in: Proc. Int. Conf. on Computer-Aided Design and Computer Graphics, Beijing - China, Aug. 1989, pp. 326-331

Knowledge Base Management Systems - the Bases of Advanced CAD

S. Deßloch, T. Härder, N. Mattos, B. Mitschang
University Kaiserslautern, Department of Computer Science,
P.O. Box 3049 , D-6750 Kaiserslautern, West Germany

Abstract

Semantic expressive representation of design objects, active system behavior combined with reasoning facilities, and efficient implementation concepts are necessary requirements for the construction of better CAD systems. Here, we describe our approach to a knowledge base management system and exemplify its usage for advanced CAD systems.

1. Motivation

Currently, there is a lot of research investigation aimed at better computer-aided design (CAD). These so-called *advanced* CAD systems comprise the overall design process, taking at least several or all design steps into account and exhibiting an active system behavior. Thus, they are capable of providing a more intelligent interface to the user, offering for example solutions or hints to current design problems and guaranteeing design-specific integrity constraints.

The approach mostly taken to yield a better suited designed environment is to couple an (existing) CAD system with a database system (DBS) or with an expert system tool (XPS tool):

- The reason to add a DBS lies in its ability to manage persistent data in an integrated and efficient way. Since different tools within the broad area of CAD (e.g. tools for concept finding, solid modeling, computation, simulation, and testing) refer to the same design objects, all aspects of these objects are represented in a unique and non-redundant manner, allowing for consistent and uniform object handling. This integrated view of different object aspects is also necessary for the integration of other CA* areas (e.g. CAM, CAP) to achieve computer-integrated manufacturing, i.e. CIM.
- On the other hand, the idea behind the usage of an XPS tool lies in an improved and more intelligent system behavior [1,2]. An active CAD system is able to provide the design engineer with appropriate design hints, relevant problem solutions, refined simulation results, and adequate diagnostic information in all stages of the design process. Thus, it could take care of, for example, whether the object under consideration is still manufacturable (considering manufacturing restrictions and company guidelines) or whether some time dependencies concerning the manufacturing process have to be taken into account. Concerning the recent upcoming terminology, these systems are also called *intelligent* CAD systems.

This probably incomplete list of arguments for the use of DBS and XPS components in an advanced CAD system, obviously reveals that a practical approach should incorporate the advantages of both DBS and XPS tools. However, it should by no means be based on an extension or coupling of existing systems. The lack of component integration in existing architectures is responsible in most cases for cumbersome handling and for oftenly quite ineffective performance. For this reason, the approach that is described here is totally different to the above mentioned ones since it is centered around a so-called *knowledge base management system* (KBMS) that integrates artificial intelligence (AI) and database (DB) techniques in an effective way.

The paper is organized as follows: firstly, we define our conception of *advance*d CAD by means of its inherent characteristics. Secondly, we show that the optimal solution could be achieved only by an efficient integration of both DB technology and AI techniques within a KBMS. Using the KBMS KRISYS [3] that has been developed at our university and some evident examples from architectural design we demonstrate the applicability of our approach.

2. Characteristics of Intelligent CAD

In *advanced CAD systems*, the 'user' is not the only active unit within the overall design process - as it is in conventional CAD systems. In such environments, the CAD system itself exhibits an active behavior being therefore capable of providing a more intelligent user interface. 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 user's decisions (e.g. in special cases). The main goal of advanced CAD is to improve the whole design process by incorporating not only geometrical but also technical and functional aspects as well as construction and manufacturing dependencies along each design step. This leads to a penetration of the design phases and to a manufacture-oriented design methodology. For example, the technical term feature modeling has come up recently to express the shift from simple geometric modeling (where it is only dealt with geometry) to a modeling concept that additionally considers technical, functional, and manufactural aspects of the design object at hand [4]. In the following, we concentrate on the most important facts inherent to advanced CAD:

