Solid modeling for production design

by R. N. Wolfe
M. A. Wesley
J. C. Kyle, Jr.
F. Gracer
W. J. Fitzgerald

This paper describes a solid modeling and interactive graphics computer system which is being used for conceptual and detailed design by the mechanical design community at IBM's Data Systems Division Laboratory in Poughkeepsie, New York. The system has evolved from research on solid modeling begun at IBM's Thomas J. Watson Research Center in the mid-70s. Its development has resulted in one of the first major production uses of solid modeling in industry. The system was first tested in pilot and limited production environments in 1981, and is now in production use as the primary design tool for mechanical portions of IBM's large computer mainframes. Its introduction, development, integration, and use are described, and its functional and performance characteristics as well as requirements for future enhancements are discussed. We conclude from our experience that solid modeling has become a significant new production tool for mechanical design.

Introduction

The major benefits predicted from the use of computer-aided design (CAD) and computer-aided manufacturing (CAM) systems are higher-quality products with fewer errors, shorter lead times, and higher productivity. In the past, the use of such systems in the computer industry has been largely for the design and analysis of electronic devices and their logical interconnection. This paper describes production design methods for the electromechanical aspects of large CPU mainframes. In this domain, methods are required that enable the designer to

- Interact effectively with the electrical and logic engineers who generate mechanical requirements.
- Work with Manufacturing to ensure that mechanical costs, as well as those of logic and memory, continue downward.
- Respond to rapidly changing requirements and be able to make major changes in design directions without major impact on schedules.

The approach to survival in this environment is the use of a design and analysis system based on solid modeling that allows the mechanical designer to create and visualize rapidly, to modify quickly and easily, and, through analysis, to guarantee consistency and accuracy.

Figure 1 shows an idealized product cycle. Historically, the cycle of information has been largely around the outer circuit shown in the figure, from Marketing to Development, to Process Engineering (PE) and Manufacturing Engineering (ME), to Manufacturing, and back through products to Marketing. Generally there were also feedback paths, for example from manufacturing back to design. This serial approach inevitably has long cycle times, particularly when iteration and feedback are necessary. A goal of CAD and CAM systems is not only to improve performance in the product cycle activities, but also to provide effective communication to allow these activities to be coordinated and to proceed in parallel. This is provided by the common resource of data, models, and tools shown in the center of the figure.

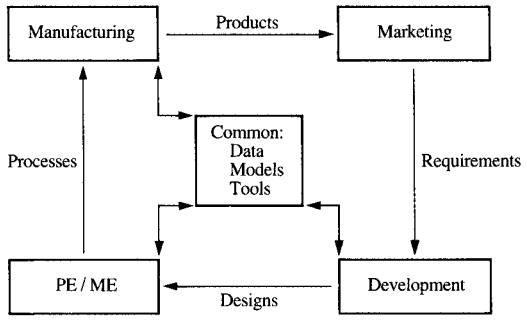


Figure 1

An idealized product cycle.

In the context of this paper, the design process has two dimensions. The first is the design domain, that is, the design of the product for function, the design of the product for form (industrial design), and the design of the manufacturing processes. The second dimension is the design phase, which is first conceptual, where rough designs are used to establish overall concepts, moving through successive refinements to detail, where final designs are completed and released to the organization that will implement them.

The method used to approach the goals of CAD and CAM outlined above is to provide an environment that allows design and extensive modeling, simulation, and analysis, all using a common data source, to ensure that designs are error-free and perform their intended function efficiently. Quite apart from the parallelism permitted by this approach, if the tools are well chosen and well designed, the productivity of designers who use them is expected to increase, and associated lead times to decrease.

Large parts of the descriptions of products and their manufacturing processes are geometric in nature. The design and analysis tools presented here concentrate on these geometric aspects, and in particular on their solid volume properties. However, although the geometric information is vital, in the context of the quantity of information needed to operate and control a complete business, geometric information is only a small part of the whole [1]. Thus, CAD and CAM systems must be able to generate and process significant amounts of nongeometric information. For example, the design system should be able to generate bills of materials and assembly descriptions, and the manufacturing system should be able to accept these as input and act on them. The CAD and CAM systems must also be able to interface with the rest of the business effectively.

In the next three sections in this paper, we discuss the environment, system, and modeling requirements for production design of electromechanical products. In the following sections, we describe the application of a specific system to the product design problem. In the concluding sections, we review the lessons learned and establish directions for future work.

Production environment requirements

In this section we discuss the requirements for the transition to a solid modeling environment that will be the basis for a production system. In order for such a transition to be possible and successful, the needs of the users and their organization must be met.

- *Users and their expectations*

The system will have a variety of types of users, including product designers and manufacturing engineers (end users), and system and application builders (internal users).

End users, who are the prime focus of this paper, generally use the system for their design and analysis activities. They may also use the facilities of the system to create specialized functions, for example to write robot programs or to create modeling "macros" to enable common strings of operations to be executed easily. They are generally provided with various forms of graphics interfaces that are discussed below, and would like a system that appears to be a well-integrated whole, with common data types and easy transfer between functional application packages.

Internal users, whose needs are not specifically addressed in this paper, are concerned with the development and maintenance of the system itself, and with the creation of new application tools for end users. They have different but largely compatible needs. To these users, a clean internal system architecture with well-defined interfaces and the availability of suitable languages for work on maintenance and enhancements are the main considerations.

For all user types, the system components must work reliably and robustly; they must be well integrated, adequately documented, and fully supported.

- *Application domain*

The system must match the needs of the application domain in both design and analysis. For example, a system for use in an aerodynamic design environment should represent the aerodynamic surfaces in a manner that meets the practices of the business, should allow design of the underlying structures that support the surfaces, and should provide analysis tools to enable, for example, calculations on airflow, drag, and lift to be performed and interpreted. If the design domain is assemblies of machined parts, then the design representation should match the types of part (for example, turned and milled), and the analysis tools should match the intended function (for example, kinematic linkages).

• *Operating procedures*
The system must match the operating procedures of the business and interface with other parts. For example, in many organizations the design and manufacturing systems are independent to the extent that there is a formal process for transferring designs known as *release to manufacturing.* Another organizational procedure known as the *engineering change procedure* may relate to the processing of alterations to a design.

• *Justification*
The use of the system must be justified. The benefits must be identified and their value estimated. Unfortunately, with the introduction of new technologies such as CAD and CAM, the existing methods for evaluation and criteria for justification do not apply well. In design, for example, designer productivity has been measured in drawings per designer per year; with CAD systems the ability of a designer to produce drawings has been dramatically increased, but the real question to be answered is *Is the design process more productive?* Also, automation of design and manufacture has many associated intangibles: for example, improved quality of the product, fewer design errors, and shorter lead times. Thus, decisions on justification of new CAD and CAM technologies often are made on the basis of high-level goals, such as error-free release, rather than detailed analysis of costs and benefits.

Finally, introducing new technology into a situation where all existing resources are straining to meet product schedules brings substantial risks of disrupting operations to a degree that negates any local benefit. The introduction must be done in a carefully controlled, phased, and checkpointed manner to ensure success.

System requirements

Our experience has been that CAD and CAM systems readily become very large and complex, often containing many hundred thousand lines of code, with many interconnection paths and many data types. Since this is a very rapidly evolving field, it is not possible to predict what the system requirements and application needs will be over the life of the system. For example, it was not foreseen that the solid modeling system described here, which was conceived originally for use in robotics, would be extended to produce an interactive CAD system for mechanical design, and that this in turn would become the basis for a silicon process modeling system (below). Hence, such systems should be extensible, and whenever practicable, new applications should be composed from existing system features. In addition, such systems must be flexible enough to respond readily to changes in operating environment, for example changes of operating system or of system-supplied functions such as graphics support or database management systems.

Thus, such systems must be buildable, maintainable, extensible through composition, and flexible. Our general approach has been to structure the system described here into a rather large number of subsystems which are, in turn, structured internally. Extensibility and composition have been provided by defining an extensible set of internal interfaces that are known collectively as the *Application Programming Interface* (see Figure 11, shown later). Flexibility to changes in operating environment is provided by localized interfaces to the environment functions. We cannot claim that the system described here achieved all these goals from its inception. Rather, it has evolved with experience to a system that essentially meets them now. However, user requirements still to be met are the basis for future work discussed in the final section.

Modeling requirements

The basic requirement of any modeling system is to provide data representations and procedures that allow all relevant questions to be answered. Thus we are concerned with the level of *semantic content* of the data and procedures. For example, in geometric modeling, a hierarchy of levels of semantic content of data may be the following:

1. Picture (picture elements).
2. 2D drawing (lines).
3. 3D line drawing (wire frame).
4. 3D solid (solid objects).
5. Assembly of 3D solids (solid objects and relationships).
6. Model of an entire physical system (geometric, electrical, magnetic properties, etc.).

In this hierarchy, a picture system allows answers to questions about picture elements, but is not able directly to answer questions about scenes. A 2D drawing system can answer questions about lines, but cannot readily answer questions about shapes; this is the typical situation with an electronic drafting system where the head of a dimension arrow has the same data structure as an element of the object. A 3D wire frame can answer questions about edges in 3D but cannot directly answer questions about volumes. A 3D solid system can answer all questions that relate to the 3D volume geometry of an object. In general, it has been found to be impractical to increase the level of semantic content automatically in a production environment [2]. A system must be designed to handle the semantic level required by its applications. In the domain of assemblies of 3D mechanical parts, the semantic level required is assemblies of 3D solids, that is, the representation of solid objects and relationships between them. The relationship set should be extensible. It should at least include relative position, hierarchical composition, assembly, kinematic joints, etc. At the highest semantic level shown, the system requires the ability to model, coordinate, and analyze geometry and other relevant physical phenomena, for example, electrical, magnetic, etc.

