as the hovercraft (Fig. 1).

THE ROLE OF ARTIFICIAL INTELLIGENCE IN DESIGN

We have been using computers for some time now. Computers are getting more and more powerful, yet they are still basically 'stupid'. They cannot do most of those things we regard as simple but, which in some way, are indicators of intelligence, e.g. pattern recognition or more than trivial examples, making subjective decisions, making associations, and understanding. We tend to regard solving a chess problem by exhaustively enumerating all possibilities as 'stupid' compared to an 'intelligent' method wherein only relevant moves are examined. But if the computer 'stupidly' solves these problems better than 'intelligent' humans – what then? Do we reevaluate our definition of intelligence? However, some tasks cannot be done by pure 'brute strength' as there will be too many possibilities and, in fact, most practical problems are of this kind. Even in the chess domain, deepening the search by looking further ahead can be self-defeating as this increases the number of positions to investigate, most of these positions being useless. Therefore, even with the ever increasing power and speed of computers, there will always be a point where intelligence is required to selectively use that power.

Humans do tasks that computers, as yet, can't, although this is the field that artificial intelligence is addressing. Most of these tasks are important for design. Some of these tasks are:

- humans make subjective judgements. Most of the important decisions that are made in the design

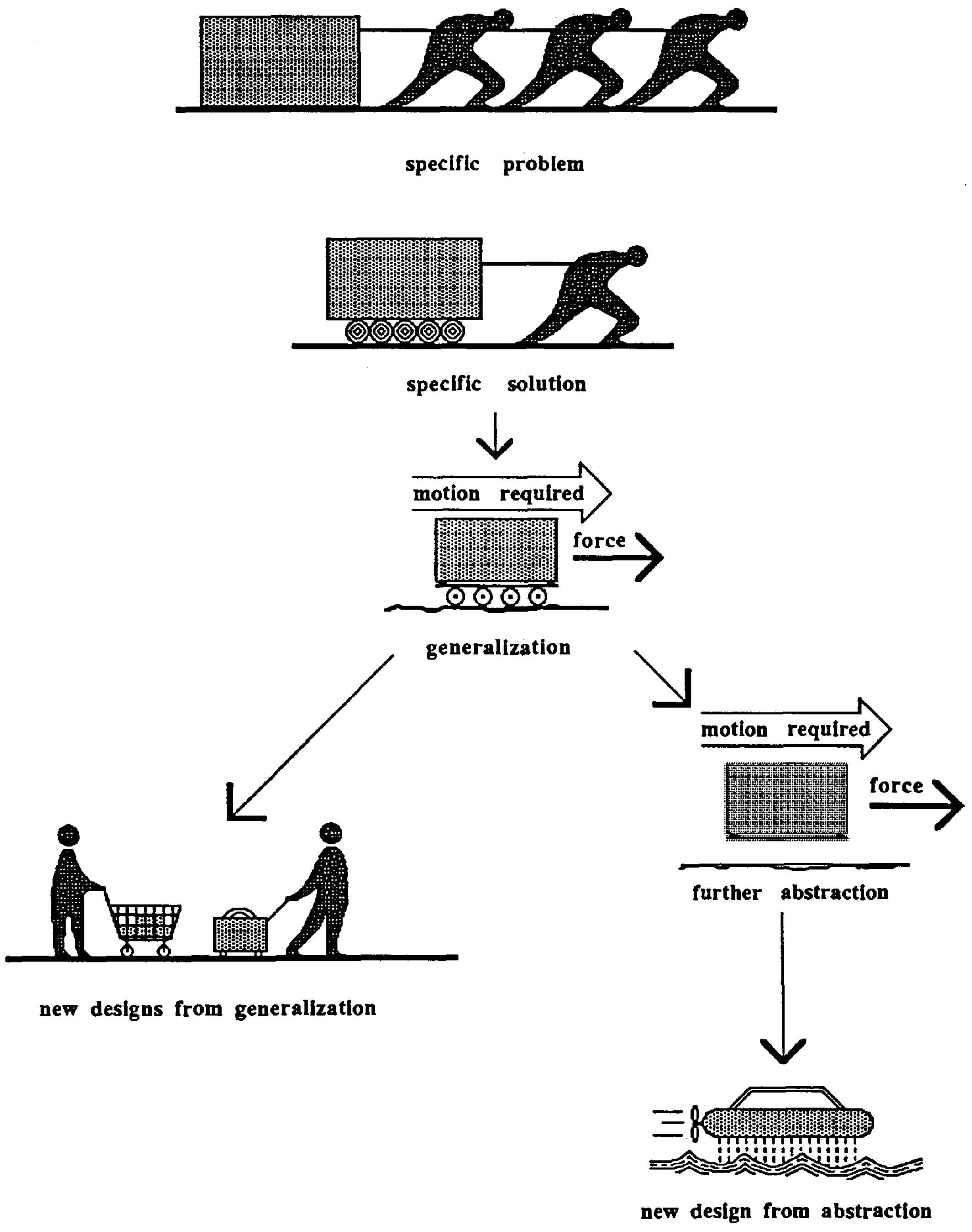


Fig. 1. Generalizing and specifizing

process are subjective. For example, selecting which criteria are the governing factors; ranking the criteria.

- humans trade brute strength for selective processing in which only relevant factors are recognized. For example, experienced chess players don't even see bad moves – experienced designers only 'see' relevant designs.
- humans have the ability to generalize by recognizing what factors are relevant and generalizing over these factors. Computers have to be told which factors are relevant.
- humans operate in world of uncertainty – 'beyond a reasonable doubt' (no such thing as certainty anyway and, therefore, we would be reduced to inactivity if we tried to operate within absolute certainty). This process of coming to conclusions on the basis of a 'reasonable' assumption leads to, overall, reasonable solutions to complex problems, if not, necessarily, the 'best' solution.
- the formation of a web of associations as needed. Obviously, not every combination of elements can be stored. The relevant associations seem to be formed as needed for a particular situation.
- humans have the ability to configure individual elements (words, forms, etc.) into an overall form depending on the context.
- humans seem to regard experiences as patterns rather than individual elements. They have a highly developed technique of pattern matching.

Computers, at present, are good for processes involving deductive logic in which they follow a predetermined inevitable path. Humans work abductively and inductively and, overall, achieve good results even if those results are not perfect. If we can make computers perform some of the above tasks, then we will go some way to make them more 'intelligent' and thus perform those tasks in which intelligence seems to be critical as in design.

