Assessment of features technology

J J Shah

Features encapsulate the engineering significance of portions of the geometry of a part or assembly, and, as such, are important in product design, product definition, and reasoning, for a variety of applications. Feature-based systems have demonstrated some potential in creating attractive design environments and in automating the geometric reasoning required in applications such as process planning and manufacturability evaluation. The paper reviews the major concepts and approaches that are in use in feature-based modeling. Several methodologies are prevalent for creating feature models and databases. These fall broadly into the categories of interactive definition, automatic recognition/extraction, and design by features. Within each, there are several subcategories, which are discussed and compared in the paper. Also presented are several schemes popular for representing features. They include augmented graphs, syntactic strings in grammars, and objects in object-oriented programming. Feature interactions and validation issues are outlined. Attempts at developing feature taxonomies are also summarized.

FEATURE TYPES AND TAXONOMIES

Several studies have identified features used in various applications. Some have done so for the express purpose of cataloging features, while others have done it in the course of building feature-based reasoning systems. For example, the John Deere Company identified form features that are manufacturable by machining and sheet forming in terms of generic shape, dimensional parameters, and attributes. The list of features and their properties listed in the report was exhaustive, as was the list of materials (by name), heat treatments, and data needed for specifying tolerances. The intention of the study was to identify all data that was needed by process-planning systems. Cunningham and Dixon have also identified features used in many manufacturing applications.

The number of features is not finite, but it may be possible to categorize them into groups or classes. Classification may be useful in the following ways. First, if features could be classified into families, and their properties identified, then perhaps mechanisms could be designed to support each family, instead of special methods being supported for each feature. Second, feature classification could perhaps lead to some common terminology. Third, a feature taxonomy could be useful in developing product data exchange standards.

Several schemes have been proposed for classification that are based entirely on shape, rather than on the application. A scheme was developed by Pratt and Wilson for CAM-I, and adopted for the form-features information model of the Product Data Exchange Specification (PDES). PDES classifies features as follows:

- **Passages**: subtracted volumes that intersect the preexisting shape at both ends.
- **Depressions**: subtracted volumes that intersect the preexisting shape at one end.
- **Protrusions**: added volumes that intersect the preexisting shape at one end.
- **Transitions**: regions involved in the smoothing of intersection regions.

Much of the initial work on features seems to have come from a desire to devise methods to extract part geometry from geometric modelers from which process plans, GT codes, and NC programs could be generated. Thus, the manufacturing view of features is that features represent shapes and technological attributes associated with manufacturing operations and tools. Because the concept of features is deeply entrenched in geometric modeling, the viewpoint of the modeling community is that features are groupings of geometric or topological entities that need to be referenced together. With the advent of the 'design by feature' approach, the definition now seems to be much broader: features are elements used in generating, analyzing, or evaluating designs.

Although many different definitions appear in the literature, the common denominator seems to be that features represent the engineering meaning of the geometry of a part or assembly. Feature-based modeling systems support additional levels of information beyond those available in geometric modelers. The availability of high-level information makes the design environment more attractive, produces a richer definition of the product, and allows one to automate downstream applications to a higher degree.
• **Area features**: dimensionality-2 elements defined on faces of preexisting shape.
• **Deformations**: shape-changing operations, such as bending, stretching etc.

Of the six classes shown above, the first three have definite volumetric associations, and are either sweeps or rulings.

Form features have also been classified on the basis of the role they play in design. Cunningham and Dixon classified features as static or kinetic. Static features were divided into the following groups:

- primitives (major shape),
- add-ons (local modifications),
- intersections (nature of interaction of primitive and add-on),
- whole form (attributes for entire part),
- macros (combinations of primitives).

Kinetic features were defined as entities that encapsulate energy or motion transfer.

Some other distinctions made in References 7, 10 and 13 between feature types are as follows:

- **Form features**: elements related to nominal geometry.
- **Precision features**: acceptable deviations from nominal form/size.
- **Technological features**: performance parameters etc.
- **Material features**: material composition, treatment, condition etc.
- **Assembly features**: part relative orientation, interaction surfaces, fits, kinematic relationships.

**INTERACTIVE FEATURE DEFINITION**

This methodology involves the predefinition of the geometric model. Therefore, the data structure of the geometric model is a major factor in the design of the definition procedure. Two popular schemes are boundary representation (B-rep) and constructive solid geometry (CSG). In B-rep, an object is modeled by a graph corresponding to a hierarchy of topological entities (faces, edges, vertices), and validated via Euler–Poincaré equations. In CSG, an object is usually modeled as a binary tree whose leaf nodes are half-spaces or primitives, and whose interior nodes are regularized Boolean operators.

**Boundary models**

In the boundary-models method, a 2D/3D wireframe or B-rep solid model is created using a contemporary geometric-modeling package. The database created is then read by a program that renders an image of the part on a CRT to allow the user interactively to pick the topological entities (edges, faces) needed to define a feature. This information can be augmented with attributes such as tolerances and finish, or high-level nominal parameters (such as hole diameter). This approach has been used largely for inputting data to programs for process planning and NC tool-path generation. The work reported in References 2, 14 and 15 can be considered examples of this approach. The 'explicit boundary' feature proposed in Reference 11 also falls into this category, because it is used to group faces from an existing geometric model. R H Johnson’s dimension and tolerance study for CAM-I is also based on this approach.

