Artificial Intelligence Review **16:** 87–118, 2001.

Efficient Case-Based Structure Generation for Design Support

KATY BÖRNER
Indiana University, SLIS, Bloomington, IN 47405, USA (E-mail: katy@indiana.edu)

Abstract. This paper describes a general approach to support *case-based structure generation* named *Conceptual Analogy* (Börner 1997). The approach can be used to support design tasks in domains that do not allow the acquisition of a complete and consistent set of constraints or rules but that do provide a larger set of past experiences. The experiences – also called cases – may resemble for example CAD data of room layouts or pipe systems. Thus, cases represent solutions to particular problems without an explicit, a priori separation of problem and solution variables. The paper starts with a description and formalization of the *case-based structure generation* task and a discussion of why standard case-based reasoning and other approaches are not applicable. Then, the new approach of *Conceptual Analogy* (CA) is introduced. CA applies conceptual clustering to organize cases represented by graphs into hierarchical classes of *structurally similar* cases. These case classes are then represented by *concepts*. Given a problem, i.e., a partial solution, the hierarchy of concepts is searched for *applicable* concepts, i.e., concepts that allow the generation of at least one solution to the given problem. Applicable concepts are used to generate a set of solutions that can be ordered according to their *quality*. Properties of the approach as well as complexity results are presented. An architectural design domain and task where the approach has been applied successfully, is used for illustration and for practical evaluation. Finally, the approach and its implementation are compared to two systems that aim at the support of similar design tasks. The paper concludes with an assessment of the future direction of this research.

Keywords: analogical reasoning, case-based design, case-based reasoning, conceptual clustering, concept representation, structural similarity

1. Introduction

Design tasks are inherently complex. Oftentimes, it is not possible to acquire a complete and consistent set of constraints or rules. This might be due to a poor ratio of knowledge acquisition time to the time saved by using the system or to the fact that domain experts might have a hard time externalizing their knowledge. Therefore, rule-based approaches or constraint satisfaction techniques cannot be applied to support human designers. For some tasks, a set of prior experiences – so-called cases – is available and case-based reasoning (CBR) (Kolodner 1992) seems to be the natural problem solving method. In CBR, domain knowledge is stored in four knowledge containers as identified by (Richter 1995): the *case base*, the *vocabulary used*

to represent cases, the *similarity measure*, and *solution transformations*. We assume that the knowledge required to evaluate solutions is included in the container for *solution transformation*. Reasoning proceeds via the CBR cycle comprising *retrieve, reuse, revise,* and *retain* as described in (Aamodt and Plaza 1994).

However, in some architectural design tasks, which we call *case-based structure generation* tasks, the knowledge about the similarity between a new and a previous problem or about valid adaptations (solution transformations) of past cases is hard or impossible to acquire. This leaves *cases*, their *vocabulary*, and general *quality* criteria such as *'solutions should resemble past cases as close as possible'* as the only knowledge available for design support. Even worse, cases resembling CAD designs of buildings represent solutions only. Information about which part of the solution represents the original problem is lost. The question that this paper tries to answer is: Is there a way to support *case-based structure generation* tasks efficiently?

The paper starts with a description and formalization of *case-based structure generation* tasks. It explains why existing approaches cannot be applied efficiently. Based on this, the approach of *Conceptual Analogy* is introduced. Basic notions and notations are given and knowledge organization and analogical reasoning are explained. Main features of the approach are presented and complexity results are discussed. Following this, the implementation of the approach and its application in the domain of architectural design is demonstrated. Finally, we relate the approach to other research in case-based design and conclude with an outlook.

2. Case-based structure generation

Design is concerned with the composition of an artefact from single parts that may be either known and given or just newly created. Constraints on the artefact may be rigidly or informally defined. In design, experts refer to past cases frequently and research in case-based design is growing continuously (Oxman and Voß 1996; Maher and Pu 1997; Voß 1997b; Börner 1998a).

Subsequently, we introduce a specific design task, named *case-based structure generation*. It has a number of features that differ from standard case-based reasoning tasks.

First of all, cases that resemble CAD drawings of built houses represent solutions to particular problems without an explicit, a priori separation of problem and solution variables.

Second, problems are completed by (partial) transfer of past cases showing a high structual similarity. However, design solutions are hardly ever identical. Frequently, two or more previous solutions must be combined to

solve a new problem. Therefore, retrieval has to supply cases that can be combined to solve the current problem.

Third, in order to design a new room layout, pipeline system, roof construction, etc. the topological relationships between building elements such as the rooms, the pipes, or the roof parts must be considered in order to retrieve and properly combine a set of past designs. That is, design cases need to be represented structurally (e.g., as graphs) to facilitate structural match and structural combination. The match and combination of structural case representations poses serious computational problems. Fortunately, many topological structures can be represented by trees (as opposed to graphs) which reduces the time spent for match and case combination. Oftentimes, a central, unique building element can be identified for a set of cases that support one specific task type. For room layout, the entrance area might be unique. Pipe systems typically have just one main outlet etc. These unique elements can be represented by the unique tree root. This reduces the computational effort required to map two cases or a case and a problem. The structural match between cases resembles the discovery of identical connected subgraphs (called components) of trees. Aiming at the preservation of the main structure, we require that this common structure contains the central, unique building element that is represented by the tree root node.

Fourth, it is assumed that structurally similar cases, i.e. cases that have a large identical connected subgraph, meet all constraints of a particular task type and can therefore be combined.

Fifth, transferring larger structures minimizes the problem of bad case combinations. Therefore, we require, that each path – from root note to leave – in the final solution has to resemble a path – from root note to leave – in one of the original cases. To ensure this, each set of cases that can be used to generate a new design solution has to meet a so called *valence criterion*. We will come back to this in Section 2.1.

Sixth, solutions are correct w.r.t. the cases combined. If a solution can be generated by transferring a single past case – which is very rare in building design – the solutions would receive a quality of one meaning: *This design has been built and can be built again.* If several past designs have been combined to generate a solution, its *quality* increases with its resemblance to the common structure of cases combined. Note that external constraints which are not represented in a case may result in a non-acceptance of the solution by the user.

Furthermore, we can assume that in the selected domain and task there is usually enough memory space and preprocessing time available. However the problem solving time has to be minimized.

2.1. *Formalization of case-based structure generation*

The verbal description of *case-based structure generation* tasks can be formally defined. Subsequently, we formalize the notion of case, case-base, problem, solution, solution set, similarity measure, and solution quality.

We use the following graph theoretic notions: A *graph* $g = (V^g, E^g)$ is an ordered pair of vertices V^g and edges E^g with $E^g \subseteq V^g \times V^g$. A *set of graphs* is denoted by G. The *combination graph* $g^{(G)}$ of a set G of graphs equals the union of the vertices/edges of the graphs in G:

$$g^{(G)} = (\bigcup_{i=1}^{|G|} V^{g_i}, \bigcup_{i=1}^{|G|} E^{g_i}) = (V^{(G)}, E^{(G)}).$$

Formally, a *structurally represented case* $c = (V^c, E^c)$ is a tree containing a unique root node v_R. A *case base* CB is a finite set of cases. A *problem* provides a set of vertices including the same unique v_R and perhaps some edges, i.e., in graph terminology it is a forest. A *solution* of a problem contains the problem vertices and edges and adds those and only those vertices and edges from structurally similar cases that are necessary to connect all problem vertices and edges.

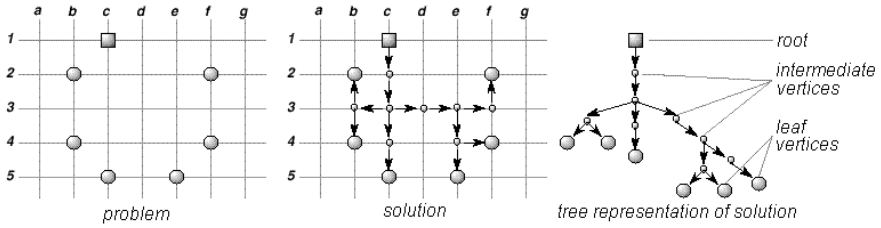

Figure 1. Graph-based representation of problem and solution.

Example: A graph-based represented problem (a partial design of a pipe system) and its solution (a final design) are depicted in Figure 1, left and middle. Both are mapped onto an equally spaced grid. The main access of the pipe system – represented by a square – is mapped onto the root vertex, denoted by v_R. Outlets – represented by circles – correspond to leaf vertices V_L. Pipes – represented by arrows – are mapped onto edges E. Intermediate vertices V_I represent locations along the pathway. Figure 1, right illustrates the tree structure of the solution. Note that different domains and tasks may require a different mapping of objects onto tree representations.

