John S Gero and Mary Lou Maher (eds) (1998)
Computational Models of Creative Design IV
Key Centre of Design Computing, University of Sydney

USING ANALOGY TO EXTEND THE BEHAVIOUR STATE SPACE IN DESIGN

JOHN GERO AND VLADIMIR KAZAKOV
Key Centre of Design Computing
Department of Architectural and Design Science
University of Sydney NSW 2006 Australia

Abstract. *We propose an exploration model of design based on the extension of the space of design behaviours using analogy. The analogy is drawn on the basis of structure similarity or structure and behaviour similarity between source and target designs. New behaviours are introduced from the source design. This paper describes such a process and discusses its significance for creative design.*

1. Introduction

One way to describe a design process is through a structure-behaviour-function (S-B-F) framework (Gero, 1990). Each design here is represented as a triplet of points in three different spaces, which represent three levels of abstraction: the structure space, S which describes the elements and their relationships in an artefact (list of design variables); the behaviour space B of derived properties of the artefact (list of performance or criteria variables) and the function space F which defines the artefact's teleology. Essentially, structure space includes all the variables whose values can be directly decided upon and measured from the artefact (or its computational model) and behaviour space includes all the variables whose values are important to the quality of design and which cannot be measured directly from a model of the artefact but should be computed using only variables whose values can be `measured' in structure space. The union of these three spaces produces the state space of a design problem D=S B F, Figure 1. A model of design based on this framework includes also knowledge which is necessary to support mappings between S, B and F, ranges of variables, constraints, etc. Ranges of variables and constraints define a feasible subset of D: D **D** which contains all feasible designs and where search for the best or an appropriate design is to be conducted. Correspondingly, the feasible subsets

of three subspaces are: S **S**, B **B** and F **F**. The design state space D represents a conceptual view of designing.

Routine design processes in this framework are characterised by a fixed D and **D** and are represented by a search process in a well-defined, fixed and bounded (although in some case countably infinite) space **S** which is governed by a set of criteria defined by components of **B** (Gero, 1996). In other words, the space of all possible designs **S** is given and the goal is to choose which one of them has the best values of the set of given behaviours **B**. This space of design alternatives **S** is defined (known) a priori before the designing process is started and remains the same throughout this process. The set of desired behaviours required from an artefact is also chosen a priori before the designing process is started and remains unchanged throughout this process. The constancy of the spaces **S** and **B** simply means that a complete set of design variables is chosen a priori and what is left to be done by the designer is to choose an appropriate set of values for these variables. In this model routine designing is viewed essentially as an optimisation or constraint satisfaction problem with search state space **S** and multi-criterion objective defined by **B**.

It is widely accepted that one way creative designing processes differs from routine designing processes is that the search processes are augmented by exploration processes during which a new design state space is created (Gero, 1996).

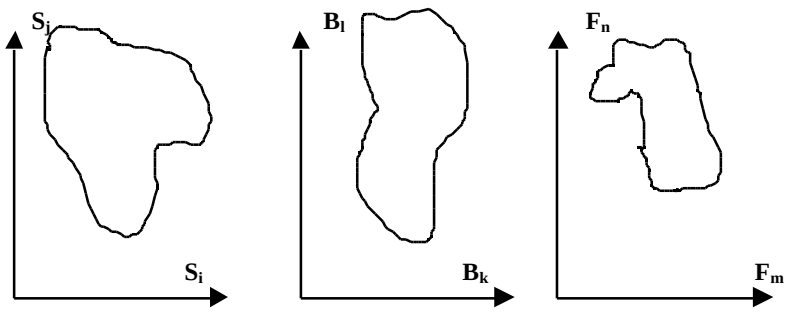


Figure 1. The three subspaces which constitute the state space of design (Gero, 1990).

Usually this new state space is not created from scratch but rather by modification of the current space: $\mathbf{D} \rightarrow \mathbf{D}^{\mathbf{m}}$. By definition this modification always includes the addition of new areas to the state space, Figure 2. Gero

(1990), argued that if the new modified $\mathbf{D}^{\mathbf{m}}$ still is a subset of the same initial **S** and therefore can be reduced to changes in ranges of existing design variables and does not require introduction of additional variables then a search in such $\mathbf{D}^{\mathbf{m}}$ corresponds to an innovative design process. If the replacement of **D** with $\mathbf{D}^{\mathbf{m}}$ does require adding some new design variables – new dimensions to S (that is, S^{m} S) only then is there a creative design process. This approach allows us to model a creative design process as sequence of periods when a temporarily fixed S is searched followed by periods when S is replaced with a modified S^{m} The block-schema of an algorithm where search and exploration stages are executed cyclically is shown in Figure 3. One can view this approach as an attempt to model a creative design problem as a sequence of routine design problems.

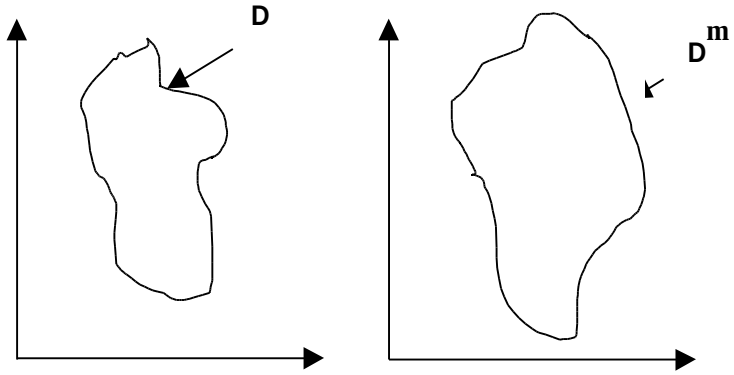


Figure 2. The change in a feasible subset of state space **S**

Gero (1996) suggested that five fundamental processes can be used as computational engines to produce a new $\mathbf{S}^{\mathbf{m}}$ - emergence, combination, mutation, first principles and analogy. In this paper we will use analogy. Although any of the three components spaces **S**, **B** and **F** can be changed when **S** is replaced with $\mathbf{S}^{\mathbf{m}}$ the majority of the creative design models so far proposed deal with the changes to a structure state space **S** only and leave the **B** and **F** spaces unchanged. They yield a search in a changing space of possible designs **S** (that is, search accompanied by introduction of new structure variables) governed by the same fixed set of behaviours **B**. In this paper we describe a design approach which is driven by the requirement to

modify the behaviour state **B** ⊄ **B**^{m} produced by the introduction of new dimensions to **B**. This approach is a particular case of a general approach, shown in Figure 3. Its exploration stage always includes the replacement of

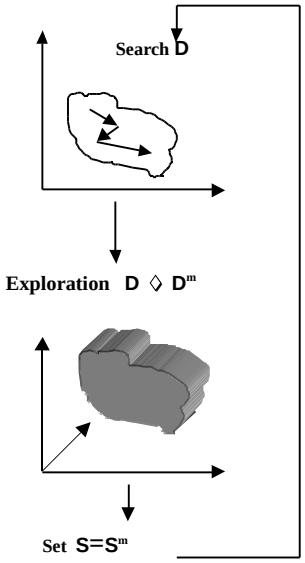


Figure 3. A model of a creative design process as a cycle of search and exploration stages.

B by a higher dimensioned **B**^{m} and sometimes a consequential replacement of **S** by a higher dimensioned **S**^{m}, Figure 4. Since **B**^{m} ⊄ **B** this is a model of a creative and not an innovative design process. Essentially, this means that the criteria which are used to evaluate designs are changed during designing. Those criteria changes may or may not require a change in the corresponding structure space.

2. An Analogy-Based Creative Design Model

Analogy is accepted as one important process in modelling creative designing. Basically if one tries to solve some problem (called a target problem) using analogy he/she looks for some other problem which is

somehow similar to the target problem (called a source problem) and whose solution is known and then attempts to construct a solution to the target problem by adapting a solution from the source problem (transformational analogy) or using a successful problem solving process from source problem (derivational analogy). Many formalisation of analogy processes has been developed, see for example Carbonell (1986), French (1995), Keane (1988), Mitchell (1993), and Prieditis (1988).