**CSG models**

With CSG modelers, the above technique cannot be used, because any change to the CSG tree or graph makes the boundary model obsolete. A data structure called VGraph was proposed for the interactive definition of features. The graph contained entities called VFaces (user-defined portions of boundary faces), SFears (surface features, which are groups of VFaces), VEdges (user-defined subsets of an object’s edges), for comparison, it is convenient to classify these methods into three broad groups:

- **Interactive feature definition**: A geometric model is created first, and then features are defined by human users picking entities on an image of the part.
- **Automatic feature recognition**: Here, also, a geometric model is created first, and then a computer program processes the database to discover and extract features automatically.
- **Design by features**: The part geometry is defined directly in terms of features; geometric models are created from the features.

These concepts are illustrated schematically in Figure 1. Each of these categories and their subcategories are discussed in the next section.

**INTERACTIVE FEATURE DEFINITION**

This methodology involves the predefinition of the geometric model. Therefore, the data structure of the geometric model is a major factor in the design of the definition procedure. Two popular schemes are boundary representation (B-rep) and constructive solid geometry (CSG). In B-rep, an object is modeled by a graph corresponding to a hierarchy of topological entities (faces, edges, vertices), and validated via Euler–Poincaré equations. In CSG, an object is usually modeled as a binary tree whose leaf nodes are half-spaces or primitives, and whose interior nodes are regularized Boolean operators.

**Boundary models**

In the boundary-models method, a 2D/3D wireframe or B-rep solid model is created using a contemporary geometric-modeling package. The database created is then read by a program that renders an image of the part on a CRT to allow the user interactively to pick the topological entities (edges, faces) needed to define a feature. This information can be augmented with attributes such as tolerances and finish, or high-level nominal parameters (such as hole diameter). This approach has been used largely for inputting data to programs for process planning and NC tool-path generation. The work reported in References 2, 14 and 15 can be considered examples of this approach. The ‘explicit boundary’ feature proposed in Reference 11 also falls into this category, because it is used to group faces from an existing geometric model. R H Johnson’s dimension and tolerance study for CAM-I is also based on this approach.

**CSG models**

With CSG modelers, the above technique cannot be used, because any change to the CSG tree or graph makes the boundary model obsolete. A data structure called VGraph was proposed for the interactive definition of features. The graph contained entities called VFaces (user-defined portions of boundary faces), SFears (surface features, which are groups of VFaces), VEdges (user-defined subsets of an object’s edges), for comparison, it is convenient to classify these methods into three broad groups:

- **Interactive feature definition**: A geometric model is created first, and then features are defined by human users picking entities on an image of the part.
- **Automatic feature recognition**: Here, also, a geometric model is created first, and then a computer program processes the database to discover and extract features automatically.
- **Design by features**: The part geometry is defined directly in terms of features; geometric models are created from the features.

These concepts are illustrated schematically in Figure 1. Each of these categories and their subcategories are discussed in the next section.

**INTERACTIVE FEATURE DEFINITION**

This methodology involves the predefinition of the geometric model. Therefore, the data structure of the geometric model is a major factor in the design of the definition procedure. Two popular schemes are boundary representation (B-rep) and constructive solid geometry (CSG). In B-rep, an object is modeled by a graph corresponding to a hierarchy of topological entities (faces, edges, vertices), and validated via Euler–Poincaré equations. In CSG, an object is usually modeled as a binary tree whose leaf nodes are half-spaces or primitives, and whose interior nodes are regularized Boolean operators.

**Boundary models**

In the boundary-models method, a 2D/3D wireframe or B-rep solid model is created using a contemporary geometric-modeling package. The database created is then read by a program that renders an image of the part on a CRT to allow the user interactively to pick the topological entities (edges, faces) needed to define a feature. This information can be augmented with attributes such as tolerances and finish, or high-level nominal parameters (such as hole diameter). This approach has been used largely for inputting data to programs for process planning and NC tool-path generation. The work reported in References 2, 14 and 15 can be considered examples of this approach. The ‘explicit boundary’ feature proposed in Reference 11 also falls into this category, because it is used to group faces from an existing geometric model. R H Johnson’s dimension and tolerance study for CAM-I is also based on this approach.

**CSG models**

With CSG modelers, the above technique cannot be used, because any change to the CSG tree or graph makes the boundary model obsolete. A data structure called VGraph was proposed for the interactive definition of features. The graph contained entities called VFaces (user-defined portions of boundary faces), SFears (surface features, which are groups of VFaces), VEdges (user-defined subsets of an object’s edges), for comparison, it is convenient to classify these methods into three broad groups:

- **Interactive feature definition**: A geometric model is created first, and then features are defined by human users picking entities on an image of the part.
- **Automatic feature recognition**: Here, also, a geometric model is created first, and then a computer program processes the database to discover and extract features automatically.
- **Design by features**: The part geometry is defined directly in terms of features; geometric models are created from the features.