Given a new problem, only cases that show a high structural similarity are combined to generate its solution. *Structural similarity* of a graph set G can be

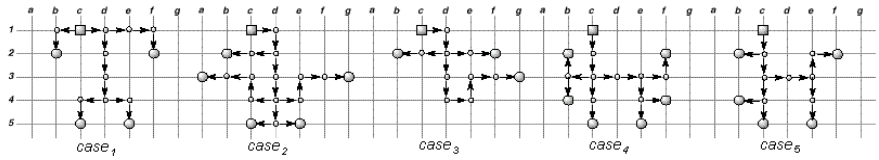

Figure 2. Graph-based representation of five cases.

defined as the quotient of the number of edges of the common graph divided by the number of edges of the combination graph $CG(G)$. The *common graph* $MG(G)$ of a set of graphs G is defined as the connected subgraph of the intersection of its edges and vertices. A graph is called connected if there is a path connecting every pair of vertices. A set of graphs can have several connected common subgraphs with an equal number of edges. In order to derive unique common graphs we require the common graph to contain the unique root node v_R.

$$MG(G) = (V^{mg}, E^{mg}) \textit{ with } (V^{mg}, E^{mg}) \textit{ is a connected subgraph of } (\bigcap_{g\in G} V^g, \bigcap_{g\in G} E^g) \ \wedge \ v_R \in V^{mg}.$$

The *structural similarity* measure σ then maps a set G of graphs into the interval [0, 1].

$$\sigma(G) := \begin{cases} 0 & if \ \ |\bigcap_{g\in G} E^g| = 0 \\ \frac{E^{mg}}{|\bigcup_{g\in G} E^g|} & otherwise \end{cases}$$

This similarity measure is also supported by psychological literature defining the similarity of two objects by the percentage of matching attributes, i.e., the *number of matching attribute values* divided by the *total number of attributes* considered. An example is the *Hamming distance* (Hamming 1980), one of the most commonly used similarity measures for binary attributes.

Note that this similarity measure defines the similarity of a set of cases and not between two problems or a problem and a case as in standard CBR. The similarity measure can therefore be seen as a cohesion measure (see the discussion in section 3.4.1).

Based on this, a *case class* CC can be defined as a non-empty subset of a case base CB that groups cases of high structural similarity. In particular, we are interested in so-called *valid case classes* that can be applied to generate solutions. The set of case classes showing a high structural similarity, named $C(CB)$ can be defined recursively. It equals all those elements of the power-set $\mathcal{P}(CB) = 2^{|CB|} = \{\emptyset, C_1, \ldots, C_{2^{|CB|}-1}\}$, which

1. are *not empty*,
2. show a *high similarity*, i.e., they have only one element or correspond to unions of similar, disjunct case classes $CC_k \cup CC_l$ with $CC_k \cap CC_l \neq \emptyset$:
   (i) $\{CC \mid CC \in \mathcal{P}(CB) \ \wedge \ |CC| = 1\} \subseteq C(CB)$
   (ii) $\{CC_k \cup CC_l \mid CC_k, CC_l \in C(CB) \wedge \ \nexists CC_m \in C(CB)$
   $(((CC_k \cup CC_l) \ \cap CC_m = \emptyset) \ \wedge \ (\sigma(CC_k \cup CC_m) > \sigma(CC_k \cup CC_l)))\} \subseteq C(CB)$,
3. and have a combination graph $(V^{(CC)}, E^{(CC)})$ that contains no inner vertex with a *valence d* larger than one:
   $CC(CB) = \{CC \mid CC \in C(CB) \wedge \ \nexists v \in V_I^{(CC)}(d(v) > 1)\}$.

Note that all *case classes* in $C(CB)$ meet criterion 1. and 2., but not the *valence criterion*. The 3rd criterion, named valence criterion minimizes bad case combinations. It ensures that the *valence* of each intermediate vertex, which equals the number of edge ends at that vertex, is not larger than one. In order to be applicable to generate new solutions case classes must meet all three criteria.

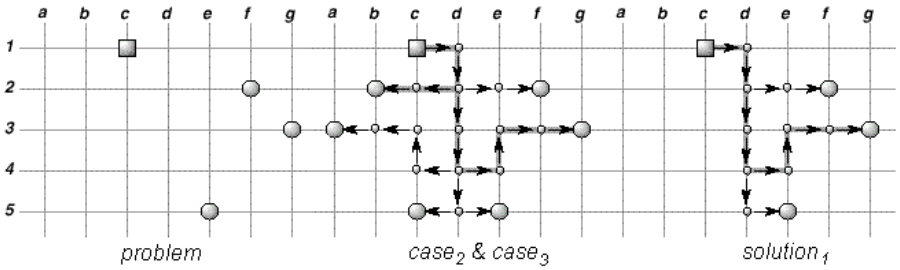

Figure 3. Problem, combination graph of $case_2$ and $case_3$ and $solution_1$.

Example: Given five cases as depicted in Figure 2 and a new problem as depicted in Figure 3, left, the most similar cases must be selected. None of the five concrete cases can be used to generate a problem solution. However, the combination graph of $case_2$ and $case_3$ (see Figure 3, middle) can be applied to derive $solution_1$ (see Figure 3, right). Usually, more than one solution exists. For example, by combining $case_1$ to $case_3$ three other problem solutions can be obtained (see Figure 4).

The set of all solutions for a problem p, that can be generated by combining cases in case class CC is called solution set $S_{CC,p}$. The set of all solutions that can be generated using the entire case base CB equals the union of all solution sets derived from valid case classes $CC(CB)$ in CB and is denoted by $S_{CB,p}$.

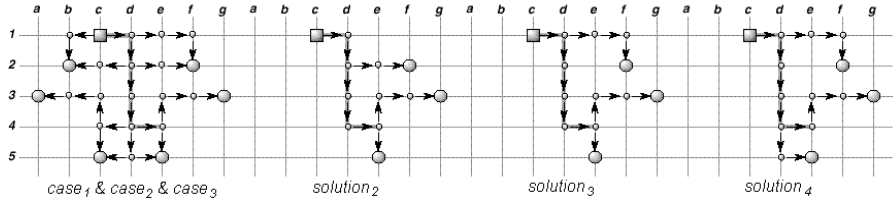

Figure 4. Combination graph of $case_1$ to $case_3$ and $solution_2$ to $solution_4$.

Finally, the set of solutions is ordered according to their quality μ where the quality is defined by the relative frequency of solution edges w.r.t. the case class CC applied. In general, the *relative frequency* P_G of an edge (v_i, v_j) of the combination graph of G equals the number of graphs in G containing this edge divided by the number of graphs in G:

$$P_G((v_i, v_j)) := \frac{|\{g \in G \mid (v_i, v_j) \in E^g\}|}{|G|}.$$

The *relative frequency* P_G of a set of edges E relative to a set of graphs G equals:

$$P_G(E) := \frac{1}{|E|} \sum_{(v_i, v_j) \in E} P_G((v_i, v_j)).$$

Based on this, the quality μ of a solution $s = (V^s, E^s)$ generated using a case class CC is defined as:

$$\mu(CC, s) := P_{CC}(E^s) = \frac{\sum_{i=1}^{|CC|} |E^s \cap E_i^{(CC)}| * \frac{i}{|CC|}}{|E^s|} \in [0, 1].$$

Note that the information about the problem part as well as the CC applied to generate the solution is not stored in the case base. Whenever the user accepts a certain solution its quality value is set to one. That is, all cases in the case base have a quality of 1.

Example: The cases shown in Figure 2 can be used to generate four solutions that are depicted in Figures 3 and 4. $Solution_1$ consists of 12 edges. Eight edges have a relative frequency of one. The remaining four edges have a relative frequency of 1/2. Thus, $solution_1$ has a quality of

$$\mu(\{case_1, case_2\}, solution_1) = \frac{8 * 1 + 4 * \frac{1}{2}}{12} = \frac{5}{6}.$$

Solutions $solution_2$ to $solution_4$ have a quality of $\frac{24}{33}$, $\frac{25}{36}$, and $\frac{23}{39}$ respectively.

2.2. *Why standard case-based reasoning cannot be applied*

Given that cases are the main repository of knowledge to solve *case-based structure generation* tasks, CBR seems to be the appropriate problem solving method. Compared to other tasks, case-based structure generation tasks show the following distinctive characteristics:

- Cases represent solutions to particular problems without an explicit, a priori separation of problem and solution variables.
- The structure of cases is important and thus cases are represented by *graphs*.
- Hardly any information about the relevance of features guiding the selection of similar cases is available.
- *Retrieval* has to return a valid case class that can be *combined* to generate solutions.
- *Similarity* is not defined for a pair of problems but for a set of cases. It can be seen as a cohesion measure (see the discussion in section 3.4.1).
- *Adaptation* mainly corresponds to adding, eliminating, or combining objects and their relations. In general, there exists more than one solution.
- The *quality* of solutions equals the relative frequency of solution edges w.r.t. the case class applied.

Taken together, standard approaches to CBR are not applicable to solve this task.

2.3. *Why graph-based approaches cannot be applied*

In *case-based structure generation* tasks, the case selection, the case combination, and the solution evaluation proceeds via graph-based case representations. Therefore, graph algorithms can be applied to test the validity of 2^N, $N = |CB|$ case classes to generate all solutions and to order them w.r.t. their quality μ. That is, the application of graph algorithms would lead to a *correct* set of solutions. The memory space used to store the concrete cases would be *linear* in N. However, the problem solving time required to search for valid case classes would be *exponential* in N, and thus not feasible for real world applications (see also the discussion in subsection 3.4.5).

3. Conceptual Analogy

Conceptual Analogy (CA) is a general approach that was developed to support *case-based structure generation* tasks efficiently. It relies on conceptual clustering and the representation of valid case classes by concepts

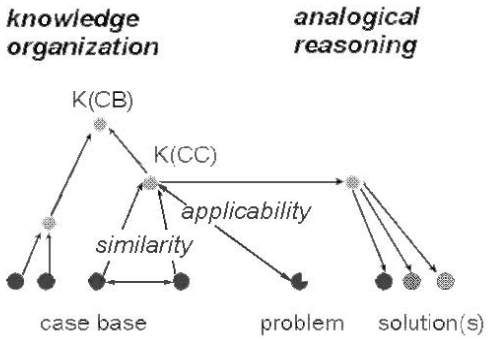

Figure 5. Knowledge organization and analogical reasoning.

to facilitate the efficient selection and combination of cases in analogous situations.