First of all, an *integrated product model* that can be efficiently managed is a mandatory prerequisite. This product model contains several submodels describing geometrical, topological and technical aspects, structural and sizing aspects as well as functional and dynamic aspects of the objects under design. Additionally, production information and manufacturing information has to be included. For example, it seems to be highly desirable to model the machines and their operations to be able to check for manufacturability of the actual design object (e.g. considering manufacturing restrictions and company guidelines). Obviously, it is quite hard to keep these different representations belonging to the same design object consistent within each one and among all its representations. Thus, it is necessary to use a modeling system that enables accurate semantic expressiveness to ensure a high degree of semantic integrity already at the modeling level and to relieve the application program from this part of integrity checking. In engineering applications, there are several integrity constraints concerning geometry, topology, technical and functional behavior of the current object. Of course, there are higher levels of consistency based on the above mentioned ones, e.g. stress analysis, manufacturability or quality assurance.

Another important aspect are the different kinds of dynamism typically found in a design process. Each CAD system has to keep track of all design decisions and their results. This leads to version graphs, design alternatives, and configurations each of them having its own semantics with respect to a specific application area. The system is able to maintain these dependency structures only if there are adequate measures to express the semantics and the reactions due to changes. Furthermore there must be an ability to cope with changing environments. For example, new technologies or new manufacturing guidelines have been introduced or new machines with new operations have been bought. These changes have a significant impact on the optimality of the design and must be noticed for this reason. Additionally, a good system allows its user interface to be adapated to the user's conception. Thus it should be possible for a user to build up his own name space, his own design methodologies or to select standard designs from libraries or catalogues etc. Alltogether, easy adaptability is an important system property.

Along with the process of design there are a lot of changes that have to be recorded and a lot of subsequent actions that have to be triggered due to dependency or derivation relationships. If the system offers enough knowledge and reasoning facilities it is capable to maintain these relationships automatically thereby simplifying the application programs or even the user's work [5]. Furthermore, it is able to propose some design alternatives or similar design plans perhaps using libraries that contain standard design plans or parameterized skeleton plans. Additionally, the automatic generation of production or manufacturing plans connecting the phases of design and manufacturing becomes conceivable. With this, it is also possible to generate for example NC code, assembly data, and quality assurance data.

Summing up, our conception of advanced CAD comprises the following properties:

- *semantic object representation*, i.e. sufficient structural and behavioral modeling facilities to enable a semantic expressive representation of each design object
- *active system* behavior combined with *reasoning concepts* to guarantee a consistent design and to provide appropriate design hints, relevant problem solutions, refined simulation results, and adequate diagnosis information in all stages of the design process.

Despite the efficient provision of these concepts a practical system has to guarantee durability, consistency, and isolation of each design phase with respect to the whole design process by means of an appropriate design-transaction concept.

3. The KRISYS Approach

3.1 Overview of the System Architecture

As shown in figure 1, KRISYS is architecturally divided into three hierarchically ordered layers. They are responsible for a stepwise abstraction process and for the realization of the corresponding tasks within each layer. The application layer realizes an external interface where the knowledge is viewed in a more abstract and functional manner. This object-abstraction interface is defined by the powerful query and manipulation language KOALA (KRISYS Object Abstraction LAnguage) that keeps the end-user or application programs independent from the representation of the underlying knowledge model. The terms of the knowledge model of KRISYS, called KOBRA (KRISYS OBject-centered RepresentAtion Model), are defined by a mixed knowledge representation framework. Thus, the engineering layer implementing KOBRA realizes at its interface an object-centered view of knowledge representation and manipulation to the knowledge engineer. The task of the lowest layer is to efficiently cope with storage of knolwedge. At this level, most of the issues are related to traditional DB technology applied to large KB, possibly shared by multiple users: storage structures, access techniques, efficiency, integrity features, transaction support, etc. Therefore, this layer is realized by a non-standard database system (NDBS) which seems to be quite advantagous in a KBMS architecture for a number of reasons [6]. NDBS are much more powerful than traditional DBS and are, for this reason, able to satisfy knowledge maintenance requirements. Our implementation layer consists of two distinct modules. The DBS kernel is responsible for the storage and management of the KB. The Working-Memory System embodies a '*nearby application locality*' concept, thereby substantially reducing DBS calls as well as the path length when accessing the objects of the KB. This component manages a special main memory structure, called working-memory, which acts as an application buffer, in which requested KB objects are temporarily stored.

