

## 5B.3

---

# An Immersive Topology Environment for Meshing

Steven J. Owen<sup>1</sup>, Brett W. Clark<sup>1</sup>, Darryl J. Melander<sup>1</sup>, Michael Brewer<sup>1</sup>,  
Jason F. Shepherd<sup>1</sup>, Karl Merkley<sup>2</sup>, Corey Ernst<sup>2</sup>, and Randy Morris<sup>2</sup>

<sup>1</sup> Sandia National Laboratories <sup>†</sup>, Albuquerque, New Mexico  
`{sjowen,bwclark,djmelan,mbrewer,jfsheph}@sandia.gov`

<sup>2</sup> Elemental Technologies, American Fork, Utah  
`{karl,corey,randy}@elemtech.com`

**Summary.** The Immersive Topology Environment for Meshing (ITEM) is a wizard-like environment, built on top of the CUBIT Geometry and Meshing Toolkit. ITEM is focused on three main objectives: 1) guiding the user through the simulation model preparation workflow; 2) providing the user with intelligent options based upon the current state of the model; and 3) where appropriate, automating as much of the process as possible. To accomplish this, a *diagnostic-solution* approach is taken. Based upon diagnostics of the current state of the model, specific solutions for a variety of common tasks are provided to the user. Some of these tasks include geometry simplification, small feature suppression, resolution of misaligned assembly parts, decomposition for hex meshing, and source and target selection for sweeping. The user may scroll through a list of intelligent solutions for a specific diagnostic and entity, view a graphical preview of each solution and quickly perform the solution to resolve the problem. In many cases, automatic solutions for these tasks can be generated and executed if the user chooses. This paper will discuss the various diagnostics and geometric reasoning algorithms and approaches taken by ITEM to determine solutions for preparing an analysis model.

**Key words:** geometry simplification, sweep decomposition, design through analysis, hexahedra, sweeping, assembly meshing, imprint/merge, meshing user interface

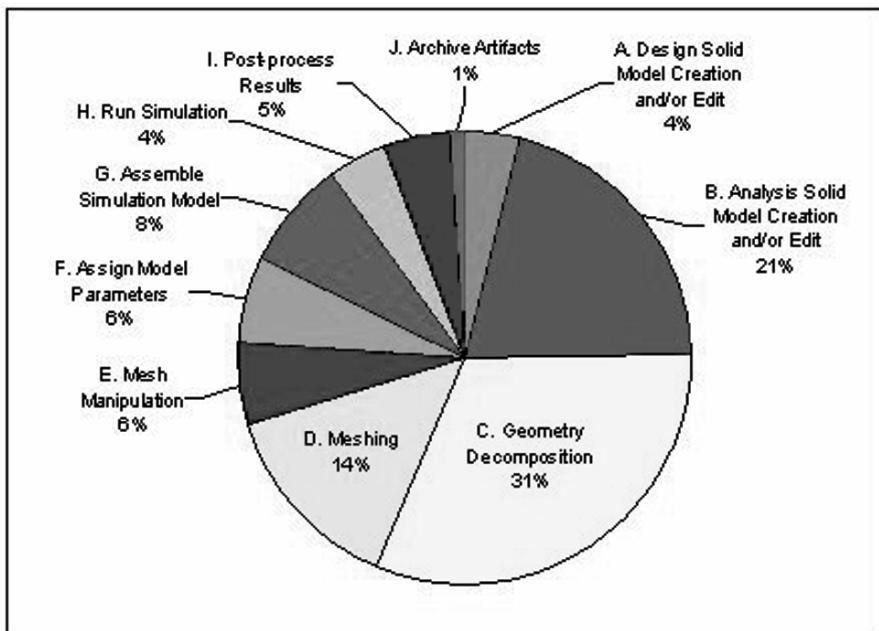
## 1 Introduction

At a cursory inspection of the computational simulation process, the creation of a mesh may seem like a relatively trivial task. In most cases, significant energy and thought is put into the numerics for computing the physics of the

---

<sup>†</sup>Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000

system. A mesh may often be thought of as simply a means to represent the geometric domain of the system and its significance frequently diminished. Once the problems to be simulated advance beyond simple academic prototypes of blocks and cylinders, the true magnitude of the meshing problem readily becomes apparent. It is not unusual for the meshing process to take upwards of three-quarters of the entire simulation time. At Sandia National Labs, for instance, a survey[1] of analysts was conducted in 2005 to determine where the bulk of their time was being spent in modeling and simulation.



**Fig. 1.** Approximate percent of time taken by analysts to accomplish tasks in the modeling and simulation process at Sandia National Laboratories

Analysts were asked to quantify the amount of time they spend in each of 10 separate tasks. Analysts were selected from a wide variety of disciplines including modal, linear and non-linear structural, heat transfer, fluid flow and radiation transport. Figure 1 shows a summary of the results of this survey. Of significant note is the relatively large amount of time devoted to building the analysis solid model, geometry decomposition, meshing, and mesh manipulation (Tasks B through E). These tasks were reported to take 73% of the total time as compared to just 4% to actually run the simulation. These statistics illustrate where the major bottlenecks remain in the simulation process.

With the current state of the art, these tasks are inherently very user interactive. Analysts at Sandia have access to almost any state-of-the-art model-

ing and simulation tools developed within the lab and available commercially. They often choose to use methods that perform better with a hexahedral mesh definition. While tetrahedral methods are adequate for many situations, specific advantages are frequently cited for using hexahedral meshes. State of the art meshing tools, such as CUBIT<sup>3</sup> are developed to support the meshing and geometry needs of the analysts at Sandia.

CUBIT uses a toolbox approach to providing a meshing solution. Incorporating state-of-the-art algorithms for quadrilateral, triangle, tetrahedral and hexahedral meshing it tries to address a diverse range of mesh generation needs from across many disciplines throughout the Laboratories. As a solid-model based system it allows the user the flexibility of importing existing CAD models from commercial tools such as Pro/Engineer<sup>4</sup> and Solidworks<sup>5</sup>, but also includes many tools for generating a solid model directly within CUBIT. Models that have been developed in commercial solid modeling tools are rarely created with simulation in mind and must frequently be simplified. Geometric translation errors introduced as a result of incompatible modeling standards between commercial tools further complicate the model preparation and can be time consuming to address. Geometry decomposition, another time-consuming task, is also often needed to provide suitable topology for hexahedral meshing algorithms.

CUBIT, first developed at Sandia in the early 1990s as a research platform for new geometry and meshing research, has become the most used tool by Sandia's engineers for generating meshes for simulation on a day-to-day basis. It is also available world-wide through a government or academic use license as well as for commercial distribution. With CUBIT's many sophisticated and technically advanced tools developed for a wide range of application areas, it can be an unwieldy endeavor to become proficient enough with the software to quickly generate a mesh from a complex geometry. As a result, the *Immersive Topology Environment for Meshing* (ITEM) was developed.

