in: Proc. IEA/AIE '91 - 4th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Kauai, Hawaii, June 1991, S. 409-418

A Knowledge-based Approach to Intelligent CAD for Architectural Design

Mattos, N.M., Deßloch, S., Leick, F.-J.
University of Kaiserslautern, Department of Computer Science
P.O. Box 3049, D-6750 Kaiserslautern, Germany
e-mail: mattos@informatik.uni-kl.de

ABSTRACT

In this paper, we present the conception of an intelligent CAD application for architectural design and describe its realization using the knowledge base management system KRISYS. Our application integrates all relevant information regarding the design of one-story houses into a product model. It divides the design process in three main phases, in which the product model is used as basis for the execution of the functional, topological, and geometrical design. In the paper, we give an overview of this product model and describe the system activities that are carried out in each of the design phases.

1. INTRODUCTION

Over the last few years, computer support in product planning, design, and construction has gained more and more importance within the computer science community. A lot of research activity has been focusing on the development of intelligent CAD systems aimed at a continual, efficient, and integrated support of the design process as well as of its overall organization [GAPR88,SGL89].

In intelligent CAD systems, the designer is not the only active unit within the overall design process (as is the case in conventional CAD systems). The CAD system can also exhibit an active behavior, providing a more intelligent interface to the user. It supports system-enforced checking of complex integrity constraints even across design steps to guarantee a consistent design. It can offer appropriate design hints, relevant problem solutions, refined simulation results, and adequate diagnostic information at all stages of the design process [Ph88,SR86]. In some sense, there is a kind of 'partnership' between designer and system, where it is possible to switch between automated design guided by the system (e.g., in standard cases) and human design controlled by the designer's decisions (e.g., in special cases).

One main goal of intelligent CAD is therefore the improvement of the design process by incorporating not only geometrical, but all relevant design aspects as well as functional dependencies at each design step. As a consequence, the design process is done iteratively, but not by means of stand-alone tools converting the output data structures of previous design steps into internal data structures and vice-versa before the tools can accomplish their functions. Intelligent CAD systems, in contrast to conventional CAD systems, divide the design process into design phases which subsequently supplement incomplete object specifications, stepwise enrich object descriptions with functional, topological, and geometrical aspects, and support improvements or refinements by means of a feedback to each previous phase. Since all design phases refer to the same design objects, all aspects of these objects are represented in a unique and non-redundant manner (in the so-called product model), allowing for a consistent and uniform object handling.

Considering the above characteristics of intelligent CAD systems, we have constructed a prototype application for architectural design. It integrates of all relevant design information in a product model which is then used as basis for the execution of the functional, topological, and geometrical design of one-story houses. Corre-

in: Proc. of the 4th Int. Conf. on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, Koloa, Kanai, Hawaii, 1991.

sponding to these three different design specifications, the design process is divided into three main phases, thereby reflecting the methodology of human architects.

In this paper, we describe our intelligent CAD application for architectural design, sketching its conception and giving hints as to the approaches exploited to accomplish the intelligent CAD concepts mentioned above. Our application has been implemented on the basis of a knowledge base management system (KBMS) called KRISYS [Ma91, Ma88b], that integrates artificial intelligence (AI) and database system (DBS) techniques in an effective way [DHMS90, DHMM89]. Due to the space limitations of this paper, we cannot present such an implementation of architectural design in detail. For this reason, we will concentrate on the representation of an enriched design description and on how to achieve the desired active system behavior.

2. AN OVERVIEW OF THE APPLICATION

The goal of our intelligent CAD system is to support the architectural design of one-story houses for families, based on requirements and needs specified by the user. All these specifications as well as all relevant information concerning the design are explicitly represented in an integrated product model internally managed by our system. This product model contains several submodels describing geometrical, topological, and functional aspects of the objects under design, thereby providing distinct views of our representation. The definition of the product model was accomplished appropriately by exploiting the object-centered representation framework of KRISYS.

KRISYS allows a natural description of design objects, integrating all its aspects into one KB object. Abstraction concepts (classification, generalization, association, and aggregation) [Ma88a, BMW84] were used as the basic mechanisms for describing the organizational semantics of the application domain. Built-in, abstraction-related, reasoning facilities (inheritance, set-properties, membership stipulations, and implied predicates) [Ma88a, RHMD87] were exploited to continuously guarantee the structural and semantic integrity of the product model. Distinct aspects of the design object were represented using different abstraction concepts or distinct hierarchies of the same concept. One hierarchy was employed for representing geometrical, another for functional, and a last one for topological information about the object. This has the advantage that different aspects of a design are easily distinguishable in the model. Following this approach, each room of a house is described by an object, which is either a component of one of these hierarchies, or contains properties relating it to them. Therefore, a room can be viewed from a functional, a topological, and a geometrical point of view and acts as a link between the different hierarchies, thereby guaranteeing the overall consistency of the design.

In the following, we will first explain how the distinct aspects of a design object are represented by means of abstraction hierarchies and integrated into the room descriptions. All these hierarchies are manipulated during the design process, dynamically generating objects for the representation of important design information and incrementally completing the specification of such objects in order to construct an architectural sketch of the house. These manipulations, which are performed in different phases of the design process, are described in more detail in the subsequent sections of this chapter.

2.1. A GENERAL OVERVIEW OF THE REPRESNTATION

As already mentioned, the house under design can be seen from a functional, a topological, as well as a geometrical point of view.

2.1.1. FUNCTIONAL VIEW

The functional aspects are expressed in terms of <u>usages</u>, which can be associated with the rooms of a house. We have introduced a generalization hierarchy into our system, which contains the information about usages relevant for the design process (see figure 2.1 for a partial view of this hierarchy). There can be different levels of abstraction when considering the usages. For example, both 'reading' and 'watching-tv' may be seen as the more abstract usage 'leisure'. These different levels of abstraction are also reflected by the levels of detail in the requirements for the house specified by the designer (see chapter 2.2 for details).

The usage hierarchy contains information about <u>all</u> the usages the system knows about. This information is exploited for a specific design by creating instances of the usages which are required for the house (either by specification of the designer or by default assumptions). In our example (see figure 2.1), usage instances are created

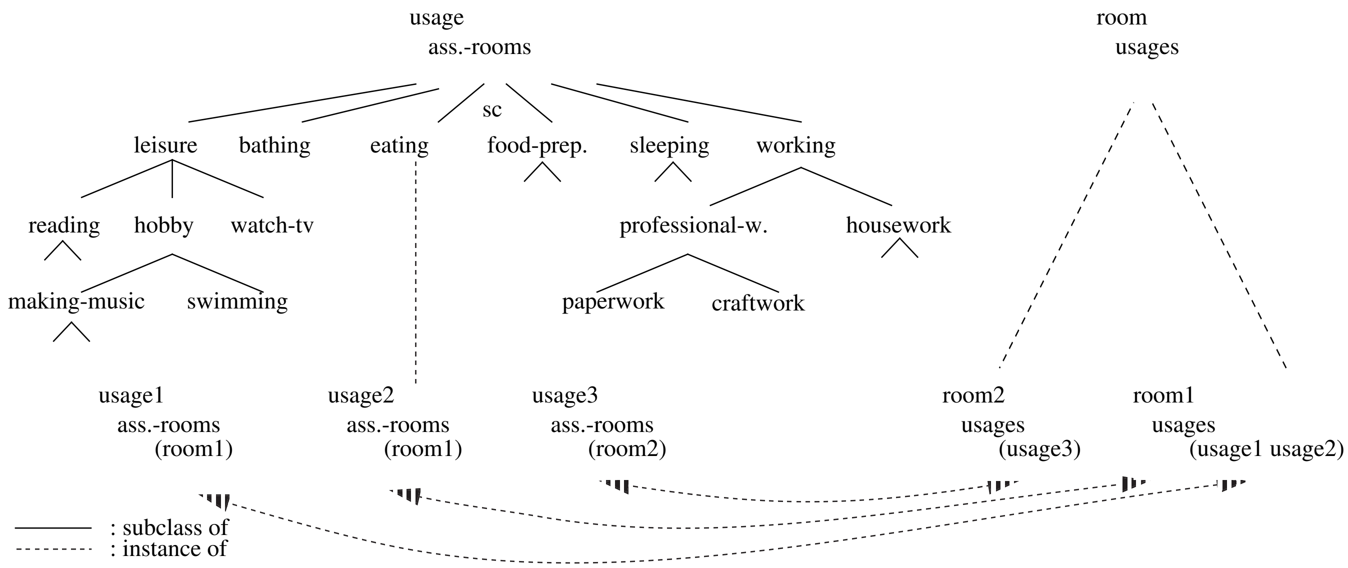

Figure 2.1: Representation of functional aspects

among others for 'food-preparation', 'eating', and 'leisure'. During the design process, rooms are created (as instances of the class 'room') and associated with the specific usages in order to appropriately fulfill the functional requirements for the house. The attribute 'usages' in the class 'room' and 'ass.rooms' (associated-rooms) in the class 'usage' (as the 'inverse' attribute), which are inherited respectively by the rooms and usages, are used to represent this relationship. In figure 2.1, for example, 'room 1' is associated with the instances of 'leisure' and 'eating', while 'room 2' is supposed to be used for 'food-preparation'.

2.1.2. TOPOLOGICAL VIEW

The topological aspects of a house under design are characterized by <u>orientations</u> (e.g., a room is located at the south side of the house) and by <u>adjacencies</u> in the sense that two rooms, for example a bedroom and a bathroom, are located next to each other to achieve a direct connection between them (e.g., through a door). They are represented in a separate hierarchy with the class 'topology' as its root and the two subclasses 'orientation' and 'adjacency' (see figure 2.2). The instances of these two subclasses describe the specific topological constraints known by the system, which are established either through legal restrictions or default-assumptions. Please note that orientations as well as adjacencies are primarily specified according to usages, and not to rooms. For example, the orientation 'sleep-east' is associated with the usage 'sleeping', and 'bathing-sleeping' relates the usages 'bathing' and 'sleeping', which is specified in the 'usages'-attribute of the objects. This is important, since a single room may be considered with respect to several usages (e.g., room 1 in figure 2.1) and is then affected by several topological restrictions. The topology hierarchy itself is not modified or extended (e.g., by creating new instances) during the design session. It is merely used to propagate the topological restrictions to the

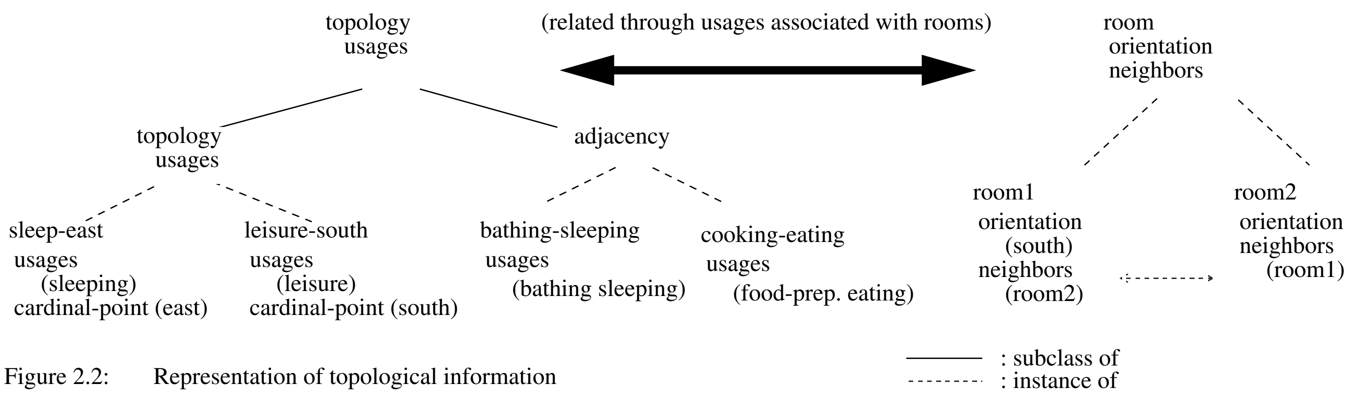

Figure 2.2: Representation of topological information

rooms created by the system according to the specific usages associated with them. Rooms are therefore described by two topological attributes: 'orientation', which contains the cardinal points of the rooms according to the orientations of usages, and 'neighbors', relating adjacent rooms.

2.1.3. GEOMETRICAL VIEW

Besides functional requirements and topological information, the geometrical aspects are fundamental for actually representing and producing an architectural sketch of the house under design. These are characterized in terms of geometrical areas. A house consists, from this point of view, of several areas which may (recursively) be split into subareas that finally are identical to specific rooms. We have represented this tree-like structure having rooms of the house as its leaves with the help of the aggregation concept: an area is considered as an aggregate consisting of component areas (i.e., subareas). This aggregation hierarchy is shown in figure 2.3. All areas are instances of the class 'area' (e.g., 'area 1'), thereby inheriting geometrical attributes such as 'position', 'dimension' (i.e., lengths of the sides), and 'size'. They can be components of other areas (e.g., 'area 1' is a component of 'house') and in turn have other areas as components (e.g., 'area 2' and 'area 3'). Since rooms are considered as areas themselves, the class 'room' is a subclass of 'area', but with the restriction that rooms (e.g., 'room 1' and 'room 2') cannot have other areas as components (i.e., they are 'atomic' components).

2.1.4. INTEGRATED VIEW

As already mentioned, rooms play an important role in the integration of the different hierarchies, acting as a link between them. This is illustrated in figure 2.4, where the full specification of the room 'room 1' is shown. From a functional point of view, 'room 1' is associated with the usages 'usage 1' and 'usage 2', which are respectively instances of 'leisure' and 'eating'. Its topological role is described by its orientation ('south') and its neighbors ('room 2'). Geometrical aspects are characterized by its attributes 'position', 'dimension' and 'size', and by the fact that it is a component of 'area2'. An integrated view of this information is especially relevant for the maintenance of the overall consistency of the different hierarchies. For example, the size of a room is closely related to the usages intended for it. This information has to be exploited by the process designing the geometrical areas of the house.

2.2. THE PROCESSING PHASES OF OUR DESIGN SYSTEM

In the following sections, we will take a closer look at the different design phases supported by the system and show how they utilize the information represented in the above hierarchies. Such hierarchies not only provide three different views of an architectural design, but also reflect the three different phases carried out by human architects during a design process.

