INTEGRATION OF GEOMETRIC MODELING AND ADVANCED FINITE ELEMENT PREPROCESSING

By

Mark S. Shephard
Rensselaer Polytechnic Institute

and

Peter M. Finnigan
General Electric Corporate Research and Development Center

ABSTRACT

The structure to a geometry-based finite element preprocessing system is presented. The key features of the system are the use of geometric operators to support all geometric calculations required for analysis model generation, and the use of a hierarchic boundary-based data structure for the major data sets within the system. The approach presented can support the finite element modeling (FEM) procedures used today as well as the fully automated procedures under development.

1. INTRODUCTION

The generation of numerical analysis models is one of the major steps in the computer-aided engineering (CAE) process. Of primary concern is the disproportionate amount of the entire design/analysis process that is currently dedicated to this task. If significant productivity gains are to be achieved in CAE, this bottleneck must be reduced. In the long term, this means the automation of the entire finite element process, which would include such things as adaptive analysis and optimization techniques. In the short term, this means improving the basic model generation tools and developing preprocessing systems that employ advanced geometric modeling and more powerful data structures. This paper presents the overall design of a geometry-based FEM system that will address today's needs, as well as provide a foundation for the fully automated systems of tomorrow.

The most obvious reason for a better integration of finite element and geometric modeling is to avoid the need to redefine the geometry during finite element modeling. The second is to make more direct use of the functionality present in advanced geometric modeling systems. The third reason is the creation of a more unified design/analysis environment that employs a geometry-based (object) problem description. This includes geometry-based analysis attribute specification, which is not only a necessary part of an object-based problem definition, but is the most efficient method of prescribing this information. Finally, it is only with the close integration of geometric and finite element modeling procedures, that FEM can be fully automated.

In addition to the requirements placed on a preprocessing system by the above needs, it must give the analyst all the model generation functions needed to efficiently create controlled element meshes. At this time, that requires the system to support bottom-up, mapped, and fully automatic mesh generators such that they can be used in various combinations in a consistent manner. This also means that flexible, geometry-based mesh control specification techniques

be used.

The majority of the current finite element preprocessing systems have been developed in an evolutionary manner, independent of geometric modeling systems. Interfacing, not integration, results between the two systems. With the recent advances in geometric modeling procedures, there is a desire to make more direct use of the geometry available for the generation of the finite element model. To date, there has been limited success in integrating geometric and finite element modeling. The major factors deterring this integration are:

1. Generally, finite element preprocessing systems are designed to construct a finite element model by directly building the object's description in terms of topologically simple shapes (i.e., triangles, quadrilaterals, tetrahedrons, hexahedrons, etc.). This approach is not well suited for general geometric manipulation.

2. The data structures within a finite element preprocessor are designed to house little more than the most basic of mesh construction information and the mesh data (i.e., node point coordinates and element definitions). They do not possess a general geometric data structure to house the original geometric definition of the object, nor do they maintain relationship information which explicitly couples the finite elements themselves to the geometry from which they came.

3. The geometric modeling systems do not make their intrinsic geometric manipulation capabilities readily available to other applications on the system which require such functionality.

The majority of effort that has been expended on improving the level of integration between finite element and geometric modeling has been aimed at particular modeling systems. The finite element preprocessing developers have added their own geometric modeling capabilities, or the geometric modeling developers have extended their systems to include finite element model generation. Although these approaches represent improvements, they tend to lack generality and represent a large duplication of software development effort. In addition, they typically lead to systems that have lopsided strengths; such as geometric modeling systems that are well suited for developing finite element models but poor for other applications. The position taken in this paper is that the developers of geometric modeling systems should concentrate on providing advanced geometric modeling functionality, FEM developers should concentrate on the advancement of finite element modeling, and that these two groups work together to integrate their respective capabilities in a cohesive manner.

A general integration of geometric and finite element modeling requires not only the transfer of data but also the transfer of functionality from the geometric modeling system to the finite element modeler. This paper presents a methodology that addresses this need. It must be noted that the implementation of these methodologies requires a major expansion of the data structures underlying the finite element modeling system, the strict adherence to prespecified operators to interact with a geometric modeling system, and the construction of those operators. The successful implementation of such a

system depends on:

1. Finite element modeling developers recognizing the need for change.

2. Finite element modeling developers working closely with geometric modeling developers to better understand each others requirements and limitations.

3. Geometric modeling developers providing the requisite modeling functionality for FEM.

Section 2 indicates an approach to the modular integration of geometric and finite element modeling. Section 3 indicates the type of data structures required in a geometry-based finite element modeling system. Section 4 gives a more specific indication of how the various finite element model generation procedures would operate within such a system. Section 5 addresses the remaining open questions of the development of a geometry-based finite element modeling system.

2. APPROACH TO MODULAR INTEGRATION WITH GEOMETRIC MODELING

The first key to the development of a finite element modeling system that can be efficiently integrated with various geometric modeling systems is the use of a general data structure that can support various geometric forms. The second key aspect of this integration is the use of a general set of geometric communication operators [1]. A geometric communication operator is a procedure designed to perform a geometric function, given specific information about the operation and the geometry involved. The operator would return the result of the operation and/or modify the geometric representation to reflect the invocation of the operation.

The information passed directly to the geometric communication operator has a general structure. Any data, specific to a particular geometric modeling system, would be extracted directly from its geometric database by a geometric communication operator. Therefore, it is only necessary to provide a set of geometric communication operators for each new geometric modeling system that the finite element modeler is to be interfaced. No changes within the finite element modeling system need be applied. The approach discussed in this section is consistent with the CAM-I work on an applications interface for geometric modeling [2,3]. The advantage of this approach is obvious; it avoids the need to reproduce all the geometric modeling functionality within the finite element modeling system. This advantage is absolutely necessary if finite element modeling systems are to be interfaced with the various forms of geometric modelers being developed. Once a set of geometric communication operators are agreed on, the operators can be constructed by the developers of a geometric modeling system. Hopefully, the majority of them can be extracted directly from the modeling capabilities already present in the system.

The geometric communication operators needed for finite element modeling can be grouped into the following five categories:

1. BASIC QUERY - A request for geometric information that is intrinsically a part of the geometric modeling database.

2. DERIVED DATA QUERY - A request for geometric information not directly stored in its database. The determination of the requested data requires the performance of geometric calculations. A DERIVED DATA QUERY will not alter the contents of the geometric modeler's database.