THE DESIGN PROCESS

We have stated that design is the process of arriving at a structure description from a function description and that in order to carry out this process designers must have prior knowledge of the vocabulary of the structure in their domain. They must know about the behaviour of the structure properties, how they relate to functions and how configurations of these elements are generated and manipulated. To move from the function to the structure, designers have to carry out the following steps:

1. *Understand a problem and formulate its functions and behaviours.*
   This may involve decomposition of goals to operational objectives in the form of criteria or constraints. For example the problem stated as 'designing a portable shelter' may need to be decomposed into such objectives as

   It must be compact, lightweight, waterproof, easily erectable and demountable.

   or into more easily evaluable behaviours (often termed objectives) such as

   The packed volume must be less than V cm.
   The weight must be less than W gm.
   The average time for erection must be less than T min

   Not all problems can be foreseen at the start of a design, since humans are not omniscient. Some problems appear because of the selection of a particular vocabulary element. For example, the selection of timber may bring problems of termite protection which were not stated at the beginning of the design. Thus, the problem formulation will not be complete in every respect but need only be sufficient to begin a design process. However, as the design proceeds, any such new function requirements must be identified and satisfied.
2. *Arrive at a satisfactory structure vocabulary from which to select.*
   If we are designing an Art Deco piece of furniture then we must use that vocabulary.
3. *Select satisfactory structure vocabulary elements.*
   The selection must be based on expected behaviours.
4. *Configure structure vocabulary elements.*
   This assumes some knowledge on how vocabulary elements can be configured.
5. *Select among competing solutions.*
   Usually, there will be many ways to configure elements into some configuration and there is a need to select those which satisfy the required objectives. There may also be a need to select among those configurations which satisfy the objectives but to various degrees.

The knowledge of the elements involved and how to manipulate and configure them is syntactic knowledge. Using syntactic knowledge alone can lead to the configuration of a design, as typified by shape grammars[27]. However, the end result does not necessarily guarantee the satisfaction of the function requirements, although it does guarantee a 'legal design' as defined by the grammar. Interpretative knowledge is required to evaluate any such design as to its satisfaction of given requirements. This process becomes one of generate and test. However, where the solution space is large, a lot of fruitless work may be expended before a satisfactory design is found. A more fruitful approach would be to guide the development of the generation by introducing evaluation at each stage. This is the approach taken by Dawkins[3] in his work on cumulative selection. At each step of the generative process several possible alternatives are postulated and each part solution, generated so far, is evaluated to see if it either meets any of the given requirements or, alternatively, does not contravene any requirement. Since it is difficult to relate the performance of part solutions to overall required performances, selection may be difficult and many alternatives may have to be kept. This process of generating and testing at each stage of the generative process can be achieved only if vocabulary elements have information regarding their actual behaviours in various aspects.

In order to translate a set of function requirements into a structure or design description we need, usually, to follow the following process. First, the functions are expressed as the expected behaviours that a design would need to exhibit, e.g. transparency, economy, security, etc. These will have to have values which satisfy the given requirements. These values may be stated or derived from the context. For example, in designing a door to a building, the security required may be stated as being high or, alternatively, if the door is required to be an external door then this requirement will be derived from its given context. A set of behaviour variables can be determined, e.g. percentage of light transmitted, construction cost, maintenance cost, strength, etc., so that when these behaviour variables attain certain values, the expected behaviour will be met. For the behaviour values to attain the required values certain values of structure variables, also called design variables, must be chosen. The mapping between the behaviour variables and the structure variables is carried out using causal knowledge whereby the values of the behaviour variables can be derived from given values of the structure variables. To arrive at the desired structure variables we must compare the expected values of the behaviour variables with the actual variables of the behaviour variables derived from the particular values of the structure variables. Obviously, the behaviours that are relevant when evaluating a structure variable are dictated by the expected behaviours as derived from the function description.

While the aim of design is the production of a structure description, design may be classified into the following three broad categories:

1. *Routine design* – in which the design space of the functions, expected behaviours and structure variables is known and the problem is one of instantiating values for structure variables.
2. *Innovative design* – in which certain aspects of the defined design space need to be modified or extended,

as no existing solution within that space meets the requirements.

3. *Creative design* – in which the design space must be created. Note that while, in some sense, all design is creative, in that something which did not previously exist is created, the definition here applies to that process where the structure is unknown and the knowledge is scanty.

PROTOTYPES AS A CONCEPTUAL SCHEMA

Given that designers design from experience, we therefore need a system of storing this experience in a coherent structure. This structure must ensure that associations between experiences are kept so that the appropriate experiences are brought to bear as required. Such a system should allow for the three categories of design, described above, to be carried out. Moreover, the system must allow for design to be commenced no matter how detailed the given specification. The system, itself, should provide guidance as to what further information is required for the design to become more specific.

A system based on the notion of *prototypes*[14], as a generalization of groupings of design elements, provides a framework for storing and processing design experience. The prototype represents a class of elements from which instances of elements can be derived. It contains the necessary function and structure descriptions as well as behaviours and knowledge in a generic sense. Variables and methods are also provided. An instance is derived by inheriting any property, variable, and/or method from the generic prototype. A prototype may be related to other prototypes and an instance may need to inherit properties from instances of those prototypes. The system, therefore, constructs its own hierarchy.

Components of a prototype

A prototype needs to represent the function properties, structure properties, expected behaviours, the relationships to any other prototypes necessary and the knowledge required to find values for structure variables from the function description through behaviour. The function properties include the intended function, and the expected behaviours as attributes and variables. The structure properties include the vocabulary, the prototype description, configurational knowledge, as well as the actual behaviours as attributes and variables of the prototype. The vocabulary will include those elements that are essential to the existence of the prototype and those which are optional. The description will include typological properties as well as other attributes, such as dimensions, material, etc. Knowledge is required for every mapping from a property to another property. Knowledge is required to map from the behaviour attributes to the behaviour variables and to the description required. There will be constraints both on the function side and on the structure side. Constraints on the function side will generally be translated into requirements to be met, whereas constraints on the structure side will generally serve to prune the set of possibilities.

In addition to the elements described above, any design situation exists within a particular context. This context serves to define particular areas of interest. In some cases the context may merely define a set of function requirements whereas in other cases the context may define some, if not all, of the structure properties. For example, given that we want to design the engineering structure for a 50-storey high office building of square plan this will define a requirement with regards to wind loads and, in addition, may define the engineering structural system type and the material.

Figure 2 shows the model of a prototype schema consisting of function properties, behaviour properties and structure properties all existing within envelopes of knowledge and context. The function description is divided into function properties and behaviour properties where the function properties include the goal (or goals) and the requirements while the behaviour properties include the required (expected) and the actual behaviour attributes and variables. The goal or goals are the intended function of the prototype. The requirements are divided into those requirements which must always be met and those which may have to be met depending on the particular problem at hand. The behaviour properties form the core of this model. The expected behaviours are derived from the function properties required whereas the actual behaviours are derived from the structure description. However, the selection of the type of behaviour to be derived from the structure description is dictated by the expected behaviours. For example, given the structure description of a door, we would not expect to derive its aromatic properties since this is not a property which has any bearing on its function. It is in this behaviour core, where there are commensurate elements, which allows us to evaluate the suitability of a prototype for a given design situation.

