A CASE BASED ARCHITECTURAL DESIGN APPLICATION FOR RESIDENTIAL UNITS

DINA TAHA, SAMIR HOSNI, HISHAM SUEYLLAM
Alexandria University, Faculty of Engineering
Email address: ditaha@idsc.net.eg, samir_hosni@hotmail.com, sueyllam@yahoo.com

AND

BERND STREICH, MICHAEL RICHTER
Technical University of Kaiserslautern
Email address: streich@rhrk.uni-kl.de, richter@informatik.uni-kl.de

Abstract. Case Based Reasoning (CBR) is an AI approach that is widely used in many fields. When it's applied in the design field, it is frequently called Case Based Design (CBD). Its main idea resides in drawing analogies between past cases and the new case to be solved so that the user can make use of past experiences when solving a new problem. The work presented here describes a prototype application under development that makes use of CBR in the field of architectural design. The application is to act as a helping tool for architects in the pre-design phase by supplying them with an adequate number of similar past architectural cases to the design problem they have at hand. The different modules of the application will be presented and discussed, as well as the tools used to develop them.

1. Introduction

1.1. WHAT IS CASE BASED REASONING?

One of the frequently used applications in the AI field is CBR. What made CBR systems a big success is that they are based on the human way of thinking, reasoning and learning (Aamodt and Plaza, 1994), which was proved by the work of Schank (1982), Anderson (1983) and others. According to Riesbeck and Schank (1989), the human thinking is not based on logic but on retrieving and processing the right information at the right time. A typical CBR application has to deal with the following processes:

first, to identify the current problem situation, then search the case-library and select the case that is most similar to the current situation, use this case to solve the current problem, evaluate the solution, and finally update the case-library with the new case solved (Figure 1).

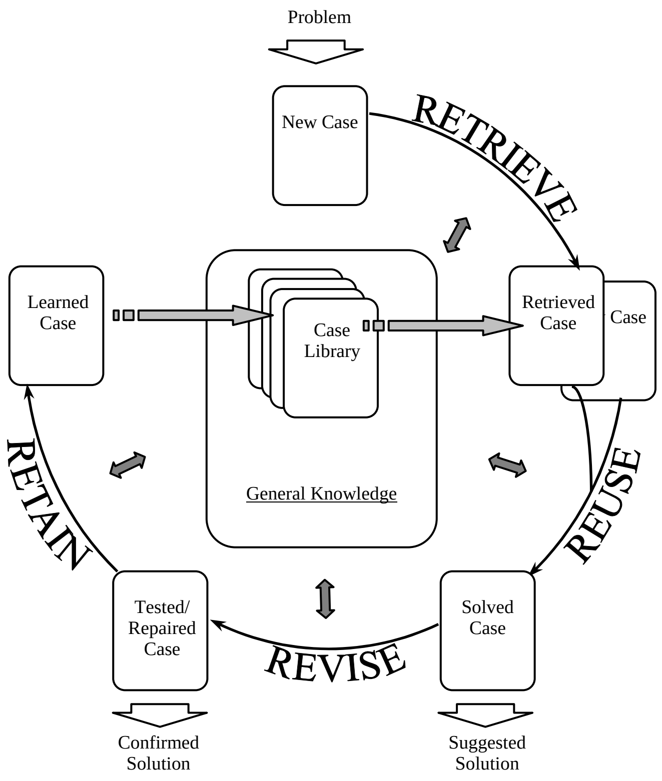


Figure 1. The CBR Circle (Aamodt and Plaza, 1994).

Watson and Marir (1994) state that the CBR cycle rarely exists as described above, but usually the human role takes part within the cycle. They state that many CBR tools act primarily as case retrieval and reuse systems, with the adaptation part being undertaken by managers of the case base. They believe that not being totally automated, and involving the human role in the decision making process should not be considered as a weakness in the system, a point of view which matches the assistant systems' argumentation (Brooks, 1996).

1.2.WHY DO ARCHITECTS NEED SUCH AN APPLICATION?