3. GEOMETRIC MODELING OPERATION - A request is made that invokes one or more geometric modeling operations such that the geometric model is altered in the process.

4. ATTRIBUTE SPECIFICATION - The geometry-based specification, modification, or deletion of the model's attributes.

5. GENERAL UTILITY - These would contain the operators that request general, geometry independent information on a model.

A large number of geometric communication operators are needed for finite element modeling. They will most likely be built in two levels. The low level operators will represent atomic geometric operations such as Euler operators [4] or specific Boolean operations [5]. The higher level operators, those oriented toward finite element modeling, would be constructed primarily from the low level operators. This approach has the advantage of insulating the FEM system from changes in the geometric modeling system because only the internals of the higher level layer would be affected. Thus, the FEM system could be interfaced to a new geometric modeling system with relative ease assuming it provided, in some form, the low level operators. It should be noted, that since the type and amount of information stored in a geometric modeler's database is a function of modeling approach and implementation, an operator that is a BASIC QUERY in one system may be DERIVED DATA QUERY in another. A few example operators are listed below:

BASIC QUERY

1. RETURN_GEOMETRY_COEFFICIENTS - return the coefficients used in the definition of a geometric entity.

2. RETURN_TOPOLOGICAL_ENTITY - return the definition of the requested topological entity.

3. RETURN_ENTITY_ASSOCIATIVITY - return a specific set of associativities for a topological entity.

DERIVED DATA QUERY

1. DETERMINE_DISTANCE_BETWEEN - calculate the minimum distance between two geometric entities.

2. POINT_CLASSIFY - determine if a given point is inside the object, outside the object, or on the surface of the object.

3. DETERMINE_INTERSECTIONS - calculate the intersections between two geometric entities.

GEOMETRIC MODELING OPERATION

1. ADD_ENTITY - add a given entity to the geometric model.

2. SPLIT_ENTITY - break a given entity into multiple entities in a prescribed manner.

3. PERFORM_BOOLEAN - carry out a Boolean operation between two specified entities.

ATTRIBUTE SPECIFICATION

1. PLACE_ATTRIBUTE - apply an attribute to a given entity.

2. MODIFY_ATTRIBUTE - modify a given attribute on a given entity.

GENERAL UTILITY

1. GET_MODEL - retrieve a given model form the database.

2. SAVE_MODEL - store the current model in the database.

Although the concept of geometric communication operators represents the most general method to tie the functionality of geometric modeling systems to geometry-based applications, the geometric modeling systems available today do not fully support this concept. This is expected to change over the next few years. The move to more open architectures, the increased pressures from applications, a maturing of geometric modeling systems, and specific research on the creation of such operators are contributing to this change. Developers of application software should expect the availability of specific sets of low level operators in the near future.

3. DATA STRUCTURES IN A GEOMETRY-BASED PREPROCESSOR

This section first provides an overview of the data structures required in a truly geometry-based preprocessor before discussing the details of a particular implementation. It is important to understand, in very general terms, the data structures themselves and how they interact.

There are a number of possible ways to group the requisite preprocessing data. The one listed below was selected because it uses the minimal number of data sets that provide a logical separation of information needed for finite element modeling. The data sets include:

- The MODEL data set
- The ATTRIBUTE data set
- The MESH data set

The MODEL data set contains the geometric and topological data that defines

the domain to be meshed. The ATTRIBUTE data set contains both the analysis attribute data (e.g., material properties, boundary conditions, etc.) and the mesh control data. The MESH data set contains the finite element mesh generated for the model. Each data set has its own structure tailored to meet its special requirements. The data structures are related through a well defined set of pointers which provide the mechanisms through which all non-MODEL data is tied to the MODEL data. The information content of these data structures and their relationships are described in more detail in the sections which follow.

3.1. Model Data Structure

The most fundamental data to the preprocessor is the geometry. The work described in this paper uses the concept of a non-manifold geometric modeling topology representation [4]. In a manifold representation, the area surrounding any point on a surface (in the limit) is "flat." In a non-manifold representation, the "flatness" criterion is not a requirement.

Historically, solid modeling systems have employed manifold representations. However, this has caused problems when non-manifold results would occur as a natural part of the model building process. One major benefit of a non-manifold representation is that it permits wire frame, surface, and solid models to coexist in the same system concurrently. Relative to finite element modeling, it appears that the developers have not appreciated the importance of a well defined topological model and that the topology which does exist in these systems has been evolutionary. Not unexpectedly, they are inadequate to support major advances in automation.

There is a close parallelism between finite element modeling and geometric modeling with the three representations. That is, because of the abstractions associated with FEM, all three geometric representations may be necessary simultaneously. For example, many "real world" models are typically comprised of a combination of solid, shell, and beam elements. Although not exclusively, these element types lend themselves to being modeled with the "corresponding" geometric representations. That is, the shell portion of the model with a surface representation, the beam portions with a wire frame representation, and of course, the solid elements with a solid representation. The logical conclusion is that a non-manifold data structure that can support the three forms of geometric modeling, can also support the geometric aspects of finite element modeling. A single representation opens the possibility for the finite element modeling system to make direct use of geometric operators developed in support of the geometric modeling system.

The non-manifold geometry/topology hierarchical model used is depicted in Figure 1. In addition, the relationships between geometry and topology are shown.

3.1.1. Model Data Definitions

This section provides a set of working definitions for the various geometric and topological entities used for geometric modeling. In addition to the basic definition being stored in the data structure, its origin or purpose will also be stored. That is, if an entity originates from the geometric

modeler, or if an entity is added for the purpose of controlling the mesh or applying a boundary condition, it will be so identified.

.1.1.1. Geometric Entities

There are four geometric entities which the system will support as defined below:

- POINT -- A point is a geometric entity specified by a triple of numbers representing its position in space.
- CURVE -- A curve is a geometric entity representing the trajectory of a point through space. Curves can be infinite in extent.
- SURFACE -- A surface is a geometric entity representing a two-dimensional locus of points. Surfaces can be infinite in extent.
- VOLUME -- A volume is a geometric entity representing a three-dimensional locus of points.

It should be noted that in most geometric modeling systems, volumes in space are defined in terms of a set of surfaces that enclose it. It is, however, desirable to support the possibility of a specific volume geometric entity which adds the ability to house volumes with internal definitions in the system [10,11].

