Boundary Representation in Practice

Michael Fraser

There is an essential contradiction between the making of buildings or built environments in a three-dimensional modeler and the graphic control of this process. Three-dimensional modeling is a constructive activity, in which solids are assembled as they would be in an actual structure; it benefits the designer. Presentation and documentation, on the other hand, are prescriptive activities that direct some of the construction and all the visualization and criticism of the proposal; they benefit the user and builder.

A building while being designed can be visualized and criticized from its solid model, and the model can take a variety of forms depending on its part: computer-based, drawn in orthographic or perspective projection, constructed of cardboard or wood, or described narratively by means of text, programmatic data, performance model or animation. However, practicing architecture is the process of recording and communicating the decision making process and the contractual obligations that result. In actual practice, in contrast to the designer-directed ideal, more participants are brought in sooner at the beginning of a project and with more publicity, which in turn means keeping more, not fewer, records. As the profession evolves, records of the string of design decisions will become more automated, more carefully structured and more retrievable. More buildings will be “tracked” and exposed to review in this way because public environmental sensitivity will improve. The communication between a single designer and his own thoughts will become less and less important.

This disparity between designing and recording is the cause of the delay in integrating computers into design. Commercial CAD systems perform many prescriptive tasks that are also handled well by, for example, desktop publishers: providing visual clues, directing attention, organizing detail, clarifying and isolating errors, and identifying the participants in the building process. Design sketchers and 3-D modelers perform, instead, the constructive activities of assembly, experimentation, optimization, insight, and exploration of alternatives. They do not find themselves to “directing the eye” of users or builders toward the important or significant geometric elements in the model and away from those that do not require compliance. Modelers also do not encourage visualization at several scales, several levels of detail, simultaneously.

We traditionally defend this disparity as part of the familiar linear process: conceptualize, design, evaluate, detail. First we decide what to build, and only then do we determine how, by whom and when. This is the “problem seeking” ([Pena 1977]) model that has dominated modern architecture and around which a traditional architectural design practice is organized: partner-in-charge sets goals, project manager gathers facts and establishes objectives, project designer proposes alternatives and then, and only then, detailers execute solutions.

But an actual project is in fact cyclical. The massing and general application of materials are approved in their sketch form, the design is developed quickly, and a design development set is produced with the hope that it all comes in under budget with materials and components that will actually be available in time to install them. The traditional, often imaginary scheme-design-detail progression produces good architecture only in those cases where the building type, materials, availability and methods of assembly are predictable and the participants, including suppliers and clients, have long experience in satisfying one another's expectations. For complex projects requiring continuous communication and continuous change, the linear model discourages necessary feedback.

The team approach, in contrast, connects owners, owner’s employees and other users, construction
managers or contractors, designers and some suppliers at the outset of a project. The team can identify likely problem areas and concentrate design efforts where they are most needed. Last-minute changes are more likely to be consistent with the original program and plans. The design intent, made clear at the outset, has a better chance to survive through bids and field changes. And the design solution takes into account the method of building, integrating craftsmanship, materials, assembly, and detailed appearance rather than just general massing, volumes and surfaces. Two-dimensional drafting systems have continued to dominate the design profession because prescriptive design-aid software can smooth the transition to team-based integrated design in a way that constructive, three-dimensional modelers cannot.

The Modeler/Documenter

Solid modelers at their present stage of development are appropriate for the schematic design phase of the obsolescent linear design method, but they do not yet handle communication, feedback, prescriptive graphics and participatory change well enough to be used in actual practice. Unless solid modeling exploits the advantages of prescriptive drafting systems, it will be left behind in the isolated world of the individual designer. Client expectations of the degree of possible polish and customization have risen. Much of the responsibility of the architect has been usurped by construction managers, clients, consultants, lawyers, facilities managers, value engineers, and suppliers. We must be able to make more, not fewer, last-minute changes. Detailed building costs must be predicted from the beginning of the design, along with structural feasibility, availability of materials, the environmental and energy impact of material choices, and continually changing feedback from the surprising variety of people who will occupy a client’s building. All these concerns must simultaneously occupy the center of the designer’s field of vision while entire schemes are abandoned and reinvented, often immediately before or during a client approval session.

Three features of a good, powerful drafting system (one that requires a certain amount of experience to master) make it appropriate for a fast-feedback modern design process:

- The methodical and standardized work, especially detailing, can be stored in libraries. The user interface encourages fast, unambiguous searches and snapping of named and attributed objects to 2-D points.
- Changes are easier to make, again because of the simplicity of attaching data-rich objects to 2-D entities.
- Document appearance can be edited during design changes, so that construction is more carefully prioritized and directed and more consistent with design intent.

