An Object Oriented Approach to Geometry Defeaturing for Finite Element Meshing

Anton V. Mobley, Michael P. Carroll, and Scott A. Canann
ANSYS, Inc.
275 Technology Drive
Canonsburg, PA 15317
{tony.mobley, michael.carroll, scott.canann}@ansys.com

Abstract. Automatic finite element mesh generation of CAD generated data has been a goal of finite element meshing codes for years. However, the lack of accuracy and the amount of detail in this data have made this a daunting task. In essence, the CAD data needs to be defeatured to overcome accuracy deficiencies and to remove excessive detail. In this paper, an object-oriented approach to automatic geometry defeaturing is presented. The geometric and finite element data abstractions are given, along with the basic algorithms used. These algorithms deal with near tangencies, coincident edge precision discrepancies, poor intersection curve accuracy, and small geometrical features. Along with this discussion, examples of these types of defeaturing are given.

Keywords. Defeaturing, dirty geometry, object-oriented FEM, CAD import, meshing

1. Introduction

1.1 Importance of work

Though computer aided design (CAD) and computer aided engineering (CAE) can trace their origins to the advent of digital computing, their evolution since have taken them on quite different paths. CAD was originally created to automate the laborious job of 2-D drafting and it remained in the 2-D realm for quite some time. In the CAE world, the finite element analysis (FEA) technique became one of the most popular methods, in which the model was idealized by breaking it down into simple triangular and quadrilateral shapes (know as elements). However, it was realized early that 2-D physical phenomena idealizations were of limited use at best. Applying the same techniques to the tetrahedral and hexahedral shaped elements, full 3-D elements were constructed. The construction of these idealization models, especially in 3-D, becomes tedious and error prone. Specialized software was written to facilitate the construction of these types of models, but the only geometric definition was the nodal locations and element connectivity.

With the development of better solid, parametric, and feature-based modeling techniques, the creation of complex 3-D solid models dominated the next stage of CAD development. The FEA packages had also grown in their sophistication, but they had been created with a bias for creating models that could easily create finite element models. Moreover, modeling was not as high a priority for FEA packages, as was the development of more complex idealization techniques. The solid modeling in these packages lagged behind the more focused CAD packages.

While the creation of solid geometry from both FEA and CAD sound very similar, in fact the created models are quite different. The CAD packages are more interested in creating visually acceptable models that can display a model in its “computer” environment with all the details of the finished product. In addition, CAD models are loaded with details that bring no value to the FEA idealization. Most of the time, these details inhibit the creation of an ideal FEA model by introducing areas that require high element size transitions. Usually, this occurs away

from the FEA operator's area of interest in the model. The FEA package does a great deal of work for little or no gain in the idealization's accuracy. In addition, FEA programs want "clean" geometry and topology in which FEA meshes can easily be generated. Since no real standard form of geometric representation is used by any of the major CAD packages, the transfer of the geometry and topology from one package to another is difficult, if not impossible. The job of reading CAD models into FEA packages is difficult and error prone. Some CAD packages made forays into the analysis realm by creating meshing capabilities inside their modelers. This bypassed the translation problems but generated new ones. Often, CAD packages create FEA idealizations that are too large to be solved by the current technology, that create invalid FEA elements, or that make assumptions that make FEA solvers fail.

However, engineering management sees the creation of two models (one for CAD and one for FEA) as a duplicitous effort and is constantly looking for solutions to this fundamental problem. In addition, since FEA is the end user of the CAD geometry, the inability to transfer the models and operate on them was perceived to be an FEA problem. This paper examines some of the techniques that can be used to handle CAD data in the FEA process.

1.2 Background and previous work

In the August 1997 issue of Engineering Automation Report [5], the readers' survey identified the ability to exchange geometric data between the different vendors' software packages as the most important current issue in the CAD/CAM/CAE community.

Marc Halpern from DH Brown Associates [7] presents the findings of an extensive study that shows that the median time to complete full analyses, including convergence studies, has been reduced 48% from 1991 to 1996. This speedup is credited to advances in mesh generation techniques, the impact of adaptivity, and improvements in CAD-FEA integration. Although this progress has been impressive, Halpern states that the state-of-the-art has still not yet gotten to the point where it can satisfy the requirements of the early design practice. As proof, his study shows that the median time to perform a finite element analysis to verify a design is 4 days and the average time is 7 days. Designers make changes to their designs at a faster pace than that. Therefore, significant advances are still required.

Although dealing with CAD solid models in FEA products is a fairly recent phenomenon, an impressive amount of work has already been done. In this section, a brief synopsis is given for many of the articles and products that include such capabilities, emphasizing those that address one or more of the following:

- Geometric validity checking and repair – especially for geometry import
- Dimensional reduction
- Model idealization or simplification, i.e., defeaturing

Many FEA packages, including ANSYS [8], Firmin and Walsh [6], provide a limited set of manual, bottom-up repair and rebuilding geometry tools. This type of geometry repair involves deleting the volumes, surfaces, and curves near the problem, creating new geometry, and then creating new volumes from the newly modified curves and surfaces.

Dey [3] developed a method that searches for small edges that create small angles, and then collapses all of the small edges that won't invalidate the mesh or reduce the dimensionality of the model. He also presents techniques for collapsing triangular faces or tetrahedral elements that have extremely large dihedral angles. He combines edge and face swaps with these two operators, to increase the method's success rate.

Armstrong [1], Price [10], Jones [9], and Donaghy [4], et al, use the medial object[1] transform of the geometry as a tool to:

- Aid in the decomposition of general solids into sub-regions for mapped meshing,
- Identify slender regions for dimensional reduction, and
- Recognize small features for suppression.

Butlin [2] uses geometric type feature suppression to eliminate small features.

The bulk of this work is at the geometry/topology level. As such it is similar to the feature suppression found in Section 3. However, the authors have found that geometry level feature suppression is not sufficient to guarantee meshing success. This paper extends this work to mesh level suppression that is presented in Section 5. Although this mesh level suppression has some similarities to the dimension reduction methods found above, it does provide a unique tool for the meshing solution.

1.3 Paper overview

This paper presents an object-oriented approach to performing automatic validation, repair, and small feature suppression at the CAD and FEA model level.