“The education in architectural design relies heavily upon the use of cases as a vehicle of discourse between the teachers and the students; the hope being that particulars in the given cases offer a holistic view of design issues that are difficult to articulate or view if they were taken up separately” (Dave et al., 1994). The process of using past knowledge to solve new design problems continues from being used in the education process to being widely used in design offices. In order to substantiate that, we performed a survey among a sample of architects. We prepared a questionnaire that addressed several issues such as how architects deal with past experiences, where they look for them, how much time they spend in searching for similar designs, how much time they spend in analyzing them, …etc. The questionnaire was sent to a sample of architects that represent the international community (Egypt, Germany, and USA). Fifty responses were received and it was found that 100% of the architects who participated in the questionnaire stated that they reviewed previous design cases in one way or another. 59.5% of them stated that they *always* or *most of the times* reviewed previous designs either made by themselves or by others, whereas 40.5% stated that they reviewed previous cases only *sometimes*. It was also clear that a significant percentage of the pre-design phase total number of hours was spent in searching for past cases. In particular, 216 hours were spent in searching for cases, while 345 hours were spent is studying and analyzing them (38.5 % and 61.5 %, respectively).

1.3. AIM OF THE WORK

This study aims at developing an architectural application to aid architects in the pre-design phase. The application is meant to reduce the time architects spend to look for similar cases to the problem they have at hand. It leaves the case adaptation process to be manually carried out by the architects themselves, since this is a process that needs the architect’s creativity, not the system’s efficiency. Since architects prefer graphic representation, and usually develop their conceptual designs using different sorts of diagrams and simple sketches, the developed application uses graphical description of the architectural cases instead of textual. With queries represented as bubble-diagrams, the application checks its case-library and retrieves those cases that are similar to the given query.

Such a system can be of direct benefit in housing design for large population groups where the traditional white table approach cannot yield user-specific designs, especially if there is limited time and financial resources committed to the architectural design. And which might result in duplicating inappropriate prototypes in hundreds or thousands, not suited to varying user profiles (Raduma, 2000).

2. How Does this Application Differ from Previous Ones?

During the Second International Conference on Artificial Intelligence in Design the issue of how CBR techniques can facilitate the design process was raised. Researchers were divided into two main categories, the first attempting to develop systems that *do* designs, while the other group was aiming at developing *tools* to assist the human designers (Domeshek and Kolodner, 1993). In the coming section, we'll present some of the existing systems, review their components and functionalities, and recite how our application differs from them.

2.1. ARCHIE-II

Archie-II is the product of collaboration between architects, computer scientists, design researchers and environmental social scientists at the AI lab of Georgia Tech's College of Computing. It is a *case-based design aid* (CBDA) that provides access to past experience so that human designers can adapt the cases for use in a present situation (Zimring, Bafna and Do, 1996; Heylinghen, 2000). It supports architects during the conceptual design phase through three means: (1) raising design issues, (2) proposing responses to design issues, and (3) identifying pitfalls and opportunities (Domeshek and Kolodner, 1993). This is accomplished by not only collecting successful cases, but also by having some cases that didn't work out as was hoped for. Cases were not just described, but the case materials have been drawn from post-occupancy evaluation studies, which were collected for example from the builder, concerning the difficulty of construction; the user, concerning the difficulty of living within certain parts of the building; or even from the maintenance staff. Evaluations of the current buildings serve to highlight the good and the bad features of the design (Domeshek and Kolodner, 1993; Heylinghen, 2000).

The case-library of Archie-II contained at the beginning only cases about courthouses and libraries. Later, tall buildings and handicapped access cases were also added. Since libraries and courthouses were regarded as very complex cases, they were broken down into several pieces of proper size and content (called stories). Each story was represented by: (1) a set of text strings (a title, citation, summary, and full story text), (2) a bitmap graphic intended to accompany and illustrate the story's text, (3) descriptive information including some indication of the major building systems involved and the major outcome, (4) a list of annotation points locating the story on relevant blueprints, (5) links to other related stories, (6) links to relevant guidelines, which suggest ways of thinking about certain problems instead of giving ready-made solutions, and (7) a list of descriptive indexes that provide access to the story in response to user queries.