Another important requirement of modeling systems is *completeness* of coverage at both the level of handling all cases and the level of having sufficient objects. For example, geometric algorithms should be able to handle all relative positions of objects, or objects with holes (as opposed to only handling objects without holes). Also, all necessary objects should be in the system, so that, for example, if a question is asked about the behavior of a mechanism, all components are available for analysis.

Finally, the modeling system should be *numerically robust*; that is, the answers it gives should always be valid to a known tolerance, and the system should never fail for numerical reasons. In general, no known geometric modeling system is totally robust, but some systems have been engineered to an acceptably high level of robustness. This is an important area for ongoing research.

- *Solid modeling*

The digital representation of solid objects has received considerable attention in the past fifteen years. An important part of this work was done in the Production Automation Project at the University of Rochester and led to the formalization of the concepts of constructive solid geometry (CSG) and boundary representation (b-rep) [3].

CSG provides for the composition of objects from Boolean union, intersection, or difference of other objects. Typically, this composition is represented by a tree hierarchy (a CSG tree), and the leaf nodes are convex volume primitives which may be represented as the intersection of half spaces. The CSG tree has many attractive properties and can be shown formally to produce valid objects. In general, properties at a higher node are calculated by processing the sub-tree under the node, and may be subject to numerical error. Further, software processing the sub-tree to answer each question may lead to slow performance, a problem that is currently being addressed by design of special CSG engines [4]. Although the tree structure concept was derived and analyzed in terms of its ability to ensure validity of objects, it has proven to be a very significant feature for user productivity in production use in that it allows a designer to impose a meaningful internal structure on an object. Coupled with named objects, the structure can make the search for a wanted component in a complex design much easier.

The b-rep approach to representing solids has its theoretical basis also [5]. In b-rep, a solid is represented by the topology and geometry of its oriented boundary. There are many deep problems involved with keeping the topology and geometry consistent numerically [6]. B-rep algorithms are generally complex because of the number of special cases that have to be handled and suffer from numerical problems. However, they have sufficient performance to allow interactive systems to be built, and are well suited to answering questions that depend directly on the shape of the surface of an object; for example, how a numerically controlled (NC) machine tool cutter should be moved to produce a surface.

Another approach to solid modeling is spatial enumeration, where space is divided into cells marked for occupancy or emptiness. Clearly, for high-resolution use, fixed-size cells are not practicable. Recursively subdivided approaches have been the most successful [7, 8] and have led to specialized products for representation of nonanalytic objects such as biological organs.

In practice, we believe that the most successful systems to date have been hybrid in nature, combining both the formal structure and primitives of the CSG tree with the computational advantages of the b-rep.

Another basic underpinning of geometric modeling technology is the idea of procedural description of geometric objects [9], in which programs are written in a descriptive language to specify an object. Execution of the program with auxiliary procedures allows questions to be answered about the object and, in particular, allows a data structure to be created. Early experience with a system with this capability showed the value of having interfaces that allowed access to both data and functions, and allowed users to create procedural descriptions, perhaps by recording their dialogues for parameterized replay.

- *The Geometric Design Processor (GDP)*

The CAD and CAM system that is the basis for this paper is the Geometric Design Processor [10] which operates with a semantic content between levels 5 and 6 above. As indicated above, the pedigree of GDP lies in the Procedural Description work of Grossman [9] and in the need for a solid modeling system to support robotics [11]. With the addition of an interactive graphics interface [12], it became the basis for a CAD system. In this form it has supported research in solid modeling and has become the production design system described in this paper.

GDP is a hybrid CSG tree and b-rep system. Objects are described in a CSG tree in terms of primitives whose type, analytic form, and parameters are retained. The volume primitives cuboid, cylinder, cone, hemisphere, and translated and rotated 2D profiles are supported. An example of the use of the analytical form of the primitives is for NC programming (see below). For computational reasons, a polyhedral approximation to the boundary (the system's b-rep) is generated and used for many of the operations. A major feature of GDP is a topologically complete polyhedral Boolean operator which has been engineered to an acceptably high degree of numerical robustness. This Merge operator has made production use of GDP feasible. It has also made possible application programs like the silicon process modeler (OYSTER, see below).

GDP was implemented in PL/I, initially under the VM operating system and later for MVS also. The implementation was structured to be substantially

independent of model size. For example, objects are self-defining (using the PL/I REFER option), intermediate results in geometric operations are kept in lists rather than tables, and extensive use is made of the PL/I storage management capabilities. This approach has allowed GDP to progress from the "Blocks World" of its first robotics experiments to complex models of large assemblies running to many megabytes.

GDP components

The work leading to the present form of GDP has been driven by a desire to match the modeling system to the productivity and quality needs of the design users. As more designers used the system, development became a synergistic process in which the solid modeler and user methodologies changed and grew together. There have been six releases of GDP, beginning in January 1983. The major components of the present version (Release 6) are shown in **Figure 2** and are described briefly below.

Base solid modeler The fundamental capabilities of the solid modeler [10, 12] remain the base for the current version. They allow entry of volume primitives, combining primitives with Boolean operations to form higher-level solid objects, calculation of mass properties, interference detection, graphic editing of objects, viewing of objects from any viewpoint with a number of different rendering techniques, manipulation of the hierarchical assembly tree structure, and the recording and rerun of parameterized user command sequences (the Macro facility).

Two-dimensional geometric construction Mechanical designers have traditionally been trained to design using 2D orthographic views, and the integration of traditional 2D design methods with solid modeling techniques was essential. Therefore, a complete 2D geometric construction facility was developed and integrated into the 3D system [13]. It allows the definition of a "working plane" at any orientation and position in 3D space and construction of 2D geometry in that plane. Geometric construction facilities equivalent to those of 2D design systems such as CADAM® [14] are available to construct points, lines, circles, and arcs. Parallel, perpendicular, and tangent constraints as well as offset conditions may be specified. Intersection and tangent points of the elements mentioned above are computed from graphic inputs and stored as double-precision floating-point numbers, so that sufficiently accurate data are available for sweeping out solids from the 2D profiles.

Points and edges can be projected onto a working plane from the solid model. Two-dimensional profiles on a working plane can be translated or swept into solid objects. Thus, the 2D and 3D design environments are tightly integrated.

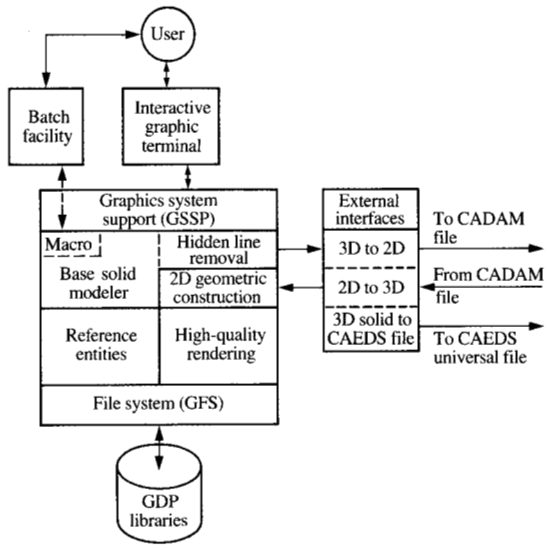


Figure 2

Major subsystems of GDP in the production environment.

Graphics system support Graphics support is provided by the Graphics Support Subroutine Package (GSSP) [15]. GSSP is a device-independent graphics interface supporting a wide range of devices, including IBM's 5080, 3250, and 3277/Graphics Attach devices on both VM and MVS operating systems. Its structured display lists provide flexibility and good interactive performance. The user interface, supported by GSSP, accepts typed commands or lightpen selections from a command menu.

Reference entities This set of functions supports hierarchical design and data sharing by allowing large structures to be composed of assembly, subassembly, and part models linked together in a hierarchical tree referencing external models, for example those contained in standard parts libraries.

By allowing multiple references to a single copy of the data representing an object, reference entities reduce storage requirements. In addition, the latest release level of each referenced model is used each time a reference is resolved, which may be explicitly at a user's command or implicitly when a referring model is fetched into virtual storage. The use of reference entities, described more fully in the subsection "Product structure," has enabled the modeling of large mechanical systems, a major strength of GDP.

File system Database support is provided by the Gemini File System (GFS). GFS is a high-level DASD interface

permitting shared read/write access to data by multiple concurrent users on multiple MVS and VM processors. Data integrity is preserved even when multiple users write to the same file. Other features of GFS include high performance, high data security, and robustness.

Data in GFS are organized into atomic libraries, which are the units of sharing. Each atomic library comprises multiple members, which in turn represent solid models and other design data. To support version control, atomic libraries can be read in sequence, a concept referred to as a composite library.

High-quality rendering In addition to flat color shading on the IBM 5080, high-quality rendering of solid models is provided by a ray-tracing algorithm [16]. The major issues were to develop a robust algorithm that provides realism in synthesized pictures, and to provide adequate performance on large models. Command menus allow light sources, color, surface characteristics, and picture size to be defined. A "quick look" feature gives the interactive user a small preview of the rendering before a batch job is started to produce the final picture. The resulting images are stored in GFS files for rapid interactive replay.