With the ultimate goal of reducing the time to generate a mesh for simulation, ITEM has been developed within the CUBIT Geometry and Meshing Toolkit to take advantage of its extensive tool suite. Built on top of these tools it attempts to improve the user experience by accomplishing three main objectives:

1. Guiding the user through the workflow
2. Providing the user with smart options
3. Automating geometry and meshing tasks

---

<sup>3</sup><http://cubit.sandia.gov>

<sup>4</sup><http://www.proengineer.com>

<sup>5</sup><http://www.solidworks.com/>

## 1.1 Guiding the user through the workflow

In software of any complexity where usage may be occasional or infrequent, the overhead of learning the new tool to a point of proficiency may be daunting. Given a solid model that may have been designed for manufacturing purposes, an analyst may be faced with generating a mesh. They may not be working with CUBIT on a daily basis, but would like to take advantage of the powerful tools provided by the software.

To address this, ITEM provides a wizard-like environment that steps the user through the geometry and meshing process. For someone unfamiliar with the software, it provides an interactive, step-by-step set of tools for accomplishing the major tasks in the process. For those more familiar with the tools, it serves as a reminder of the major tasks, but is flexible enough to accommodate a more iterative approach, allowing them to jump between major tasks easily. Currently restricting the workflow to models requiring three-dimensional, solid elements, ITEM uses the following steps:

1. Define the Geometric Model: Import a CAD model or create geometry within the CUBIT environment.
2. Set up the model: Define basic information such as element shape, volumes to be meshed and element sizes or budgets.
3. Prepare the geometry: Detect and remove unwanted geometric features on the CAD model, resolve problems with conformal assemblies and identify and provide suggestions to make the geometry sweepable.
4. Meshing: Perform the meshing operation and provide feedback if it is unsuccessful.
5. Validate the mesh: Check element quality and perform mesh improvement operations.
6. Apply boundary conditions regions: Define regions where boundary conditions may be applied using nodeset, sideset and block definitions.
7. Export the mesh: Define a target analysis code format and export the mesh.

## 1.2 Providing the user with smart options

Solid models used for analysis may have a huge variety of different characteristics that could prevent them from being easily meshed. Questions such as, “What are the problems associated with my model?”, “What are the current roadblocks to generating a mesh on this model?” and “What should I do to resolve the problems?”, are constantly being asked by the analysts as they work with models. Without an extensive knowledge of the tools and algorithms, it may be difficult to answer these questions effectively.

ITEM addresses this issue by providing smart options to the user. Based on the current state of the model, it will automatically run diagnostics and determine potential solutions that the user may consider. For example, where

unwanted small features may exist in the model, ITEM will direct the user to these features and provide a range of geometric solutions to the problem. Scrolling through the solutions provides a preview of the expected result. The user can then select the solution that seems most appropriate and execute the solution to change or simplify the geometry. This *diagnostic-solution* approach is the basis for the ITEM design and is the common mode of user interaction while in this environment. This contrasts with the more traditional *hunt-and-guess* approach of providing the user with an array of buttons and icons from which they may choose and guess what may result. On the other hand, ITEM serves in effect as an expert providing guidance to the user as they proceed through the geometry and meshing process.

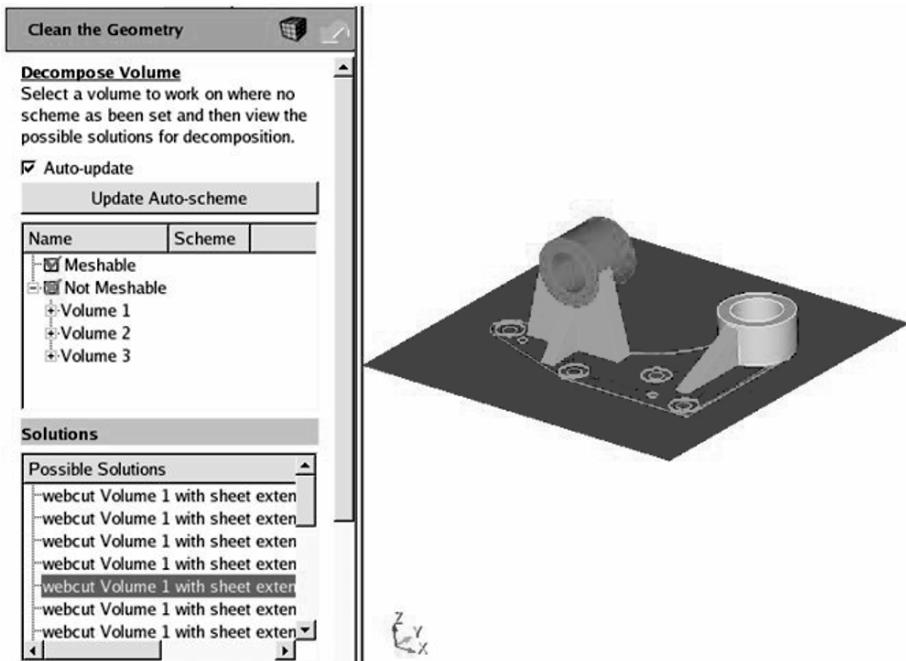
To illustrate the *diagnostic-solution* approach, Figure 2 shows an example of one of the panels in the ITEM environment. In this panel a diagnostic is run to determine what volumes are not yet meshable based on the criteria for sweep meshing[19]. After selecting one of the volumes, a set of potential operations for decomposing the model is computed and presented to the user as a solution list. Selecting or browsing the solution list will preview the decomposition, shown to the right of the panel. Once satisfied with one of the solutions, the user may quickly perform the displayed operation, which in turn will update the diagnostics and display a new set of volumes for consideration.

### 1.3 Automating geometry and meshing tasks

With all of the advanced research and development that has gone into the meshing and geometry problem, a push-button solution for any arbitrary solid model may seem like the ideal objective of any meshing tool. Although for many cases this would be the best solution, for others it may not even be desirable. A push-button solution assumes a certain amount of trust in the geometric reasoning the software chooses to provide. This may be more trust than an occasional user who is tasked with a high consequence simulation may be willing to give. Even if the user is willing to accept full automation, in many cases, the geometric complexity of the model may be beyond the capability of current algorithms to adequately resolve.

Alternatively, once users are familiar with the characteristics of the solutions that the software provides, they may not be concerned with examining and intervening on every detail of the model creation process. Instead, in the interest of increasing efficiency, they may want the fastest solution possible. As a result, the idea of providing the option for full user automation while still maintaining the alternative for user control is a central objective of ITEM.

For various characteristic geometric problems that are encountered in a solid model, ITEM can determine from the potential geometric solutions, which may be most applicable and apply that solution without any user intervention. For many configurations of geometry, a completely automated solution may be available. For others, ITEM may be able to automate only



**Fig. 2.** Example of a GUI panel in ITEM illustrating the diagnostic-solution approach

a portion of the process. Where an adequate solution cannot be determined automatically, the smart options described above are available to help guide the user. As new advances in geometric reasoning and advanced meshing algorithms are developed, ITEM will incorporate these into the solutions for automation.

