Interactive, Adaptive, Computer-aided Design

Katy Boerner

University of Freiburg, Centre for Cognitive Science
Institute of Computer Science and Social Research
79098 Freiburg
GERMANY

A general framework of a system that supports building engineering is presented It accounts for a set of desirable features. Among them are (1) graphical man-machine interaction, (2) high interactivity to facilitate the acquisition of the huge amount of knowledge necessary to support design, (3) incremental knowledge acquisition as the basis for incrementally increasing system support, and(4) adaptability to the tasks which are tackled, the distinctive features of the domain, and user preferences. This paper provides the underlying assumptions and basic approaches of the modules constituting this framework and sketches the current implementation.

Keywords: graphical interfaces, incremental knowledge acquisition, knowledge organization, case-based reasoning, design prototypes, logical reasoning

I Introduction

Building engineering is one of the keystones to economic competitiveness. As a consequence, computational models which support design in this domain are important research topics. To meet the expectation that computer-aid enables the production of better designs in a shorter time, the tools built on these models need to be fully integrated into the workflow of architects. This paper presents the framework of a support system for building engineering which is highly interactive. Graphical man-machine interaction provides the basis for a reaction in close co-operation with the user. The knowledge base of the system and its support is adaptable to the tasks to be tackled, the distinctive features of the domain, and the preferences of users. The huge amount of knowledge necessary to support design is acquired automatically, i.e., without bordering the user to answer thousands of questions. Machine learning techniques are applied to provide the knowledge necessary to support subgoaling, browsing, and design.

Genuine models are hard to construct for building engineering because of the complexity, uncertainty, and vagueness prevailing in design decisions. Human designers browse through prior layouts to inspire and guide their work. According to our experience and to current literature as well [1,2,3,4],case-based reasoning (CBR) seems to be an appropriate problem solving method. CBR as the main poblem solving method forces the acquisition of cases and appropriate similarity measures over case-sets. Questions arising from this are: What ‘grainsize’ should cases have? How should cases be represented? How should 'similarity' in a computational effective but at the same time structural selective manner be defined? Finally, how should the huge amount of knowledge necessary to support building design be acquired and organized? Most work in case-based design (CBD) aims to support creative design. It focuses on index structures [1,4] to label and retrieve prior layouts via hypermedia browsing systems, e.g. ARCHIE 11[5], or on adaptation through dimensionality reduction to provide an interactive architectural design assistant, e.g., CADRE [3], which provides case adaptation but leaves case selection to the user. No satisfying method is available which automatically retrieves and adapts prior layouts. This may be due to the fact that retrieval for adaptation requires the definition of similarity in terms of the adaptation knowledge available. Usually, it is impossible to determine similarity without processing the computationally expensive adaptations.

Adaptation-guided retrieval, as implemented in DejaVu [6], judges the similarity of each case in terms of the handcoded, domain (in)dependent adaptation procedures that can perform the needed adaptation of case elements. In geometrical layout design, adaptation mainly corresponds to adding, eliminating, or substituting objects and their relations. Because of the variety and the possible combinations of these modifications, adaptation knowledge is hard to acquire by hand. To extract this knowledge from prior experiences automatically, a different representation of adaptation knowledge and a different definition of similarity is necessary.

Complex case representations are needed because we do not only have to consider the geometrical attribute values of single objects, but, most of all, their topological relations. On the other hand, these representations increase the computational expense in retrieving, matching, and adapting cases. Short response times are crucial for the acceptance and usage of CBD systems. Graphical user interaction is another desirable feature of a design support system [7]. However, graphical interfaces restrict system input and output to CAD layouts. To our knowledge, there is no method which automatically extracts the knowledge needed for CBD (i.e., complex case representations, the relevance of object features and relations, and proper adaptations) from attribute-value representations of prior layouts.

The sections of this paper may be summarized as follows. Section 2 presents the outline of the framework. Section 3 provides an introduction to the application domain including its formalization.

Section 4 sketches the modules for knowledge acquisition. Section 5 introduces the modules providing design support. The final part of the paper discusses the framework presented here and gives some pointers to related work.