Users can retrieve design cases by two methods. They can search for a specific design issue in addition to one or more of the following features: building space, functional component, stakeholder, and building's life cycle (i.e. safety in courtroom, glare in reading zones, etc.). Or they can browse the different cases through the bi-directional links embedded by the authors of the stories that link different stories that have some features or guidelines in common (Zimring, Bafna and Do, 1996; Heylinghen, 2000).

Archie-II is another system that applies what Watson and Marir (1994) referred to as systems that rarely exist -as described previously- with all four phases of the CBR-cycle made available. Also here, the manipulation phase is left completely to the user. The architect is supplied with the past stories that are related to the design task at hand, and he bears the responsibility of understanding and applying (or ignoring) the information presented (Domeshek and Kolodner, 1993).

2.2. CADRE / IDIOM

CADRE, and afterwards IDIOM are two systems developed at the Swiss Federal Institute of Technology in Zurich (ETH Zurich) and in Lausanne (EPFL) as an interdisciplinary work between architects and computer scientists. They belong to that category of systems that aims at automatically generating the complete design.

The focus point in CADRE has been on the adaptation of design cases to new environments (Heylighen, 2000). Each case in the case-library was presented as a complete 3-dimensional geometric model. For each case there is a graphic thumbnail as well as several representations that can be useful for the designer. Representations might be geometry-based (i.e. a CAD model), topology-based (i.e. a graph of adjacency relations between the different spaces) or grammar-based; these could be stored with the case or computed on demand. Each case can also be presented in one of four levels of detail. (1) All details: showing all available information about the case; (2) aggregate (or zones): showing groups of spaces (i.e. showing all public spaces); (3) center-lines: showing the geometry of the spaces without details; and (4) bounding box: displaying the smallest bounding volume of the case.

The user browses the case-library, and manually selects the case he prefers to be adapted to his own design program, or he selects multiple cases to be combined together to compose the final design. He also specifies the new problem context, which usually is a site description in addition to a design program. The new site layout and orientation and the specific program requirements result into a set of constraints that is derived from the discrepancies between the selected case in its original context and after its insertion into the new design context. Dimensional and/or topological modifications are automatically carried out to adapt the selected case to the new context. Dimensional adaptations change the values of the numerical parameters that describe the geometry of the design, without removing or adding spaces or elements. On the other hand, topological adaptations alter the layout of the case by adding, removing, or rearranging spaces and elements. In case that all constraints are resolved, one or more design solutions are obtained and displayed to the user. Otherwise the user either has to alter his set of constraints or choose another case to be adapted (Dave et al, 1994; Heylighen, 2000).

IDIOM stands for Interactive Design using Intelligent Objects and Models. The idea behind this project -as Lottaz (1996) describes it- was to develop a prototype of a design tool, which shows the usefulness and convenience of the paradigms of CBD together with constraints, preferences and models in architectural design. It integrates several ideas aiming to assist architects during their design task such as: CBD, constraints to represent knowledge, constraint solving, models to represent domain-knowledge, activation and re-activation of preferences, interactive adaptation of solutions, as well as hints and critique. As previously stated, IDIOM comes from the same working group as CADRE. And through the experience gained from CADRE, there were major modifications in IDIOM. It was noticed that architects rarely reuse a whole design case, as was presented in CADRE. Therefore, the case-library of IDIOM is built up of parts of designs. Cases are divided into five groups: living rooms, kitchens, bedrooms, bathrooms, and hallways. Each case holds geometrical information as well as a list of constraints that needs to remain fulfilled after combination with other cases. There are three types of constraints that control the output: (1) case related constraints, (2) user related constraints, and (3) model related constraints. In addition to these several constraints, the system also accepts preferences with priority. Preferences are sort of constraints, which the designer would like to be fulfilled, but in case they contradict with other fixed constraints, they are deactivated.