These concepts are illustrated schematically in Figure 1. Each of these categories and their subcategories are discussed in the next section.
AUTOMATIC FEATURE RECOGNITION

The input required for many applications programs, such as process planning, NC part programming, Group Technology coding etc., includes both the geometry and features. Various techniques have been developed to obtain this input directly from a geometric-modeling database. This is popularly referred to as feature recognition, although the output of some techniques is not in the form of features, but rather machining volumes. These methods typically assume that all machining will be done by milling, and so it is not necessary to know the specifics of a feature, other than its boundaries corresponding to the final machined surfaces. For example, it does not matter if a machining volume is a rectangular pocket or an L-shaped slot, because tool paths can be generated without this distinction. For this reason, this paper distinguishes between machining-region recognition and feature recognition, and the discussion below is organized accordingly.

Machining-region recognition

Much of the work in this area seems to have been focused on 2-1/2 D milling. Grayer and Parkinson generated NC tool paths from recognized inner and outer boundaries, usually 2D profile curves, by offsetting the curves. Holes have commonly been dealt with separately. Machining-region recognition techniques have been classified here into four categories:

- sectioning techniques,
- convex-hull algorithm,
- cell decomposition,
- geometric reasoning to determine tool paths.

Section techniques

The part volume is sliced with a number of parallel planes that are perpendicular to the assumed tool approach direction. The intersection of the plane and the part model (usually a B-rep model) defines the boundary of the part at the plane. In the sectioning approach taken by Grayer and Parkinson, a boundary representation modeler was used. Bobrow used a CSG modeler. A similar procedure was applied to IGES data. To generate NC code, Grayer decomposed 3D space into slices, and passed each slice into an existing area-clearance program. Parkinson used sectioning to deal with 3D faces (planes not parallel or perpendicular to spindle sections or a general curved surface). The 3D faces were intersected with planes parallel to the xz or yz planes. The intersection curves produced thus were split into straight-line segments so as to keep within a given tolerance. These straight-line segments were offset by the tool radius in the direction of the surface normal at their start and end points, and at a convex edge (concave in the finished part) with a boundary face. Bobrow's technique also generated ordered parametric curves on the surface to be machined by slicing with parallel planes. CAM-I used the sectioning technique for the volume decomposition of complex depressions. This is discussed below.

Convex-hull algorithm

Woo developed the decreasing convex-hull algorithm, in which the volume to be removed was decomposed into machining volumes. Figure 2 shows the convex hull for an object. The difference between the object and its convex hull was computed recursively, until the null set was obtained (i.e. until the object equalled its convex hull). The object could then be represented as a sequence of convex volumes with alternating signs. The decomposition was not always useful, as it could result in a removal volume that did not correspond to a single machining operation (an odd-shaped feature). Also, the base stock was of awkward shape sometimes, because it was the convex hull of the initial shape rather than standard bar stock.

Cell decomposition

Cell-decomposition techniques typically use a spatially enumerated model of the part, or part decomposed into a number of cells. The cells that are to be removed by machining are recognized. Armstrong used this strategy to produce a milled part for a given part and stock geometry. A spatially ordered cell decomposition of the part was produced by a lattice of planes parallel to the major axes, as shown in Figure 3. The cells and infinite planes were then positioned coincident with each planar face, and tangential to each cylindrical halfspace. Each cell was classified as either a stock cell, a part cell, or a semipart cell. To derive roughing cuts, the algorithm selected cells accessible by the tool, and a path was simulated. If no collision with the desired part was detected, the path was concatenated with previous paths. If a collision was detected, the tool was lifted to a safe plane. For finishing, the spatially ordered cells were scanned until one was found to have a face requiring machining. Then adjacent cells were examined for concatenation of that face or linked faces until a path became closed or a boundary was encountered.

Yuen et al. developed an approach that was, in essence, similar to Armstrong's, except that it used octree subdivision for spatial ordering. The octree representation of the product and workpiece was
converted into a quadtree representing a 2D projection. The cutter path was simulated as a zigzag pattern on the plane of the quadtree to remove most of the stock on that plane. Then, the path was simulated as the offset boundary of the product quadtree to remove the cusps. Another quadtree plane was then selected, and the above procedure repeated until the bottom was reached. The final shape produced by this application was not smooth, because the profile corresponded to the shape of the projected quadtree of each layer, which was not the exact shape of the product. Yamanouchi also developed an octree-based method for generating noninvasive machining paths.27 Kanai et al.28 defined a cell geometry by a rational parametric tricubic interpolation formula. This enabled sculptured and analytic surfaces to be represented in a uniform data structure. By a coordinate and a weight value being set in the vertex and control-point block of the structure, the face of a cell could be set not only to an analytic (plane, cylindrical, conical, spherical) surface, but also to a bicubic Bezier surface. Groups of faces in the cell constructed model that could be created by the same cutting operation were collected (by machining-face groups (MFGs)). MFGs were of three levels: Level 1 corresponded to holes, Level 2 to 2D contours or pockets, and Level 3 to 3D surfaces. For Level 1, the subdata structure was a location and a diameter, for Level 2, it was a linked form of 2D rational cubic Bezier-curve segments, and, for Level 3, it was a set of rational Bezier surface patches.