2 Overall framework

Figure 1 sketches the system architecture for an interactive, adaptive, computeraided design support system including man-machine interaction. While grey boxes denote modules that perform knowledge acquisition, the white boxes indicate the modules that support the design process. Arrows are labeled by input and output data.

As a GRAPHICAL INTERFACE we use the hypermedia drawing environment DANCER developed at the Institut fuer Industrielle Bauproduktion (IFIB), University of Karlsruhe [8]. While employing the system like a standard CAD drawing-tool the user creates new objects, resizes them, assigns labels to them, etc. System support is presented graphically as well. The user is always in a position to accept, modify, or reject suggestions. If the actual problem can't be solved by the system it is allocated to the user. User actions, i.e., the creation, deletion, and labeling of objects as well as the acceptance, modification, or rejection of system proposals are recorded in a trace. The trace feeds into two different modules that acquire the knowledge needed for design support incrementally.

Forms, forcing the user to answer a set of predefined questions, may be used to acquire additional knowledge.

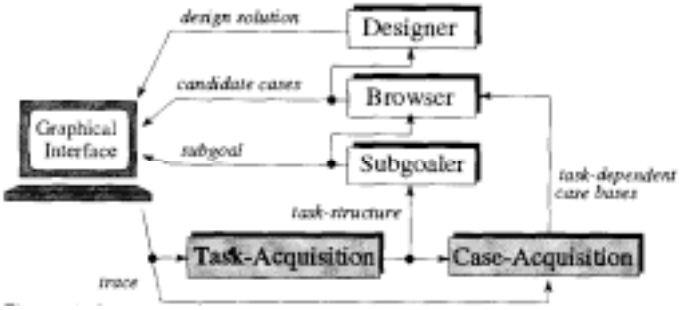

Figure 1: System architecture and man-machine interaction

The module named TASK ACQUISITION, takes the trace and automatically extracts predecessor relations between object types (e.g., room-outlets have to be designed before the furniture may be arranged etc., accesses have to be designed before they are connected etc.) from them. The object types and their relations are represented as a semantic

network, named task structure. Aiming at a 'task-oriented user support' [9] the grain size of cases corresponds to the grain size of the elementary decisions of the architect. Thus, the CASE ACQUISITION module uses the task structure and geometrical data (e.g., room size, area size) to extract cases from the traces. For efficiency reasons, we store cases which support one task (e.g., the design of interconnections for fresh air) in a task-dependent case base.

The system support may be divided into

(i) guidance in selecting the next subgoal, i.e., the next task to be tackled in design;

(ii) suggesting a set of candidate cases (CAD-layouts) able to give some hint about how to solve the actual problem; and

(iii) generating an adapted design solution which fits the selected subgoal.

These support modes are provided by the SUBGOALER, BROWSER, and DESIGNER module respectively. The modules and their underlying approaches are discussed in detail in section 3 and 4.

2 Application domain and its formalization

The application domain used to delineate our rationale of incremental, adaptive knowledge acquisition and organization as well as interactive design support is industrial building engineering. We focus on installations in buildings with a complex infrastructure. Here, the main problem is how to layout subsystems for fresh and used air, electrical circuits, warm, cold, and used water, computer networks, phone cables, etc. Using the construction kit MID! suitable for installations according to ARMILLA [10], layouts are formed exclusively from a huge variety of catalogue objects, which are arranged on a predefined grid.

In the following we will introduce the notions of projects and task structures. As for memory organization in terms of knowledge application during design support, we need to define cases, case bases, and case classes. Furthermore, the definition of two kinds of similarity measures is necessary. The formalizations are needed to introduce the modules for knowledge acquisition and reasoning in section 3 and section 4.

2.1 Projects

To represent CAD-like drawings, we use the representation scheme A4 [8]. A4 allows the graphical and attribute-based representation of objects (e .g., concrete objects like rooms, pipes, chairs etc., but also more abstract objects like areas for the entire house or the climate system, etc.). The former representation is used as the main basis for manmachine interaction. The latter representation constitutes the basis for semi-automatical knowledge acquisition and support.

Graphically, A4-objects are represented by geometric objects. Different states during the design of a building correspond to different configurations of objects. Ellipses are a substitute for rectangles circumscribed by the ellipses. Using ellipses instead of rectangles is an unaccustomed but very useful trick: Ellipses overlap only in a few points. The readability of drawings becomes essentially improved. For reasons of readability, we have restricted ourselves to two dimensions here.