The basic idea is illustrated in Figure 5. During knowledge organization, conceptual clustering is applied to organize cases hierarchically into case classes CC. Subsequently, each class of similar cases is represented by a concept $K(CC)$.

In the second stage, which resembles analogical reasoning, the concept hierarchy is searched in a top-down manner for applicable concepts. These concepts are applied to generate, evaluate, and explain solutions. Both subtasks process graphs but are grounded on attribute value representations of cases (see section 4).

Central to memory organization and resoning is finding the relational match between cases and the new solution. Instead of mapping attribute values, the topological relationships between objects that play a similar role (e.g. outlets, rooms etc.) need to be mapped. Objects and relationships are transferred from the cases in the case base to the new solution. Therefore, this structural match and subsequent structure combination can be termed an 'analogical mapping and transfer'. In fact, the heart of several approaches to analogical reasoning is the discovery of mappings between hand selected, properly represented target and source samples. An example is the *Solar system/Rutherford atom* analogy or the *Water-flow/Heat-flow* analogy by (Falkenhainer et al. 1986). CA goes beyond most analogical reasoning approaches in that it establishes a structural mapping between several previous cases and one partial solution. In addition, it transfers not just one source case but combines and adapts past solutions if necessary. Last but not least, it evaluates the quality of the generated solutions.

We proceed by introducing the details of CA's knowledge organization and analogical reasoning and present a formalization of used terminology in section 3.3.

Table 1. Similarity metric for CC_1 to CC_5

$\sigma(CC_i \cup CC_j)$	CC_1	CC_2	CC_3	CC_4	CC_5
CC_1	1	6/24	5/20	0	0
CC_2		1	10/19	0	0
CC_3			1	0	0
CC_4				1	8/18
CC_5					1

3.1. *Knowledge organization*

Knowledge organization starts with a case base $CB = \{c_1, \ldots, c_N\}$, where $N = |CB|$, that provides a significant number of structurally represented cases as well as the structural similarity measure σ (see definition in section 2).

Nearest-neighbor-based, agglomerative, unsupervised conceptual clustering is applied to create a hierarchy of case classes grouping cases of similar structure. Conceptual clustering starts with a set of singleton vertices representing *case classes*, each containing a single case c_i, $i = 1, ..., N$, represented by a graph. The set of all singleton case classes is called *partition* CCP^0 (see definition in section 3.3). The similarity matrix of all case classes in the partition is determined and the two most similar case classes CC_k and CC_l over the entire set are merged to form a new case class $CC = CC_k \cup CC_l$ that covers both. The two merged case classes are deleted from the actual partition and the newly formed case class is added. This process is repeated for each of the remaining $N - 1$ case classes. Merging of case classes continues until a single, all-inclusive cluster remains. Information about which case classes have been merged is stored as well as the similarity values of the resulting case classes. At termination, a uniform, binary *hierarchy of case classes* is left.

Subsequently, each case class is represented by a concept. Case classes that do not meet the valence criterion are represented by an empty set. All other concepts equal $n = |CC|$ (possibly empty) graphs showing the same relative frequency of their edges relative to the cases in CC.

Example: Table 1 shows the similarity metric of CC_1 to CC_5 representing $case_1$ to $case_5$ depicted in Figure 2. The similarity of each case to itself is one. The common graph of CC_4 and any other case class contains no edge resulting in a similarity of zero. Case classes CC_2 and CC_3 show the highest similarity of 10/19 and are merged to CC_6.

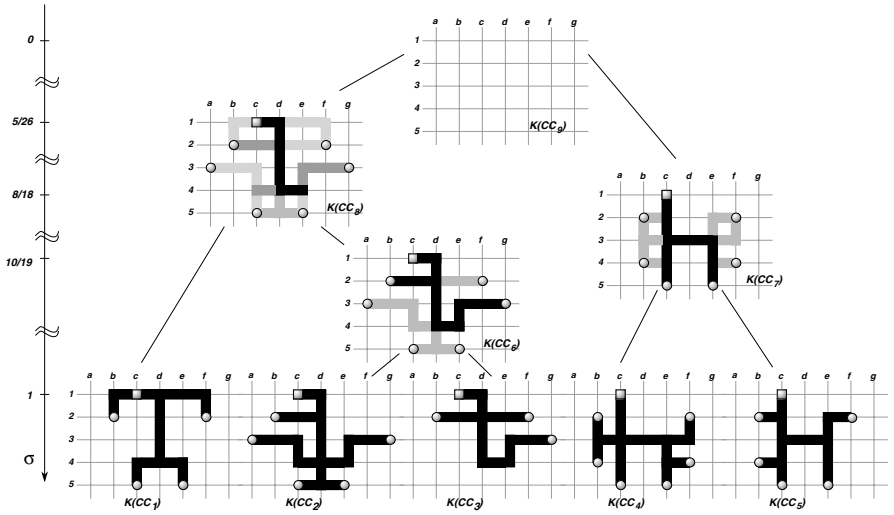

Figure 6. Concept hierarchy representing case classes CC_1 to CC_9.

Figure 6 shows the organization of the five cases into a concept hierarchy. N cases are represented by $2N - 1$ case classes respective concepts $K(CC)$. Leaf vertices in the concept hierarchy correspond to concrete cases and are represented by the cases themselves, e.g., $K(CC_i) = \{c_i\}$ with $i = 1 \ldots 5$. Generalized concepts in the concept hierarchy are labeled $K(CC_6)$ to $K(CC_9)$. $K(CC_6)$ to $K(CC_8)$ are characterized by sets of graphs. Case class no. 9 does not fulfill the valence criterion. Its concept $K(CC_9)$ equals the empty set. Edges with a relative frequency of one are shown in black, edges with less relative frequency are drawn in grey. The similarity of each concept is given on the left hand side. As for concept $K(CC_6)$, ten edges with a relative frequency of one are divided by 19 edges altogether, resulting in a similarity of 10/19.

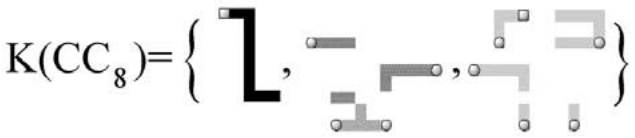

Figure 7. Concept representation of case classes CC_8.

Concept $K(CC_8)$ that represents $CC_8 = \{case_1, case_2, case_3\}$ and comprises three graphs with a relative frequency of their edges of 1, 2/3, and 1/3 is illustrated graphically in Figure 7. The relative frequency of solution edges is visualized by different hues of grey (black standing for a relative

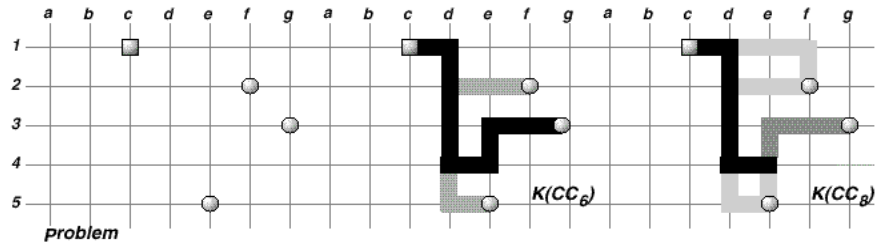

Figure 8. Application of $K(CC_6)$ to solve p_1.

frequency of 1). It can be applied to generate $case_1$ to $case_3$ as well as combinations thereof.

3.2. *Analogical reasoning*

Analogical reasoning is based on concepts exclusively. Given a new problem, the concept hierarchy is searched for applicable concepts in a top-down fashion. If the list of applicable concepts is not empty then the concepts are applied subsequently until all solutions are generated or all applicable concepts are used. Finally, solutions are evaluated and ordered w.r.t. their *quality*. Let us have a look at an example.

Example: Figure 8, (left) depicts the $problem$ from Figure 3. Exploiting the concept hierarchy shown in Figure 6 the search for applicable concepts starts on the top, i.e., the applicability of $K(CC_9)$ is determined first and returns -1.

Subsequently, the examination of concept $K(CC_8)$ returns 5/26; $K(CC_1)$ returns -1; $K(CC_6)$ returns 10/19; $K(CC_2)$ returns -1; $K(CC_3)$ returns -1; $K(CC_7)$ returns -1. Concepts $K(CC_4)$ and $K(CC_5)$ are not considered because their superordinate concept is not applicable. Exactly two concepts: $K(CC_6)$ and $K(CC_8)$ are selected. They are depicted in Figure 8, middle and right. Vertices and edges that are not needed to connect all problem vertices were eliminated.

One problem solution can be generated by applying $K(CC_6)$. Three more solutions are generated by applying $K(CC_8)$. All four solutions are depicted in Figure 9. Note that they resemble $solution_1$ to $solution_4$ in Figures 3 and 4. Their quality values are $\frac{5}{6}$, $\frac{24}{33}$, $\frac{25}{36}$, and $\frac{23}{39}$ respectively.

Again, the relative frequency of solution edges is visualized by different hues of grey (black standing for a relative frequency of 1). Note that all applicable, non-empty superordinate concepts of an applicable concept allow the generation of the solutions of their subordinate concepts with less or

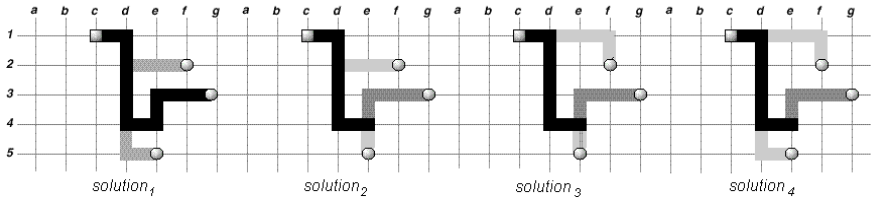

Figure 9. Resulting solutions.

equal quality values. That is, $solution_1$ can be derived by applying concept $K(CC_8)$. However, the solution quality would be $8 * 1$ plus $4 * 1/3$ divided by 12, i.e., $\frac{7}{9}$ instead of $\frac{5}{6}$.

