Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 124 (2015) 226 – 238

Procedia Engineering

www.elsevier.com/locate/procedia

24th International Meshing Roundtable (IMR24)

Generating analysis topology using virtual topology operators

Christopher M Tierney[a]*, Liang Sun[a], Trevor T Robinson[a], Cecil G Armstrong[a]

[a]*Queens University Belfast, School of Mechanical and Aerospace Engineering, Ashby Building, Beflast BT9 5AH, Northern Ireland*

Abstract

Virtual topology operations have been utilized to generate an analysis topology definition suitable for downstream mesh generation. Detailed descriptions are provided for virtual topology merge and split operations for all topological entities, where virtual decompositions are robustly linked to the underlying geometry. Current virtual topology technology is extended to allow the virtual partitioning of volume cells. A valid description of the topology, including relative orientations, is maintained which enables downstream interrogations to be performed on the analysis topology description, such as determining if a specific meshing strategy can be applied to the virtual volume cells. As the virtual representation is a true non-manifold description of the sub-divided domain the interfaces between cells are recorded automatically. Therefore, the advantages of non-manifold modelling are exploited within the manifold modelling environment of a major commercial CAD system without any adaptation of the underlying CAD model. A hierarchical virtual structure is maintained where virtual entities are merged or partitioned. This has a major benefit over existing solutions as the virtual dependencies here are stored in an open and accessible manner, providing the analyst with the freedom to create, modify and edit the analysis topology in any preferred sequence.

Peer-review under responsibility of organizing committee of the 24th International Meshing Roundtable (IMR24)

Keywords: Virtual Topology; decomposition; hex meshing algorithms

1. Introduction

Geometry preparation and manipulation for meshing is an important aspect of finite element analysis (FEA) processes, but can often consume an overwhelming proportion of the overall analysis cycle [1]. The geometry handling cost heavily depends on the analysis being performed, the toolsets being used and the expertise of the

* Corresponding author. Tel.: +00442890974277.
E-mail address: christopher.tierney @qub.ac.uk

Peer-review under responsibility of organizing committee of the 24th International Meshing Roundtable (IMR24)
doi:10.1016/j.proeng.2015.10.135

analyst. For example, tetrahedral (Tet), hexahedral (Hex), mixed-solid element or mixed dimensional meshes may be appropriate depending on the type of analysis being performed. The meshing style in turn dictates the level of geometry manipulation necessary to achieve a fit-for-purpose mesh. Suitable Tet meshes are frequently generated using only geometry clean-up operations to remove features that may hinder the mesh generator. In contrast, due to the lack of robust automated Hex mesh generation technology, it is often necessary to utilize decomposition techniques in addition to geometry clean-up in order to successfully create a Hex mesh. Decomposition methods partition complex domains into simpler sub-domains which can be readily Hex meshed using automated algorithms such as mapping, sub-mapping, mid-point subdivision and sweeping.

The time spent executing the preferred clean-up and decomposition operations are influenced by the toolsets selected for each task. A plethora of geometry-based tools exist for directly modifying the geometry of domain. These tools tend to be time intensive when used by the analyst and can produce discrepancies in the model. An example is the use of sew or join operations to merge faces for geometry clean-up, where there is no guarantee a surface definition for the combined entities can be generated. Furthermore, these geometry-based CAD approaches require analysts to have significant CAD modelling experience, which imposes a significant and unnecessary burden on the analyst. Direct geometry modifications, their overheads, and sometimes unpredictable results can be avoided using virtual topology operations [2] for model editing. The principal role of virtual topology is to modify the topology whilst leaving the geometry undisturbed. Other notable advantages include the ease of reversing virtual operations in comparison to geometry-based operations and the analyst-friendly nature of virtual topology toolsets. Virtual topology is one of the underlying technologies in Simulation Intent [3]. Defining Simulation Intent involves capturing high level modelling and idealization decisions in order to create an efficient and fit-for-purpose analysis. For example, if an analyst specifies the desired solution accuracy and time available then automated decisions can be made which dictate analysis inputs such as mesh type or target element size in order to achieve the desired results. Furthermore, the target element size dictates the input metric for virtual topology clean-up operations based on entity size. Whilst virtual topology is commonly used for geometry clean-up it has yet to be entirely embraced for volume decomposition for meshing where complex domains can be Hex meshed using a divide-and-conquer approach to generate meshable primitives. This work describes a virtual topology implementation that has been embedded within a commercial CAD package and linked to the underlying CAD geometry. Virtual decomposition is implemented for an industrial use-case model based upon the input from a thin- sheet identification tool. The generation of an analysis topology using virtual topology operations enables tools to operate in the presence of virtual topology including the identification of appropriate meshing strategies for subset cells.

