*Proceedings of the Tenth International IFIP TC5*
*WG-5.2; WG-5.3 Conference*
*PROLAMAT 1998*

# Database support for concurrent digital mock–up

*S. Berchtold, H.–P. Kriegel, M. Pötke*

*Institute for Computer Science,*
*University of Munich*

*Oettingenstr. 67, 80538 München, Germany,*
*Phone +49-89-2178-2190, Fax +49-89-2178-2192,*
*{berchtol, kriegel, poetke}@informatik.uni-muenchen.de*

**Abstract**

This paper presents new techniques to extend digital mock–up applications by state–of–the–art database functionality for concurrent engineering. We propose a database architecture for the efficient management of spatial, structural, and thematic engineering data. This includes methods for representing and indexing CAD files as well as multi–step query processing. Evaluations of the database system show that the proposed query processor performs collision queries on a 2.5 gigabyte CAD database within seconds, plus the time required by the digital mock–up tool.

**Keywords**

**digital mock–up, concurrent engineering, CAD database**

## 1 INTRODUCTION

In this paper, we propose a new technique introducing database technology into the file–based world of CAD and electronic prototyping. This is an important issue because today, hundred thousands of CAD files of a car or a plane may occupy terabytes of distributed secondary and tertiary storage, representing a scenario the file–based organization is obviously inappropriate for. New commercial electronic prototyping tools, however, require easy and efficient access to this data in order to significantly reduce the time and cost of product development (Virtual Mockup Progress, 1996). In the car industry, for example, late engineering changes caused by problems with fit, appearance or shape of parts account for 20–50 percent of die cost (Clark and Fujimoto, 1991). Therefore, DMU (digital mock–up) tools employ sophisticated voxel–based algorithms for a fast and early detection of colliding parts.

Unfortunately, there is no support for concurrent engineering. Rather, these systems are not capable of handling more than a few hundred parts and they require as input a small, well-assembled list of the CAD files to be examined. With the traditional file–based approach, each user has to select these files manually. This can take hours or even days of processing time for the parts may be generated on different CAD systems, spread over many file servers and belong to a variety of users. In a concurrent engineering process several cross-functional project teams may be recruited from many departments, such as engineering, production, quality assurance, or supply. Each team will have to develop its own parts as a contribution to the whole product. However, the team working on section 12B of a jet–plane may not want to bother about the location and the format of *each* single CAD file of the adjacent sections 12A and 12C. In order to do a quick check of fit or appearance, they simply want to download only the *colliding* parts. On the other hand, the Internet is gaining in importance for industrial file exchange. For example, an engineer working in the United States may want to perform collision queries with her latest design for the product of a European customer. What she needs is a fast and comfortable DMU interface to the customer's CAD database having an appropriate data management on the internal level.

Furthermore, without support of structural information, the user has to take care of different variants, versions and multiple instances of parts. To efficiently manage geometric and structural data, we therefore propose an enterprise–wide CAD database giving the cross-functional project teams easy and state–of–the–art access to all relevant parts of the product. Our technique allows a scenario of an integrated, concurrent development process and thus is applicable to most major production companies. In addition, our technique provides all advantages of a database application, such as recovery and concurrency control, when integrated in a commercial database system. We implemented and tested a prototype of the system called DEEP (Database Extension to Electronic Prototyping), combining

 

spatial database and CAD technologies into a unique and complete database solution for electronic prototyping.

## 2 DATABASE SUPPORT FOR THE DIGITAL MOCK–UP

The problem of the DMU may be briefly characterized as follows: *"Given a set of parts, each described by a CAD file and additional structural information such as 'Car model XY consists of drive system and body, where drive system consists of engine and transmission, and so on', determine all metallic parts of the engine colliding with any part of the body. Moreover, take into account that we may have multiple variants of engines and other parts."* Therefore, three categories of data are relevant to this process: geometric data (the CAD files), structural data (the product structures), and thematic data (such as material, temperature or color of parts). A database solution to the problem must efficiently manage all of them.