In section 2, a brief overview of the class structures is presented for the CAD topological and geometric entities used for the boundary representation. Section 3 presents geometry-based defeaturing, which performs small feature recognition and suppression for specified topological and geometric entities. Section 4 describes the object-oriented C++ classes used to represent the FEA model. Section 5 presents FE model-based defeaturing, where feature suppression is done using the representation of the FEA model described in section 4. Section 6 shows some results, including example meshes using the presented defeaturing scheme. Conclusions are then given in Section 7.

2. Base Classes for CAD Topology and Geometry Representation

2.1 Overview

Although many packages share the same underlying solid modeling kernel, almost every CAD package has it own internal topological and geometric representation. This adds to the complexity of maintaining connections from CAD to FEA. Since, as stated previously, FEA is an end user of CAD models, so a *superset* of the possible topological data structures has to be chosen for the FEA topological representation. If the FEA topological structure chosen does not support all of the CAD's topological representations, then data modification of the CAD model has to also be performed in addition to the data transfer. These modifications can further decrease the reliability of the CAD to FEA model connection.

A topology data structure is an ideal candidate for the data abstractions that object oriented languages such as C++ provide. A well-chosen abstraction is essential since all geometric and topological information must be able to be represented. The model itself can be abstracted as a boundary representation, **Brep**. In addition, each entity (vertex, edge, face, or body) has been abstracted into a base class that was given the name **Cell**. The **Cell** class contains four different types of data (each of which will be discussed in detail in the subsequent sections):

1. Attributes
2. Topological Traversal

[1] The medial object is a medial axis in 2D and a medial surface in 3D. In 2D, it can be thought of as the collection of the centers of the largest circles that can be placed entirely inside the model, without crossing the boundary curves. In 3D, instead of circles, spheres are used.

3. Geometric Query
4. Geometric Data

This scheme matches the paradigm that is used by most CAD systems. In particular, the authors have worked with ACIS, Parasolid (UG), ProEngineer, and Computervision.

2.2 Attributes

Attributes contain information such as the dimensionality and type of the entity and a topological identification number (*topoid*). The *topoid* uniquely identifies the entity in the model. The **BRep** contains a global function that when given a *topoid* can return a pointer to the **Cell** for that entity. Conversely, each **Cell** has a member function that will return its *topoid*. Data base support such as saving and resuming the entity are also categorized as an attribute of the **Cell**.

2.3 Topological Traversal

The ability to move up or down the topological tree in the model is accomplished by another class called **CellUse**. **Cell** has a "has a" relationship to the **CellUse** class. A "has a" relationship describes the situation where one class contains the other, thus the **Cell** class "has a" **CellUse** class as one of its data members. Upward traversal is accomplished by calling the **Cell** member function **use** and using the member function **bound** (see Figure 1) allows for downward traversal in the structure.

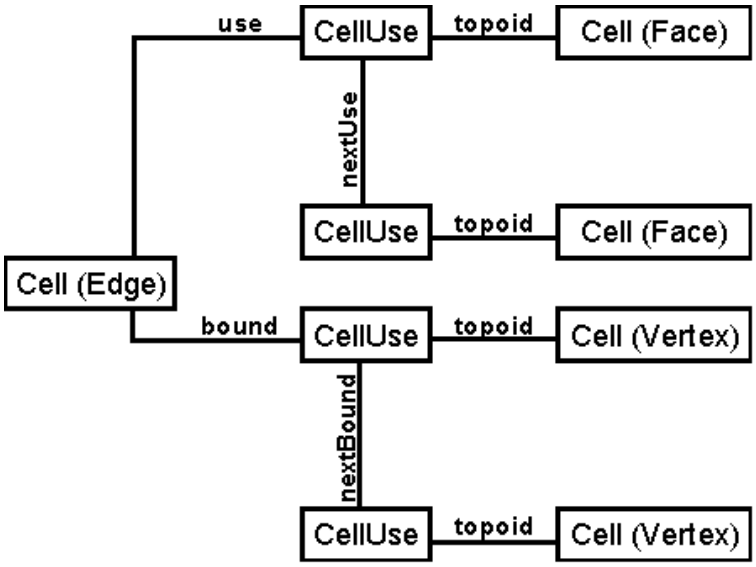

Figure 1.
Upward and downward traversal chart

2.4 Geometric Query

Two types of geometric functions are needed for meshing. The first is computational in nature. This includes the ability to query a geometric entity or evaluate a point on it, such as projecting a point onto the entity, evaluating the normal and tangent information, or finding the geometric and parametric bounds for the entity. **GeomQuery** was the name given to the abstract base class that was developed to abstract these functions. The **Cell** has a "has a" relationship to the **GeomQuery** class. These Geometric Query objects can use internal functions for the above data or link back to the CAD package for these evaluations.

2.5 Geometric Data

The meshing algorithms are written to the **GeomQuery** interface described above. In early implementations, no local curve or surface data was kept. Instead, the CAD packages' functions were used directly to evaluate data. However, it was quickly learned that the CAD package's calculation overhead was prohibitive. The interfaces exposed by the CAD packages are not optimized for cases where many evaluations are repeatedly performed for each geometric entity. The performance for meshing was found to be unacceptable for even the most benign curve and surface types. Therefore, local surface and curve classes have been added and objects from these classes are then cached in memory and optimized specifically for meshing tasks. The virtual base class that was exposed from the boundary representation was called the **GeomData** class. The **Cell** has a "has a" relationship with the **GeomData** class.

3. CAD-Based Feature Suppression

3.1 Overview

Since CAD models are feature rich, an ability to determine which features to include in the FEA idealization without having to go back to the CAD model is beneficial. This type of model defeaturing is found in Butlin [2] and Jones [9]. With this ability to recognize and suppress features, the chances of successfully meshing a CAD model increases. In addition, the resulting meshes usually contain far fewer elements. Another similar approach is presented here.