Cell-decomposition techniques are well suited for generating roughing cuts. This is because the part is discretized into cells of a definite shape (usually cuboidal), which results in tool paths that are approximate representations of the boundaries.

Geometric reasoning
Press29 developed a rule-based algorithm that identified individual cutting operations by goal-driven search. Using a boundary representation, the program starts searching at a vertex, and follows an edge to the next connected edge, and so on, until it returns to the original edge. If the procedure succeeds, a profile cut is generated. If not, subloops are traversed for each disjoint section. Thus, the program generates paths for clearing areas and pockets without actually recognizing features.

Feature recognition
Feature recognition differs from machining-region recognition in that portions of the geometric model are compared with predefined generic features to identify instances that match the predefined ones. Specific tasks in feature recognition may include the following:

- searching the database to match topologic/geometric patterns,
- extracting recognized features from the database (removing a portion of the model associated with the recognized feature),
- determining feature parameters (e.g. hole diameter, pocket depth),
- completing the feature geometric model (edge/face growing, closure etc.),
- combining simple features to obtain higher-level features.

Both volume-based methods, such as cavity volumes and delta volumes, and surface-based methods have been devised. It is difficult to classify recognition methods into a clean taxonomy, because there is considerable overlap between the various techniques. Therefore, some key aspects of feature-recognition methods will be discussed instead, with the understanding that a recognition approach could be synthesized by combining the alternatives in various ways. The aspects that will be discussed are:

- matching,
- entity growing,
- volume decomposition,
- recognition from CSG trees.

Matching
Generic features are first formalized in terms of their geometric and/or topologic characteristics. Then, search algorithms are devised to determine which of these characteristics are present in the geometric model (or reconstituted or augmented model). As solid-model data structures are usually graph structures, graph matching has been a popular method for feature recognition. Pure graph matching done on unaugmented solid models amounts to topological matching, i.e. the characteristics are based on the number of entities, topologic type, connectivity and adjacency. If matching were done this way, features of very different semantics would be classified as being the same. Therefore, some subclassification using geometric relationships is necessary. Kyprianou30 devised an entity-classification system based on the magnitude of the angle of intersection. Edges were classified as convex, concave, smooth convex and smooth concave, as shown in Figure 4. Kyprianou's recognition algorithm is discussed.
later, because it falls under syntactic, rather than graph, matching. However, the concept of edge classification has been used widely in augmenting graph models.

Joshi and Chang\textsuperscript{31} used this classification concept in their augmented adjacency graph (AAG). Generic features were defined as AAGs. The nodes of the graph were used to represent faces, and arcs represented edges shared by each set of faces; the arcs were given an attribute value of 0 or 1 to flag the edges as concave or convex. However, this scheme was not designed to capture the specifics of a feature; for example, a straight rectangular slot and a dovetail slot were treated as the same feature. Sakurai and Gossard\textsuperscript{32} also used graph matching, but with a richer set of possible characteristics to describe features. In addition to the number of faces, face connectivity, and edge/face classification, they also used checks on geometric information, such as parallelism, perpendicularity and face geometry.

Another technique that has been applied to matching is syntactic pattern recognition, which is popular in vision systems. In these systems, geometric patterns are described by a series of, typically, straight, circular, or other curved-line segments. Simple patterns can be concatenated to give compound patterns. Languages have been developed for describing these sequences algebraically and manipulating them with operators that form a grammar. Features can be recognized by parsing the feature against the object’s description in the grammar. Syntactic pattern recognition was applied to features by Kyprianou\textsuperscript{30} and by Choi\textsuperscript{33}. Jakubowski et al.\textsuperscript{24} and Staley et al.\textsuperscript{35} used strings of straight lines and curve segments to recognize 2D profiles of holes.

Kyprianou’s algorithm\textsuperscript{30} was based on a shape grammar that used convex/concave classification (depicted above in Figure 4). Edges were classified as convex, concave, smooth convex and smooth concave. Smooth edges were reclassified as concave/convex on the basis of local curvature. Vertex classification was based on the vertex’s incident edges; if two or more incident edges were concave, the vertex was deemed to be concave; otherwise, it was classified as convex. Loops were classified as convex (all edges convex), concave (all concave) and hybrid (mixture). Faces of the object were marked primary if they contained a concave edge or an inner loop, and primary faces were ordered on the basis of the number of concave edgesets. A hierarchical faceset data structure was built by processing the geometric model, which by then contained entities tagged by the above classification. The feature grammar was used to determine features from rules and faceset data structure. A metalanguage was developed for specifying GT schemes\textsuperscript{36}; the faceset data structure was integrated to derive the GT code.

Pinilla, Finger, and Prinz\textsuperscript{37} also developed a graph grammar to describe objects in nonmanifold representations. The grammar was applied to a restricted set of features in the injection-molding domain. Objects were represented by topological graphs augmented by geometric information. Recognition amounted to the identification of a subgraph in the object’s augmented topology graph that could be generated by a grammar associated with the feature.

