In: Proc. of Int. Workshop on Theory and Practice of Geometric Modeling, Blaubeuren, Germany, 1988.

Information Structures and Database Support for Solid Modeling

Th. Härder, Ch. Hübel, B. Mitschang
University Kaiserslautern, West Germany

Abstract

The question we are going to investigate is how to map solid representations to DB structures and how to process this information efficiently. Starting from analytical representations based on analytical methods we discuss the use of constructive solid geometry and boundary representation models with various refinements. Furthermore, additional submodels (organizational, technological, physical) are considered in order to obtain an overall product model. This model representing all important aspects of a complex design object may serve to derive special object representations needed by existing engineering tools or by mathematical methods (e.g. finite elements).

Today's DBMS are unable to meet the increasing requirements of engineering applications that would prefer to use a DBMS. To alter this situation, a new generation of DBMS architectures tailored to the demands of such enhanced applications have to be developed. As a consequence, the flexibility and expressiveness of data models as well as the handling of application objects must be greatly improved before interactive design work can be supported. We outline our data model concepts and architectural decisions to provide effective data management support. Our DBMS architecture consists of a neutral kernel part running on a server machine and an application layer tailored to solid modeling tasks which together with the application, i.e. the solid modeler is allocated to the workstation .

1. Benefits of data-driven CAD integration

During the past decade, a large number of CAD programs was developed, each serving a specific task in a complex engineering environment. Data management and consistency requirements were handled by the individual programs thereby providing efficient solutions tailored to the particular needs. As a consequence, this approach has led to systems where program and data structures are intrinsically combined and where integrity constraints are directly incorporated in the program code.

Currently, many research efforts are directed towards the design of integrated engineering systems to avoid the drawbacks of disharmonizing CAD tools put together in a practical application. Integration of existing isolated solutions, however, is very difficult or even impossible; hence, more general ways have to be investigated. One promising approach is the integration of all tools and tasks via the underlying data structures. The key idea requires a unique and non-redundant representation of each object under consideration to allow for consistent and uniform object handling. Since multiple tools (or applications) refer to the same objects, manipulation of them has to be controlled to prevent inconsistencies. Therefore, all data management functions must be removed from the individual CAD tools: a database management system (DBMS) provides access to all objects in a standardized manner. Of course, the representation of these objects is critical for the capability to integrate new tools as well as for the overall system performance.

The object representation should not be biased towards specific application requirements; it rather should be neutral to meet the modeling needs of different applications equally well. Otherwise, some tools are favoured at the expense of others. On the other hand, performance restrictions may be satisfied for a tool refering to a 'fitting' representation whereas an oblique object representation would cause a high run-time penalty for another tool. For similar reasons, all (or at least many) aspects of such objects have to be represented. Furthermore, these objects must be described explicitly. This information concerning the types and relationships of all design objects is kept in the DB schema. To preserve consistency in a multiple CAD tool environment, it is strictly necessary to make all integrity constraints visible to these programs. Therefore, they have to be specified by some formal language and added to the object descriptions. The DB schema then contains all unique (and, as far as the DBMS view is concerned, complete) object descriptions. It is obligatory for all CAD tools to use only the schema information for object manipulation, and the DBMS's task is to guarantee the corresponding assertions.

After these general representation requirements of integrated information systems we briefly discuss those more specific to integrated engineering systems. In the database field, engineering objects are called complex objects: they exhibit an internal structure, may have additive or complementary object descriptions, and require complex integrity constraints to specify semantic aspects of the corresponding real world entity. Furthermore, they embody behavioral properties from an application point of view. For example, a solid has an inner structure obtained by the aggregation of subsolids as well as a surface. To automatically control certain space relationships, appropriate integrity constraints have to be specified. If the solid represents a screw, an operation 'twist' may be associated with it. Since such object properties are similar in all engineering application domains (e.g. CAD/CAM), the notion of a complex object allows for their integration in a natural way. In practical design work, various relationships spanning application domains have to be utilized in any case. For example, design steps are tightly connected with manufacturing steps in the sense that a workpiece under design has to be frequently controlled to see whether it can be manufactured (using the existing machines and the standard parts available) and whether it meets the specific restrictions of the manufacturer.

Long-term integration requirements may not be limited to the design and production of objects. If we attempt to integrate the business-oriented information of an enterprise, we reach the starting point of a vision called CIM. Although such a concept requires business, manufacturing, and design information, it is quite natural to view this as different aspects and complementary descriptions of complex objects and hence, to use it for a data-driven system integration.

In this paper, we derive and motivate step by step information structures for integrated CAD tools. A particulary challenging application area is solid modeling for the construction of workpieces.

2. Information Modeling and Data Modeling

After having addressed the benefits of data driven CAD integration, we would now like to discuss now how to achieve such an integrated solution. In general, there are four steps towards a domain spanning tool connection via data integration:

- Firstly, one has to make explicit the prevailing information structure in each application domain and in all relevant branches of each domain. In our case, the domains coincide with CAD, CAM, CAP, etc.

and the branches (e.g. in the CAD environment) correspond to the fields of construction, drawing, visualization as well as to calculation (e.g. finite element method, etc.).

- Secondly, all these information structures have to be integrated, that is, redundant information must be eliminated or must be described in an explicit way. In addition, the relationships among the respective application domains and/or the branches have to be realized. On doing this, one obtains the domain spanning information structure which determines the starting point for the next step.
- Thirdly, the delivered information structure has to be mapped onto an appropriate data structure, the so-called database schema.
- In the fourth step, finally, one has to encode application algorithms, which work on this centralized database schema, thereby using the functionalism of the DBMS at hand.