These are the three features that solid modelers most severely lack. Modelers, especially those that emphasize intuitive ease in the sketching stage, tend to emphasize elegance and consistency, sacrificing data richness and variety. CAD drafting, on the other hand, is often complex, idiosyncratic, peculiar to different office settings, and peripheral; drafters concentrate on printing, dimensioning, leader lines, keynotes, and other communication devices that would, if drawn during sketching and design, cripple intuition. Modelers must be general; drafters must be specific.

An integrated modeler/documenter must provide, simultaneously, a general massing and rendering device that allows progressive levels of detail, along with a framework on which specific details and notations can be “hung.” It must allow parametric modeling without building in a specialized user interface. An integrated modeler/documenter — the “edges modeler” which will be described here — must rethink the traditional contract drawing set from the bottom up. Its users should lose no time reinventing assembly details; standard libraries must be fully available at the start of schematics. The design must proceed unevenly; some elements will be finished and specified before other parts are even schematically designed. Changes that are propagated through a design must be allowed to completely rearrange previously detailed parts without requiring redrawing. Control of appearance and prescriptive power must be available early on in design sketches. And non-graphic attributes must be expected to be attached to every drawing at all times.

There are now commercially available modelers that allow sections to be cut to create building plans, sections and elevations. Unfortunately, these are usually outlines that must be filled in with shading and descriptive infill, whose line weights must be completely reworked to account for changes in...
emphasis and scale, and to which drawing notes must be painstakingly added. Worse, the 3-D model cannot be articulated into finer detail after the section is developed. Modelers must be improved to allow for continuous integration of detail, graphic finish and textual and numeric data no matter how or how late design decisions change. The eventual goal is to link all geometry to data-based text and hypermedia, so that the communication process is not dependent on drawing technique.

Architectonic space is characterized by the unioning and intersecting of the empty volumes contained within walls and ceilings of given thicknesses. This approach, void modeling [Vessios 1987], encourages the interpenetration of open spaces, complexity in the intersection of bounding planes, and the accretion of intricate occupiable space. But it does not easily accommodate complexity and changes of materials, thicknesses, construction methods and links to data and text within the planes themselves.

The architectural operations of intersecting, unioning and differentiating occupiable space are formal, sculptural and preliminary. In practice, this phase of design is individual, quick, and intuitive; it is characterized by extensive but not exhaustive study. An architect will dash off many spatial studies but not spend time on any single space. In most cases, he or she will not show most of the tentative spatial alternatives to the users of the space, but will instead throw them away. There is little prospect that a client will be interested in all the alternatives that might be generated; it is a personal, loosening-up, exploratory device, meant to be used in privacy, to evoke an individual's intuition.

The vast bulk of architectural study and communication is spent by both designers and technicians on the articulation of the resulting spaces. The general space-shaping operation almost always faces these constraints:

- New construction is usually closely bounded by regulatory envelopes. Freewheeling expressionist formalism can occur only on a few isolated sites.
- Formalist treatment of volumes is often concentrated on a few key pivot points. Here, the design process focuses in on modeling operations at a smaller and more intricate scale, directing the attention of the building's users toward the critical junctions in the design.
- Much and progressively more design is the rehabilitation of existing buildings, for environmental and emotional reasons as well as economic. In a rehabilitation job, grand spatial operations are strongly constrained and must instead be intimately responsive to specific structural connections, existing materials, and fire codes.
- The design of highly technological installations for research, teaching or treatment may require only the placement of some cabinets and dividers, followed by an intensive study of mechanical supply, storage, display and circulation. Spatial operations are closely dependent on non-spatial engineering information.
- Some magnificent architecture, northern European in particular, is assembled from manufactured components; only the foundation is made of a material, concrete, which is subject to a subtractive sculptural design approach. The glazing method, panel sizes, rhythms of light and shadow, repetition of structural connections, and edge conditions — especially at the edge of the roof — are more important determinants of the character of the space than are the interpenetrations of voids surrounded by a generalized solid material.

The alternative to designers working directly on drafting machines is the "gray area" identified by [Harfmann and Majkowski 1992, p. 107] between the conceptual model and the accurate component model. Harfmann and Majkowski approach the geometric conflicts between the designer's volumetric model and the finished component assemblies by performing a minimax test on the possible extent of solid components, subtracting this extent from the surrounding void, and reiteratively "adding back" void material as the solid components' boundaries are traced. While it can handle the geometric complexity of design development, it cannot isolate materials, connections or structure.