External interfaces The 3D to 2D transformation algorithm creates 2D projections of the solid model, eliminates redundant overlapping lines, converts faceted circles, arcs, and ellipses to their analytical representation, and sends this 2D geometry, as well as dimensions and tolerances, to the CADAM system where it is used to generate engineering drawing files. These drawing files, transmitted electronically, are the official release to Manufacturing. Two-dimensional drawing data in existing CADAM files can also be converted to 2D working planes in GDP, where 2D profiles can then be translated or rotated to create solid models. In this way, old designs that were originally created as 2D drawing files can be integrated into new 3D designs. The boundary representation of a solid model can be transformed to the universal file format of the Computer Aided Engineering Design System (CAEDS®) [17] for finite-element mesh generation and analysis.

Batch facility This facility comprises a set of full-screen panels, system procedures, and programs that allow batch jobs to be invoked and executed. By running a user command stream captured with the Macro facility, the system can be run in batch mode.

Production design of products

In this section we consider the design of products in an industrial environment. Although the design of tools and fixtures for manufacturing is a closely related design process, it is not considered explicitly here. The vehicle for this presentation is the production design of large CPU mainframes.

The general philosophy of operation is to start with design goals which may be geometric (for example, the "footprint" of a frame), and/or functional (for example, the heat load that must be extracted from an enclosure) and, by a process of successive refinement, proceed from a rough conceptual design to a fully detailed final design for release to Manufacturing. At each step of the process, the design is analyzed using appropriate analysis tools, for example a heat flux calculator; the user drives the sequence of design and analysis, supported by the system and its tools and interfaces.

In the product design context, completeness of coverage means that all parts that make up a level of design must be in the system in compatible form. For example, in the design of a power supply, all the components of the power supply must be available for detailed geometric layout and packaging of the supply. At the next higher level of design, the packaging of power supplies in a frame may be considered, and here it may be adequate to represent the power supplies by an approximate envelope. Thus, the concepts of hierarchical design and the flow from conceptual to detailed design have a common need for going back and forth between detailed and approximate designs.

• *Task definition—design of large CPU mainframes*
The primary objective of the IBM Poughkeepsie Laboratory is to design IBM's high-performance mainframe computers. The Mechanical, Power, Thermal (MPT) design organization provides tool support for the mechanical design of these computers. The output designs are then released to a manufacturing organization for fabrication and assembly.

The typical IBM computer mainframe is comprised of tubular welded steel frames that house electrical components, cables, and logic gates of assorted types. A complete frame is therefore an assembly of a large number of components, many of which are relatively simple (brackets, ducts, etc.), and many of which are standard off-the-shelf items (electrical fittings, fasteners, etc.). The design process involves both the specification and design of the individual components and their packaging into a frame. All elements of the design must be tested mechanically, electrically, and thermally before being released to Manufacturing.

In this large-frame design environment, completeness means that all the components in a frame must be represented in the system to a suitable level of detail; thus the tubular-frame external geometry must be rather complete, but the internal details of electronic components are usually not needed. It can be expected that some individual component models will be large (in bytes) and topologically complex (for example, the main tubular frame itself) and that the overall assembly of many parts will be very large.

The design process includes selection of materials and components, conceptual design, detailed design, checking,

and functional testing of the final design. The major goals of introducing a CAD system in this environment are to allow the quality of the design to be increased (for example by allowing designers to explore novel layouts with reduced cost or ease of assembly properties), to reduce the error rate in design by providing comprehensive tools for analyzing packaging and layout, to allow rapid response to design change requests, to interact in parallel with the manufacturing organization, and to speed up the design process. All this must be done in a cost-efficient manner.

The design of a computer mainframe is created by a large number of designers who are formed into teams, assigned specific portions of the overall design, and required to coordinate with one another in producing the desired results.

Existing design environment

When 3D solid modeling was introduced to the Poughkeepsie design community in 1980, most of the mechanical designers had been trained to use the interactive graphic methods of CADAM, IBM's existing internal production mechanical design system. As used in Poughkeepsie at that time, CADAM was a 2D automated drafting system for the creation of geometry, the addition of annotation, and the communication, storage, and release to Manufacturing of the mechanical design; the CADAM files were the master design database.

- *Early production testing*

In order to try out the solid modeling technology in a production environment, an initial six-month pilot test was undertaken late in 1980, followed by the first application test.

Pilot test

The goal of the pilot test was to answer basic questions about the feasibility of solid modeling technology for production design of mainframes. A real, but not high-priority, representative design problem was chosen and assigned to an experienced designer, who then went through the normal sequence of design steps. Solid modeling was to be used for as much of the sequence as possible. A high-frequency power supply design (see **Figure 3**) was chosen because it was reasonably complex yet small enough to be handled by a single person.

The entire design process was exercised. Design checking and the engineering change processes were considered particularly important because of the high cost and long turnaround time for processing changes in production. The sequence of design steps was as follows:

Designing and checking

- Conceptual design. Replace the sketch pad with an approximate solid layout.

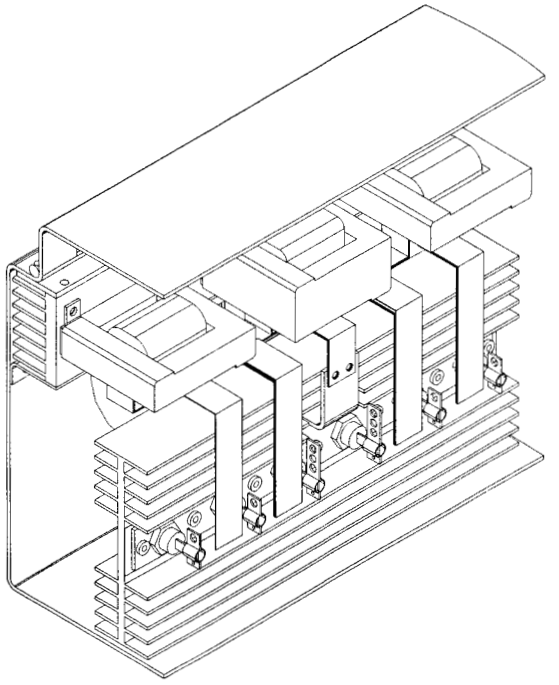


Figure 3

High-frequency power supply designed in the pilot test.

- Detailed design. Flesh out the conceptual design solid model with sufficient detail to produce release engineering drawings.
- Checking. Ensure valid component designs and error-free packaging—in particular, component alignment and lack of interferences.

Release to Manufacturing

Since 2D drawings were the required output to Manufacturing, matching this interface required

- Sending views to the 2D drafting system.
- Adding dimensions and annotation in 2D.
- Creating engineering drawings.

Processing an engineering change

- Enter the change.
- Check the changed geometry.
- Resend views to the 2D drafting system.
- Make the required annotation changes and generate new drawings.

First application test

The first production use of solid modeling in August 1981 was to route power cables in the 3090 Power Distribution

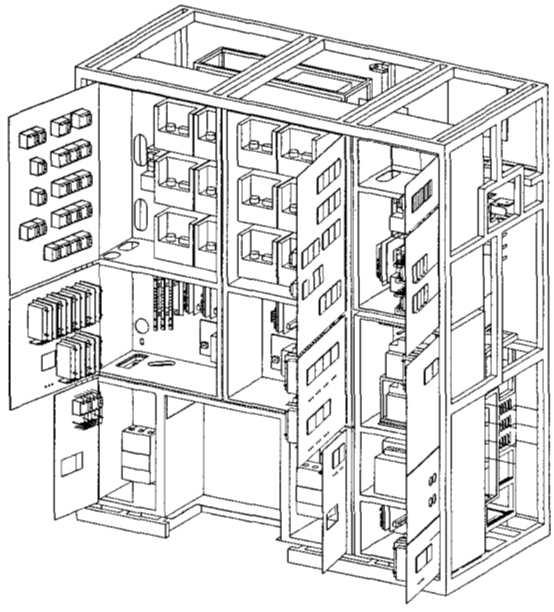


Figure 4

Model of an IBM 3090 system frame used for GDP cable routing.

Frame. Design of the routing for cables within a frame was traditionally done by taking measurements on a wooden mock-up of a frame, a time-consuming, error-prone procedure that came late in the design cycle.

The methodology varied from that of the pilot test in the following ways:

- Since most of the frame had already been designed using the 2D CADAM system, the 3D model was generated interactively from that design.
- In order to improve efficiency, a new primitive object called a line string was introduced into GDP. A line string has no volume and can be used to represent the center line of a cable path in a complex frame.
- Special engineering drawings called Cable Reference drawings were automatically created algorithmically from the line string.

Using the methodology described above, the cables were routed in the 3D model, cable reference drawings were produced, and finally the cables were manufactured and installed. The exercise showed the potential for replacing at least some of the costly physical mock-up modeling with computer solid modeling, and the cables were found to fit into a production frame with an accuracy that previously had not been achieved.

The model used for the GDP cable routing test is shown in **Figure 4**.

- *Production design methodology*

The current methodology for designing with 3D solid modeling, defined and verified in the Pilot Production Test, is shown in **Figure 5**. In the horizontal dimension the design sequence is shown; 3D solid design, 3D to 2D transform, 2D drafting, and formal release for physical modeling. In the vertical dimension, the use of libraries is shown. The design begins in a user's private libraries, where the actual 3D geometry and 2D annotation work is done, and moves through a series of other libraries as it acquires more detail and gains the approvals that make it more formal and more public.