The expected difficulty concerned with step one and two lies mainly on the application side: a deep knowledge in each application domain is required in order to find out and formalize the relevant information. On the other hand, steps three and four determine the problems on the data management side. Indeed, adequate data modeling and effective data processing constitute the most severe challenges for DB research in this area.

In the following, we would like to explain some examples in the environment of solid modeling. We are going to introduce information structures for a number of solid representation schemes. These information structures are illustrated using the well known Entity/Relationship diagrams (ER-diagram) [1]. In the second part, we will outline an integrated information structure, which constitutes the base of a domain spanning representation of design objects. Finally, we will emphasize some problems in modeling and processing the derived data structures based on conventional data models, and we will formulate several requirements for more adequate DBMS support.

2.1 Information Structures for Solid Modeling Schemes

There is a number of conceivable methods for the representation of 3-dimensional objects [2, 3]. Table 1 shows some of the most important ones. According to the actual point of view, each of these solid representation schemes has various advantages or disadvantages. For example, if one wish to calculate common solid properties like volume or center of gravity, you will need other information as in the case of solid visualization or generation of 2-dimensional drawings. In the following, we wish to look at some solid modeling schemes and their inherent information structures.

pure primitive instancing / parameterized solid representation
analytical & semi-analytical representation methods (algebraical based / free formed solids)
cell decomposition / cell enumeration (oct tree representation / grid methods)
boundary solid representation (including several surface and curve representation schemes)
constructive solid geometry

Table 1: Some of the most important solid representation schemes

Pure Primitive Instancing

The first example of solid representation we wish to explain in more detail is called pure primitive instancing. A solid is described by its type and by a list of parameters. Each type is associated with a particular procedure, which is able to interpret the corresponding parameters, and evaluate some of the above mentioned common properties as well as the 2-dimensional image of the type's instanciation. Since these procedures encapsulate the whole information about the corresponding solids, this representation scheme is rather inflexible, although the variation of parameters is allowed. This variation results in similar solids that only have the same complexity, whereas the modeling of more complex solids requires the programming of more complex procedures. Of course, these may use more simple ones, but this fact cannot be explicitly reflected in the information structure. Fig. 1a shows the related ER-diagram, which bears the information about type definition and parameterization. The procedural information is suggested by the doted cloud associated to the solid type entity. In addition, Fig. 1b illustrates an example of an parameterized workpiece.

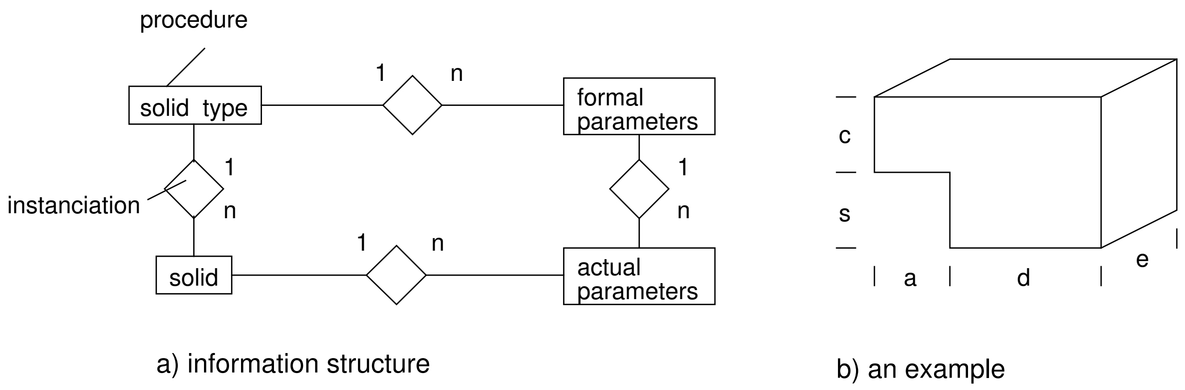
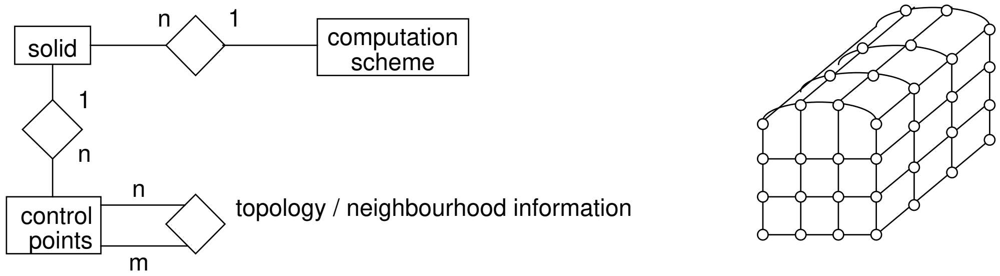


Fig. 1: Pure primitive instancing

Analytical/Semi-Analytical Representation Methods

As the next example, we are going to look at the analytical/semi-analytical representation methods, which determine the points that belong to a particular solid by a set of algebraical/geometrical coefficients. The semi-analytical methods provide the division of a given solid into subsolids that can be represented in an analytical way. Examples are Bezier or B-spline solids that are represented by a set of tricubic, analytically described solids. In this case, the characteristic coefficients are given by so-called control points. The computation scheme of the analytical representation methods plays a similar role to the procedures in the case of pure primitive instancing. In comparison to this approach, the analytical representation method is more formal, that means, the computation scheme is not encoded as a procedure that is written in a conventional programming language, but it is described as a set of parametrical matrices. The parameters for matrix generation are determined by the algebraical/geometrical coefficients (e.g. control points). The derived matrix defines some kind of enumeration of all solid points: multiplying the matrix with a three-valued vector whose components vary in a fixed interval, you can evaluate successively all desired solid points.