3.3. *Formalization of Conceptual Analogy*

The approach of Conceptual Analogy can be formalized mathematically as follows: Each partition CCP of a case base CB – generated during memory organization by conceptual clustering – groups cases into certain case classes based on their similarity and is defined as follows:

$$CCP = \{CC_i \mid \bigcup_i CC_i = CB \wedge \forall i, j \forall c_1, c_2, c_3 (i \neq j \wedge (CC_i \cap CC_j = \emptyset) \wedge$$

$$((c_1, c_2 \in CC_i \wedge c_3 \in CC_j) \rightarrow \sigma(\{c_1, c_2\}) \geq Max_{l \in \{1,2\}} \sigma(\{c_l, c_3\})))\}$$

where σ is the similarity measure defined in section 2 and Max returns the maximal similarity value.

The resulting hierarchy PH of partitions equals:

$$PH = (CCP^0, CCP^1, \ldots, CCP^{N-1}) \;\; with \;\; |CCP^{v-1}| - 1 = |CCP^v|.$$

Partition CCP^0 contains $N = |CB|$ case classes. All other partitions CCP^v have one element less than the previous partition CCP^{v-1}, i.e., $|CCP^{v-1}| - 1 = |CCP^v|$.

The set of all case classes in PH equals the set $C(CB)$ of non-empty case classes of high structural similarity:

$$C(CB) = CCP^0 \cup CCP^2 \cup \ldots \cup CCP^{N-1}.$$

Note that $C(CB)$ contains *case classes* that do not meet the *valence criterion* as defined on page 89.

In order to decrease the complexity of case class selection and solution evaluation, each case class CC is represented by a *concept* $K(CC)$. Case classes that have a combination graph which contains inner vertices with

a *valence* larger than one, i.e., that are not valid, are represented by an empty set. All other case classes are represented by a *concept* $K(CC) := \{m_i^{(CC)} \mid i = 1, ..., |CC|\}$, where the vertices and edges of the graphs $m_i^{(CC)} = (V_i^{(CC)}, E_i^{(CC)})$ equal

$E_i^{(CC)} = \{(v_l, v_k) \mid (v_l, v_k) \in E^{(CC)} \wedge P_{CC}((v_l, v_k)) = \frac{i}{|CC|}\}$,

$V_i^{(CC)} = \{v \mid \exists (v_l, v_k) \in E_i^{(CC)} \ (v = v_l \vee v = v_k)\}$.

Thus each graph shows an identical relative frequency of its edges. Figure 7 showed an example concept representation.

Note that the set of non-empty concepts equals the set $CC(CB)$ of valid case classes defined in section 2. That is, all non-empty concepts can be applied to generate solutions. The concrete cases are stored to enable the dynamic reorganization and update of concepts.

During analogical reasoning, concepts representing case classes of high structural similarity can potentially generate solutions of higher quality and should be preferred. Correspondingly, the *applicability* α of a concept $K(CC)$ to solve a problem $p = (V^p, E^p)$ is defined as

$$\alpha(K(CC), p) := \begin{cases} -1 & \text{if } V^p \not\subseteq V^{(K(CC))} \vee E^p \not\subseteq E^{(K(CC))} \\ \sigma(CC) & \text{otherwise} \end{cases}$$

Whenever $0 \leq \alpha(K(CC), p) \leq 1$ holds, then at least one solution of p can be generated by applying $K(CC)$. In general, there exists more than one applicable concept. Applicable concepts can be ordered w.r.t. their α value. The concept showing the highest α value is called the *most applicable concept*. It shows the highest structural similarity and solves the problem.

We use an applicability measure instead of the already defined similarity measure because problems may not contain any edges and thus their structural similarity to any set of cases could be zero. In addition, case classes with a high similarity are typically too concrete to generate a solution.

Instead of *adapting* one or more cases to solve the problem, the concept representation $K(CC)$ of a case class CC is used to generate a set of adapted solutions $S_{CC,p}$ for a problem p. The set of all solutions $S_{CB,p}$ of a CB for a problem p equals the union of solution sets $S_{CC,p}$.

Finally, the set of solutions is ordered w.r.t. their *quality*. The quality μ of a solution $s = (V^s, E^s)$ depends on the relative frequency of its edges w.r.t. the case class used. In order to determine $P_{CC}(E^s)$, every solution edge E^s has to be mapped with every edge of a case. CA's concepts which represent each case class by graphs showing an identical relative frequency of their edges:

$$K(CC) = \{(V_1^{(CC)}, E_1^{(CC)}), \ldots, (V_{|CC|-1}^{(CC)}, E_{|CC|-1}^{(CC)}), (V^{MG(CC)}, E^{MG(CC)})\}$$

allows one to determined solution quality efficiently:

$$\mu(K(CC), s) := \frac{\sum_{i=1}^{|K(CC)|} |E^s \cap E_i^{(CC)}| * \frac{i}{|K(CC)|}}{|E^s|} \in [0, 1].$$

Note that this definition of μ is equivalent with the definition given in section 2.1.

3.4. *Discussion*

The remainder of this section discusses features that are unique to the approach of *Conceptual Analogy*. Among these features are the *structural similarity measure*, the *concept representation*, the *types of adaptations*, and the *set of potential solutions* that can possibly be derived from a case base. Last, but not least, complexity results are presented.

3.4.1. *Structural similarity*

The similarity measure used by *Conceptual Analogy* is defined over sets of cases represented by graphs (Börner 1993; Jantke 1994). Therefore, it can be seen as a *cohesion measure* that gives an impression of the unity of a set of cases.

Corresponding to the definition given on page 89, the similarity of a graph set G equals the quotient of the number of edges of the common graph $MG(G)$ divided by the number of edges of the combination graph $CG(G)$. Let B be the set of all trees with root vertex v_R and $G, G\prime$ be subsets of B. $\mathcal{P}(B)$ refers to the power-set of B. If $a, b \in B$ then $b \sqsubseteq a$ denotes that b is either identical or a subgraph of a.

The *common graph mg* of two graphs $a, b \in G$ is defined as:

$$MG'(a, b) = (V^{mg}, E^{mg}) \; with \; (V^{mg}, E^{mg}) \; is \; a \; connected \; subgraph \; of$$

$$(V^a \cap V^b, E^a \cap E^b) \; \wedge \; v_R \in V^{mg}$$

and maps into a structured, partially ordered space:

$$MG' : B \times B \rightarrow B.$$

The computation of MG is *commutative* and *associative*:

(1.) commutativity: $\forall a, b \in G(MG'(a, b) = MG'(b, a))$
(2.) associativity: $\forall a, b, c \in G(MG'(MG'(a, b), c) = MG'(a, MG'(b, c)))$.

Therefore, the common graph of a set of trees can be defined as:

$$MG(G) := \begin{cases} g & if \;\; G = \{g\} \\ MG'(MG(G'), g) & if \;\; G = G' \bigcup \{g\} \; with \; G' \neq \emptyset. \end{cases}$$

This definition is equivalent to

$$MG(G) = (V^{mg}, E^{mg}) \textit{ with } (V^{mg}, E^{mg}) \textit{ is a connected subgraph of}$$

$$(\bigcap_{g \in G} V^g, \bigcap_{g \in G} E^g) \ \wedge \ v_R \in V^{mg}.$$

Thus the similarity measure is an effective, total function:

$$\sigma : \mathcal{P}(B) \setminus \{\emptyset\} \rightarrow [0, 1]$$

with $|G| = 1 \leftrightarrow \sigma(G) = 1$. Furthermore, it holds:

(i) Idempotence: $\forall a \in G \ (MG(MG(\{a\})) = MG(\{a\}) = a$

(ii) Embedding: $\forall a, b \in G \ ((MG(\{a, b\}) \sqsubseteq a) \wedge (MG(\{a, b\}) \sqsubseteq b))$

(iii a) Monotonicity concerning set inclusion: $G \subseteq G' \ \rightarrow \ \sigma(G) \geq \sigma(G')$

(iii b) Monotonicity concerning subgraph: $\forall a, b, c \ \in \ G \wedge a \ \sqsubseteq \ b \ \rightarrow$ $MG(\{a, c\}) \sqsubseteq MG(\{b, c\})$.

Other approaches such as MACS (Bartsch-Spörl and Tammer 1994) and TOPO (Coulon 1995) define the *structural similarity* between structured case representations, i.e., graphs, via their maximal common subgraph. A comparison of CA with both approaches was presented in (Börner 1998) and is summarized in section 5.

3.4.2. *Concept representation*

Conceptual Analogy is unique in the concept representation it uses. We list the main features of the concept representation used by CA below. See (Komatsu 1992) for a review of theories of conceptual structure.