3.1.1.2. Geometric Modeling Topological Entities

The topological entities form a hierarchy which, when coupled with geometry, provide a complete definition of the part. A brief description of each of the geometric modeling topological entities to be used in the model data is given below.

- VERTEX -- A vertex is the topological equivalent of a three-dimensional point in space. It is typically used to bound an edge. There is always a vertex at the joining of edges. Vertices may also be used as a boundary of a face or shell.
- EDGE -- An edge is the topological equivalent of a geometric curve (i.e., line, arc, spline). It is bounded by two vertices. An edge may be closed, in which case the starting and ending vertices are the same.
- LOOP -- A loop consists of an ordered, closed, connected, set of edges. A loop bounds a face.
- FACE -- A face consists of a portion of a shell. A face is bounded by at least one loop, and may be internally bounded by further interior loops (i.e., holes).
- SHELL -- A shell consists of a set of faces which bound a region. A shell may consist of a connected set of faces which form a closed volume or may be an open set of adjacent faces, a wire frame, a combination of these, or even a single point. In the case of a solid model,

one shell is required to define the external boundary and additional shells are required to define voids within the solid.

- REGION -- A region is a volume of space. A region has one exterior shell and one interior shell for each void contained within it.

- MODEL -- A model is a collection of regions. Regions within a model may be distinct because of physical separation in space, or simply because a user wishes to keep them logically distinct.

3.2. Attribute Data Set

Any form of numerical analysis, requires the following information:

1. A complete specification of the physics of the problem to be analyzed.

2. Specification of the desired level of domain discretization.

3. The specification of the required analysis control parameters.

In general terms, this information is referred to as the attribute data for the model.

The attribute data structure will contain all of the information, past the geometric definition of the object, that is needed to complete the description of the problem. Attribute data includes both geometric and non-geometric information. Data which is geometric in nature must be tied to the original geometric definition of the object.

A number of different modeling attribute types are needed for finite element analysis. A partial list of these includes:

1. Analysis program control data.
2. Case information.
3. Finite element type declaration information.
4. Nodal (skewed) coordinate system data.
5. Material property data.
6. Physical property data.
7. Mesh control data
8. Essential boundary condition data

    - e.g., displacements in a stress problem or temperatures in a heat conduction problem.
9. Natural boundary condition data

    - e.g., pressures in a stress problem or fluxes in a heat conduction problem.
10. Initial condition data.

3.2.1. Classes of Attributes

Finite element modeling attributes can be categorized by class, depending on how they interact with geometry. Three distinct classes of attribute data have been identified, and are described below:

CLASS #1:

Attribute data in class #1 is characterized by data which is required for the analysis to be performed but which it totally independent of the geometric definition of the model.

CLASS #2:

Attribute data in class #2 is characterized as data which is attached directly to geometric entity data, and which can be described completely in terms of that geometric entity.

CLASS #3:

Attribute data in class #3 is characterized by data which is attached directly to geometric entity data but which needs auxiliary geometric data (henceforth known as attribute specification geometry) to help define the attribute. That is, the attribute may not be conveniently described in terms of the geometric entity to which it is attached, and thus two pieces of geometric entity data are required; a piece of geometry data and a piece of attribute specification geometry data.

The basic distinction between class #2 and class #3 data is that class #2 data needs a single geometric entity to define both (a) the associativity of the attribute with the model and (b) the attribute's definition. Class #3 data, on the other hand, requires a piece of geometric data to define its associativity with the model, and in addition, requires attribute specification geometry to define the attribute.

As an aid to understanding class #3 data, consider the case of an arbitrarily complex flat plate in the xy-plane. Suppose that the structure was subjected to a normal pressure load which was linearly varying as a function of y. If one attempted to describe this pressure solely in terms of pressure values along the edges, one would not have a uniquely defined pressure surface, and thus the pressures on individual elements could be evaluated incorrectly. Alternatively, the pressure could very simply be specified by defining an auxiliary piece of geometry; in this case a rectangular face which covered the entire 2-D domain and four pressure values, one at each of the corners. Pressure on individual elements could then be evaluated in a totally unambiguous manner.

3.2.2. Attribute Specification Geometry

Attribute specification geometry is simply geometry plus topology which is used to help define the physics of certain attributes. It can be thought of as being auxiliary to the part definition. It has no direct links to the part's geometric data structure. The attribute specification geometry will be an intrinsic part of the definition of the attributes. It is stored in the MODEL data structure but is referenced through the attributes. The attribute

specification geometry maintains the same hierarchic structure as the models geometry/topology.

The purpose of attribute specification geometry is twofold. First, it provides a mechanism for allowing simpler and more efficient specification of attribute data. Secondly, it provides a means for evaluating attribute data directly and unambiguously.

3.2.3. Attribute Data Structure

The relationships between attribute data and geometry are as follows. The model's topological entities point to attributes. An attribute contains its definition along with two pointers. One pointer points back to the geometric entity to which it is "tied." The other pointer points to the attribute specification geometry which is used to help define the attribute. If this geometry is, in fact, the same as the model's geometry, then the attribute specification geometry pointer is a null pointer. Figure 2 shows the general data structure for a generic attribute data type. It indicates how attributes are specified and associated with geometry/topology. The dimensionality of the attribute topology can be the same or lower order as the model's topology to which it points; it can never be of higher order. That is, an edge is a one-dimensional entity. The permissible associated attribute's topological entities are "vertex" and "edge."

3.3. Mesh Data Structures

In addition to the hierarchy of geometric modeling entities discussed earlier, there will also be a hierarchy of finite element entities, the MESH data structure (Figure 3), which will be used to define the elements themselves. This is a departure from the way in which finite elements have historically been defined (i.e., an element of a specific type with a list of nodes which define the connectivity).

The finite element entities have two types of data associated with them. The first is the modeling topology data, and the second is the finite element attribute data (i.e., material properties, physical properties, etc.). Each finite element entity points to the lowest order modeling topology entity which it is inherently a part. For example, a fe-edge which is on the surface of a region would point to the face on which it lies, rather than the region itself. It is this relationship that permits attribute data which is tied to the geometry to be evaluated on an element by element or node by node basis.