2.2.1. GENERATING A FUNCTIONAL VIEW OF THE HOUSE

Specifying usages

In the first phase of the design process, our system questions the user as to his/her requirements. These may be formulated on a high semantic level (e.g., 'I need to work at home and preferably not on the side of the street.'). Furthermore, the specification of such requirements may be incomplete in the sense that they are not suffi-

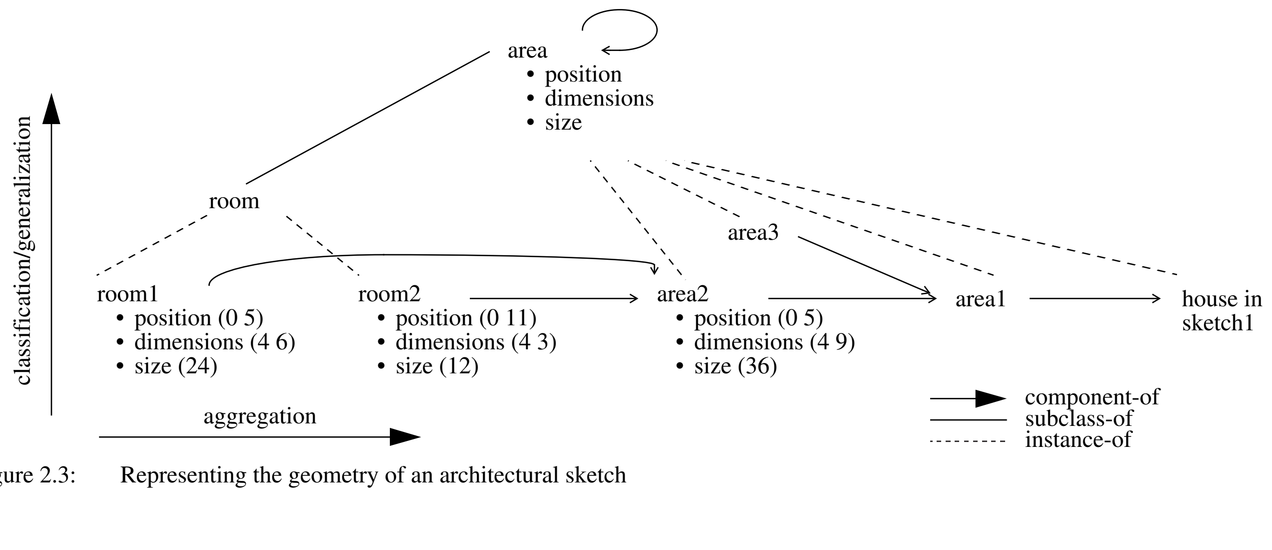

Figure 2.3: Representing the geometry of an architectural sketch

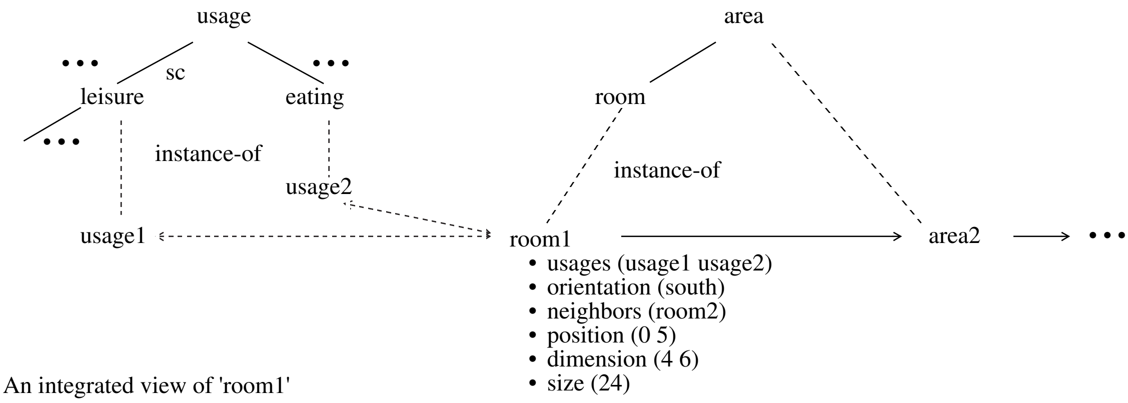

Figure 2.4: An integrated view of 'room1'

cient to directly achieve a final design stage. For this reason, the system utilizes its knowledge (expressed by rules representing standard requirements, laws, etc.) to supplement the user-sketched blueprint. In other words, a significant part of the knowledge of our architectural design system is represented as rules, which are responsible for the tasks of the system, such as the derivation of activities or usages that are standard in every house or required by law (e.g., eating, sleeping, heating...) and are not necessarily specified by the user.

There are several levels of abstraction in which activities can be derived by the system. They correspond to the levels of the generalization hierarchy defining the usages (figure 2.1) and reflect different degrees of detailing of the functional aspects of a house. Actually, every level expressed in this hierarchy represents a valid degree of detailing of a functional design. This means that our system, just like human architects, can specify functional requirements at any level, depending on the degree of detailing it 'believes' to be necessary in each particular case. That is, it makes use of information about the 'context'[1] in which a particular house is being constructed in order to dynamically determine whether it is necessary to further detail one or another functional aspect of the house. For example, if the members of a family play different sports or have several hobbies, a detailed analysis of the partial hierarchy under 'leisure' (see figure 2.1) is required, whereas a house in which the parents wish to work needs a detailed investigation of activities under 'working'. So, what makes the system move further into a more detailed level is its 'ability' to consider information about the 'context' and thereby to realize the necessity to further detail some functional aspects.

Every time the system enters a more detailed level, the user has the chance to express his/her requirements that correspond to this level and the system itself derives additional activities by default assumptions. For example, when moving down from 'leisure' into the next lower level, the system can assign the activity 'watch-tv' even if the user does not give any other information about his/her activities regarding 'leisure'. Whenever an activity is defined either by the user or the system, an instance of the corresponding 'usage' class is created by the system in order to represent this functional requirement.

The result of this design step is therefore a set of requirements expressed in terms of activities to be supported by the house being constructed, that is, a functional description of the house represented in terms of the instances created in all levels of the 'usage' hierarchy. Note that in this first phase, our system does not deduce any kind of rooms (such as kitchen, bedroom, etc.) that the house has to have. It first abstracts from particular rooms, analyzing a house only from a functional point of view in the same way as human architects. This is particular important because several combinations between activities and rooms are possible when specifying rooms in the following step of this design phase. For instance, the activity 'eating' can be 'associated' with a kitchen, a large living room, a dining room, etc. Since activities directly influence the characteristics of a room (e.g., the size of a room strongly depends on the activities and the number of people performing such activities in the room), they are viewed as a group of constraints that should be satisfied after the rooms of the house have been specified.

1. This involves general information about the user and his/her family, such as job, sex, age, habits, etc.

Deriving rooms

In the next step, our design system specifies rooms that can satisfy the assigned functional constraints. These rooms have, at this step, no geometrical or topological information associated with them, but already represent the actual rooms that have to be considered when generating the real sketch of the house later on. This generation process is carried out in a stepwise fashion in which every instance of usage that was created in the first step is examined and assigned to either an existing room or to a room just created for this purpose. Since the associations between usages and rooms can be defined either by the system or the user, this process is guided by integrity constraints that ensure that both the user specifications and deductions of our system will generate room descriptions reflecting a semantically correct state.

Our system exploits the integrity constraints to control the overall process of this second step, checking the actions of the user and triggering systems operations to complete user specifications by enforcing the semantic integrity of the KB. When the user directly specifies an activity for a particular room, our system exploits such constraints to check whether this new activity is compatible with a previously asserted usage. There are statutes that strictly determine incompatible usages. The architect has, for example, to consider that facilities for both cooking and bathing should not be provided in the same room. For this reason, we have attributes expressing 'contradictory activities' in the objects describing activities, which contain all incompatible activities for every activity in the KB. Procedures represented as demons attached to the attribute expressing the usage of a room are immediately activated when a usage is added in order to check if one of the old usages is a 'contradictory activity' of the new one. Upon detecting an inconsistency, an error is reported, automatically eliminating the previous change of the KB.

Another approach applied by the system when integrity constraints are violated is to trigger additional operations to transform an inconsistent KB state into a consistent one, instead of rejecting the changes. This triggering mechanism is automatically carried out by our design system, whenever, for example, information is incompletely specified by the user. The mechanism is also desirable within our integrated product model, where changes concerning one representation of an object (e.g., its functional aspects) should automatically cause changes in the other representations (e.g., the geometry). Such an approach is realized by either demons to trigger the appropriate changes or by rules to dynamically derive information and insert it into the room description. As already mentioned, the user of our design system may specify activities or usage for the house, but need not to relate them to any rooms. This specification is then completed by the system in this second step. For example, when the user specifies the activity 'professional-working', the system activates rules associated with the 'professional-working' object in the KB. These rules either generate a new room for the activity (e.g., the office room) or establish a relationship between the activity and an already existing room (e.g., the living-room), considering, for example, extra places for a work desk. Note that the modification of the size of the living-room because of the association of the activity is an example of the use of rules to keep track of the dependencies between functional and geometrical aspects of the product model. This kind of dependency is exploited by our system at the next phase of the design process to complete the descriptions of the rooms.

2.2.2. DETERMINING ROOM CHARACTERISTICS

Defining orientations and adjacencies

The goal of the second design phase is to completely specify the topological aspects of the house and the geometrical aspects that directly depend on the functional and topological ones. Again, these aspects are either determined by the user (e.g., 'I preferably do not want to work on the side of the street') or derived by the system. Both kinds of definitions are restricted by the constraints and dependencies existing in this application world which are described in our product model as constraints involving various levels of granularity of knowledge: attributes of objects (e.g., the size of a main sleeping-room may not be smaller than 12 square meters), objects (e.g., the functional view of an object, such as the activities associated with it, must be in accordance with its geometrical and topological view, i.e., its size and position within the house), and groups of objects (e.g., connections between rooms). Note that most of these constraints are derived from laws and statutes that rule minimum areas for rooms, the amount of direct illumination in square meters that a room must have and consequently its position within the house (inner or outer), the kind of ventilation (direct or indirect)

that a room should have and therefore its location and connections to other rooms.

System derivations are carried out by exploiting the topology hierarchy (figure 2.2), in which orientations and adjacencies of 'usages' are defined, in order to propagate this information respectively to the attributes defining orientation and neighbors of the rooms. Geometrical characteristics, such as the size of the rooms, are also derived on the basis of the usages that have been assigned to a particular room and of laws or statutes as previously mentioned.

Improving the design diagram

At the end of this step, our system has a complete description of the rooms, containing the location of a room within the house (e.g., bathroom and bedroom should be at the east side of the house), the connections between rooms (e.g., the kitchen should be connected directly to the entrance, but not to the bedroom), sizes of rooms, etc. This description is then used by the system to produce the so-called design diagram, which is usually employed by human architects to analyze and ameliorate complex designs before a concrete sketch is generated. This diagram is actually implicitly represented in our product model, given by the rooms, their topology, geometry, and usages. It is a graph representation of the object being designed (in our case a house) in which nodes and edges symbolize respectively rooms and connections between rooms (see figure 2.5).

Improvements are then performed by transforming the graph into another one, presenting some important characteristics. First, such a graph has to be planar since connections in a house cannot cross each other. This can be achieved by either removing some connections or creating additional rooms, such as a corridor. Secondly, the graph cannot have regular rooms with an excessive number of junctions to other rooms, as for example, the 'living/dining' room illustrated in figure 2.5a. To reduce the number of such junctions between regular rooms, special (connection) rooms (such as corridors, foyers, entrance halls, etc.) can be introduced for this purpose, as shown in figure 2.5b. Note that the transformation of the graph has to be carried out in several steps since the introduction of a connection room (the foyer in figure 2.5b) does not necessarily reduce the number of junctions between regular rooms to an acceptable minimum. In the example, a corridor was then created to reduce the connections of the 'living/dining' room and of the bathroom (figure 2.5.c). Finally, further transformations may be performed to eliminate inadequate connections (e.g., the connection between the two bedrooms in figure 2.5b is through the bathroom), to reduce the communication between groups of rooms (e.g., rooms in a private area such as bedrooms and bathrooms should not be connected directly to rooms in the social area), to split very large rooms (thereby reorganizing the connections), etc.

Certainly, the system considers the constraints mentioned above at each transformation of the design diagram in order to produce only diagrams satisfying them.

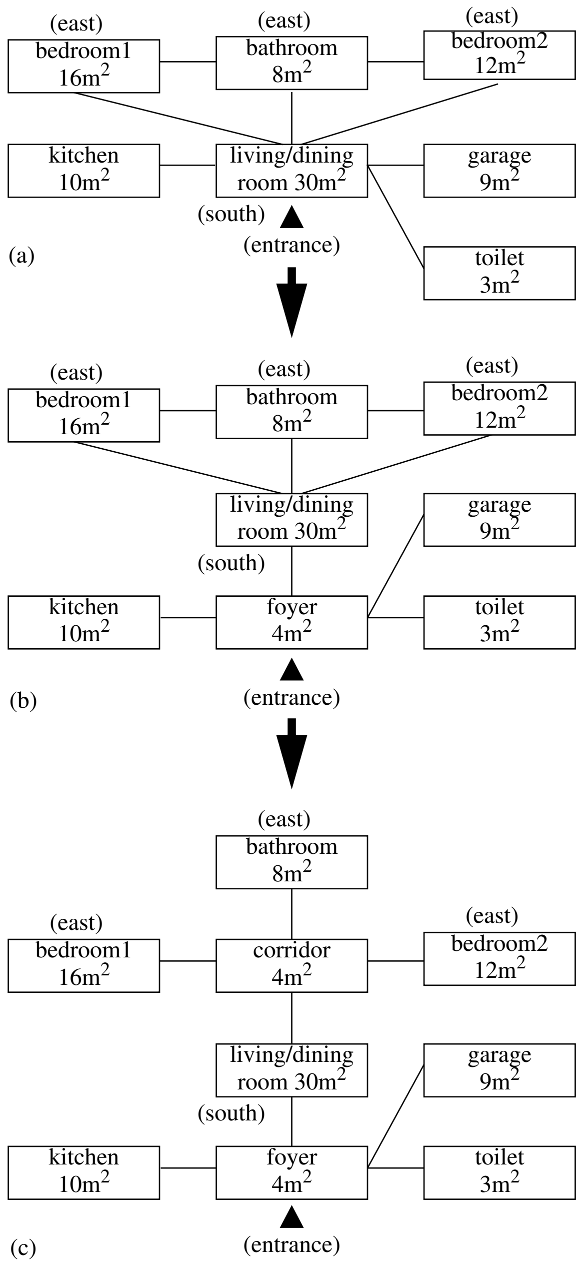

Figure 2.5: Improvement of the design diagram of a house

2.2.3. CONSTRUCTING AN ARCHITECTURAL SKETCH

In the last phase of the design process, our system uses the complete specification of the rooms generated during the previous phases to construct an architectural sketch of the house. As mentioned above, the geometrical aspects used to represent and produce this design sketch are characterized in terms of geometrical areas. The algorithms performing the actual geometrical design, i.e., partitioning the house into geometrical areas corresponding to the rooms (once their usage, functionality, and other requirements are fixed), are modeled as operations of the class 'area' by means of methods.