For all non-empty $K(CC) \neq \emptyset$ that represent valid case classes in $CC(CB)$ it holds that:

1. The number of cases in a case class equals the cardinality of its concept: $|CC| = |K(CC)|$.
2. The common graph of a CC equals the graph in $K(CC)$ with a relative frequency of one: $MG(CC) = m^{(CC)}_{|CC|}$.
3. The relative frequency of edges decreases with increasing path length to the root vertice:
   $\forall \, m_i^{(CC)}, m_j^{(CC)} \in K(CC) \ : \ (v_x, v_y) \in E_i^{(CC)} \ \wedge \ (v_y, v_z) \in E_j^{(CC)} \ \rightarrow$ $i > j$.
4. The union of the vertices and edges of the graphs in $K(CC)$ equals the combination graph $(V^{(CC)}, E^{(CC)})$ of CC:

$$V^{(CC)} = \bigcup_{c \in CC} V^c = \bigcup_{m^{(CC)} \in K(CC)} V^{m^{(CC)}} \wedge E^{(CC)} = \bigcup_{c \in CC} E^c = \bigcup_{m^{(CC)} \in K(CC)} E^{m^{(CC)}}.$$

5. The structural similarity of a case class CC can be defined via its concept:

$$\sigma = \frac{|E^{m^{(CC)}_{|CC|}}|}{|\bigcup_{m^{(CC)} \in K(CC)} E^{m^{(CC)}}|}.$$

These features enable the efficient selection and application of concepts as well as an efficient evaluation of solutions.

3.4.3. *Types of structural adaptations*

The adaptation done by CA resembles case combination followed by the elimination of vertices and edges that do not contribute to a problem's solution. Four types of structural adaptations can be distinguished with regard to the *number* of cases that are *partially or completely* transferred. They are listed in Table 2.

Table 2. Types of structural adaptations

Type	$MG(CC) \sqsubseteq s$	$MG(CC) \not\sqsubseteq s$
$\|CC\| = 1$	(1.) complete, single case transfer	(2.) partial, single case transfer
	$MG(\{c, p\}) \sqsubset s \ \wedge \ s = c$	$MG(\{c, p\}) \sqsubset s \sqsubset c$
$\|CC\| > 1$	(3.) complete, multiple case transfer	(4.) partial, multiple case transfer
	$MG(CC \cup \{p\}) \sqsubseteq MG(CC) \wedge$	$MG(CC \cup \{p\}) \sqsubseteq MG(CC \cup \{s\}) \wedge$
	$MG(CC) \sqsubseteq s \sqsubseteq g^{(CC)}$	$MG(CC \cup \{s\}) \sqsubseteq s \sqsubset g^{(CC)}$

In all four types of adaptation, the common graph $MG(CC \cup \{p\})$ of a case class and a problem is a part of the solution s. Furthermore, the solution is always a part of the combination graph $g^{(CC)}$ of CC. That is, $MG(CC \cup \{p\})$ and $g^{(CC)}$ determine the lower and upper boundaries of the solution space.

Each solution can be categorized into one of the four types of adaptations. The complete set of solutions might contain solutions that were derived using different types of adaptation.

3.4.4. *Set of potential solutions*

Let V_L^c and $V_L^{(CC)}$ be the sets of leaf nodes of case c and combination graph $g^{(CC)}$ of a case class, respectively. Then, the set of *potential solutions* $POT(CC)$ that can be derived from a valid case class CC by case combination equals:

$$POT(CC) = \{c \mid c \sqsubseteq g^{(CC)} \wedge V_L^c \subseteq V_L^{(CC)}\}$$

with
1.) $CC_i \cap CC_j = \emptyset \not\rightarrow POT(CC_i) \cap POT(CC_j) = \emptyset$ and
2.) $\forall s \in S_{CC,p}\, \exists c \in POT(CC)\ (s = c)$.

Thus, the number of potential solutions is many times higher than the number of concrete cases. Assume the combination graph of a valid case class CC has n leaf vertices. The number of different, potential solutions with k leaf vertices ($1 \leq k \leq n$) that can be generated from CC equals the number of combinations of n leaf vertices taken k at a time, i.e.,

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

Consequently, the number of potential solutions equals

$$|POT(CC)| = \sum_{k=1}^{n} \frac{n!}{k!(n-k)!} = 2^n - 1.$$

Assume for example, the combination graph of a case class shows 10 leaf vertices, then the number of potential solutions of this case class equals 1023. To store all potential solutions separately would require an enormous memory space (and enormous retrieval time). Again, the concept representation used by CA reduces this memory space and retrieval time substantially.

3.4.5. *Complexity results*
This section discusses the correctness and completness of *Conceptual Analogy* for solving the task of *case-based structure generation*.

Correctness: The set of concepts used by CA represents the set of valid case classes. CA's search strategy guarantees that all applicable concepts are found and are applied to generate the set of solutions. CA uses the quality measure μ to order the set of solutions. That is, CA is correct w.r.t. the task specified in section 2.

As in standard CBR, the quality of cases stored in CB influences the quality of solutions suggested.

Memory space: Let N be the number of cases in CB. The memory space required by CA to store all concrete cases by singleton concepts is N. At most $N-1$ concepts $K(CC_j)$, $j = N+1, \ldots, 2N-1$ represent more than one case, i.e., $k_j := |K(CC_j)| = |CC_j|$ and $k_j > 1$. Each of those concepts is represented by a set of graphs $m_i^{(CC_j)}, i = 1...k_j$. Each graph contains edges with an identical relative frequency of $\frac{i}{k_j}$. Case classes which do not meet the valence criterion are represented by empty concepts. In the *worst*

case, the concept hierarchy equals a degenerated list and no concept is empty. Here, the number of graphs representing $N-1$ concepts $K(CC_j)$, $j = N+1, \ldots, 2N-1$ equals

$$\sum_{i=2}^{N} i = \frac{N*(N+1)}{2} - 1.$$

Thus, the memory space is *quadratic* in N. In order to consider the number of edges of cases, we multiply the number of graphs with the average number of edges resulting in

$$\frac{\sum_{c \in CB} |E^c|}{|CB|} * \left(\frac{N*(N+1)}{2} - 1 \right).$$

However, the higher the similarity of a case class the fewer edges are used to represent the graphs of its concept and the less memory space is needed.

Preprocessing time: The time needed for knowledge organization comprises the time to *determine partition* CCP^0, the time required for *conceptual clustering*, plus the time to determine the *concept representations* (see also section 3.1).

Note that we only consider the number of basic operations required. The time needed for graph matching, combination, and evaluation depends on the concrete case graph representations which may vary considerably accross domains and tasks.

Determination of CCP^0*:*
In order to determine the partition CCP^0, N cases must be assigned to N concepts. The number of required assignments is linear to the number of cases.

Conceptual clustering:
The number of similarity comparisons required to determine partition CCP^0 equals:

$$\sum_{i=1}^{N-1} i = \frac{N*(N-1)}{2}$$

and in each subsequent partition $N-2$, $N-3$, ..., 1 (given that only one row and column are recomputed):

$$\sum_{i=1}^{N-2} i = (N-2) + (N-3) + \ldots + 1 = \frac{(N-2)*(N-1)}{2}.$$

Both values sum up to:

$$\sum_{i=1}^{N-1} i + \sum_{i=1}^{N-2} i = 2 * \sum_{i=1}^{N-2} i + (N-1) = 2 * \frac{(N-2) * (N-1)}{2} + (N-1) = (N-1)^2.$$

Subtracting the last comparison (comparing the CB to itself) results in $(N-1)^2 - 1$ similarity comparisons and thus a complexity of $O(N^2)$.
The complexity of the similarity comparisons depends mainly on the number of edges of the common graph.

Determination of the concept representation:
Given a case base with N cases, maximum $N-1$ concept representations for case classes containing more than one case must be determined. The complexity of determining a concept representation depends on the number of edges of its common graph as well as the number of edges of its cases.
Taken together, knowledge organization is *quadratic* in N.

Problem solving time: comprises the time required to *compute all applicable concepts*, to *generate solutions*, and to *evaluate solutions*. Again, we consider only the number of basic operations required.

Determine applicable concepts:
If a concrete concept $K(CC)$ with $\sigma(CC) = 1$ stored in a leaf node of the concept hierarchy is applicable, then a larger number of concepts has to be checked for applicability. However, solution generation and evaluation become easier. If only general concepts are applicable, then fewer comparisons are needed to determine these concepts, but solution generation and evaluation might require more computation effort.

Generation of solutions:
In the *worst case*, all N cases allow the generation of a solution for the problem, and all case classes that are represented by non-empty concepts are valid. Here, the applicability of at most $2N-1$ concepts has to be determined resulting in a complexity of $O(N)$.
Solution combination is constrained by the fact that only edges and corresponding vertices are transferred that lead to the complete connection of all problem vertices and edges by a tree structure. Given that all case classes and their concept representations meet the *valence criterion* there are no alternative paths to consider.

Table 3. Complexity of graph algorithms and *Conceptual Analogy*

Complexity	Graph Algorithms	Conceptual Analogy
memory space	$O(N)$	$O(N^2)$
preprocessing	—	$O(N^2)$
problem solving	$O(2^N)$	$O(N)$

Evaluation of solutions:
The number of solutions depends on the number and type of edges of a problem as well as the structure of the combination graph of the applied case classes. Sorting the set of solutions with $k = |S_{CB,p}|$ has an maximum complexity of $O(k\ log\ k)$. Thus, the problem solving time is *linear* in N.

Comparison: The number of basic operations required to solve *case-based structure generation* tasks for graph algorithms and for the approach of Conceptual Analogy is shown in Table 3. That is, CA reduces the problem solving time from *exponential time* to *linear time* in the number of cases in CB.

However, CA requires knowledge organization to proceed analogical reasoning. No *ad hoc* queries are possible. The extraction and storage of the concept hierarchy requires additional (pre)processing time and memory space.