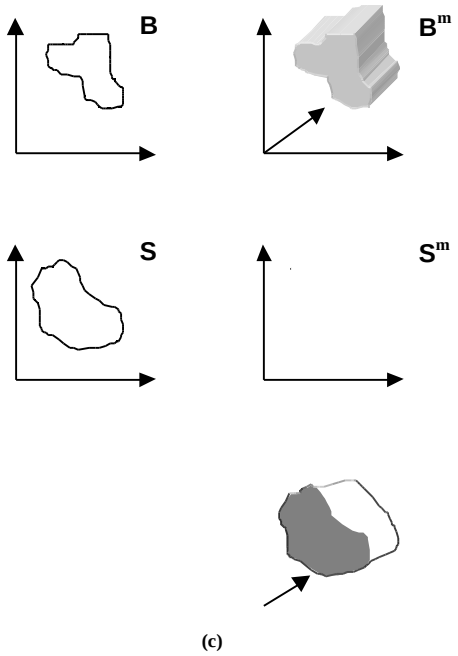


Figure 4. The exploration stage of the proposed algorithm. (a) The behaviour state space is expanded, two possible effects on the structure state space follow, either (b) the existing structure variables are sufficient to calculate values of the new behaviours although constants or their value ranges may change, or (c) new structure variables are required.

Usually analogical problem solving in design is used when a routine design fails to yield a satisfactory design solution. For the given target design it consists of finding and retrieving an analogous source design from a design archive which is then adapted to suit the requirements of the target design. Thus, in terms of the S-B-F framework the analogy-based designing process includes three steps:

(1) matching and retrieval stage: the similarity of each design from an archive and the target design is evaluated; the design with the

highest degree of similarity $\mathbf{D_S}(i)$**:** (for which $\mathbf{SIM}(\mathbf{D_T},\mathbf{D_S}(i))=\mathbf{max_k SIM}(\mathbf{D_T},\mathbf{D_S}(k))$, where **SIM**(*,*) is a real function which measures the degree of similarity (matching) between its parameter spaces, $\mathbf{D_S}(i)$ is the i-th source design state space, $\mathbf{D_T}$ is the target design state space) is retrieved for further processing;

(2) mapping and transformation stage: $\mathbf{D}^{\mathbf{m}}=\tau(\mathbf{D_T}\ \ \mathbf{M}(\mathbf{D_S}(i)))$. Here **M** is a mapping operation and τ is a transformation operator. Mapping adds some design variables from $\mathbf{D_S}$ (i) into $\mathbf{D_T}$ and then τ transforms the result of this operation into a modified design space $\mathbf{D}^{\mathbf{m}}$.

(3) knowledge construction stage: first, the knowledge from the target design state space is passed to the modified design state space; then the analysis of the $\mathbf{D}^{\mathbf{m}}$ is carried out to determine if any the relationships in it are not supported by its current knowledge; if some are not then additional parts of knowledge from the source state space are identified and added to the current knowledge of the modified design state space. We use standard realisations of this knowledge construction step (see for example Qian and Gero, 1996).

(4) Depending on which subspace of **D** is used to match the source and target designs and which is used to construct the new $\mathbf{D}^{\mathbf{m}}$ the analogy-based design process could belong to one of three following types:

A similarity between structure spaces $\mathbf{D_S}$ and $\mathbf{D_T}$ is used to match and retrieve a source design, the similarity function is defined as $\mathbf{SIM}(\mathbf{S_T},\mathbf{S_S})$. Then parts of $\mathbf{S_S}$ are used to modify $\mathbf{S_T}$ by mapping $\mathbf{S_S}$ on to it and then transforming its result: $\mathbf{S}^{\mathbf{m}}=\tau(\mathbf{S_T}\ \ \mathbf{M}(\mathbf{S_S}))$. $\mathbf{B_T}$ and $\mathbf{F_T}$ become $\mathbf{B}^{\mathbf{m}}$ and $\mathbf{F}^{\mathbf{m}}$ without change. Then the analysis of $\mathbf{S}^{\mathbf{m}}$ is used as a guide for the knowledge construction process by adding the necessary knowledge components from the source design to the knowledge in the target design. The operations of this algorithm is shown in Figure 5.

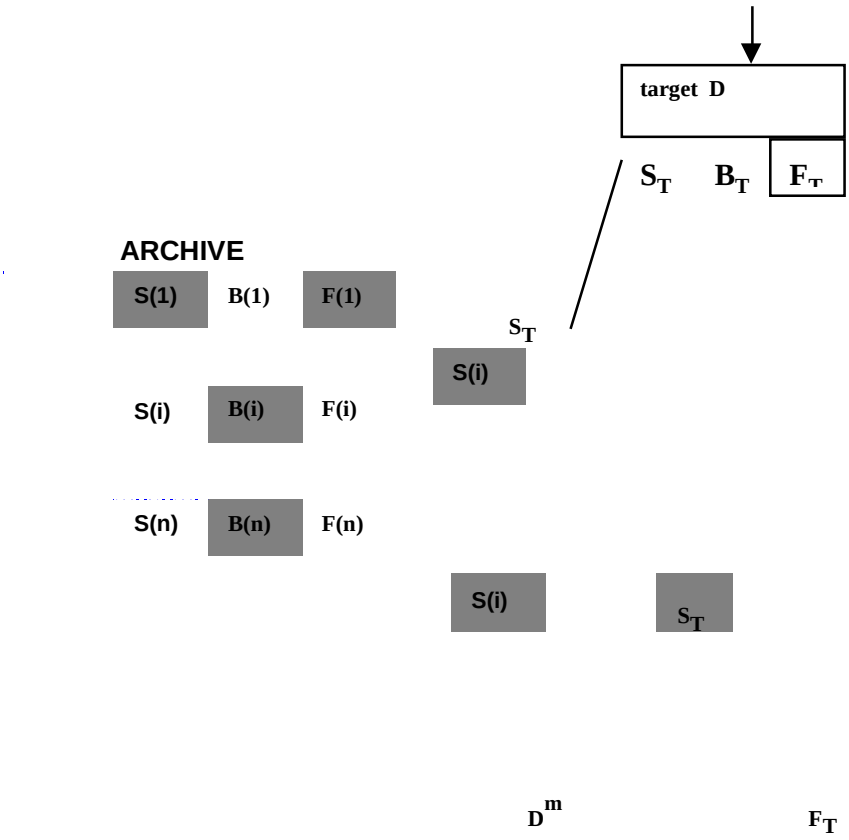


Figure 5. The analogy-based design algorithm of type A, based on structure-structure matching and structure-structure mapping.

A similarity between behaviour spaces $\mathbf{B_S}$ and $\mathbf{B_T}$ is used to match and retrieve a source design , the similarity function is defined as $\mathbf{SIM(B_T,B_S)}$, where $\mathbf{B_T}$ and $\mathbf{B_S}$ are behaviour component-spaces of $\mathbf{D_T}$ and $\mathbf{D_S}$ correspondingly. The structure space $\mathbf{S_T}$ is used to modify the structure space of the target design by mapping $\mathbf{S_S}$ onto it and transforming the result of this operation: $\mathbf{S^m}=\tau(\mathbf{S_T}\ \ \mathbf{M(S_S)})$. $\mathbf{B_T}$ and $\mathbf{F_T}$ become $\mathbf{B^m}$ and $\mathbf{F^m}$

without change. Then the analysis of $\mathbf{S}^{\mathbf{m}}$ is used as a guide for any knowledge construction process by adding the required knowledge components from the source design to the knowledge from the target design. The operations of this algorithm are shown in Figure 6.

A surface similarity between structure spaces is used to match and retrieve a source design, the similarity function is defined as $\mathbf{SIM}(\mathbf{S_T},\mathbf{S_S})$. The mapping and transformation steps here may consist of one or two stages, depending on the $\mathbf{S_T}$, and $\mathbf{S_S}$ chosen. During stage 1 parts of $\mathbf{B_S}$ are used to modify $\mathbf{B_T}$ by mapping $\mathbf{B_S}$ on it and transforming the result of this operation: $\mathbf{B}^{\mathbf{m}}=\tau(\mathbf{B_T}\ \ \mathbf{M}(\mathbf{B_S}))$. Then an analysis is carried out (using the knowledge from both target and source designs) to determine if $\mathbf{S_T}$ contains all the variables on which the behaviours from $\mathbf{B}^{\mathbf{m}}$ depend. If it does, then $\mathbf{S_T}$ and $\mathbf{F_T}$ become $\mathbf{S}^{\mathbf{m}}$ and $\mathbf{F}^{\mathbf{m}}$ without change. Otherwise stage 2 of the algorithm is executed, during which these missing variables are identified in $\mathbf{S_S}$ and are mapped onto $\mathbf{S_T}$: $\mathbf{S}^{\mathbf{m}}=\tau_{\mathbf{1}}(\mathbf{S_T}\ \ \mathbf{M_1}(\mathbf{S_S}))$. Then follows the stage of the knowledge construction induced by $\mathbf{S}^{\mathbf{m}}$ and $\mathbf{B}^{\mathbf{m}}$. The operations of this algorithm are shown in Figure 7. Note that if stage 2 is not executed then the algorithm does not add any new dimensions to $\mathbf{S_S}$. The resulting transformation $\mathbf{S_S}$-_$\mathbf{S}^{\mathbf{m}}$ is shown in Figure 4(b). If stage 2 is executed then the algorithm always adds new dimensions to $\mathbf{S_S}$. The resulting transformation $\mathbf{S_S}$_$\mathbf{S}^{\mathbf{m}}$ is shown in Figure 4(c). Many computational models of design that apply realisations of this type of analogical reasoning have been developed (see Goldschmith (1994), Navinchandra (19910 and Qian and Gero (1996)). The majority of them can be classified as belonging to either type A or type B of the above-described classification. As a rule they draw the analogy on the basis of structure or behaviour similarity between source and target design state spaces and then use the source problem's structure to construct a modified structure space for the target problem. In terms of Figure 3, their exploration steps include modification of structure space **S** (jointly with corresponding knowledge) only and leave **B** and **F** spaces unchanged. That is, they are attempting to change the current problem's (the target problem's) structure space **S** by injecting parts and features (new structure design variables) from $\mathbf{S_S}$ into **S** or by replacing **S**