The modeling of solids is enabled by the modification of control points. However, the generation of more complex solids using more simple ones is not reflected immediately in the representation scheme. The corresponding ER-diagram is sketched in Fig. 2. The neighbourhood relationship represents the topology among control points (cf. the example in Fig. 2), which plays a relevant role for the matrix evaluation.



a) information structure

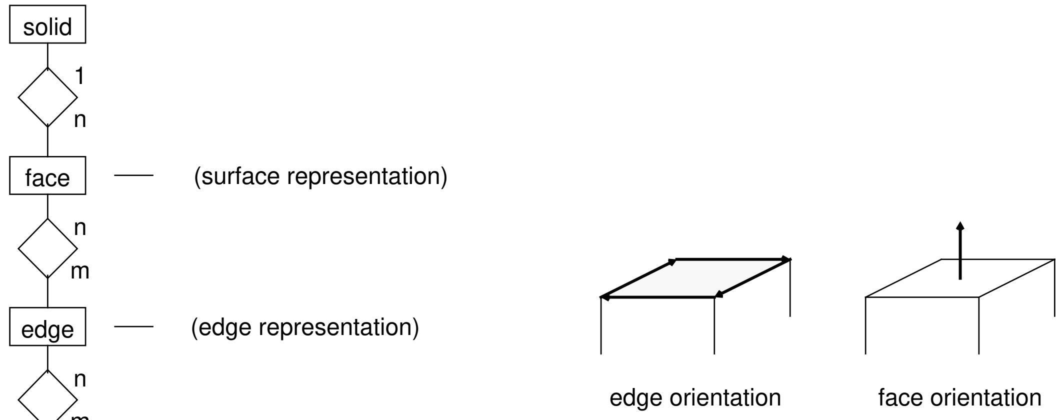
b) an example

Fig. 2: Semianalytical representation method using control points

Boundary Representation Scheme

An other example of a commonly used solid representation scheme is called boundary representation (BREP). The description of a solid is determined by its boundary faces. These have to be described by their surrounding edges, and the edges, finally, can be represented by their defining vertexes. This topological representation hierarchy is associated with a geometrical representation part. So, each face is related to surface information that may be represented via a surface representation scheme. Similar to the solid representation scheme, several methods for surface representation exist such as analytical methods and parameterized surface prototyping, etc. Accordingly, the geometrical part related to edges is organized as a curve representation scheme. Finally, the vertexes are associated to the points that are represented by their coordinates.

The BREP information structure is often enriched by additional information in order to simplify the operations that work on the BREP scheme. The extension commonly concerns the context/neighbourhood information of geometrical objects. The loop information gives an important example: the inside of a particular face is defined by the orientation of its surrounding edges (the face's inside is to be on the right side of the related edges (cf. Fig. 3)). In the same way, the orientation of a face (represented by the direction of a normal vector e.g.) can indicate the inside of the addressed solid.



a) information structure

b) making plausible the additional information

Fig. 3: Boundary Representation Scheme (BREP)

The modeling of solids (represented by its boundary) is achieved by the modification of the basic elements: points/vertexes, curves/edges, and surfaces/faces. This fact introduces some problems, because the modification of a single basic element may result in an inconsistent solid representation (e.g. hanging points, curves, or surfaces). So, a couple of basic manipulations is required to transform a consistent BREP into another consistent one. But, this fact cannot be regarded by the BREP information structure. Similarly, constructive issues like the successive generation of a solid using less complex ones, are not directly reflected. On the other hand, the BREP scheme seems to be a rather expressive solid representation method, because it offers the capability of a heterogeneous surface representation. For this reason, all solids that consist of representable surfaces are representable as well.

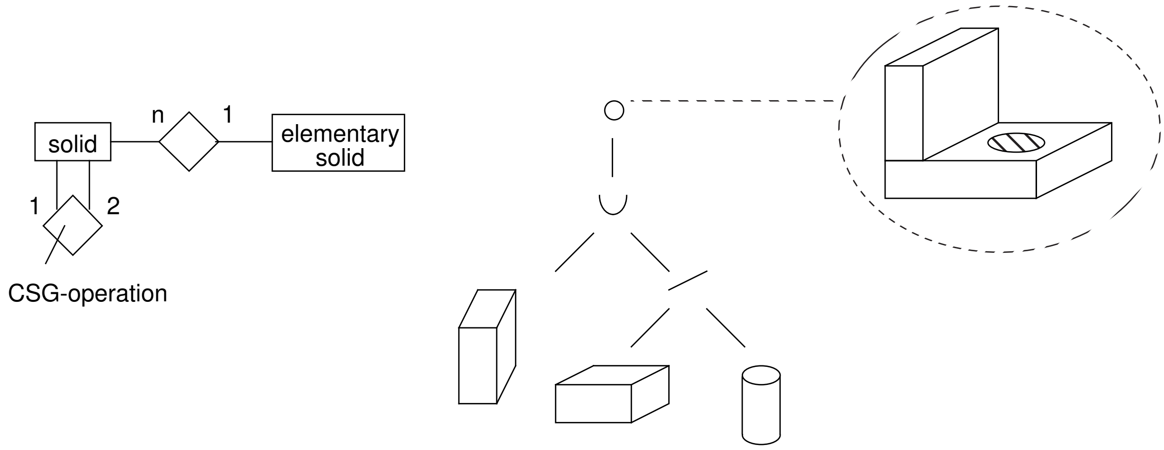
Restricting the class of representable solids to the polygonal solids yields the simplest case of BREP, and with this, you simplify the related operational efforts significantly. The curves are defined to be straight lines, and the surfaces are planar and bounded by a list of straight lines. So, all the geometrical information may be encapsulated by the points. Fig. 3 shows the ER-diagram for this simple case, and it illustrates the semantics of the scheme extensions which are outlined above.