At the beginning of this phase, an area representing the geometrical aspects (size, dimension, etc.) of the whole house is generated, and all rooms derived in the previous phases are associated with this area. The methods performing the geometrical design start with this area, dividing it into two subareas and then choosing one of them to deal with in the next step. In each of the following steps, the methods divide a chosen area into two subareas by means of an breadth-first strategy, thereby generating an tree-like structure represented as an aggregation hierarchy. Each node of the tree therefore corresponds to a design decision since they express a possible partition of an area. After each division, the rooms associated with the area being split are distributed between the new subareas, and the geometrical aspects of the two resulting areas (the size, the dimension and the position) are calculated by the system. The algorithm performs this way until all areas contain exactly one room. At this point, the areas represent the leaves of the tree and express a sketch of the house.

The topological constraints (orientation and adjacencies) as well as the geometrical properties (i.e., the required size) of the rooms belonging to an area are used to decide how an area is to be partitioned. For example, rooms having the same orientation should be grouped together, and rooms that are connected to each other are either not separated in two different areas or kept in adjacent areas as long as possible. If there is no area that can be further divided into subareas, without causing the violation of some constraint, the algorithm uses backtracking to generate alternative decision nodes.

If the construction of the architectural sketch succeed, i.e., every leave of the tree represents exactly one room, the user is asked to accept the actual sketch. At this point, the user can then require an explanation of the design decisions, which is easily supplied since each node of the area tree represents a decision taken by the system. If the user rejects the solution generated by the system, alternatives can be derived in a non-redundant way by backtracking in the tree and exploiting previous design decisions. Different design alternatives are kept in the KB until the user makes his decision. The user can also add new (topological or even functional) requirements or delete existing ones, leading to a partial repetition of previous design phases and to the revision of the decisions of the system. For example, adding a new adjacency constraint can invalidate design decisions stated previously, causing an redesign of certain areas or the whole house. Deleting constraints is especially required if no geometrical sketch of the house can be generated by the system.

After the design session is finished, the resulting sketch is kept in the KB. A user may therefore start a new session with the result of a previous one as the basis for his new design.

3. CONCLUSIONS AND OUTLOOK

In this paper, we have given an overview of our conception of an intelligent CAD application in the area of architectural design and sketched its realization using the knowledge model of KRISYS, a KBMS prototype developed at our university.

The system incorporates several features that we regard as essential for intelligent CAD. First of all, it shows an active system behavior. At all design stages, the system provides the user with appropriate design hints, relevant problem solutions, and diagnostic information, that he may utilize to improve his design. The ability of the application to actively support the user in his design activities is especially required when the case of incomplete design specifications is considered. If the user gives only part of the information that is necessary to successfully terminate a design step, the system tries to complete such specifications, utilizing the design knowledge incorporated in its knowledge base. Therefore, the overall design process can be seen as a kind of iteration between user and system. However, the extent of activities performed by the system is still being determined by the user. He may decide whether to leave certain characteristics of the design object unspecified in order to let them be derived by the system, but also may modify system-derived information at all design stages.

Another major goal during the conception and realization of our architectural design system was to come as close as possible to the design process performed by human experts (architects). Therefore, we have divided the overall design process into several design phases. In the first phase, only the functional aspects of a house are considered in terms of usages or activities (e.g., leisure, eating, sleeping). In the second phase, the house is designed with respect to its topological properties (i.e., orientations and adjacencies of rooms). In the last phase, the actual geometrical representation is fixed (in terms of geometrical areas), yielding a design sketch of the house. In order to effectively support these different design phases, an explicit separation of functional, topological, and geometrical aspects emerged as an essential feature to increase flexibility (e.g., with respect to the association of usages and rooms) and abstraction (e.g., viewing a house in terms of usages without considering distinct rooms). Nevertheless, the different aspects of a house under design, which are relevant at different design phases, are intertwined with each other through complex dependencies, which actually determine the global consistency of the design. The conception of our architectural system is therefore based on the notion of an integrated product model, containing representations of all different aspects of a design object (i.e., functional, topological, and geometrical). The dependencies between these distinct representations are also made explicit in the product model and may consequently be maintained and exploited by the system in order to achieve a consistent and construction-oriented design.

In summary, what can be considered as one of the major issues of our intelligent CAD application is the improvement of the overall design process by incorporating different design aspects (geometrical, functional, topological) which are related to different processing or design phases but are all integrated into the same design object (product model). Therefore, significant parts of the semantics of the application world, which are in general buried in various programs responsible for different design phases, became explicitly represented in an integrated product model, where they are available for all system components.

When examining the above described system characteristics, it becomes evident that knowledge modeling aspects can be considered as a key issue to the support intelligent CAD systems [HM85, HM90]. The KBMS KRISYS [Ma91, Ma88b] provides a rich spectrum of modeling concepts, which has been extensively employed in the implementation of our architectural application. In this implementation, the abstraction concepts provided by KRISYS were primarily important for the support of a semantically enriched object description. Additionally, they were employed to define a means for object organization which, in turn, was useful to describe distinct application aspects. This was done, however, in an integrated way. In order to keep the different representations of a design object consistent among all its representations, procedural mechanisms for knowledge representation (i.e., demons, methods, etc.) were used to ensure a high degree of semantic integrity and to relieve the user (or application program) from this part of integrity checking. Moreover, the integration of behavior into the application model in the form of procedural attributes, user-defined functions, or methods made it possible to describe actions in which application objects are involved, thereby permitting the integration of application-oriented operations into the system. Finally, reasoning facilities were required as basic mechanisms to deal with incomplete specifications as well as to control the overall application process, thereby also providing the necessary active system behavior.

Although we have covered several interesting issues with our architectural prototype implementation, the system is in its current version far from satisfactory. Additional improvements will therefore be part of our future work. One of the main goals will be related with the modeling of constraints, which has turned out to be one of the central issues in our system. The representation of constraints has to become more flexible in order to support different kinds of dependencies (e.g., needs of the user, laws) with different properties (hard or soft design goals). A more explicit and compound representation has to be achieved, which should also allow queries on constraints (e.g., 'give all constraints relevant for kitchens'). Additionally, the constraints should play a more active role in determining and guiding the system behavior. Nevertheless, the user should still be able to determine a suitable degree of system activity, which may force the constraints to be considered only as (passive) integrity constraints.

Another goal will be the improvement of the second and third design phases. The design diagram representing the topology of a house should be represented in a more explicit fashion in order to better support additional operations for its modification and exten-

sion. The geometrical design, that is generated by the system and may be accepted or rejected possibly leading to alternative designs, should significantly be extended so that the user can directly position or modify certain rooms at will in a graphic interface. The system will then only check the consistency of the sketch provided by the user.

Other efforts which are related to the support of more complex designs will involve providing alternative design versions at all stages of the design process (e.g., alternative design diagrams based on the same usages of rooms, etc.), which are then maintained by the system. We will possibly extend our application to support the design of many-story houses, thereby also considering aspects of cooperative work in this context.

Acknowledgment

We would like to thank the architect Claudia Mattos, who greatly contributed to the conception and realization of the design application described in this paper.

REFERENCES

BMW84 Borgida, A., Mylopoulos, J., Wong, H.K.T.: Generalization/Specialization as a Basis for Software Specification, in: On Conceptual Modelling (Perspectives from Artificial Intelligence, Databases, and Programming Languages), Topics in Information Systems, (eds.: Brodie, M.L., Mylopoulos, J., Schmidt, J.W.), Springer-Verlag, New York, 1984, pp. 87-114.

DHMM89 Deßloch, S., Härder, T., Mattos, N., Mitschang, B.: KRISYS: KBMS Support for Better CAD Systems, in: Proc. 2nd International Conference on Data and Knowledge Systems for Manufacturing and Engineering, Gaithersburg - Maryland, Oct. 1989, pp.172-182.

DHMS90 Deßloch, S., Hübel, C., Mattos, N., Sutter, B.: KBMS Support for Technical Modeling in Engineering Systems, in: Proc. 3rd International Conference of Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Charleston - South Carolina, July 1990, pp. 790-799.

GAPR88 Grabowski, H., Anderl, R., Patzold, B., Rude, S.: The Development of Advanced Modeling Techniques - Meeting the Challenge of CAD/CAM-Integration, in: Proc. 4th CIM Europe Conf., May 1988.

HM85 Hartzband, D., Maryanski, F.: Enhancing Knowledge Representation in Engineering Databases, in: Computer, Vol. 18, No. 9, Sept. 1985, pp.39-48.

HM90 Härder, T., Mattos, N.M.: An Enhanced DBMS Architecture Supporting Intelligent CAD (invited lecture), in: Proc. of Int. Conf. TECHNO-DATA '90, Berlin, Dec. 1990.

Ma88a Mattos, N.M.: Abstraction Concepts: the Basis for Data and Knowledge Modeling, in: 7th Int. Conf. on Entity-Relationship Approach, Rom, Italy, Nov. 1988, pp. 331-350.

Ma88b Mattos, N.M.: KRISYS - A Multi-Layered Prototype KBMS Supporting Knowledge Independence, in: Proc. Int. Computer Science Conference - Artificial Intelligence: Theory and Application, Hong Kong, Dec. 1988, pp. 31-38.

Ma90 Mattos, N.M.: An Approach to Knowledge Base Management - requirements, knowledge representation, and design issues -, Lecture Notes in Artificial Intelligence (subseries of Lecture Notes in Computer Science), Springer, to be published.

Ph88 Pham, D.T.: Expert Systems in Engineering, Springer-Verlag, 1988.

RHMD87 Rosenthal, A., Heiler, S., Manola, F., Dayal, U.: Query Facilities for Part Hierarchies: Graph Traversal, Spatial Data, and Knowledge-Based Detail Supression, Research Report, CCA, Cambridge, MA, 1987.

SGL89 Spur, G., Germer, H.-J., Lehmann, C.: Impact of Geometric Modeling for Computer Integrated Manufacturing, in: International Symposium on Advanced Geometric Modeling for Engineering Applications (IFIP & GI), Berlin, FRG, 1989.

SR86 Sriram, D., Rychener, M.: Expert Systems for Engineering Applications, special issue of IEEE Software, Vol. 3, No. 2, March 1986.

in: Proc. IEA/AIE '91 - 4th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Kauai, Hawaii, June 1991, S. 409-418

A Knowledge-based Approach to Intelligent CAD for Architectural Design

Mattos, N.M., Deßloch, S., Leick, F.-J.

University of Kaiserslautern, Department of Computer Science

P.O. Box 3049, D-6750 Kaiserslautern, Germany

e-mail: mattos@informatik.uni-kl.de

ABSTRACT

In this paper, we present the conception of an intelligent CAD application for architectural design and describe its realization using the knowledge base management system KRISYS. Our application integrates all relevant information regarding the design of one-story houses into a product model. It divides the design process in three main phases, in which the product model is used as basis for the execution of the functional, topological, and geometrical design. In the paper, we give an overview of this product model and describe the system activities that are carried out in each of the design phases.

1. INTRODUCTION

Over the last few years, computer support in product planning, design, and construction has gained more and more importance within the computer science community. A lot of research activity has been focusing on the development of intelligent CAD systems aimed at a continual, efficient, and integrated support of the design process as well as of its overall organization [GAPR88,SGL89].

In intelligent CAD systems, the designer is not the only active unit within the overall design process (as is the case in conventional CAD systems). The CAD system can also exhibit an active behavior, providing a more intelligent interface to the user. It supports system-enforced checking of complex integrity constraints even across design steps to guarantee a consistent design. It can offer appropriate design hints, relevant problem solutions, refined simulation results, and adequate diagnostic information at all stages of the design process [Ph88,SR86]. In some sense, there is a kind of 'partnership' between designer and system, where it is possible to switch between automated design guided by the system (e.g., in standard cases) and human design controlled by the designer's decisions (e.g., in special cases).

One main goal of intelligent CAD is therefore the improvement of the design process by incorporating not only geometrical, but all relevant design aspects as well as functional dependencies at each design step. As a consequence, the design process is done iteratively, but not by means of stand-alone tools converting the output data structures of previous design steps into internal data structures and vice-versa before the tools can accomplish their functions. Intelligent CAD systems, in contrast to conventional CAD systems, divide the design process into design phases which subsequently supplement incomplete object specifications, stepwise enrich object descriptions with functional, topological, and geometrical aspects, and support improvements or refinements by means of a feedback to each previous phase. Since all design phases refer to the same design objects, all aspects of these objects are represented in a unique and non-redundant manner (in the so-called product model), allowing for a consistent and uniform object handling.

Considering the above characteristics of intelligent CAD systems, we have constructed a prototype application for architectural design. It integrates of all relevant design information in a product model which is then used as basis for the execution of the functional, topological, and geometrical design of one-story houses. Corre-

in: Proc. of the 4th Int. Conf. on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, Koloa, Kanai, Hawaii, 1991.

sponding to these three different design specifications, the design process is divided into three main phases, thereby reflecting the methodology of human architects.

In this paper, we describe our intelligent CAD application for architectural design, sketching its conception and giving hints as to the approaches exploited to accomplish the intelligent CAD concepts mentioned above. Our application has been implemented on the basis of a knowledge base management system (KBMS) called KRISYS [Ma91, Ma88b], that integrates artificial intelligence (AI) and database system (DBS) techniques in an effective way [DHMS90, DHMM89]. Due to the space limitations of this paper, we cannot present such an implementation of architectural design in detail. For this reason, we will concentrate on the representation of an enriched design description and on how to achieve the desired active system behavior.

2. AN OVERVIEW OF THE APPLICATION

The goal of our intelligent CAD system is to support the architectural design of one-story houses for families, based on requirements and needs specified by the user. All these specifications as well as all relevant information concerning the design are explicitly represented in an integrated product model internally managed by our system. This product model contains several submodels describing geometrical, topological, and functional aspects of the objects under design, thereby providing distinct views of our representation. The definition of the product model was accomplished appropriately by exploiting the object-centered representation framework of KRISYS.

KRISYS allows a natural description of design objects, integrating all its aspects into one KB object. Abstraction concepts (classification, generalization, association, and aggregation) [Ma88a, BMW84] were used as the basic mechanisms for describing the organizational semantics of the application domain. Built-in, abstraction-related, reasoning facilities (inheritance, set-properties, membership stipulations, and implied predicates) [Ma88a, RHMD87] were exploited to continuously guarantee the structural and semantic integrity of the product model. Distinct aspects of the design object were represented using different abstraction concepts or distinct hierarchies of the same concept. One hierarchy was employed for representing geometrical, another for functional, and a last one for topological information about the object. This has the advantage that different aspects of a design are easily distinguishable in the model. Following this approach, each room of a house is described by an object, which is either a component of one of these hierarchies, or contains properties relating it to them. Therefore, a room can be viewed from a functional, a topological, and a geometrical point of view and acts as a link between the different hierarchies, thereby guaranteeing the overall consistency of the design.