3.2 Feature Recognition

Listed below are the checks that are made to eliminate features. In order to quantify "smallness", a tolerance, T, is specified:

- C1 -- Cylindrical faces bounded by two circular external edges. The height of the cylinder is smaller than T. (This would represent a boss feature.)
- C2 -- Conic faces that are bounded by two external circular edges. The height of the cone is smaller than T. (This would represent a chamfer feature.)
- C3 -- Toroidal faces that are bounded by three or four circular arcs. The minor radius of the torus is smaller than T. (This would represent fillet features.)
- C4 -- Spherical faces bounded by two, three, or four arcs. The radius of the sphere is smaller than T. (This would represent a fillet feature.)
- C5 -- Faces that are bounded by two circular edges. The distance between the edges is smaller than T. (This represents small holes and conic hole features.)

The elimination of these features from the model is done by simply merging one of the edges into another and removing the face from the topological traversal list of entities for the part. There is also a need to check if the operation can take place at all. For instance, a thin disc might be detected as a small height cylinder. Its removal would collapse the entire model on itself and prevent the analysis model from being performed.

The choice of which edge to keep and which edge to remove can be made by setting up a precedence order for each of the defeaturing operands. Either topological checks and/or geometric checks can be used to determine which edge is to be retained and which is to be discarded.

For each case given above, an algorithm can be created to remove the feature from the model.
For cases C1and C2:

Given: Global defeaturing tolerance, T

Angle tolerance, A
A cylindrical/conic face, F, with a height, H, bounded by two circular edges, E1 and E2.

If H <= T,

Let F1 be the neighboring face attached to E1 and F2 be the neighboring face attached to E2.

Geometric Check:

First, verify that the collapsing of the cylinder would be a valid geometric option. This prevents the collapsing of a face that could degenerate the model as described previously.

1. Evaluate a set of points, P1, along E1 and evaluate normals of points on F1 to create normal set N1.
2. Project points P1 onto E2 to create new set of points P2.
3. Evaluate normals of P2 on F2 to create N2.
4. For each normal pair in N1 and N2, find angle between normals.
5. If angles are within the angle tolerance A, go to Topological Checks. Otherwise, perform no action.

Topological Check:

The topological checks are to determine which edge is to be retained in the model and which edge is to be removed. The checks are implemented such that material is added to the solid model.

1. If E1 is on an interior loop of F1, substitute E1 for E2 in F2 topological structure (i.e. E1 becomes the outer loop for F2).
2. If E2 is on an interior loop of F2, substitute E2 for E1 in F1 topological structure (i.e. E2 becomes the outer loop for F1).
3. Eliminate F from topological structure.

3.3 Small Edge Detection and Removal

Cases C3 to C5 remove edges that are below the tolerance, T, as edges are removed the small faces are also compressed out as a by-product.

Another topological operation that can be used to simplify a CAD model is the removal of edges whose lengths are smaller than the tolerance T. As edges are removed from the model, the attached faces are updated. During updating, face topologies can disappear from the CAD model entirely. Since this is not a closed form procedure like the previously discussed operations, a more comprehensive method for determining the precedence of each edge and face is required. Small edges may also be prevented from being compressed out of the model if they do not pass certain criteria. As previously mentioned, compressing out edges in certain cases can invalidate the model. Take, for example, a thin rectangular plate. If the thickness of the plate is less than the defeaturing tolerance, the compression of the model would make the 3D model degenerate. Using a similar geometry check, this condition can be trapped by using normal comparisons as were discussed in the previous section.

For the cases where the compression of a small length edge can compress an entire face from the model, the following procedure can be used to determine if the face should be compressed -- and if so, which edge should be retained:

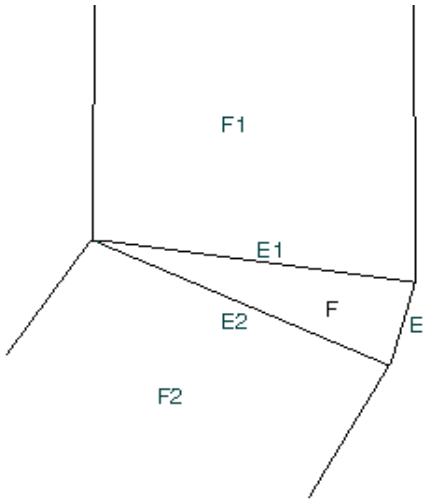

Figure 2.
Suppression of small length edge E and sliver face F.

Given: Global defeaturing tolerance T
A face, F, bounded by three edges E, E1, and E2

If E <= T,

Assume that it is determined that edge E qualifies to be compressed out of the model. Let F1 be the neighboring face attached to E1 and F2 be the neighboring face attached to E2 (see Figure 2).

Geometric Checks:

Verify that the collapse can happen. If so, then determine the edge precedence order. That is, determine which edge (E1 or E2) should be retained and which should be removed.

Validity check:

1. Evaluate a set of points P1 along E1 and evaluate normals of points on F1 to create normal set N1.
2. Project points P1 onto E2 to create new set of points P2.
3. Check the distances between points pairs in P1 and P2. If any distance is greater than tolerance T, do not compress E.

Precedence check:

1. Evaluate normals of P2 on F2 to create normal set N2. At this point, we want to see if one of the faces F1 or F2 is above the other in order to attempt to add material to the model in the defeaturing process.
2. Create a set of local Cartesian coordinate systems C1 at each point in P1 using the normals in N1 as its local z direction.
3. Create a set of local Cartesian coordinate systems C2 at each point in P2 using the normals in N2 as its local z direction.
4. Transform the points in P2 into C1 to create P2 and points in P1 into C2 to create P1.
5. If the local z coordinate for each member in P2 is negative and the local z coordinate for each member of P1 is positive, *retain edge E1*.
6. If the local z coordinate for each member in P1 is negative and the local z coordinate for each member of P2 is positive, *retain edge E2*.
7. Else,
8. If E1's length > E2's length *retain E1*.
9. If E2's length > E1's length *retain E2*.

10. Else, *retain E1*.

At this point, F is removed from the model and if E1 is retained, the edge list for F2 is updated. If E2 is retained the edge list for F1 is updated.