As in CADRE, the user browses the case-library and selects the cases he wants to insert into his previously given site, and it’s the system that adapts the selected cases to the new environment, trying to fulfill all fixed constraints and as many preferences as possible. IDIOM calculates the feasible solution space through conflict resolution with preferences and dimensionality reduction, and selects a solution that involves minimal changes to the case and to the current design. The result is meant to be nothing more than a proposal to the architect, which he may accept as is, or modify (Lottaz, 1996; Heylighen, 2000).

2.3. SEED

SEED stands for Software Environment to support Early building Design. Its purpose, as stated by its developers, is to provide support at the preliminary design of buildings in all aspects that can gain from computer support (Flemming, Coyone, and Snyder, 1994). The system should not only analyze and evaluate solutions, but should also be able to rapidly generate designs. SEED is divided into three modules, where each module offers the user several generation capabilities ranging from stepwise construction under the designer's control to the fully automated generation of design alternatives. The first module is concerned with the architectural program. By getting the building context and the overall function and size of the building as input (e.g. elementary school for 300 pupils), it generates an architectural program or design brief. The second module -The Schematic Layout design Module- takes the specifications of the functional components, the context, the budget, and the architectural program as input, and generates a layout of the functional units (e.g. the distribution of the zones over different floors). The last module -Schematic Configuration Design Module- takes the schematic layout and programmatic requirements as input, and generates a 3-dimensional configuration of the building (Flemming, Coyone, and Snyder, 1994; Heylighen, 2000).

2.4. FABEL

FABEL is a joint research project supported by the German Ministry for Research and Technology (BMFT) and is carried out by six different organizations and universities in Germany. It is a hybrid system, which applies case-based, rule-based, as well as model-based reasoning. It contains a collection of tools, called *specialists*, each of which addresses one of the CBR processes (Heylighen, 2000).

Objects in FABEL don't only have a position and a graphic representation as in classic CAD systems, but they also have a type. As described by Gebhardt et al (1997), design objects may be concrete elements of a building like columns, walls, doors, and windows, or they might be abstract elements, such as zones of certain use or climate. Each element is presented by the following: (1) coordinates: the spatial position of the object; (2) aspect: the subsystem to which this element belongs (i.e. construction, façade, heating element, etc.); (3) morphology: which describes the function of the element (i.e. pipelines and beams have a ‘c’ connection function); (4) precision: elements are divided into three levels of precision: zones, bounding boxes and elements; and (5) size: which distinguishes the level of concern being the whole plant, the whole building, one floor, a single room, or an area within a room.

Cases in FABEL can be manually selected from the case-base by the user, or automatically retrieved by the different *specialists* of the system. They are afterwards automatically manipulated through different tools. Some tools are for topological manipulation, such as TOPO, while others are for dimensional manipulation, such as AAAO and AgentEX (FABEL-Report, 1994; Coulon, 1995; Gebhardt et al, 1997; Heylighen, 2000).

2.5. THE DESIGNED APPLICATION

Regarding the CBR processes included in the application, unlike IDIOM, where the user manually selects the cases from the case-library while the application automatically manipulates the selected cases to generate the solution, our application makes use of the automated retrieval and retaining phases of the CBR-cycle, but leaves the manipulation phase to be carried out manually by the user.

While systems like Archie-II use textual case representation in building their case-library, our targeted application uses a simple graphical representation. This graphical representation enables the user to conduct his search via a simple bubble-diagram -which is widely used by architects- while in Archie-II the user searches the case-library by strings. The simple case description enables the user to easily update the case-library with his newly solved design problem, so that the case-library reflects new designs. On the other hand, in Archie-II due to the sophisticated case representation, adding any cases to the case-library is a time consuming and complicated task for the system developers let alone for the users. This leads to a static case-library with a limited number of cases.

Cases in IDIOM as well as in Archie-II are no complete buildings, but just separate rooms, zones or functions. It is thought that it’s not the room itself that can be successful or not, but it’s the room within its context that counts. Using parts of cases as in IDIOM, or describing selected functions and zones as in Archie-II should not for sure tune with all users of the system.