Although ITEM utilizes a variety of common meshing algorithms for meshing surfaces and solids, a full description of these methods is beyond the scope of this document. Instead, ITEM has primarily been designed to be *meshing algorithm independent*. The diagnostics and solutions proposed in ITEM are developed with the objective of being able to successfully utilize one or more of CUBIT's meshing schemes.

The remainder of this document highlights some of the key aspects of the analysis model preparation process and describes how they are addressed within the context of ITEM. These aspects focus primarily on preparing the geometric model for meshing, with some focus on final mesh quality. They include removing small features, resolving problems in a conformal assembly, building a sweepable topology, and improving mesh quality. For each of these aspects of model preparation, a description of the typical problems encountered will be defined along with proposed diagnostics that can detect these

situations. Once a problem has been detected, the basic logic for determining a list of solutions to address the specific problems are outlined.

## 2 Geometry Clean Up

Meshing packages have the challenge of dealing with a host of geometry problems. Many of these problems can be generalized as file translation issues. Typically, the geometry used in a meshing package has not been created there but in one of many CAD packages. Exporting these files out of CAD and into a neutral file format (IGES, STEP, SAT) accepted by the meshing software can introduce misrepresentations in the geometry. If the CAD and meshing packages do not support the same file formats, a second translation may be necessary, possibly introducing even more problems.

Another complication caused by file translation is that of tolerances. Some CAD packages see two points as coincident if they are within 1e-3 units, while others use 1e-6. If the meshing software's tolerance is finer than the CAD package's, this disparity in tolerance can cause subsequent geometry modification operations in the meshing package to inadvertently create *sliver* features, which tend to be difficult and tedious to deal with. This tolerance problem also causes misalignment issues between adjacent volumes of assemblies, hindering the sharing of coincident geometry in order to produce a conformal mesh.

Modeling errors caused by the user in the CAD package is another problem that the meshing package has to correct. In the CAD package, the user may not create the geometry correctly, or there may simply be very detailed components causing some parts to overlap, or introduce small gaps between parts that should touch. Many times these problems are detected in the meshing package at a point when it is not feasible to simply go back into the CAD system and fix the problem, so the meshing package must be capable of correcting it.

Several approaches for addressing the geometry cleanup problem have been proposed in the literature [2, 3, 4, 5]. They typically provide operations that are automatically applied to the geometry once one or more topology problems have been identified. While very effective in many cases, they generally lack the ability for the user to have control over the resolution of these CAD issues while still maintaining the option for automation. The proposed environment provides tools to both diagnose these common issues and to provide a list of solutions from which the user may select that will correct the problems. The specific diagnostics and solutions for dealing with small features, whether created from geometry misrepresentations or inadvertently from imprinting are first addressed.

## 2.1 Small Feature Detection and Removal

The small feature removal area of ITEM focuses on identifying and removing small features in the model that will either inhibit meshing or force excessive mesh resolution near the small feature. Small features may result from translating models from one format to another or may be intentional design features. Regardless of the origin small features must often be removed in order to generate a high quality mesh.

ITEM will recognize small features that fall in four classifications: 1) small curves, 2) small surfaces, 3) narrow surfaces, and 4) surfaces with narrow regions. These operations may involve either real, virtual or a combination of both types of operations to remove these features. A virtual operation is one in which does not modify the CAD model, but rather modifies an overlay topology on the original CAD model. Real operations, on the other hand directly modify the CAD model. Where real operations are provided by the solid modeling kernel upon which CUBIT is built, virtual operations are provided by CUBIT's CGM [6] module and are implemented independently of the solid modeling kernel. The following describes the diagnostics for finding each of the four classifications of small features and the methods for removing them.

### *Small Curves*

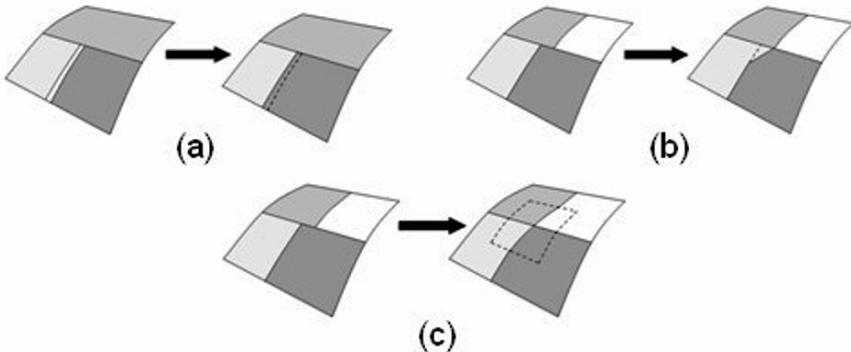
*Diagnostic:* Small curves are found by simply comparing each curve length in the model to a user-specified characteristic small curve size. A default  $\epsilon$  is automatically calculated as 10 percent of the user specified mesh size, but can be overridden by the user.

*Solutions:* ITEM provides three different solutions for eliminating small curves from the model. The first solution uses a virtual operation to composite surfaces. Two surfaces near the small curve can often be composited together to eliminate the small curve as shown in Figure 3(a)

The second solution for eliminating small curves is the collapse curve operation. This operation combines partitioning and compositing of surfaces near the small curve to generate a topology that is similar to pinching the two ends of the curve together into a single point. The partitioning can be done either as a real or virtual operation. Figure 3(b) illustrates the collapse curve operation.

The third solution for eliminating small curves is the remove topology operation. This operation can be thought of as cutting out an area around the small curve and then reconstructing the surfaces and curves in the cut-out region so that the small curves no longer exist. [7] provides a detailed description of the remove topology operation. This operation has more impact on the actual geometry of the model because it redefines surfaces and curves in the vicinity of a small curve. The reconstruction of curves and surfaces is done

using real operations followed by composites to remove extra topology introduced during the operation. Figure 3(c) shows the results using the remove topology operation.



**Fig. 3.** Three operators used for removing small curves (a) composite; (b) collapse curve; (c) remove topology

### Small and Narrow Surfaces

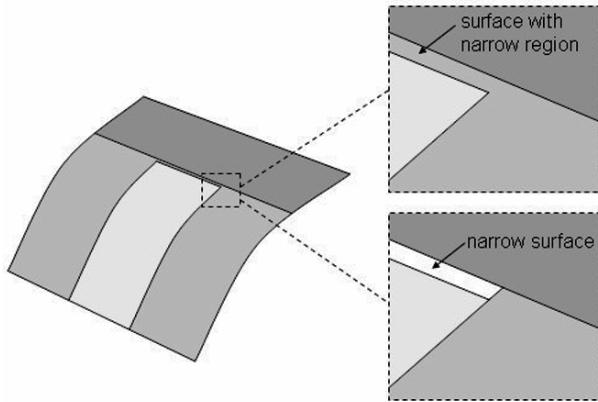
ITEM also addresses the problem of small and narrow surfaces. Both are dealt with in a similar manner and are described here.