Conceptual design

Conceptual design is usually carried out by a small group of mechanical designers, making many iterations in a process of design refinement. A major advantage of solid modeling has been the ability to make most of these iterations in the min cost loop of Figure 5. Enhanced visualization from solid models and design analysis functions such as interference detection, alignment checking, mass properties calculations, and interfaces to finite-element analysis have been major factors in the successful use of solid modeling, enabling design problems to be solved early in the design cycle.

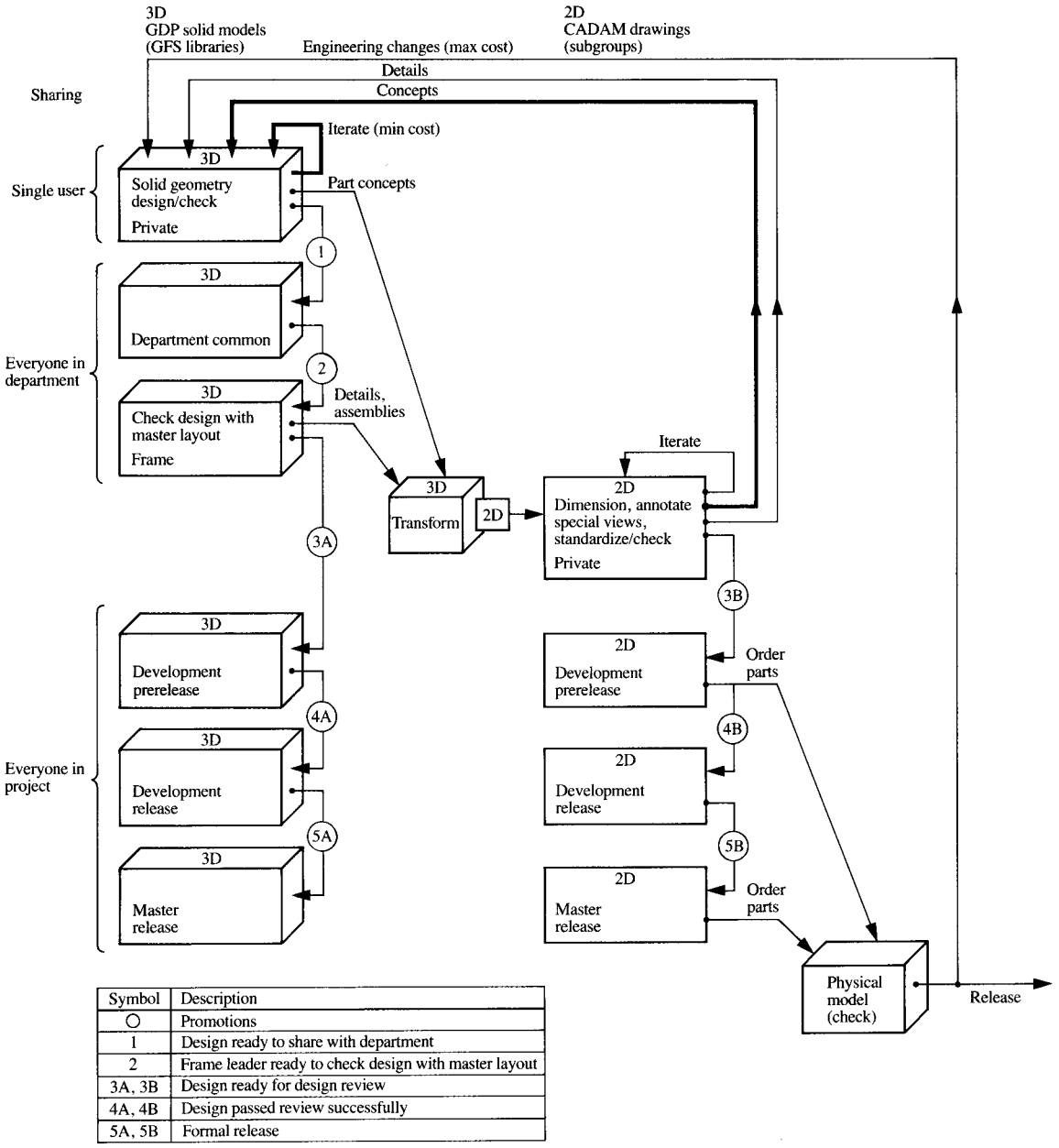
The final iterations take the longer Concepts loop, which is, however, automated and accurate because of the 3D to 2D conversion algorithm.

High-quality rendering is also a valuable asset during the conceptual phase. It allows industrial designers to visualize the exterior of each conceptual design shortly after it is proposed (see **Figure 6**). The time and effort usually devoted to building wooden mock-ups are substantially reduced or eliminated.

By the end of the conceptual design phase the mechanical system has been partitioned into frames, and the approximate size, shape, weight, and external appearance of each frame is determined. In addition, the major subassemblies, critical distances, and approximate power and cooling requirements are known.

At this time an organization and a design methodology are put together to carry out the detailed design. Frame leaders are assigned to coordinate work on each frame. Designers are chosen and assigned sections for detailed design, some working exclusively on a single frame, others working on components to be shared by multiple frames.

The library structures are now defined to reflect the organizations, their design methodology, and the product to be designed. Because the formal release process requires 2D drawings, parallel libraries are created for 3D geometry (in GDP) and 2D drawings (in CADAM files). They will hold



Symbol	Description
○	Promotions
1	Design ready to share with department
2	Frame leader ready to check design with master layout
3A, 3B	Design ready for design review
4A, 4B	Design passed review successfully
5A, 5B	Formal release

Figure 5

The methodology for using solid modeling in production.

design data as the data are promoted through Stages 1 to 5 in Figure 5 en route to final release of the product.

Figure 7 shows a typical library organization allowing access by designers in two departments to their private libraries, libraries shared by all frame and department members and libraries of common parts and released parts to be shared by all system designers. The numbers in the figure are a typical library access sequence for a designer; for

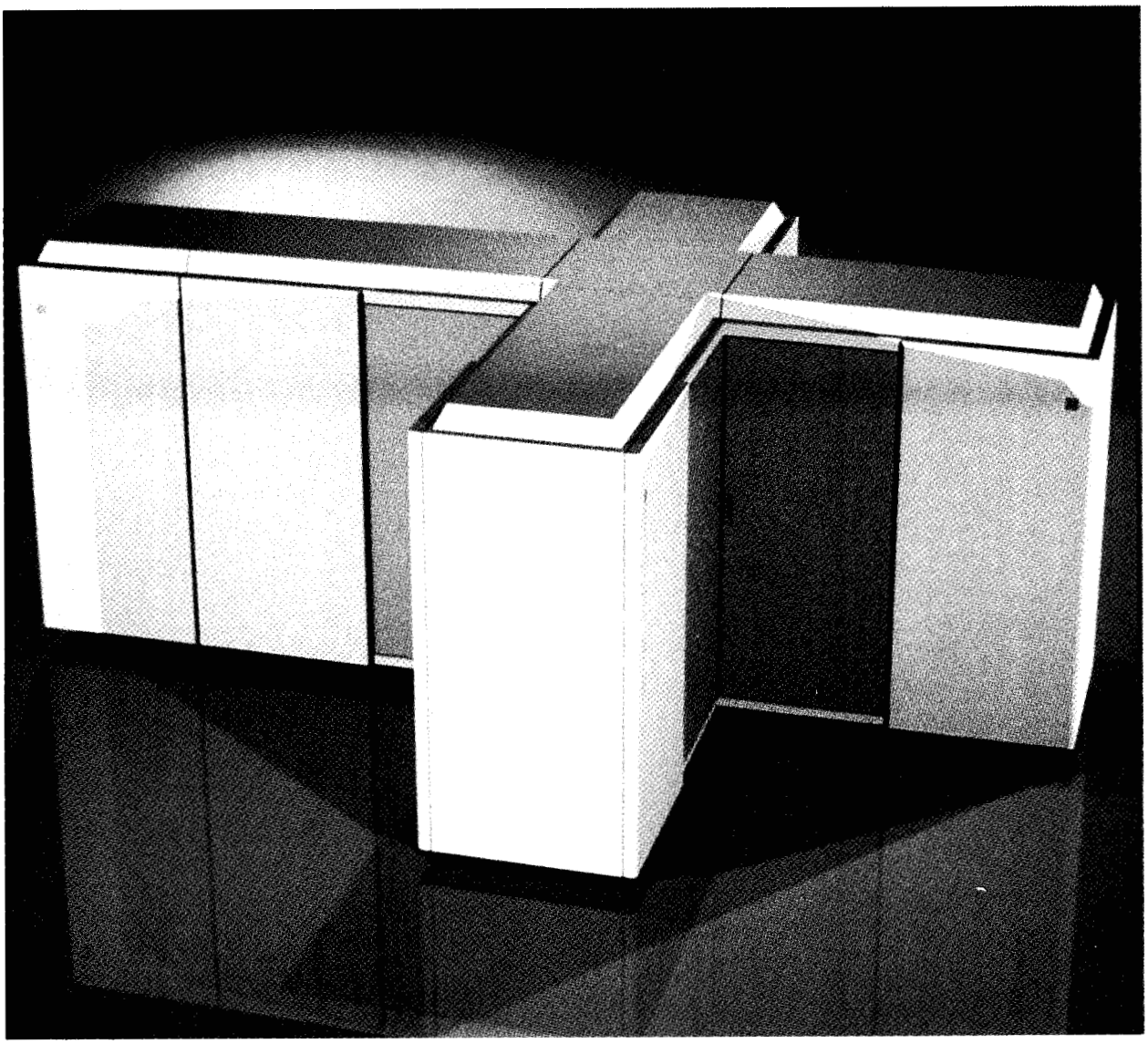


Figure 6

High-quality rendering of frame model.

example, access would most frequently be to a private library, then to a master library of designs in process for the current release, then to a library containing parts with development release part number/change number combinations assigned, etc.