4. SYN: An architectural design assistant

The approach of *Conceptual Analogy* has been fully implemented in SYN, a module within a highly interactive, adaptive design assistant system (Belz et al. 1995). The implemented system interacts with users via the manipulation of CAD layouts describing real buildings. Figure10 depicts a problem (left) and its solution (right) of a design task in which a number of fresh air outlets (objects of type zul-v-h-8) must be connected to the main access (object of type zul-v-h-4) via pipes (objects of type zul-v-h-6). SYN supports architects in the design of pipe systems and adapts its support to different needs based on past user interactions. See (Börner 1995a, 1997) for a more detailed description of the implementation.

The CAD-like interface of SYN represents designs by attribute values describing the position and extension of objects in three dimensions as well as their type. A representation function is needed to translate the attribute value representation of a layout (problem or case) into its graph representation. A

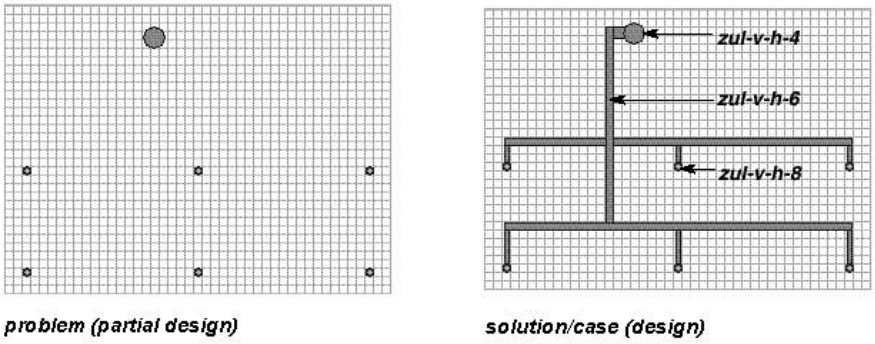

Figure 10. A design problem and its solution.

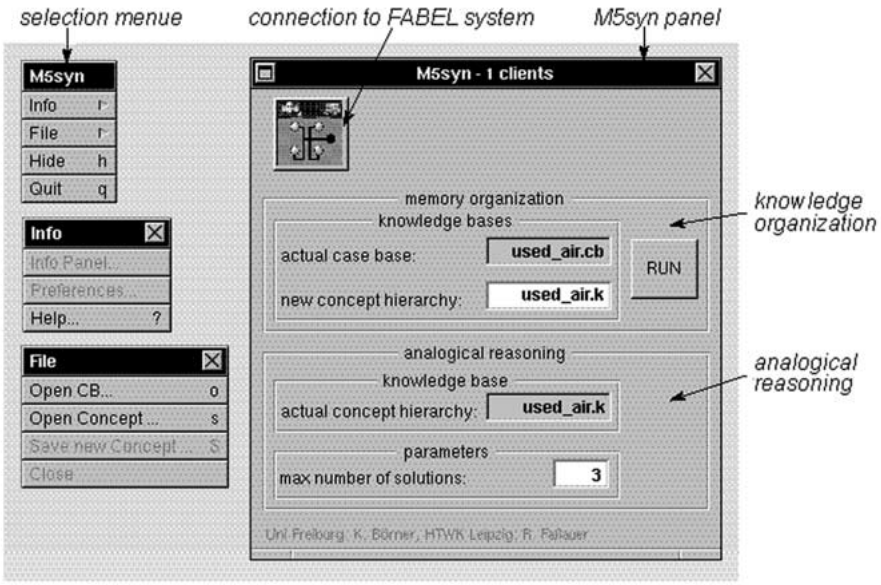

Figure 11. SYN: Selection menu and expert panel.

rerepresentation function is required to translate the graph representation of a solution into its attribute value representation. The same rerepresentation function can be used to translate the common graph of the used case class into its attribute value representation so that it can be displayed. In addition, geometric transformations such as translation or rotation must be considered. Mirroring is not a valid operation in this domain.

SYN's interface consists of a selection menu (left in Figure 11), an expert panel (Figure 11, right) as well as a naive interface (depicted in Figure 12, left). The selection menue allows the user to load a case base from the central

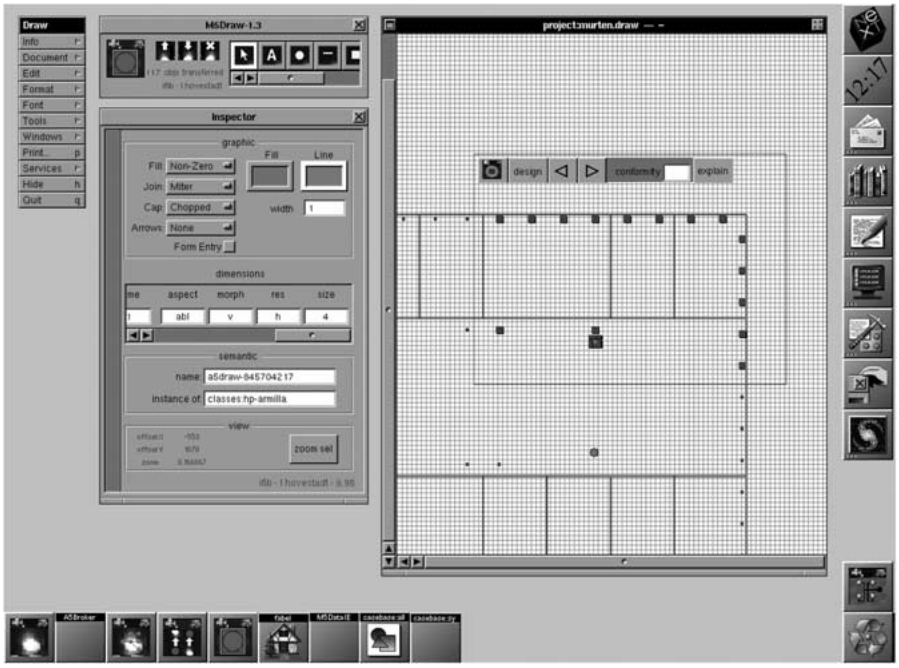

Figure 12. Graphical problem selection via the naive interface.

data base or to select an already organized concept base from the local data base. Using the expert panel, a user can connect SYN to the main design assistant system and specify a maximum number of solutions to be generated. Knowledge organization can be started by pushing the RUN button.

Figure 12 shows a typical work session. The drawing specifies the design of a building. The room layout as well as the used-air outlets of a climate system have been designed. The Naive Interface of a SYN client was copied into the drawing. The main outlet as well as the sub-outlets of an used-air region have been selected via the mouse. They specify the current problem.

The solution, its explanation via the common graph of the applied case class, as well as the solution evaluation is presented in Figure 13. Here, solution objects correspond to pipes that are placed directly in the drawing. The pipe layout is generated based on the cases and their concept representations given in Appendix A. If the user asks for an explanation, the common structure of the applied concept is provided below the solution as shown in Figure 13. The quality of the solution corresponds to a numeric value (here 0.1).

If the solution is accepted by the user, it is stored in the case base, and the concept hierarchy must be reorganized in order to incorporate the new case.

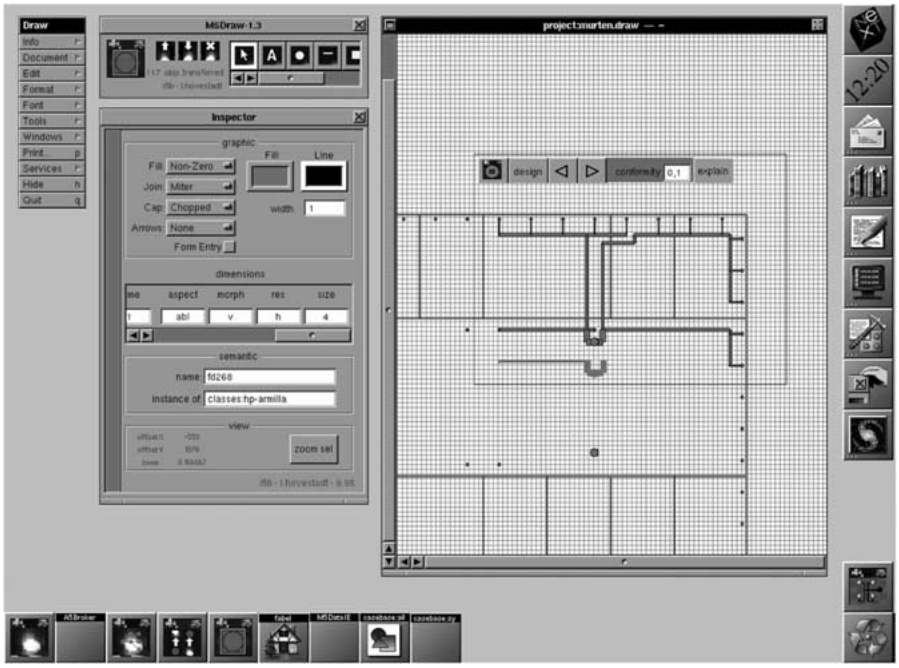

Figure 13. Graphical presentation of the design solution.

This way, the support a user receives is based on her/his past solutions. Note that a case-base filled with bad solutions will most likely result in rather bad design suggestions. Even worse, the system will assign high quality values to those solutions that closely resemble these bad designs.

While the attribute value representation of cases and its translation into graphs is strongly domain dependent, the algorithms for knowledge organization and analogical reasoning work on graph representations and are domain independent.