Constructive Solid Geometry

Constructive solid geometry (CSG) marks the key towards more constructive aspects in solid representation. The CSG scheme is organized as a tree, the CSG tree. The nodes of this tree coincide with operations chosen from a couple of so-called regular operations (e.g. union, intersection, difference, translation, rotation ,as well as scaling). The leaf nodes correspond to primitive solid elements chosen from a set of elementary solids (e.g. tetrahedron, cube, cylinder, etc.). The expressiveness of the CSG scheme depends extremely on the variety of this set. By only using a cube, it results in a quite limited class of representable solids. The ER-diagram of the CSG scheme is shown in Fig. 4.

In order to visualize a particular solid or to calculate some geometrical properties, an evaluation process must be performed. This process has often a recursive nature, that is, the property of a particular solid (represented by a CSG tree) can be evaluated easily if the properties of all subordinate solids (represented by their corresponding CSG subtrees) have already been evaluated.

The modeling of CSG solids is achieved easily by extensions to the corresponding CSG tree, i.e. by simple tree operations. So, the modification facility is restricted to the regular CSG operations, which are defined to be consistent and which result in consistent solid representations. Therefore, starting with a set of consistent elementary solids, each CSG tree represents a real and consistent solid.



a) information structure b) an example

Fig. 4: Constructive Solid Geometry (CSG)

From our point of view, the solid's geometry does not constitute the most important issue in the CSG representation, but the solid's construction does. The CSG tree embodies the path of construction, i.e. the way in which the solid may be generated using the elementary solids. For this reason, a CSG tree is often called a structure tree, too. The constructual information seems to be rather suitable for the derivation of structural properties such as the usage of parts in other parts (i.e. some kind of mechanical assembly or bill of material), and the connection with physical or technological information such as issues involving manufacture etc. For example, the subtraction of a particular cylinder can be associated with a real manufactural step (drilling operation) in the CAM environment. Therefore, the CSG representation scheme offers a useful gateway to other domains allowing an easy domain integration [4].

2.2 Integration of Several Solid Modeling Aspects

One of the most important motivations which induced us to think about integration in solid modeling is the concept of an overall and domain spanning model that includes all the heterogeneous aspects which will occur during construction and production of complex design objects. Commonly, this concept is embodied by the term 'product model'. Fig. 5 illustrates a conceivable product model comprising the information outlined in 2.1. The overall model is divided into several submodels (physical, geometrical, structural, and technological model). The structural and the geometrical part refer to the different principles of the different solid representation schemes (e.g. type-definition/instancation, parameterization, geometry vs. topology, constructive representation, etc.).

The structural model is refined by a combination of pure primitive instancing and CSG representation. So, we have solid types and formal parameters which are related to solid instances and actual parameters. The relationship between types and instances are determined by the terms 'instanciation' and 'construction', i.e. they reflect the creation of new instances of a given type, and the generation/definition of a new type using multiple subordinated instances. The relationship between solid instances allows for the CSG-oriented representation bearing the actual CSG operations. Since multiple solid instances of the same type may have the same geometry, the solid type entity is related to the geometrical information.