In the following, we will first explain how the distinct aspects of a design object are represented by means of abstraction hierarchies and integrated into the room descriptions. All these hierarchies are manipulated during the design process, dynamically generating objects for the representation of important design information and incrementally completing the specification of such objects in order to construct an architectural sketch of the house. These manipulations, which are performed in different phases of the design process, are described in more detail in the subsequent sections of this chapter.

2.1. A GENERAL OVERVIEW OF THE REPRESNTATION

As already mentioned, the house under design can be seen from a functional, a topological, as well as a geometrical point of view.

2.1.1. FUNCTIONAL VIEW

The functional aspects are expressed in terms of usages, which can be associated with the rooms of a house. We have introduced a generalization hierarchy into our system, which contains the information about usages relevant for the design process (see figure 2.1 for a partial view of this hierarchy). There can be different levels of abstraction when considering the usages. For example, both 'reading' and 'watching-tv' may be seen as the more abstract usage 'leisure'. These different levels of abstraction are also reflected by the levels of detail in the requirements for the house specified by the designer (see chapter 2.2 for details).

The usage hierarchy contains information about all the usages the system knows about. This information is exploited for a specific design by creating instances of the usages which are required for the house (either by specification of the designer or by default assumptions). In our example (see figure 2.1), usage instances are created

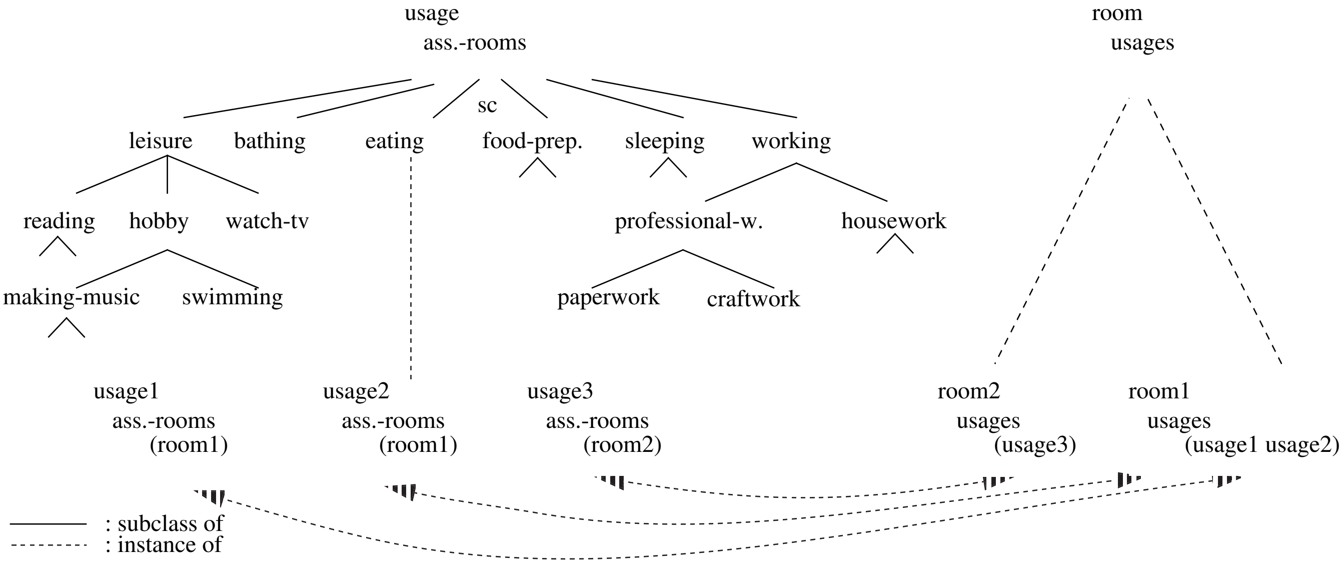

Figure 2.1: Representation of functional aspects

among others for 'food-preparation', 'eating', and 'leisure'. During the design process, rooms are created (as instances of the class 'room') and associated with the specific usages in order to appropriately fulfill the functional requirements for the house. The attribute 'usages' in the class 'room' and 'ass.rooms' (associated-rooms) in the class 'usage' (as the 'inverse' attribute), which are inherited respectively by the rooms and usages, are used to represent this relationship. In figure 2.1, for example, 'room 1' is associated with the instances of 'leisure' and 'eating', while 'room 2' is supposed to be used for 'food-preparation'.

2.1.2. TOPOLOGICAL VIEW

The topological aspects of a house under design are characterized by <u>orientations</u> (e.g., a room is located at the south side of the house) and by <u>adjacencies</u> in the sense that two rooms, for example a bedroom and a bathroom, are located next to each other to achieve a direct connection between them (e.g., through a door). They are represented in a separate hierarchy with the class 'topology' as its root and the two subclasses 'orientation' and 'adjacency' (see figure 2.2). The instances of these two subclasses describe the specific topological constraints known by the system, which are established either through legal restrictions or default-assumptions. Please note that orientations as well as adjacencies are primarily specified according to usages, and not to rooms. For example, the orientation 'sleep-east' is associated with the usage 'sleeping', and 'bathing-sleeping' relates the usages 'bathing' and 'sleeping', which is specified in the 'usages'-attribute of the objects. This is important, since a single room may be considered with respect to several usages (e.g., room 1 in figure 2.1) and is then affected by several topological restrictions. The topology hierarchy itself is not modified or extended (e.g., by creating new instances) during the design session. It is merely used to propagate the topological restrictions to the

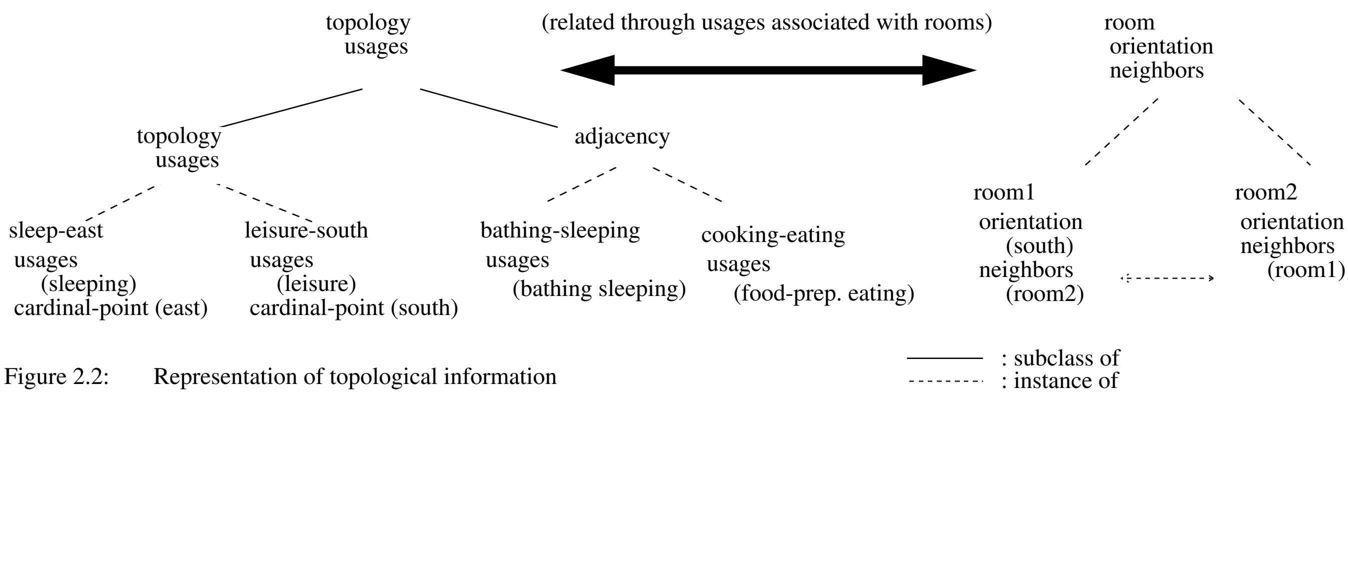

Figure 2.2: Representation of topological information

rooms created by the system according to the specific usages associated with them. Rooms are therefore described by two topological attributes: 'orientation', which contains the cardinal points of the rooms according to the orientations of usages, and 'neighbors', relating adjacent rooms.

2.1.3. GEOMETRICAL VIEW

Besides functional requirements and topological information, the geometrical aspects are fundamental for actually representing and producing an architectural sketch of the house under design. These are characterized in terms of geometrical areas. A house consists, from this point of view, of several areas which may (recursively) be split into subareas that finally are identical to specific rooms. We have represented this tree-like structure having rooms of the house as its leaves with the help of the aggregation concept: an area is considered as an aggregate consisting of component areas (i.e., subareas). This aggregation hierarchy is shown in figure 2.3. All areas are instances of the class 'area' (e.g., 'area 1'), thereby inheriting geometrical attributes such as 'position', 'dimension' (i.e., lengths of the sides), and 'size'. They can be components of other areas (e.g., 'area 1' is a component of 'house') and in turn have other areas as components (e.g., 'area 2' and 'area 3'). Since rooms are considered as areas themselves, the class 'room' is a subclass of 'area', but with the restriction that rooms (e.g., 'room 1' and 'room 2') cannot have other areas as components (i.e., they are 'atomic' components).

2.1.4. INTEGRATED VIEW

As already mentioned, rooms play an important role in the integration of the different hierarchies, acting as a link between them. This is illustrated in figure 2.4, where the full specification of the room 'room 1' is shown. From a functional point of view, 'room 1' is associated with the usages 'usage 1' and 'usage 2', which are respectively instances of 'leisure' and 'eating'. Its topological role is described by its orientation ('south') and its neighbors ('room 2'). Geometrical aspects are characterized by its attributes 'position', 'dimension' and 'size', and by the fact that it is a component of 'area2'. An integrated view of this information is especially relevant for the maintenance of the overall consistency of the different hierarchies. For example, the size of a room is closely related to the usages intended for it. This information has to be exploited by the process designing the geometrical areas of the house.

2.2. THE PROCESSING PHASES OF OUR DESIGN SYSTEM

In the following sections, we will take a closer look at the different design phases supported by the system and show how they utilize the information represented in the above hierarchies. Such hierarchies not only provide three different views of an architectural design, but also reflect the three different phases carried out by human architects during a design process.

2.2.1. GENERATING A FUNCTIONAL VIEW OF THE HOUSE

Specifying usages

In the first phase of the design process, our system questions the user as to his/her requirements. These may be formulated on a high semantic level (e.g., 'I need to work at home and preferably not on the side of the street.'). Furthermore, the specification of such requirements may be incomplete in the sense that they are not suffi-

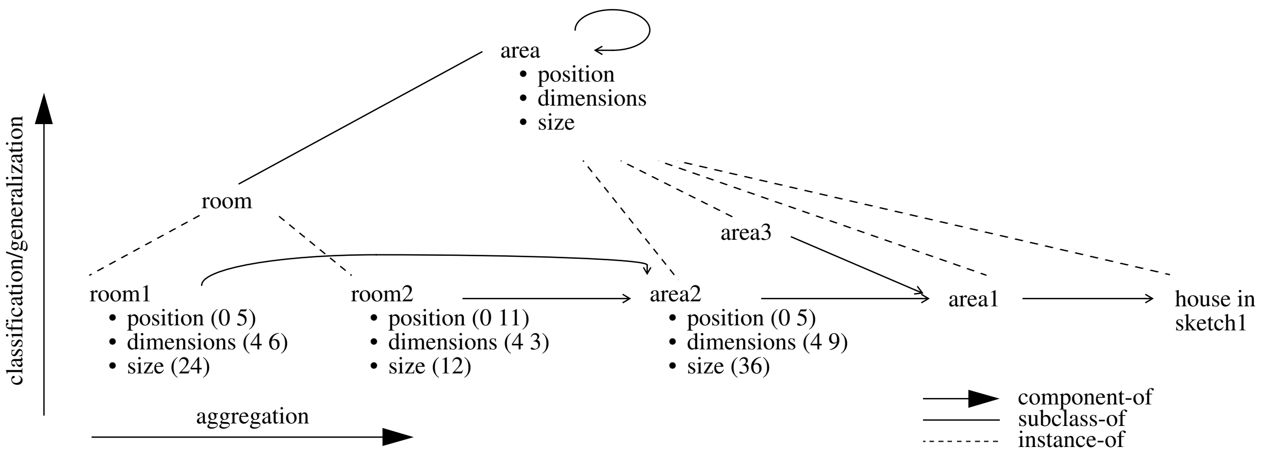

Figure 2.3: Representing the geometry of an architectural sketch

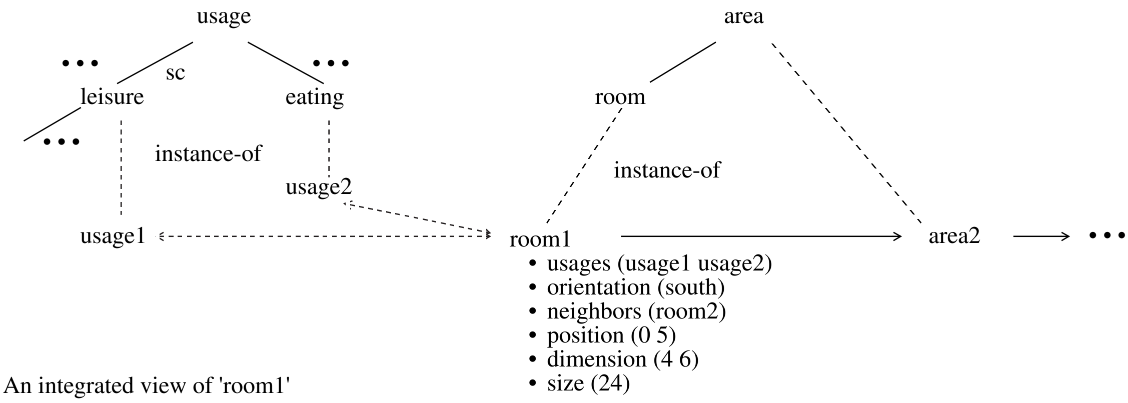

Figure 2.4: An integrated view of 'room1'

cient to directly achieve a final design stage. For this reason, the system utilizes its knowledge (expressed by rules representing standard requirements, laws, etc.) to supplement the user-sketched blueprint. In other words, a significant part of the knowledge of our architectural design system is represented as rules, which are responsible for the tasks of the system, such as the derivation of activities or usages that are standard in every house or required by law (e.g., eating, sleeping, heating...) and are not necessarily specified by the user.

There are several levels of abstraction in which activities can be derived by the system. They correspond to the levels of the generalization hierarchy defining the usages (figure 2.1) and reflect different degrees of detailing of the functional aspects of a house. Actually, every level expressed in this hierarchy represents a valid degree of detailing of a functional design. This means that our system, just like human architects, can specify functional requirements at any level, depending on the degree of detailing it 'believes' to be necessary in each particular case. That is, it makes use of information about the 'context'[1] in which a particular house is being constructed in order to dynamically determine whether it is necessary to further detail one or another functional aspect of the house. For example, if the members of a family play different sports or have several hobbies, a detailed analysis of the partial hierarchy under 'leisure' (see figure 2.1) is required, whereas a house in which the parents wish to work needs a detailed investigation of activities under 'working'. So, what makes the system move further into a more detailed level is its 'ability' to consider information about the 'context' and thereby to realize the necessity to further detail some functional aspects.