4. Finite Element Data Abstraction

4.1 Overview

As with the CAD boundary representation, finite element data naturally lends itself to object oriented data abstraction. The classical way of representing a finite element is to break the representation into two entities. The first entity is the node that has the global 3D location. The node represents the geometry of the finite element representation. The next entity, which represents the topology of a finite element, is the element. An element contains a list of nodes that make up its shape. A quadratic triangular element will have a list of six nodes: one node for each corner and one node for the midpoint of each side. The design used in this paper is a mix of the classical finite element topological entity in conjunction with the CAD boundary data. Instead of viewing the finite element as a list of nodes, the element is a collection of topological entities that the classical finite elements can be derived from. The topological entities relate the node back to its defining geometry. This was felt to be a more generic and natural representation for the problem. The implementation in this paper uses tri/tet elements; however, the concepts could be extended to quad/hex elements.

4.2 Geometric Representation

The class that was created to represent all geometric information in the finite element structure is Mesh Boundary Point (**MBPoint**)**.** The **MBPoint** contains more than just the 3D location in space of the node. It is also a container for all of the geometric information that is known about that point. Another class called **Geompt** was created to store information that is specific to one CAD entity. The **MBPoint** has a "has a" relationship with a link list of **Geompt** objects. The **Geompt** contains the *topoid* of the CAD entity, the node's parametric location for that entity, and other attributes. Some of these attributes are the flags that tell if the node actually lies on the entity within a tolerance, whether the entity's parametric representation is current, and whether this entity is the prime entity representation for this **MBPoint.** For instance, if a node lies on a vertex of a block, it would have one **Geompt** of the vertex (which would be marked as it's prime entity), six **Geompt** objects from the three edges and three faces that are attached to the vertex.

The member functions of the **MBPoint** class associate the operands of the CAD's **Brep GeomQuery** class to the finite element node. The operations of evaluating, getting normal and tangent information, and projecting on a specific entity are all member functions of the **MBPoint** class. Other functions such a moving a point from one location to another are encapsulated in the **MBPoint** class. This function takes care of managing the **Geompt** list and, if needed, updating the parametric locations.

4.3 Topological Representation

With the geometric information removed from the finite elements, the problem of representing this in a class hierarchy becomes much easier. If we ask the question, "What are the topological entities that represent the finite element class?", a very natural representation falls out. First, the nodes that represent the vertices of the finite element are abstracted into a topological class, which we called **MBE0** (**M**esh **B**oundary **E**lements **0**-dimension). The edges of the finite element abstract into a class **MBE1**, the faces into **MBE2**, and the volume elements into **MBE3**. The **MBE0** has a "has a" relationship with the **MBPoint** class. Most geometric operations are accessed through the interface of the **MBE0** into the **MBPoint** class. Since our finite element specification called for quadratic elements, the **MBE1** class also has a "has a" relationship with the **MBPoint** class. The coordinate retrieved from this **MBPoint** represents the mid side node in the quadratic finite element.

The **MBE1** class also contains pointers to the two **MBE0**s that bound it. The **MBE2** class contains pointers to the **MBE0**s and **MBE1**s that bound it and the **MBE3** class contains pointers to the **MBE0** and **MBE2** objects that bound it.

There are also upward and downward traversal methods for each class. These traversals are not limited to one level up or down, but have been implemented, for instance, to allow the traversal of all **MBE3**s attached to a specific **MBE0** object.

5. FE Model-Based Defeaturing

5.1 Overview

As previously discussed, defeaturing of the CAD model can significantly reduce the number of elements, increase the robustness of meshing, and decrease the time necessary to mesh the model. However, the techniques discussed in Section 3 do not cover all CAD topological and geometric problems. Recognition and suppression of features that could cause meshing problems becomes difficult at the **BRep** level beyond those discussed in Section 3. However, it became apparent that to achieve a high level of meshing success, more forms of defeaturing were needed.

After doing analysis on CAD models that failed to mesh, several causes were identified for meshing failures. Most of the meshing failures occurred while meshing the faces of the CAD models. The faces that failed had one of several attributes that seem to cause the meshing problems. These included extremely small length edges as compared to the entire face, tangencies between two edges at a shared vertex, and edges that were coincident (within a tolerance). Though some of these can be identified at the **BRep** level, there are no easy topological changes that can be made to modify the model.

5.2 MBE Level Defeaturing

Instead of working at the **BRep** level, MBE level defeaturing procedures work with the data generated during the meshing process. This process first meshes the edges of the model. This is followed by meshing the faces and then for volume meshes, the meshing of the interior. After the first stage is complete and **MBE0** and **MBE1** objects have been created on the edges and vertices, the following algorithm was implemented to scan these objects and collapse near coincident or tangent edges into one another. The basic algorithm looks for nodes that are close to edges that do not contain the node. This will happen when the model has a slight rise from one face to another, as shown in Figure 11. It can also happen when a model has a face with edges that are nearly coincident or tangent, as shown in Figure 13.

Given: Global defeaturing tolerance, T
Length ratio tolerance, R
Angle tolerance, A
Average length of edges, L
Face normal tolerance, FN
Merging tolerance, MT

1. Let Node1 belong to the set of **MBE0** objects. Let *MBN1* be a set of **MBE1s**, such that E belongs to *MBN1* if the distance(Node1,E) < L. Let *SMBN1* be a subset of *MBN1*, where E belongs to *SMBN1* if E contains Node1 - Figure 3 presents such a case where Node1 has four edges that are close to it. Two of the edges are in *SMNB1* and the other two are not.
2. Find an edge, Edge1, in *MBN1* such that Edge1 does not contain Node1 and Edge1 shares a common face with Node1. If there are none, get another Node1 from the set of **MBE0** objects and start with Step 1.

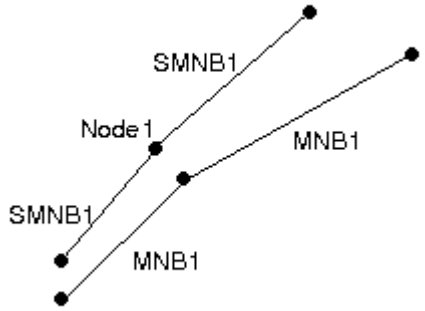

Figure 3.
Node1 is in the proximity of edges in the set MNB1.