Detailed design

Shape design Solid shapes are designed by creating primitive solids and/or solids swept from 2D profiles in working planes, and then combining them with Boolean operations to form higher-level shapes. The primitive solids, working planes, and Boolean operations are saved in the CSG tree for ease of editing. The logical structure and geometry of each shape are represented in the model. Tree editing commands allow the structure to be changed. This lowest-level model is called a "part" model, and is analogous to a part to be used in an assembly.

The List report in **Figure 8** shows the CSG tree structure of a part model named BUSHING. The part was made by the Boolean combination of three primitive cylinders. The order of the cylinders gives the Boolean combination sequence. CYL1 and CYL2 were combined with a Boolean union (SOLID). A Boolean difference (HOLE) was then

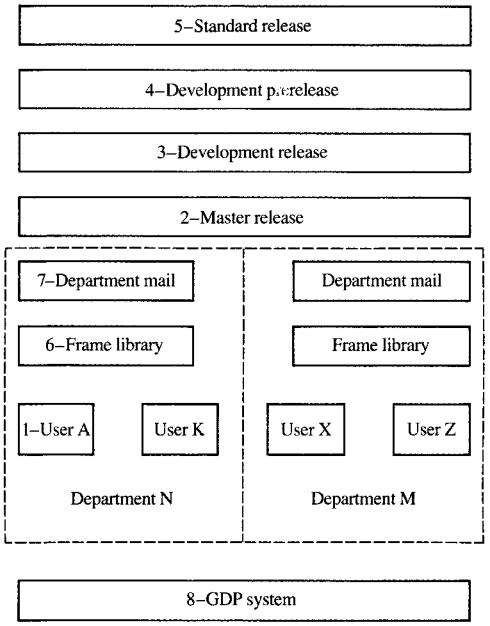


Figure 7

Typical library organization for model data.

performed, subtracting the cylindrical hole CYL3. The polyhedral boundary representations of the primitives and the object resulting from these operations are stored at each node of the tree. Thus GDP is a hybrid CSG/b-rep system.

Product structure The Family Tree Report shows the structure in a GDP assembly model, with part models as leaves of the tree. Initially there was little concern on the part of the designer with product structure in the solid model. Designers were concerned with shapes, but tree structure was considered a complexity to be avoided if possible. With time, however, the partitioning of models into parts, subassemblies, and assemblies has become a major portion of the designer's business.

Guided by the conceptual design and consultation with manufacturing engineers, the detail designer creates models of parts and assemblies. The Reference Entity function supports this product structure definition by allowing subtrees of a model to be broken out and made into new models that are then filed and replaced by references to them. Thus a new part or assembly can be created in the context of existing structure and geometry. Alternatively, models of existing parts, subassemblies, and assemblies can be combined in hierarchical relationships.

List

Level-Model Name	Description
1–BUSHING	PART,SOLID,POLY
2–CYL1	CYLINDER,SOLID,POLY
2–CYL2	CYLINDER,SOLID,POLY
2–CYL3	CYLINDER,HOLE,POLY

Figure 8

CSG tree structure of a GDP model.

Family Tree Report

Level-Model Name	Description
1–PULASSY	PULLEY ASSEMBLY
2–LPLATE	LEFT PLATE
2–RPLATE	RIGHT PLATE
2–SHAFASSY	SHAFT ASSEMBLY
3–BUSHING	BUSHING
3–SHAFT	SHAFT
3–WHEEL	WHEEL
3–BUSHING	BUSHING

Figure 9

Assembly structure of a GDP model.

A GDP model with external references is called an "assembly" model. For example, the assembly structure of PULLASM is displayed in the Family Tree Report in **Figure 9**. It shows that BUSHING is used in a subassembly named SHAFASSY, which is in turn used in an assembly named PULASSY.

Each IBM part or assembly is identified by a unique part number/change number combination. This unique name can be used as the file name of its model. Thus, assembly models with Reference Entities identified by their unique names allow the product structure of very large designs to be captured. For example, Figure 4 shows a typical computer

frame. It is one of eight frames in the IBM 3090. The assembly and subassembly structures for a single frame contain on the order of a thousand parts, and designs like these are growing to encompass entire computers. Bill of Materials and Where-used reports are automatically produced from these assembly trees.

Design checking Checking is involved at many levels of the design process. The Boolean operations have implicit checking that ensures the validity of the resulting polyhedral shape; that is, the resulting shape can safely be used in other Boolean operations or by other system algorithms. Explicit interference-checking and hole-alignment algorithms are available to ensure that parts in assemblies do not occupy the same space and fit together properly. Reports such as List, Family Tree, and Bill of Materials allow the product structure of large assemblies to be checked. In addition, the graphic outputs facilitate visual checking. Manual checking is still required for the annotation and text on finished engineering drawings.

The frame leader is responsible for checking the overall frame design. The frame library is used to collect the latest models of frame components for interference checking and to generate the reports for monitoring frame product structure.

Formal release and physical modeling
Design geometry captured in solid models must be transferred to CADAM drawings (each drawing a series of 2D views) as a first step in formal release of the design to Manufacturing. Automatic transformation from 3D to 2D has eliminated object line drafting, greatly increasing accuracy and productivity over manual drafting. After the geometry is received in CADAM, the dimensions and annotation are added to create engineering drawings. In CADAM, a graphic display is used for pointing to the 2D geometry created from the solid model by the 3D to 2D transform algorithm, and the system responds by automatically creating dimension text that accurately and reliably documents the distances and angles in the 3D solid model. Manual checking by someone who understands the design is still necessary, however, to ensure that the dimensions selected actually reflect design and manufacturing intent and that proper tolerances have been assigned. The section entitled "Dimensioning and tolerancing" discusses the automated tolerance checking facilities recently added to the system.

Parallel libraries, maintained for solid models in the GDP files, and 2D views of these models in the CADAM files (see Figure 5), are synchronized manually by frame leaders and database administrators as the formal release progresses. The same section discusses the dimensioning and documentation facilities recently added to integrate the 2D drawing data into the solid model to alleviate the problems of dual 2D and 3D libraries.

The Development Pre-Release libraries hold design data in preparation for a Design Review. When the Design Review is completed successfully, the data are moved to the Development Release libraries. After physical modeling is successful, the designs are moved into the Release libraries. These libraries provide master documentation for the design. Drawings are released to Manufacturing from the CADAM Release library.

Engineering changes
Major time savings and reduction of errors are realized in making engineering changes using a solid modeling system. Changes are made and checked out in the 3D solids environment using the analysis functions available there. The availability of the CSG tree and reference entities greatly improves the speed and accuracy with which a change can be made. The tree structure is a logical, self-documenting description of the system for engineers (often different from the ones who did the original design) who are designing the Engineering Change (EC). An EC to the released solid model must be propagated to CADAM drawings. An automatic EC Compare function highlights differences between the old and new designs by crossing out deleted lines and displaying new lines in bold line style.

Current research and development

This section describes further topics that are the subject of current research and development.

- *Dimensioning and tolerancing*

An important aspect of the manufacturing environment is the need to design the product for economical manufacture. The biggest need here is to represent and process the ideas of tolerance on parts. In general, the more tightly the tolerances on a part are held, the easier it will be to assemble and function, and the more expensive it will be to fabricate. A facility for entering 3D dimensions and tolerances into a solid model [18] meeting IBM dimensioning standards including geometric forms symbols and frames was added and released for production use in September 1986. At the time this paper was written, experience with the facility was insufficient for evaluation. Dimensions are 3D objects in the solid model that associate dimension and tolerance values with geometry and topology. Dimensions can be entered and displayed in 2D orthographic or 3D isometric views. In the production release is a new facility for automated tolerance analysis using the dimensions and tolerances entered with the facility described above. It currently provides worst-case and statistical analysis in one axial direction at a time. In addition, an experimental 3D geometric tolerance analysis capability based on ANSI standards is being developed [19, 20].

Not yet in the production release is an operating documentation facility that allows sheets of engineering drawings with multiple views to be generated from the 3D

objects and dimensions and stored in the solid model. Once established, these views are automatically updated when the 3D solid geometry is changed, including the dimension lines and text (compliant dimensions). If the geometry associated with a dimension extension line is removed, the user is notified graphically and the dimension is not discarded until the geometry associated with both extension lines is removed. Automatic design checking of this nature is very difficult when the design is done in one system and dimensioning in another. Therefore, these facilities provide for the creation of an integrated 2D/3D database, so that maintaining synchronism between the 3D solids master design data and the 2D engineering drawing data will be more automatic, less error-prone, and faster.

• *Numerical Control programming*

The basic automation technology for fabrication of parts is the numerically controlled (NC) machine tool. An NC programming facility using solid modeling and interactive computer graphics is being designed and implemented. The objective is to use 3D solid model data, generated by the Product Development organization, to create the NC machine tool motion programs to fabricate precision-machined parts in the Manufacturing organization. This will involve a new form of data release to Manufacturing; that is, 3D solid models instead of 2D drawings.