*Diagnostic:* Small surfaces are found by comparing the surface area with a characteristic *small area*. The characteristic small area is defined simply as the characteristic small curve length squared or  $\epsilon^2$ .

*Narrow surfaces* are distinguished from *surfaces with narrow regions* by the characteristic that the latter can be split such that the narrow region is separated from the rest of the surface. Narrow surfaces are themselves a narrow region and no further splits can be done to separate the narrow region. Figure 4 shows examples of each. ITEM provides the option to split off the narrow regions, subdividing the surface so the narrow surfaces can be dealt with independently.

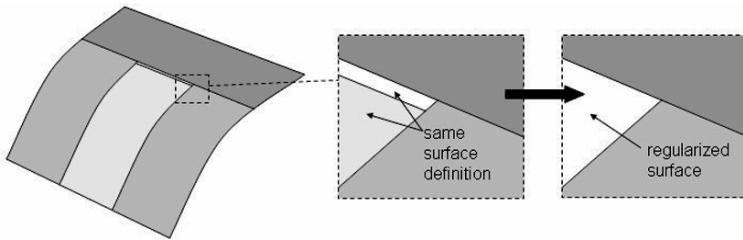
Narrow regions/surfaces are also recognized using the characteristic value of  $\epsilon$ . The distance,  $d_i$  from the endpoints of each curve in the surface to the other curves in the surface are computed and compared to  $\epsilon$ . When  $d_i < \epsilon$  other points on the curve are sampled to identify the beginning and end of the narrow region. If the narrow region encompasses the entire surface, the surface is classified as a narrow surface. If the region contains only a portion of the surface, it is classified as a surface with a narrow region.

*Solutions:* ITEM provides four different solutions for eliminating small and narrow surfaces from the model. The first solution uses the regularize operation. Regularize is a real operation provided by the solid modeling kernel



**Fig. 4.** Two cases illustrating the difference between surfaces with narrow regions and narrow surfaces

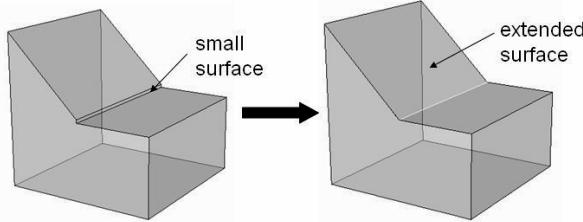
that removes unnecessary/redundant topology in the model. In many cases a small/narrow surface's normal may be the same as a surface next to it and therefore the curve between them is not necessary and can be regularized out. An example of regularizing a small/narrow surface out is shown in Figure 5.



**Fig. 5.** When the small surface's underlying geometric definition is the same as a neighbor the curve between them can be regularized out.

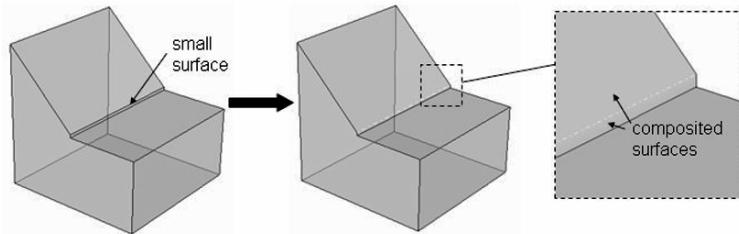
The second solution for removing small/narrow surfaces uses the remove operation. Remove is also a real operation provided by the solid modeling kernel. However, it differs from regularize in that it doesn't require the neighboring surface(s) to have the same geometric definition. Instead the remove operation removes the specified surface from the model and then attempts to extend the existing adjacent surface definitions by computing new trimming curves at their intersections to close the volume. An example of using the remove solution is shown in Figure 6.

The third solution for removing small/narrow surfaces uses the virtual composite operation to composite the small surface with one of its neighbors.



**Fig. 6.** The remove operation extends an adjacent surface to remove a small surface

This is very similar to the use of composites for removing small curves. An example is shown in Figure 7.

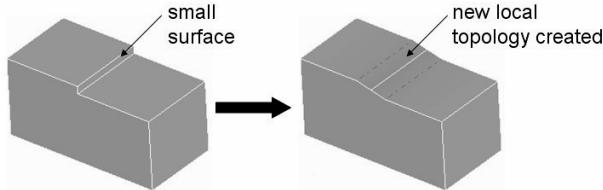


**Fig. 7.** Composite solution for removing a narrow surface

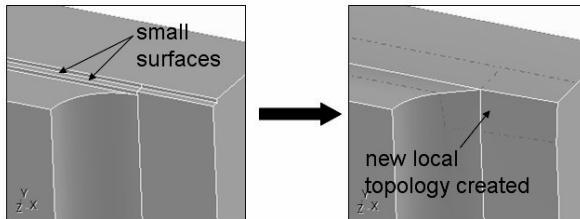
The final solution for removing small/narrow surfaces uses the remove topology operation[7]. The remove topology operation behaves the same as when used for removing small curves in that it cuts out the area of the model around the small/narrow surface and replaces it with a simplified topology. In the case of a small surface where all of the curves on the surface are smaller than the characteristic small curve length, the small surface is replaced by a single vertex. In the case of a narrow surface where the surface is longer than the characteristic small curve length in one of its directions, the surface is replaced with a curve. The remove topology operation can be thought of as a local dimensional reduction to simplify the topology. The remove topology operation can also be used to remove networks of small/narrow surfaces in a similar fashion. Examples of using the remove topology solution to remove small/narrow surfaces are shown in Figures 8 and Figure 9.

## 2.2 Resolving Problems with Conformal Assemblies

Where more than a single geometric volume is to be modeled, a variety of common problems may arise that must be resolved prior to mesh generation. These are typically a result of misaligned volumes defined in the CAD package or problems arising from the imprint and merge operations in the meshing



**Fig. 8.** Remove topology solution for removing a narrow surface



**Fig. 9.** Remove topology solution for removing a network of narrow surfaces

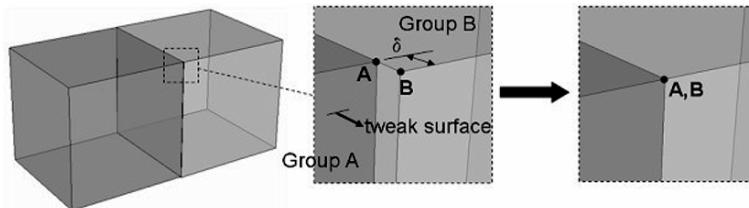
package. [8] describes the issues and proposes an automatic solution for resolving the imprint/merge problem where a discrete version of the geometry is used. ITEM addresses some of the same problems by allowing the option for user interaction as well as full automation using the CAD geometry representation. Two main diagnostics to detect potential problems are utilized: the misalignment check and overlapping surfaces check. Associated with both of these are solutions that are specific to the entity and from which the user may preview and select to resolve the problem.