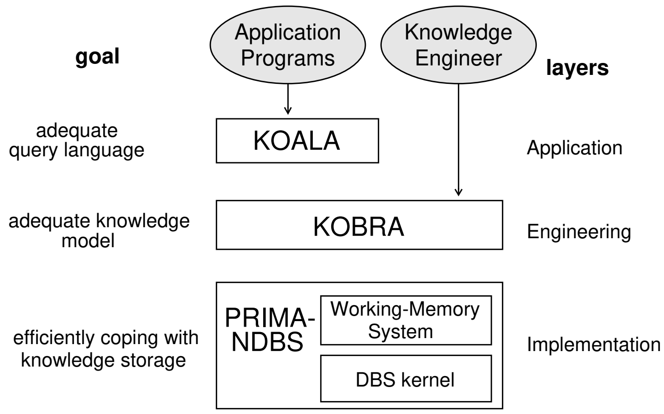

Figure 1: Overall system architecture of KRISYS

3.2 Short Notes on the Knowledge Model of KRISYS

KOBRA supports an *object-centered representation* of the application world. That is, every entity of this world is represented as a *schema*, which has attributes to describe its characteristics and is clearly identified by its name. A schema corresponds to a frame or unit in other systems and must not be confused with a DB-schema. Attributes are either *slots*, representing properties and relations to other objects, or *methods* describing the behavior of the entity. In order to characterize an object in more detail, attributes can be further described by *aspects* (possible values and cardinality specifications, default values, etc.).

For structuring the KB, the knowledge model supports the *abstraction concepts* of classification, generalization, association, and aggregation [7]. These concepts are seen as special, predefined relationships between objects, defining the overall organization of a KB as a kind of complex network of objects. KRISYS supports an *integrated view* of KB objects: there are no different representations for classes, sets, instances, complex objects, etc. They are all integrated within a schema, which may describe an object related to many others via several abstraction relationships. Therefore, the same schema can, for example, represent a class with respect to one object, and a set or even an in-

room 1	
orientation	-
position	-
size	16
unit	square-meters
neighbors	room 2, room 4
usage	sleeping

Figure 2: Sample schema description

stance with respect to another. As a consequence, the difference between data and meta-data, which is usually apparent in existing data models, is eliminated in KRISYS, so that *meta-information is integrated into the KB*. The semantics of the abstraction concepts are guaranteed by *built-in reasoning facilities* provided by the system [7]. Finally, to represent procedural characteristics of the application world, KOBRA offers methods, demons, and rules, with which behavior of design objects, reactions to real world events, and the problem solving know-how may be respectively represented.

3.3 Using KRISYS to Model Advanced CAD-Systems

As an example of an advanced CAD-system supported by KRISYS, we use an architectural design application currently being implemented. The task of this system is to design one-story houses for families based on requirements and needs specified by the user. During a first design-phase, it questions the user as to his requirements, which may be incomplete in the sense that they might not be sufficient to directly achieve a final design stage. The system utilizes its knowledge (e.g. standard requirements, laws, etc.) to supplement the user-sketched blueprint, creating additional rooms and properties of rooms and constructing a functional description of the house. In a second phase, our application uses this description to generate an architectural sketch. The user may accept this sketch or reject it, causing the system to generate alternative ones. New requirements can also be added, or existing ones can be removed, leading to a (partial) repetition of the first design phase.