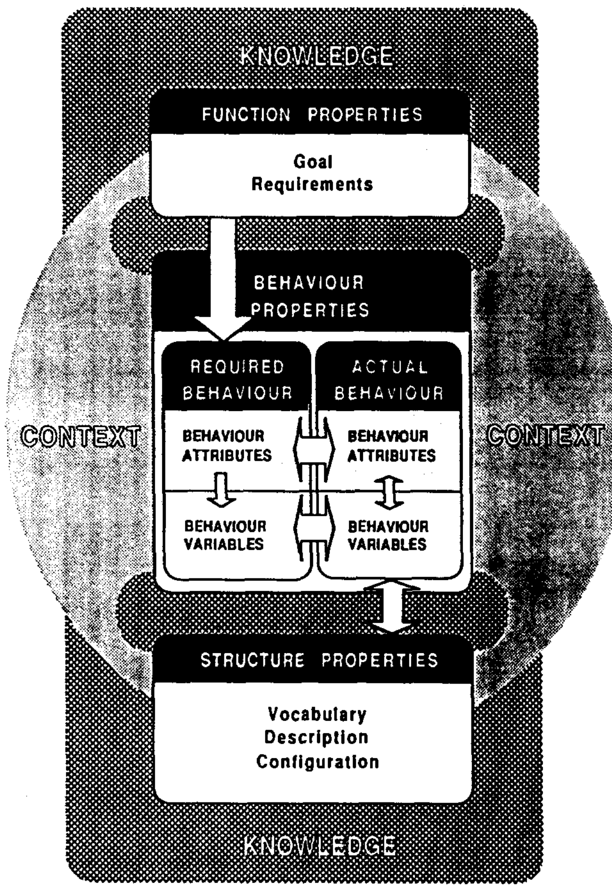


Fig. 2. Diagram of prototype schema

Parts and partitions

'Parts' within a prototype are vocabulary elements, prototypes in their own rights, forming part of a structure decomposition whereas 'partitions' form part of a conceptual decomposition, mainly for convenience and efficiency. Partitions embody only some of the attributes of the prototype but more than one partition of a prototype may have the same attributes. A prototype is described by one value (and one value only) from each of the chosen partitions. Prototypes may be partitioned in different ways according to the particular design's perceptions. For example, a house prototype may be partitioned according to the following partitions: massing type, cost type, style. As such a particular type of house could be described as a two-storey, low-cost, traditional house, whereas another house type could be described as a split-level, medium-cost, ranch style.

Without the partition concept each description would have to be a prototype itself, the number of prototypes becoming unmanageable. On the other hand treating each partition as a prototype in exactly the same way as any other prototype, but as subclasses of another prototype, means that there is no mechanism for stating that a prototype is described by a particular grouping of prototypes. The partition concept allows for efficiency in that is reduces redundancy in the description and allows for a full description to be configured as necessary.

Representation

The representation used in the prototype schema is framelike and uses, basically, the same notions of slots, facets, values and methods. However, the prototype 'frame' has a structured classification. Each prototype is divided into five main categories, namely: function properties, behaviour properties, structure properties, knowledge and context. The a_kind_of, a_partition_of and types classifications together with the vocabulary classification describe the typological and structure hierarchy within the prototype schema, allowing for the usual inheritance methods to be used. Other elements in the description category are the design variables whose values have to be found for the design to be carried out. The configuration may be given as a textual description, a description in a CAD system or as a set of generative rules.

Example of a prototype

For an example of a prototype schema see Figs 3, 4, 5 and 6. Figure 3 shows part of the prototype of the class of design elements called 'beam'; Figs 4 and 5 show two *partitions* of the beam prototype, partitioned along support and cross-sectional lines. Figure 6 shows an example of an I-Beam as one type of the Section_Type partition of the Beam prototype. It can be seen that a partition, in almost all respects, is the same as a prototype, the only difference being that it forms part of an overall prototype but is not in itself a complete prototype. Depending on the design context, it is not necessary to select values for every partition. For example, at an early design stage it may only be necessary to select that the type of beam be a cantilever without selecting values for the section type. Figure 6 shows that the actual behaviour attribute of the I-Beam has the value 'good for bending'. This value allows for the selection of the I-Beam section whenever the problem will state that efficiency in bending is a requirement. On the other hand, if efficiency in torsion is required then, since this is an attribute of an O-Beam, the O-Beam section would be selected. Similarly, at the prototype level, there needs to be some value that allows the Beam prototype to be selected rather than a Truss or a Cable prototype when the requirement is for transferring linear loads to supports.

A request to design a beam will cause the Beam prototype to be accessed. On the other hand, a request may be for a structure element to transfer a set of linear loads over a span to supports. In this case, beams, trusses and cables, for example, would be applicable. In order to differentiate between these, information as to attributes required is necessary. For example, the attributes required may be for a stiff, medium-weight and economical structure, thus causing the Beam prototype to be selected. If a further requirement states that torsional stiffness is required then the O-Beam prototype will be accessed. This requirement can come either from user specifications or from the system during the course of trying to instantiate a value for a design variable. Design requirements may also include constraints on the behaviour variables, e.g. the fire-resistance rating must be greater or equal to 2 hours and on the structure description, e.g. the material must be concrete.

In order to find values for the design or structure variables, knowledge is required. This can be in the form of rules and/or procedures or any other knowledge representation schema. Note that rule 'R3' in the Beam prototype contains a procedure called 'check_bending'. Obviously before this procedure can be executed fully some information will have to be known or found, namely the span, loads, the type of beam, material, etc.

INSTANCES

Prototypes are a representation schema for appropriate design concepts and knowledge. In this they are repositories of knowledge. There are two ways in which they can be used. In the first way, each prototype that is selected because it is applicable is copied to produce an instance and all the design processes occur on that instance[16]. Thus a particular design of some artefact constitutes one instance or a set of instances of one or more prototypes. This means that all of its structure variables have been given values. In some cases, not every variable needs to be instantiated. In the early stages of designing a door, for example, the door dimensions, opening type and direction of opening will be found without any interest in the type of construction or material. However, at the end of a design the prototypes remain unchanged.