Attribute-based, A4-objects are represented by values for a fixed set of geometric attributes and type attributes. Geometric attributes that describe placement and extension are: {placement-in-x, extension-in-x, placement-in- y, extension-in-y}. The type of an object is denoted by the values of the attributes {aspect, morphology, resolution, size}. Aspect relates to the subsystem of the building (e.g., z=supply-air). Morphology denotes the general function of the area addressed (e.g., a=linkage, e=development, v=connection). The third attribute specifies the kind of resolution employed (e.g., b=area, h=bounding-box). Size relates to the part of the building that is envisaged (e.g., 4=hallway, 6=room, 8=areas within a room).

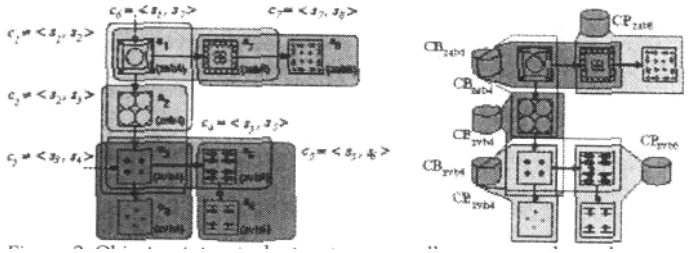

Figure 2: Objects, states, task-structure as well as cases and case bases

2.2 Task structure

CAD-like drawings correspond to snapshots of different stages of problem solving. The design of objects of a unique type may be seen as an elementary design step or task and therefore grouped to constitute states.

Figure 2 (left) depicts the design of eight different states named s_1 .s_8 of a supply air system. Each state is represented by a set of objects. The geometrical attributes of objects are represented graphically. Their types are represented by attribute values. As an example, state s_1 of type zab4 shows five objects that represent the areas of supply-airlinkage which cover the hallway.

The task to be supported by case-based reasoning is the creation of new objects based on previously designed objects. Given a certain design state, say s_1 in Figure 2 (left), the design of two states s_2 or s_7 may be tackled. Designing s_2 first, s_3 on the basis of s_2 or s_7 on the basis of s_1 may be tackled next etc. In Figure 2 (left), arrows are used to denote these predecessor relations between states. The set of states together with their relations constitute the task structure. The weighting on state transitions may be used to state their frequency, i.e., preference.

2.3 Cases, case bases, and case classes

As for reasoning, design fragments have to be connected to each other. Aiming at a task-oriented user support [9], the grainsize of cases is identical to the grainsize of tasks. More specifically, cases connect preconditions of tasks to the completed task itself. The objects which are necessary, i.e., the preconditio to work a task out, is called the case problem. The objects which consitute the solution of the tasks are represented by the case solution.

Different design steps, i.e., tasks, require different design strategies. For example, return air accesses are connected by using the shortest admissible path but connections for supply air accesses take curved tracks to achieve reduction of the noise caused by the air. The support of each task requires access to task-dependent case bases that provide cases which share identical solution types. For an illustration see Figure 2 (right). Given the problem objects, of type 'zab4', case c-l= {s-1, s-2} supports the design of 'zeb4'-objects. ext, these zeb4- objects of s_2 become the problem and are solved by designing objects of type zvb4, i.e., s_3, applying c_2={s_2, s_3}. Depending on the stage in problem solving a state can either take the role of a problem or of a solution. Note that the application of the task structure guarantees a homogeneous distribution of cases for the different tasks occurring in design.

Attribute-based cases are defined by the geometrical and type attribute values of immediately subsequent states. There is an important difference between the geometrical and the type attribute values describing the solution. While the type-attribute values represent the next subgoal, the geometrical attribute values represent the specific design solution. Thus, the type attribute values refer to the peculiarities of states with different case bases and similarity relations. In contrast, the geometrical attributes represent the geometry of a set of objects.