Object structure

Everything existing in the application domain is expressed as an object of the KOBRA model, the so-called *schema*, in which descriptive, operational, and organizational aspects of the real world are integrated. In figure 2, we give an example of a schema representing a certain room of a house. Its properties are described by the attributes and attribute values: the size of the room is 16 square-meters, it is intended to be used for sleeping and is adjacent to two other rooms, namely 'room 2' and 'room 3'. The orientation and position of the room are not yet specified or determined. Via the aspect 'unit' for the 'size'-attribute, 'square-meters' is fixed as unit for the room size.

KB organization

KRISYS supports an *integrated view* of the KB by means of different abstraction concepts. A partial view of our example KB is presented in figure 3. The schema 'bed 1' has, for example, three different 'roles' at the same time: it is an instance of 'beds', an element of 'users-furnishings', and a component of 'room 1'.

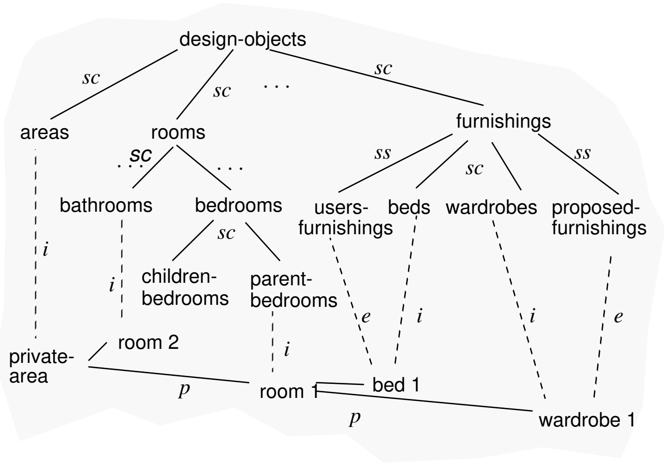

Figure 3: An integrated view of the KB

.As already mentioned, the semantics of the abstraction concepts is guaranteed by KRISYS via *built-in reasoning facilities* [7]. E.g. inheritance, which is carried out according to the generalization/classification-relationships, is the reasoning as to the structure of an object. 'Room 1' inherits all properties of 'parents-bedrooms', 'bedrooms', 'rooms', and 'design-objects'. The aggregation-relationships are the basis for reasoning with so-called implied predicates. For example, the fact that 'room 1' has a size of 16 square-meters implies that the area in which it is contained must be at least 16 square-meters in size. The concept of association allows the definition of set properties. For the final costs involved with the new house, the prices of all furnishings proposed by the system have to be considered. The set property 'total-price' of 'proposed-furnishings' is defined as the sum of all the prices of its elements. Upon changes, like the insertion of a new piece of furniture into the set, the new total price is automatically recalculated by KRISYS.

Supporting an integrated product model

An object-centered representation can be directly utilized for the definition of an integrated product model. It allows a natural description of the design object, integrating all aspects of the product into one KB-object. The abstraction concepts can be used as the basic mechanisms for describing the organizational semantics of the application domain. Distinct aspects of the design object may be modeled using different abstraction concepts or distinct hierarchies of the same concept (e.g. one class hierarchy for representing geometrical, another for functional, and a last one for manufacturing information about the object), which has the advantage that they are easily distinguishable in the model. An integration of all aspects into one object is easily achieved by overlaying the corresponding hierarchies.

In our architectural design system, we have described aspects concerning the usage, adjacency, and geometry of objects within three distinct classes (see figure 4). The class 'usage-objects' describes all aspects related to the usage of a design-object (e.g. a room) or the activities associated with it. Information about the adjacency of objects (e.g. when two rooms are connect via a common door) is captured by the class 'adjacency-objects', while all geometrical aspects, such as the size or the position of a room within the house, are covered by 'geometrical objects'. The class 'design-objects' inherits all attributes of its superclasses and therefore integrates the different aspects related to the design object. Since the design objects, like areas, rooms and furnishings are also embedded in an aggregation and an association hierarchy, their role as complex objects, object components, or elements are also considered. .