5. Related work

The approach of *Conceptual Analogy* and its implementation in SYN resembles two other approaches MACS (Bartsch-Spörl and Tammer 1994) and TOPO (Coulon 1995), that have been developed in the context of the German research project named FABEL (Voß 1997a).[1] All three approaches aim to solve *case-based structure generation* tasks. All use a *structural*

[1] See (Voß 1994; Börner 1995b; Voß et al. 1994) for surveys of the diverse approaches and tools to case retrieval and case adaptation that have been developed in FABEL.

similarity measure that defines the similarity of graph-based case representations via their maximal common subgraph (*mcs*) and transfer case parts to solve a new problem. However, there are far reaching differences between the specific case representations, the organization of the case base, and the way the *mcs* is used in retrieval and adaptation. A detailed description of the approaches and their comparison was presented in (Börner 1998a). We present a brief summary here.

TOPO represents cases by labeled graphs. MACS represents cases by arbitrary graphs. In order to compare graph representations, MACS and TOPO apply graph matching algorithms (clique search and backtracking) that are known to be NP-complete.

To reduce matching complexity, TOPO applies the *Fish & Shrink* (Schaaf 1995) retrieval algorithm. The retrieval algorithm works over the attribute values of case representations and selects one source case. This way, the number of computationally expensive structural comparisons is reduced to finding the match between one selected source case and the problem. TOPO concentrates on case adaptation and requires a separate retrieval system or an user to provide a case that can be adapted to solve the current problem.

Similar to CA/SYN, MACS organizes the case-base dynamically during a memory organization process. Instead of organizing cases in a concept hierarchy, MACS uses a two-level case organization. The lower partition contains the concrete cases grouped into classes of similar cases. The upper partition contains graphs describing the *mcs* of each case class. During retrieval, MACS performs a two stage retrieval selecting the case class with the most similar *mcs* first and searching in its cases for the most similar concrete case(s). This reduces the number of NP complete graph matches required to search through N cases to $2\sqrt{N}$ in the best case.

CA/SYN's restriction to represent cases by trees reduces expressibility, but offers the advantage of efficient matching. In addition, the tree representation of cases guarantees unique mcs. This does not hold for the graph representations used in TOPO or MACS. Here, domain specific selection rules need to be defined or user interaction is necessary to select the most suitable *mcs*. This can be advantageous during retrieval if different points of view on two graphs – representing a *mcs* and a problem – need to be implemented. It may not be acceptable to organize huge case bases efficiently.

During adaptation, TOPO investigates the compatibility of object types and relation types of layouts. The frequency of relations in past layouts is exploited to come up with preferable positions for solution objects. MACS realizes a simple variant of case adaptation by adding all walks of the case which do not have an isomorphic mapping in the problem but begin and end with vertices of their mcs. CA/SYN is unique in that it represents cases by a

hierarchical concept hierarchy and by its definition of *applicability* allowing the efficient selection of the most similar concept that is neither too general not too concrete, guarantees the generation of a problem solution, and the evaluation of solution quality.

Online CBR resources such as *www.ai-cbr.org* or *www.cbr-web.org* list a growing number of elaborated case-based design systems, that have been proposed for a variety of domains such as building design, electro-mechanical design, design of pharmaceuticals, or VLSI design. However, to our knowledge, none of these approaches aims at the support of *case-based structure generation* tasks. Therefore, a comparison of other approaches with CA/SYN would have to be made at a fairly general level and would resemble the discussions in Section 2.

6. Conclusions

The paper started with the formal definition of a special kind of design task, named *case-based structure generation*. The applicability of existing CBR approaches and graph-based approaches to solve this task was discussed in subsections 2.2 and 2.3. Based on this, the approach of *Conceptual Analogy* was introduced, exemplified, and discussed. Section 4 described the implementation of CA in SYN and its application to provide efficient support in the domain of architectural design. Section 5 related Conceptual Analogy and its implementation to other systems that aim at the support of *case-based structure generation* tasks.

Taken together, the approach of CA can be applied advantageously to support design tasks that match the task of *case-based structure generation* formalized in section 2.1. The selection of an appropriate graph representation for cases is domain and task specific. However, the definitions of *case classes, concept hierarchy, structural similarity, adaptability,* and *quality* are generic.

Coming back to the question stated in the introduction – the answer is yes. *Conceptual Analogy* provides a way to solve *case-based structure generation* tasks based on cases and a quality measure. Knowledge about the *similarity* and *applicability* measure was derived from these two sources. Neither of the two measures encode *domain knowledge*. Instead they represent *control knowledge* and are used to guide the search for applicable concepts and the generation of solutions. Solution generation can be seen as a special kind of *optimization problem*, i.e., finding an optimal solution to a given problem, case base, and quality measure (Bergmann and Wilke 1998).

7. Outlook

Artificial intelligence approaches like CA offer efficient ways to support certain real-world tasks such as architectural design. However, a major problem concerning assistance systems is the limited interaction capability of the currently used human-computer interfaces. Often, human-computer interaction is restricted to non-intuitive use of keyboard, mouse, and screen. Extensive training is required to handle professional programs such as a CAD tool effectively.

Virtual Reality techniques and fast computer graphics offer new ways to create efficient and intuitive human-computer interaction. Multimodal interaction directly in three-dimensional space, using two hands and audio feedback, enables the user to formulate design ideas in a much more intuitive way. *Sculptor* (Kurmann 1995) or *Sketch* (Zeleznik et al. 1996) are only two out of a growing number of available systems that support modeling in 3D. Additionally, multimodal VR interfaces permit extensive tracing of human responses that go far beyond recording mouse events, see also (Watson and Oliveira 1998; Börner and Vorwerg 1998). The resulting behavioral protocols can serve as a basis to support human problem solving in complex tasks that involve spatially organized information. VR interfaces combined with Artificial Intelligence techniques like CA may lead to adaptive human-computer interaction (Börner 1998b) that feels more intuitive and may provide easy-to-use support for novices and greater effectiveness for experts.

VegoWelt is a Smart Virtual Environment that uses a children's playroom scenario for demonstrating and evaluating the support of manipulation activity (Börner 1999a, 1999b). In *VegoWelt*, human-computer interaction proceeds via direct manipulation of virtual, three-dimensional building blocks. The approach of *Conceptual Analogy* is applied to derive concept hierarchies out of user generated designs as well as to support the design of complex assemblies by applying existing concepts. This way, knowledge structures (concepts) and the design support based on them are adapted. In parallel, human-computer interaction gracefully progresses from novice's support on a very concrete level to sophisticated support on a more general level.

Acknowledgements

The research on Conceptual Analogy was based on work in the project *FABEL: Integration of Cases, Rules, and Models for Design Support* and was supported by the German Ministry for Research and Technology (BMBF) under contract no. 413-4001-01IW104. It was strongly inspired by work of

Bipin Indurkhya and *Douglas Hofstadter* to model human analogy-making in idealized domains. I wish to thank Klaus P. Jantke, Gerhard Strube, Michael M. Richter, and Hans-Dieter Burkhard for their assistance and advice. Research on *Smart Virtual Environments* such as VegoWelt was fundet by a Postdoc fellowship of the German Academic Exchange Center (DAAD). I would like to thank Bipin Indurkhya, Michael Gasser, Michael M. Richter, and the anonymous reviewers for many insightful comments on an earlier version of this paper.

Appendix A. Case base and derived concepts

This appendix contains a sample case base and concepts representing pipe layouts in real buildings. These cases have been used by the design assistant SYN to generate the solution depicted in Figure 13 for the problem selected in Figure 12.

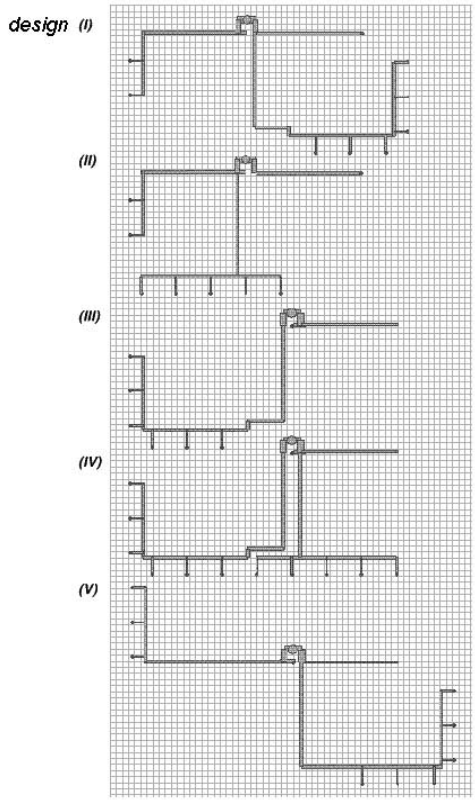

Figure 14. Used-air pipe layouts of different buildings.

Figure 14 shows a case base of five designs presenting used-air pipe layouts. The layouts have been positioned on the ARMILLA* grid and have been rotated such that the main access is in the top left corner.