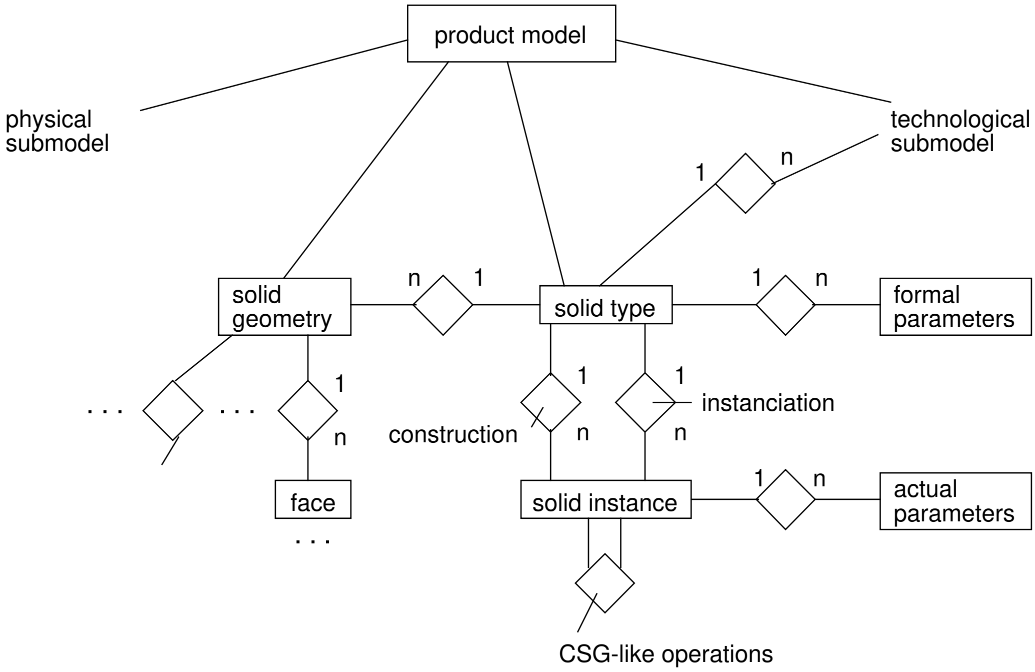


Fig. 5: A domain spanning information structure

The geometrical model is proposed to be boundary based, because the BREP scheme offers a great variety for the description of geometrical aspects. However, since we have an explicit relationship between the structure and the geometry of a solid, the actual kind of geometrical representation does not influence the structural issues. The equivalent among different representations of a single geometrical object may be described explicitly via a unique structural entity solid type. Combining different geometrical representations to the same solid type presents the opportunity to maintain alternative geometrical representations for special purposes (e.g. octtree, or grid representation for finite element calculations).

A profitable linkage to the technological model should be based on a connection to the structured model using the solid type information. As already mentioned above, the CSG oriented solid representation allows the association with manufactural processes. Furthermore, this seems to be a good starting point for feature-based construction concepts that combine design and manufactural issues for manufacturing purposes [4].

2.3 Aspects of Data Modeling and Data Processing

At the beginning of this chapter, we have outlined four steps towards data driven tool integration. Having applied step one (i.e. explicit description of the prevailing information) and step two (i.e. integration and embedding into a domain spanning information structure) to the area of solid modeling, now we would like to consider a conceivable realization of the steps three and four.

The mapping of the given information structure onto a data structure capturing the same information may be carried out in an automatic/semi-automatic way. Using, for example, the well known relational data model to describe the desired data structures, one can perform a simple transformation:

- Map the entities onto relations; take over the entity information by defining corresponding relational attributes; determine one attribute or a couple of attributes as primary key, or introduce a special identifier attribute.
- Map the complex relationships (of type n:m) onto relations, whose primary key is composed of the primary keys of both participating relations.
- Map the other relationships (of type 1:n or 1:1) onto a key/foreign-key association. These connections are realized by introducing the primary key of one relation as one attribute (in the sense as a foreign key) in the other relation.

Applying this procedure to the ER diagram of the simple BREP scheme (cf. Fig. 3), we will get the relational database schema shown in Fig. 6. Investigating the database schema in a little bit more detail reveals some initial problems. In order to identify the tuples, several relations have been extended by additional identifier attributes, since the remaining attributes do not identify a tuple. For example, the tuple of the vertex/point relation are not identified by their coordinates (multiple tuples with the same coordinates in the BREP of different solids could exist). The identifier attributes which have been introduced must be handled explicitly by the application program or the user himself. This may be quite troublesome because of the great number of tuples.

Solid	(solid_id, ...)
Face/Surface	(face_id, solid_id, normal x, normal y, normal z, ...)
FEconnection	(face_id, edge_id, orientation)
Edge/Curve	(edge_id, ...)
EVconnection	(edge_id, vertex_id, position)
Vertex/Point	(vertex_id, coord x, coord y, coord z, ...)

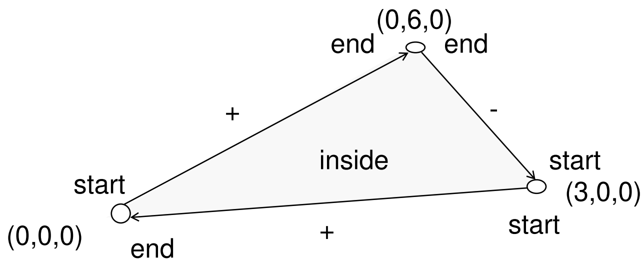
Fig. 6: Relational DB-schema for the polygonal BREP scheme

A further, more important problem concerns the application algorithms that are to work on the centralized database using the access functions of a DBMS. These algorithms often require a great number of mutual related tuples of various relations. In modeling a CSG representation, for example, an application algorithm for solid visualization uses the CSG tree (i.e. a recursive structure) as a whole. In the same way, a BREP modeling algorithm may need all the information about a particular face (i.e. a network-like structure containing the face/surface tuple, all the related edge/curve tuples as well as the associated vertex/point tuples) in order to perform a test operation that examines whether or not a given point lies in a face. In other words, the unit of data that is used by an application algorithm during a unit of processing is commonly heterogeneous and complexly structured. However, such a unit of data cannot be defined and cannot be handled as a whole by means of the relational data model (as well as any other conventional one).

```
SELECT ALL
FROM Face/Surface, Edge/Curve, Vertex/Point
WHERE  Face/Surface.face_id = 4711 AND
       FEconnection.face_id = Face/Surface.face_id AND
       Edge/Curve.edge_id = FEconnection.edge_id AND
       EVconnection.edge_id = Edge/Curve.edge_id AND
       Vertex/Point.vertex_id = EVconnection.vertex_id
```

a) sample query

temp_relation	(face_id,normal x,normal y,normal z, ... ,orientation,edge_id, ... ,position,vertex_id,coord x,coord y,coord z)
	4711 , 0 , 0 , 1 , ... , + , 17 , ... , start , 5 , 0 , 0 , 0
	4711 , 0 , 0 , 1 , ... , + , 17 , ... , end , 6 , 0 , 6 , 0
	4711 , 0 , 0 , 1 , ... , - , 18 , ... , end , 6 , 0 , 6 , 0
	4711 , 0 , 0 , 1 , ... , - , 18 , ... , start , 7 , 3 , 3 , 0
	4711 , 0 , 0 , 1 , ... , + , 19 , ... , start , 7 , 3 , 3 , 0