The basic idea of our technique is to maintain a consistent and up–to–date database for geometric, structural and thematic attributes of all parts, that can be accessed by any engineer involved in the project. To allow fast queries on geometric data, we developed new algorithms for the approximation of triangulated CAD data by a set of bounding boxes. These approximating bounding boxes are stored in an adapted and optimized R*–tree (Beckmann et al., 1990). As our database additionally contains the structural and thematic attributes of the parts, we are able to perform collision and distance queries (as shown in figure 1) for a part or a collection of parts against the whole product within seconds, with full support of structural and thematic properties. As our experiments show, our technique is efficient even for very large volumes of CAD data.

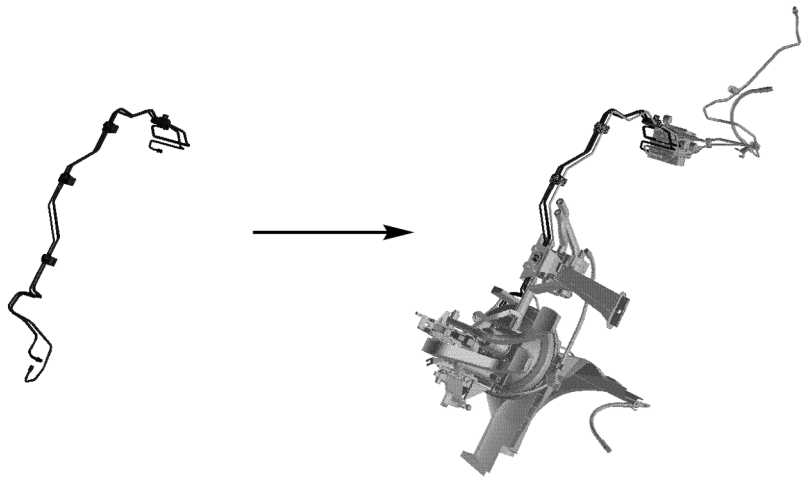

Figure 1 Example for a collision query result

## 2.1 Geometric decomposition and approximation

The most important benefit of CAD files in a modern production environment is their usage within the manufacturing process which requires a very high accuracy (at least 0.2–0.5 mm in the car industry). But to determine a collision of the exhaust system with the rear–axle, we do not need the geometric information to be that precise for the whole calculation. Thus, in order to index the exact spatial data, we decided to store only conservative approximations (space reduction: 1/50–1/350 with a 20 mm precision).

*Triangle–based approximation*

The basic idea of our first approximation algorithm is to decompose the accurate surface triangulations of the CAD parts into partitions of triangles and then approximate each relatively small partition by a minimum bounding box (BB) suitable to be stored in an R*–tree.

In the first step of our algorithm, we transform a flat triangulated representation of each CAD part into a graph having nodes for the triangles and edges to represent topological neighborhood relationships between them. Next, we traverse the graph by applying a best–first search starting from an arbitrary node *s* and using the Euclidean distance to *s* as heuristic function to be minimized. Let $\boldsymbol{v}$ be the sum of the normal vectors of the triangles visited so far (weighted by the area of the triangles) and $m(\boldsymbol{v})$ the coordinate axis which is the best approximation of the direction of $\boldsymbol{v}$. Then, the extension *e* of the triangles along $m(\boldsymbol{v})$ yields a criterion to stop the search and approximate the resulting component by a minimum bounding box (cf. figure 2a). An additional criterion to complete a component is the ratio of the area of the visited triangles to the area of their bounding box, considered in projection along $m(\boldsymbol{v})$. Thereby, the dead space