3. Why Residential Units?

As stated earlier, the application is intended to be a helping tool for architects during the pre-design phase. For the first prototype, low-income and middle-income residential units are chosen to build up the case-library. This is due to several reasons:

1. Residential units are typical buildings with a *finite* number of possible layouts (specially when we limit them to low-income and middle-income units). That makes it feasible to create a case-library that contains almost all possible residential layouts.
2. Also being a typical building-type permits a certain level of repetition, especially in early conceptual design phases (Chandrasekaran, 1990). That encourages architects more to view old cases and analyze how some design problems were tackled in them.
3. The more limited the budget for any design project is, the more an important role time plays. Since low-income and middle-income residential design projects are usually low budgeted and limited in time, we considered an application that reduces the amount of time spent on such projects to be useful and appreciated.
4. Last but not least, the more design constraints there are, the easier the design task is. While developing our first prototype we wanted to choose a simple building type, which consists of a limited number of elements and relations to simplify the system design. Residential units were the most appropriate building type, especially when we limited the application to low-income and middle-income units. This ensures a limited number of spaces and connections. We also added the constraint that all spaces should be within one level. Duplexes and multi-level residential units were excluded from our case-library.

4. How Does the Application Work?

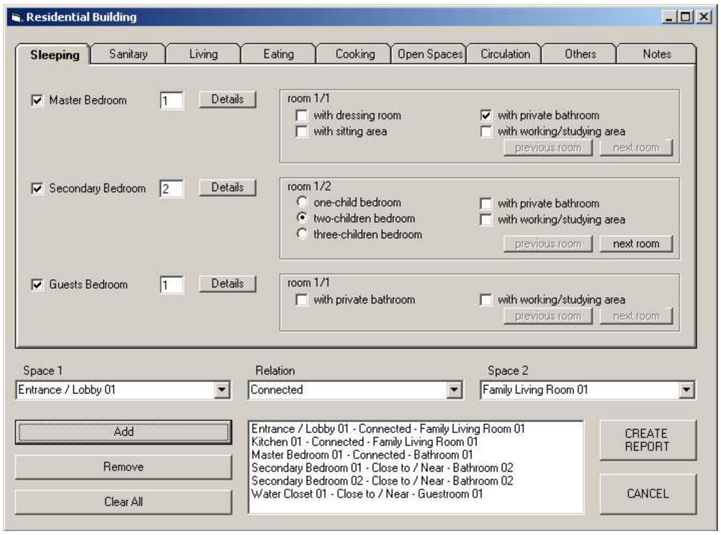
4.1. GENERAL IDEA

While designing this application, we always had the processes architects go through in consideration. Architects may differ in their methods of developing designs, but still, there are some broad lines they all apply. As Domeshek and Kolodner (1993) put it: “the designer needs to learn about the client’s problem and about prior art; unfortunately, this is a time consuming, costly, and omission-prone step in design. In architecture, common methodology calls for a special team - the programmers- to gather data on requirements and prepare a program that makes all requirements concrete. The designers then work to satisfy the client's concrete needs as made explicit in the program." They also state that designers tend to visit buildings, survey existing literature, and try to relate their current design task to famous precedents. And these are the same lines we tried to follow through the design of our application.

4.2. FIRST MODULE

Any design project starts with writing down the program. In some cases, this program is made available to the architect at the start of the project, and in other cases the architect develops it with the client. This program includes what types of rooms should be included, their sizes, the (connectivity) relationship they have to each other, and any special requests the client might express.

Our application also starts with writing down the design program. A user-friendly form (Figure 2) was designed, giving the architect a simple way to fill out the clients' needs.



[✓] Master Bedroom 1

room 1/1
[ ] with dressing room
[✓] with private bathroom
[ ] with sitting area
[ ] with working/studying area

[✓] Secondary Bedroom 2

room 1/2
( ) one-child bedroom
(●) two-children bedroom
( ) three-children bedroom
[ ] with private bathroom
[ ] with working/studying area

[✓] Guests Bedroom 1

room 1/1
[ ] with private bathroom
[ ] with working/studying area

Space 1: Entrance / Lobby 01
Relation: Connected
Space 2: Family Living Room 01

Entrance / Lobby 01 - Connected - Family Living Room 01
Kitchen 01 - Connected - Family Living Room 01
Master Bedroom 01 - Connected - Bathroom 01
Secondary Bedroom 01 - Close to / Near - Bathroom 02
Secondary Bedroom 02 - Close to / Near - Bathroom 02
Water Closet 01 - Close to / Near - Guestroom 01

Figure 2. Filling out the form.