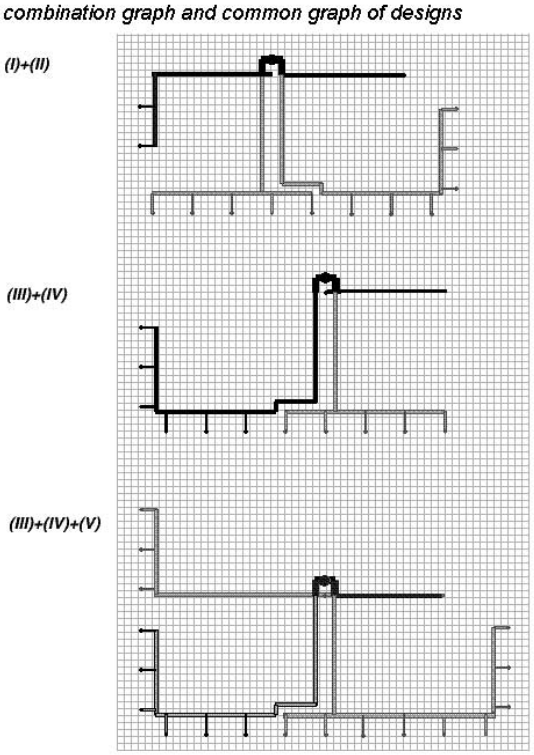

Figure 15. Combination graph and common graph of pipe layouts I–V.

During knowledge organization the five cases were organized hierarchically into a case class hierarchy. First, case (II) and (IV), then case (I) and (II), subsequently case (III), (IV), and (V) and finally all five cases were grouped. The common graph representing all five cases contains inner vertices with a valence larger than one. It can not be used to generate new solutions and $K(CB)$ is represented by an empty set. Thus, the concept hierarchy consists of five concepts each representing a single case as well as three other non-empty concepts. The common graphs and the combination graphs of the latter are depicted in Figure 15.

The labels of grouped cases are given on the left hand side. Their common graph is drawn in black. Their combination graph is given in grey and black. To generate the solution shown in Figure 13 three cases labeled (III), (IV) and (V) were combined.

Figure 16 illustrates the modifications that are required to generate a solution. Additional vertices and edges (marked by circles) had to be eliminated. The entire layout had to be turned

* ARMILLA is a pipe layout system developed by (Haller 1985). It provides different predefined grids for the layout of various pipe systems among others.

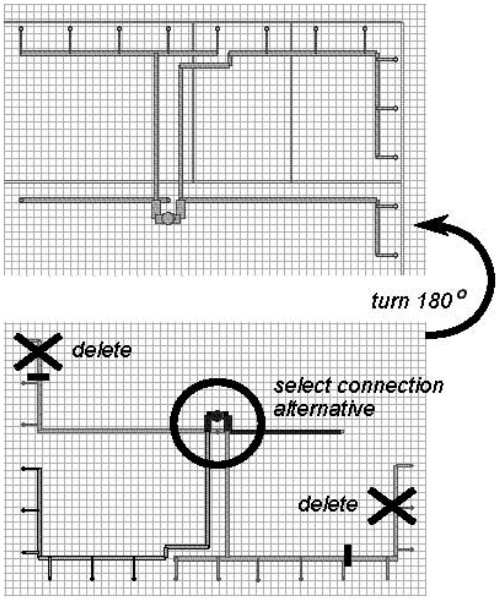

Figure 16. Solution generation by design combination.

180 degree. There exist two alternative connections to the main access and thus two solutions of different quality. The qualitatively best solution is depicted in Figure 16, top.

References

Aamodt, A. & Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AICOM* **7**: 39–59.

Bartsch-Spörl, B. & Tammer, E.-C. (1994). Graph-Based Approach to Structural Similarity. In A. Voß (ed.), *Similarity Concepts and Retrieval Methods*. GMD, Sankt Augustin, 45–58.

Belz, B., Gräther, W., Groß, E., Walther, J., Oertel, W. & Hovestadt, L. (1995). FABELIDEA 2, Intelligente Designunterstützung für Architekten, Version 2: Fallretrieval. FABEL-Report 30, GMD.

Bergmann, R. & Wilke, W. (1998). Towards a New Formal Model of Transformational Adaptation in Case-Based Reasoning. In H. Prade (ed.), *Proceedings of the 13th European Congerence on Artificial Intelligence (ECAI-98)*. John Wiley and Sons.

Börner, K. (1993). Structural Similarity in Case-Based Design. In D. Janetzko and T. J. Schult (eds.), *2nd German Workshop on CBR (Fälle in Hybriden Systemen)*, 1–13.

Börner, K. (1995a). Interactive, Adaptive, Computer Aided Design. In M. Tan & R. Teh (eds.), *The Global Design Studio. Proceedings of the 6th International Conference CAAD Futures*. Singapore: CASA, School of Architecture, 627–734.

Börner, K. (1995b). Modules for Design Support. Fabel-report 35, GMD, Sankt Augustin.

Börner, K. (1997). *Konzeptbildende Analogie: Integration von Conceptual Clustering und Analogem Schließen zur effizienten Unterstützung von Entwurfsaufgaben. Dissertation*. DISKI 177, Universität Kaiserslautern.

Börner, K. (1998a). CBR for Design. In B. Bartsch-Spörl, S. Wess, H.-D. Burkhard & M. Lenz (eds.), *Case-Based Reasoning Technology: From Foundations to Application*. LNAI, chap. 8. Springer Verlag.

Börner, K. (1998b). Concept-Based, Adaptive Human-Computer Interaction. In J. Dunnion, G. O'Hare, S. Nuallain, R. Reilly & B. Smith (eds.), *Proceedings of the 9th Irish Conference on Artificial Intelligence and Cognitive Science*, 103–109.

Börner, K. (1999a). VegoWelt: A Smart Virtual Playroom. *International Journal of Design Computing, 2*. WWW:http://www.arch.usyd.EDU.AU/kcdc/journal/vol2/dcnet/sub6/.

Börner, K. (1999b). Welcome to VegoWelt. *YLEM Newletter, July/August*. WWW:http:// www.ylem.org.

Börner, K. & Vorwerg, C. (1998). Applying VR Technology to the Study of Spatial Perception and Cognition. In *Proceedings of Mind III: The Annual Conference of the Cognitive Science Society of Ireland, Theme: Spatial Cognition*, Vol. 1/3, 128–134.

Coulon, C.-H. (1995). General Geometric and Topologic Retrieval and Adaptation (TOPO). In Börner, K. (ed.), *Modules for Desing Support*. Gesellschaft für Mathematik und Datenverarbeitung mbH (GMD). FABEL-Report 35.

Falkenhainer, B., Forbus, K. D. & Gentner, D. (1986). The Structure-Mapping Engine. In *Proceedings of the American Association for Artificial Intelligence*. Philadelphia, 272–277.

Haller, F. (1985). *ARMILLA ein Installationsmodell: Instrumentarium zur Planung von Leitungssystemen in Hochinstallierten Gebäuden*. Germany: IFIB, Univerisität Karlsruhe.

Hamming, R. W. (1980). *Coding and Information Theory*. Englewood Clifs, NJ: Prentic Hall.

Jantke, K. P. (1994). Nonstandard Concepts of Similarity in Case-Based Reasoning. In H.H. Bock, W. Lenski & M. M. Richter (eds.), *Information Systems and Data Analysis: Prospects–Foundations–Applications, Proceedings of the 17th Annual Conference of the GfKl, Universität Kaiserslautern, 1993*. Springer Verlag, 29–44.

Kolodner, J. L. (1992). An Introduction to Case-Based Reasoning. *Artificial Intelligence Review* **6**: 3–34.

Komatsu, L. K. (1992). Recent Views of Conceptual Structure. *Psychological Bulletin* **112**(3): 500–526.

Kurmann, D. (1995). Sculptor – A Tool for Intuitive Architectural Design. In M. Tan & R. Teh (eds.), *Proceedings of the 6th International Conference on CAAD Futures'95*. Singapore, 323–330. WWW:http://caad.arch.ethz.ch/~kurmann/sculptor/.

Maher, M. L. & Pu, P. (1997). *Issues and Applications of Case Based Reasoning in Design*. Lawrence Erlbaum.

Oxman, R. & Voß, A. (1996). CBR in design. *AI Communications* **9**(3): 128–137.

Richter, M. M. (1995). The Knowledge Contained in Similarity Measures. Invited Talk at ICCBR'95, Universität Kaiserslautern. WWW:http://wwwagr.informatik.uni-kl.de/lsa/ CBR/Richtericcbr95remarks.html.

Schaaf, J. W. (1995). Fish and sink: An Anytime-Algorithm to Retrieve Adequate Cases. In M. M. Veloso & A. Aamodt (eds.), *Case-Based Reasoning Research and Development: First International Congerence on Case-Based Reasoning (ICCBR-95)*. Springer, 538–547.

Voß, A. (1994). Similarity Concept and Retrieval Methods. Fabel-report 13, GMD, Sankt Augustin.

Voß, A. (1997a). Case Design Specialists in FABEL. In M. L. Maher & P. Pu (eds.), *Issues and Applications of Case Based Reasoning to Design*. Mahwah, NJ: Lawrence Erlbaum Associates.

Voß, A. (1997b). Case Reusing Systems – Survey, Framrwork and Guidelines. *The Knowledge Engineering Review* **12**(1).

Voß, A., Bartsch-Spörl, B., Börner, K., Coulon, C. H., Dürschke, H., Gräther, W., Knauff, M., Linowski, B., Schaaf, J. & Tammer, E. C. (1994). *Retrieval of Similar layouts-about a very Hybrid Approach in FABEL*. In J. Gero & F. Sudweeks (eds.) *AI in Design'94*. Dordrecht: Kluwer Academic Publishers, 625–540.

Watson, I. & Oliveira, L. (1998). Virtual Reality as an Environment for CBR. In *Advances in Case-Based Reasoning*, Vol. 1488 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 448–459.

Zeleznik, R. C., Herndon, K. P. & Hughes, J. F. (1996). Sketch: An Interface for Sketching 3D Scenes. In *Proceedings of the 23rd Annual Conference on Computer Graphics*, 163–170.