#### *Resolving Misaligned Volumes*

*Diagnostics:* The *near coincident vertex check* or *misalignment check* is used to diagnose possible misalignments between adjacent volumes. This diagnostic is performed prior to the imprint operation in order to reduce the sliver surfaces and other anomalies which can occur as a result of imprinting misaligned volumes. With this diagnostic, the distance between pairs of vertices on different volumes are measured and flagged when they are just beyond the merge tolerance. The merge tolerance,  $T$ , is the maximum distance at which the geometry kernel will consider the vertices the same entity. A secondary tolerance  $T_s$  is defined where  $T_s > T$  which is used for determining which pairs of vertices may also be considered for merging. Pairs of vertices whose distance  $d$  is  $T < d > T_s$  are presented to the user, indicating areas in the model that may need to be realigned. Although not yet implemented at this writing, the misalignment check should also detect small distances between vertices and curves on adjacent volumes.

*Solutions:* When pairs of vertices are found that are slightly out of tolerance, the current solution is to move one of the surfaces containing one vertex of the pair to another surface containing the other vertex in the pair. Moving or extending a surface is known as *tweaking*. Solutions for determining which surfaces to tweak are generated as follows:

- Given that **vertex A** and **vertex B** are slightly outside of tolerance  $T$  by a distance  $\delta$  as shown in Figure 11.
- Gather all surfaces that contain **vertex A**. Call this group of surfaces **Group A**.
- Gather all surfaces that contain **vertex B**. Call this group of surfaces **Group B**.
- For each surface in **Group A**, extend it out twice its size. Call this surface **extended A**
  - See if **extended A** overlaps within a distance  $> T$  and  $< \delta$  to any surface in **Group B**.
  - If such an overlap pair is found, present two mutually exclusive solutions:
    - **tweak surface A to surface B**
    - **tweak surface B to surface A**



**Fig. 10.** Example of a solution generated to correct misaligned volumes using the tweak operator

The result of this procedure will be a list of possible solutions that will be presented to the users. They can then graphically preview the solutions and select the one that is most appropriate to correct the problem.

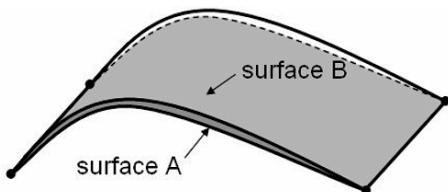
#### *Correcting Merge Problems*

The merge operation is usually performed immediately following imprinting and is also subject to occasional tolerance problems. In spite of correcting misalignments in the volume, the geometry kernel may still miss merging surfaces that may occupy the same space on adjacent volumes. If volumes in an assembly are not correctly merged, the subsequent meshes generated on the volumes will not be conformal. As a result, it is vital that all merging issues be resolved prior to meshing. The proposed environment provides a diagnostic and several solutions for addressing these issues.

*Diagnostic:* An overlapping surface check is performed to diagnose the failed sharing of topology between adjacent volumes. In contrast to the misalignment check, the check for overlapping surfaces is performed after the imprinting and merging operations. The overlapping surface check will measure the distance between surfaces on neighboring volumes to ensure that they are greater than the merge tolerance apart. Pairs of surfaces that failed to merge and that are closer than the merge tolerance are flagged and displayed to the user as potential problems.

*Solutions:* If imprinting and merging has been performed and a subsequent overlapping surface check finds overlapping surface pairs, the user may be offered three different options for correcting the problem: force merge, tolerant imprint of vertex locations and tolerant imprint of curves.

If the topology for both surfaces in the pair is identical, the force merge operation can generally be utilized. The merge operation will remove one of the surface definitions in order to share a common surface between two adjacent volumes. Normally this is done only after topology and geometry have been determined to be identical, however the force merge will bypass the geometry criteria and perform the merge. Figure 11 shows a simple example where the bounding vertices are identical but the surface definitions are slightly different so that the merge operation fails. Force merge in this case would be an ideal choice.



**Fig. 11.** Example where the merge operation will fail, but force merge will be successful

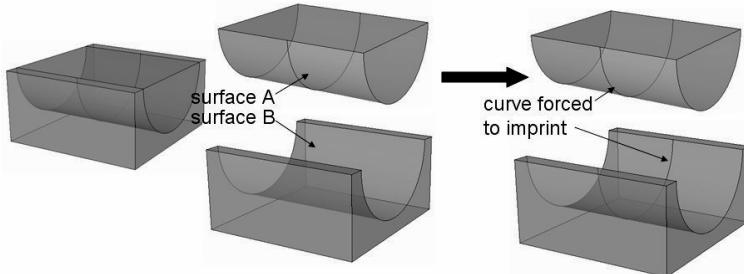
The force merge operation is presented as a solution where a pair of overlapping surfaces are detected and if any of the following criteria are satisfied:

- All curves of both surfaces are merged
- All vertices between the two surfaces are merged and all the curves are coincident to within 1% of their length or 0.005, whichever is larger
- All the curves of both surfaces are either merged or overlapping and a vertex of any curve of one surface that will imprint onto any other curve of the other surface cannot be identified
- At least one curve of one surface may be imprinted onto the other and if both surfaces have an equal number of curves and vertices, and the overlapping area between the 2 surfaces is more than 99% of the area of each surface. This situation generally prevents generating sliver surfaces

- At least one vertex of **surface B** may be imprinted onto **surface A**, and if both surfaces have equal number of curves and vertices, and the vertex(s) of **surface B** to imprint onto **surface A** lies too *close* to any vertices of **surface A**
- All the curves of both surfaces are either merged or overlapping and no vertices of any curve of **surface A** will imprint onto any other curve of **surface B**

Individual vertices may need to be imprinted in order to accomplish a successful merge. The solution of imprinting a position x,y,z onto surface A or B is presented to the user if the following criteria is met

- Curves between the two surfaces overlap within tolerance, and a vertex of **curve A** lies within tolerance to **curve B** and outside tolerance to any vertex of **curve B**. Tolerance is 0.5% of the length of the smaller of the 2 curves or the merge tolerance (0.0005), whichever is greater.



**Fig. 12.** Curve on **surface A** was not imprinted on **surface B** due to tolerance mismatch. Solution is defined to detect and imprint the curve

In some cases one or more curves may not have been correctly imprinted onto an overlapping surface which may be preventing merging. This may again be the result of a tolerance mismatch in the CAD translation. If this situation is detected a tolerant imprint operation may be performed which will attempt to imprint the curve onto the adjacent volume. Figure 12 shows an example where a curve on **surface A** is forced to imprint onto **surface B** using tolerant imprint, because it did not imprint during normal imprinting. The solution of a curve of **surface A** to be imprinted onto **surface B** may be presented to the user if all 3 of the following conditions are satisfied:

- there are no vertices to imprint onto the owning volume of either surface
- curve of **surface A** is not overlapping another curve of **surface B**
- curve of **surface A** passes tests to ensure that it is really ON **surface B**

### 3 Building a Sweepable Topology

The hex meshing problem presents a number of additional challenges to the user that tetrahedral meshing does not. Where a good quality tetrahedral mesh can generally be created once small features and imprint/merge problems have been addressed, the hexahedral meshing problem poses additional topology constraints which must be met. Although progress has been made in automating the hex meshing process, the most robust meshing algorithms still rely on geometric primitives. Mapping [9] and sub-mapping [10] algorithms rely on parametric cubes and sweeping [11, 12] relies on logical extrusions. Most real world geometries do not automatically fit into one of these categories so the topology must be changed to match the criteria for one of these meshing schemes. ITEM addresses the hex meshing topology problem through three primary diagnostic and solution mechanisms.