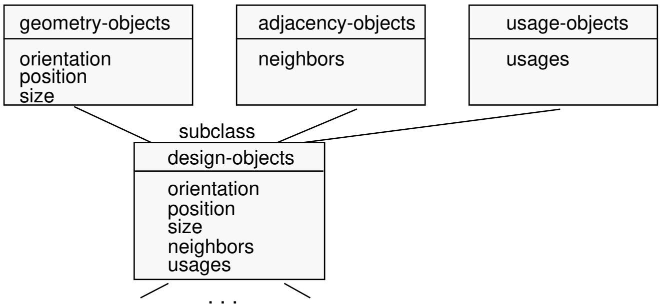

Figure 4: Different aspects of the design objects

Integrating behavior into the application model

Operations and actions in which objects are involved are included into a schema description as procedural attributes (methods). Our application utilizes methods for several tasks. For example, the algorithms performing the actual geometric design, i.e. partitioning the house into geometric areas corresponding to the rooms (once their usage, functionality, and other requirements are fixed) can be modeled as operations of 'geometry-objects'. The end-user of the system is of course not directly concerned with operations upon geometrical representations. His interaction with the application is usually settled at a higher semantic level. E.g. the method 'add-neighbors', defined in 'adjacency-objects', is performed when the user adds new requirements about the connections between rooms currently under design. Since these new requirements can invalidate previously stated geometric design decisions, the method 'add-neighbors' may, in turn, call methods defined in 'geometry-object' to cause a redesign of certain areas or of the whole house.

Maintaining the semantic integrity

A significant part of the application world semantics is embodied in restrictions of and dependencies between certain aspects of the world. KRISYS provides several mechanisms for explicitly describing integrity constraints and integrating them into the application model. Figure 5 gives an example using the class 'design-objects'. Every design-object is described by exactly one size, which is expressed through the value '[1 1]' of the cardinality aspect. Using the 'possible-values' aspect, we can state, that the minimum size of a room is 2 square meters. This aspect can also be used to ensure a constraint similar to the referential integrity in database systems: in order to assure that the values of the usage-slot, which represents the relationship between a design-object and its usages, always reference an actual instance of the class 'activities', we assert '(INSTANCE-OF activities)' as the value of the appropriate possible-values aspect.

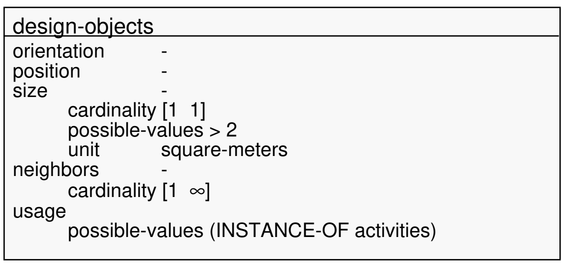

Figure 5: Specifying constraints by using predefined aspects

More complex integrity constraints can be expressed using so-called *demons* and/or *rules*. For example, our application has to check whether an activity specified for a specific room is compatible with previously asserted usage. There are statutes that strictly determine incompatible usage. The architect has, for example, to consider that facilities for both cooking and bathing must not be provided in the same room. For this reason, we define a slot called 'contradictory-activities', which contains all incompatible activities for every activity in the KB. A demon is then attached to the usage slot of every room (see figure 6), which is 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.