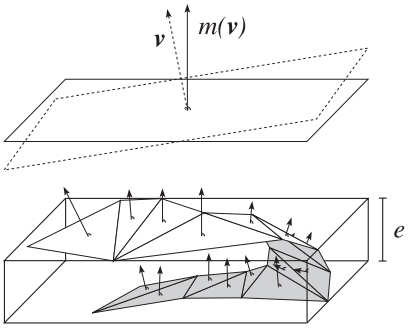


Figure 2a
Triangle-based approximation

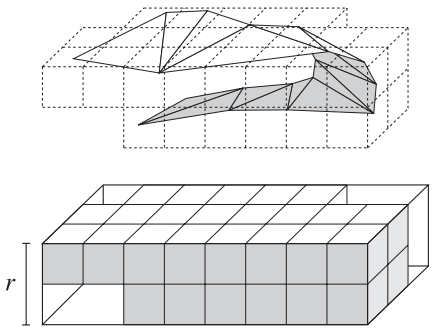


Figure 2b
Voxel-based approximation

in the generated components and their spatial overlap can be controlled. The search is continued from the neighboring nodes of the component. The decomposition and approximation of the CAD file is finished if all nodes have been visited. This algorithm has a worst–case complexity of O($n$ * log $n$), where $n$ is the number of triangles.

*Voxel–based approximation*

Because of common geometric and topological inconsistencies of triangulations, we developed another algorithm, which first approximates the CAD file by a set of voxels and then partitions and approximates the set of voxels with a resolution $r$ in a second step (cf. figure 2b). The algorithm is rather similar to the algorithm operating on the triangulation. But, as the set of voxels is likely to be much smaller than the set of triangles, and geometric and topological properties of voxels are known in advance, this algorithm is about twice as fast and has only O($n$) worst–case time complexity. The generation of a voxel set from a triangulation is computationally very expensive, but most DMU tools require it anyway. With the voxels already computed, our voxel–based approximation is about 40 times faster than the triangle–based version. Some resulting approximations are shown in figures 3a and 3b.

Note, that we may not only approximate triangulations derived from static CAD parts, but dynamic envelopes derived from moving parts as well.

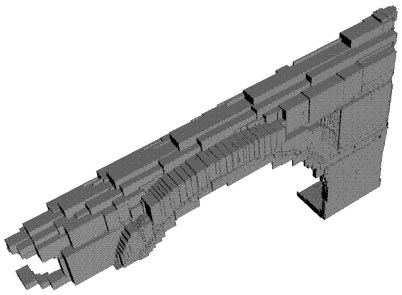

Figure 3a
Triangle-based approximation

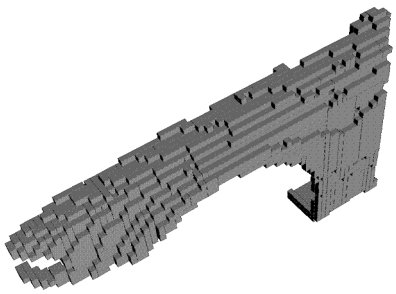

Figure 3b
Voxel-based approximation

## 2.2 Spatial indexing

We collect the files from the various CAD systems of the company, convert them by applying the algorithms described above and create a corresponding R*–tree in the database. For each CAD part the R*-tree contains all the bounding boxes.

The R*–tree is a height–balanced index structure for the management of bounding boxes on secondary storage. It comprises internal nodes each storing an

array of bounding boxes and pointers to sub–nodes, the directory, and data pages that store the actual objects. Each bounding box in an internal node describes the area occupied by the objects in the corresponding sub–tree, however, the bounding boxes are allowed to overlap with each other. Whenever a new object has to be inserted into the tree, the right data page is selected by traversing down the directory. If the new object doesn't fit into the data page, the page is split similar to a B–tree and a new entry in the according internal node is created. This split might be propagated to higher levels of the tree. In order to query the R*–tree, we simply traverse the directory in a top–down matter and decide for each entry in each visited internal node if the sub–node is affected by the query. If so, we recurse into that sub–tree; otherwise, we can ignore the sub–tree.

## 2.3 The DEEP architecture

The architecture of our DEEP system consists of four major components, handling geometric, structural, and thematic data and the corresponding CAD files. These components are depicted in figure 4.

The core data structure of the *geometric component* is the R*–tree. It contains approximated spatial information on all variants and versions of all parts of the product. The problem of dealing with multiple instances of parts is rather similar to the well–treated problem of accessing multiversion data (Bercken and Seeger, 1996). However, the approaches described there handle versions in a temporal way and therefore are not adequate for our variants. Therefore we assign a unique four–byte identifier to each variant and version of a part. Let us refer to this identifier as *surrogate*, for it is a pointer–like representation of a part in the product. As proposed by Brinkhoff et al. (1993), each bounding box in a data page of the R*–tree is labelled with its surrogate to connect the spatial index to the other components of the system.