1. Detecting and suggesting decomposition operations
2. Recognizing nearly sweepable topologies and suggesting source-target pairs
3. Detecting and compositing surfaces to force a sweep topology.

#### 3.1 Decomposition for Sweeping

Automatic decomposition has been researched and tools have been developed which have met with some limited success [13, 14]. Automatic decomposition requires complex feature detection and sub-division algorithms. The decomposition problem is at least on the same order of difficulty as the auto-hex meshing problem. Fully automatic methods for quality hexahedral meshing have been under research and development for many years [15, 16, 17]. However, a method that can reliably generate hexahedral meshes for arbitrary volumes, without user intervention and that will build meshes of an equivalent quality to mapping and sweeping techniques, has yet to be realized. Although fully automatic techniques continue to progress [18], the objective of the proposed environment is to reduce the amount of user intervention required while utilizing the tried and true mapping and sweeping techniques as its underlying meshing engine.

Instead of trying to solve the all-hex meshing problem automatically, the ITEM approach to this problem is to maintain user interaction. The ITEM algorithms determine possible decompositions and suggest these to the user. The user can then make the decision as to whether a particular cut is actually useful. This process helps guide new users by demonstrating the types of decompositions that may be useful. It also aids experienced users by reducing the amount of time required to set up decomposition commands.

*Diagnostics:* The current diagnostic for determining whether a volume is mappable or sweepable is based upon the autoscheme tool described in [19]. Given

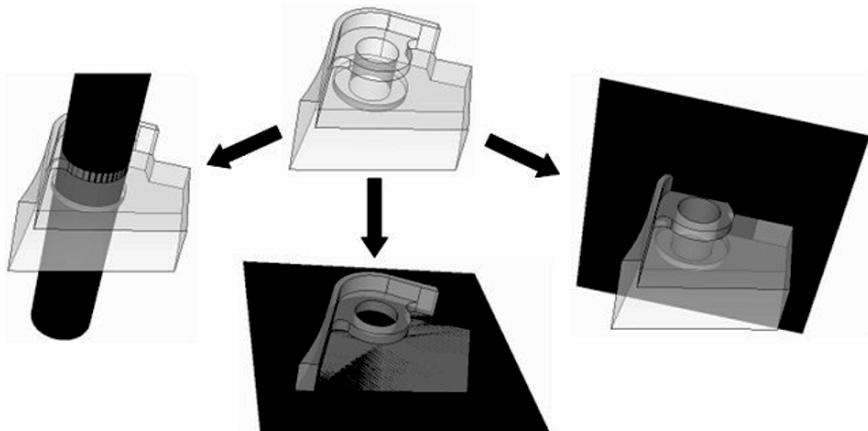
a volume, the autoscheme tool will determine if the topology will admit a mapping, sub-mapping or sweeping meshing scheme. For volumes where a scheme cannot be adequately determined, a set of decomposition solutions are generated and presented to the user.

*Solutions:* The current algorithm for determining possible cut locations is based on the algorithm outlined in [13] and is described here for clarity:

- Find all curves that form a dihedral angle less than an input value (currently  $135^\circ$ )
- Build a graph of these curves to determine connectivity
- Find all curves that form closed loops
- For each closed loop
  - Find the surfaces that bound the closed loop
  - Save the surface
  - Remove the curves in the closed loop from the processing list
- For each remaining curve
  - Find the open loops that terminates at a boundary
  - For each open loop
    - Find the surfaces that bound the open loop
    - Save the surfaces
- For each saved surface
  - Create an extension of the surface
  - Present the extended surface to the user as a possible decomposition location.

This relatively simple algorithm detects many cases that are useful in decomposing a volume. Future work will include determining symmetry, sweep, and cylindrical core decompositions. These additional decomposition options should increase the likelihood of properly decomposing a volume for hexahedral meshing.

Figure 13 shows an example scenario for using this tool. The simple model at the top is analyzed using the above algorithm. This results in several different solutions being offered to the user, three of which are illustrated here. As each of the options is selected, the extended cutting surface is displayed providing rapid feedback to the user as to the utility of the given option. Note that all solutions may not result in a volume that is closer to being successfully hex-meshed. Instead the system relies on some user understanding of the topology required for sweeping. Each time a decomposition solution is selected and performed, additional volumes may be added, which will in turn be analyzed by the autoscheme diagnostic tool. This interactive process continues until the volume is successfully decomposed into a set of volumes which are recognized as either mappable or sweepable.



**Fig. 13.** ITEM decomposition tool shows 3 of the several solutions generated that can be selected to decompose the model for hex meshing

### 3.2 Recognizing Nearly Sweepable Regions

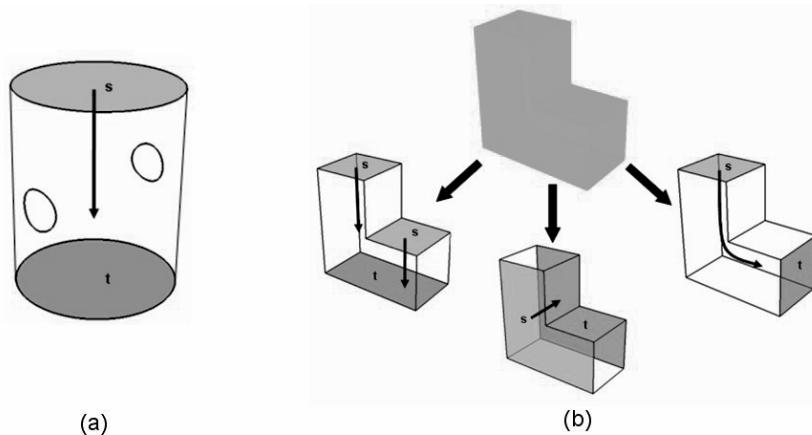
The purpose of geometry operations such as decomposition is to transform an unmeshable region into one or more meshable regions. However, even the operations suggested by the decomposition tool can degenerate into guesswork if they are not performed with a specific purpose in mind. Without a geometric goal to work toward, it can be difficult to recognize whether a particular operation will be useful.

Incorporated within the proposed ITEM environment are algorithms that are able to detect geometry that is nearly sweepable, but which are not fully sweepable due to some geometric feature or due to incompatible constraints between adjacent sections of geometry. By presenting potential sweeping configurations to the user, ITEM provides suggested goals to work towards, enabling the user to make informed decisions while preparing geometry for meshing.