Every time the system enters a more detailed level, the user has the chance to express his/her requirements that correspond to this level and the system itself derives additional activities by default assumptions. For example, when moving down from 'leisure' into the next lower level, the system can assign the activity 'watch-tv' even if the user does not give any other information about his/her activities regarding 'leisure'. Whenever an activity is defined either by the user or the system, an instance of the corresponding 'usage' class is created by the system in order to represent this functional requirement.

The result of this design step is therefore a set of requirements expressed in terms of activities to be supported by the house being constructed, that is, a functional description of the house represented in terms of the instances created in all levels of the 'usage' hierarchy. Note that in this first phase, our system does not deduce any kind of rooms (such as kitchen, bedroom, etc.) that the house has to have. It first abstracts from particular rooms, analyzing a house only from a functional point of view in the same way as human architects. This is particular important because several combinations between activities and rooms are possible when specifying rooms in the following step of this design phase. For instance, the activity 'eating' can be 'associated' with a kitchen, a large living room, a dining room, etc. Since activities directly influence the characteristics of a room (e.g., the size of a room strongly depends on the activities and the number of people performing such activities in the room), they are viewed as a group of constraints that should be satisfied after the rooms of the house have been specified.

1. This involves general information about the user and his/her family, such as job, sex, age, habits, etc.

Deriving rooms

In the next step, our design system specifies rooms that can satisfy the assigned functional constraints. These rooms have, at this step, no geometrical or topological information associated with them, but already represent the actual rooms that have to be considered when generating the real sketch of the house later on. This generation process is carried out in a stepwise fashion in which every instance of usage that was created in the first step is examined and assigned to either an existing room or to a room just created for this purpose. Since the associations between usages and rooms can be defined either by the system or the user, this process is guided by integrity constraints that ensure that both the user specifications and deductions of our system will generate room descriptions reflecting a semantically correct state.

Our system exploits the integrity constraints to control the overall process of this second step, checking the actions of the user and triggering systems operations to complete user specifications by enforcing the semantic integrity of the KB. When the user directly specifies an activity for a particular room, our system exploits such constraints to check whether this new activity is compatible with a previously asserted usage. There are statutes that strictly determine incompatible usages. The architect has, for example, to consider that facilities for both cooking and bathing should not be provided in the same room. For this reason, we have attributes expressing 'contradictory activities' in the objects describing activities, which contain all incompatible activities for every activity in the KB. Procedures represented as demons attached to the attribute expressing the usage of a room are immediately activated when a usage is added in order to check if one of the old usages is a 'contradictory activity' of the new one. Upon detecting an inconsistency, an error is reported, automatically eliminating the previous change of the KB.

Another approach applied by the system when integrity constraints are violated is to trigger additional operations to transform an inconsistent KB state into a consistent one, instead of rejecting the changes. This triggering mechanism is automatically carried out by our design system, whenever, for example, information is incompletely specified by the user. The mechanism is also desirable within our integrated product model, where changes concerning one representation of an object (e.g., its functional aspects) should automatically cause changes in the other representations (e.g., the geometry). Such an approach is realized by either demons to trigger the appropriate changes or by rules to dynamically derive information and insert it into the room description. As already mentioned, the user of our design system may specify activities or usage for the house, but need not to relate them to any rooms. This specification is then completed by the system in this second step. For example, when the user specifies the activity 'professional-working', the system activates rules associated with the 'professional-working' object in the KB. These rules either generate a new room for the activity (e.g., the office room) or establish a relationship between the activity and an already existing room (e.g., the living-room), considering, for example, extra places for a work desk. Note that the modification of the size of the living-room because of the association of the activity is an example of the use of rules to keep track of the dependencies between functional and geometrical aspects of the product model. This kind of dependency is exploited by our system at the next phase of the design process to complete the descriptions of the rooms.

2.2.2. DETERMINING ROOM CHARACTERISTICS

Defining orientations and adjacencies

The goal of the second design phase is to completely specify the topological aspects of the house and the geometrical aspects that directly depend on the functional and topological ones. Again, these aspects are either determined by the user (e.g., 'I preferably do not want to work on the side of the street') or derived by the system. Both kinds of definitions are restricted by the constraints and dependencies existing in this application world which are described in our product model as constraints involving various levels of granularity of knowledge: attributes of objects (e.g., the size of a main sleeping-room may not be smaller than 12 square meters), objects (e.g., the functional view of an object, such as the activities associated with it, must be in accordance with its geometrical and topological view, i.e., its size and position within the house), and groups of objects (e.g., connections between rooms). Note that most of these constraints are derived from laws and statutes that rule minimum areas for rooms, the amount of direct illumination in square meters that a room must have and consequently its position within the house (inner or outer), the kind of ventilation (direct or indirect)

that a room should have and therefore its location and connections to other rooms.

System derivations are carried out by exploiting the topology hierarchy (figure 2.2), in which orientations and adjacencies of 'usages' are defined, in order to propagate this information respectively to the attributes defining orientation and neighbors of the rooms. Geometrical characteristics, such as the size of the rooms, are also derived on the basis of the usages that have been assigned to a particular room and of laws or statutes as previously mentioned.

Improving the design diagram

At the end of this step, our system has a complete description of the rooms, containing the location of a room within the house (e.g., bathroom and bedroom should be at the east side of the house), the connections between rooms (e.g., the kitchen should be connected directly to the entrance, but not to the bedroom), sizes of rooms, etc. This description is then used by the system to produce the so-called design diagram, which is usually employed by human architects to analyze and ameliorate complex designs before a concrete sketch is generated. This diagram is actually implicitly represented in our product model, given by the rooms, their topology, geometry, and usages. It is a graph representation of the object being designed (in our case a house) in which nodes and edges symbolize respectively rooms and connections between rooms (see figure 2.5).

Improvements are then performed by transforming the graph into another one, presenting some important characteristics. First, such a graph has to be planar since connections in a house cannot cross each other. This can be achieved by either removing some connections or creating additional rooms, such as a corridor. Secondly, the graph cannot have regular rooms with an excessive number of junctions to other rooms, as for example, the 'living/dining' room illustrated in figure 2.5a. To reduce the number of such junctions between regular rooms, special (connection) rooms (such as corridors, foyers, entrance halls, etc.) can be introduced for this purpose, as shown in figure 2.5b. Note that the transformation of the graph has to be carried out in several steps since the introduction of a connection room (the foyer in figure 2.5b) does not necessarily reduce the number of junctions between regular rooms to an acceptable minimum. In the example, a corridor was then created to reduce the connections of the 'living/dining' room and of the bathroom (figure 2.5.c). Finally, further transformations may be performed to eliminate inadequate connections (e.g., the connection between the two bedrooms in figure 2.5b is through the bathroom), to reduce the communication between groups of rooms (e.g., rooms in a private area such as bedrooms and bathrooms should not be connected directly to rooms in the social area), to split very large rooms (thereby reorganizing the connections), etc.

Certainly, the system considers the constraints mentioned above at each transformation of the design diagram in order to produce only diagrams satisfying them.

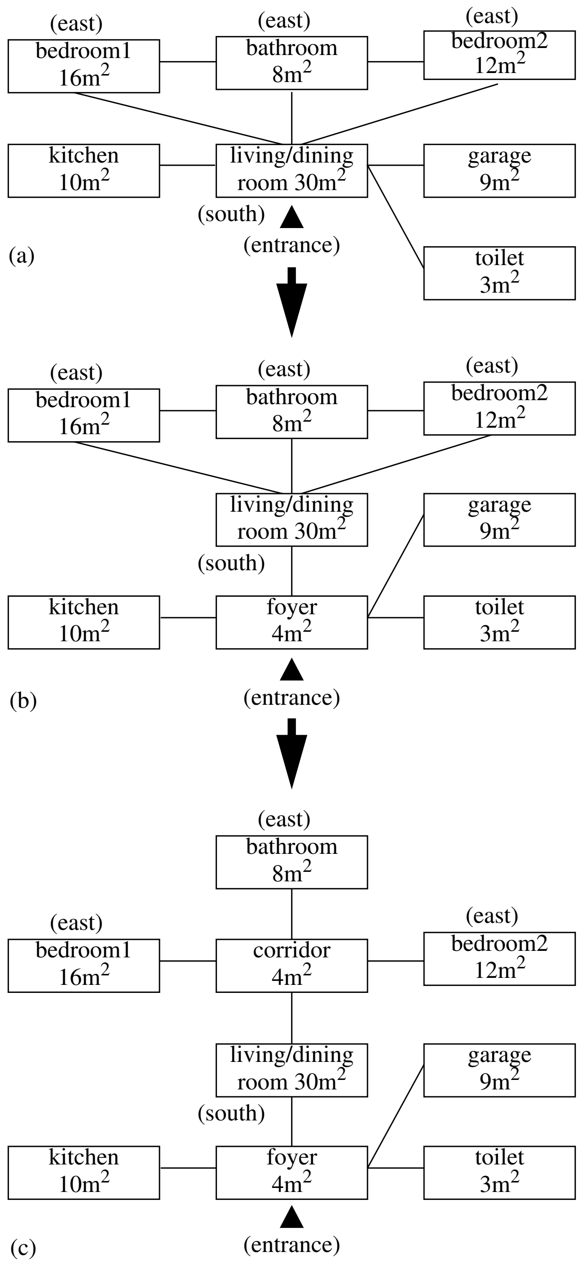

Figure 2.5: Improvement of the design diagram of a house

2.2.3. CONSTRUCTING AN ARCHITECTURAL SKETCH

In the last phase of the design process, our system uses the complete specification of the rooms generated during the previous phases to construct an architectural sketch of the house. As mentioned above, the geometrical aspects used to represent and produce this design sketch are characterized in terms of geometrical areas. The algorithms performing the actual geometrical design, i.e., partitioning the house into geometrical areas corresponding to the rooms (once their usage, functionality, and other requirements are fixed), are modeled as operations of the class 'area' by means of methods.

At the beginning of this phase, an area representing the geometrical aspects (size, dimension, etc.) of the whole house is generated, and all rooms derived in the previous phases are associated with this area. The methods performing the geometrical design start with this area, dividing it into two subareas and then choosing one of them to deal with in the next step. In each of the following steps, the methods divide a chosen area into two subareas by means of an breadth-first strategy, thereby generating an tree-like structure represented as an aggregation hierarchy. Each node of the tree therefore corresponds to a design decision since they express a possible partition of an area. After each division, the rooms associated with the area being split are distributed between the new subareas, and the geometrical aspects of the two resulting areas (the size, the dimension and the position) are calculated by the system. The algorithm performs this way until all areas contain exactly one room. At this point, the areas represent the leaves of the tree and express a sketch of the house.

The topological constraints (orientation and adjacencies) as well as the geometrical properties (i.e., the required size) of the rooms belonging to an area are used to decide how an area is to be partitioned. For example, rooms having the same orientation should be grouped together, and rooms that are connected to each other are either not separated in two different areas or kept in adjacent areas as long as possible. If there is no area that can be further divided into subareas, without causing the violation of some constraint, the algorithm uses backtracking to generate alternative decision nodes.

If the construction of the architectural sketch succeed, i.e., every leave of the tree represents exactly one room, the user is asked to accept the actual sketch. At this point, the user can then require an explanation of the design decisions, which is easily supplied since each node of the area tree represents a decision taken by the system. If the user rejects the solution generated by the system, alternatives can be derived in a non-redundant way by backtracking in the tree and exploiting previous design decisions. Different design alternatives are kept in the KB until the user makes his decision. The user can also add new (topological or even functional) requirements or delete existing ones, leading to a partial repetition of previous design phases and to the revision of the decisions of the system. For example, adding a new adjacency constraint can invalidate design decisions stated previously, causing an redesign of certain areas or the whole house. Deleting constraints is especially required if no geometrical sketch of the house can be generated by the system.

After the design session is finished, the resulting sketch is kept in the KB. A user may therefore start a new session with the result of a previous one as the basis for his new design.

3. CONCLUSIONS AND OUTLOOK

In this paper, we have given an overview of our conception of an intelligent CAD application in the area of architectural design and sketched its realization using the knowledge model of KRISYS, a KBMS prototype developed at our university.

The system incorporates several features that we regard as essential for intelligent CAD. First of all, it shows an active system behavior. At all design stages, the system provides the user with appropriate design hints, relevant problem solutions, and diagnostic information, that he may utilize to improve his design. The ability of the application to actively support the user in his design activities is especially required when the case of incomplete design specifications is considered. If the user gives only part of the information that is necessary to successfully terminate a design step, the system tries to complete such specifications, utilizing the design knowledge incorporated in its knowledge base. Therefore, the overall design process can be seen as a kind of iteration between user and system. However, the extent of activities performed by the system is still being determined by the user. He may decide whether to leave certain characteristics of the design object unspecified in order to let them be derived by the system, but also may modify system-derived information at all design stages.

Another major goal during the conception and realization of our architectural design system was to come as close as possible to the design process performed by human experts (architects). Therefore, we have divided the overall design process into several design phases. In the first phase, only the functional aspects of a house are considered in terms of usages or activities (e.g., leisure, eating, sleeping). In the second phase, the house is designed with respect to its topological properties (i.e., orientations and adjacencies of rooms). In the last phase, the actual geometrical representation is fixed (in terms of geometrical areas), yielding a design sketch of the house. In order to effectively support these different design phases, an explicit separation of functional, topological, and geometrical aspects emerged as an essential feature to increase flexibility (e.g., with respect to the association of usages and rooms) and abstraction (e.g., viewing a house in terms of usages without considering distinct rooms). Nevertheless, the different aspects of a house under design, which are relevant at different design phases, are intertwined with each other through complex dependencies, which actually determine the global consistency of the design. The conception of our architectural system is therefore based on the notion of an integrated product model, containing representations of all different aspects of a design object (i.e., functional, topological, and geometrical). The dependencies between these distinct representations are also made explicit in the product model and may consequently be maintained and exploited by the system in order to achieve a consistent and construction-oriented design.

In summary, what can be considered as one of the major issues of our intelligent CAD application is the improvement of the overall design process by incorporating different design aspects (geometrical, functional, topological) which are related to different processing or design phases but are all integrated into the same design object (product model). Therefore, significant parts of the semantics of the application world, which are in general buried in various programs responsible for different design phases, became explicitly represented in an integrated product model, where they are available for all system components.