The Object Based Modeler (OBM described by Antonio Saggio [Saggio 1992, p. 50; see also Eastman 1991] is hierarchical and incremental, although it does not solve for interferences. Saggio's incremental modeling has generated a "Concept-Testing" conjectural studio environment that supplants the traditional "Analysis-Synthesis" linear model. OBM, which relies heavily on instantiation, encourages the full development of concurrent alternatives — exactly
the emerging model of practice. Therefore, OBM handles the development of materials and connections simultaneous with general design decisions. But OBM should be refined to prevent incorrect intersections at the solid terminators of the hierarchy and should not be allowed to create instances and freely insert and scale them; parametric stretching ("variational modeling") of parts of details, rather than stretching and inserting entire details, is not only a more accurate method of design development, but it better reflects the non-repetitive quality of the actual design process.

A hierarchical structure can keep both the descriptive volumes and the assembled components in the same model and respond best to the cyclical design development process. "Sketching" is perfectly possible in a 3-D drafting machine like AutoCAD, so long as there are AutoLISP or AutoCAD Development System programs, written in C, to expedite the quick creation of volumes by means of stretching and copying. If the modeler is allowed to stretch rough, over-all space models and attach detailed parametric components to those vertices and edges of the model that he or she knows in advance, then subsequent design changes can preserve the previously finished detail work. Hartmann and Majkowski’s "gray area" can be bridged again and again if a robust geometric description allows a single geometric area to be occupied simultaneously by several entities at different stages of detailed finish. It requires only that the details be correctly "carried" on the edges and vertices of the volumes, so that dimensional changes to the general volume will cascade through the hierarchy and transform the "contained" solid components.

What will make the "edges modeler" described here useful is its recognition that a great deal of the working-out of design is dimensioning. Ambitious and baroque spatial operations are possible in 2-D CAD systems because a designer, sitting at a machine, can cast angles, stretch parts of a volume and measure, or automatically dimension, the result. Exploring certain rhythmic patterns of tile, brick, and glass, impossible by hand, is often delightful on a drafting machine. Dozens and dozens of sketch alternatives at each stage of design development are made possible by standard layer naming conventions, instance blocks with non-graphics attributes, redlining utilities, and extensive text and graphics power. Well-managed and standardized CAD systems require surprisingly little "training," good designers can and do work from start to finish on computers.

But these typical activities are too "technical" and data-rich for most solid modelers. An exception is the complex curves used on projects in Frank Gehry’s office [SDRC 1991], which are done not with an architectural solid modeler but with IBM’s CATIA aerodynamic modeler. Architectural solid modelers do not provide data-intensive services like libraries of details, which have attributes such as connection points, parametric dimensioning, and visible links to commercial databases that contain non-graphic data. Some of this attribution problem has been solved in aerodynamic and machine-tooling variational modelers like I-DEAS by SDRC [Novitski 1992] but these modelers favor constructive solid geometry (CSG) operations on homogeneous solids like steel and aluminum.

CSG modelers work well for integrated manufacturing operations, which are distinguished from architecture because:

- Significant time is spent on a single mechanical design, because it will result in savings on a long production run, whereas architecture is always one-off.
- Manufacturing on lathes and drill presses is a subtractive operation that is closely analogous to the Boolean operations used to create the model.
- Tool paths run through a uniform and predictable solid matrix made of a single material.
- Mass property analysis is an important part of the design.

Architectural models are unpredictable, heterogeneous, additive rather than subtractive, sometimes very roughly tolerated, prone to modification during production, and very, very time-consuming to design in proportion to production time. One or two carefully tolerated models of a metal part will result in hundreds or thousands of pieces produced and sold, but it takes dozens of alternatives to design each small part of a unique building.

Proposed Structure of the Edges Modeler

What is needed is a solid modeler that respects the variety and mutability of architectural materials. Rendering requires that CSG models be converted to bounded surface representations before materials...
and colors are applied. The “edges modeler” proposed here posits that a kind of boundary representation is preferable to constructive solids during the generation of the model.

Central to this proposal is the notion that there are very few homogeneous masses of architectural materials, such as solid plaster, solid machined metal, wood equally stressable along all axes, carved marble, and unreinforced poured concrete panels, masonry of a particular thickness, number of wythes and coursing, glass, shingles, paint and fabric. Rather than sculpt a generalized solid form and choose the construction method and surface material afterward, a good designer knows the surfaces and structural orientation of all the materials before he or she begins to use them. Currently, solid modelers do not encourage “play” with materials linked to performance and numerical data, within the general void volume.