Using Visual Basic 6 © (VB) a multi-tab form was designed. The several tabs stand for the different room-types a residential unit might include, such as sleeping rooms, sanitary rooms, living rooms, etc. Using this form, the architect decides the types and the number of rooms the design should include. Some of the rooms have a *details button* that enables the architect to specify special features for each specific room. For example, secondary bedrooms might be for one child, two children or three children; they might include a private bathroom, or have an extra playing/studying area. These details are used to compute the area of each room. An extra tab is added to allow both the architect and the client to specify any special wishes or ideas. The form also allows the user to specify the (connectivity) relationships between the different rooms selected (i.e. the guestroom should not be adjacent to the master bedroom, etc.). It gives the user the possibility to add, revise and remove any notes or relationships whenever he wants to.

After completing this form, different reports and files are automatically generated. The first report is intended for the client to revise to make sure that all his requests are mentioned on paper; it acts as a contract between the client and the architect. Two other files are generated from that form, which will be used by the second module of the application.

4.3. SECOND MODULE

Based on the selections the architect made in the first step, a bubble diagram is automatically generated (Figure 3). This bubble reflects only the specific rooms with the calculated areas, as well as the relationships between the different rooms. The current position of the rooms doesn't play any role in the retrieval process in the third module. That's why we don't generate all possible solutions for the bubble diagram, but only the first possible solution that satisfies all relationship conditions.

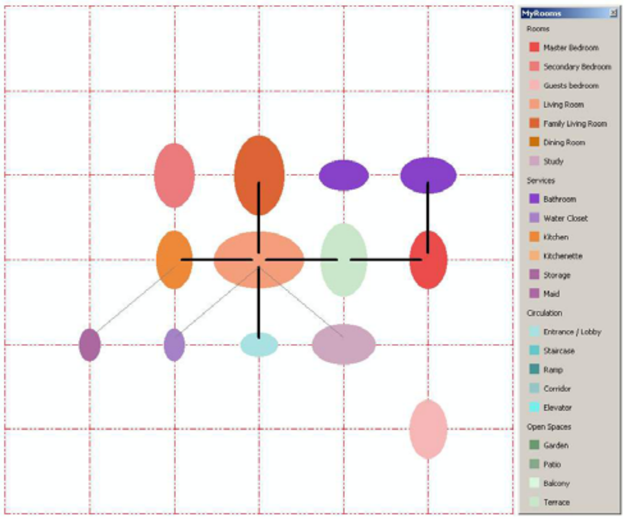


Figure 3. The generated bubble diagram.

This module was realized with ArcMap ™ 8.2 by customizing it using Visual Basic for Application © (VBA). In this module we make use of some features widely used in Geographic Information Systems (GIS) applications (such as topological relations). So instead of spending much time in developing those features from scratch, we decided to save time and make use of the available features in ArcMap, especially that it allows customization with VBA.

The two other files that were generated by the fist module are used here. The first file is a tab-delimited file that includes each room and its area, while the second file includes the different (connectivity) relationships generated from the VB form. Within ArcMap, several functions and routines were written that make use of those two files to specify a location for each room that satisfies the relationships conditions. Using these locations, a bubble diagram is generated. At this point, the architect might accept the bubble diagram as is, or he might modify it according to his views and concepts. He's allowed to change room sizes, add and/or remove rooms, alter room-connectivity, and specify room-orientation if he likes to. To make this process simple, the bubble diagram is generated using graphics, not features (feature is an ArcMap terminology that describes an element - may it be a line, a point or a polygon- that has a link to the database; while graphics are elements drawn on the screen with no linkage to any tables). After this diagram is finalized, the graphics are transformed into features, so that each element (room or connection) has a record in the database, and its record is updated with the modifications the architect might have introduced. This module ends by generating the query that will be used in the third module. The query is automatically generated from the final version of the bubble diagram the architect has modified. It represents the different rooms, their sizes and orientation as well as their topology.