Unlike the decomposition solutions presented in the previous section, the purpose of recognizing nearly sweepable regions is to show potential alternative source-target pairs for sweeping even when the autoscheme tool does not recognize the topology as strictly sweepable. When combined with the decomposition solutions and the forced sweep capability described later, it provides the user with an additional powerful strategy for building a hexahedral mesh topology.

*Diagnostics:* In recognizing nearly sweepable regions, the diagnostic tool employed is once again the autoscheme tool described in [19]. Volumes that do not meet the criteria defined for mapping or sweeping are presented to the user. The user may then select from these volume for which potential source-target pairs are computed.

*Solutions:* The current algorithm for determining possible sweep configurations is an extension of the autoscheme algorithm described in [19]. Instead of rejecting a configuration which does not meet the required sweeping constraints, the sweep suggestion algorithm ignores certain sweeping roadblocks until it has identified a nearly feasible sweeping configuration. The suggestions are presented graphically, as seen in Figure 14(a). In most cases, the source-target pairs presented by the sweep suggestion algorithm are not yet feasible for sweeping given the current topology. The user may use this information for further decomposition or to apply solutions identified by the forced sweepability capability described next. The sweep suggest algorithm also provides the user with alternative feasible sweep direction solutions as shown in Figure 14(b). This is particularly useful when dealing with interconnected volumes where sweep directions are dependent on neighboring volumes.



**Fig. 14.** (a) ITEM displays the source and target of a geometry that is nearly sweepable. The region is not currently sweepable due to circular imprints on the side of the cylinder. (b) Alternative feasible sweep directions are also computed

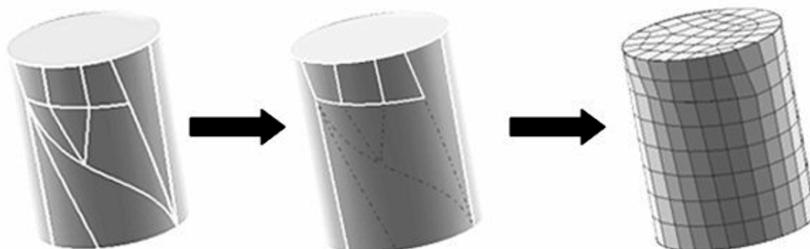
### 3.3 Forced Sweepability

In some cases, decomposition alone is not sufficient to provide the necessary topology for sweeping. The forced sweepability capability attempts to force a model to have sweepable topology given a set of source and target surfaces. The source-target pairs may have been identified manually by the user, or defined as one of the solutions from the sweep suggestion algorithm described above. All of the surfaces between source and target surfaces are referred to as linking surfaces. Linking surfaces must be mappable or submappable in order for the sweeping algorithm to be successful. There are various topol-

ogy configurations that will prevent linking surfaces from being mappable or submappable.

*Diagnostics:* The first check that is made is for small curves. Small curves will not necessarily introduce topology that is not mappable or submappable but will often enforce unneeded mesh resolution and will often degrade mesh quality as the mesh size has to transition from small to large. Next, the interior angles of each surface are checked to see if they deviate far from 90° multiples. As the deviation from 90° multiples increases the mapping and submapping algorithms have a harder time classifying corners in the surface. If either of these checks identify potential problems they are flagged and potential solutions are generated.

*Solutions:* If linking surface problems are identified ITEM will analyze the surface and generate potential solutions for resolving the problem. Compositing the problem linking surface with one of its neighbors is a current solution that is provided. ITEM will look at the neighboring surfaces to decide which combination will be best. When remedying bad interior angles the new interior angles that would result after the composite are calculated in order to choose the composite that would produce the best interior angles. Another criterion that is considered is the dihedral angle between the composite candidates. Dihedral angles close to 180° are desirable. The suggested solutions are prioritized based on these criteria before being presented to the user. Figure 15 shows an example of a model before and after running the forced sweepability solutions. The top and bottom of the cylinder were chosen as the source and target surfaces respectively.



**Fig. 15.** Non-submappable linking surface topology is composited out to force a sweepable volume topology

## 4 Mesh Quality

Advancements in the mesh generation algorithms have significantly reduced the amount of quality problems seen in the initially generated mesh. Further, ITEM generally relies on the most robust meshing algorithms available

in CUBIT, specifically sweeping for hexahedral mesh generation[12] and the Tetmesh-GHS3D[20] meshing software<sup>6</sup>. However, some problems can still exist, and therefore ITEM has integrated quality diagnostics and solution options.

*Diagnostics:* After the mesh has been generated, the user may choose to perform element quality checks. ITEM utilizes the Verdict[21] library where a large number of mesh quality metrics have been defined and available as a modular library. If no user preference is specified, ITEM uses the Scaled Jacobian distortion metric to determine element quality. This check will warn users of any elements that are below a default or user-specified threshold, allowing various visualization options for displaying element quality.

*Solutions:* If the current element quality is unacceptable, ITEM will present several possible mesh improvement solutions. The most promising solutions are provided through ITEM's interface to two smoothers: mean ratio optimization and Laplacian smoothing. These are provided as part of the Mesquite[22] mesh quality improvement tool built within CUBIT. The user has the option of performing these improvements on the entire mesh, subsets of the mesh defined by the element quality groups, or on individual elements. The Laplacian smoothing scheme allows the users to smooth just the interior nodes or to simultaneously smooth both the interior and boundary nodes in an attempt to improve surface element quality.

## 5 Conclusion

A new approach to presenting the problem of preparing a finite element mesh to an intermittent user of modeling and simulation technology has been proposed. The Immersive Topology Environment for Meshing (ITEM) addresses a wide range of problems and issues commonly encountered during this process. Its intent is to reduce the learning, and re-learning often associated with complex software tools and to ultimately reduce the time to mesh. This is accomplished through a step-by-step wizard-like approach where users may address common problems by using built-in diagnostics and are then presented with specific intelligent solutions to these problems.

Table 1 summarizes the problems addressed by the proposed environment along with associated diagnostics and solutions. Details of each of the diagnostics and solutions are discussed within the body of the paper.

At this writing the proposed ITEM environment is still under development with plans for release shortly. The current set of diagnostics and solutions defined in Table 1 represent a reasonable set of tools for preparing models for analysis, however it is recognized that these tools will be modified, tuned and expanded based on user feedback and experience and as new technology is developed.