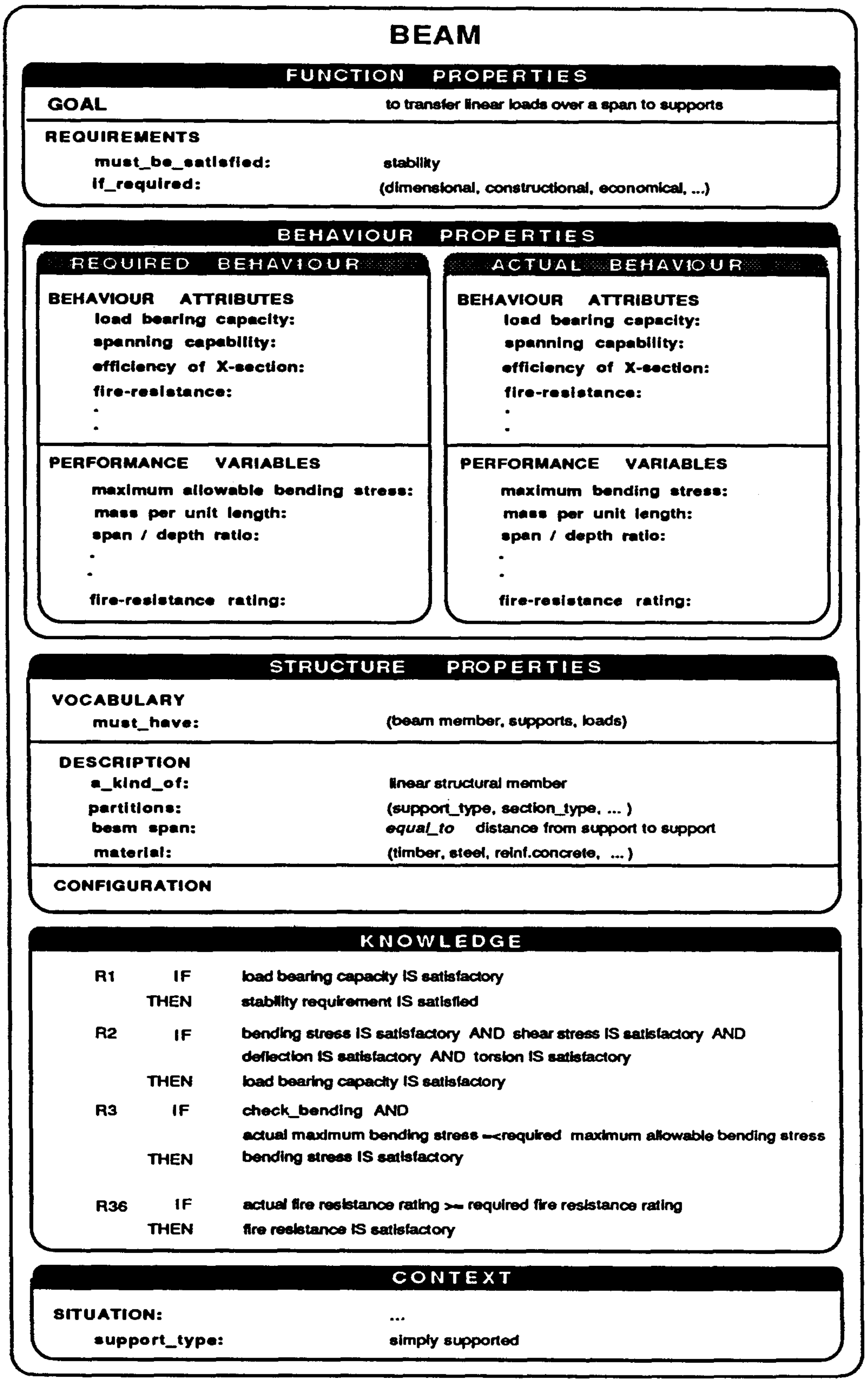
In the second way, the prototypes are the instances. Thus, each time design occurs we would expect them to change. This is a requirement for innovative and creative design[21].

CURRENT RELATED PROJECTS

Prototype-based routine design systems

Five aspects are under investigation and implementation. These are:

(i) representation of prototypes;
(ii) contents of prototypes;
(iii) design reasoning via instances;



BEAM

FUNCTION PROPERTIES

GOAL	to transfer linear loads over a span to supports
REQUIREMENTS	
must_be_satisfied:	stability
if_required:	(dimensional, constructional, economical, ...)

BEHAVIOUR PROPERTIES

REQUIRED BEHAVIOUR	ACTUAL BEHAVIOUR
BEHAVIOUR ATTRIBUTES	BEHAVIOUR ATTRIBUTES
load bearing capacity:	load bearing capacity:
spanning capability:	spanning capability:
efficiency of X-section:	efficiency of X-section:
fire-resistance:	fire-resistance:
.	.
.	.
PERFORMANCE VARIABLES	PERFORMANCE VARIABLES
maximum allowable bending stress:	maximum bending stress:
mass per unit length:	mass per unit length:
span / depth ratio:	span / depth ratio:
.	.
.	.
fire-resistance rating:	fire-resistance rating:

STRUCTURE PROPERTIES

VOCABULARY	
must_have:	(beam member, supports, loads)
DESCRIPTION	
a_kind_of:	linear structural member
partitions:	(support_type, section_type, ...)
beam span:	*equal_to* distance from support to support
material:	(timber, steel, reinf.concrete, ...)
CONFIGURATION	

KNOWLEDGE

R1	IF	load bearing capacity IS satisfactory
	THEN	stability requirement IS satisfied
R2	IF	bending stress IS satisfactory AND shear stress IS satisfactory AND deflection IS satisfactory AND torsion IS satisfactory
	THEN	load bearing capacity IS satisfactory
R3	IF	check_bending AND actual maximum bending stress =<required maximum allowable bending stress
	THEN	bending stress IS satisfactory
R36	IF	actual fire resistance rating >= required fire resistance rating
	THEN	fire resistance IS satisfactory

CONTEXT

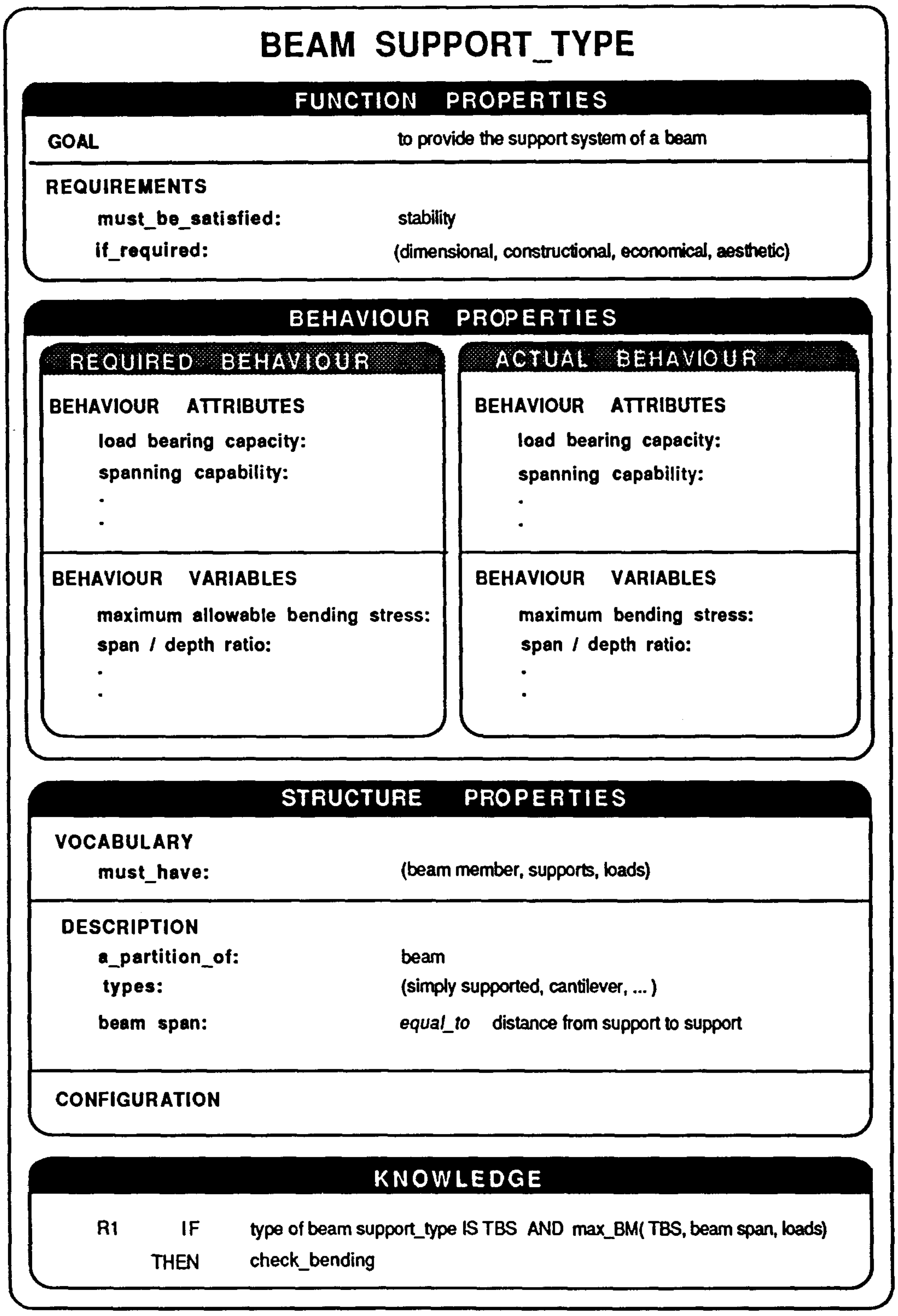
SITUATION:	...
support_type:	simply supported