with an adapted $\mathbf{S_S}$. The set of criteria which evaluate designs remain constant in such algorithms.

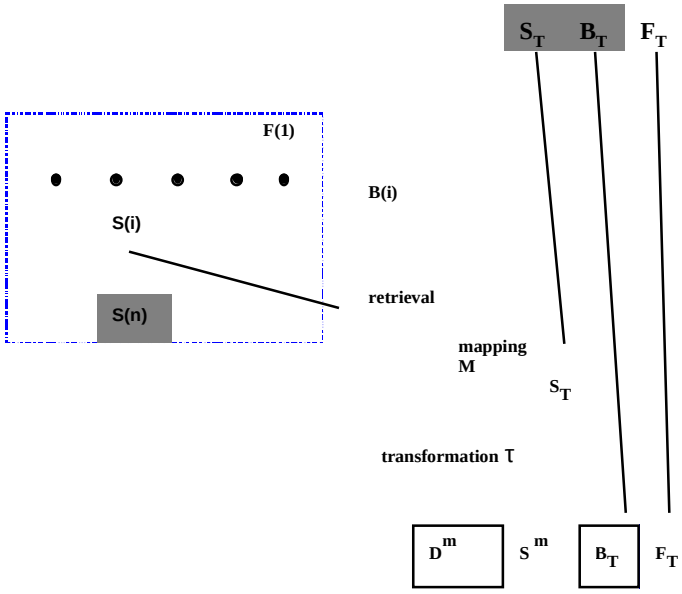
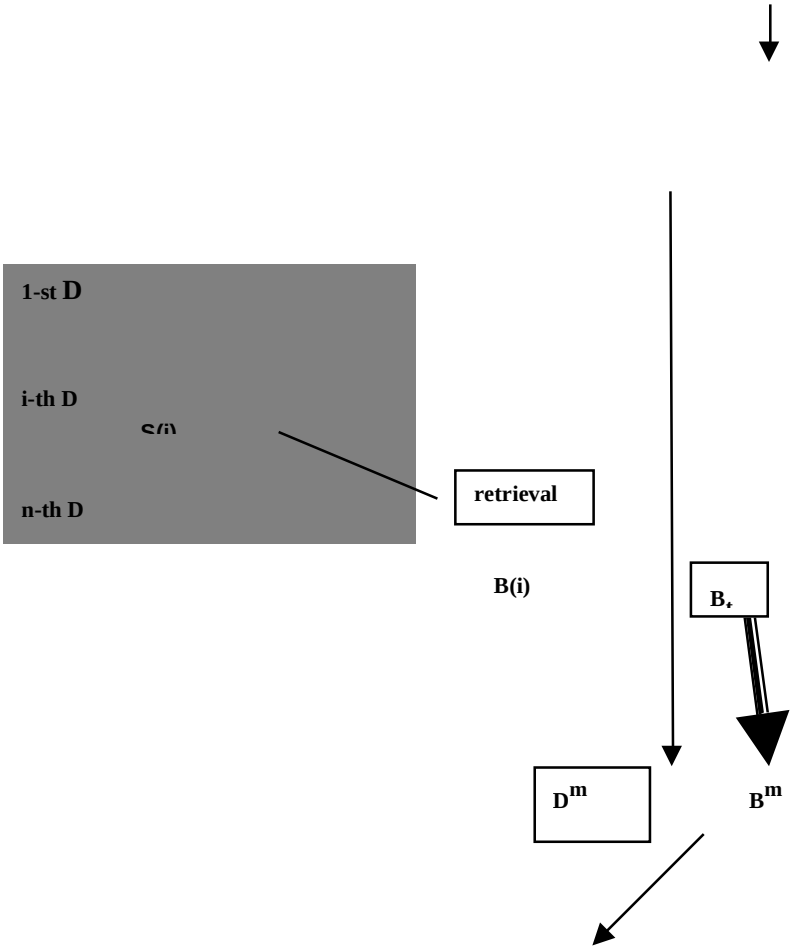


Figure 6. The analogy-based design algorithm of type B, based on behaviour-behaviour matching and structure-structure mapping

In constructing a computational model of creative design processes the assumption that similarity in behaviour of two design problems may lead to useful additional structure features for the target design or that the similarity in structure may cause further structure similarities to be useful (on which the above-mentioned works are based) seems no more valid than the assumption that similarity in structure between two designs may cause some behaviours from one to be useful for the other. Designers may well draw an analogy on the basis of structure or structure and behaviour similarity and then try to apply behaviours from designs which have been matched. In this paper we present a computational approach of type C which, unlike the other analogy-based design processes, follows this strategy. Here again it is assumed that a design state space for the current design is given and a design


1-st D
i-th D
S(i)
n-th D
retrieval
B(i)
B+
D^m
B^m

state space of the source designs (not necessarily from the same domain) is retrieved and has structure or structure and behaviour spaces similar to the corresponding spaces of the current (target) design. Then the system tries to bring across a part of the source design into the target. Here this part consists of some of the behaviours which are present in a source but not present in the target design. It is also possible that some of the current behaviours from $\mathbf{B_T}$ are then discarded. Essentially this means that a source design is found which has a similar structure space and then some of its criteria are transferred to the target. If these new criteria (behavioural components) are expressible in terms of the target structure space variables (in another words, if the model of the target design has enough to support these new behaviours) then all that is required here is knowledge that is necessary to express these new behaviours. Hence in this case only the behaviour space **B** is changed but **S** stays the same, Figure 4(b). In many cases, the additional behaviours which are brought into the design cannot be expressed in terms of the variables of the target structure state space. Then stage 2 of the algorithm shown in Figure 7 should be used which yields a modification of the structure space which is shown in Figure 4(c).

In order to make this operation meaningful the usefulness of the added behaviours needs to be evaluated. We do not consider this problem fully here. We only note that a necessary condition for these behaviours to be useful is the requirement that they be able to discriminate between designs in the design space. In other words, one should be able to generate points in the structure space which have various values of these new behaviours. If the structures are not able to provide distinguishable values of the new behaviour variables then, prima facie, these new behaviours cannot be useful. Another requirement is that if an introduction of new behaviours into the design leads to an introduction of some additional structure variables into it then the designs with the resulting structure should be able to achieve at least a Pareto performance with respect to the old behaviours as designs with old structures. That is, an analogy of type C should not cause a decline in the behaviours which are already present in the design. If it does reduce existing behaviours but the added behaviours turn out to be useful then it is likely, that a modified structure produced by stage 2 of type C algorithm needs further modification.

3. Analogical Design Retrieval Process

3.1 DESIGN PROTOTYPES

We assume that the design state space is represented using design prototype *P* (Gero, 1990)

$$P=\{F, B, S, Ex, K_c, K_q, K_r\},$$

where

B = a finite set of behaviour variables,

Ex = a finite set of exogenous variables which describe external conditions,

F = a finite set of function variables,

K_c = computational knowledge,

K_r = relational knowledge,

K_q = qualitative knowledge,

S = a finite set of structure variables.

Since the proposed algorithm does not affect the function spaces **F**, we omit the *F* part of design prototype which describes the function space. Each design variable has a label and a range of valid values associated with it. We assume that the labels which are used to code design prototypes in a system are standardized and codified so that a simple label comparison could be used to identify similar labels. Otherwise some kind of linguistic analysis based on a hierarchical term structure needs to be used in a system to find and compare generalized meanings of labels, etc.