Henderson\textsuperscript{4} used concepts from both graph matching and syntactic pattern recognition to devise a rule-based system for ‘template’ matching. Features were formalized by templates that consisted of pattern rules. Templates were defined for both general features (such as holes) and specific features (e.g. flat-bottomed, constant-diameter hole). A general rule for a hole, for example, was used to encode the following generic properties of a hole:

- The hole begins with an entrance face. All subsequent faces of the hole share a common axis. All faces of the hole are sequentially adjacent. The hole terminates with a valid hole bottom.\textsuperscript{4}

Rules such as these were expressed as a set of both geometric and topologic conditions, each of which had to be tested separately; all conditions had to be satisfied for the rule to be satisfied. Henderson created, in effect, a graph grammar for feature recognition. The recognition and extraction algorithm involved the following steps: determine cavity volume (difference between stock and part), recognize general features in each cavity, classify general features into specific features, create and subtract the volume corresponding to each feature from the cavity, and repeat all the above steps until there are no residual cavities. The scope of the study was limited to sweep features.

Entity growing
In many feature-extraction algorithms, recognized features are removed by adding/subtracting a volumetric shape that corresponds to the feature. As the
Figure 5. Edge extension to form a feature volume

A recognized feature is not always a closed volume; new faces may have to be added to close the feature volume. This is referred to as entity growing. Some methods use face extension, and others use edge extension. In both cases, new topological entities are also created by intersections. In the method developed by Falcidieno et al.,

edges of faces are extended to generate volumes, as shown in Figure 5; this also creates new edges and vertices.

Sakurai and Gossard created feature volumes by adding half spaces corresponding to feature faces. The negative of the volume was extracted from the object to continue feature recognition. An example is shown in Figure 6.

The problem with both of the above methods is that only convex volumes can be created. These methods do not work for a pocket with an undercut, for example. Dong and Wozny devised a face-extension method that overcame this problem to some extent. The neighboring faces of a feature were intersected to generate new vertices and edges. New faces were constructed by traversing the edges. If multiple loops on an edge graph were found, some vertices were eliminated because they did not lie on the face boundary.

CAM-I's volume-decomposition work, which predates Reference 39, recognized two problems with surface extension. First, it discovered that many volumes could not be created by extending existing faces, because new faces need to be added. Second, it found that a systematic approach for face extension could not be devised for the general case. For these reasons, the CAM-I work used face extension only for 'primitive' features, which were simple shapes, such as holes, and used sectioning for complex shapes. The CAM-I volume-decomposition algorithm is discussed in the next section.

**Volume decomposition**

The purpose of volume decomposition is to identify material to be removed from a base stock, and to break down this volume into units corresponding to distinct machining operations. This technique is classified as a feature-recognition technique, because it distinguishes between the types of features that need to be produced by machining. Generally, the total material volume to be removed by machining is found by a Boolean difference between the stock and the finished part. This volume must then be decomposed into units that correspond to practical machining operations that match machining features, as shown in Figure 7 for the example part. A well known work on volume decomposition is that done by General Dynamics for CAM-I. The purpose of the project was to achieve a high degree of automation for generating NC programs for parts defined by 'noncomplex' surfaces (planar, quadric and cylindrical). An algorithm was developed for operating on a B-rep model of the total volume to be removed, augmented with tool-accessibility codes for each face. A library of generic delta volumes existed in the system; new delta volumes could be added by users. This set of generic delta volumes was required to meet the criteria for completeness and richness, as

Figure 6. Half-spaces method of volume creation

Figure 7. Part volume decomposed into its machining features

[Delta volumes, showing accessibility with solid arrows and connectivity with broken arrows.]
discussed below.

Subcategories of the design-by-feature approach are
and location parameters and various attributes. Three
Generic feature definitions are placed in a library from

These two criteria guaranteed that any machining
volume could be decomposed into a set of generic
delta volumes. Decomposition was carried out in two
major stages. First, the primitive (parameterizable)
 volumes were recognized and extracted by surface
extension. Because surface extension could be done
by interrogating the B-rep model, considerable
computations were saved. Complex depressions
(2-1/2 pockets) were recognized by sectioning. A set
of crosssections was constructed by using a set of
planes that were perpendicular to the cutter axis. Then,
relationships between adjacent crosssections were
determined to decompose the volume into disjoint
‘super-delta volumes’, which were decomposed further
based on tool accessibility. All delta volumes had to
have at least one accessible face. A face on a delta
volume was inaccessible if it coincided with a face of
the finished part. If a face was partly coincident
with the finished part, it was assigned connectivity to another
delta volume with which the rest of the face was
coincident. Finally, all delta volumes were compared
with generic delta volumes and classified.

Recognition from CSG trees
Work in the area of recognition from CSG trees is
limited to contributions made by only a few studies.40,41
A major problem is the nonuniqueness of CSG trees
representing an object. Lee et al.40 considered the
primitives in the CSG model to be represented by their
principal axes in local coordinates. These principal axes
were collected and clustered according to spatial
relationships. Features were then extracted based on
conditions predefined for the feature. For example, the
definition of a rounded edge involves the positioning
of two cubes and a cylinder. Features were located in
this manner, but the nonuniqueness was not addressed.
The main contribution of these projects40,41 was to
provide a basis for manipulation of the CSG tree. Once
the features were located, the feature representations
were unified by grouping nodes into a subtree,
and replacing subtrees with equivalent subtrees. An
algorithm41 presented to support these functions details
moving a node up one level in the CSG tree.