Fig. 3. Example of a beam prototype

(iv) user interfaces; and
(v) cognitive models.

Representation of prototypes

Current systems represent prototypes using frames. In addition to the information contained in the example in Fig. 3, qualitative knowledge concerning relationships between function, behaviour and structure and between their variables is represented. Alternate representations based on graphs, dependency networks and associative networks are being investigated. Other research of interest in the Unit is concerned with neural networks representation.



BEAM SUPPORT_TYPE

FUNCTION PROPERTIES

GOAL	to provide the support system of a beam
REQUIREMENTS	
must_be_satisfied:	stability
if_required:	(dimensional, constructional, economical, aesthetic)

BEHAVIOUR PROPERTIES

REQUIRED BEHAVIOUR	ACTUAL BEHAVIOUR
BEHAVIOUR ATTRIBUTES	**BEHAVIOUR ATTRIBUTES**
load bearing capacity:	**load bearing capacity:**
spanning capability:	**spanning capability:**
.	.
.	.
BEHAVIOUR VARIABLES	**BEHAVIOUR VARIABLES**
maximum allowable bending stress:	**maximum bending stress:**
span / depth ratio:	**span / depth ratio:**
.	.
.	.

STRUCTURE PROPERTIES

VOCABULARY	
must_have:	(beam member, supports, loads)
DESCRIPTION	
a_partition_of:	beam
types:	(simply supported, cantilever, ...)
beam span:	*equal_to* distance from support to support
CONFIGURATION	

KNOWLEDGE

R1	IF	type of beam support_type IS TBS AND max_BM(TBS, beam span, loads)
	THEN	check_bending

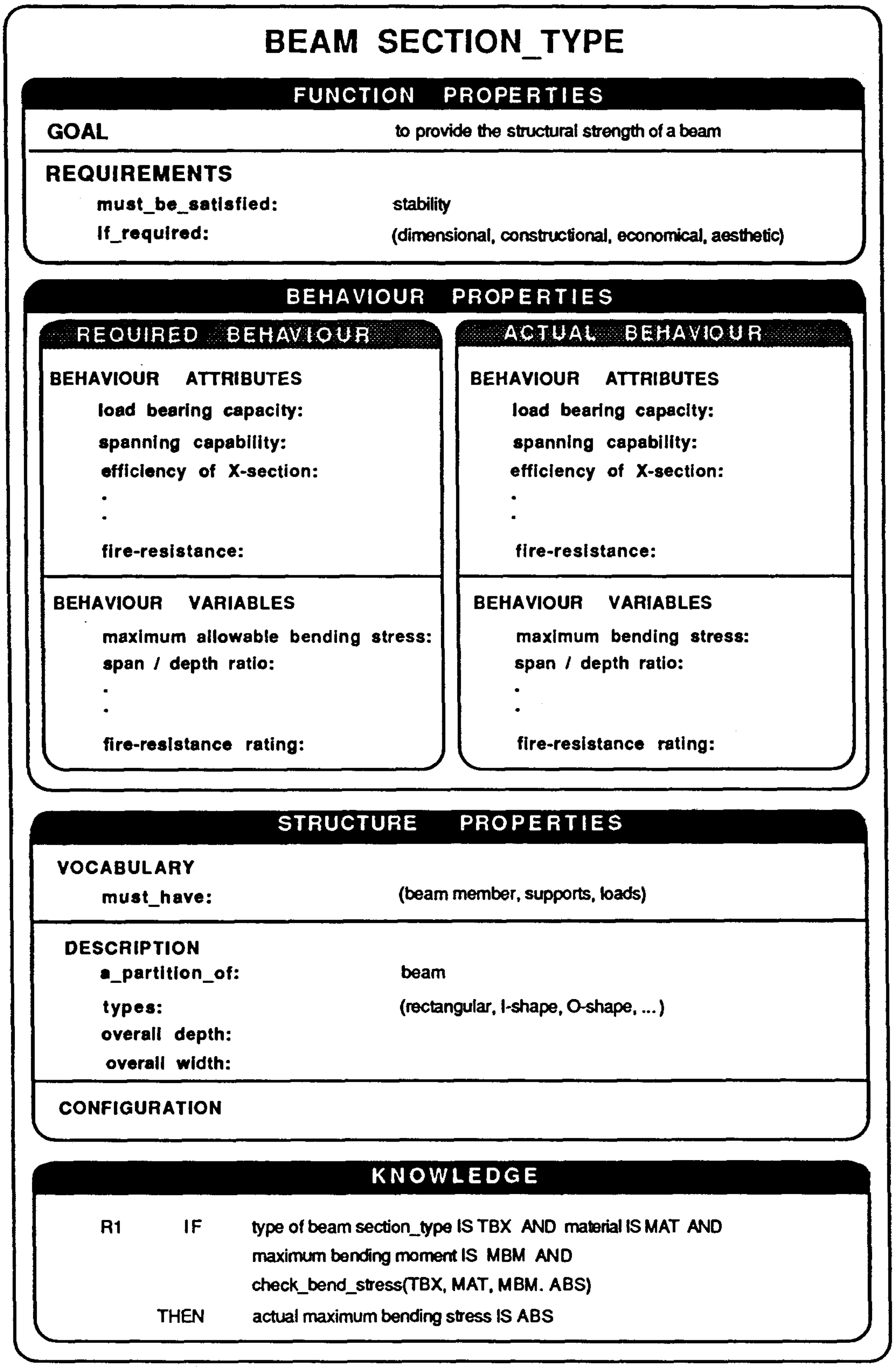
Fig. 4. Beam_support partition

Contents of prototypes

Two domains are being encoded as prototypes. The first deals with window design in buildings. This articulation of design experience is proving to be useful in itself as a means of increasing the designer's understanding of the knowledge being used in design. The second domain being encoded as prototype deals with the thermal/environmental design of buildings and their building services.