Pointers from the geometry to individual element components (i.e., fe-nodes, fe-edges, fe-faces, etc.) are also desirable for efficient postprocessing activities. The storage penalties for maintaining these relationships are easily offset by the performance gains which can be achieved.

3.3.1. Finite Element Entity Definitions

What follows is a set of working definitions for the various finite element entities.

- FE-NODE -- A fe-node is a three-dimensional point in space. Typically, it is used to define a fe-edge; however, it can also be used to help define a finite element. For example, the mid-face node used in a Lagrange parabolic element is not associated with a fe-edge, but rather with the fe-face itself. In addition, a reference node used to define the center of curvature or the plane in which a beam element lies would point to the element rather than the edge of that element. Finally, a fe-node could be used to define the element explicitly such as the concentrated mass element. A fe-node may lie on a vertex, an edge, a face, or be completely contained within a region.

- FE-EDGE -- A fe-edge is a combination of topology plus implied geometry. That is, the nodes used in the definition of an edge are used to bound the geometric curve, and at the same time can be used to define the geometry (i.e., straight line, parabola, or cubic) for the element itself. A fe-edge can be used to define a fe-face, as with planar or solid elements, or can be used to define the finite element directly, as is the case with truss and beam elements. It may lie on an edge, a face, or be completely contained within a region.

- FE-FACE -- A fe-face is bounded by fe-edges. It is either used to define the surface of a "planar" finite element or is used in combination with other fe-faces to bound a solid finite element. Topologically, a fe-face will be either triangular or quadrilateral in nature. It may lie on a face, or be completely contained within a region.

- FINITE ELEMENT -- Depending on the type, a finite element can be a fe-node, a fe-edge, or a collection of fe-faces. It may be completely contained within a region. However, it is true only for solid finite elements that the entity "finite element" can point to a region because other finite element types will have lower order entities which point to various geometric entities.

3.3.2. Advantages of a Finite Element Entity Hierarchy

At first glance, the MESH data structure, with its hierarchy of finite element entities, may seem too elaborate, perhaps even wasteful of valuable storage. However, on closer inspection some distinct advantages emerge. The most powerful advantages come from the links to the other data structures. These relationships are discussed in the next section. Independent of the benefits which accrue due to these links, a number of other benefits surface as enumerated below:

- It provides an organization for handling any type of finite element in a uniform manner.

- It provides direct access paths to higher order entities from lower order entities which make it very convenient to do such things as bandwidth minimization, postprocess the results of elements associated with a given set of nodes, etc.

- It makes it possible to interrogate the finite element model using a geometric entity as a key word for searching.

- It provides a mechanism which supports mesh generation on the basis of topologically simple cells (i.e., quadrilaterals, triangles, hexahedrons, etc.) which corresponds to linear finite elements, providing a direct path to upgrade to higher order elements without going back to the mesh generator. All higher order fe-nodes can easily be placed precisely on the appropriate associated geometric entity.

3.4. Relationships of the Three Data Structures

The power of the implementation is derived from two sources; the data structures themselves and the relationships between the structures. Figure 4 shows the relationships which exist amongst the three data structures. It is, in fact, these links that provide the necessary structure for claiming to be a geometry-based system. These links provide a bond between the data structures which permit the system to respond in a cohesive manner.

The links are automatically established during the model generation process. The natural progression of events is something like this:

1. The part is generated via a geometric modeling session. The geometric entities are loaded into the MODEL data structure.

2. Model attributes are defined and loaded into the attribute data structure. Since the attributes are associated with the model's geometry, two-way pointers are established between the MODEL data and the ATTRIBUTE data. In addition, any necessary attribute specification geometry is generated and stored in the MODEL data structure. Links are also established between the ATTRIBUTE data structure and the attribute specification geometry.

3. One of the attributes is mesh control data. Having this information, mesh generation can proceed, and the resulting node and element data is stored in the MESH data structure. During the mesh generation process, the associations which exist between the finite element mesh data and the part definition are known, and thus pointers between the MESH data structure and the MODEL data structure can be generated.

4. Since both the MESH data and the ATTRIBUTE data point to the MODEL data, the attribute data can then be evaluated on a node by node or element by element basis. The links between the MESH data and the ATTRIBUTE data structures are established at this point.

This completes the model building process. It accomplishes what it was intended for; to use a completely geometry-based approach to produce an analysis model.

4. DESIGN OF A GEOMETRY-BASED PREPROCESSOR

The approaches and data structures outlined above form the basis on which an advanced geometry-based preprocessing system can be built. The remaining capabilities needed are the actual finite element model definition procedures and the user interface.

The best form of user interface for this system is an interactive graphics front end. This is obviously the most convenient form of interface for the specification of geometry and geometry-based information. Even for those cases where the geometry to be meshed is identical to that obtained from the geometric modeler, and an automatic mesh generator is used, there is still the need for the specification of the analysis attribute information in terms of the geometric model. Until fully automatic, adaptive procedures are available, the system must support the entire range of finite element mesh generation procedures. These are most efficiently operated in an interactive graphics mode.

Geometric operations within the system will be carried out making heavy use of the capabilities of the geometric modeling systems to which it is interfaced. It is important that geometric modeling functions be presented in a form appropriate for finite element modeling. This may be different than the way they are presented in the geometric modeling system. In addition, it must be recognized that different geometric modeling systems will not provide the same sets of geometric modeling functions. The two level approach to the implementation of the geometric communication operators provides a method to deal with both of these concerns. The high level geometric communication operators for finite element modeling would be designed to fit directly into the modules of the preprocessing system. Since they are constructed by the combination of the low level geometric communication operators, which represent the actual tie to the geometric modeler, they need not necessarily be altered when a new modeler is interfaced to the system. If a particular geometric modeler does not provide specific low level operators used by a high level operator, it may be possible to reconstruct the high level finite element operators by a different combination of low level operators.

The geometric modeling functions needed by a complete finite element preprocessing system are extensive. They include a full set of high level operators, such as the Boolean operators, for the construction and modification of geometry, as well as for use by automatic mesh generators to decompose the geometry into a valid finite element mesh. A full range of geometric interrogation operators will be required for use by the mesh generation algorithms, the mesh checking procedures, the geometric construction operations, and the attribute specification procedures. Finally, a full range of bottom-up geometric modeling functions are needed to allow the analyst to define all or part of a geometry.