4.4. THIRD MODULE

It's in the third module, where we search the case-library for cases that are similar to the generated query. Again, for not reinventing the wheel, we used a commercial system as a base for our application. As stated in CBR-Works' reference manuals (tec:inno GmbH, 2000): "CBR-Works by tec:inno is a software development package suited for intelligent solutions in a variety of domains and environments. It supports the user to design complex knowledge models. Concepts, types, similarity measures, weights and filters can be created, edited and maintained." For our application we had to use the development version of the program instead of the commercial one, to be able to adjust the similarity measures. This is due to the complexity of the model and the tailored similarity measures that an architectural object requires.

Since our aim is to develop a system that is based on graphical representation of the cases, we had to develop a model that can handle such representation. The developed model (Figure 4) represents the residential unit by the following objects:

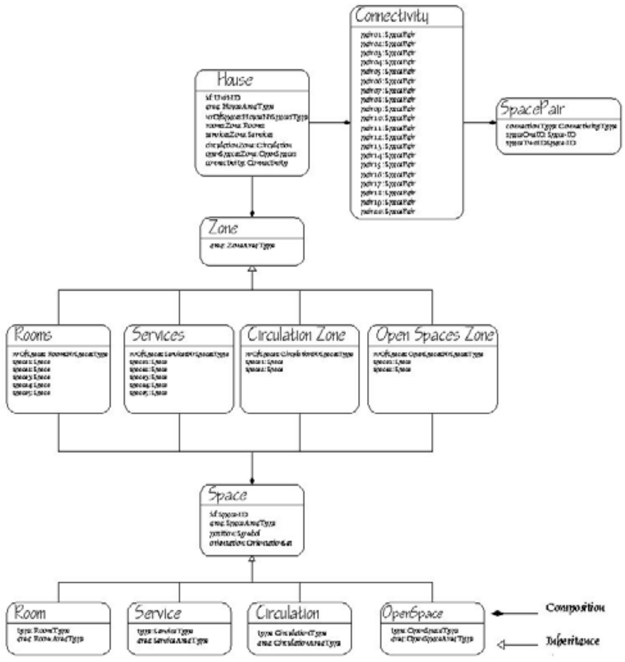


Figure 4. The developed Model.

Each residential object has a set of simple attributes (ID, area, and number of spaces), as well as a set of complex attributes (rooms-zone, services-zone, circulation-zone, open spaces-zone, and connectivity). Complex attributes are in fact other objects that have themselves different attributes. For example, rooms-zone is an object with two simple attributes (area and number of spaces) and other five complex attributes representing five spaces that this zone might include. Each space is represented by five simple attributes (ID, area, position, orientation, and space-type).

Each attribute is limited to a certain data type and a range. Some attributes have a simple data type such as integers, symbols or Booleans; but in most cases enumerations and sub-ranges are to be used to limit the value any attribute should have. For example, a space within the rooms-zone has its type limited to: master bedroom, secondary bedroom, guestroom, living room, family living room, dining room, or study; while a space within the services-zone has its type limited to: kitchen, kitchenette, bathroom, water closet, maid's room, or storage. This limitation helps reducing the number of comparisons made while comparing the query with the different cases.

A default set of weights is given to the different attributes that describe the residential unit. The architect will be given the possibility to adjust those weights according to his needs. While some attributes are used to compute the similarity between the query and the cases, other attributes are used as a filter to reduce the number of cases that are taken into consideration.

4.5. FOURTH MODULE

After calculating the similarity between the query and the cases in the case-library, the user is presented with those cases that are mostly similar to his query. The architect might use these cases to develop the design at hand. Then he should update the case-library with the new solved case. At the time of writing this paper, this fourth module for updating the case-library was not yet developed.