(The position attribute marks the role of a point with respect to a particular edge. Similary, the orientation attribute indicate the direction of an edge with respect to a particular face)

b) sample result data

Fig. 7: Evaluation of face/surface information

Of course, the retrieval of the involved tuples may be conceivable in some cases. Fig. 7 illustrates an example: the desired unit of data is determined by the boundary face of a given solid and may be retrieved by evaluation of the outlined select statement. The result data is represented as a set of homogeneous, flat tuples constituting a temporary relation. Commonly, this relation can be processed by application programs in a sequential way. This processing scheme is not very suitable, since application algorithms (especially in the solid modeling area) require repeatedly some kind of random access to the data that is covered by more than one tuple. Additionally, if the unit of data was evaluated by relational join operations (cf. example in Fig. 7), the modification of the result data is not addressed directly by conventional DBMS interfaces. In a few other cases, even the retrieval of the required unit of data is not conceivable in a direct way. Recursive structures like the CSG tree, for example, have to be evaluated by the application programs performing recursive/iterative procedures.

So far, we have discussed the structural aspects of the objects in solid modeling. Apart from these, a number of procedural properties that are involved in the representation schemes exist, too. For example, pure primitive instancing is mainly based on the procedural representation of the solids' properties. Similarly, the operations associated with a solid representation scheme often determine the semantics and the consistency of a particular representation. Therefore, it should also be feasible to map these procedural/operational issues into the DBMS environment. Unfortunately, there is no opportunity to do so using conventional DBMS.

Summarizing and generalizing the considerations addressed in this section, we are able to highlight the essential aspects from the database point of view: the concepts for data modeling and data processing (the data models as well as their programming interfaces) that are commonly offered by conventional DBMS have proved to be inadequate. The prevailing application objects represented as units of data cannot be handled in their entirety, what finally results in a significant loss of performance. After all, this is the main reason why conventional DBMS are not generally applied for data management tasks in solid modeling as well as any other engineering application area.

In the following, we would like to light the way towards a more adequate and more efficient DBMS support, in particular for engineering applications.

3. Database Support

Satisfactory data management in the sketched application areas above requires adequate data modeling facilities as well as effective architectural and implementation issues for efficient data processing. In the following, we firstly discuss the necessary data model issues suggested in section 2.3. Then we concentrate on suitable architectural concepts for mapping the application objects and on reasonable implementation concepts. Last of all, we take a look at the most important concepts providing the efficiency which is urgently needed.

3.1 Data Model Issues

From the database point of view the engineering objects are known as complex objects, thereby serving for a natural integration of the different object properties. In this section, we wish to identify the data model properties most useful for the support of complex objects. It was already pointed out that complex objects exhibit an internal structure. A closer look at this structure and the structured components reveals the following characteristics (cf. section 2.3 and [6]):

- network-like object structure
  Complex objects may share common subobjects, i.e. the same components are part of different objects. These objects are called non-disjoint. Such a component sharing is caused by a complex (n:m) relationship. In contrast to this, the other relationship types (1:1, 1:n) lead to disjoint objects with disjoint components.
- recursive object structure
  Adequate and natural object handling requires recursive definition as well as manipulation of object structures. Complex objects are called recursive if they may be composed of objects of the same type; otherwise, they are called non-recursive.
- dynamic object views
  Depending on the current need of the application (i.e. the level of abstraction and the phase of processing), a variety of different views of the object is desirable.

The most important conclusion drawn from this characterization is related to the handling of relationships. Dynamic object definition requires flexible representation and derivation of the relationships either at run time or supported by previously derived data caching techniques. Therefore, symmetric relationship representations seem to be mandatory, which allow for the desired freedom in building up dynamically defined complex objects. For reasons of efficiency, even complex relationships should be represented directly (without auxiliary constructs, e.g. relationship relations) and should support bidirectional traversal and use of objects.

All data model issues discussed so far are summarized in the term structural object orientation, because they contribute to the internal structure of complex objects. The consistency of this complex object structure is obtained by means of structural integrity constraints; these are maintained for each specified relationship type by observing referential integrity restrictions, type definitions and cardinality ranges.

Beyond their internal structure, complex objects embody, from an application point of view [7], behavioral properties and complex integrity constraints to specify the semantic aspects of the corresponding real world entity in sufficient detail. The following concepts model this external behavior of complex objects:

- abstraction concepts
  Semantic enhancement to the above mentioned syntactic relationships are provided by the abstraction concepts of classification, generalization, association and aggregation. In [8] the semantic expressiveness gained by means of these abstractions is extensively described.
- operational capabilities
  Operational expressiveness implies the explicit specification of sufficient application semantics using operational descriptions and corresponding behavioral integrity constraints.

Such a semantically enriched complex object concept enables appropriate forms of data abstraction and encapsulation relieving the application from the burden of maintaining intricate object representations and checking of complex integrity constraints. As a consequence, we achieve adequate operational support at the application interface (also called application model interface), also termed behavioral object orientation.