DESIGN BY FEATURES
In the design-by-features approach, features are
incorporated in the part model from the beginning.
Generic feature definitions are placed in a library from
which features are instanced by specifying dimension
and location parameters and various attributes. Three
subcategories of the design-by-feature approach are
discussed below.

* Pro-Engineer is a product of Parametric Technology.
Point handles were used for positioning and orienting features and for establishing relationships between two or more features. Line handles were used to represent vectors, which included information such as depth of hole or length of slot. Position tolerances were associated with position vectors. The position of a feature and tolerance stackups were derived by procedures that traced through the hierarchical chain of reference features.

**Synthesis by features**

Systems that allow one to design by adding or subtracting features without a starting base stock are discussed in this section. The concept of ‘implicit’ features of Pratt and Wilson comes under this category, as do prototype systems discussed in References 6, 9 and 45–49 and the commercial system Cimplex, formerly from ATP (now Cimplex Inc.). Generic features are predefined in terms of rules and procedures. Procedures may include methods for instancing, modifying, copying, deleting features, generating solid models, deriving certain parameters, and validating feature operations. For example, Chung developed a prototype system for creating solid models in GENMOD. The ASU Features Testbed used a parallel representation: a boundary model described in Reference 50, and a constructive model that represented the volume or cavity corresponding to the feature (referred to as the feature-producing volume (FPV)), with a union or difference operator. This solid representation was not specific to any particular solid modeler; when an ‘evaluate’ command was issued, all FPVs were combined, and commands were translated through an interface to solid-modeler-specific commands.

The features stored in libraries may be application-oriented: Reference 6 defined casting features, Reference 51 defined injection-molding features, and Reference 46 provided features needed in gating design for investment casting. Reference 46 supported a general-purpose template for adding new features to the library. The ASU Features Testbed consisted of two shells, one for part design and the other for mapping and applications. The shells contained mechanisms for defining generic features, adding features to the library, and using features in design. This allows user organizations to customize the system, avoiding the difficulty of working with a hard-coded set of features.

In Reference 47, features were positioned in a global coordinate system, while, in most others, features were positioned relative to other features or geometric entities. Shah et al. developed an inheritance mechanism for creating a network for determining dependent parameters from attributes of other features or algorithmic procedures. The advantage of this method is that, when a design change takes place, the change is propagated to all affected features.

Ostrowski presented the conceptual design of a system that supported both human-interactive feature definition and design by features using CSG models. Definitions were based on Requicha and Chan’s set operators described in Reference 36 but without the VGraph structure. In addition to face and edge features, methods for volume and vertex features were also supported. All features were defined by an entity called ‘feature type (BoolOp, FPV, PV)’ where FPV and PV were CSG trees representing the feature-producing volume and part volume, respectively. BoolOp was a Boolean set operator that could take on values of union, difference or intersect (similar to the in, out, on of Requicha and Chan). Features were defined as a list of parameters, primitives, FPV, PV, BoolOp, Feature Class and Rules. They were then stored in a library from which they could be instantiated. Requicha also extended his CSG-based method to support design by features through the use of feature operators (insert, delete, modify). An ‘expander’ program generated the necessary CSG-tree manipulations. Requicha et al. propose to exploit the spatial-locality concept developed by Tilove to perform validation checks on feature operations.

**FEATURE REPRESENTATION**

Features may be represented at various levels. For example, one could represent them by the process by which they may be created or by the resultant geometric model. The former representation has been termed implicit or unevaulated, and procedural; the latter has been termed explicit or evaluated. Features may be defined more abstractly as a neutral description without any specification of how the feature is to become part of the geometric model. Explicit representations have commonly been used in interactive and automatic feature recognition and explicit representations in design by features.

Some of the structures used for geometric representation of features are:

- augmented graphs,
- algebraic (syntactic),
- delta volumes,
- constraint-based B-rep,
- point accessibility codes.

Augmented graphs are usually based on face adjacency; the areas of the graph are attributed with information on edge classification and geometric relationships. Some examples of these were discussed above in the feature-recognition section. Syntactic languages have also been devised that encapsulate adjacency, connectivity, geometric orientation and convexity/concavity of feature entities, although their use has been limited to 2D. Delta volumes are complete B-rep models of closed spaces associated with tool-accessibility codes for faces and connectivity information. Figure 7 shows accessibility and connectivity codes, for example delta volumes.
Constraint-based B-rep schemes have been devised by several researchers independently⁶,⁸,⁵⁹, and applied recently⁷,⁶⁰. Geometric constraints of feature size, shape, orientation and position were accommodated in a tree structure based on faces. The interior nodes were Boolean intersect operators or geometric transformations, and leaf nodes were faces or sets of faces. Faux⁶² used four sets of rules: structuring rules to define connectivity of faces, selection rules to define permitted classes, combination rules to define implicit constraints, and sizing rules to define explicit constraints. Roy and Liu⁵⁹ represented objects as structured face adjacency graphs (SFAG). The graph was a hierarchy of FAG representations; the top level represented the final object, and lower ones the unfinished object. There were pointers from SFAG to primitive faces. Burchard⁵⁵ also used constraint-based modeling, but with CSG trees instead. Reference 60 cast this structure in an object-oriented form; the characteristics of object-oriented modeling are discussed in the next paragraph.