When examining the above described system characteristics, it becomes evident that knowledge modeling aspects can be considered as a key issue to the support intelligent CAD systems [HM85, HM90]. The KBMS KRISYS [Ma91, Ma88b] provides a rich spectrum of modeling concepts, which has been extensively employed in the implementation of our architectural application. In this implementation, the abstraction concepts provided by KRISYS were primarily important for the support of a semantically enriched object description. Additionally, they were employed to define a means for object organization which, in turn, was useful to describe distinct application aspects. This was done, however, in an integrated way. In order to keep the different representations of a design object consistent among all its representations, procedural mechanisms for knowledge representation (i.e., demons, methods, etc.) were used to ensure a high degree of semantic integrity and to relieve the user (or application program) from this part of integrity checking. Moreover, the integration of behavior into the application model in the form of procedural attributes, user-defined functions, or methods made it possible to describe actions in which application objects are involved, thereby permitting the integration of application-oriented operations into the system. Finally, reasoning facilities were required as basic mechanisms to deal with incomplete specifications as well as to control the overall application process, thereby also providing the necessary active system behavior.

Although we have covered several interesting issues with our architectural prototype implementation, the system is in its current version far from satisfactory. Additional improvements will therefore be part of our future work. One of the main goals will be related with the modeling of constraints, which has turned out to be one of the central issues in our system. The representation of constraints has to become more flexible in order to support different kinds of dependencies (e.g., needs of the user, laws) with different properties (hard or soft design goals). A more explicit and compound representation has to be achieved, which should also allow queries on constraints (e.g., 'give all constraints relevant for kitchens'). Additionally, the constraints should play a more active role in determining and guiding the system behavior. Nevertheless, the user should still be able to determine a suitable degree of system activity, which may force the constraints to be considered only as (passive) integrity constraints.

Another goal will be the improvement of the second and third design phases. The design diagram representing the topology of a house should be represented in a more explicit fashion in order to better support additional operations for its modification and exten-

sion. The geometrical design, that is generated by the system and may be accepted or rejected possibly leading to alternative designs, should significantly be extended so that the user can directly position or modify certain rooms at will in a graphic interface. The system will then only check the consistency of the sketch provided by the user.

Other efforts which are related to the support of more complex designs will involve providing alternative design versions at all stages of the design process (e.g., alternative design diagrams based on the same usages of rooms, etc.), which are then maintained by the system. We will possibly extend our application to support the design of many-story houses, thereby also considering aspects of cooperative work in this context.

Acknowledgment

We would like to thank the architect Claudia Mattos, who greatly contributed to the conception and realization of the design application described in this paper.

REFERENCES

BMW84 Borgida, A., Mylopoulos, J., Wong, H.K.T.: Generalization/Specialization as a Basis for Software Specification, in: On Conceptual Modelling (Perspectives from Artificial Intelligence, Databases, and Programming Languages), Topics in Information Systems, (eds.: Brodie, M.L., Mylopoulos, J., Schmidt, J.W.), Springer-Verlag, New York, 1984, pp. 87-114.

DHMM89 Deßloch, S., Härder, T., Mattos, N., Mitschang, B.: KRISYS: KBMS Support for Better CAD Systems, in: Proc. 2nd International Conference on Data and Knowledge Systems for Manufacturing and Engineering, Gaithersburg - Maryland, Oct. 1989, pp.172-182.

DHMS90 Deßloch, S., Hübel, C., Mattos, N., Sutter, B.: KBMS Support for Technical Modeling in Engineering Systems, in: Proc. 3rd International Conference of Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Charleston - South Carolina, July 1990, pp. 790-799.

GAPR88 Grabowski, H., Anderl, R., Patzold, B., Rude, S.: The Development of Advanced Modeling Techniques - Meeting the Challenge of CAD/CAM-Integration, in: Proc. 4th CIM Europe Conf., May 1988.

HM85 Hartzband, D., Maryanski, F.: Enhancing Knowledge Representation in Engineering Databases, in: Computer, Vol. 18, No. 9, Sept. 1985, pp.39-48.

HM90 Härder, T., Mattos, N.M.: An Enhanced DBMS Architecture Supporting Intelligent CAD (invited lecture), in: Proc. of Int. Conf. TECHNO-DATA '90, Berlin, Dec. 1990.

Ma88a Mattos, N.M.: Abstraction Concepts: the Basis for Data and Knowledge Modeling, in: 7th Int. Conf. on Entity-Relationship Approach, Rom, Italy, Nov. 1988, pp. 331-350.

Ma88b Mattos, N.M.: KRISYS - A Multi-Layered Prototype KBMS Supporting Knowledge Independence, in: Proc. Int. Computer Science Conference - Artificial Intelligence: Theory and Application, Hong Kong, Dec. 1988, pp. 31-38.

Ma90 Mattos, N.M.: An Approach to Knowledge Base Management - requirements, knowledge representation, and design issues -, Lecture Notes in Artificial Intelligence (subseries of Lecture Notes in Computer Science), Springer, to be published.

Ph88 Pham, D.T.: Expert Systems in Engineering, Springer-Verlag, 1988.

RHMD87 Rosenthal, A., Heiler, S., Manola, F., Dayal, U.: Query Facilities for Part Hierarchies: Graph Traversal, Spatial Data, and Knowledge-Based Detail Supression, Research Report, CCA, Cambridge, MA, 1987.

SGL89 Spur, G., Germer, H.-J., Lehmann, C.: Impact of Geometric Modeling for Computer Integrated Manufacturing, in: International Symposium on Advanced Geometric Modeling for Engineering Applications (IFIP & GI), Berlin, FRG, 1989.

SR86 Sriram, D., Rychener, M.: Expert Systems for Engineering Applications, special issue of IEEE Software, Vol. 3, No. 2, March 1986.

in: Proc. IEA/AIE '91 - 4th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Kauai, Hawaii, June 1991, S. 409-418

A Knowledge-based Approach to Intelligent CAD for Architectural Design

Mattos, N.M., Deßloch, S., Leick, F.-J.

University of Kaiserslautern, Department of Computer Science

P.O. Box 3049, D-6750 Kaiserslautern, Germany

e-mail: mattos@informatik.uni-kl.de

ABSTRACT

In this paper, we present the conception of an intelligent CAD application for architectural design and describe its realization using the knowledge base management system KRISYS. Our application integrates all relevant information regarding the design of one-story houses into a product model. It divides the design process in three main phases, in which the product model is used as basis for the execution of the functional, topological, and geometrical design. In the paper, we give an overview of this product model and describe the system activities that are carried out in each of the design phases.

1. INTRODUCTION

Over the last few years, computer support in product planning, design, and construction has gained more and more importance within the computer science community. A lot of research activity has been focusing on the development of intelligent CAD systems aimed at a continual, efficient, and integrated support of the design process as well as of its overall organization [GAPR88,SGL89].

In intelligent CAD systems, the designer is not the only active unit within the overall design process (as is the case in conventional CAD systems). The CAD system can also exhibit an active behavior, providing a more intelligent interface to the user. It supports system-enforced checking of complex integrity constraints even across design steps to guarantee a consistent design. It can offer appropriate design hints, relevant problem solutions, refined simulation results, and adequate diagnostic information at all stages of the design process [Ph88,SR86]. In some sense, there is a kind of 'partnership' between designer and system, where it is possible to switch between automated design guided by the system (e.g., in standard cases) and human design controlled by the designer's decisions (e.g., in special cases).

One main goal of intelligent CAD is therefore the improvement of the design process by incorporating not only geometrical, but all relevant design aspects as well as functional dependencies at each design step. As a consequence, the design process is done iteratively, but not by means of stand-alone tools converting the output data structures of previous design steps into internal data structures and vice-versa before the tools can accomplish their functions. Intelligent CAD systems, in contrast to conventional CAD systems, divide the design process into design phases which subsequently supplement incomplete object specifications, stepwise enrich object descriptions with functional, topological, and geometrical aspects, and support improvements or refinements by means of a feedback to each previous phase. Since all design phases refer to the same design objects, all aspects of these objects are represented in a unique and non-redundant manner (in the so-called product model), allowing for a consistent and uniform object handling.

Considering the above characteristics of intelligent CAD systems, we have constructed a prototype application for architectural design. It integrates of all relevant design information in a product model which is then used as basis for the execution of the functional, topological, and geometrical design of one-story houses. Corre-

in: Proc. of the 4th Int. Conf. on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, Koloa, Kanai, Hawaii, 1991.

sponding to these three different design specifications, the design process is divided into three main phases, thereby reflecting the methodology of human architects.

In this paper, we describe our intelligent CAD application for architectural design, sketching its conception and giving hints as to the approaches exploited to accomplish the intelligent CAD concepts mentioned above. Our application has been implemented on the basis of a knowledge base management system (KBMS) called KRISYS [Ma91, Ma88b], that integrates artificial intelligence (AI) and database system (DBS) techniques in an effective way [DHMS90, DHMM89]. Due to the space limitations of this paper, we cannot present such an implementation of architectural design in detail. For this reason, we will concentrate on the representation of an enriched design description and on how to achieve the desired active system behavior.

2. AN OVERVIEW OF THE APPLICATION

The goal of our intelligent CAD system is to support the architectural design of one-story houses for families, based on requirements and needs specified by the user. All these specifications as well as all relevant information concerning the design are explicitly represented in an integrated product model internally managed by our system. This product model contains several submodels describing geometrical, topological, and functional aspects of the objects under design, thereby providing distinct views of our representation. The definition of the product model was accomplished appropriately by exploiting the object-centered representation framework of KRISYS.

KRISYS allows a natural description of design objects, integrating all its aspects into one KB object. Abstraction concepts (classification, generalization, association, and aggregation) [Ma88a, BMW84] were used as the basic mechanisms for describing the organizational semantics of the application domain. Built-in, abstraction-related, reasoning facilities (inheritance, set-properties, membership stipulations, and implied predicates) [Ma88a, RHMD87] were exploited to continuously guarantee the structural and semantic integrity of the product model. Distinct aspects of the design object were represented using different abstraction concepts or distinct hierarchies of the same concept. One hierarchy was employed for representing geometrical, another for functional, and a last one for topological information about the object. This has the advantage that different aspects of a design are easily distinguishable in the model. Following this approach, each room of a house is described by an object, which is either a component of one of these hierarchies, or contains properties relating it to them. Therefore, a room can be viewed from a functional, a topological, and a geometrical point of view and acts as a link between the different hierarchies, thereby guaranteeing the overall consistency of the design.

In the following, we will first explain how the distinct aspects of a design object are represented by means of abstraction hierarchies and integrated into the room descriptions. All these hierarchies are manipulated during the design process, dynamically generating objects for the representation of important design information and incrementally completing the specification of such objects in order to construct an architectural sketch of the house. These manipulations, which are performed in different phases of the design process, are described in more detail in the subsequent sections of this chapter.

2.1. A GENERAL OVERVIEW OF THE REPRESNTATION

As already mentioned, the house under design can be seen from a functional, a topological, as well as a geometrical point of view.

2.1.1. FUNCTIONAL VIEW

The functional aspects are expressed in terms of <u>usages</u>, which can be associated with the rooms of a house. We have introduced a generalization hierarchy into our system, which contains the information about usages relevant for the design process (see figure 2.1 for a partial view of this hierarchy). There can be different levels of abstraction when considering the usages. For example, both 'reading' and 'watching-tv' may be seen as the more abstract usage 'leisure'. These different levels of abstraction are also reflected by the levels of detail in the requirements for the house specified by the designer (see chapter 2.2 for details).

The usage hierarchy contains information about <u>all</u> the usages the system knows about. This information is exploited for a specific design by creating instances of the usages which are required for the house (either by specification of the designer or by default assumptions). In our example (see figure 2.1), usage instances are created

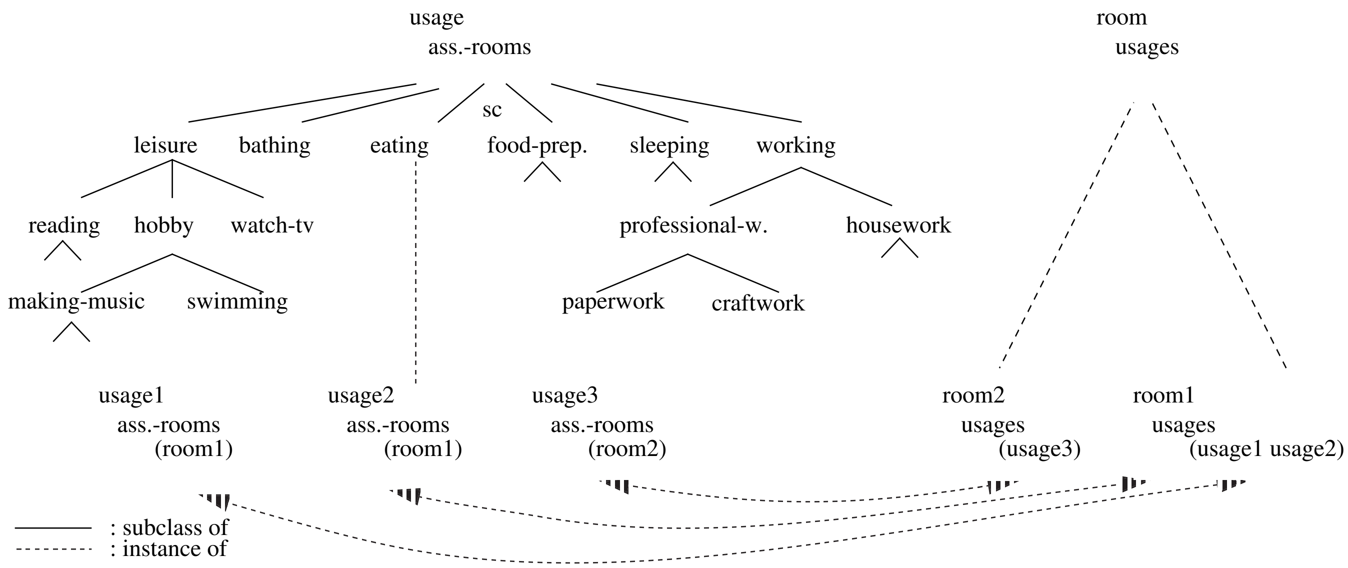

Figure 2.1: Representation of functional aspects

among others for 'food-preparation', 'eating', and 'leisure'. During the design process, rooms are created (as instances of the class 'room') and associated with the specific usages in order to appropriately fulfill the functional requirements for the house. The attribute 'usages' in the class 'room' and 'ass.rooms' (associated-rooms) in the class 'usage' (as the 'inverse' attribute), which are inherited respectively by the rooms and usages, are used to represent this relationship. In figure 2.1, for example, 'room 1' is associated with the instances of 'leisure' and 'eating', while 'room 2' is supposed to be used for 'food-preparation'.