3.2 Architectural Issues

The question we would like to address now is how the above sketched data model issues are embedded in a suitable system architecture. Fig. 8 illustrates the software architecture allowing for an adequate mapping of the application objects. As a key idea, the overall system architecture is divided into two parts:

- The so-called NDBS-kernel offers application-independent data management functions embodying structural object orientation.
- The application layer (AL) provides application-specific support substantiating behavioral object orientation as well as the handling of most semantic issues.

On the one hand, the advantage of this approach is the application orientation of the AL offering the so-called application model interface, i.e. the desired application objects and operations. On the other hand, the kernel unifies all neutral data representation, structuring, and access facilities in an efficient manner. Thus, the kernel provides an application-independent data model (or data model implementation) as a basis for the semantically enriched application-specific models within the AL. The mapping of the application model interface to the data model interface is specific for each application domain. Hence, different AL exist which offer tailored interfaces.

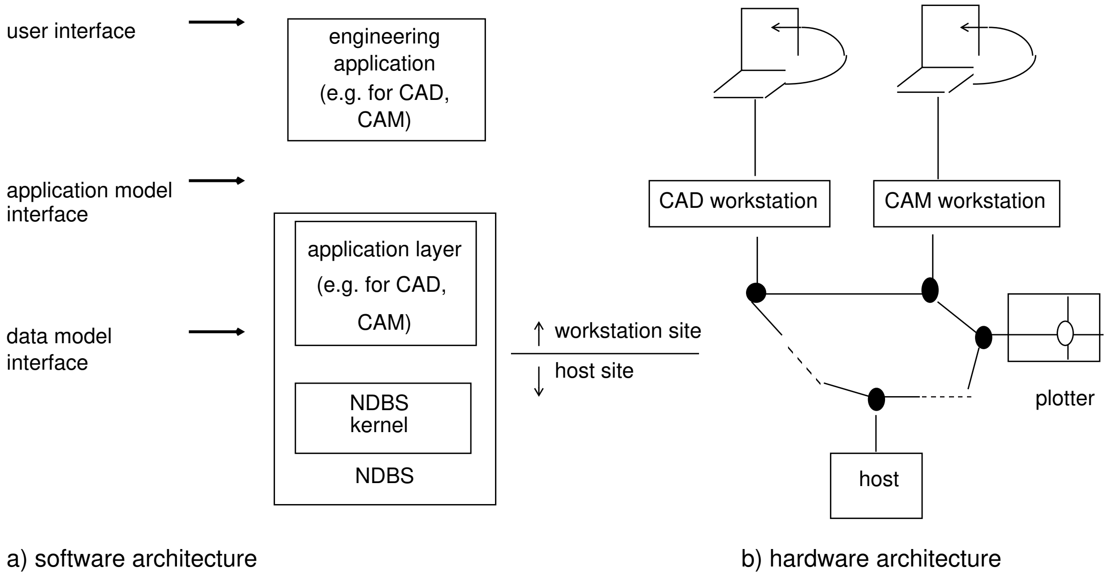


Fig. 8: Architectural of issues NDBS-based engineering information systems

The specific nature of engineering applications leads to a workstation-based hardware architecture (Fig. 8) with decentralized and autonomous processors, coupled via a local area network, and having access to central services (e.g. data management, plotting, and printing facilities). This workstation-oriented processing could be effectively enhanced by delegating the AL together with the entire engineering application to the workstation and the NDBS kernel to the host system that handles all database requests.

3.3 Efficiency Issues

The homogeneity between hardware and software architecture (cf. Fig. 8) offers some important advantages influencing the overall performance:

- A set-oriented language together with the proposed structural object orientation available at the data model interface allows for the requesting of sets of structured objects, thereby minimizing the communication overhead between workstation and host.
- The design objects are temporarily stored in the workstation where they are organized in a special main memory structure called object buffer which offers fast operational access. Hence, the locality of data references during major processing steps could be kept and exploited within the workstation site.
- For recovery purposes and for saving particular design states, copies of the design objects may be preserved in private database allocated to each workstation.
- Distributed processing allows for mutual failure masking that yields an increased application autonomy, i.e. failure on the workstation should not bother the server and vice versa.

All these concepts are embedded into an overall processing model provided by the AL that is extensively discussed in [9]. Furthermore, this processing model introduces the basic concepts of a design transaction established by a checkout/checkin mechanism guaranteeing concurrency isolation for the design objects at hand. This processing model avoids the duplication of work by distributing the work to do among workstations and server. Thus, increasing the workstation capacity promises true performance gains.

Last but not least, it should be mentioned that there are a number of appropriate measures to enhance the overall performance of the kernel, e.g. special clustering techniques at the access path and storage structure level or various caching techniques materializing complex objects, thus avoiding their dynamic derivation [10,11].

4. Conclusions

Only the synopsis of a number of quite different aspects

- starting with the modeling issues of structural object orientation and behavioral object orientation in addition to the semantic organizational aspects of the abstraction concepts, and
- working up to the architectural concepts of the kernel architecture and its homogeneity to the work-station-based hardware architecture,
- in combination with some performance determining implementation issues (e.g. processing model, object buffer) aiming at minimal data transfer and communication overhead

provides the basis for a DBMS supported interactive design work in distributed environments [12]. The prototype database system PRIMA-NDBS [10] tries to incorporate all these aspects that should fortify each other in a fruitful way to provide effective DB-support for engineering applications.

At the moment, we are performing broad 'in-the-field' validation of these system concepts. Initial practical experience in VLSI design and 3D modeling has confirmed our design decisions concerning data model and system architecture. The elapsed time could be reduced by a factor of 50-100 than compared to our first approach using only a commerically available database system. Optimization and further adjustment of architectural concepts may further improve the overall performance.