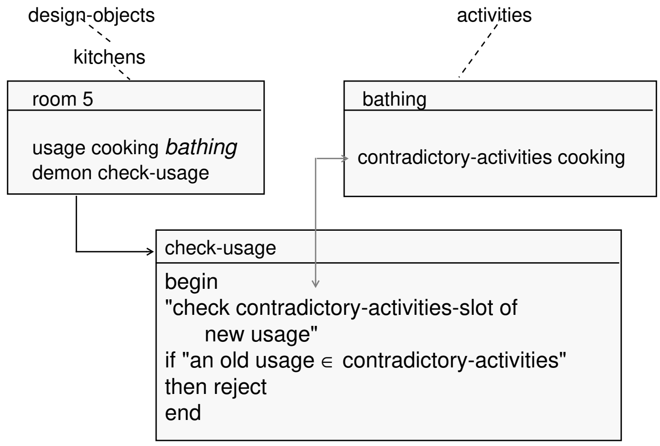

Figure 6: Using demons for checking integrity constraints

Supporting an active system behavior

Demons and rules are also used to achieve an active system behavior. An intelligent CAD system should support and guide the user throughout all design phases, suggesting alternative product designs or possible solutions to his design requirements. A significant part of the knowledge of our architectural design system is represented as rules, which are responsible for the activities of the system. For example, the user may specify activities or usage for the house, but need not relate them to any rooms. This task is then automatically performed by the system. E.g. when the user specifies the activity 'working', the system activates rules associated with the 'working' object in the KB that either 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, or generate a new room (e.g. the office room) for the activity. The location of a room within a certain area (e.g. bathroom, bedroom are parts of the private-area) and the connections between rooms (e.g. a kitchen is connected to the dining-room) can also be inferred as well as the size of a room, which strongly depends on the activities and the number of people using the room. Standard sizes of doors and windows as well as prices of material and equipment are considered in order to keep the financial restrictions given by the user at the beginning of his session.

All the information specified during the session or derived by the application is then used to construct an architectural sketch of the house. The user may accept this sketch or refuse it, which causes the design system to present alternative sketches. Upon additional requirements posed by the user, it has to repeat previous design phases and probably revise decisions made earlier. An explanation facility is also provided for making design decisions plausible to the user.

Extending the integrated product model

A last important issue is related to the notions of extensibility. Since KRISYS does not make any difference between information and meta-information describing the application model, all information is contained in the KB and may be modified. It is therefore easy to keep track of changes in the application environment because the application model may easily be extended or changed. An intelligent CAD system may therefore easily provide user operations for defining new object types and extending or browsing through catalogues since these operations are already supported by KRISYS. As a consequence, our architectural design application may be extended to include the design of many-story houses or other kinds of buildings (e.g. bungalows, villas) in a natural way. Also operations could be provided, that allow the end-user to define a new class of rooms with special properties (sound insulation), that he might require for his house.

3.4 The Implementation Layer

KRISYS makes use of the PRIMA-NDBS [8], also developed at our university, to realize the implementation layer. It offers neutral, yet powerful mechanisms for managing the KB. PRIMA is the PRototype Implementation of the MAD model, whereas the MAD model is an acronym that stands for Molecule Atom Data model [9]. This non-standard data model is an extension to the relational model in that relations are named atom types and tuples are now termed atoms. Furthermore, all relevant relationships between entities, i.e. the foreign-key/primary-key connections between tuples, are explicitly specified in the meta data and represented in the database in a direct and symmetrical way. Therefore, complex objects, called molecules, are definable as higher level objects (being a part of the net of interconnected atoms) and are seen as a structured and coherent set of possibly heterogeneous atoms. From a more general point of view, the MAD model is a direct implementation of the well-known entity-relationship model. The flexibility of the MAD model stems from its dynamic definition and handling of complex objects based on direct and symmetric management of network structures and recursiveness. These concepts enable the appropriate mapping of knowledge structures to data model structures in an effective and straightforward way as described below.