Object-oriented implementation is now quite common for feature-based modeling.⁶,⁸,⁹,⁴⁴,⁶⁰ Knowledge about generic features is clustered into a unit called an 'object'. Knowledge may be in the form of parameters, rules, procedures etc. This allows features to be handled in a uniform way for creation, deletion and manipulation, while differences between the features are taken care of by the rules and procedures embedded in a features definition. The clustering of feature-specific knowledge within each feature object allows the features to handle and manage changes to themselves internally, while the global operations are uniform. Objects are never concerned with internal operations on other objects, although changes may be requested through the proper protocol. Object-oriented implementation also exploits similarities between members of feature families by using property inheritance.

A 4-level data structure was used in Reference 9, and is shown for an abstract object in Figure 8. The top level records feature relationships, followed by generic properties of each feature at the second level, instance parameters at the third, and feature-producing volumes (solid model) at the lowest level.

**FEATURE VALIDATION**

There are no universally applicable methods for checking the validity of features. It is up to the person defining a feature to specify what is valid or invalid for a given feature. Typical checks that need to be done are: compatibility of parent/dependent features, limits on dimension, and inadvertent interference with other features⁹,⁴⁸. In a study for CAM-I⁵⁵, Shah et al. enumerated the following types of feature interactions:

- interaction that makes a feature nonfunctional,
- nongeneric feature(s) obtained from two or more generic ones,
- feature parameters rendered obsolete,
- nonstandard topology,
- feature deleted by subtraction of larger feature.

---

---

**Figure 8. 4-level data structure of a feature model**
• feature deleted by addition of larger feature.
• open feature becomes closed.
• inadvertent interactions from modifications.

Such interactions can be problematic, both in design by generic features and topology-based feature recognition. Requicha\textsuperscript{49} has examined validation from a machining viewpoint. They require that features satisfy rules of the following types:

• presence rules: nonnull portion of feature present in the part volume,
• nonintrusion rules: producible without gouging,
• accessibility rules: approachable by the appropriate cutter,
• dimensional rules: guarantee that dimensions are manufacturable.

FEATURE SPACES

Feature sets depend not only on product type (casting, sheet metal, electronic-circuit board etc.), but also on a person’s viewpoint or engineering task (design, finite-element analysis, process planning etc.). Therefore, there exists no unique set of features. However, for a given product type and given application, one may be able to devise an adequate set of features that serve one’s purpose. Such a set is termed a feature space. Feature spaces may overlap or be disjoint. To create an integrated feature-based system, one must provide both feature extensibility and mechanisms for mapping features between features spaces. A review of this subject is given in Reference 13.

DISCUSSION

The human-assisted definition method is easy to implement, and it can work off IGES or modeler-specific databases of contemporary systems. Only features needed for an application (e.g. process planning) need be identified. For models containing a large number of features, this method can be time-consuming. In many current implementations, the burden of picking valid entities lies on the user. The job can be made easier with some changes; for example, if the user indicates that he/she wants to identify a flat-bottom hole, the system prompts that he/she must pick a cylindrical face and a planar face; when the entities are picked, the system will check that they are of the appropriate type. Procedures must exist for automatically deriving the diameter and depth of the hole from the geometric model. The number of topological entities is arbitrary, and often depends on intersections performed in the construction of the model. For example, the cylindrical surface of the hole may be represented by several faces. Several systems allow one to merge topological entities lying on the same geometric entity, and so this will always be a prerequisite.

Considerable progress in feature recognition has been made. Principal among the advantages of feature recognition is the use of geometric-modeler databases, or even IGES (one system\textsuperscript{54} was based on IGES 2.5). Another advantage is that recognition can be made application-specific, allowing each application program to have its own recognition program. More work is needed in handling interacting features.

Sectioning techniques suffer from many inherent problems, but they are still the techniques most commonly used. They are successful with simple 2-1/2D parts that do not have any undercut portions. The presence of undercuts, inclined surfaces and nonplanar surfaces cause complications. When many features occur in the same plane, each feature is machined before movement to the next plane. This yields nonoptimal tool paths. Also, the slicing planes must be chosen appropriately, such that critical sections (i.e. portions where the crosssection changes drastically) are not omitted.

Tree manipulation suffers a drawback owing to the nonuniqueness of CSG. Convex-hull decomposition algorithms often do not produce a usable decomposition, because this can result in removal of volumes that do not correspond to a single machining operation (an odd-shaped feature). Also, the stock shape can be awkward because it is the convex hull of the initial shape. Only a handful of researchers have worked on tree manipulation or volume decomposition. Most of the work has been in the area of graph or pattern matching to locate features.

Woodwark\textsuperscript{64} provided some insight into difficulties faced in feature recognition. He observes that complicated curves and disjoint features in particular have presented major problems in feature recognition with B-reps. However, some advances have been made on these fronts in recent years. For example, Reference 31 developed methods to deal with disjoint features in a restricted domain. Woodwark\textsuperscript{64} suggested that the set-theoretic approach may be better suited to feature recognition, in spite of its nonuniqueness of representation. He proposed that set-theoretic models be first simplified by restricting the range of primitives and orientations, the ways that the primitives may interact, and allowable forms of set-theoretic expressions. In the author’s opinion, such restrictions will not be acceptable to most designers. A second proposal found in Reference 64 was to use templates for matching to the model. This has also now been tried in a limited way in Reference 40.