3. First check for a node that is close to other edges. This indicates a rise or coincident edge condition. Let Node2 be the projection of Node1 onto Edge1, and let D=distance(Node1,Node2) – see Figure 4. –

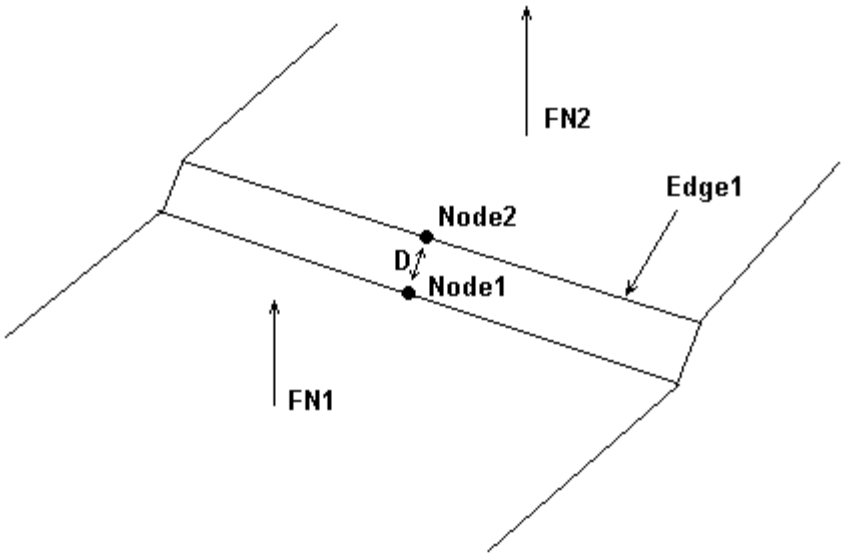

Figure 4.
Node2 is the projection of Node1 on Edge1. FN1 and FN2 are the face normal sets.

4. Let SL = maximum (length of edges belonging to *SMBN1*).
   Let AL = maximum (SL, Length of Edge1).
   If D < T, or the ratio of D /AL > R, then continue with step 7.
5. Check for a tangency condition. If EN has a **MBE0** that is shared by Edge1, let A1=angle(Edge1,EN). See Figure 5.
6. If the angle A1 < A, then continue with step 7. Else, go to step 2 - see Figure 5.
7. Now Node1 is a candidate to be merged out of the model along with it's connected edges. Create another candidate at Node2. First check that Node2 is not close to one of the endpoints of Edge1. Let EP1 and EP2 be the endpoints for Edge1. If distance(EP1,Node2) < MT, then set Node2=EP1. Else if distance (EP2,Node2) < MT, then set Node2=EP2. Else split Edge1 at Node2 and create a new **MBE0** at this point.
8. Edge1 was chosen such that it and Node1 have a face in common. Look at the faces that are not in common. If these normals are in the same direction, then the node can be merged out. If one has a tangent edge, a nearly

coincident edge, or a slight rise from one face to another, this is exactly the case. However if one has a small cut feature in the model, this is not the case.

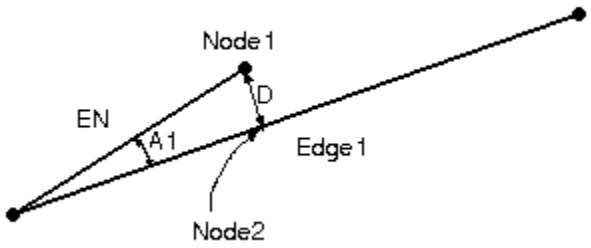

Figure 5.
Edge EN contains node Node1 and shares a node with Edge1.
A1 is the angle between edges EN and Edge1

9. Let FN1 be the face normals from Node1 and FN2 be the face normals from Node2. If angle(FN1,FN2) > FN, another candidate node is retrieved and the procedure starts again at step 1. (This results in a split of Edge1, which will result in better element quality, but no merging is possible.) See Figure 6.

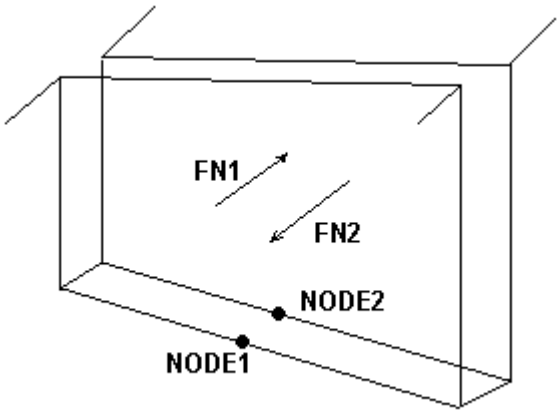

Figure 6.
This situation shows an example where Node1 and Node2 cannot be merged.

10. Node1 and Node2 can now be merged. It must now be determined which of the **MBE0** objects is to be retained in the model and which one is to be deleted. This done be assigning an order of precedence to each **MBE0**, the **MBE0** with the highest order of precedence is retained and the other **MBE0** removed. The order of precedence is determined by first checking the **BRep**'s topological entities that attach to each **MBE0**. If one **MBE0** is attached to an entity of a lower dimensionality, then it will be retained. If both **MBE0**s **BRep**'s lowest dimension entity are the same go to step 11.
11. The next operation in determining the order of precedence is to create a set of planes at Node1, PL1, from the Node1's location and the face normals, FN1. Then Node2's location is then compared against these planes, PL1, to determine if Node2 is above or below each of the planes. The procedure is then repeated for Node2. If either **MBE0** lies below each of the planes, this it is chosen to be merged out of the model. This is consistent with the goal to add material when defeaturing the model. If neither **MBE0** fulfills these criteria, then go to step 12. See Figure 7.