Mapping the knowledge model to the data model

In the previous chapter, we have seen that KOBRA presents all its concepts for building a KB well integrated in its central construct: the schema. From a structural point of view, one may observe that the schema (cf. figure 2) is composed of attributes which, in turn, consist of aspects, suggesting a MAD schema that contains three atom types (corresponding respectively to 'schema', 'attribute', and 'aspects') connected via the references, i.e. relationships, has_attributes and has_aspects. The MAD schema diagram in figure 7 shows a graphical view of atom types (rectangles) and their interconnections (double arrows). Here, aspect specifications are shared between several attributes preventing redundancy (especially in case of inherited attributes). We explicitly exploit the capability of the MAD model to handle network structures in a direct and non-redundant manner. The organizational axes of KOBRA, i.e. attributes with abstraction semantics, are not represented as ordinary 'attributes' but as recursive MAD references between the atom type 'schema'. Since for each MAD reference, there always exists a back reference, each organizational axis is represented by a pair of MAD attributes of type reference (shown as hatched double arrows in figure 7).

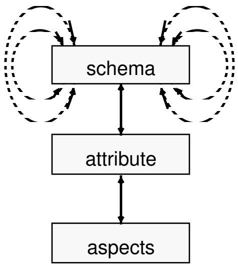

Figure 7: Generic KOBRA schema expressed in terms of the MAD schema diagram

Transformation of operations

On one hand, KOBRA operations can be viewed as general functions to work on the schema objects as a whole. On the other hand, there are also operations to directly manipulate a schema objects' attributes or their aspects. To exemplify our transformation approach, we just present one complex KOBRA operation together with the corresponding MAD statements. Other operation transformations and a more detailed discussion can be found in [9]. The data manipulation language defined for the MAD model is syntactically related to the well-known and easy to understand SQL language developed for relational database systems. Therefore, it is easy to capture the semantics of MAD statements: projection (SELECT) and qualification (WHERE) clause are quite similar to SQL, whereas the FROM-clause of MAD provides the important capability to dynamically express molecules just by writing the corresponding molecule structure, thereby introducing the dynamic complex object facility.

The operation in figure 8, expresses the insertion of a new instance-attribute, to our KB. Since the insertion of an instance-slot (in the example, number_of_windows) provokes its inheritance, it is necessary to obtain the whole generalization hierarchy before inserting the attribute. This is accomplished by the first MAD statement, which retrieves the whole generalization hierarchy beneath the schema object 'rooms'. Such a hierarchy is specified as a recursive complex object starting with the root atom type 'schema' moving on firstly to all subclasses (using the reference type 'has_subclasses' in a recursive manner) and then to their instances (exploiting the reference type 'has_instances'). Additionally, for all schema objects, their attributes are also retrieved (taking the reference type has_attributes). The molecule diagram of this generalization hierarchy is also depicted visualizing the evaluation process of the corresponding molecules as introduced before.

Example: Insertion of the instance-slot 'number_of_windows' to the schema named 'rooms'

1. Retrieval of the whole subclass hierarchy with corresponding instance schemas

```
SELECT ALL
FROM   generalization-hierarchy
       (schema-(.has_attributes-attributes,
                .has_instances-schema.has_attributes-attributes)
       (RECURSIVE: schema.has_subclasses-schema))
WHERE  generalization-hierarchy.schema(root).name='rooms'
```

2. Modification within the KOBRA layer:
   - addition of the instance-slot in the root schema, i.e. the schema named 'rooms'
   - inheritance as instance-slot to all subordinate classes
   - inheritance as own-slot to all subordinate instance schemas

3. Update of the database:

```
MODIFY  generalization_hierarchy
FROM    generalization_hierarchy
```

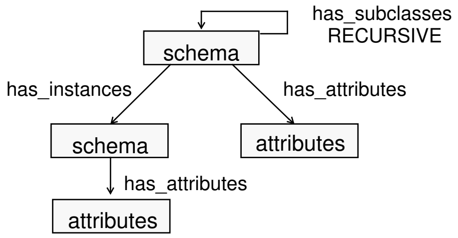

Figure 8: Insertion of an instance-slot expressed in terms of the MAD language

After having modified the hierarchy in the KOBRA layer the complete schema hierarchy is then updated in the DB via a modify statement. This operation changes the specified molecule in accordance to the given hierarchy by inserting atoms not yet stored and modifying the corresponding connections. In this example, it should have become clear that the recursiveness and the dynamic complex-object concept of the MAD model and its language could be extensively exploited to yield a direct and natural transformation of the KOBRA operations.