The method of assembly of a building generates its architectural character; the architecture occurs at the edges. The depths of window and door heads and sills, reveals at slabs, treatment of the edges of the roof, masonry joints at corners, expansion joints, ledges and the grade line are where architectural decisions are made. Edges are the interpenetration of two or more different materials, but solid modelers usually treat edges as the intersection of a generalized homogenous solid that will be referred to in this paper as corpus, or an indistinguishable mass made of a single material that cannot be articulated into smaller geometric elements. Corpus is specified by performance standards, structural behavior and methods of manufacture; it includes plaster, stone, marble, glass, some pressed woods, solid metal, plastics, air, named occupied volumes, geometric prisms of uniformly translucent atmosphere, uniform smoke (for fire-behavior models), and three-dimensional open volumes which are required for access to an adjoining surface. In the edges modeler, natural and instantiated corpora are followed by X,Y,Z substrates identifying their bounding edges — which, unlike any noncorporal data, correspond to a simple scaling of a 1x1x1 corporal primitive.

In a very few cases, corpora will have geometric characteristics, e.g. carved marble, especially at bases and capitals of columns, pored precast concrete panels without reinforcement, stone heads and sills, statuary, or intricate unreinforced floor slabs. Otherwise, architectural geometry is characterized mostly by intersections of unlike materials at edges. Edges are generated either by Boolean intersections of panelized corporeal elements or by the termination of one material at the boundary of another. Traditional contract documents describe a building almost entirely by its edges. Each detail is a cross-section of an edge of an assembly of materials that are usually of one or two fixed dimensions: studs, sheathing, connectors, predimensioned beams and columns, reinforcing at specified intervals, gutters, pieces of flashing, and membranes. By assembling all the edges in a network, a detailer (presumably) fully geometrically describes a project (e.g., Figure 1, showing a detailed building section overlaid on its 3-D building “model”). Each edge has two fixed, prespecified dimensions, and its length is inherited from the location of its endpoints within the overall assembly. In the individual detail drawing, this third dimension, length, which extends in a direction normal to the plane of the drawing, is said to be indeterminate. When an edge appears in a network in the edges modeler, it will have one of its three substracts (X, Y or Z) replaced with an “n”.

Some of the edges are indeterminate in two dimensions and are faces. A detail of a window sill will show a cross-section of a linear edge, fixed in two dimensions, for mullions, muntins, sills, caulking, gaskets, and extruded metal parts, but it will also show a part of a pane of glass, which extends for some not predefined or immediately apparent (indeterminate) distance up to the head. Because in this individual detail we do not know the height of the window that is cut by splicing lines, or its depth, which is invisible (since it is normal to the plane of the drawing), we say that the glass is indeterminate in two dimensions and is therefore a face. It is shown with two “n” substracts.

Contract documents are less effective at showing the intersections of edges at vertices than at edges. Sometimes, as in a complicated flashing connection or roof overflow scupper, the detailer will draw an axonometric view. Otherwise, the termination of a building corner at its gutter, or the junction of a sill and jam, is left up to the contractor or manufacturer. Often the two intersecting edges will have similar or analogous sections, and can be simply mitered together. In most cases, the intersection is hard to visualize until it is executed; an experienced architect will presume, intuitively, how a contractor is likely
to build the joints at the vertices and be prepared to solve problems in the field. So a fully-determinate detail, fixed in all three dimensions, is not well handled by conventional orthogonal architectural construction documents. The vertices of an assembly are more likely to be illustrated in the product literature—premanufactured connectors, grommets, gaskets, caps, hinges, miters, elbows, vanes, and pivots—than on the architect’s contract documents.

A solid modeler needs to select appropriate vertex treatments: mitering, cutting, welding, hinging or adding a premanufactured corner piece. For the edges modeler, a vertex and the subassembly associated with it, inserted at a single point but perhaps carrying other, nongeometric data or parametric expressions, will be called a dot and identified with three preestablished subscript dimensions.