The IBM APT NC Processor [21] uses a statement-oriented language that is compiled and executed in batch mode. However, the approach to implementing the new NC facilities is to use GDP as an interactive, solid-model-based, graphical front end to APT. The NC programmer directs the display of the cutter on the screen, with GDP automatically following the contour of the solid model where it can. The functions provided by APT are available through a menu-driven interface.

The result of the interactive session is the creation of a source program which is sent to the APT compiler. APT creates a cutter location file from which machine codes are generated by a postprocessor to drive a specific machine to cut the part. This file is used by GDP to verify the correctness of the path, which is done by sweeping a model of the cutter (and optionally the chuck) along the path and checking for interferences between the cutter, clamps, and fixtures. Verification is also performed by creating the volume swept out by the model of the cutter and subtracting that swept volume from a model of the raw stock from which the part is to be cut. In this application the analytic forms of the curved surfaces, rather than their polyhedral approximations, are used to create parametric definitions of the surfaces for APT. APT, in turn, creates accurate cutter location information.

• *Silicon process modeling*

The presentation of this paper has so far considered the design of CPU frames. Another design domain involves the packaging of chips onto substrates and, in turn, into housing assemblies. Most of the solid modeling technologies covered so far are also applicable to this domain. To extend the use of solid modeling technology one step further across processor design, a GDP-based silicon process modeler has been developed.

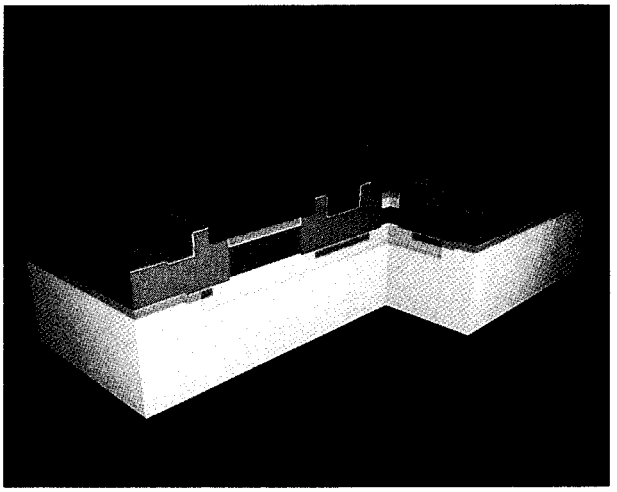


Figure 10

Section view of an OYSTER model of a bipolar transistor.

A system called OYSTER [22] for parametric simulation and analysis of the fabrication steps of very large-scale integrated circuit devices is being developed using the Application Programming Interface (API) of GDP. The application takes as input a set of 2D mask definitions and a set of statement-oriented process step definitions; for example, deposit material, apply photoresist, expose photoresist, wash, and etch. It then generates for each step in sequence the solid model that simulates the 3D geometric aspects of the result of executing the step. At any step, and especially after the last step, the component parts may be analyzed automatically to determine geometric, mechanical, thermal, and electrical properties. Statistical effects may be incorporated to allow investigation of alignment tolerance buildup and yield. **Figure 10** shows a section view of an OYSTER model of a bipolar transistor.

• *Kinematics*

In products and manufacturing machinery, kinematic linkages are common. An experimental 3D kinematics facility has been added to GDP, providing a way to deal conveniently with solid model spatial mechanisms—to create, graphically display, and edit them, and to command, drive, or animate them. Links in the mechanism may be any object or rigid assembly. Prismatic and revolute joints can be handled in the current implementation. A dynamic

interference analysis algorithm has been integrated with kinematics so that collisions can be detected when the mechanism is activated.

- *Sculptured surfaces*

The availability of well-integrated sculptured surface and solid modeling techniques is essential to a successful mechanical CAD system. Experimental sculptured surface definition and manipulation facilities, whose basic mathematical routines were taken from the Numerical Geometry System (NGS) [23, 24], were integrated into GDP. The system creates Ferguson-Coons, Bezier, and B-spline curves and surfaces. Surfaces are discretized into triangular planar facets such that the difference between the distance along a facet edge and the true curve between the facet end points is less than a specified tolerance. From this approximation to the surface, the vertices are offset to produce a thin-shelled solid in the GDP polyhedral format. Boolean operations and mass property calculations can be performed on these thin solids just as they can be performed on any other polyhedron. Further work will permit the system to save the parametric surface patch definitions in the CSG tree, in the same manner that parameters are saved for other primitive solids, so that solids can be recreated after the patch parameters are modified. High-precision machining will then be possible from the parametric definitions, just as GDP can now machine curved surfaces by using the parametric representation of the primitives in the CSG tree.

Discussion, projections, and conclusion

Here we summarize our experience with the introduction of solid modeling in IBM's mainframe design process.

- *Results*

User acceptance User acceptance of solid modeling has been high because of the benefits of its use in mainframe design. The benefits include the following:

- Better perception and understanding of designs. This has been largely due to the ability to visualize, understand, and interact with solid models on a graphics screen.
- Higher productivity for designers, resulting in faster design iterations and hence more iterations, resulting in improved designs.
- Faster and more accurate response to engineering changes. The ability to structure the design of parts in the CSG tree and assemblies in the assembly tree has made it easier and faster to locate and access models of the equipment to be changed and to edit the existing models graphically to make the desired correction or improvement.
- Higher confidence in the correctness and accuracy of the design when it is completed. The ability to analyze exhaustively via algorithms that operate on solid models means that there are essentially no errors released in the classes of checking covered by the analysis tools; for example, interference checking, hole and fastener alignment checking, tolerance checking, automatic production of views for drafting, etc. As more classes of automatic checking are built into the design process, more of the design becomes error-free at release.
- Support for hierarchical design of very large mechanical systems. This was a major requirement, and it has become a major strength of GDP. The software architecture and programming techniques used provide for model sizes limited only by hardware or operating system address space restrictions.

User interface From the beginning, the design of the user interface has provided for selection of function by either picking from a menu or typing an equivalent command. This proved to be a good decision. Beginners tend to use the prompting provided by menus, but experienced users are often delayed by having to traverse a hierarchy of menus to select a command they know and can immediately execute by typing one or a few key strokes.

The developers originally believed that 3D design with primitive solids would satisfy designers' requirements. This was not entirely the case. Although much of the design can be done in 3D, there are compelling reasons for retaining some of the 2D design methodology that originated on the drafting board and was automated with electronic drafting systems; intersections, tangencies, parallel and perpendicular constraints are functionally important and are very often more easily created in two dimensions than in three. A key to the success of solid modeling was the design of a user interface that provided efficient integration of the 2D and 3D methodologies.

Another correct decision made in the beginning was to allow primitive solids to be defined by pointing; that is, one could point to existing geometry on the screen and create a primitive solid "in place," rather than entering the parameters, creating the solid in a standard orientation on the screen, and then rotating and translating it into its desired position. This procedure eliminated much redundant, error-prone retyping of parameter data and improved user productivity.

Completeness and accuracy of shapes Initial concern over the completeness of object shapes that could be handled was largely allayed. The primitive solids and swept 2D profiles that are supported have been adequate for shapes encountered in frame design. The effects of polyhedral approximations on accuracy were reduced by using more facets where necessary (there are effectively no limitations on the number of facets that can be used, but there is cost in space and time) and by using the analytic forms of the

primitive solids. It is interesting to note that creating intersection and tangent points to high precision in 2D and sweeping them into solids often results in a more accurate model because the facet lines and vertices associated with these intersections are on true surface boundary intersections, whereas using Boolean operations to create the equivalent shapes from primitive solids may result in facet lines and vertices that numerically are not on the true surface intersections.

Robust Boolean operations Although numerical round-off error in Boolean operation algorithms is known to be a problem in solid modelers, GDP has been very robust in this regard.

Memory Memory requirements were identified as a major problem from the beginning. Moreover, the ability of GDP developers to increase the available model size has always been outpaced by the designers' demand for bigger models. This is treated further in the section entitled "Remaining issues and future directions."

Computer graphics Evaluation of the performance of computer graphics systems is usually measured in milliseconds to respond to a user command. The objective has traditionally been to maximize the number of user interrupts per unit time. For solid modeling, where a single user command normally accomplishes much more than a single command in, for example, a drafting system, instantaneous response is much harder and more costly to provide. The goal is to minimize the overall time to implement a given design; with GDP, the number of user interactions with the system is very much reduced, but the average time per interaction is higher. Users recognize the higher function in solid modeling commands, appreciate its value, and accept longer response times. Of course, users always desire and developers attempt to provide faster responses.

Another difference of interactive solid modeling from a computer graphics point of view is the larger number of vectors required per display image when hidden-line suppression is not being used. For example, four vectors are required to display a rectangle in a 2D drafting system. An axial view of a cuboid in a solid modeling system produces the same rectangle picture but requires 12 vectors, one for each edge of the cuboid. Therefore, techniques for minimizing display buffer overflow are essential. Further, axial views of solid models normally have overlaid lines. When the solid is drawn with solid lines dashed, methods must be used to synchronize dashes and gaps so that overlaid dashed lines do not appear as solid lines.

Shaded color images are responsible for improving the visualization and understanding of designs in solid modeling. Graphic hardware to create these images from polyhedral representations is very much needed to improve system performance. The requirements of a graphic display for solid modeling are quite different from those of a display for drafting. Higher (or special) function, higher performance, and large display list capacity are key characteristics.

Table 1 IBM 3081 CPU usage for GDP drawing functions.