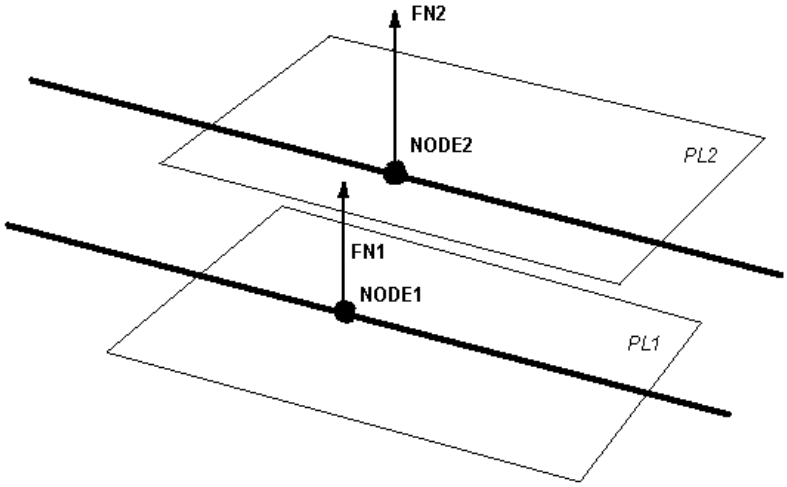

Figure 7
Face normal sets FN1 and FN2 when coupled with nodal positions of Node1 and Node2 will define a set of planes PL1 and PL2.

12. The next test in the order or precedence is to see which **MBE0** would move the least distance if merged. Node1's location is projected onto all of Node2's faces and the maximum distance that Node1's location is from any of Node2's faces is found. Then the procedure is done for Node2. The node whose maximum distance is smallest is retained and the other one is removed.
13. Merge the node with the lowest order of precedence into the other node. Using the situation portrayed in Figure 4, it can be shown that Node2 would have a higher order of precedence than Node1 (using the procedure in step 11 and shown in Figure 7). After merging Node1 into Node2, material would be "added" to the model. See Figure 8.

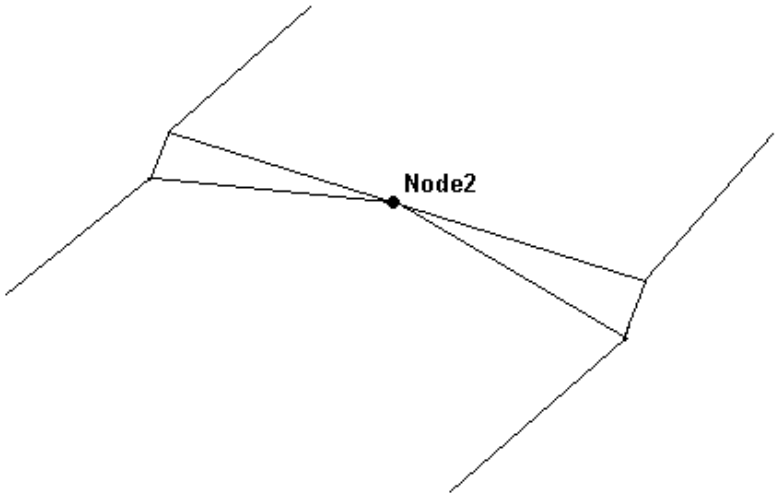

Figure 8
After merging Node1 into Node2 the resulting boundary would appear like this figure. Further defeaturing could take place because of the tangencies and/or the small edges surrounding Node2.

6. Results and Examples

6.1 Example of topological defeaturing of the CAD model.

A parametric model was constructed of a disk, shown below in Figure 9. The four holes in the disk, the fillet, and the recess where defined with parameters. The parameters were changed such that these features became small compared to the size of the model and highlighted. This model was then analyzed by fixing the inner cylinder and applying a moment to the outer cylinder. Two analyses were performed, the first without defeaturing and second with the highlighted features suppressed. The meshing time and the number of elements for each model are presented in Table 1. The meshes for each analysis are shown in Figure 10. It can be seen that the suppression of the small features dramatically reduces the number of elements and time to mesh. This model is contrived to illustrate this point and is not meant to claim that the use of engineering judgement should not be used on whether feature suppression is important to the analysis.

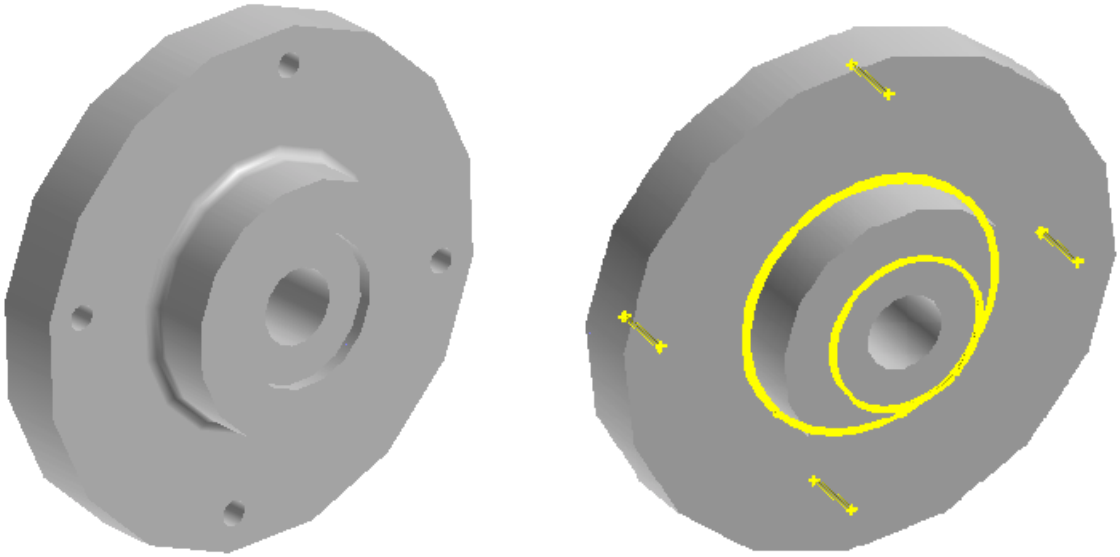

Figure 9.
Parametric disk model. On the left is the original model. The figure on the right shows the model after parametric changes have produced small features to be removed from the topological model.

	Number Of Elements	CPU Time (Sec.)
Model Without Defeaturing	14,743	60
Model With Defeaturing	1,038	11

Table 1.