During source retrieval, mapping and transformation the proposed algorithm uses only part of the information which is kept in design prototype. Namely, it uses the structure and behaviour and the relational knowledge. We represent these parts of a design prototype as a structure + behaviour graph, Figure 8 or as a set of two tables, Table 1. We assume

TABLE 1. Part of design prototype required by the algorithm in the form of two tables. First table represents the structure and the second one contains dependencies of behaviours on structure elements.

Element	**label**	**attribute**	**relation**	**label**
e_1	*****			
		a_1 (1)		*****

		a_2 (1)		*****
			r_1	*****
			r_2	*****
e_2	*****			
		a_1 (2)		*****
		a_2 (2)		*****
			r_2	*****

		dependencies	
behaviour	**label**	e_1	e_2
b_1	string	no	yes

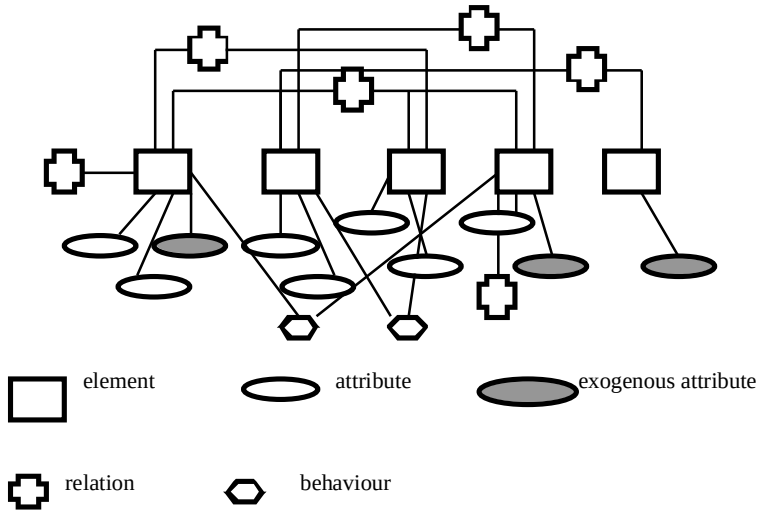


Figure 8. Structure+behavour graph of a design prototype

that structure has three types of design variables: (1) elements (components) of structure {e} depicted in Figure 8 with rectangles; (2) attributes (describe properties of elements) {a} depicted in Figure 8 with ovals and (3) relations (describe relationship between elements 1,…,n) {*r*(1,…,n)}, depicted in

Figure 8 with crosses. Filled ovals denote exogenous attributes. The rest of the exogenous variables are not shown in Figure 8 and are kept in a separate unstructured section of a design prototype. Hexagons denote behaviours. Connecting lines denote the connections (relationships) between corresponding entities and thus represent of relational knowledge K_r. Note that we define relation as a general type constraint, including relation of element with itself as a constraint which is imposed on some attributes of an element. In the graphical form that means that crosses can also be attached to an end of only one branch.

3.2 STRUCTURE MATCHING AND STRUCTURE SIMILARITY MEASUREMENT

Basically two types of similarity are recognised - surface similarity when two problems have the same attributes and structural similarity when they have the same underlying casual dependency network (Holyoak and Koh 1987; Thagard et al. 1994). The proposed algorithm is based on the evaluation of a surface similarity between two design structure spaces or, more accurately, between their design prototypes. The surface similarity can be defined in many different ways. We define it as proportional to the maximal total number of matching structure elements in two design prototypes. The corresponding function, which measures this similarity, is defined as

$$\mathbf{SIM}(\mathbf{S_T},\mathbf{S_S}) = \mathbf{SIM}(P_\mathbf{T},P_\mathbf{S}) = \{\max N_m\} / N(P_\mathbf{T},P_\mathbf{S}), \quad (1)$$

where N_m is double the maximal total number of matching pairs of elements in prototypes $P_\mathbf{T}$ and $P_\mathbf{S}$ (maximal means that from many possible lists of matching pairs of elements from $P_\mathbf{T}$ and $P_\mathbf{S}$ one list should be chosen for which N_m is maximal) and $N(P_\mathbf{T},P_\mathbf{S})$ is the total number of structure elements in both these prototypes. Note that in this formula as well as in all similar formulas in this paper, relative numbers could be substituted with absolute numbers.

Two structure elements in two design prototypes are defined as matching if sufficient number of their attributes and relations match. Two attributes which could design variables or exogenous variables or two relations are defined as matching if their labels match (that is, their labels match as character strings). These heuristic definitions seem to correspond the

intuitive picture of surface structure similarity but should not be considered unique or universally applicable.

Let us give these definitions more formally. Consider two structure elements $\{e_1\}$ from $P_{\mathbf{T}}$ and $\{e_2\}$ from $P_{\mathbf{S}}$. Element $\{e_1\}$ has k attributes $\{a_1(1),\ldots a_k(1)\}$ and l relations $\{r_1(1,..),\ldots r_l(1,..)\}$ associated with it. Similarly, element $\{e_2\}$ has p attributes $\{a_1(2),\ldots a_p(2)\}$ and q relations $\{r_1(2,..),\ldots r_q(2,..)\}$ associated with it. We define that two structure elements $\{e_1\}$ and $\{e_2\}$ match if the following condition holds

$$NS(e_1,e_2)= N(a(1),a(2))/(k+p) + N(r(1),r(2))/(l+q) > \varepsilon, \quad (2)$$

where $\varepsilon > 0$ is the threshold which should be chosen experimentally; $N(a(1),a(2))$ is the doubled number of pairs of attributes of $\{e_1\}$ and $\{e_2\}$ whose labels match label $\{a(1)\}\omega$ label$\{a(2)\}$; $N(r(1),r(2))$ is the doubled number of pairs of relations which depend on $\{e_1\}$ and $\{e_2\}$ and whose labels match label$\{r(1)\}\omega$ label$\{r(2)\}$. Note, that the maximal value of $NS(e_1,e_2)$ is 2, therefore ε is not greater than 2.

In terms of structure + behaviour graphs of two design prototypes the problem of computing $\mathbf{SIM}(P_{\mathbf{T}},P_{\mathbf{S}})$ takes the form of finding such partial mapping between these two graphs that the maximal possible number of rectangles are connected with solid arrows (which depict matching structure elements), Figure 9. Here two rectangles could be connected if a sufficient number of their attributes and relations are matched and thus are connected with dashed lines in Figure 9. Function **SIM** basically gives the ratio of the number of rectangles connected with the solid arrows to the total number of rectangles in both graphs.

In the general case when one calculates the **SIM** function and generates corresponding lists of matching elements, matching attributes and matching relations combinatorial explosion may occur. We suggest the use of a sub-optimal greedy heuristic algorithm to ensure tractable performance. This algorithm includes two greedy searches on two levels: one which computes $NS(e_1,e_2)$ and builds a list of matching attributes and a list of matching relations (that is, when it draws dashed arrows between ovals and crosses which are one branch way from rectangles e_1, and e_2); and the other when it uses the $NS(e_1,e_2)$ found to build a list of matching elements (that is, to draw solid lines between rectangles). Both searches are greedy in a sense that they do not produce the most complete (optimal) set of matches because they

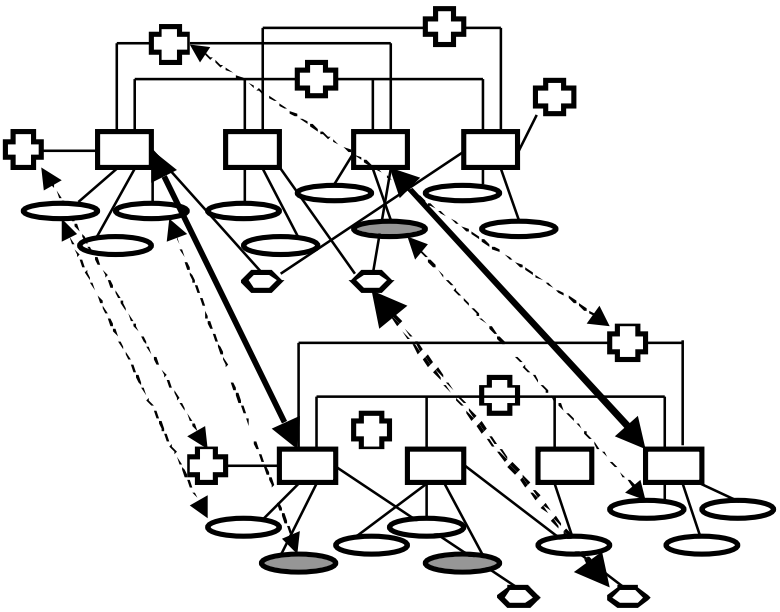


Figure 9. Matching of two design prototypes. Structure matching. The objective here is to draw as many solid arrows as possible between structure elements (rectangles) in two graphs. Solid lines could be drawn between rectangles whose attributes and relations are connected with dashed arrows. Two attributes or relations can be connected with dashed lines if their labels match. Behaviour matching. The objective here is to draw as many double dashed arrows as possible between behaviours (hexagons) in two graphs. Two behaviours can be connected if their labels match

build their set of matches from the first match they encounter. This structure matching algorithm is shown in Appendix A.