The *structural component* contains the product structures and their structure tables. Product structures are trees representing the structural data of the product, i.e. information about parts, variants, multiple instances of parts, and their position in the product. They are retrieved from the manufacturer's EDM (engineering data management) database. The structure tables are used to locate the surrogates in the product structures.

In order to manage thematic attributes, we additionally store a set of indexed tables in the *thematic component* of the database system. These may be traditional B–trees using surrogates, too, for the representation of parts. Finally the *CAD component* maintains a table to look up the full path of the CAD file for each surrogate. Optionally we may add a link table to quickly determine the data pages that store the approximating bounding boxes of a part. As the approximation of a part usually spreads over many data pages, this table will get very large.

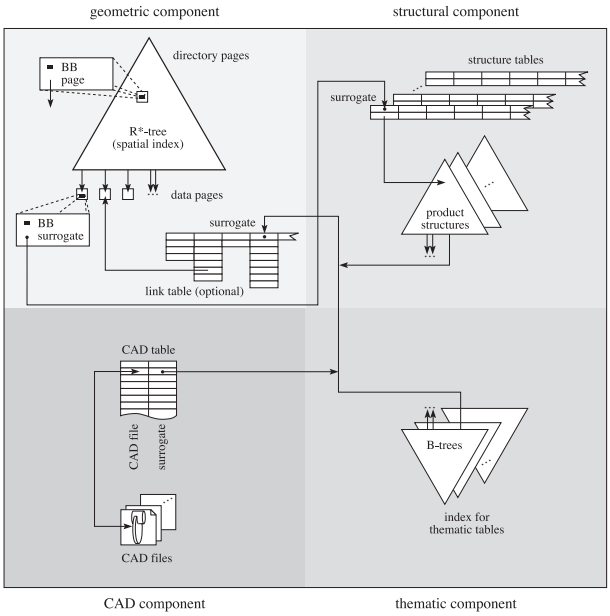


Figure 4 DEEP architecture for concurrent engineering

## 2.4 Query processing

The query processor in our system follows the multi–step query processing paradigm as proposed by Orenstein (1989). To specify the query parts, the user just clicks on the respective items in a product structure (using a graphical interface) or explicitly selects the corresponding CAD files. These can be precomputed kinematic envelopes to allow dynamic examinations as well. As a preprocessing step, we generate approximating bounding boxes of the selected query parts. With the optional link table, we may retrieve these approximations directly from the data pages of the R*–tree (in general this is more efficient). We enlarge the resulting bounding boxes in case of a distance query. Then, we query the R*–tree with the set of bounding boxes and obtain a set of surrogates, representing a superset of the parts fulfilling the geometric query condition. These are the candidates of the geometric filter step. This set is *complete*, since the approximation of CAD files has been designed to be conservative. But it is certainly not *sound*, i.e. some candidates will not fulfil the query condition – they have to be eliminated by the next steps.

In the third step (the structural filter), we reduce the set of geometric candidates by eliminating self collisions and collisions with variants of the query parts, because these collisions will never occur in reality. Additionally, we filter surrogates by enforcing structural query conditions, e.g.: *"report collisions with*

*parts of the V8 engine only"*. In the fourth step of query processing (the thematic filter), we query the thematic tables in the database to check attributes of the resulting surrogates according to thematic query conditions, e.g.: *"report collisions with plastic parts only"*. In the last step (the refinement step), we simply query the CAD table to download the CAD files of the resulting surrogates into an electronic prototyping tool. Note that this step is application specific, depending on the kind of examination the user intends to do. We may use a DMU tool for exact collision tests or for detection of material and temperature conflicts. Or we may use the CAD files of the resulting parts to perform FEM simulations or even build physical prototypes. Finally, we again visualize the result of the query in the product structure.