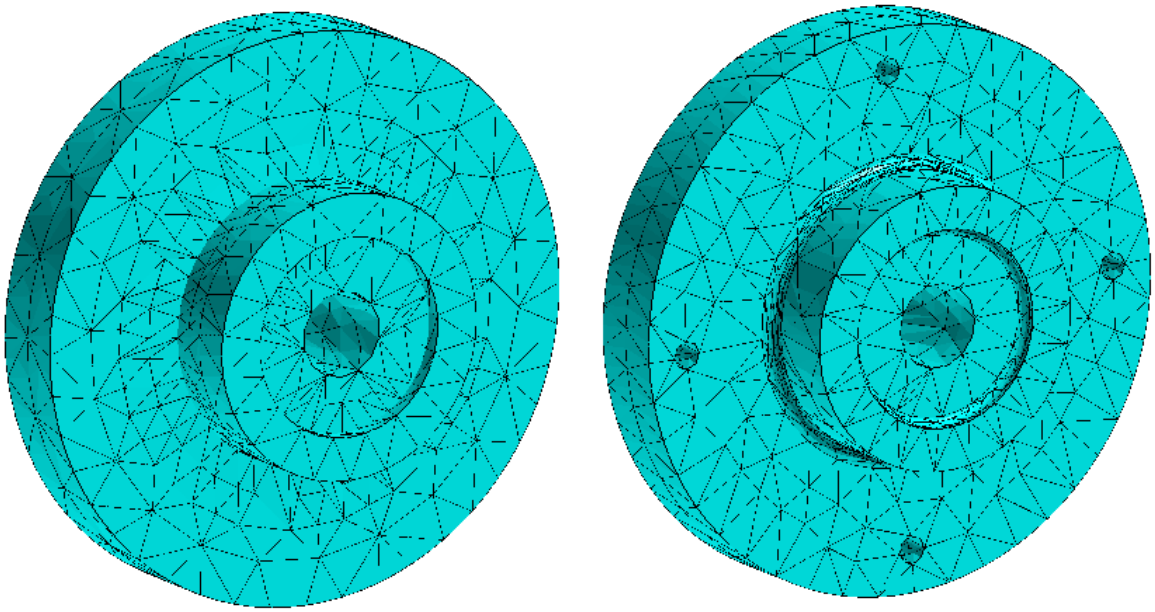

Figure 10.
Finite element model with (model on the left) and without (model on the right) defeaturing.

6.2 Finite element defeaturing

Two examples are given showing faces that have degenerate exterior loops. In the first model, the attachment of the upper block is off by a small distance and creates a degenerate edge along the top. (See Figure 11, notice what appears to be a single line across the top of the block, but is, in reality, a coincident edge). Figure 12 shows the finite element mesh along that top section after MBE1 defeaturing has been applied.

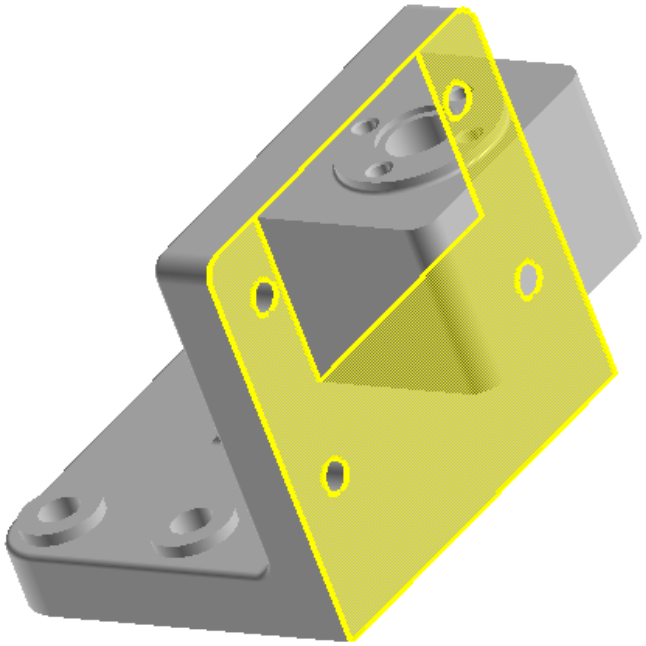

Figure11.
Highlighting of a face with a degenerate edge.

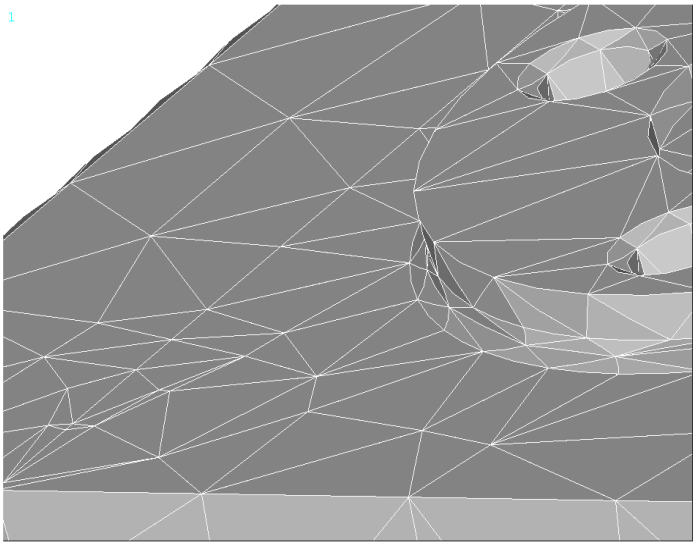

Figure 12.
Zoom of finite element mesh showing the area of the coincident edges.

Another model of a hack saw shows a highlighted face that has a degenerate face loop (see Figure13) and a close-up of the elements where the MBE1 defeaturing has taken place (see Figure 14).