The computation of the similarity function **SIM**($P_{\mathbf{T}}$,$P_{\mathbf{S}}$) for two design prototypes $P_{\mathbf{T}}$,$P_{\mathbf{S}}$ automatically yields the following hierarchical structure of matching entities: a list of matching structure elements {E}=({$e_{\mathbf{T}}(i_1)$, $e_{\mathbf{S}}(j_2)$,},…, ({$e_{\mathbf{T}}(i_N)$, $e_{\mathbf{S}}(j_N)$,}). Each pair from that list has two sub-lists associated with it: attributes {A}= ({a(1),a(2)},…, {a(1),a(2)}), relations {R}=({r (1),r(2)},…,{r(1),r (2)}). This is an analytical representation of the matches that are shown in Figure 9 with the dashed and solid arrows.

3.3 EVALUATION OF BEHAVIOUR SIMILARITY

In practice, it is far more beneficial to retrieve all the source $P_{\mathbf{S}}(i)$ with a sufficiently high level of similarity to the target: $\mathbf{SIM}(P_{\mathbf{T}},P_{\mathbf{S}}(i)) > \alpha \max_j \mathbf{SIM}(P_{\mathbf{T}},P_{\mathbf{S}}(j))$, where $0 < \alpha < 1$ is a constant which should be chosen experimentally and maximisation is done over all prototypes in an archive. The retrieved set of $\{P_{\mathbf{S}}(i)\}$, i=1,…,d which all have structures that are highly similar to the structure of the target design, can then be ranked by the degree of similarity $\mathbf{SB}(P_{\mathbf{T}},P_{\mathbf{S}})=\mathbf{SB}(B_{\mathbf{T}},B_{\mathbf{S}})$, between their behaviour spaces and the behaviours of the target problem. $\mathbf{SB}(B_{\mathbf{T}},B_{\mathbf{S}})$ is defined as the fraction of the maximal number of behaviours in $P_{\mathbf{T}}$ which have matches in the source $P_{\mathbf{S}}$ to the total number of its behaviours. The match of two behaviours is denoted by the double dashed arrow between two hexagons in Figure 10. The value of $\mathbf{SB}(B_{\mathbf{T}},B_{\mathbf{S}})$ is simply the doubled total number of these arrows between their structure + behaviour graphs. The sources with different values of **SB** and **SIM** correspond to different design situations and require different computational strategies. This question will be considered in the next section.

Again we propose to compute $\mathbf{SB}(B_{\mathbf{T}},B_{\mathbf{S}})$ in a greedy fashion. The corresponding algorithm is shown in Appendix B. It computes the value of $\mathbf{SB}(B_{\mathbf{T}},B_{\mathbf{S}})$ and generates the list of matching behaviours {BB}.

3.4 CHOOSING A SOURCE ON THE BASIS OF STRUCTURE AND BEHAVIOUR SIMILARITY

The retrieval process proceeds in two stages. During the first stage a set of sources with high structure similarity with the target is retrieved from the archive. During the second stage the source which has the given degrees of behaviour – behaviour similarity is filtered out of this set and used for further processing. Since the high degree of structure-structure matching is fundamentally important for the validity of the proposed algorithm - the higher the similarity between $\mathbf{S}_{\mathbf{s}}$ and $\mathbf{S}_{\mathbf{t}}$ the likelier is that the replacement of **D** with $\mathbf{D}^{\mathbf{m}}$ will be productive. Once a set of source design prototypes have been identified and retrieved their behaviours are matched. If the two

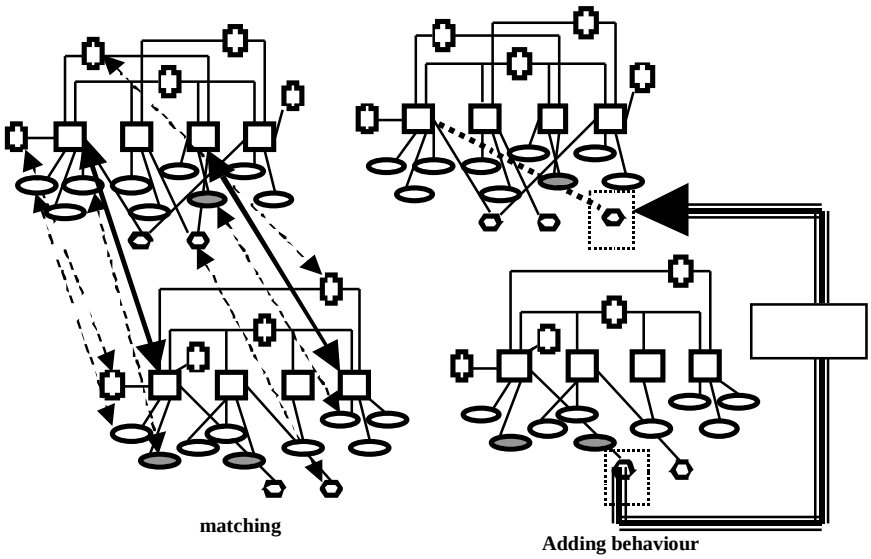


Figure 10. Adding new behaviour to design prototype.

behaviour spaces match completely then no new behaviours can be found since the source design prototypes contain no behaviours which are not already in the target. If there are no matches between the two behaviour spaces then it is unlikely that the behaviours of the source are going to satisfactorily augment those in the target. The most useful sources are likely to be those where there is considerable overlap in behaviours with the target but not a complete overlap. Therefore the best performance should be expected from the source which has high value of **SIM** and a reasonably high value of **SB** functions. The best actual choice of **SB** function's values is domain or even problem dependent and can be found experimentally.

It is clear that one should try to minimize the size of the source structure which has to be passed to the target, because transplantation of structure is a difficult and potentially dangerous operation – it could damage or destroy the coherence of the target's structure as well as viability of structure-behaviour connections in the target which is difficult and computationally expensive to fix.

The most desirable case is when all the structures in source and target are matched and all the source behaviours except a few are matched with target behaviours, $\mathbf{SIM}(P_T,P_S)=1$ and $\mathbf{SB}(P_T,P_S)$ δ1. It is equally likely here that

the passing of any of these unmatched behaviours to the target will be productive so one should try all of them. On the other hand one should expect that this operation will not require any structure space expansions (stage 1 of the Type C algorithm should be sufficient). This is probably the most favorable case to apply the proposed approach.
Let us consider a case when source and target structures as well as source and structure behaviours do not match completely. Assume the source has two unmatched behaviours b_1, that depends on the structure elements which have matches in target, and b_2, that depends on the structure elements which have no matches in target. It is clear that the passing of b_1 to the target will not require the transplantation of new structure elements to it but passing of b_2 will require that. Therefore if there is no any particular external reason to expect b_2 to be productive in the target, one should use b_1 rather than b_2 to extend the behaviours in target. The new set of behaviours should be examined (using knowledge from the prototype) , to make sure that it does not contain obsolete behaviours. If they do then they should be deleted from the modified problem.

3.5 TRANSFER OF BEHAVIOURS AND TRANSFER OF STRUCTURES INDUCED BY IT

Once the decision which source to use is taken, the set of candidate-behaviours is created by filtering all the behaviours which are in the {BB} list of matching behaviours, out of the set of behaviours of the source problem: $\{B_{\mathbf{C}}\}=\{B_{\mathbf{S}}\}-\{BB\}$. In graphical form this corresponds to making a list of all the hexagons from the source graph , which do not have dashed double arrow pointed on them, to the target graph, Figure 10.
Then for each behaviour from the list of candidates we compute the ratio of matched structure variables it depends on to the total number of structure variables it depends on. Then we try to pass the behaviours which have the highest values of this ratio. The assumption behind this choice of preferences of new behaviours to be added to the target problem is that it is more probable that the behaviours which strongly depend on the matched structure variables would require less additions and modifications to structure (see previous section) and would be more `` more useful.
The behaviour-candidates to be passed to the target problem should be also evaluated with respect to their similarity to the behaviour variables which are already present in the target problem. The comparison can be executed by finding the closest superordinate in some hierarchical tree of knowledge

labels. The behaviours which are judged to be ``too" similar by this procedure should be rejected from the list of possible new behaviours. The reason for this is clear – we assume that it is more likely that ``different" new behaviours would be useful.