2.1.2. TOPOLOGICAL VIEW

The topological aspects of a house under design are characterized by <u>orientations</u> (e.g., a room is located at the south side of the house) and by <u>adjacencies</u> in the sense that two rooms, for example a bedroom and a bathroom, are located next to each other to achieve a direct connection between them (e.g., through a door). They are represented in a separate hierarchy with the class 'topology' as its root and the two subclasses 'orientation' and 'adjacency' (see figure 2.2). The instances of these two subclasses describe the specific topological constraints known by the system, which are established either through legal restrictions or default-assumptions. Please note that orientations as well as adjacencies are primarily specified according to usages, and not to rooms. For example, the orientation 'sleep-east' is associated with the usage 'sleeping', and 'bathing-sleeping' relates the usages 'bathing' and 'sleeping', which is specified in the 'usages'-attribute of the objects. This is important, since a single room may be considered with respect to several usages (e.g., room 1 in figure 2.1) and is then affected by several topological restrictions. The topology hierarchy itself is not modified or extended (e.g., by creating new instances) during the design session. It is merely used to propagate the topological restrictions to the

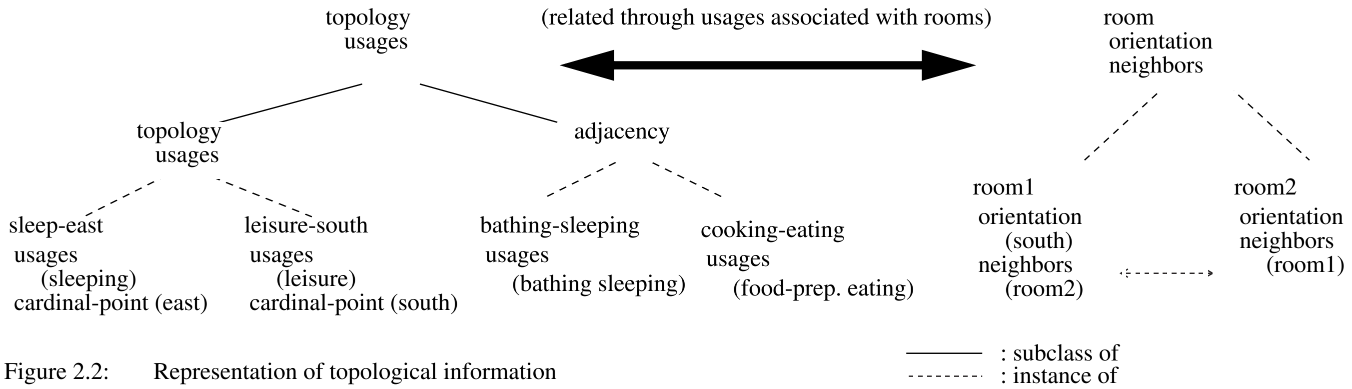

Figure 2.2: Representation of topological information

rooms created by the system according to the specific usages associated with them. Rooms are therefore described by two topological attributes: 'orientation', which contains the cardinal points of the rooms according to the orientations of usages, and 'neighbors', relating adjacent rooms.

2.1.3. GEOMETRICAL VIEW

Besides functional requirements and topological information, the geometrical aspects are fundamental for actually representing and producing an architectural sketch of the house under design. These are characterized in terms of geometrical areas. A house consists, from this point of view, of several areas which may (recursively) be split into subareas that finally are identical to specific rooms. We have represented this tree-like structure having rooms of the house as its leaves with the help of the aggregation concept: an area is considered as an aggregate consisting of component areas (i.e., subareas). This aggregation hierarchy is shown in figure 2.3. All areas are instances of the class 'area' (e.g., 'area 1'), thereby inheriting geometrical attributes such as 'position', 'dimension' (i.e., lengths of the sides), and 'size'. They can be components of other areas (e.g., 'area 1' is a component of 'house') and in turn have other areas as components (e.g., 'area 2' and 'area 3'). Since rooms are considered as areas themselves, the class 'room' is a subclass of 'area', but with the restriction that rooms (e.g., 'room 1' and 'room 2') cannot have other areas as components (i.e., they are 'atomic' components).

2.1.4. INTEGRATED VIEW

As already mentioned, rooms play an important role in the integration of the different hierarchies, acting as a link between them. This is illustrated in figure 2.4, where the full specification of the room 'room 1' is shown. From a functional point of view, 'room 1' is associated with the usages 'usage 1' and 'usage 2', which are respectively instances of 'leisure' and 'eating'. Its topological role is described by its orientation ('south') and its neighbors ('room 2'). Geometrical aspects are characterized by its attributes 'position', 'dimension' and 'size', and by the fact that it is a component of 'area2'. An integrated view of this information is especially relevant for the maintenance of the overall consistency of the different hierarchies. For example, the size of a room is closely related to the usages intended for it. This information has to be exploited by the process designing the geometrical areas of the house.

2.2. THE PROCESSING PHASES OF OUR DESIGN SYSTEM

In the following sections, we will take a closer look at the different design phases supported by the system and show how they utilize the information represented in the above hierarchies. Such hierarchies not only provide three different views of an architectural design, but also reflect the three different phases carried out by human architects during a design process.

2.2.1. GENERATING A FUNCTIONAL VIEW OF THE HOUSE

Specifying usages

In the first phase of the design process, our system questions the user as to his/her requirements. These may be formulated on a high semantic level (e.g., 'I need to work at home and preferably not on the side of the street.'). Furthermore, the specification of such requirements may be incomplete in the sense that they are not suffi-

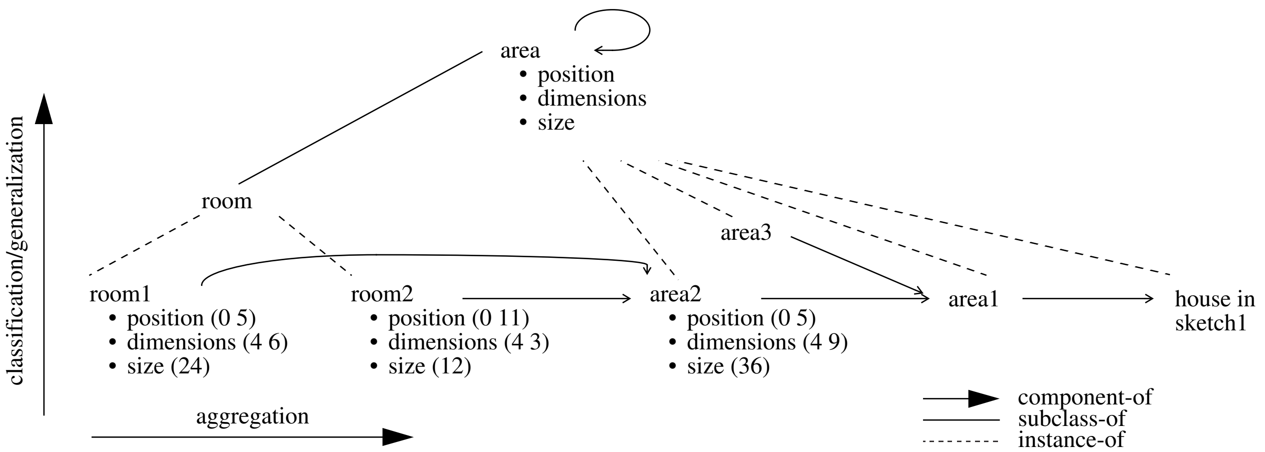

Figure 2.3: Representing the geometry of an architectural sketch

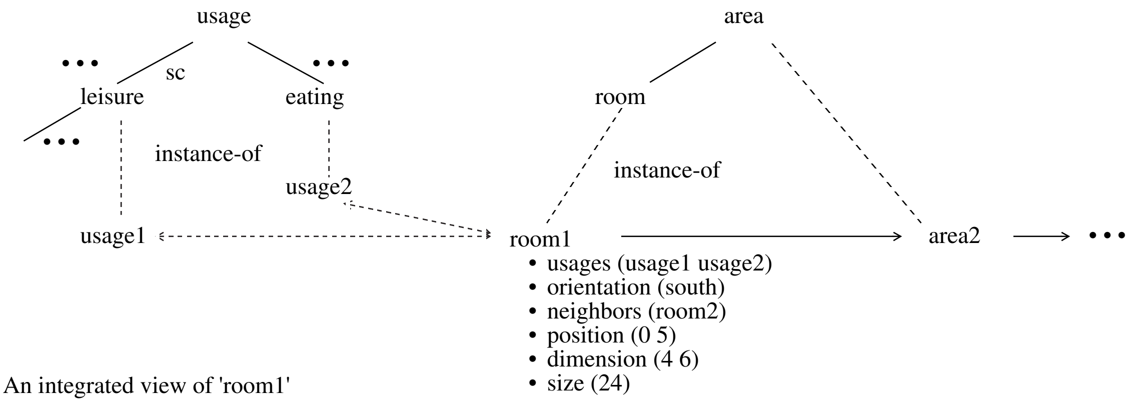

Figure 2.4: An integrated view of 'room1'

cient to directly achieve a final design stage. For this reason, the system utilizes its knowledge (expressed by rules representing standard requirements, laws, etc.) to supplement the user-sketched blueprint. In other words, a significant part of the knowledge of our architectural design system is represented as rules, which are responsible for the tasks of the system, such as the derivation of activities or usages that are standard in every house or required by law (e.g., eating, sleeping, heating...) and are not necessarily specified by the user.

There are several levels of abstraction in which activities can be derived by the system. They correspond to the levels of the generalization hierarchy defining the usages (figure 2.1) and reflect different degrees of detailing of the functional aspects of a house. Actually, every level expressed in this hierarchy represents a valid degree of detailing of a functional design. This means that our system, just like human architects, can specify functional requirements at any level, depending on the degree of detailing it 'believes' to be necessary in each particular case. That is, it makes use of information about the 'context'[1] in which a particular house is being constructed in order to dynamically determine whether it is necessary to further detail one or another functional aspect of the house. For example, if the members of a family play different sports or have several hobbies, a detailed analysis of the partial hierarchy under 'leisure' (see figure 2.1) is required, whereas a house in which the parents wish to work needs a detailed investigation of activities under 'working'. So, what makes the system move further into a more detailed level is its 'ability' to consider information about the 'context' and thereby to realize the necessity to further detail some functional aspects.

1. This involves general information about the user and his/her family, such as job, sex, age, habits, etc.

Every time the system enters a more detailed level, the user has the chance to express his/her requirements that correspond to this level and the system itself derives additional activities by default assumptions. For example, when moving down from 'leisure' into the next lower level, the system can assign the activity 'watch-tv' even if the user does not give any other information about his/her activities regarding 'leisure'. Whenever an activity is defined either by the user or the system, an instance of the corresponding 'usage' class is created by the system in order to represent this functional requirement.

The result of this design step is therefore a set of requirements expressed in terms of activities to be supported by the house being constructed, that is, a functional description of the house represented in terms of the instances created in all levels of the 'usage' hierarchy. Note that in this first phase, our system does not deduce any kind of rooms (such as kitchen, bedroom, etc.) that the house has to have. It first abstracts from particular rooms, analyzing a house only from a functional point of view in the same way as human architects. This is particular important because several combinations between activities and rooms are possible when specifying rooms in the following step of this design phase. For instance, the activity 'eating' can be 'associated' with a kitchen, a large living room, a dining room, etc. Since activities directly influence the characteristics of a room (e.g., the size of a room strongly depends on the activities and the number of people performing such activities in the room), they are viewed as a group of constraints that should be satisfied after the rooms of the house have been specified.

Deriving rooms

In the next step, our design system specifies rooms that can satisfy the assigned functional constraints. These rooms have, at this step, no geometrical or topological information associated with them, but already represent the actual rooms that have to be considered when generating the real sketch of the house later on. This generation process is carried out in a stepwise fashion in which every instance of usage that was created in the first step is examined and assigned to either an existing room or to a room just created for this purpose. Since the associations between usages and rooms can be defined either by the system or the user, this process is guided by integrity constraints that ensure that both the user specifications and deductions of our system will generate room descriptions reflecting a semantically correct state.

Our system exploits the integrity constraints to control the overall process of this second step, checking the actions of the user and triggering systems operations to complete user specifications by enforcing the semantic integrity of the KB. When the user directly specifies an activity for a particular room, our system exploits such constraints to check whether this new activity is compatible with a previously asserted usage. There are statutes that strictly determine incompatible usages. The architect has, for example, to consider that facilities for both cooking and bathing should not be provided in the same room. For this reason, we have attributes expressing 'contradictory activities' in the objects describing activities, which contain all incompatible activities for every activity in the KB. Procedures represented as demons attached to the attribute expressing the usage of a room are immediately activated when a usage is added in order to check if one of the old usages is a 'contradictory activity' of the new one. Upon detecting an inconsistency, an error is reported, automatically eliminating the previous change of the KB.

Another approach applied by the system when integrity constraints are violated is to trigger additional operations to transform an inconsistent KB state into a consistent one, instead of rejecting the changes. This triggering mechanism is automatically carried out by our design system, whenever, for example, information is incompletely specified by the user. The mechanism is also desirable within our integrated product model, where changes concerning one representation of an object (e.g., its functional aspects) should automatically cause changes in the other representations (e.g., the geometry). Such an approach is realized by either demons to trigger the appropriate changes or by rules to dynamically derive information and insert it into the room description. As already mentioned, the user of our design system may specify activities or usage for the house, but need not to relate them to any rooms. This specification is then completed by the system in this second step. For example, when the user specifies the activity 'professional-working', the system activates rules associated with the 'professional-working' object in the KB. These rules either generate a new room for the activity (e.g., the office room) or establish a relationship between the activity and an already existing room (e.g., the living-room), considering, for example, extra places for a work desk. Note that the modification of the size of the living-room because of the association of the activity is an example of the use of rules to keep track of the dependencies between functional and geometrical aspects of the product model. This kind of dependency is exploited by our system at the next phase of the design process to complete the descriptions of the rooms.

2.2.2. DETERMINING ROOM CHARACTERISTICS

Defining orientations and adjacencies

The goal of the second design phase is to completely specify the topological aspects of the house and the geometrical aspects that directly depend on the functional and topological ones. Again, these aspects are either determined by the user (e.g., 'I preferably do not want to work on the side of the street') or derived by the system. Both kinds of definitions are restricted by the constraints and dependencies existing in this application world which are described in our product model as constraints involving various levels of granularity of knowledge: attributes of objects (e.g., the size of a main sleeping-room may not be smaller than 12 square meters), objects (e.g., the functional view of an object, such as the activities associated with it, must be in accordance with its geometrical and topological view, i.e., its size and position within the house), and groups of objects (e.g., connections between rooms). Note that most of these constraints are derived from laws and statutes that rule minimum areas for rooms, the amount of direct illumination in square meters that a room must have and consequently its position within the house (inner or outer), the kind of ventilation (direct or indirect)

that a room should have and therefore its location and connections to other rooms.

System derivations are carried out by exploiting the topology hierarchy (figure 2.2), in which orientations and adjacencies of 'usages' are defined, in order to propagate this information respectively to the attributes defining orientation and neighbors of the rooms. Geometrical characteristics, such as the size of the rooms, are also derived on the basis of the usages that have been assigned to a particular room and of laws or statutes as previously mentioned.