## 3 EVALUATION RESULTS

To evaluate the practical significance of our technique, we implemented DEEP, a client–server–based prototype of our system (Pötke, 1997 and 1998). The server (containing the proposed architecture including the R*–tree) has been implemented in C++ whereas the client (a graphical query interface operating on product structures) has been implemented in Java. We integrated a commercial DMU tool for the refinement step and tested the whole system holding up to 3300 parts in the database. This corresponds to a volume of 2.5 gigabytes of triangulated CAD data, consisting of more than 50 million facets. The parts have been generated from 165 surface triangulations of high accuracy (0.2–0.5 mm), computed from original CATIA files of a major European car manufacturer. All experiments have been performed on an HP C160 workstation, having 768 Mbytes of main memory and several Gbytes of secondary storage. For the generation of approximations we employed the voxel–based algorithm described above.

The first set of experiments shows how the resolution of the approximating bounding boxes for the CAD files influences the performance of the system. As sample queries we used collision queries for each of the 165 original CAD files without restriction by any additional structural or thematic query condition. Figure 5 shows the average selectivity after the geometric and the structural filter (i.e. the ratio of the number of candidates to the number of 3300 parts stored in the database). We see that with a resolution of 20 mm, we need to download just 4.0% (130 parts) of the CAD database into the DMU tool to do a collision test for the query part. Figure 6 depicts the time consumed by the query processor for the same queries as presented in figure 5. For these experiments the link table option was turned off, but 20 mm voxel representations of the surfaces had been precomputed and persistently stored before. Thus, collision queries with a 4.0% selectivity could be performed in 3.7 seconds on the average. For the refinement step, the average time required to do a DMU on some 130 parts must be added.

Table 1 presents the size of some corresponding R*–trees and the time required to build them.

Figure 7 depicts the processing time of a collision query (geometric and structural filter, 20 mm index resolution) with a sample query part, while increasing the number of parts in the database but maintaining a constant number of variants per part. One can see that the processing time scales logarithmic with increasing number of parts. Figure 8 depicts the average collision query time of the geometric and the structural filter for a part in the database, when increasing only the number of variants per part. The query time has an asymptotically linear scale–up with increasing number of variants, due to high spatial overlap between variants of the same part.

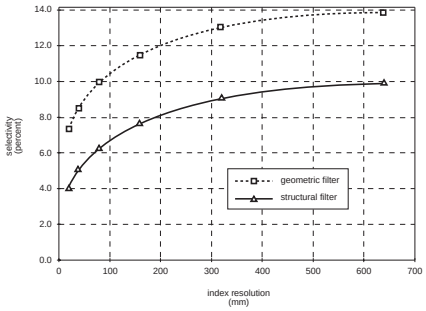


Figure 5
Selectivity of the geometric and structural filter

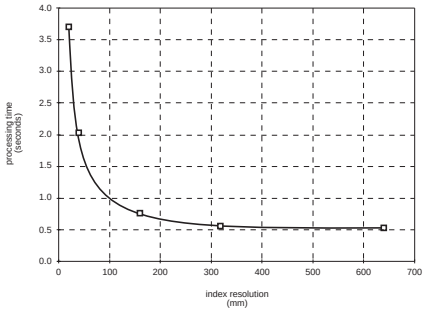


Figure 6
Processing time of the geometric and structural filter

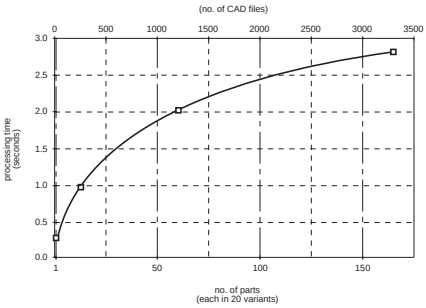


Figure 7
Increasing number of parts

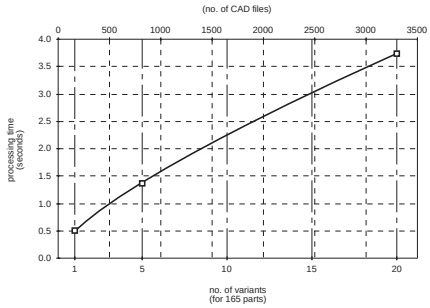


Figure 8
Increasing number of variants

Table 1 Size and creation time for the spatial index

| *Index resolution* | *Size (Mbytes)* | *Creation time (hours)* |
|---|---|---|
| 20 mm | 9.21 | 6.4 |
| 40 mm | 2.25 | 0.5 |
| 80 mm | 0.63 | 0.2 |
| 160 mm | 0.20 | 0.2 |