Once the decision about which behaviour *b* from the list of candidates $\{B_{\mathbf{C}}\}$, should be passed to the target is taken, the process of its ``transplantation" together with everything in the source structure which supports this behaviour is executed. In order to determine what does support that task a list of elements on which it depends is retrieved from the description of the source problem and {E} the list of matching elements is used to substitute the elements from the source problem with the corresponding matching elements from the target problem. Finally, one checks if the new behaviour still depends on any of the elements of the source problem. If it does not then the step 2 of passing new structures from source to target problems is not required and the knowledge construction stage begins. Otherwise the elements of the source design on which new behaviour depends and which have not been substituted with the matching elements of the target design are added to the set of elements of the target design, together with all the attributes and relations which are connected to these elements. Since relations could depend on a number of elements, for each relation which is added to the target design, its list of elements is also pulled up, scanned for the elements of the source problem, which are on the list of matches {E}, and these elements are substituted with those matches. The elements of the source design which have not been replaced with the target matches are again added to the set structure elements of the target design, etc., Figure 10. Graphically, this process consists of copying a hexagon which does not have a double dashed arrow pointing to it from the source structure+behaviour graph to the target; copying all branches that are attached to it. If the copied branch has some entity (attribute and/or relation) attached to it, then either this branch gets attached to the matching entity in the target or this entity is copied into the target, then all the branches attached to it are copied, etc. This process yields a complete self-contained and self-referential modified graph of the target design which contains the new behaviour and the minimal number of attributes and relations that are needed to support it.

If a target's design variable has a matching source design variable then the range of the corresponding variable in the modified target design is a union of both of their ranges.

4. Examples

Let us consider a few illustrative examples of using this approach. Our archive of design spaces contains two design prototypes. One is of a VLSI design problem (Zimmerman, 1997) where a board has to be designed to produce a layout of a given set of elementary units (which assumed to have rectangular shapes) on it. The behaviours here include the total length of net wiring, the area of the board, the total dead space on the board and the heat behaviour (essentially, the condition of temperature stability of the board). The second is of a disposable napkin and/or handkerchief design problem. Here a napkin and/or handkerchief has to be designed. The behaviours here include the foldability, absorption, weight, cost (specific per 1 usage), hyper-allergenity and disposability. We chose a high threshold value ε=0.5 in the condition of structure element's matching (2).

4.1 DESIGNING BUILDING FLOOR-PLAN LAYOUT

Let us consider the facility layout planning problem where a given set of rectangular departments with given areas and given aspect ratios have to be placed into a given building (Meller and Gau, 1996). The distance-based measure of the net amount of traffic flow in the building (we denote it as travel load) is used as a behaviour, jointly with dead area. Following the above-described action plan first the degree of similarity between this target and the first design prototype in the archive (VLSI) is calculated using the algorithm described in Appendix C. It turns out to be the highest possible, ie $\mathbf{SIM}(P_{\mathbf{T}},P_{\mathbf{S}})$=1. It corresponds to the following list of matching elements {E}=({e_{1T} , e_{1S}}, {e_{2T} , e_{2S}},{e_{3T} , e_{3S}}), their respective degrees of similarity are NS(e_{1T} , e_{1S})=2, NS(e_{2T},e_{2S})=1.55, NS(e_{3T},e_{3S})=1.55 (remember that the maximal possible value of NS is 2) and the corresponding matching is shown in Figure 11. The degree of similarity of this target and the napkin design prototype is $\mathbf{SIM}(P_{\mathbf{T}},P_{\mathbf{S}})$=0 (there are no matching elements, the maximal value of NS(e_T,e_S)=0.5 for any structure elements in target and source is well below the chosen threshold 1.5). Since VLSI is the only design prototype in our archive which has sufficiently similar structure to the target's structure, the system choses VLSI design as the source design. The behaviour similarity between VLSI and facility layout prototypes $\mathbf{SB}(B_{\mathbf{T}},B_{\mathbf{S}})$=0.33 because there is only one matching pair of

behaviours {BB}=({b_{1S},b_{1T} }), Figure 11. This leaves two behaviour

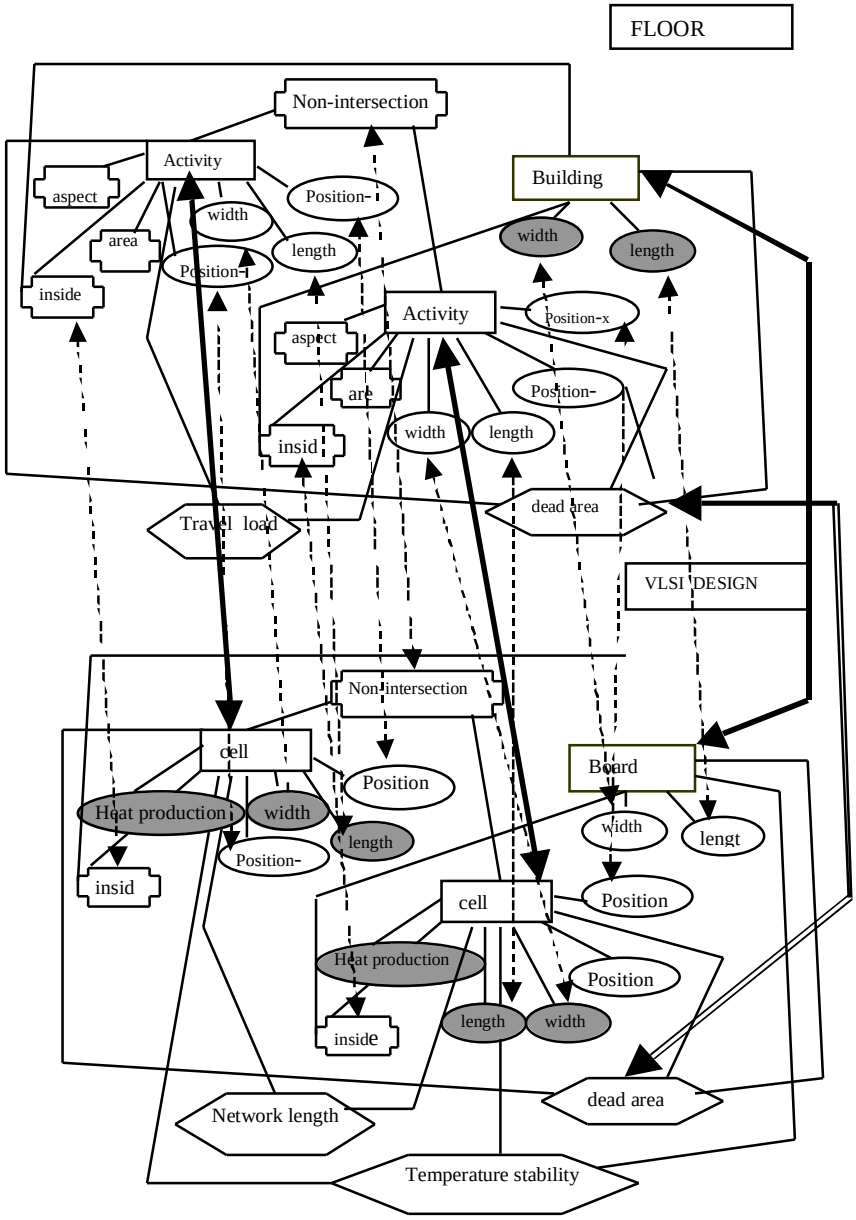


Figure 11. Matching of facility layout and VLSI design prototypes

candidates b_2 and b_3. But the source's b_2 is actually similar to the target's b_2 - they both are distance-based and belong to the same taxonomic tree. Therefore we are left with only one behaviour candidate b_3 to be added to the set of target's behaviours. In order to support it, new attribute a_5 - analogous of the source's attribute a_5 - is added to the lists of attributes of the structure elements e_2 and e_3. Note, that since physical laws which govern heat transfer processes in buildings and in VLSI are different, the knowledge constructed by copying the knowledge from VLSI design prototype needs to be replaced by the domain specific knowledge.