2. State-of-the art virtual topology implementations

Virtual topology is not new [4], but through using a number of state-of-the-art geometry pre-processing and meshing packages it is apparent that its advantages are not yet readily available in all of them and there is no agreed description for virtual topology which can be exported between packages. Some of the packages considered were Abaqus CAE [5], Ansys Workbench [6], CUBIT [7], ICEM CFD [8] and Siemens NX [9].

The aforementioned packages predominantly recognize virtual topology as a geometry clean-up tool for mesh generation. Typically a series of virtual topology operations are available to merge small edges/faces below the target element size, collapse small angles or to partition entities to constrain the mesh direction and/or size for boundary condition application. The availability of these virtual topology capabilities vary between the different CAE packages as does the level to which they can be accessed through the user interface and API.

One drawback to current implementations is that many packages do not allow virtual topology to be viewed, interrogated or edited once it has been applied. Restricting the analyst from viewing the virtual topology that has been defined is a major hindrance within analysis workflows, especially for complex models where numerous virtual operations are utilized. This leaves it extremely challenging for different analysts to share models and interpret the abstraction decisions of others. The lack of accessibility means the analyst is restricted from using favored toolsets within different aspects of the analysis process. It has been demonstrated in earlier work [10] that the ability to extract and store virtual topology relationships enables reuse within other packages.

Some vendors have elected to implement virtual topology on a polygonal representation of the design model. This simplifies the automatic detection and definition of virtual topology operations but relies on a robust

relationship between geometric and polygonal representations. This can compromise the ability to preserve virtual topology after even simple design changes. A simple yet powerful illustration of this is shown in Fig. 1 (a) for a geometric model generated within Siemens NX. Virtual merge operations are used to remove the edges highlighted in Fig. 1 (a) and create the simplified virtual model in Fig. 1 (b). Once the geometric representation is edited and the chamfer size increased, Fig. 1 (c), the previously applied virtual operations are no longer in the model, Fig. 1 (d).

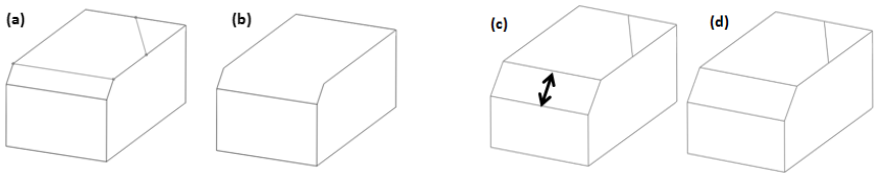

Fig. 1. (a) Base geometric model; (b) Virtual polygonal representation; (c) Chamfer size increased; (d) Virtual topology operations reset.