In general, there is no information available about solution paths or adaptation operators. Due to the application domain, attribute-value representations are not sufficient to determine the similarity of cases in terms of adaptability. For illustration, consider Figure 2, state s_5. Given the geometrical attribute values of the four partial object arrangements, their identity after rotation and reflection is difficult to determine. Reasoning

has to proceed through object relations instead of through attribute values of single objects. To re-represent geometrical layouts in terms of their topology, we use an algebraic case representation which was inspired by [11]. It enables the representation of the topology of cases by ground terms (i.e., terms containing no variable like 'above(a,b)) of some term algebra. See [12,13] for a detailed explanation. We assume the existence of a function that transforms the attribute-based representations into structural representations and vice versa.

Structural case representations are the basis for dividing task-dependent case bases into case classes. Here, case classes are defined as sets of cases sharing similar topology. Each case class may be represented by their prototypical topology (their most most specific generalization [14]) represented by a term as well as the modification rules (inverse substitutions) that were applied to derive this prototype from the concrete cases. A prototype refers either to the best instance (see prototype theory [15]) or to a more general representation of a case class. The former holds if background knowledge (e.g., geometrical transformations) is available which reduces every case of a class to one unique instance of the case class. The latter is applied, if generalization is used to derive the prototype of a case class. Combinations of generalization and geometrical transformations are possible. As proposed by [16], the prototype represents common features with constants or functions. It acts as a holding form for typical object topologies. Distinctive features are represented by variables. Modifications, named c_rules and a_rules, define the replacement of subterms by variables and variable instantiations respectively.

2.4 Similarity measures

As for the retrieval of prior cases out of huge data sets, the definition of a similarity measure is necessary.

To determine a set of candidate cases supporting the same task, an attribute-based similarity measure will do. The moment we want to retrieve cases for design support, we need to define a similarity measure that takes structural features of cases, i.e., their topology, as well as the adaptation knowledge available into account. Thus, we need to define two different similarity measures.

Attribute-based similarity of two problem states may be done in a standard way. See [17] for different approaches. For instance, the Hamming distance of attribute values may be determined and transformed into a similarity measure. Weights may be used to denote the relevance of attribute values.

Structural similarity assessment proceeds by comparing the actual problem with the prototypes representing the case classes of the a task-dependent case base. Therefore, the prior modifications (c_rules) are applied to fit the problem to the prototype (see also

3 Modules for knowledge acquisition

There are two modules that acquire the knowledge needed for design support. Both are characterized as follows.

3.1 Task acquisition

The module for TASK-ACQUISITION extracts preferred state sequences, i.e., the task structure, from several user traces. More exactly, it records predecessor relations between object. Thus, based on state sequences for specific buildings, we derive complex task structures which correspond to the alternatives and peculiarities when designing different buildings. Weights that denote the frequency of state transitions enable the handling of multiple choices. TASK-ACQUISITION may be provided by the module named ACTION. The documentation of its implementation and evaluation may be found in [18].

3.2 Case acquisition

Knowledge acquisition for design is performed by a module named CASEACQUISITION. This module uses the task structure like a cookie-cutter to stencil cases from user traces. Cases are given in A4 representation, i.e., by attribute-value pairs. For efficiency reasons, cases that support unique tasks are stored in task-dependent case bases. The CASE-ACQUISITION module was implemented separately in order to cut cases of unique grainsize automatically and in a task-oriented way.

4 Modules for design support

As for design support the framework proposed provides three different modules. These are called SUBGOALER, BROWSER, and DESIGNER.

4.1 Subgoaler

Subgoaling refers to the capability of knowing what to do subsequently at any stage of problem solving. Given an actual state of the design, subgoals have to be created, selected, and linked. Solution paths that exclude or interact with each other are possible. Subgoaling support is provided by the SUBGOALER that uses the task structure to propose the next subgoal(s) to be tackled. The SUBGOALER may even call the BROWSER or DESIGNER for the design solution to the actual problem.

4.2 Browser

Retrieval of candidate cases is possible by activating the BROWSER. Inputs are the geometric problem description and the next subgoal to be addressed. If the user wants design support, a set of candidate cases CC is selected and presented. The appropriate threshold value for the required similarity is strongly user, domain, and task dependent. Browsing capabilities provide modules like FA V, ASM, or PIX [17] that compare (weighted) attribute values using a similarity measure working over attribute-value representations of cases.