The new behaviour derived for the facility layout problem by analogy with VLSI design, b_3 – heat transfer, does not require the introduction of new structure variables, only new attributes of existing variables. It has the capacity potentially to differentiate between a variaty of layouts and as a consequence provides a pressure to produce designs which would not have been produced previously

4.2 DESIGN OF A NAPPY (DIAPER)

The target problem here is a problem of designing a nappy. Since disposable napkin and handkerchief had been invented much earlier than disposable nappy it is possible that the first disposable nappy may have been designed using this type of reasoning. Again first the degree of structure similarity between the target and each prototype in the archive are computed. It turns out that there is no similarity in structure (at least according to our definition of structure similarity) between this target and the VLSI design, $\mathbf{SIM}(P_{\mathbf{T}},P_{\mathbf{S}})=0$. In other words no matching elements are found, the maximal degree of matching between any structure elements of the target and any structure element of the source is $NS(e_T,e_S)=0.5$. The degree of similarity of this target and disposable napkin design prototype is $\mathbf{SIM}(P_{\mathbf{T}},P_{\mathbf{S}})=0.5$. The corresponding list of matching elements is $\{E\}=(\{e_{1T},e_{1S}\})$ and $NS(e_{1T},e_{1S})=2$, Figure 12. Because the target's structure possesses enough similarity to the napkin's structure and no similarity to the VLSI structure, our system chooses the napkin design as the source problem. The behaviour similarity between napkin and nappy prototypes is $\mathbf{SB}(B_{\mathbf{T}},B_{\mathbf{S}})=1.66$. Two source's behaviours, that do not have matches among target behaviours, make the list of behaviour-candidates for

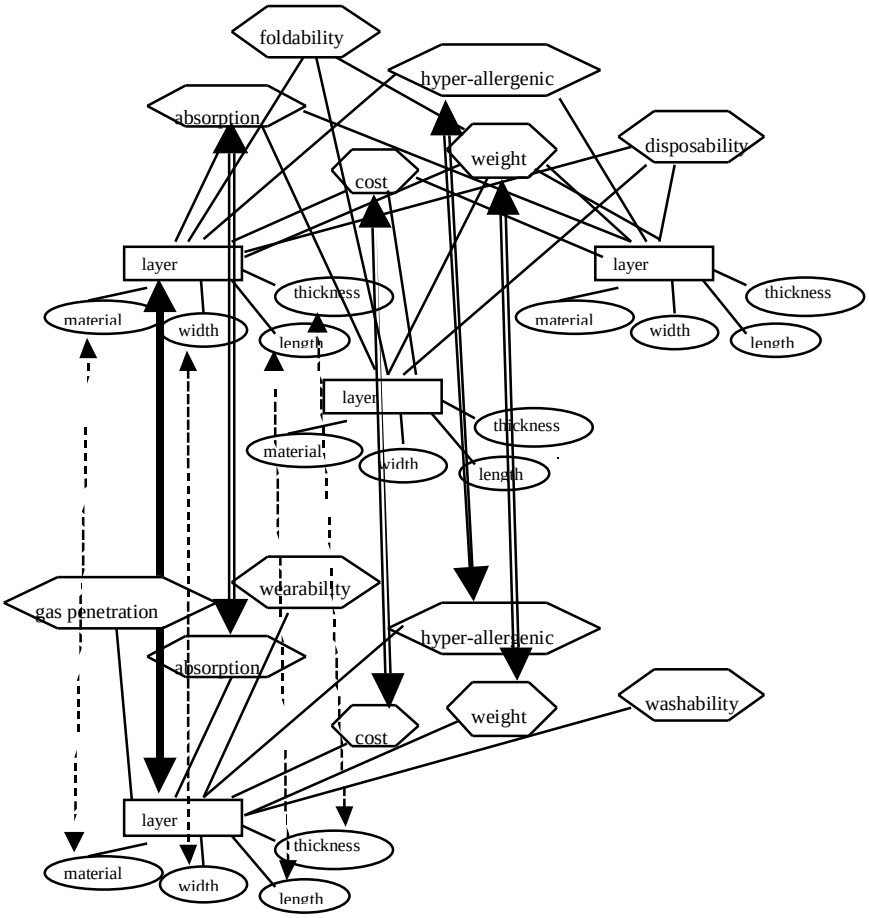


Figure 12. Matching of napkin and nappy design prototypes

transplantation: $\{B_c\}=(b_1, b_6\}$. But b_1 (foldability) is similar to the b_1 (wearability) of the target problem. Hence, it is unlikely that its addition to the target will be productive and it is rejected from the list of candidates. Because not all the elements on which source's behaviour b_6 depends have matches among elements of target's b_6 , the adding of b_6 to the list of target's behaviours does require adding new structure elements (2 new layer) with their corresponding attributes to the target design. This new behaviour can not be supported simply by replacing the ranges of the source's variables with the target's ranges of their matches. Note, that adding disposability to

the range of the target's behaviours makes one of its initial behaviours obsolete and it should be dropped from the list modified target's behaviours.

5. Discussion

The use of analogy in designing has been largely restricted to expending the structure space of designs. The effect of this is to introduce the capacity to produce designs that could otherwise not have been produced. In one sense this matches the notion of creative designing. Given that designing within the S–B–F framework provides the potential to be creative not only in structure space but also in behaviour (and function) space, this paper has presented an approach to the expansion of the behaviour space – an expansion which produces new behaviours in an analogous manner to the expansion of the structure space when using analogy as the expansion process. There could be one of two effects of this expansion. The behaviour space could be expanded without any requirement that the structure space be expanded or the behaviour space could be expanded with a consequential expansion of the structure space.

In the first case different structures can be produced from the same set of structure variables. The differences are driven by the different behaviours the designs now have to exhibit. These designs are novel in the sense that they may involve values of structure variables which would never have been chosen without the additional behaviours. Thus, this process maps onto that of innovative designing. These new designs, however, are unlikely to have been produced using the original behaviours

In the second case different structures can be produced because new structure variables have been introduced as well as new behaviour variables. Both the new behaviours and the new structures drive the differences. The new structures that have been introduced into the target are directly related to the new behaviours. Thus, this process maps onto that of creative designing. These new designs, however, are likely to include those, which could never have been produced with the original behaviours and original structures.

Let us note that approach is routinely used in drug design (Dean, 1995), where molecular similarity (structure similarity in design science terminology) provide a useful lead in the search for bioactive molecules with appropriate properties.

Acknowledgments

This work is directly supported by a grant from Australian Research Council, computing resources are provided through the Key Centre of Design Computing.

References

Carbonell, J.G.:1986, Derivational analogy: a theory of reconstructive problem solving and expertise acquisition, *in*, Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (eds) *Machine Learning: An Artificial Intelligence Approach*, Tioga, Palo Alto, California, pp. 137-161.

Dean, P.M. (ed.):1995, *Molecular Similarity in Drug Design*, Chapman Hall, Glasgow.

French, R.:1995, *The Subtlety of Sameness: A Theory of Computer Model of Analogy-Making*, MIT Press, Cambridge, Mass.

Gero, J.S.:1990, Design prototypes: a knowledge representation schema for design, *AI Magazine* **11**(4): 26-36.

Gero, J.S.:1996, Creativity, emergence and evolution in design, *Knowledge-Based Systems* **9**: 435-448.

Gero, J.S. and Kazakov, V.: 1998, Adaptive expansion of search spaces using emergent features, *in* J. Slaney, G. Antoniou and M. Maher (eds), *AI'98*, Griffith University, Brisbane, Australia, pp.25-36.

Goldschmith, G.: 1994, Visual analogy in design, *in* R. Trappl (ed.), *Cybernetics and Systems '94*, Singapore World Scientific, pp. 507-514.

Holyoak, K.J. and Koh, K.: 1987, Surface and structure similarity in analogical transfer, *Memory and Cognition* **15**(4): 332-340.

Keane, M.T.: 1988, *Analogical Problem Solving*, Ellis Horwood, Chichester.

Keane, M.: 1988, On retrieval analogues when solving problems, *Quarterly Journal of Experimental Psychology* **39A**: 29-41.

Meller, R.D., and Gau. K.: 1996, Facility layout problem: recent and emerging trends and perspectives, *Journal of Manufacturing Systems* **15** (5): 351-366.

Mitchell, M.:1993, *Analogy-Making as Perception: A Computer Model*, MIT Press, Cambridge, Mass.

Navinchandra, D.: 1991, *Exploration and Innovation in Design*, Springer-Verlag, New York.