Design reasoning via instances

The architecture of a prototype-based design system has been designed and implemented on SUN workstations. The system provides for prototype selection from user specified requirements, instance creation, instance refinement, variable selection and variable value computation. A variety of reasoning processes can be utilized although the first one to be implemented is a type of constraint propagation. Here the variables whose



BEAM SECTION_TYPE

FUNCTION PROPERTIES

GOAL	to provide the structural strength of a beam
REQUIREMENTS	
must_be_satisfied:	stability
if_required:	(dimensional, constructional, economical, aesthetic)

BEHAVIOUR PROPERTIES

REQUIRED BEHAVIOUR	ACTUAL BEHAVIOUR
BEHAVIOUR ATTRIBUTES	**BEHAVIOUR ATTRIBUTES**
load bearing capacity:	load bearing capacity:
spanning capability:	spanning capability:
efficiency of X-section:	efficiency of X-section:
.	.
.	.
fire-resistance:	fire-resistance:
BEHAVIOUR VARIABLES	**BEHAVIOUR VARIABLES**
maximum allowable bending stress:	maximum bending stress:
span / depth ratio:	span / depth ratio:
.	.
.	.
fire-resistance rating:	fire-resistance rating:

STRUCTURE PROPERTIES

VOCABULARY	
must_have:	(beam member, supports, loads)
DESCRIPTION	
a_partition_of:	beam
types:	(rectangular, I-shape, O-shape, ...)
overall depth:	
overall width:	
CONFIGURATION	

KNOWLEDGE

```
R1    IF      type of beam section_type IS TBX AND material IS MAT AND
              maximum bending moment IS MBM AND
              check_bend_stress(TBX, MAT, MBM. ABS)
      THEN    actual maximum bending stress IS ABS
```

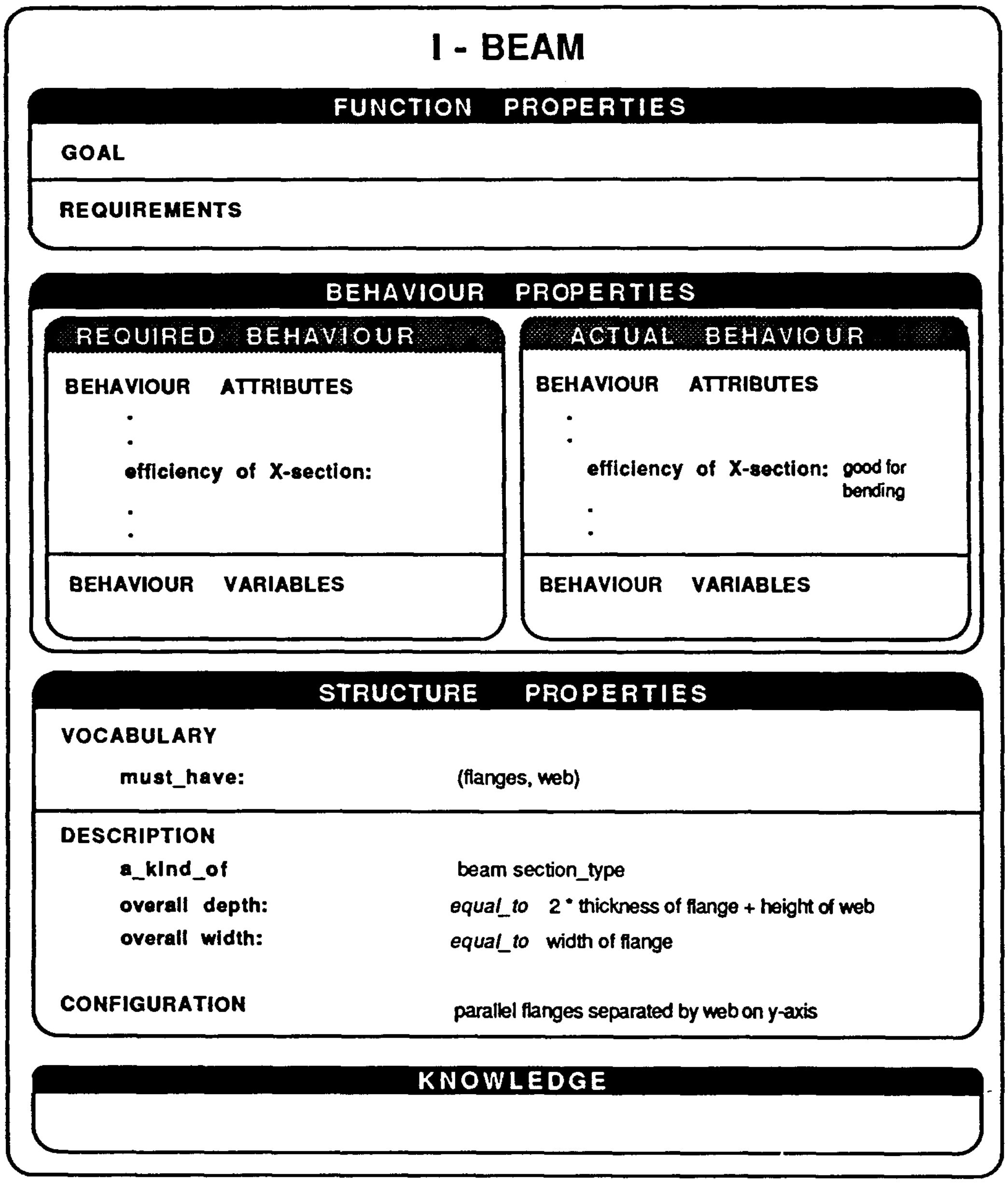
Fig. 5. Beam_section partition

values cannot be computed in the current instance are propagated as new requirements which are used to select further prototypes (Fig. 7).

User interfaces

An object-oriented, window-based, pulldown-menu, mouse-driven user interface is being implemented to allow the following:

creation of prototypes
viewing of prototypes
modification of prototypes
viewing of instances
viewing of prototype-instance hierarchy
user control of processes.



I - BEAM

FUNCTION PROPERTIES

GOAL

REQUIREMENTS

BEHAVIOUR PROPERTIES

REQUIRED BEHAVIOUR

BEHAVIOUR ATTRIBUTES	
•	
•	
efficiency of X-section:	
•	
•	

BEHAVIOUR VARIABLES

ACTUAL BEHAVIOUR

BEHAVIOUR ATTRIBUTES	
•	
•	
efficiency of X-section:	good for bending
•	
•	

BEHAVIOUR VARIABLES

STRUCTURE PROPERTIES

VOCABULARY	
must_have:	(flanges, web)
DESCRIPTION	
a_kind_of	beam section_type
overall depth:	*equal_to* 2 * thickness of flange + height of web
overall width:	*equal_to* width of flange
CONFIGURATION	parallel flanges separated by web on y-axis

KNOWLEDGE

Fig. 6. Example of I-Beam type

Cognitive models

Protocol studies of design fixation among civil engineers are being undertaken to provide cognitive support for the role of precedence in design.

Prototype-based non-routine design

Three aspects are under investigation and implementation. These are:
(i) model of creative design;
(ii) mutation in creative design; and
(iii) analogy in creative design.