## 4 CONCLUSION

In this paper, we proposed a database architecture for the efficient management of spatial, structural, and thematic engineering data. We demonstrated that database technology is required to handle the huge amount of distributed data. Our architecture allows a comfortable and fast interaction of concurrent users. We developed a query processor and showed its efficiency by evaluating its performance on very large amounts of data. We would like to emphasize that our database approach reduces hours or even days of processing time in the traditional file–based approach to the order of seconds. Moreover, all engineers share a single common source of CAD files for digital mock–up which significantly reduces the communication effort within and between the project teams.

For the future, we plan to integrate DEEP into one of the commercial object–relational database systems, which support R–trees as part of a spatial data module. Furthermore, we intend to investigate the integration of DEEP with novel CAD applications such as similarity search of parts (Berchtold, 1997).

## 5 REFERENCES

Beckmann N., Kriegel H.–P., Schneider R. and Seeger B. (1990) The R*–tree: An Efficient and Robust Access Method for Points and Rectangles. Proceedings ACM SIGMOD, **1990,** 322–31.

Berchtold S. (1997) Geometry-based search of similar parts. (in german), Ph.D. thesis, Shaker Verlag.

Bercken v.d.J., Seeger B. (1996) Query Processing Techniques for Multiversion Access Methods. Proceedings 22nd Conference on Very Large Databases VLDB, **1996**, 168–80.

Brinkhoff T., Horn H., Kriegel H.–P., Schneider R. (1993) A Storage and Access Architecture for Efficient Query Processing in Spatial Database Systems. Proceedings 3rd SSD, in *Advances in Spatial Databases* (ed. D. Abel, B.C. Ooi), Lecture Notes in Computer Science, **692**, 357–75.

Clark K.B. and Fujimoto T. (1991) Product Development Performance – Strategy, Organisation, and Management in the World Auto Industry. Harvard Business School Press, Boston, Mass.

Orenstein J.A. (1989) Redundancy in Spatial Databases. Proceedings ACM SIGMOD, **1989,** 294–305.

Pötke M. (1997) Database Extension to Electronic Prototyping. Advanced student work, University of Munich.

Pötke M. (1998) Database support for digital mock–up in mechanical engineering. (in german), Diploma thesis, University of Munich.

Virtual Mockup Progress (1996). CAD Report, Dec. 1996, CAD/CAM Publishing Inc.

## 6 BIOGRAPHY

*Stefan Berchtold* is currently working in the area of query processing and data mining in high-dimensional data spaces. His major research contributions were in the area of high-dimensional index structures and similarity search in multimedia and CAD databases. He received an MS degree in Computer Science from the Technical University of Munich in 1993 and a PhD in Computer Science in 1997 from the University of Munich, Germany. Currently, he is with the AT&T Laboratories in Florham Park, USA.

*Hans–Peter Kriegel* is a full professor for database and information systems in the Institute for Computer Science at the University of Munich, Germany. His research interests are in spatial database systems, particularly query processing, performance issues, similarity search, high–dimensional indexing, and parallel systems. Similarity search in CAD database systems led him to the area of electronic prototyping and digital mock–up. Kriegel received his MS and PhD in 1973 and 1976, respectively, from the University of Karlsruhe, Germany.

*Marco Pötke* is a graduate student of Computer Science at the University of Munich, Germany. His main interests are in information systems and computational logic. Pötke has just finished his diploma thesis on database support for digital mock–up in mechanical engineering, focusing on aspects of efficient representation and management of engineering data.