Prieditis, A.(ed.): 1988, *Analogica*, Morgan Kaufmann, Pitman, London.

Qian, L. and Gero, J.S.,: 1996, Function-behaviour-structure paths and their role in analogy-based design, *AIEDAM* **10**: 289-312.

Thagard, P., Holyoak, K., Nelson, G. and Gochfield, D.: 1994, Analog retrieval by constraint satisfaction, *Artificial Intelligence* **46**: 259-310.

Zimmerman, G.: 1997, CAD tools for VLSI design, http://galway.informatik.uni-de/projects/

Appendix A.

STRUCTURE MATCHING ALGORITH

The algorithm consists of external and internal cycles. The internal cycle is encapsulated as a building block in the external cycle. The internal cycle constructs a set of matching attributes and relations for two given elements. The external cycle produces the set of matching elements. The external cycle has the following structure:

I. (*initialisation*) Set structures $S_{\mathbf{S}}$ and $S_{\mathbf{T}}$ unmarked. Marking means that each entity in a structure is given additional logical variable which can take two values **.false.** if the entity is unmarked and **.true.** if it is marked. Create empty list of matching structure elements.

II. Repeat the following cycle until all the elements in one of the structures $S_{\mathbf{S}}$ and $S_{\mathbf{T}}$ are marked.

A. Take unmarked structure element e_1 from the target prototype.

1. Take unmarked structure element $e_2(k)$ from the target prototype and compute function $NS(e_1,e_2(k))$.
2. Repeat step 1 for all unmarked $e_2(k)$.

B. Find $e_2(i)$ for which $NS(e_1,e_2(i))=\max_k NS(e_1, e_2(k))$.

C. Add the pair $\{e_1, e_2(k)\}$ to the list of matching structure elements. Create a sub-list of matching attributes and relations for $\{e_1, e_2(k)\}$ by copying from the corresponding list produced during computation of NS (see the description of the algorithm which computes NS in the next section).

D. Mark e_1 and $e_2(k)$ and all their attributes.

III. Compute $N(P_{\mathbf{T}},P_{\mathbf{S}})$ by counting as the number N of pairs (e_1,e_2) in the final list of matching structure elements and multiplying it on 2.

Function $NS(e_1,e_2(k))$ is calculated also sub-optimally in a greedy fashion. We assume that the element $e_2(k))$ has p attributes $\{a_1(k),\ldots a_p(k)\}$ and q relations $\{r_1(k),\ldots r_q(k)\}$ associated with it. The corresponding algorithm has the following structure:

I. (*initialisation*) Create two empty lists of matching attributes and relations. All the attributes and relations, connected to e_1 and $e_2(k)$ are unmarked by design (see above).

II. Take unmarked attribute a_1 of the element e_1.

A. Set counter i=1.

B. Take attribute $a_i(k)$. If it is unmarked then
   1. Compare the label of a_1 with the label of $a_i(k)$.
   2. If they match (label{a_1}__label{$a_i(k)$}) then
      a) Add the pair {a_1,$a_i(k)$} to the list of matching attributes.
      b) Mark a_1 and $a_i(k)$.
      c) Go to step III.

C. Sep i=i+1.
D. If i <= p repeat steps II.B-II.D.
E. Mark a_1.

III. Repeat steps II until all the attributes of e_1 are marked.
IV. Compute N(a(1),a(2)) as the doubled length of the list of matched attributes.
V. Take unmarked relation r_1 of the element e_1.

A. Set the counter i=1.
B. Take relation $r_i(k)$. If it is unmarked then
   1. Compare the label of r_1 with the label of $r_i(k)$.
   2. If they match (label{r_1}__label{$r_i(k)$}), then
      a) Add the pair {r_1,$r_i(k)$} to the list of matching relations.
      b) Mark r_1 and $r_i(k)$.
      c) Go to step VI.

C. If i=q mark r_1.
C. Set i=i+1.
D. If i <=q repeat steps IV.B-IV.D.

VI. Repeat step V until all the relations of e_1 are marked.
VII. N(r(1),r(k)) is computing as the double length of the list of matched relations;
VIII. NS(e_1,e_2(k)) is computed as the sum of N(a(1),a(2)) and N(r(1),r(2)). If NS(e_1,e_2) <__ then we set NS(e_1,e_2)=0.

Appendix B.

BEHAVIOUR MATCHING ALGORITH.

The computation of **SB**($B_\mathbf{T}$,$B_\mathbf{S}$) proceeds as follow (it is assumed here that the target has g behaviour variables {b_1(1),…, b_g(1)}):

I. Unmark all the behaviours of both $B_\mathbf{T}$ and $B_\mathbf{S}$. Create an empty list of matching behaviours of the source problem {BB}.
I. Take unmarked behaviour of the source problem b_2.

A. Set counter i=1.
B. If behaviour $b_1(1)$ is unmarked
    1. Compare the label of $b_1(1)$ with the label of b_2.
        a) If they match (label($b_i(1)$)__label(b_2))
            (1) Add $\{b_1(1),b_2\}$ to {BB}.
            (2) Mark b_2 is marked.
            (3) Go to III.
    2. If i=g mark b_2.
    3. Set i=i+1.
    4. If i <=g repeat steps 1-4.

II. Repeat step II until all behaviours in the source are market.

The value of $\mathbf{SB}(B_{\mathbf{T}},B_{\mathbf{S}})$ is then computed as the ratio of the difference between the total number of behaviour variables in $B_{\mathbf{T}}$ and the length of the list {BB}.

Appendix C.

MAPPING AND TRANSFORMATION.

The following algorithm is used to pass new behaviour and its structure support from source to target:

I. the hexagon is copied from the source to the target; Unmark all the branches.
II. If branch **br** which is attached to it is unmarked then copy it into the source;
A. if such branch in source has an oval attached to it then
    1. if this oval is connected with target's oval with dashed arrow this branch in target gets attached to this oval;
    2. otherwise
    b) the oval is copied from source to the target and placed in the end of new branch;
B. if such branch in source has a cross attached to it;
    1. if this cross is connected to the cross in a target with a dashed arrow
        a) this branch in target gets connected to this matching cross;
    2. otherwise

a) this cross is copied from source to target at the end of this branch;
b) if this cross in source has other branches attached to it;
(1) apply the continuation of this process of either attaching branches to the matching entities or adding unmatched entities, adding branches which go out of these entities, etc. until there is no hanging branches without element, relation or attribute on both ends.

III. Mark **br**.
IV. Repeat steps II-III until all branches are marked.

This process yields a complete self-contained and self-references modified graph of the target problem is built which contains the new behaviour and the minimal number of all its attributes and relations that are needed to support it.

Appendix D.

NOMENCLATURE

S - the structure space,
B - the behaviour space,
F - the function space,

$\mathbf{D}_{-}$ - the complete design state space,
D – the feasible subset of **D** ,
$\mathbf{D}^{\mathbf{m}}$ – the feasible subset of the modified **D** ,
$\mathbf{B}^{\mathbf{m}}$ - the behaviour space of the modified design,
$\mathbf{S}^{\mathbf{m}}$ - the behaviour space of the modified design,
$\mathbf{F}^{\mathbf{m}}$ - the function space of the modified design,

$\mathbf{S}_{\mathbf{T}}$ – the feasible subset of the target design **S**,

$\mathbf{S}_{\mathbf{S}}$ – the feasible subset of the source design **S**,
SIM -the real function which measures the degree of similarity of two structure spaces.

τ-analogical transformation operator,

M-analogical mapping operator,

P-design prototype,
B - a finite set of behaviour variables,
Ex - a finite set of exogenous variables which describe external conditions,
F -a finite set of function variables,

K_c - computational knowledge,

K_r - relational knowledge,

K_q - qualitative knowledge,
S- a finite set of structure variables.

SB($B_{\mathbf{T}}$,$B_{\mathbf{S}}$) - a function which measures the degree of similarity of behaviour spaces,
NS(e_1,e_2)- a function which measures the degree of similarity of two structure elements,
{E}- a list of matching structure elements in two design prototypes,
{BB}-a list of matching behaviours in two design prototypes,

{$\mathrm{B}_{\mathbf{C}}$}-a list of behaviour candidates to be transferred from source to target.

This paper is a copy of: Gero, J. S. and Kazakov, V. (1999) Using analogy to extend the behaviour state space in creative design, *in* J. S. Gero and M. L. Maher (eds), *Computational Models of Creative Design IV*, Key Centre of Design Computing and Cognition, University of Sydney, Sydney, Australia, pp. 113-143.