	CPU time (3081 seconds) *1-megabyte model*
Wire frame display	0.38
3D to 2D Transform	2
Hidden lines removed	16
Flat color shaded	135
Ray-traced rendering	1080

Model structure To reach the level of automatic checking necessary to achieve error-free release, solid models must accommodate mechanical systems in all their detail and in natural hierarchical structures of assemblies, subassemblies, components, and parts. The development of reference entities was a major factor in solving this problem. Large virtual memory savings are achieved by referencing a model instead of reproducing its data for every instance of its use. Reference entities promote the use of standard parts libraries, thereby standardizing designs, lowering cost, and reducing design time. Version control is easier to implement with the resolution of references when assembly models are fetched. Designers can be notified by the system when the level of a referenced part changes. Although the creation and maintenance of model structure was not easily understood by detail designers, it has become a common tool for frame leaders who have responsibility for overall frame structure.

System independence The need to be able to run under different operating systems and to incorporate different support packages was identified early in the development cycle and led over time to a system architecture that made such flexibility possible. Operating-system-dependent code has been localized to a small number of routines. The Graphic Support Subroutine Package provides efficient, device-independent graphics support. The Gemini File System provides operating-system-independent database support.

Batch support Although GDP was conceived as an interactive system, as models and assemblies became larger, the need to be able to run computation-intensive functions (for example, Boolean operations, hidden-line-removed drawing, 3D to 2D transforms, ray-traced high-quality image generation) in batch mode became apparent (see **Table 1**).

Table 2 Operational data showing the growth of GDP in production use: CPU minutes are given for equivalent IBM 3081 times. Disk storage (DASD) is given in megabytes (MB).

	1983	*1986*	
	Cable routing	*Designer*	*Frame leader*
Users	2	20	
Max model (MB)	3	10	16
Virtual storage (MB)	8	14	16+
DASD (MB) per user	5	85	143
Interactive CPU minutes per connect hour	0.41	2.1	
Batch CPU minutes per connect hour	0.82	1.5	

The Macro capability of GDP was used to describe operation sequences to be performed in batch mode, and interactive menu-driven facilities were provided to allow quick and easy batch job submission directly from GDP. A production solid modeling system would not be economically feasible without a good batch facility.

Release to Manufacturing The interface to IBM Manufacturing (including subcontractors) is the engineering drawing. Therefore, the first interface to Manufacturing from solid modeling had to be the interface to engineering drawings. Automatic production of views for engineering drawings from solid models was necessary, as was transformation of these 2D projections to CADAM file formats. The difficulty of producing release drawings from solid models and the amount of computing time consumed in creating them were not anticipated. Elimination and/or concatenation of overlaid lines in 2D projections was essential to control the amount of data created; CADAM files are limited in size. In addition, CADAM dimensioning functions operate only on the analytic representation of circles and arcs, not faceted representation. Development of the EC Compare function was a large unanticipated programming effort required to provide an electronic "marked-up brownline" to highlight drawing changes for manufacturing engineers.

A weakness in 2D drafting systems is the difficulty of producing an isometric view from 2D orthographic views. Automatic production of isometrics is a strength of a solid modeling system. Therefore, a natural interface to Manufacturing is the production of isometric exploded-view drawings for assembly. By graphically disassembling the model of an assembly and saving the views along the way, pictorials for assembly instructions can be created easily and accurately.

Numerical Control programming of milling machines from solid models is another interface that is just becoming available but has not had production testing.

Introduction of new technology The risks associated with the introduction of a new technology into an existing environment have been mentioned above. In the case of solid modeling for IBM mainframes, the risks were minimized by the early availability of the 3D to 2D transform algorithm which allowed production of releasable drawings from GDP. Thus, both 2D and 3D designs could exist compatibly in the same development organization. Initially, a small portion of the design effort was committed to solid modeling to give it a production test, and the remainder was done in 2D as usual. The output to Manufacturing was 2D drawings in both cases. If problems had occurred in solid modeling, 2D drawings would have existed at that point and the design would have been completed in 2D.

- *Remaining issues and future directions*

Computational resources As the system has evolved, user expectations have continually grown and always seem to exceed the capability of the system at any given time. To derive maximum benefit from solid modeling, all the design in all its detail needs to be represented in the model. As designs become more complex and more detailed, models grow to sizes larger than can be handled with acceptable performance. This has been and still remains the most difficult challenge.

In order to measure and justify the cost of the required computing resources, data have been gathered at various times during the development and use of the system. We include a sample of these data to provide prospective users of solid modeling with real examples of computing resources used.

A performance checkpoint was first taken during the production use of GDP for cable routing in 1983. Each designer used the VM/370 operating system with an eight-megabyte virtual machine. Model space for the IBM 3090 frame model was about three megabytes. Each designer used about 1.5 minutes of System/370 Model 168 CPU time (approximately equivalent to 0.4 minutes of IBM 3081 CPU time) for each hour of interactive connect time. In addition, each generated 3 minutes (about 0.8 IBM 3081 minutes) of off-shift batch work for each hour of interactive connect time. Most of the off-shift work was 3D to 2D transformation for creating engineering drawings.

Table 2 is a comparison of the 1983 operational data with 1986 operational data, showing the growth of production use. CPU seconds for 1983 have been converted from System/370 Model 168 to equivalent IBM 3081 times. Storage is given in megabytes. Because the data for frame leaders and designers are different, 1986 data are shown for both types of users.

Comparison of these data with other applications indicates that a designer's interactive CPU utilization was

approximately equivalent to that of an APL user and approximately three times greater than that of a designer using the 2D CADAM system.

Table 1 shows the CPU time required to perform a number of different graphic display functions on a solid model running the current production algorithms on an IBM 3081 with the VM/370 operating system and an IBM 5080 display. The one-megabyte model contains about 8700 edges. The wide range of times illustrates the presence of both highly interactive and computation-intensive operations.

Larger, faster CPUs with larger address space (for example, IBM's 370 Extended Architecture) and special hardware for Boolean operations, hidden-line removal, and computer graphics will continue to be needed. Likewise, the need for faster algorithms will always be present.

Database and communications Designers need to consider all product definition data, not just solid models. All engineering data need to be consolidated, indexed, and shared among diverse applications. For example, thermal and electrical schematics contain data that need to be accessed by the solid modeler. Stress and thermal analysis programs receive geometric data from a solid modeler and in turn provide inputs to the modeler which change the design. Downstream applications that plan assembly processes and model work cells need inputs from the solid modeling design system, and may use the modeler's algorithms and programs, for example, for checking interference or for displaying results. Designers must also be able to communicate with and receive data from sources outside the designer's immediate environment. This means that solid model data must be integrated with a larger product database and communication system, and that an open modeling system architecture must be provided that allows solid modeling data and algorithms to be easily shared by new applications.

System development facilities As the system has grown bigger, the cost of maintenance, support, and the implementation of new applications has become harder to control. The architecture of GDP has evolved in response to growth and now approaches a modular structure with an extensible interface to both function and data known as the Application Programming Interface (see **Figure 11**). New applications may now be written to the interface. Its extensibility allows new applications to be composed from existing ones. However, the cost of introducing changes to existing components, for example, adding a new data type or changing the interface to a function, is still high. Current work on systems architecture is addressing these problems and is directed toward the use of object-oriented systems built with object-oriented languages exemplified by AML-X [25] (see Figure 11).

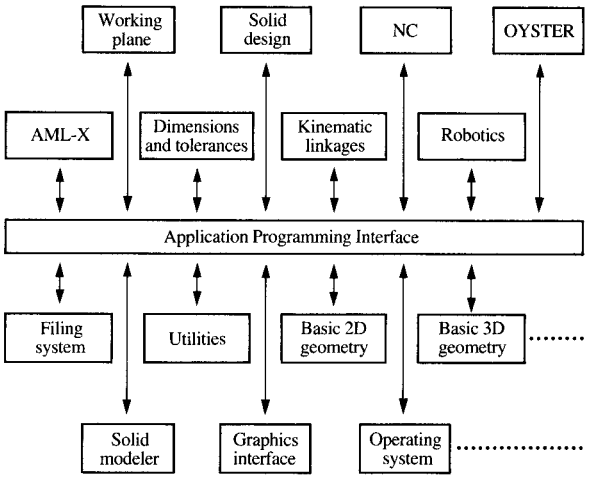


Figure 11

Architecture of GDP Application Programming Interface.

Conclusion

It is not yet possible to make precise analytical comparisons with previous mechanical design methodologies. However, with endorsement by users and their management, solid modeling has become a central technology in the design of IBM's large high-performance computers. Its use is spreading from the mainframe design and packaging described in this paper to other areas of the product ranging through the levels of electronic packaging to the design of semiconductor chips. Thus its versatility and usefulness extend from the exterior design of the largest mainframe computers to the small details of the circuits that drive them.

Acknowledgments

The work described here is the product of software development done by past and present members of the Solid Modeling Applications Development, Solid Modeling Technology, and MPT Data Technology departments in the IBM Data Systems Division (DSD), Poughkeepsie, New York, and the Design Automation Department, IBM Thomas J. Watson Research Center, Yorktown Heights, New York. Definition of user requirements and testing of the system were done by past and present members of the Advanced Processor System Packaging Department, IBM DSD, Poughkeepsie.

The Computer Integrated Manufacturing Lab at Lehigh University, under the direction of Professor Emory Zimmers, was instrumental in the design and development of the 2D geometric construction, dimensioning, and tolerancing, and NC programming

facilities. The Center for Interactive Computer Graphics at Rensselaer Polytechnic Institute, under the direction of Professor Michael Wozny, was instrumental in the design and development of the high-quality color-rendering facilities. The Computer-Aided Design Lab at the University of California at Los Angeles, under the direction of Dr. Larry Lichten, was instrumental in the design and development of the sculptured-surface-to-solids facilities.