Corpus, dot, edge and face together make up the solid model that is described, although clumsily, still, by a traditional set of construction documents. Single and double line walls are sectional descriptions of faces, entities that are indeterminate in two dimensions (until a dimension note is attached—see Figure 2, a hypothetical office-type room which will be further developed in this example). In the case of double line walls, the face carries a numerical attribute that determines wall thickness, the layers are detailed elsewhere. Where the wall thickness changes, where two or more faces intersect, or where the wall is interrupted by a column, the description of the joint is an edge that can be completely described, except for its indeterminate length, with a two-dimensional cross section. Even though an assembly of columns, walls, changes in material and volumes of surrounding open access may be large and complex, the entire complex edge assembly is predefined in a separate design operation—it refers the builder to a separately-drawn detail. Therefore, there are only two interesting characteristics of an edge in an overall architectural plan or building section, no matter how complex: the coordinates of its endpoints, whose extent along one axis (the indeterminate axis) is dimensioned in the overall building plan or section; and a cross-reference to a more finely detailed model, whose dimensions and configuration have been defined somewhere else.

The edges modeler maintains this limitation: An edge carries only the indeterminate dimension and instances of a cross-section which are dimensioned elsewhere. Where a number of faces are grouped
together and intersect — the best example being the four walls around an office — the document's user may be assisted by a room number label or shaded area in plan. Often, especially in section, a three-dimensional grouping of planes into a volume is ambiguous, with spaces flowing into one another. While a solid modeler is well prepared to create these irregular, flowing spaces, it does not do so well in identifying them, assigning non-graphic attributes and isolating them from one another. Void modeling does not by its nature encourage the additional graphics and notation needed for grouping faces and vertices together and cross-referencing associated non-graphic data.

The Network Structure

The edges modeler sacrifices the traditional modeling emphasis on faces, three-dimensional objects and volumes, in favor of a hierarchical description of a three-dimensional network of edges and vertices. In this modeler the vertices are dots which refer to subassemblies of further edges and vertices or to a corpus. When the edge hierarchy can be fully searched to non-geometric search terminators (which refer to contractually-specified building materials or corpora), the network model is said to be resolved; every dot and edge defines the intersection of exclusively corporeal volumes. A completely resolved model can be rendered with no gaps and tested for interferences.

Remember that the term "corpus" can include non-solids; atmosphere, volumes of access, boundaries describing ownership or construction staging, etc. The only difference is that these corpora can be overlapped whereas, usually, overlapping solid corpora signal interference errors. The non-graphic data determine whether and when two ownership volumes, for example, are allowed to overlap. Because a corpus is a uniform specification for a volume of any dimension, the character of the corporeal volume is established entirely with non-geometric data: "high-strength concrete," "galvanized steel" (not "galvanized sheet steel" or "steel rod"), "rental space belonging to Toys 'R Us," "access area to be used by wheelchairs," or "water." By contrast, "16 ga. sheet steel" is a doubly-indeterminate face entity, "1/4" steel rod" is an edge and "Bolt as scheduled" is a dot.

The philosophical structure advocated by the edges modeling approach is that it can distinguish between terminal and non-terminal nodes in a network search and that it can identify geometric adjoining surfaces distinct from geometric overlapping volumes. All records in an architectural instance or case base have a geometric location in relation to the overall model, and the (tentatively three-dimensional) geometric model is the basis on which all other information is indexed. Geometric searches are best handled by a three-dimensional network of vertices and edges to which either geometric or non-geometric data can be attached. We can assign any value, behavior or logical relationship to any vertex, edge, face or set of edges with only three syntactical restrictions: every element has a geometrical point (the origin) attaching it to a parent element, some network elements (corpora) terminate a hierarchical search, and some network elements (solids) cannot be overlapped.

This structure will most easily store most of the geometric and non-geometric information, including decision-making sequences, that occur in an architectural project. The edges modeler will also provide a starting point and "reduced instruction set" of grammatical relationships upon which to build expert systems, for interference checking, top-down search, bottom-up search except in cases of instancing (described below), graphic search and multimedia-based interactive graphic search. Presentation graphics (hatching, weighting, text, and pointers) can be stored in the same structure as world space models. The particular application or behavior of a volumetric, graphic or alphanumeric element is of no concern to the modeler; it is merely a database structure that will keep the entire geometric model readily accessible to a variety of user interfaces.
The basic structure, analogous to an overall plan or building section in a conventional drawing set, is a non-directed three-dimensional network of edges and vertices. Dimensions are carried on the edges. The internal representation does not store edge data separately from vertices; each vertex carries an ordered list of vertices to be paired with it, and a pair of associated edge vectors, one originating at each vertex, to be carried with each vertex pair. Faces and volumes are always traced from, and inherit their characteristics from, edges. An edges modeler can also keep phasing and alternative-scheme information, on a fourth or greater dimension, by linking groups of volumes in time sequences.