3.5 Summing up

In KRISYS, the integration of both technologies has been achieved in a natural way, reflected in the stepwise abstraction process realized at each layer. The implementation layer of KRISYS treats knowledge structures simply as a kind of network of complex objects which are consistently, reliably, and efficiently managed. At the engineering layer, these structures obtain semantics, known only by the KOBRA model, remaining, for this reason, outside of the NDBS component. So, the NDBS is not overloaded by application specific aspects, being, therefore, able to concentrate on performance and on other relevant DB aspects. On the other hand, the KOBRA model is not bounded by the semantics provided by the MAD model, being, as a consequence, able to come nearer to the application semantics by offering a rich and powerful spectrum of concepts for modeling.

4. Conclusions

In summary, KRISYS supports powerful and flexible constructs for an accurate representation of real world information:

- Structural object orientation (i.e. the representation of complex objects) is provided, supporting the abstraction concept of aggregation.
- Behavioral object orientation is yielded by the integration of procedural information (methods) into the object description.
- Data orientation is provided by demons, allowing flexible reactions on certain events.
- Intensional information may be represented by rules in order to dynamically derive new data.
- For the organizational structuring of information, abstraction concepts are supported in an integrated fashion, where the same object can act in different roles. The semantics of these concepts are incorporated into the system via built-in reasoning facilities, which guarantee the structural and application-independent semantic integrity of the stored information.
- Application-dependent semantic integrity is maintained explicitly via aspects, demons, and rules.

Furthermore, the architecture and design decisions made (e.g. working-memory system, NDBS usage) are a prerequisite for persistent, reliable, and efficient management of the KB on secondary storage. Therefore, we can conclude that this integration of DB and AI techniques fortifies each other, aiming at a base system - eg. KRISYS - that effectively supports advanced CAD.

Though our approach looks like having removed some main obstacles on the way to better CAD, a lot of problems have not been taken into our consideration (eg. version or design-transaction issues). Of course, it remains very important to gain practical experience with KRISYS in some in-the-field applications. Therefore, we currently work on applications in the areas of diagnoses, planning, and design that will offer more insight into modeling and efficiency requirements of advanced CAD systems.

References

[1] Pham, D.T.: Expert Systems in Engineering, Springer-Verlag, 1988.

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

[3] Mattos, N.: KRISYS - A Multi-layered Prototype KBMS Supporting Knowledge Independence, in: Proc. Int. Computer Science Conf. - Artificial Intelligence: Theorie and Applications, Hong Kong, Dec. 1988, pp. 31-38.

[4] Shah, J.J., Rogers, M.T.: Expert form feature modelling shell, in: Computer-Aided Design, Vol. 20, No. 9, November 1988, pp. 515-524.

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

[6] Härder, T. (ed.): The PRIMA Project Design and Implementation of a Non-Standard Database System, SFB 124 Research Report No. 26/88, University of Kaiserslautern, Kaiserslautern, 1988.

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

[8] Härder, T., Meyer-Wegener, K., Mitschang, B., Sikeler, A.: PRIMA - A DBMS Prototype Supporting Engineering Applications, in: Proc. 13th Conf. Brighton, UK, 1987, pp. 433-442.

[9] Mitschang, B.: A Molecule-Atom Data Model for Non-Standard Applications - Requirements, Data Model Design, and Implemlentation Concepts (in German), IFB 185, Springer-Verlag, Berlin, 1988.