On the other hand, we are looking for a modeling facility to describe the specific application models within the AL. These models are simply called object models that integrate descriptional, organizational, and operational capabilities to obtain a high level of semantic expressiveness. Such an object model is seen as the basis for achieving intelligent CAD, planning and diagnosis support. Here, we consider the integration of ideas which are known from the areas of knowledge representation, active databases, and extensible databases.

5. References

[1] Chen, P.P.: The Entity-Relationship-Model - Toward a Unified View of Data, in: ACM TODS, Vol. 1, No. 1, 1976, pp. 9-36.

[2] Requicha, A.A.G., Voelcker, H.B.: Solid Modeling: A Historical Summery and Contemporary Assessment, in: IEEE Computer Graphics and Applications, Vol. 2, No. 2, March 1982, pp. 9-24.

[3] Mortenson, M.E.: Geometric Modeling, John Wiley & Sons publ., 1985.

[4] Henderson, M.R.: Extraction of Feature Information FROM Three Dimensional CAD Data, PhD Thesis, Purdue University, West Lafayette, Indiana, May 1984.

[5] Date, C.J.: An Intorduction to Database Systems, Vol. 2, Addison-Wesley publ., 1983.

[6] Härder, T.: Non-Standard DBMS for Support of Emerging Applications-Requirement Analysis and Architectural Concepts

[7] Mattos, N.: KRISYS - A Multi-layered Prototype KBMS Supporting Knowledge Independence, in: Proc. Int. Computer Science Conf.-Artificial Intelligence: Theory and Applications, Hongkong, Dec. 1988.

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

[9] Härder, T., Hübel, C., Meyer-Wegener, K., Mitschang, B.: Processing and Transaction Concepts for Cooperation of Engineering Workstations and a Database Server, in: Data & Knowledge Engineering 3 (1988), pp. 87-107.

[10] Härder, T., Meyer-Wegener, K., Mitschang, B., Sikeler, A.: PRIMA - a DBMS Prototype Supporting Engineering Applications, in: Proc. 13th Int. Conf. on Very Large Data Bases, Brighton, UR, 1987, pp. 433-442.

[11] Mitschang, B.: Towards a unified view of design data and knowledge representation, in: Proc. 2nd Int. Conf. on Expert Database Systems, Tysons Corner, VA, 1988, pp. 33-49.

[12] Dittrich, K.R., Dayal, U. (eds.): Proc. Int. Workshop on Object-Oriented Database System, Pacific Grove, 1986.

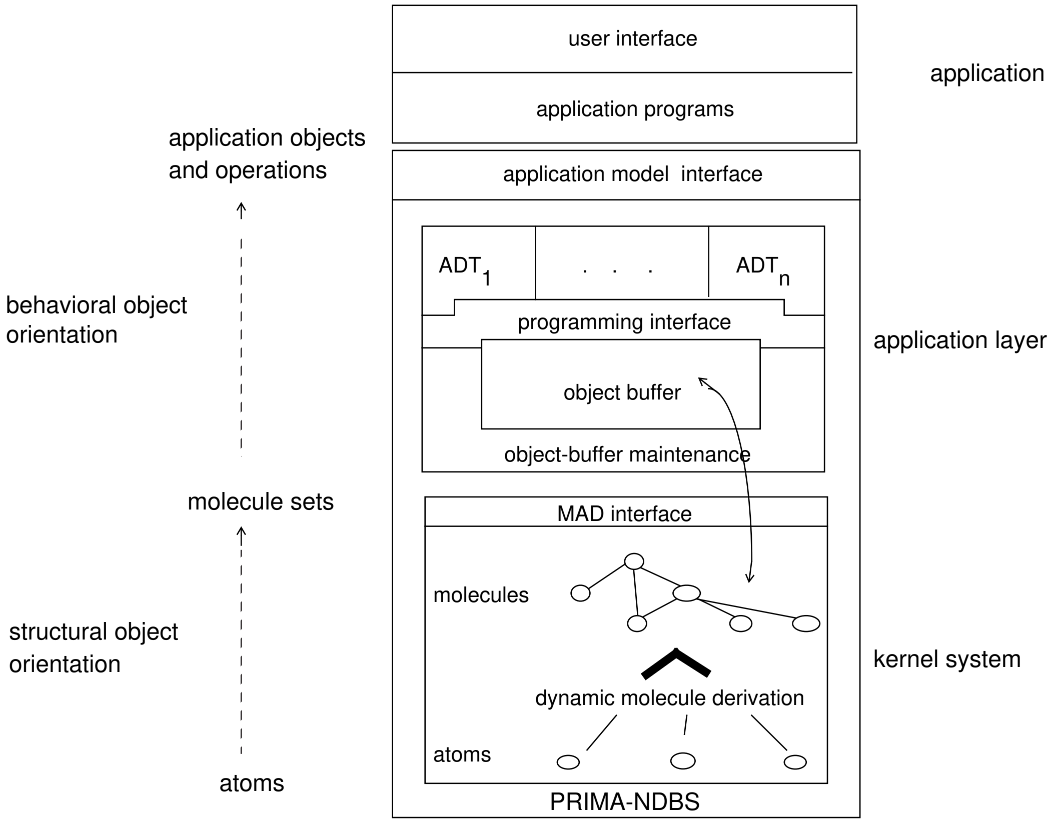


*Fig.*9: The PRIMA-NDBS architecture and its provision of object orientation