The edges model is prejudiced in favor of orthogonality. The four vertices of a typical face carry rotation information to orient the intersections of the face with adjacent faces. Each pair of intersecting edges which describe a face, if correctly oriented, will also guarantee planarity.

Each vertex ("dot") carries a pointer to a separate subnetwork that is nested one level deeper in the hierarchy. A subnetwork that is referenced by more than one parent vertex is a named instance. The parent vertex, offspring network and rotation comprise the dot, and the positioning of the subnetworks around the XYZ coordinate of the dot (by concatenation, described below) is the definition of the dot. The dot definition itself carries no dimensional information; this is all predefined within the separate definitions of each offspring subnetwork. The dot, however, carries rotational information for its intersecting edges and faces; corresponding opposite dots across a face or edge must carry complementary rotations for these same faces and edges. Curved faces are carried as spines on the bounding edges.

The separation of dimensioning from detailing is analogous to a desideratum of contract documents: on an overall plan or section, dimensions are acceptable, but detailing that is to be shown elsewhere, on partial plans or detail sections, should not be drawn or dimensioned on the overall plan. There are two very good reasons for this often-ignored rule: (1) as in any hierarchical database structure, duplication at two levels generally leads to conflict due to incomplete, inconsistent or interrupted data maintenance, and (2) because it is a prescriptive document, a general plan or section should require the executing builder or fabricator to turn to the sheet on which the detail is drawn and not allow guessing at the intent from the large-scale plan. Drawings with a particular focus always supersede overall drawings; the overall drawings should, ideally, contain no building information at all but instead simply index the details. And as evolving design methodology comes to respond more to detailing, this separation from dimensioning will be valuable; a generalized space will be able to change dimension and even configuration in response to, and without affecting, the details. Simple variational-model problems, such as scaling a piece of furniture without scaling the thickness of its frames or resizing its plan representation without distorting its lettering (see Figure 6), can be solved with the edge-and-dot approach.

In the edges modeler, each dot also carries a pointer to non-geometric data. Neither a parent nor an offspring network in the modeler/documenter needs to be physically solid. Physical or relational characteristics of edges and dots are entirely under the control of the alphanumeric data associated with the graphic object. It is only necessary to distinguish those subnetworks that contain corporeal data; these are not permitted to reference or instance subordinate networks at the dots. The dots in this case will bound a corpus whose material properties are specified non-graphically. A minimum of four vertices (usually eight, presuming orthogonality) will bound solid corpora such as ahpnom, inanimate or ownership, and a minimum of three vertices (plus a marker to identify back and front faces) will bound facial corpora such as gypsyboard of a specified thickness, a finish such as paint, or a specified surface appearance such as "reflective, but confirm choice of finish before date x/x/x." A minimum of two vertices are required to define an edge corpus like a steel rod, an edge molding specified by catalog number and not subject to further geometric decomposition, or a linear segment of an interactive animated travel path.

The modeler will have an internal interference checker to insure that physically solid corpora do not overlap. Since each subnetwork referenced by a dot has a single origin, all subnetworks on a dot, corporeal or otherwise, will be concatenated by their origins. Concatenation is the insertion of a row of subnetworks along an axis, the extent of each subnetwork determining the insertion point of the next adjacent subnetwork on the dot. For each dot pair in the parent network (assuming the dot is fully defined and not just a temporary holding position), all concatenations along the connecting edge are checked to make sure the two strings of subnetworks
do not collide somewhere along the edge. In Figure 3, which is the definition of the two lower corner dots in Figure 2, the lower left dot is the origin point of a subnetwork called CORNER which is defined in both X and Y as 5" (in this example, all Z-values are predefined as a constant value, "Z(rm#)", which could also be a number like the wall thickness. If this subnetwork were truly a dot and not an edge, the Z would be indeterminate).