Browsing may be used as a fast preselection step that restricts the number of cases needed to be investigated in a computationally expensive structural similarity assessment.

4.3 Designer

Adapted design solutions are generated by the DESIGNER module. If the user asks for design-support structural similarity assessment and adaptation proceeds. For structural similarity assessment and adaptation, previous and actual geometric attributebased descriptions are transformed into algebraic ones. Note that the new problem does not allow for automatic re-representation. It needs to be reformulated in terms of prior Fototypes. The problem may be reformulated several times, applying different prototypes. tris then categorized in the case class with the most similar prototype. The prototypical solution is transferred and the a_rules are applied to derive the adapted solution. Following this, the actual solution is transformed into its attribute-value representation to enable its graphical representation. Finally, the actual case is added to the appropriate case base. The underlying approach was named conceptual analogy (CA) [12,13,19]. It has been implemented in a module called SYN (for SYNthesis) [20].

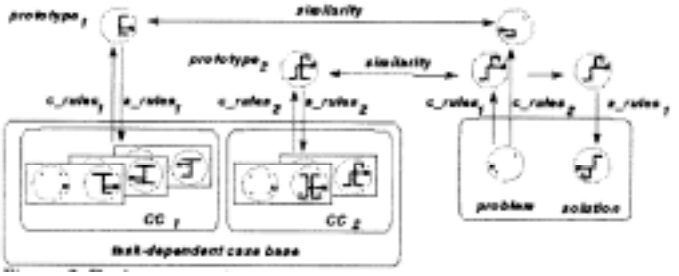

Figure 3: Design support

Figure 3 sketches memory organization and design support illustrated with cases taken from layout design. Representing outlets by small squares and the main access by a square of a larger size the case problems depict simple access patterns. The connection of these accesses is represented by the corresponding case solutions. The entire taskdependent case base is divided into two case classes CC_1 and CC_2 that conform to the common structural features of the cases. Each case class is represented by its prototype and corresponding modifications. Given a new problem as depicted on the right, it is reformulated in terms of these prototypes. The problem will be categorized into CC_2,

because of its structural similarity to prototype _2. The prototypical solution is adapted by applying the corresponding set of a_rules_2.

5 Discussion

Providing interactive, adaptive design support, the problems that relate to the amount and the structural complexity of the knowledge have to be addressed. Systems that require an immense effort for knowledge elicitation will hardly succeed. There is simply no time to feed in all knowledge required. Filling in large forms to label and save each possibly useful experience, as suggested e.g., for the ARCHIE system [5] is also difficult to integrate into the usual workflow of architects. We believe that highly user-interactive frameworks, which manage knowledge elicitation during the system usage, are a real challenge to enable useful and realizable computer-aided support in design. The close linkage of knowledge acquisition, organization, and problem solving guarantees support that is adaptable to the user and the peculiarities of the chosen domain.

6 Acknowledgements

This research has been strongly inspired by work done in the project FABEL, the general objective of which is the integration of case-based and model-based approaches in knowledge-based systems. Many thanks to Roland Fassauer and Hardy Keppler which have been involved in the implementation and the still ongoing evaluation of the framework. My thanks also go to Heiko Wode and Dietmar Janetzko for several discussions on this topic that helped shaped this research. Nick Ketley deserve thanks for critical comments on a draft of this paper. Nonetheless, the paper reflects our personal view.

This research was supported by the Federal Ministry of Education, Science, Research and Technology (BMBF) within the joint project FABEL under contract no. 4134001-01IW104. Project partners in FABEL are German National Research Center for Information Technology (GMD), Sankt Augustin, BSR Consulting GmbH, Muenchen, Technical University of Dresden, HTWK Leipzig, University of Freiburg, and University of Karlsruhe.

7 Bibliography:

[1] Goel, A. K.: Integration of case-based reasoning and model-based reasoning for adaptive design problem solving, PhD thesis, Ohio State University, 1989.

[2] Hinrichs, T. R.: Problem solving in open worlds: A case study in design, Lawrence Erlbaum Associates, Hillsdale, NJ, 1992.

[3] Hua, K. & Faltings, B.: Exploring case-based building design - CADRE, AT EDAM, 7(2), pp. 135-144,1993.