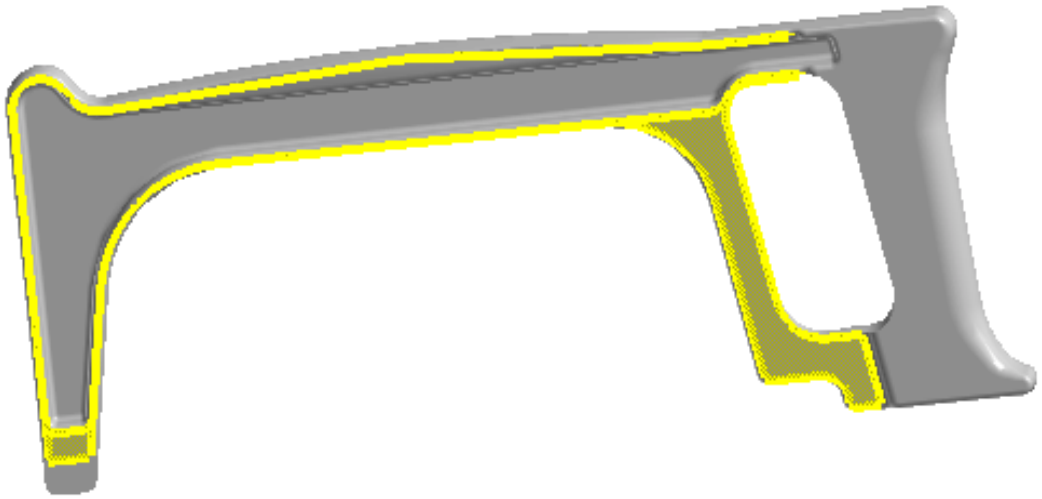

Figure 13.
Hack saw model showing degenerate face loops.

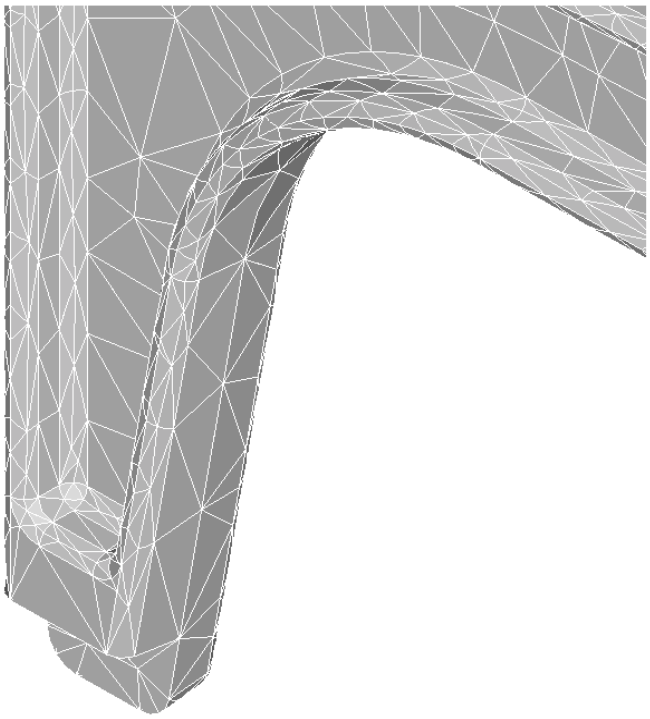

Figure 14.
Zoom of finite element mesh.

7. Conclusion

This paper presents a framework for automatically meshing real world 3D CAD models with tetrahedra. The emphasis is on dealing with the problems inherent in meshing CAD models. Processes are described which can recognize undesirable features in CAD models that can make meshing fail, give poorly shaped elements, or give too many elements. In addition, two general types of defeaturing are presented – geometry-based defeaturing and FE model-based defeaturing. These forms of defeaturing are shown to reduce model size and increase mesher robustness.

This paper also illustrates the advantages of using a well thought out data abstraction design to represent not only the CAD geometry and topology but the finite element model as well. A good foundation in object-oriented design allows the programmer of the defeaturing algorithm to focus on the algorithm and not on the data management of the underlying structures.

While the defeaturing presented in this paper is an automatic procedure, it is a tool to be used by an informed engineer. Good engineering judgment still needs to be exercised. As with any analysis, the effects of the feature suppression should be independently verified.

Acknowledgements

The authors would also like to thank ANSYS, Inc. for its support of the research, development and publication of the work and Chris Brunetti for his help in creating the examples used in this paper.

References

[1] Armstrong, C. G., D. J. Robinson, R. M. McKeag, T. S. Li, S. J. Bridgett, R. J. Donaghy and C. A. McGleenan, "Medials for Meshing and More", *Proceedings of 4th International Meshing Roundtable*, Sandia National Laboratories, p.277-288, October 1995.

[2] Butlin, G. and C. Stops, "CAD Data Repair", *Proceedings of 5th International Meshing Roundtable*, Sandia National Laboratories, pp.7-12, October 1996.

[3] Dey, S., M. S. Shephard, and M. K. Georges, "Elimination of the Adverse Effects of Small Model Features by the Local Modification of Automatically Generated Meshes," *Engineering with Computers*, Vol. 13, p. 134-152, 1997.

[4] Donaghy, R. J., W. McCune, S. J. Bridgett, C. G. Armstrong, D. J. Robinson and R.M. McKeag, "Dimensional Reduction of Analysis Models", *Proceedings of 5th International Meshing Roundtable*, Sandia National Laboratories, pp.307-320, October 1996.

[5] *Engineering Automation Report*, p. 10, August 1997.

[6] Firmin, T., and J. Walsh, "The Challenge of Transferring Solid Model Data to ANSYS," *Proceedings of the Eighth International ANSYS Conference*, Pittsburgh, Pennsylvania, 1998.

[7] Halpern, M., "Industrial Requirements and Practices in Finite Element Meshing: A Survey of Trends," *Proceedings of 6th International Meshing Roundtable*, Sandia National Laboratories, pp.399-411, October 1997.

[8] Importing Solid Models, ANSYS Modeling and Meshing Guide, Release 5.4, 000862, 2nd Edition, SAS IP, Inc., Chapter 6, September 1997.

[9] Jones, M.R., M. A. Price and G. Butlin, "Geometry Management Support for Auto-Meshing", *Proceedings of 4th International Meshing Roundtable*, Sandia National Laboratories, pp.153-164, October 1995.

[10] Price, M., C. Stops, and G. Butlin, "A Medial Object Toolkit for Meshing and Other Applications", *Proceedings of 4th International Meshing Roundtable*, Sandia National Laboratories, p.219-229, October 1995.