Model of creative design

This is a joint project with Carnegie Mellon University which aims to produce models of knowledge-based creative design using prototypes. The model is built on the notion that creativity involves the importation of a concept from outside the space of direct concern into that space. This changes and extends that space thus offering the opportunity for creativity.

Mutation in creative design

One mechanism for the introduction of a concept, in the form of a new variable, being explored is that of mutation. Initially, a set of general mutation operators for manipulating the structure part of a prototype is being developed. Through qualitative models of structure–behaviour–function relationships the new variables are incorporated into the design.

Analogy in creative design

Another mechanism for the introduction of a new concept is that of analogy. Both transformational and derivational analogy are being explored. Analogy offers the opportunity of introducing new functions and new behaviours as well as new structures.

Prototype-based expert systems

Expert systems which make use of rules have been criticised for the shallowness of their knowledge. Hybrid systems have been built combining rules with the

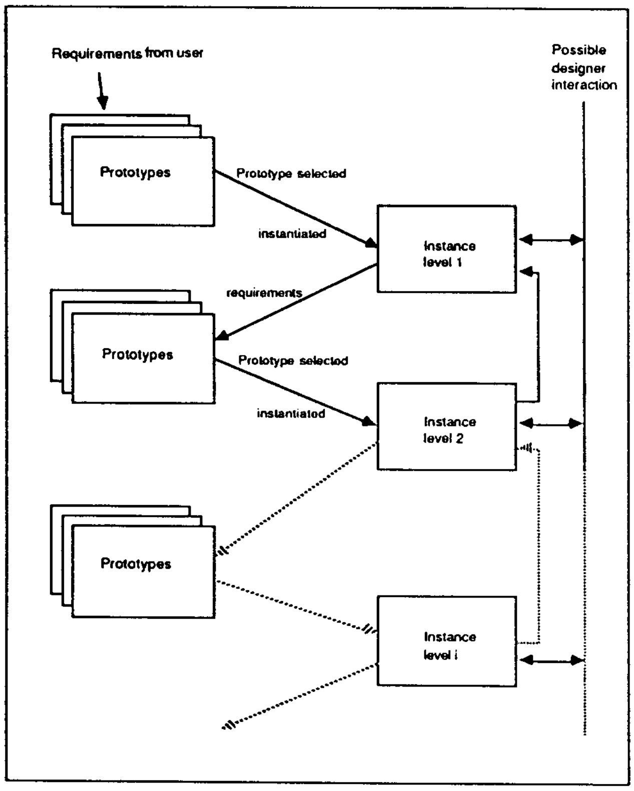


Fig. 7. Routine design with prototypes

capabilities of frames (or semantic nets) in order to add deeper knowledge. It is suggested that a prototype -based expert system using the prototype schema to represent generalized experiential, semantic knowledge can better provide this deeper knowledge. The incorporation of such a central framework readily allows for the integration of syntactical systems such as CAD systems.

A prototype-based expert system has the following components (Fig. 8):

user interface
prototype system
knowledge-based system
working memory.

User interface

The user interface allows for human–machine dialogue. It allows for queries to be put to either the knowledge-based system or the prototype system by the user or queries to be put to the user by the system and provides the display of required information. The user interface provides access to the knowledge acquisition facilities of the prototype and knowledge-based systems. In addition, the user interface allows for the integration of other applications including the creation and display of instances through graphical interfaces.

Prototype system

The prototype system serves as the repository of general knowledge about elements in its domain. Its knowledge needs to be sufficiently large to cover a wide variety of specific situations in which it may be used.

The prototype system contains the prototype acquisition facility, the general prototype base and the prototype engine. The prototype acquisition facility allows the user to create, modify or delete prototypes. There are two types of prototype bases, a general one and a problem specific one. The general prototype base contains the set of all prototypes and partitions created. The problem specific prototype base forms part of the working memory. The prototype engine allows for the manipulation of knowledge in the prototype base. It has the task of creating the most specific instance of a prototype as required including creating multiple instances of parts as required and of deriving values to specific variables of the instances as needed.

All external applications communicate with the instance base through the prototype engine.

Knowledge-based system

The knowledge-based system contains an expert system shell and the various knowledge-bases required for a particular application. The expert system shell has the usual knowledge acquisition facility, inferencing mechanisms and explanation facilities, but in addition, it has the task of knowing when to direct its queries to the prototype system, when to direct them to its facts base or alternatively to the user. Each knowledge base contains knowledge specific to some particular application. The knowledge bases are constructed in such a way that elements referred to exist in the prototype base. Otherwise the system reverts to a conventional expert system. This puts some responsibility on those (knowledge engineers or users) formulating a knowledge base to be acquainted with the prototype base or else to construct prototypes and knowledge bases to be consistent. Where the knowledge contained in a knowledge base is more specific that that contained in the prototype base the specific knowledge will be used. For example, in the prototype *building*, there may be general knowledge regarding deriving its height but in a knowledge base dealing with building regulations there may be more specific knowledge on how to measure the height of a building as required by the code. In all cases where there exist general

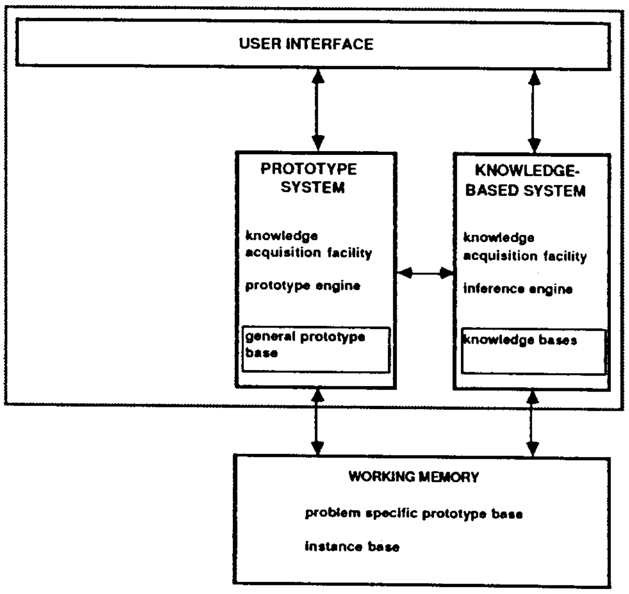


Fig. 8. Architecture of prototype-based expert system. The boxed elements, i.e. general prototype base and knowledge bases are domain specific and are created for that domain. The remainder of the elements within the shaded box comprise the shell of the system.

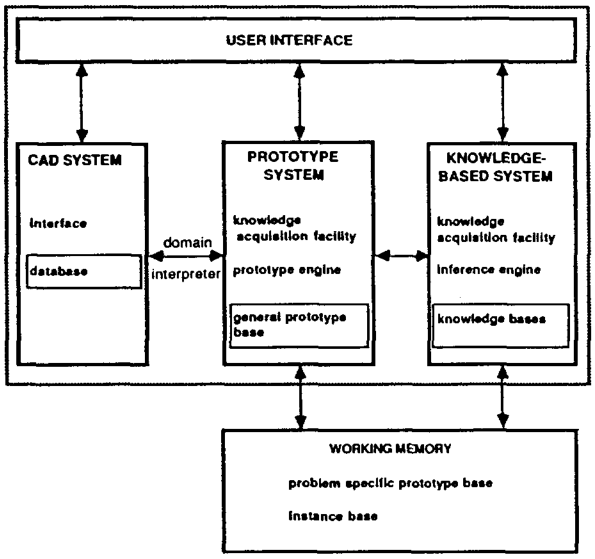


Fig. 9. Architecture of prototype-based expert system with CAD system

and specific knowledge, the specific knowledge overrides the general knowledge.