The attribute specification procedures in a geometry-based preprocessor must give the analyst a high level of flexibility in the specification of the various types of finite element attributes. The geometric specification procedures for defining analysis attributes, such as distributed loads, must allow for the convenient description of the distribution of the loads, as well as for defining the portions of the object on which they act.

Flexible procedures must be available to group attributes of the same type into sets for simple manipulation during the specification of the actual load cases to be analyzed. The reason for allowing the grouping of attributes is partly to provide convenience to the analyst, but is mainly for the purpose of allowing a greater degree of automatic validity checking in the system. By only allowing the combination of attributes of one type into sets, automatic

validity checks on attribute combinations, which are based on attribute type, can easily be done. The combination of attribute sets into analysis cases allows the application of an additional set of checks which can only be made after the analysis process control information has been indicated. Thus, the user maintains a high degree of flexibility while affording the system a means of performing validity checks at the appropriate levels.

A difficulty in the implementation of the mesh control attributes in a system that allows a variety of mesh generation approaches is devising a procedure that can operate from a single internal representation of mesh control information. Since all attributes, including mesh control information, are directly tied to entities in the geometric model, the most direct method of dealing with this specification is to tie mesh control parameters to each of the topological entities that define the object. The analyst can be given a set of procedures that allow for geometry-based specification of the mesh control information and have it properly associated with the topological entities. Since it is possible to introduce geometric entities for the sole purpose of attribute specification, this approach maintains the desired level of flexibility. The remaining question is the selection of mesh control parameters for the various topological entities that can always be meaningfully converted to the specific parameters needed by the various mesh generators. The simplest solution is to assign a single element size parameter to all entities. Although seeming simplistic, this tends to be acceptable for all entities except the edge. The reason for this is simply that most mesh generators base all their mesh control on edge information, and those that use additional parameters, typically use a single parameter per entity. The mesh control information appropriate for edges should allow for the specification of the number of elements along the edge, as well as biasing parameters to grade the size of elements in a flexible manner.

As indicated above, the preprocessor should house a variety of mesh generation procedures ranging from simple bottom-up meshing procedures through fully automatic meshing procedures. It is anticipated that as automatic mesh generators become more robust, and as the geometric modeling capabilities needed to support them continue to improve, they will tend to become the main mesh generation tool. However, until fully automated finite element modeling systems become available, there will be a continued need to support the other mesh generation approaches.

Bottom-up mesh generation will tend to be used for the quick construction of both mesh and geometry (in terms of the finite elements) for very simple objects, and for adding simple finite element entities to an object that does not contain all the geometric entities in a form convenient for generation of that portion of the finite element model. Although, these procedures will not represent the major mesh generation workhorse, their presence in the system is necessary.

Mapped mesh generators are the most popular mesh generation procedures currently available. To some extent, they are more difficult to provide the needed geometric communication operators than some of the automatic mesh generation approaches [1]. The system must contain procedures that allow the analyst to easily define the supplementary geometry needed to define the boundaries of mesh patches and to be able to select the geometric entities that

define the specific mesh patches. The process of defining these mesh patches in a geometry-based preprocessor that accepts a general geometric model as input is substantially different from preprocessors where the geometry is built in a bottom-up fashion in terms of mesh patches. The user tools needed to efficiently decompose a general geometry into a set of valid mesh patches are different from those that are efficient for defining a geometry in terms of a set of mesh patches. The preprocessing system discussed here should support both sets of capabilities.

The selection of fully automatic mesh generation procedures to be included in such a system must consider the following factors:

1. The ability of the mesh generator to produce the desired types of meshes.

2. The ease of integration of the meshing procedure with geometry through a set of geometric communication operators.

3. The computational efficiency of the mesh generator.

Since the level of complexity of geometric operators needed to integrate an automatic mesh generator with a geometric representation varies greatly [1], as does their computational efficiency, it is likely that these two factors will dictate the selection of automatic mesh generators to be included in the preprocessor.

5. OPEN QUESTIONS IN THE DESIGN OF A GEOMETRY-BASED PREPROCESSOR

The procedures presented in the previous sections address the close coupling of the geometric representation of an object and the finite element mesh used to analyze it when there is an obvious correspondence of the finite elements generated and the geometric entities in the model. The type of finite element models that yield this correspondence are those where the finite elements are dimensionally the same as the geometric entities, and when the domains spanned by the geometric and finite element model are the same. However, it becomes much less clear how to account for the coupling between the geometric and finite element models when the geometric model is simplified for purposes of finite element analysis or when the finite element mesh contains a mix of element types of different geometric order representing various portions of a solid model. Common examples of these cases include ignoring specific geometric features deemed unimportant, and the use of shell or beam elements when one or two of the geometric dimensions of specific portions of a geometric model are small compared to the others. Element types of a dimensional order less than the geometric entity they represent account for the small dimensions in terms of element parameters such as thickness and moment of inertia. Elements of this type will subsequently be referred to as indirect elements.

Historically, the concern for the proper representation of the differences between the geometric model and the finite element model have not existed. This is because the two modeling processes were carried out independently. However, the desires to make direct use of the information in the original geometric model, to maintain complete links for making model revisions easier,

and to maintain a clear history of the analysis modeling procedures used, makes it necessary to address the question of how to define and account for these differences.

A major portion of the answer to these questions lies in the data structures to be used and the procedures employed to reflect the differences between the models in the database. However, this is not the appropriate place to begin to address these questions. One of the major factors that makes this a complex question is the lack of analytic procedures, or even an agreed upon set of rules for determining when and how these modeling differences should be used. If this information were available, it would be possible to devise algorithmic approaches to carry out these processes and it would become more obvious as to the best method to account for the results of those processes. Lacking such information requires that the approach taken to address these questions be somewhat open ended, thus allowing users to carry out the operations associated with geometric simplification and the generation of indirect element types in a flexible manner.

As an example of the range of possible approaches to geometric simplification, three different approaches to account for domain differences are considered. In all cases, the finite element analyst begins with the complete specification of the geometric model. In approach one, the analyst generates the mesh interactively with a mapped mesh generator. In this case, the finite element model generation process consists of the analyst simplifying the geometric model by performing specific geometric modeling operations during the construction of the various mesh patches. With currently available finite element modeling procedures, this is an appropriate method. However, this approach does not readily lend itself to account for the specific geometric simplifications made to the model before mesh generation. Ever if the analyst was required to make the geometric simplifications, independent of the definition of the mesh patches, accounting for the simplification would require the explicit storage of both models or storing the list of modeling operations carried out during the simplification, neither of which is convenient.