Design by features has the advantage that it allows designers to transfer to the database much of the information available at the design stage. This richer and higher-level database is available for use by downstream applications. It is even possible to implement real-time manufacturability evaluation and concurrent design and process planning. However, the set of features used in design is not finite. One needs to determine how many features should be contained in the feature library, and at what level of abstraction. Also, as features are application-specific, the need for feature recognition by application does not go away when one designs by features. Finally, interactions between features can result in nongeneric shapes that do not exist in the database, or they could make some generic dimension values obsolete.

Creating feature databases unaided by geometric modelers has the disadvantages that nonsense geometry can be created, and the user cannot visualize the
geometry. In DSG, one destroys the stock model by removing features, which is a good way of providing input to CNC machines. However, the method requires the stock shape to be known a priori, and it puts process-planning responsibility on designers. Features depend on the product type (sheet metal, injection molding, machined part etc.) and on engineering applications (design, finite-element analysis, process planning). The same feature may have multiple meanings across applications. Designers will not necessarily use features that correspond to manufacturing operations, thus the feature model produced by design will not explicitly represent features needed by downstream applications. For the most part, researchers working on features have avoided this problem of mapping by concentrating on feature-based systems to support only one application and product type.

Feature mapping is a critical area for the success of feature-based systems. Many other problems must be resolved, including agreement on the definition of features, determination of the role of features in mechanical design, and data-exchange standards for features. The problems with interacting features and feature validation need further investigation. Also, transformations between varying levels of abstraction need to be better understood.

ACKNOWLEDGEMENTS

The author gratefully acknowledges the financial support received from the Geometric Modeling Program of Computer-Aided Manufacturing International (CAM-I) (Contract # C-88-GM-01) and the US National Science Foundation (Grant DMC-8713544). The paper represents part of the work done under the above contract and grant. A more detailed review of features technology can be found in the author’s report to CAM-I. The author also acknowledges the work of the following graduate students in reviewing some of the relevant literature: Mary Rogers, Palat Sreevalsan and Abraham Mathew. Special thanks are due to Palat Sreevalsan for the preparation of the manuscript of this paper.

REFERENCES

1 Grayer, A R ‘A computer link between design and manufacture’ PhD Dissertation University of Cambridge, UK (Sep 1976)
4 Henderson, M R ‘Extraction of feature information from three dimensional CAD data’ PhD Dissertation Purdue University, USA (May 1984)
9 Shah, J and Rogers, M ‘Functional requirements and conceptual design of the feature-based modeling system’ Comput.-Aided Eng. J. (Feb 1988)
12 ‘PDES Form Feature Information Model (FFIM)’ PDES Form Features Group (Jun 1988) (Coordinator: Mark Dunn)
14 Chang, T C ‘TIPPS — a totally integrated process planning system’ PhD Dissertation Virginia Polytechnic Institute, USA (Nov 1982)
16 Requicha, A A G and Chan, S ‘Representation of geometric features, tolerances and attributes in solid modelers based on CSG’ Technical Memo # 48 Production Automation Project, University of Rochester, NY, USA (Oct 1985)
17 Marisa, R ‘Proposed extensions to PADL-2’ Technical Note COMAPP, Cornell University, USA (May 1988)
18 Grayer, A R ‘The automatic production of machined components starting from a stored geometric description’ in McPherson, D (Ed.) Advances in Computer Aided Manufacture North Holland (1977) pp 137–150
21 Kumar, B, Anand, D K and Kirk, J A ‘Integration and testing of an intelligent feature extractor within
a flexible manufacturing protocol' 16th NAMRC Conf. University of Illinois, Urbana–Champaign, USA (May 1988)


23 Woo, T C ‘Feature extraction by volume decomposition’ Proc. Conf. CAD/CAM Technology in Mechanical Engineering Cambridge, MA, USA (Mar 1982)


30 Kyprianou, L ‘Shape classification in computer aided design’ PhD Dissertation University of Cambridge, UK (Jul 1980)


43 Arbab, F ‘Requirements and architecture of CAM oriented CAD systems for design and manufacture of mechanical parts’ PhD Dissertation University of California at Los Angeles, USA (1982)


45 Burchard, R L ‘Feature based geometric constraints applied to CSG’ MS Thesis Purdue University, USA (May 1987)


50 Liou, B ‘Pseudo boundary model and feature interface for design by features’ MS Thesis Dep. Mechanical & Aerospace Engineering, Arizona State University, USA (Jul 1988)


61 Ansaldi, S, De Floriani, L and Falcidieno, B ‘Geometric modeling of solid objects by using a face adjacency graph representation’ ACM Siggraph Vol 19 No 3 (1985)


65 Bhatnagar, A ‘Design and implementation of feature mapping shell with application to GT classification’ MS Thesis Dep. Mechanical & Aerospace Engineering, Arizona State University, USA (Jun 1988)


BIBLIOGRAPHY