5. Evaluation and Conclusion

The system will be tested by giving two groups of architecture-students the same design assignment. While one group will be trained to use our developed application and given access to it, the other (control) group will only use the traditional methods. By comparing the performance of the two groups with regard to the quality of their designs, the time they needed to fulfill the task, and the simplicity of using the application, we will be able to assess the impact of using this application on the efficiency and the creativity of the designers.

6. What Else Can Be Done?

For the time being the cases are compared simply by checking the room area, orientation, type and connectivity. It is planned to introduce more special cases within the comparison phase. For example, adjacent rooms could be integrated into one bigger room, and big rooms could be divided into several smaller rooms (i.e. a small dinning room, which is adjacent to a small living room, could be compared to a big living room with a dining corner).

It is also intended to apply the system over the Internet, so that the case-library could be updated by several architects with diverse styles, instead of being a stand-alone system with a very limited case-library.

Acknowledgements

First of all, we would like to thank both Quality Standards Information Technology and tec:inno for providing us with ArcMap and CBR-Works respectively. We would also like to thank Dr. Armin Stahl for his continuous support with the know-how of CBR-Works.

References

Aamodt, A. & Plaza, E.: 1994, Case-Based Reasoning, *Artificial Intelligence Communications,* **7(**1), 39-59.

Anderson, J. R.: 1983, *The Architecture of Cognition,* Harvard University Press, Cambridge.

Brooks, F. P. 1996, The Computer Scientist as Toolsmith II, *Communications of the ACM,* **39**(3), 61-68.

Chandrasekaran, B.: 1990, Design Problem Solving: A Task Analysis, *AI Magazine,* **9**(4), 22-36.

Coulon, C.-H.: 1995, Automatic Indexing, Retrieval and Reuse of Topologies in Architectural Layouts, *in* M. Tan & R. Teh (eds),*The Global Design Studio - Proceedings of the Sixth International Conference on Computer-Aided Architectural Design Futures*, Singapore, pp. 577-586.

Dave, B., Schmitt, G., Faltings, B., and Smith I.: 1994, Case-Based Design in Architecture, *in* Gero, John S. and Sudweeks, F. (eds), *Artificial Intelligence in Design,* Kluwer Academic Publishers, Doordrecht, The Netherlands, pp. 145-162.

Domeshek, E. & Kolodner, J.: 1993, Using the Points of Large Cases, *AI EDAM,* **7**(2), pp. 87-96.

Flemming, U., Coyne, R., & Snyder, J.: 1994, Case-Based Design in the SEED System, K. Khozeimeh (ed), *American Society of Civil Engineers,* pp. 446-453.

Gebhardt, F., Voß, A., Gräther, W., & Schmidt-Belz, B.: 1997, *Reasoning with Complex Cases*, Kluwer Academic Publishers, Boston.

Heylighen, A.: 2000, *In Case of Architectural Design: Critique and Praise of Case-Based Design in Architecture*, PhD, Katholieke Universiteit Leuven.

Lottaz, C.: 1996, *Constraint Solving, Preference Activation and Solution Adaptation in IDIOM,* Swiss Federal Institute of Technology.

Raduma, P.: 2000, Modeling CBR Representation for Architectural Design Reuse, *in* S. Staab & D. O'Leary (eds), *Bringing Knowledge to Business Processes*, AAAI Spring Press, pp. 1-7.

Riesbeck, C. K. & Schank, R.: 1989, *Inside Case-Based Reasoning,* Erlbaum Associates, Hillsdale, NJ.

Schank, R.: 1982, *Dynamic Memory: A theory of Reminding and Learning in Computers and People*, Cambridge University Press.

Survey of FABEL: The FABEL Consortium: 1994, Gesellschaft für Mathematik und Datenverarbeitung mbH, Report no. 2.

tec:inno GmbH: 2000, *CBR-Works 4: Reference Manual*, Kaiserslautern.

Watson, I. & Marir, F.: 1994, Case-Based Reasoning: A Review, *The Knowledge Engineering Review*, **9**(4).

Zimring, C., Bafna, S., & Do, E.: Structuring cases in a case based design aid, J. Vanegas & P. Chinowsky (eds), pp. 308-313.