The second and third approaches require the availability of a fully automatic mesh generation procedure that can ignore geometric features during the meshing process. With such a capability, the mesh generator can be passed the entire geometric model. Geometric features to be ignored are flagged appropriately. Accounting for the differences between the geometric and finite element models consist of simply examining the geometric model to see which geometric features are flagged. The second approach would consist of the analyst flagging the geometric details to be ignored while the third approach would rely on adaptive analysis procedures to determine the features to be ignored. Although the finite element modeling capabilities needed to support these two approaches are not fully available, components of them are currently being investigated. For example, the quadtree [6] and octree [7] mesh generators operate on the basis of hierarchic insertion of the geometric entities within an object's boundary file into a tree structure. Therefore, it is possible to simply identify those entities associated with the geometric features to be ignored so they are represented in an approximate manner. Although this approach may not be able to account for all desired forms of geometric simplification, it should be able to easily handle a majority of them. Efforts are currently underway to develop and test these capabilities.

The development of adaptive analysis procedures to automatically identify geometric details to be ignored, is a much more complex issue. One possible approach is to combine a set of rules employing analytic stress concentration factors with the results from an initial analysis that ignored features in order to estimate their influence [8] and to determine if they should be included.

The controlled generation of, and accounting for, the use of indirect element types is an even more complex process. The computerization of this process could make effective use of artificial intelligence techniques to help convert geometric representations to numerical analysis representations [9].

6. CLOSING REMARKS

There are a number of areas that must be addressed before fully automated finite element modeling becomes a reliable analysis tool that is an integral part the computer-aided engineering process. This paper has addressed one of those areas which is the framework of a preprocessing system that allows the complete integration of finite element modeling with geometric modeling. The two key aspects of the approach are the use of geometric communication operators and the use of advanced data structures required to store the various data sets needed in finite element modeling. The key to the data structures is the use of a single hierarchic boundary-based geometric representation for both the geometric model and the finite element model. To this, auxiliary data structures (e.g., the attribute data structure) can be linked. A boundary-based representation was selected because:

1. It is the most general form of geometric representation to which other geometric forms can be converted.

2. It is a convenient framework on which new geometric and finite element types can be quickly added.

3. It is the most natural form, since finite element modeling is dominated by boundary information.

The major penalty for the added capability of this approach is the large amount of data storage. This is unavoidable if the goal of a general, geometry-based system is to be achieved. The only way to reduce the amount of information required is to reduce the level of integration with general geometric modeling systems or to limit the number of finite element modeling procedures that can be supported.

A final advantage of the approach presented here is that it can fully support today's finite element modeling procedures while allowing the introduction of ever increasing levels of automation as fully automatic mesh generators and adaptive analysis procedures evolve. This is very important since current preprocessing systems cannot support full automation and it is only through automation of these procedures that finite element techniques can be made a reliable tool for designers and not just the experts.

REFERENCES

1. Shephard, M. S., "Finite Element Modeling Within an Integrated Geometric Modeling Environment: Part I - Mesh Generation, Part II - Attribute Specification, Domain Differences, and Indirect Element Types," Engineering With Computer, Vol. 1, 1985, pp. 61-85.

2. "CAM-I Geometric Modeling Project Boundary File Design (XBF-2)." CAM-I Report R-81-GM-02. 1, October 1982.

3. Wilson, P. R., I. D. Faux, M. C. Ostrowski, and K. G. Pasquill, "Interfaces for Data Transfer Between Solid Modeling Systems," IEEE Computer Graphics and Applications, Vol. 5, No. 1, 1985, pp. 41-51.

4. Weiler, K., "Topological Structures for Geometry Modeling," PhD Thesis, Rensselaer Polytechnic Institute, Troy, New York, 1986.

5. Requicha, A. A. G. and H. B. Voelcker, "Solid Modeling: A Historical Summary and Contemporary Assessment," IEEE Computer Graphics and Applications, Vol. 3, 1982, pp. 9-24.

6. Baehmann, P. L., S. L. Wittchen, M. S. Shephard, K. R. Grice, and M. A. Yerry, "Robust Geometrically Based Automatic Two-Dimensional Mesh Generation," TR-86007, Center for Interactive Computer Graphics, Rensselaer Polytechnic Institute, Troy, New York, 1986.

7. Yerry, M. A. and M. S. Shephard, "Automatic Mesh Generation for Three-Dimensional Solids," Comput. Struct., Vol. 20, 1985, pp. 31-39.

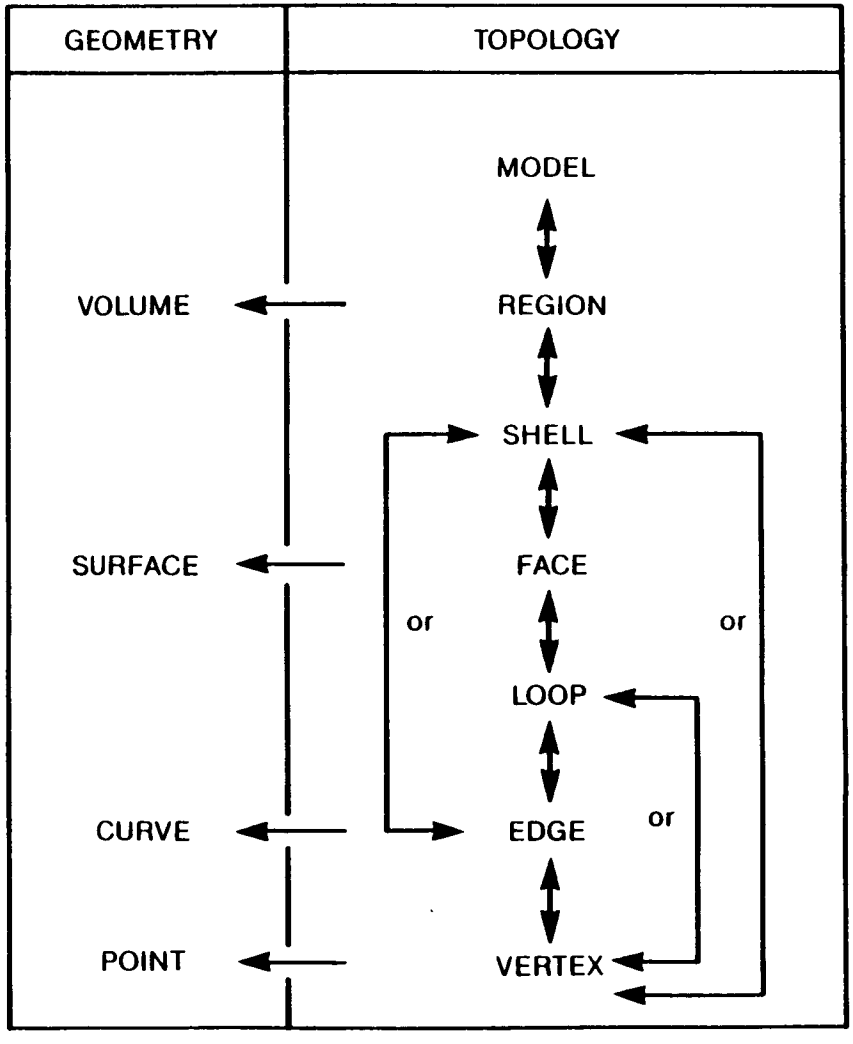
8. Shephard, M. S. and M. A. Yerry, "Toward Automatic Finite Element Modeling for the Unification of Engineering Design and Analysis," Finite Elements in Analysis and Design, Vol. 2, 1986, pp. 143-160.