Improving the design diagram

At the end of this step, our system has a complete description of the rooms, containing the location of a room within the house (e.g., bathroom and bedroom should be at the east side of the house), the connections between rooms (e.g., the kitchen should be connected directly to the entrance, but not to the bedroom), sizes of rooms, etc. This description is then used by the system to produce the so-called design diagram, which is usually employed by human architects to analyze and ameliorate complex designs before a concrete sketch is generated. This diagram is actually implicitly represented in our product model, given by the rooms, their topology, geometry, and usages. It is a graph representation of the object being designed (in our case a house) in which nodes and edges symbolize respectively rooms and connections between rooms (see figure 2.5).

Improvements are then performed by transforming the graph into another one, presenting some important characteristics. First, such a graph has to be planar since connections in a house cannot cross each other. This can be achieved by either removing some connections or creating additional rooms, such as a corridor. Secondly, the graph cannot have regular rooms with an excessive number of junctions to other rooms, as for example, the 'living/dining' room illustrated in figure 2.5a. To reduce the number of such junctions between regular rooms, special (connection) rooms (such as corridors, foyers, entrance halls, etc.) can be introduced for this purpose, as shown in figure 2.5b. Note that the transformation of the graph has to be carried out in several steps since the introduction of a connection room (the foyer in figure 2.5b) does not necessarily reduce the number of junctions between regular rooms to an acceptable minimum. In the example, a corridor was then created to reduce the connections of the 'living/dining' room and of the bathroom (figure 2.5.c). Finally, further transformations may be performed to eliminate inadequate connections (e.g., the connection between the two bedrooms in figure 2.5b is through the bathroom), to reduce the communication between groups of rooms (e.g., rooms in a private area such as bedrooms and bathrooms should not be connected directly to rooms in the social area), to split very large rooms (thereby reorganizing the connections), etc.

Certainly, the system considers the constraints mentioned above at each transformation of the design diagram in order to produce only diagrams satisfying them.

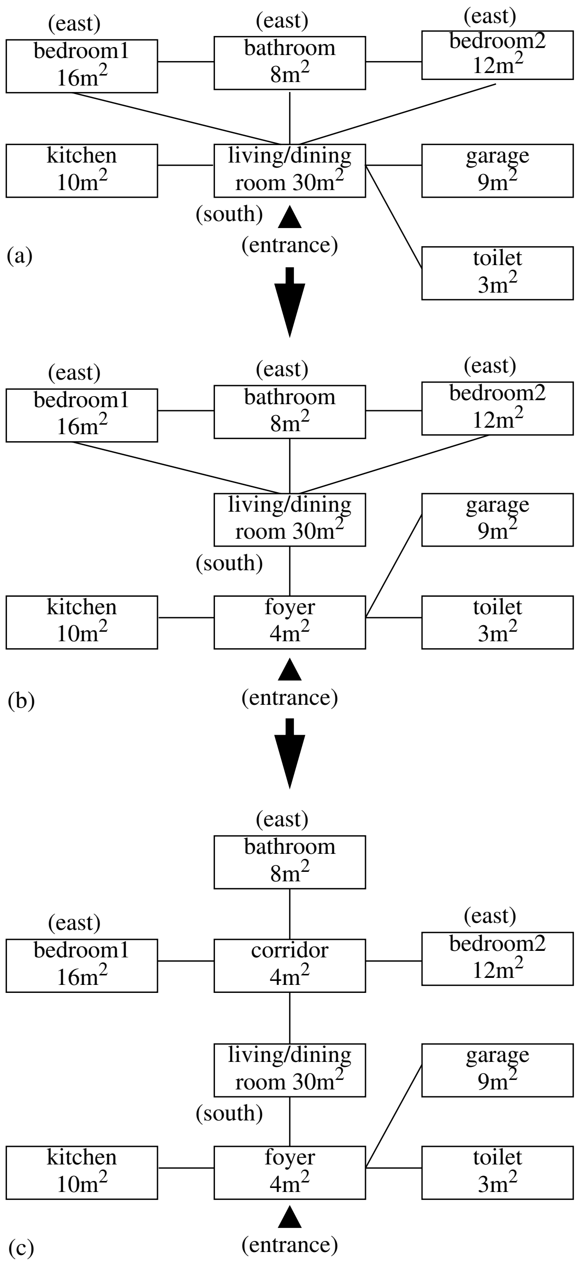

Figure 2.5: Improvement of the design diagram of a house

2.2.3. CONSTRUCTING AN ARCHITECTURAL SKETCH

In the last phase of the design process, our system uses the complete specification of the rooms generated during the previous phases to construct an architectural sketch of the house. As mentioned above, the geometrical aspects used to represent and produce this design sketch are characterized in terms of geometrical areas. The algorithms performing the actual geometrical design, i.e., partitioning the house into geometrical areas corresponding to the rooms (once their usage, functionality, and other requirements are fixed), are modeled as operations of the class 'area' by means of methods.

At the beginning of this phase, an area representing the geometrical aspects (size, dimension, etc.) of the whole house is generated, and all rooms derived in the previous phases are associated with this area. The methods performing the geometrical design start with this area, dividing it into two subareas and then choosing one of them to deal with in the next step. In each of the following steps, the methods divide a chosen area into two subareas by means of an breadth-first strategy, thereby generating an tree-like structure represented as an aggregation hierarchy. Each node of the tree therefore corresponds to a design decision since they express a possible partition of an area. After each division, the rooms associated with the area being split are distributed between the new subareas, and the geometrical aspects of the two resulting areas (the size, the dimension and the position) are calculated by the system. The algorithm performs this way until all areas contain exactly one room. At this point, the areas represent the leaves of the tree and express a sketch of the house.

The topological constraints (orientation and adjacencies) as well as the geometrical properties (i.e., the required size) of the rooms belonging to an area are used to decide how an area is to be partitioned. For example, rooms having the same orientation should be grouped together, and rooms that are connected to each other are either not separated in two different areas or kept in adjacent areas as long as possible. If there is no area that can be further divided into subareas, without causing the violation of some constraint, the algorithm uses backtracking to generate alternative decision nodes.

If the construction of the architectural sketch succeed, i.e., every leave of the tree represents exactly one room, the user is asked to accept the actual sketch. At this point, the user can then require an explanation of the design decisions, which is easily supplied since each node of the area tree represents a decision taken by the system. If the user rejects the solution generated by the system, alternatives can be derived in a non-redundant way by backtracking in the tree and exploiting previous design decisions. Different design alternatives are kept in the KB until the user makes his decision. The user can also add new (topological or even functional) requirements or delete existing ones, leading to a partial repetition of previous design phases and to the revision of the decisions of the system. For example, adding a new adjacency constraint can invalidate design decisions stated previously, causing an redesign of certain areas or the whole house. Deleting constraints is especially required if no geometrical sketch of the house can be generated by the system.

After the design session is finished, the resulting sketch is kept in the KB. A user may therefore start a new session with the result of a previous one as the basis for his new design.

3. CONCLUSIONS AND OUTLOOK

In this paper, we have given an overview of our conception of an intelligent CAD application in the area of architectural design and sketched its realization using the knowledge model of KRISYS, a KBMS prototype developed at our university.

The system incorporates several features that we regard as essential for intelligent CAD. First of all, it shows an active system behavior. At all design stages, the system provides the user with appropriate design hints, relevant problem solutions, and diagnostic information, that he may utilize to improve his design. The ability of the application to actively support the user in his design activities is especially required when the case of incomplete design specifications is considered. If the user gives only part of the information that is necessary to successfully terminate a design step, the system tries to complete such specifications, utilizing the design knowledge incorporated in its knowledge base. Therefore, the overall design process can be seen as a kind of iteration between user and system. However, the extent of activities performed by the system is still being determined by the user. He may decide whether to leave certain characteristics of the design object unspecified in order to let them be derived by the system, but also may modify system-derived information at all design stages.

Another major goal during the conception and realization of our architectural design system was to come as close as possible to the design process performed by human experts (architects). Therefore, we have divided the overall design process into several design phases. In the first phase, only the functional aspects of a house are considered in terms of usages or activities (e.g., leisure, eating, sleeping). In the second phase, the house is designed with respect to its topological properties (i.e., orientations and adjacencies of rooms). In the last phase, the actual geometrical representation is fixed (in terms of geometrical areas), yielding a design sketch of the house. In order to effectively support these different design phases, an explicit separation of functional, topological, and geometrical aspects emerged as an essential feature to increase flexibility (e.g., with respect to the association of usages and rooms) and abstraction (e.g., viewing a house in terms of usages without considering distinct rooms). Nevertheless, the different aspects of a house under design, which are relevant at different design phases, are intertwined with each other through complex dependencies, which actually determine the global consistency of the design. The conception of our architectural system is therefore based on the notion of an integrated product model, containing representations of all different aspects of a design object (i.e., functional, topological, and geometrical). The dependencies between these distinct representations are also made explicit in the product model and may consequently be maintained and exploited by the system in order to achieve a consistent and construction-oriented design.

In summary, what can be considered as one of the major issues of our intelligent CAD application is the improvement of the overall design process by incorporating different design aspects (geometrical, functional, topological) which are related to different processing or design phases but are all integrated into the same design object (product model). Therefore, significant parts of the semantics of the application world, which are in general buried in various programs responsible for different design phases, became explicitly represented in an integrated product model, where they are available for all system components.

When examining the above described system characteristics, it becomes evident that knowledge modeling aspects can be considered as a key issue to the support intelligent CAD systems [HM85, HM90]. The KBMS KRISYS [Ma91, Ma88b] provides a rich spectrum of modeling concepts, which has been extensively employed in the implementation of our architectural application. In this implementation, the abstraction concepts provided by KRISYS were primarily important for the support of a semantically enriched object description. Additionally, they were employed to define a means for object organization which, in turn, was useful to describe distinct application aspects. This was done, however, in an integrated way. In order to keep the different representations of a design object consistent among all its representations, procedural mechanisms for knowledge representation (i.e., demons, methods, etc.) were used to ensure a high degree of semantic integrity and to relieve the user (or application program) from this part of integrity checking. Moreover, the integration of behavior into the application model in the form of procedural attributes, user-defined functions, or methods made it possible to describe actions in which application objects are involved, thereby permitting the integration of application-oriented operations into the system. Finally, reasoning facilities were required as basic mechanisms to deal with incomplete specifications as well as to control the overall application process, thereby also providing the necessary active system behavior.

Although we have covered several interesting issues with our architectural prototype implementation, the system is in its current version far from satisfactory. Additional improvements will therefore be part of our future work. One of the main goals will be related with the modeling of constraints, which has turned out to be one of the central issues in our system. The representation of constraints has to become more flexible in order to support different kinds of dependencies (e.g., needs of the user, laws) with different properties (hard or soft design goals). A more explicit and compound representation has to be achieved, which should also allow queries on constraints (e.g., 'give all constraints relevant for kitchens'). Additionally, the constraints should play a more active role in determining and guiding the system behavior. Nevertheless, the user should still be able to determine a suitable degree of system activity, which may force the constraints to be considered only as (passive) integrity constraints.

Another goal will be the improvement of the second and third design phases. The design diagram representing the topology of a house should be represented in a more explicit fashion in order to better support additional operations for its modification and exten-

sion. The geometrical design, that is generated by the system and may be accepted or rejected possibly leading to alternative designs, should significantly be extended so that the user can directly position or modify certain rooms at will in a graphic interface. The system will then only check the consistency of the sketch provided by the user.

Other efforts which are related to the support of more complex designs will involve providing alternative design versions at all stages of the design process (e.g., alternative design diagrams based on the same usages of rooms, etc.), which are then maintained by the system. We will possibly extend our application to support the design of many-story houses, thereby also considering aspects of cooperative work in this context.

Acknowledgment

We would like to thank the architect Claudia Mattos, who greatly contributed to the conception and realization of the design application described in this paper.

REFERENCES

BMW84 Borgida, A., Mylopoulos, J., Wong, H.K.T.: Generalization/Specialization as a Basis for Software Specification, in: On Conceptual Modelling (Perspectives from Artificial Intelligence, Databases, and Programming Languages), Topics in Information Systems, (eds.: Brodie, M.L., Mylopoulos, J., Schmidt, J.W.), Springer-Verlag, New York, 1984, pp. 87-114.

DHMM89 Deßloch, S., Härder, T., Mattos, N., Mitschang, B.: KRISYS: KBMS Support for Better CAD Systems, in: Proc. 2nd International Conference on Data and Knowledge Systems for Manufacturing and Engineering, Gaithersburg - Maryland, Oct. 1989, pp.172-182.

DHMS90 Deßloch, S., Hübel, C., Mattos, N., Sutter, B.: KBMS Support for Technical Modeling in Engineering Systems, in: Proc. 3rd International Conference of Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Charleston - South Carolina, July 1990, pp. 790-799.

GAPR88 Grabowski, H., Anderl, R., Patzold, B., Rude, S.: The Development of Advanced Modeling Techniques - Meeting the Challenge of CAD/CAM-Integration, in: Proc. 4th CIM Europe Conf., May 1988.

HM85 Hartzband, D., Maryanski, F.: Enhancing Knowledge Representation in Engineering Databases, in: Computer, Vol. 18, No. 9, Sept. 1985, pp.39-48.

HM90 Härder, T., Mattos, N.M.: An Enhanced DBMS Architecture Supporting Intelligent CAD (invited lecture), in: Proc. of Int. Conf. TECHNO-DATA '90, Berlin, Dec. 1990.

Ma88a Mattos, N.M.: Abstraction Concepts: the Basis for Data and Knowledge Modeling, in: 7th Int. Conf. on Entity-Relationship Approach, Rom, Italy, Nov. 1988, pp. 331-350.

Ma88b Mattos, N.M.: KRISYS - A Multi-Layered Prototype KBMS Supporting Knowledge Independence, in: Proc. Int. Computer Science Conference - Artificial Intelligence: Theory and Application, Hong Kong, Dec. 1988, pp. 31-38.

Ma90 Mattos, N.M.: An Approach to Knowledge Base Management - requirements, knowledge representation, and design issues -, Lecture Notes in Artificial Intelligence (subseries of Lecture Notes in Computer Science), Springer, to be published.

Ph88 Pham, D.T.: Expert Systems in Engineering, Springer-Verlag, 1988.

RHMD87 Rosenthal, A., Heiler, S., Manola, F., Dayal, U.: Query Facilities for Part Hierarchies: Graph Traversal, Spatial Data, and Knowledge-Based Detail Supression, Research Report, CCA, Cambridge, MA, 1987.

SGL89 Spur, G., Germer, H.-J., Lehmann, C.: Impact of Geometric Modeling for Computer Integrated Manufacturing, in: International Symposium on Advanced Geometric Modeling for Engineering Applications (IFIP & GI), Berlin, FRG, 1989.

SR86 Sriram, D., Rychener, M.: Expert Systems for Engineering Applications, special issue of IEEE Software, Vol. 3, No. 2, March 1986.