The authors would like to thank Bob Zwissler for his numerous contributions to the Pilot Production Test, Brian Watt and Gerard Muenkel for the color rendering of GDP models used in the paper, Steve Williams for the discussion on user methodology, and Bob Hillerich for assistance with the figures.

References and notes

1. Joseph Harrington, Jr., "Understanding the Manufacturing Process," Marcel Dekker, Inc., New York, 1984.
2. M. A. Wesley and G. Markowsky, "Fleshing Out Projections," *IBM J. Res. Develop.* **25,** 934–954 (November 1981).
3. A. A. G. Requicha and H. B. Voelcker, "Solid Modeling: A Historical Summary and a Contemporary Assessment," *IEEE Comput. Graph. & Appl.* **2,** 9–24 (March 1982).
4. G. Kedem and J. L. Ellis, "Ray Tracing Machine," *Proceedings of the International Conference on Computer-Aided Design '84*, October 1984, pp. 533–538.
5. C. M. Eastman and K. Weiler, "Geometric Modeling Using Euler Operators," *Proceedings of the First Annual Conference on Computer Graphics in CAD/CAM Systems*, April 1979, Cambridge, MA, pp. 248–259.
6. Victor Milenkovic, "Verifiable Implementations of Geometric Algorithms Using Finite Precision Arithmetic," *Proceedings of Conference on Geometric Reasoning*, Oxford, England, July 1986; to be published in *Artificial Intelligence*, North-Holland, Amsterdam.
7. L. J. Doctor and J. G. Torborg, "Display Techniques for Octree-Encoded Objects," *IEEE Comput. Graph & Appl.* **1,** 29–38 (July 1981).
8. D. J. Meagher, "Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary Three-Dimensional Objects by Computer," *Technical Report IPL-TR-80-111*, Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, NY, October 1980.
9. D. D. Grossman, "Procedural Representation of Three-Dimensional Objects," *IBM J. Res. Develop.* **20,** 582–589 (November 1976).
10. M. A. Wesley, T. Lozano-Pérez, L. I. Lieberman, M. A. Lavin, and D. D. Grossman, "A Geometric Modeling System for Automated Mechanical Assembly," *IBM J. Res. Develop.* **24,** 64–74 (January 1980).
11. L. I. Lieberman and M. A. Wesley, "AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly," *IBM J. Res. Develop.* **21,** 321–333 (July 1977).
12. William Fitzgerald, Franklin Gracer, and Robert Wolfe, "GRIN: Interactive Graphics for Modeling Solids," *IBM J. Res. Develop.* **25,** 281–294 (July 1981).
13. Carol M. Riggin, "The Enhancement of IBM SOLID to Include Geometric Construction in a Plane," M.S. thesis, Lehigh University, Bethlehem, PA, 1984.
14. CADAM is a registered trademark of CADAM, Inc., Burbank, CA.
15. R. B. Capelli and G. C. Sax, "A Device-Independent Graphics Package for CAD Applications," *IBM J. Res. Develop.* **28,** 512–522 (September 1984).
16. Linda A. Roy, "A Ray Tracing Renderer for a Geometric Modeler," M.S. thesis, Rensselaer Polytechnic Institute, Troy, NY, December 1986.
17. CAEDS is a registered trademark of the International Business Machines Corporation.
18. R. G. Glemser, "The Feasibility of Enhancement of the Geometric Design Processor (GDP) to Include Dimensioning," M.S. thesis, Lehigh University, Bethlehem, PA, May 1984.
19. J. U. Turner, "Tolerances in Computer-Aided Geometric Design," Ph.D. thesis, Rensselaer Polytechnic Institute, Troy, NY, May 1987.
20. D. Hoy, "A Solid Model-based Tolerance Analyzer of Size and Orientation Tolerances," M.S. thesis, Rensselaer Polytechnic Institute, Troy, NY, September 1986.
21. *System/370 Automatically Programmed Tool-Advanced Contouring Numerical Control Processor: Program Reference Manual*, Order No. SH20-1414, available through IBM branch offices.
22. George M. Koppelman and Michael A. Wesley, "OYSTER: A Study of Integrated Circuits as Three-Dimensional Structures," *IBM J. Res. Develop.* **27,** 149–163 (March 1983).
23. L. Lichten and M. Samek, "Integrating Sculptured Surfaces into a Polyhedral Solid Modeling System," Technical Report, Computer-Aided Design Lab, University of California at Los Angeles, 1986.
24. J. P. Mayfield and R. M. Burkley, "Applications of a Numerical Geometry System in Engineering," *Proc. IEEE Design Automation Conf.* **13,** 25–33 (1976).
25. Lee R. Nackman, Mark A. Lavin, Russell H. Taylor, Walter C. Dietrich, Jr., and David D. Grossman, "AML/X: A Programming Language for Design and Manufacturing," *Proceedings of the Fall Joint Computer Conference*, November 2–6, 1986, Dallas, TX, IEEE Computer Society, Washington, DC.

Received October 29, 1986; accepted for publication February 3, 1987

Robert N. Wolfe *IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Mr. Wolfe received a B.S. in Radio Engineering degree from the Indiana Institute of Technology, Fort Wayne, in 1955 and joined the Military Products Division of IBM in Poughkeepsie, New York. He has also had assignments in the Data Processing, System Communications, Advanced Systems Development, and Research Divisions of IBM, and in the IBM World Trade Corporation. He has received IBM Outstanding Contribution Awards for his work in information retrieval, character and pattern recognition, computer graphics, and solid modeling. Since 1969 he has been a Research staff member at IBM's Thomas J. Watson Research Center, Yorktown Heights, New York. He is currently manager of the Design Automation Applications project in the Manufacturing Research Department, responsible for the development of the Geometric Design Processor (GDP), a three-dimensional solid modeling and computer graphics system for computer-aided mechanical design and computer-aided manufacturing.

Michael A. Wesley *IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Wesley is manager of the Design Automation Department in Manufacturing Research at the Thomas J. Watson Research Center. He was educated at Dulwich College, London, and the University of Cambridge, receiving a B.A. in mechanical sciences in 1960 and a Ph.D. in control engineering in 1966. Since 1966 he has been a Research staff member at the Thomas J. Watson Research Center and for the past fourteen years has worked on automation research. He has worked on application programming for industrial robots, the Autopass high-level robot language proposal, the Geometric Design Processor geometric modeling system, algorithms for path planning, algorithms for generation and conversion of geometric databases, and the OYSTER system for modeling silicon processes.

James C. Kyle, Jr. *IBM Data Systems Division, P.O. Box 390, Poughkeepsie, New York 12602.* Dr. Kyle received a B.S. from Virginia Military Institute in 1957 and an M.S. and a Ph.D. from Syracuse University in 1965 and 1971, respectively—all in electrical engineering. His IBM responsibilities have included implementing computer controls for integrated circuit wafer manufacturing, automating X-ray spectrophotometers in a materials analysis laboratory, developing algorithms for paint and textile color matching, and designing measurement and control strategies for laser video disks. Since 1980, he has worked on solid modeling techniques for the design of computer mainframes. He is one of seven workers in DSD Poughkeepsie to have received an IBM Outstanding Technical Achievement Award for GDP development work.

Franklin Gracer *IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Mr. Gracer, an Advisory Programmer in the Manufacturing Research Department at the IBM Thomas J. Watson Research Center, is currently involved in the application of 3D modeling to mechanical design. He received a B.S. degree from City College of New York and joined IBM as a programmer in the Service Bureau Corporation. Since moving to the Research Division in 1966, he has worked in the areas of design automation and computer graphics. Mr. Gracer has received an IBM Outstanding Contribution Award for his work on the Geometric Design Processor (GDP), a CAD/CAM system for mechanical design. He is a member of the National Computer Graphics Association and an adjunct lecturer in computer-aided design at the IBM Systems Research Institute in Thornwood, New York.

William J. Fitzgerald *IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Fitzgerald is a Research staff member in the Manufacturing Research Department at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York. He received a B.E.E. from Yale University, New Haven, Connecticut, an M.E.E. from Rensselaer Polytechnic Institute, Troy, New York, and a Ph.D. from the University of California at Berkeley, all in electrical engineering. His areas of research are algorithms, applications, and user interfaces for computer-aided design systems. Dr. Fitzgerald is currently directing work on dimensioning, tolerancing, and numerical control programming for the Geometric Design Processor (GDP) solid modeling system. He joined IBM in Poughkeepsie, New York, and worked on power systems for large computers. Dr. Fitzgerald managed a project which designed and fabricated superconducting circuits at the Federal Systems Division in Kingston, New York. He was awarded an IBM Resident Graduate Fellowship to attend the University of California at Berkeley, where he received a Ph.D. in electrical engineering in 1965. Dr. Fitzgerald was involved in planning for graphics in the Data Systems Division in Kingston, New York. In 1968 he joined the Research Division to manage an experimental input/output laboratory. There he worked on developing a raster display for computer graphics and a graphics system for mechanical drafting. This work was continued in the IBM Europe/Middle East/Africa Corporation in 1976–1977. Dr. Fitzgerald returned to the Research Division in 1978 and since that time has done research and development in solid modeling for mechanical design. He is a member of the Computer Graphics Group, the Computer Society, and the Institute of Electrical and Electronics Engineers.