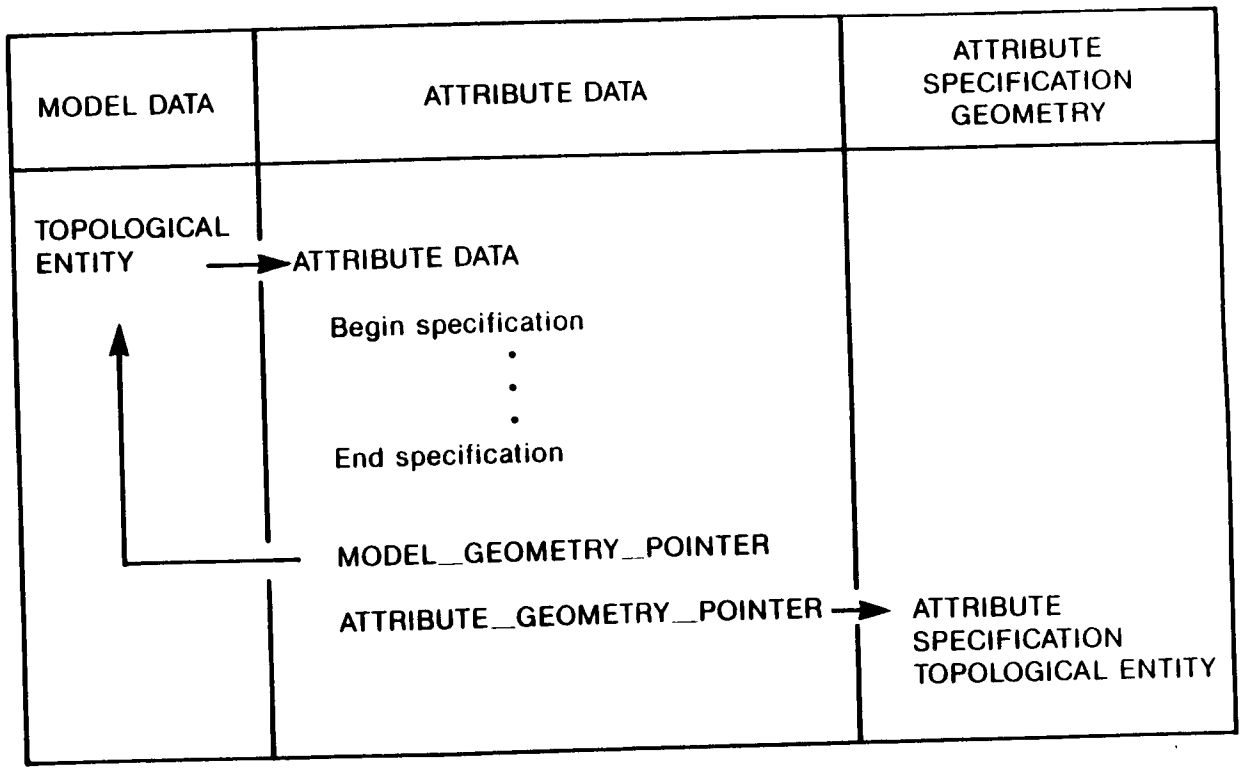
9. Gregory, B. L. and M. S. Shephard, "Design of a Knowledge Based System to Convert Airframe Geometric Models to Structural Models," Expert Systems in Civil Engineering, ASCE, New York, New York, 1986, pp. 133-144.