Working memory

The working memory consists of the problem specific prototype base and the instance base. The problem specific prototype base is constructed when a particular application requires a subset of the general prototype base. That is, only those prototypes relevant to the particular application are loaded.

All processes concerning an actual element deal with an instance of that element. All derived information relating to an instance is stored in the instance. The instance base contains the set of instances created at the time of running an application. The instance base thus constitutes the working environment of the system. Instances may be created by the system when a query is put or by the use either as a textual description or through a graphical interface. The prototype engine controls the creation of instances and external applications have to communicate through the prototype engine. The expert system shell can communicate directly to an instance (once it is created) to search for relevant information.

Interfacing with CAD systems and other applications

CAD systems and other applications need to communicate to the prototype system through a customized interpreter written specifically for the particular CAD system or application. CAD systems consist of a graphical interface and a data base. In the case of modelling systems, this database contains representations of objects and syntactical information regarding these objects. This syntactical information, namely in the form of dimensions, locations, etc. can be mapped onto the structural attributes of a prototype. An interpreter is required to map the specific form of the information in a CAD system's database to the format in the prototype. This interpreter will need to be specifically written for each different CAD system or similarly for an external application wishing to make use of the prototype system. Figure 9 shows the structure of the prototype-based system with an external CAD system.

CONCLUSIONS

This paper has argued that design is the production of a structure description from a performance description. It has further argued that in order to do so, designers draw upon a great deal of experience of particular cases but, perhaps more importantly, general concepts which they have formed from their experiences. The use of a system of prototypes as a conceptual schema is seen as developing that argument and forming a basis for a knowledge-based system for design. Prototypes provide a structure for representing a class of design allowing for the description of performance and structure properties and the knowledge required to map from one to the other. While the examples given show a frame-like representation, the prototype concept is not tied to any particular representation schema or any procedural methodology.

While at this stage of development, the main emphasis is on routine design, research is being carried out into requirements for innovative and creative design. The prototype schema lends itself readily to all three categories of design.

ACKNOWLEDGEMENTS

This research has been supported by the Australian Research Grants Scheme, the Australian Research Council, the Key Centre of Design Quality and the URG Special Project Grants. Members of the Design Computing Unit who have contributed to the current projects include Bala Balachandran, Eva Bednarska, Liang Chen, Jun Jo, Hyunsoo Lee, Sally McLaughlin, Rivka Oxman, Terry Purcell, K. W. Tham and Lena Ye. The work on prototypes has benefited from discussions with Mary Lou Maher.

REFERENCES

1 Coyne, R. D., Rosenman, M. A., Radford, A. D., Balachandran, M. B. and Gero, J. S. *Knowledge-Based Design Systems*, Addison-Wesley, Reading (to appear), 1990

2 Darke, J. The primary generator and the design process, *Design Studies*, 1979, **1**(1), 36–44

3 Dawkins, R. *The Blind Watchmaker*, Penguin Books, London, 1986

4 Dreyfus, H. and Dreyfus, S. Why computers may never think like people, *Technology Review*, 1986, 42–61

5 Gero, J. S. Operations research and engineering design: a review, *Architectural Science Review*, 1969, **12**(3), 67–77

6 Gero, J. S. Architectural optimization – a review, *Engineering Optimization*, 1975, **1**(3), 189–199

7 Gero, J. S. Dynamic programming in the CAD of buildings, in G. W. Jones and D. R. Smith, *CAD 76*, IPC Press, Guildford, 1976, 31–37

8 Gero, J. S. Note on 'Synthesis and optimization of small rectangular floor plans' of Mitchell, Steadman and Liggett, *Environment and Planning B*, 1977, **4**, 81–88

9 Gero, J. S. Computer aided design by optimization in architecture, *Design Studies*, 1980, 1(4), 227–230

10 Gero, J. S. (ed.) *Optimization in Computer-Aided Design*, North-Holland, Amsterdam, 1985

11 Gero, J. S. (ed.) *Design Optimization*, Academic Press, New York, 1985

12 Gero, J. S. (ed.) *Knowledge Engineering in Computer-Aided Design*, North-Holland, Amsterdam, 1985

13 Gero, J. S. (ed.) *Expert Systems in Computer-Aided Design*, North-Holland, Amsterdam, 1987

14 Gero, J. S. *Prototypes:* A basis for knowledge-based design, *Working Paper*, Department of Architectural Science, University of Sydney, Sydney, 1987

15 Gero, J. S. and James, I. An experiment in a computer-aided

constraint-oriented approach to the design of home units, in W. Mitchell, *Environmental Design and Research*, UCLA Regents, 1972, 20.1.1-10

16 Gero, J. S., Maher, M. L. and Zhang, W. Chunking structural design knowledge as prototypes, in Gero, J. S. (ed.), *Artificial Intelligence in Engineering: Design*, Elsevier/CMP, Amsterdam, 1988, 3–21

17 Gero, J. S. and Marmot, A. Modelling elevator lobbies, *Building Science*, 1974, **9**(4), 277–288

18 Gero, J. S. and Radford, A. D. The 'design' in computer-aided design, *Computing in Civil Engineering*, ASCE, 1981, New York, 876–890

19 Gero, J. S., Sheehan, P. and Becker, J. Building design using feedforward nonserial dynamic programming, *Engineering Optimization*, *1978*, **3**(4), 183–197

20 Hillier, B., Musgrove, J. and O'Sullivan, P. Knowledge and design, in W. J. Mitchell (ed.), *Environmental Design and Research*, 1972, UCLA Regents, 29.3.1–29.3.14

21 McLaughlin, S. and Gero, J. S. Requirements of a reasoning system to support creative design, *Knowledge-Based Systems*, 1989, **2**(1), 62–71

22 Radford, A. D. and Gero, J. S. Optimization for information in integrated environmental design, *PArC 79*, AMK, Berlin, 1979, 447–456

23 Radford, A. D. and Gero, J. S. *Design by Optimization in Architecture and Building*, Van Nostrand Reinhold, New York, 1988

24 Rosenman, M. A. and Gero, J. S. A system for integrated optimal design, in Gero, J. S. (ed.), *Design Optimization*, 1985, Adademic Press, New York

25 Schank, R. *Explanation Patterns: Understanding Mechanically and Creatively*, Lawrence Erlbaum, Hillsdale, New Jersey, 1986

26 Stanfill, C. and Waltz, D. Towards memory-based reasoning, *Communications of the ACM*, 1986, **29**(12), 1213–1228

27 Stiny, G. Introduction to shape and shape grammars, *Environment and Planning B*, 1980, **7**, 343–351