Because none of the three dimensions of CORNER is indeterminate, the next adjacent subnetwork can be concatenated to it anywhere (so long as solids do not overlap). This same point is where SIDEDOOR(X(dm#),5,Z(dm#)) and WALL(X(dm#),5,Z(dm#)) are concatenated. SIDEDOOR is defined elsewhere, so we know its X value by finding "dm#" in a door schedule; X(dm), the width of WALL, is a minimum clearance to a storage cabinet as determined by door swing and Americans with Disabilities Act regulations. Finally, WALL-STOR(l,29,z(rm#)) is concatenated along the X-axis, and this terminates concatenations along the X because the X-dimension of WALL-STOR is "1" (indeterminate). Therefore, the dot at the opposite corner of the room must have a complementary WALL-STOR(l,29,z(rm#)) terminating its concatenation "string"; the width of the horizontal edges of WALL-STOR (they are not faces because their Z-values in this case are predetermined) inherits its X-value from the X-dimension shown in Figure 2. SIDE-

DOOR(X(dm#),5,Z(dm#)), shown in Figure 4, is an "X-

string" (concatenation along X) that includes a fully-
determinate wall section called (WALL(4,5,Z(dm#))), a standard offset of this kind of door from a side wall, which is concatenated along Z to a header WALL which is indeterminate in the Z and must therefore meet a complementary WALL terminating a Z-string which starts at the ceiling. When WALL(4,5,Z(dm#)) meets the header WALL(4,li), their identical sections

Figure 3

ACADIA 1993
will result in an uninterrupted, fused, L-shaped wall surrounding the door. In the two subsequent lower levels (Figure 5), the left door jamb is defined as a concatenation terminating in JAMB-LAP, which is composed of the aluminum corpora JAMB-RETURN-CORPUS and JAMB-SIDE-CORPUS, allowing realistic rendering of the side completed jamb. Rendering could have occurred at any level of the hierarchy, but not so realistically as at the terminal corporeal level.

In AutoCAD, the implementation of the string concatenations is simpler than the formal description; the user needs only to assemble the subnetworks in blocks and instance them, using the dot as the insertion point. The doubly-directed network edges can be implemented with special pointer attributes assignable in recent AutoCAD releases.

Were a network completely defined by dots and solid corpora, a building could be defined completely by solid components floating in space or touching at their faces. Some buildings are more easily characterized this way; those of Richard Rogers and Norman Foster are examples. However, most of the visible volumes produced by non-industrialized traditional building methods carry their visibility information on their faces: paint or paper on the surfaces of gypsum wallboard, carpet, wood and metal (the last two also forming the predominant visible edges). Faces and edges can be generated from (at minimum) triplets and pairs of dots, respectively. The outermost concatenated subnetworks of each dot are left without dimensions along the concatenating axes. Dot pairs are checked for mirror-matched partial definitions. The modeler/documenter then provides a prism of connecting bounding faces to tie together each correctly matched pair of dots, or it provides an inner and outer face to connect each group of three or more dots, where each adjacent pair in the group is correctly mirror-matched.
The only difference between the handling of solids and voids is that the solids are checked for simple interference. Non-solids may carry differing overlap restrictions and, by checking overlap restrictions deeper in their hierarchies, dot definitions may overlap. There are corporeal and non-corporeal networks for non-solid data; a required access space adjacent to a surface, such as a one-meter air space that is required so that painters may reach the surface, would be defined as a corporeal face, since the access space is not permitted to be subdivided. On the other hand, a division or department identification or hatch could be further subdivided and would therefore not be corporeal but instead bounded by defined dots, edges and faces.

Nothing yet described in this modeler encourages repetition. The hierarchy is likely to become very deep; a rented office space (a network) references a group of faces that can be rendered to show alternative finishes, which in turn intersect roughly detailed edges at the corners, which are further detailed into dots. The dots can then reference face entities from catalogs (wallboard, paneling, glazing), which break down into rows of edge-type studs and dot-type connectors. In the contract document set, facial entities could be corporeal that carry non-geometric pointers to manufacturers’ catalogs or Underwriters’ Laboratories standard details (which are recognized, by number, by approving code authorities. The catalogs, in turn, can show the edges and faces of assemblies of wallboard, finish material, and studs. The application decides how deep to go in the hierarchy, and dimensional changes in schematic design cascade through to the indeterminate low-level edge, face and volumetric data without affecting the predimensioned dot definitions. The general rules of assembly are:

1. Indeterminate axes (one per edge, two per face, three per volume) inherit their dimensions from the distances specified by the user between the parent entities’ dots;
2. Defined axes (one per face, two per edge, three per dot) concatenate the offspring networks starting at each dot, in such a way that they do not overlap;
3. The last concatenated entity in a string which begins at a defined dot must mirror the last entity from the string starting at the opposite dot in the pair. The corpora used to fill in between dots in pairs create renderable edges or faces.
4. Concatenated strings do not include overlapping solid corpora.
5. Corporeal entities inherit all their dimensions; and
6. All dots may reference relational, non-geometric data.
There are some repetitive operations that should be allowed, and would be prohibitive in a traditional solid modeler. Simple repetitive instancing with rotation and scaling, as performed at this time by object-based modelers, destroys all but one or two attributes of each subnetwork. In some cases, however, the form generated by repetition should not be modeled explicitly, either because it leaves too little latitude for field conditions (e.g., where each 16'-on-center stud is placed) or takes too much time to model (e.g., all the Mullions for a curtainwall system). In these cases, it makes more sense to instance a prototype. Repeated insertions of the prototype along an edge can be taken care of with a modification to the concatenation operation. Array concatenations include an inherited dimension; e.g., the concatenation of studs at 16'-on-center stud is placed) or takes too much time to model (arrayed along a curtainwall system in 3D) is repeated until the studs fill up the inherited edge-length along the wall. Like all string operations, this repetition can be justified on the dot, centered on the edge, or (as in the vertical spacing of steps) stretched evenly to fit. However it is done, if the parent dimension changes, the modeler automatically adds or deletes instances to correctly fill the edge length. The final rule of assembly is:

> **Array concatenation at a dot inherits a dimension from the parent entity and inserts multiple instances of a single subnetwork according to the type of justification specified for the array.**

By using these seven rules, a designer may make dimensional changes to parent entities, and even change their geometry by adding vertices, without changing the geometry of the subordinate entities. The modeler checks that the designer has not “stretched the edges” so short as to overlap its two dot definitions. The modeler also takes care of repetitive instances which must be added or subtracted along the changed edge.

**Conclusion**

All modeling operations, which may be architectural expert systems, evaluators, or case search algorithms, can be built on top of this core system, as long as each logical frame or state includes at least a single threedimensional (or, with phasing included, four- or more-dimensional) insertion point. A single model carries an entire building description and, with the proper user interface, limited ranges of the hierarchy can be filtered for views, near the root of the hierarchy for schematics or near the bottom for details. For renderings, only corpora will be displayed, with associated surface attributes passed to a renderer. Communication graphics and text will be installed directly on dots, scaled or rotated for views, and filtered out for renderings.

CAD-literate firms are already performing all these cross-referencing activities and thus could be said to have started modeling. But their models are fragmented into clumsy, redundant sheets of paper; the cross references are still only attributes of instantiated graphics blocks; the hierarchy is too flat; and much of the database information is carried in separate files and in one-way extractions into spreadsheets. By pulling all these operations together into a single geometric structure — rather than ignoring them because they seem inappropriate to three-dimensional spatial design — a single model can preserve information through the entire design and building process.
References

Gero, J.S. and Mary Lou Maher, "Mutation and Analogy to Support Creativity in Computer-Aided Design," CAAD Futures '91, ed. G.Schmitt, Friedr. Vieweg & Sohn, Braunschweig/Wiesbaden, 1992. Case-based models [see also Oxman and Oxman 1992, p. 313] that distinguish semantically precedented design characteristics from those that require the identification of new dependency relationships; a case-based model carries a one-level deep hierarchy of design characteristics for each object, and instances or features to look up that where a precedent exists.


Kalay, Y.E., "Computational Modalities of Evaluation and Prediction in Design," CAAD Futures '91, ed. G.Schmitt, Friedr. Vieweg & Sohn, Braunschweig/Wiesbaden, 1992. This case-based reasoning study distinguishes the evaluative uses of the model at conceptual, development and detailed design phases, cross-indexing these design phases with technically designed objectives.


Shih, Shen-Guan, "Case-Based Representation and Adaptation in Design," CAAD Futures '91, ed. G.Schmitt, Friedr. Vieweg & Sohn, Braunschweig/Wiesbaden, 1992. Describes context-sensitive grammar, which is organized by a hierarchical structure but instead by individual prototypical architectural characteristics. Also context-free case bases, in which a form or notation that is proposed as a design solution may satisfy a goal previously established by an evaluation system.

White, Richard, "Recognizing Structures: Some Problems in Reasoning with Drawings," CAAD Futures '91, ed. G.Schmitt, Friedr. Vieweg & Sohn, Braunschweig/Wiesbaden, 1992. Compositional (bottom-up) and decompositional (top-down) tree-traversing schemes that permit graphic and non-graphic data to be searched for in a pure hierarchy or cyclic network. "A structure that the designer considers to be two spaces divided by a wall might be considered a single space by a fire safety application if the dividing wall offers no resistance to the progress of a fire" [p. 391].