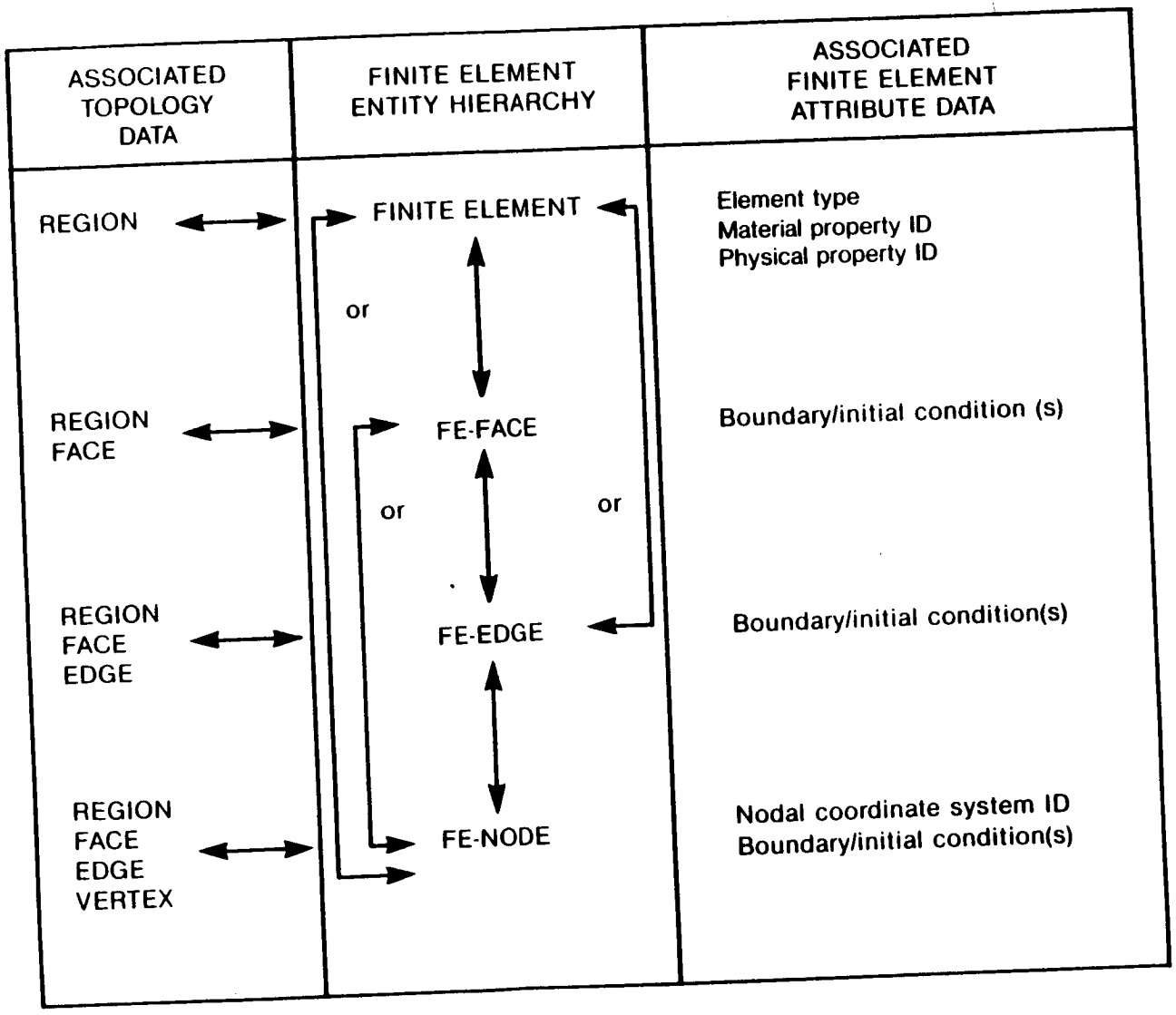
10. Casale, M. S. and E. L. Stanton, "An Overview of Analytic Solid Modeling," IEEE Computer Graphics and Applications, Vol. 5, No. 2, February 1985, pp. 45-56.

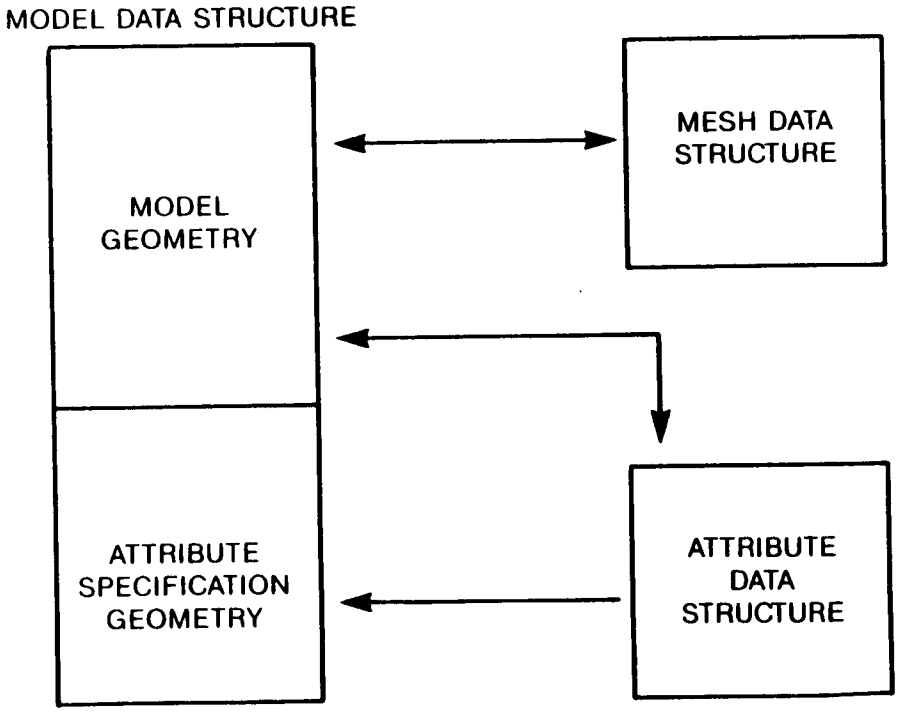
11. Farouki, R. T. and J. K. Hinds, "A Hierarchy of Geometric Forms," IEEE Computer Graphics and Applications, Vol. 5, No. 5, May 1985, pp. 51-78.

LIST OF FIGURES


GEOMETRY
TOPOLOGY
MODEL
VOLUME
REGION
SHELL
SURFACE
FACE
or
or
LOOP
CURVE
EDGE
or
POINT
VERTEX


MODEL DATA
ATTRIBUTE DATA
ATTRIBUTE SPECIFICATION GEOMETRY
TOPOLOGICAL ENTITY
ATTRIBUTE DATA
Begin specification
End specification
MODEL_GEOMETRY_POINTER
ATTRIBUTE_GEOMETRY_POINTER
ATTRIBUTE SPECIFICATION TOPOLOGICAL ENTITY


ASSOCIATED TOPOLOGY DATA
FINITE ELEMENT ENTITY HIERARCHY
ASSOCIATED FINITE ELEMENT ATTRIBUTE DATA
REGION
FINITE ELEMENT
Element type
Material property ID
Physical property ID
or
REGION FACE
FE-FACE
Boundary/initial condition (s)
or
or
REGION FACE EDGE
FE-EDGE
Boundary/initial condition(s)
REGION FACE EDGE VERTEX
FE-NODE
Nodal coordinate system ID
Boundary/initial condition(s)


MODEL DATA STRUCTURE
MODEL GEOMETRY
MESH DATA STRUCTURE
ATTRIBUTE SPECIFICATION GEOMETRY
ATTRIBUTE DATA STRUCTURE