[4] Kolodner, J. L.: Case-based reasoning, Morgan Kaufmann Publishers, San Mateo, CA,1993.

[5] Domeshek, E. A. and Kolodner, J. L. A case-based design aid for architecture. Proc. Second International Conference on Artificial Intelligence in Design, Kluwer Academic Publishers, Norwell, MA, pp. 497-516,1992.

[6] Smyth, B. & Keane, M.T. Retrieving adaptable cases: The role of adaptation knowledge in case retrieval, Topics in Case-Based Reasoning - Selected Papers from the First European Workshop on Case-Based Reasoning (EWCBR-93), Wess S., Althoff K.-D. & Richter M.M. (eds.), LNAI, volume 837, Springer Verlag, pp. 209-220, 1994.

[7] Pearce, M., Goel, A. K., Kolodner, J. L., Zimring, C., Sentosa, L. & Billington, R. Case-based design support: A case study in architectural design, IEEE Expert, pp. 14-20, October 1992.

[8] Hovestadt, L., A4 - digital building - extensive computer support for the design, construction, and management of buildings, CAAD Futures 93, U. Flemming & S. van Wyk, North-Holland, pp. 405-422, 1993.

[9] Janetzko, D., Boerner, K., Jaeschke 0. & Strube, G.: Task-oriented knowledge acquisition and reasoning for design support systems, R. Bisdorff (ed.), First European Conference on Cognitive Science in Industry, Luxembourg, pp.l53-l84, 1994.

[10] Haller, F., ARMILLA ein Installationsmodell: Instrumentarium zur Planung von Leitungssystemen in hochinstallierten Gebaeuden, IFIB, University of Karlsruhe, Germany,1985.

[11] O´Hara, S. E. & Indurkhya, B., Incorporating (re)-interpretation in case-based reasoning, Topics in Case-Based Reasoning - Selected Papers from the First European Workshop on Case-Based Reasoning (EWCBR-93), Wess S., Althoff K.-D. and Richter M. M. (eds.), LNAI, volume 837, Springer Verlag, pp. 246-260,1994.

[12] Boerner, K., Structural similarity as guidance in case-based design. Topics in Case-Based Reasoning - Selected Papers from the First European Workshop on Case-Based Reasoning (EWCBR-93), Wess S., Aithoff K.-D. and Richter M. M. (eds.), LNAI, volume 837, Springer Verlag, pp. 197-208, 1994.

[13] Boerner, K., Towards formalizations in case-based reasoning for synthesis, AAAI-94 Workshop on Case-Based Reasoning, David W. Aha, Seattle, Washington, 177181, 1994.

[14] Plotkin, G. D.: A note on inductive generalization, Meltzer, B. and Michie, D. (ed.), Machine Intelligence 5, American Elsevier, pp. 153-163,1970.

[15] Rosch, E., Cognitive Reference Points, Cognitive Psychology (7), pp. 531-547, 1975.

[16] Gero, J. S., Design prototypes: A knowledge representation schema for design, AT Magazine, 11(4), pp. 26-36, 1990.

[17] Voss, A., (ed.): Similarity concepts and retrieval methods, Gesellschaft fuer Mathematik und Datenverarbeitung mbH (GMD), FABEL-Report no. 13, 1994.

[18] Keppler, H., ACTION: Ein Modul zur Extraktion von Aufgabenstrukturen aus Benutzer-Traces, Gesellschaft fuer Mathematik und Datenverarbeitung mbH (GMD), FABEL-Report no. 32, 1995.

[19] Boerner, K., Conceptual analogy, AAAI 1995 Fall Symposium Series: Adaptation of Knowledge for Reuse, D. W. Aha and A. Ram, November 10-12, Boston, MA, 1995

[20] Boerner, K. & Fassauer, R.: Analogical layout design (SYN). In Modules for design support, Katy Boerner (ed.), Gesellschaft fuer Mathematik und Datenverarbeitung mbH (GMD), FABEL-Report no.35, 1995.

[21] Boerner, K., & Janetzko, D., Case-based learning for design. 2nd European Workshop on Case- Based Reasoning, M. Keane, J. P. Haton & M. Manago (eds), Chantilly, France, 273-281, 1994.