---

<sup>6</sup><http://www.distene.com>

Problem	Diagnostic	Solutions
Small Curves	Curve length $< \epsilon$	1. composite surfaces 2. collapse curve 3. remove topology
Small Surfaces	Surface area $< \epsilon^2$	1. regularize
Narrow Surfaces	$d_i < \epsilon$ for all curves on surface	2. remove/extend surfaces 3. composite surfaces 4. remove topology
Surfaces with Narrow Regions	$d_i < \epsilon$ for some curves on surface	1. split off narrow region and treat as narrow surface
Misaligned volumes	Near coincident vertex or misalignment check	1. tweak surf A to surf B 2. tweak surf B to surf A
Unmerged surfaces	Overlapping surfaces check	1. force merge 2. imprint vertices 3. imprint curves
Non-sweepable/mappable topology	Autoscheme tool	1. cut locations based upon dihedral angles and connectivity graph
Nearly sweepable	Autoscheme tool + sweep suggestions	1. suggested source/target pairs
Linking surfaces not mappable	Linking surfaces: 1. Curve length $< \epsilon$ 2. Interior angles deviate significantly from $90^\circ$	1. composite surfaces
Poor mesh quality	Quality metric $<$ threshold	Mean Ratio or Laplacian smoothing applied to: 1. entire mesh 2. element quality group 3. individual elements

**Table 1.** Summary of problems and associated diagnostics and solutions that are addressed with ITEM

Prior to release, extensive user testing will be performed in order to determine the impact that ITEM has on the time to mesh. This will include a series of prescribed models that will be meshed by several intermittent users of meshing software. Metrics will be gathered comparing times to complete the mesh in ITEM compared with previous technology. Understanding the difficulty of accurately measuring the time to mesh, the testing and metrics gathering procedure will attempt to control for factors including learning, user expertise and model complexity. These factors are outlined as follows:

1. *Learning*: Much of the meshing procedure involves a trial and error process of learning a strategy for model cleanup and decomposition. The second time a user attempts to mesh a model, they will very likely be able to be more efficient regardless of which system they use. As a result, a single tester will not attempt the same model more than once regardless of which system they use. The order in which they use ITEM and the previous technology will also be interspersed to randomize the effect of learning one system over the other.
2. *User Expertise*: Depending on how much experience a particular user has with a specific software system will effect how quickly they can complete a task compared to others with much less experience. An attempt will be made to enlist analysts with equivalent experience, however the timed results for any one particular model will be averaged across all users to reduce the effect of user expertise.
3. *Model Complexity*: Very complex models will inherently take longer to prepare and mesh than models of less complexity. Averaging the time taken for all models for any one user should reduce the effect of model complexity.

Since many diverse human factors may be involved, it is clear that any solution to gather metrics to gauge improved time to mesh will be flawed in some way. It is however healthy and important to implement these measurements to independently measure the effectiveness of ITEM or any proposed system claiming to reduce the time to mesh. The results will ultimately provide insights and new input for further improvement.

While it is recognized that it is still a work in progress, the main contribution of the current work includes the infrastructure proposed for presenting and managing the model preparation process and its potential impact on reducing the time to generate analysis models.

## References

1. Hardwick, Mike (2005) DART System Analysis Presented to Simulation Sciences Seminar, Sandia National Laboratories, Albuquerque, NM June 28, 2005
2. Tautges, Timothy J. (2001), Automatic Detail Reduction for Mesh Generation Applications, *Proceedings, 10th International Meshing Roundtable*, pp.407-418

3. Sheffer, A. (2001), Model simplification for meshing using face clustering, *Computer-Aided Design*, Vol. 33, No. 13, pp. 925-934(10)
4. Butlin, Geoffrey and Clive Stops (1996) CAD Data Repair, *5th International Meshing Roundtable*, pp.7-12
5. Mezentsev, Andrey A. (1999) Methods and Algorithms of Automated CAD Repair For Incremental Surface Meshing, *Proceedings, 8th International Meshing Roundtable*, pp.299-309
6. Tautges, Timothy J. (2000) The Common Geometry Module (CGM): A Generic, Extensible Geometry Interface, *Proceedings, 9th International Meshing Roundtable*, pp. 337-348
7. Clark Brett W. (2007) Removing Small Features with Real Solid Modeling Operations, Submitted to *16th International Meshing Roundtable*
8. White, David R. and Sunil Saigal (2002) Improved Imprint and Merge for Conformal Meshing, *Proceedings, 11th International Meshing Roundtable*, pp.285-296,
9. Cook, W. A. and W. R. Oakes (1982) Mapping methods for generating three-dimensional meshes, *Computers In Mechanical Engineering, CIME Research Supplement*:67-72, August 1982
10. Whiteley, M., D. White, S. Benzley and T. Blacker (1996) Two and Three-Quarter Dimensional Meshing Facilitators, *Engineering with Computers*, Vol 12, pp. 155-167
11. Knupp, Patrick M. (1998) Next-Generation Sweep Tool: A Method For Generating All-Hex Meshes On Two-And-One-Half Dimensional Geomtries, *Proceedings, 7th International Meshing Roundtable*, pp. 505-513
12. Scott, Michael A., Matthew N. Earp, Steven E. Benzley and Michael B. Stephenson (2005) Adaptive Sweeping Techniques, *Proceedings, 14th International Meshing Roundtable*, pp. 417-432
13. Yong Lu, Rajit Gadh, and Timothy J. Tautges (1999) Volume decomposition and feature recognition for hexahedral mesh generation, *Proceedings, 8th International Meshing Roundtable*, pp. 269-280
14. Staten, Matthew L., Steven J. Owen, Ted D. Blacker (2005) Unconstrained Paving & Plastering: A New Idea for All Hexahedral Mesh Generation, *Proceedings, 14th International Meshing Roundtable*, pp.399-416
15. Blacker T.D. and Meyers R.J. (1993) Seams and Wedges in Plastering:A 3D Hexahedral Mesh Generation Algorithm, *Engineering with Computers*, Vol. 2, No. 9, pp. 83-93
16. Folwell, Nathan T. and Scott A. Mitchell (1998) Reliable Whisker Weaving via Curve Contraction, *Proceedings, 7th International Meshing Roundtable*, pp.365-378
17. Price, M.A. and C.G. Armstrong (1995) Hexahedral Mesh Generation by Medial Surface Subdivision: Part I, Solids With Convex Edges, *International Journal for Numerical Methods in Engineering*, Vol. 38, No. 19, pp. 3335-3359
18. Staten, Matthew L., Robert A. Kerr, Steven J. Owen, Ted D. Blacker (2006) Unconstrained Paving and Plastering: Progress Update, *Proceedings, 15th International Meshing Roundtable*, pp.469-486
19. White, David R. and Timothy J. Tautges (2000) Automatic scheme selection for toolkit hex meshing, *International Journal for Numerical Methods in Engineering*, Vol. 49, No. 1, pp. 127-144

20. George, P.L., F. Hecht and E. Saltel (1991) Automatic Mesh Generator with Specified Boundary, *Computer Methods in Applied Mechanics and Engineering*, Vol. 92, pp. 269-288
21. Stimpson, CJ, Ernst, CD, Knupp, P, Pebay; P, and Thompson, D. (2007) The Verdict Geometric Quality Library, *Sandia Report SAND2007-1751*
22. Brewer, Michael, Lori Freitag Diachin, Patrick Knupp, Thomas Leurent and Darryl Melander (2003) The Mesquite Mesh Quality Improvement Toolkit, *Proceedings, 12th International Meshing Roundtable*, pp. 239-250