One common theme amongst the aforementioned meshing packages is the lack of robust virtual splitting operations for volume cells. Virtual volume partitioning would allow an analyst to virtually decompose a domain for Hex meshing. Splitting operations exist to partition edges and faces and have been utilized to decompose domains and generate a virtual topology network of multi-block faces for quad meshing [11]. Sheffer [12] described the generation of Hex meshes using a medial axis approach to virtually decompose an initial volume into sub-volumes meshable by means of basic meshing methods. The virtual volume decomposition implemented within the GAMBIT meshing suite [13] imposes the constraint that all edges of the partitioning faces are required to be connected to the volume to be partitioned resulting in multiple connected partitioning faces needing to be merged. In other work by Sheffer [14] virtual volume decomposition was introduced but lacked detailed description of its implementation. CUBIT offers a virtual volume partitioning solution at the mesh level where a meshed domain can be partitioned by specifying distance criteria to select elements and define the partitioning location. In similar work a volume sub-mapping approach utilizes a virtual decomposition to pre-process the geometry for hex meshing but requires that all boundary surfaces have been meshed with a quadrilateral mapping algorithm [15]. ICEM CFD [8] provides a block decomposition solution by manually creating and overlaying a network of simple block topologies on top of the geometry. The block topology can be thought of as a virtual representation of the geometry but without having to carry out the decomposition as links to the geometry tend to be manually specified in order to capture the underlying shape of the domain. These virtual operations tend to be implicit whereby multiple block faces can be linked to a single geometric face or a single block face can represent multiple geometric faces. These can be thought of as virtual split and merge operations. Apart from the virtual decomposition contributions mentioned above most packages and research tend to primarily use direct geometric operations to decompose volume cells [16]. Successful partitioning of the geometric representation can be troublesome, especially where non-manifold conditions (shared faces, dangling edges etc.) are generated within manifold environments. Even if the domain has been successfully decomposed it is common for unwanted geometric and topological entities, such as sliver faces, to be generated which require further work to remove. The use of direct geometric partitioning alongside a virtual description defined on a polygonal representation demands that partitioning operations need to be carried out prior to any virtual operations. This can be detrimental, especially if the virtual topology fails to update. In this work it is shown that it should be possible to execute virtual merge or split operations in any order desired by the analyst. This is achieved by storing the virtual topology dependencies alongside a description of the analysis topology. It is described herein how virtual merge and split procedures, including volume decomposition, are implemented and how the relationship between the virtual entities and the analysis topology is stored. Formulation of an analysis topology enables multiple versions of the original topology to be generated, interrogated and manipulated as deemed necessary by an analyst.

3. Defining virtual topology operations

This section describes the implementation of virtual merge and split operations and the generation of an appropriate analysis topology, where entities (real or virtual) can be merged or partitioned. A valid topological description, including the relative orientations of entities, is maintained within the analysis topology definition

which helps downstream virtual operations and enables the analysis topology to be interrogated for meshing purposes. Entity labels are assigned within the database to be unique identifiers, but have been labelled here based on entity type (f for face, e for edge and v for vertex).

3.1. Virtual merge operations

Virtual merge operations create a single virtual superset entity by removing the common lower dimension topology between adjacent entities being merged. Entities being merged must be adjacent and have the same manifold dimension. They are referred to as host entities of the virtual superset entity, which inherits the manifold dimension of the hosts.

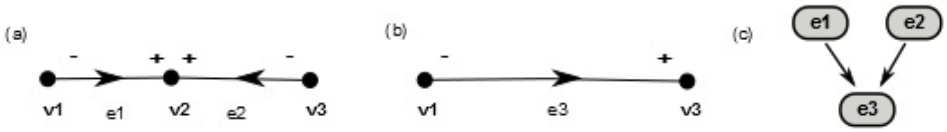

Fig. 2. (a) Host edges to be merged; (b) virtual superset edge; (c) virtual dependencies.

Fig. 2 illustrates the merge operation for edge entities. Adjacent edges ‘e1’ and ‘e2’ are shown in Fig. 2 (a) along with their bounding vertices and the orientation of each vertex relative to its bounded edge. If the underlying edge tangent points towards a vertex then it is assigned a positive orientation relative to the edge, otherwise it is negative, Fig. 2(a). The bounding topology of the virtual superset edge, ‘e3’, is established by finding the uncommon boundary, vertices ‘v1’ and ‘v3’, between the host edges being merged, Fig. 2 (b). Consistent orientations of the bounding vertices are maintained by selecting the orientation of the first uncommon vertex, ‘v1’, and assigning the second vertex, ‘v3’, in the opposite direction. The orientation of the superset edge relative to any bounded faces is set to be the same as the host edge bounded by the first uncommon vertex. The virtual topology dependencies between the virtual and host faces is recorded, Fig. 2 (c).

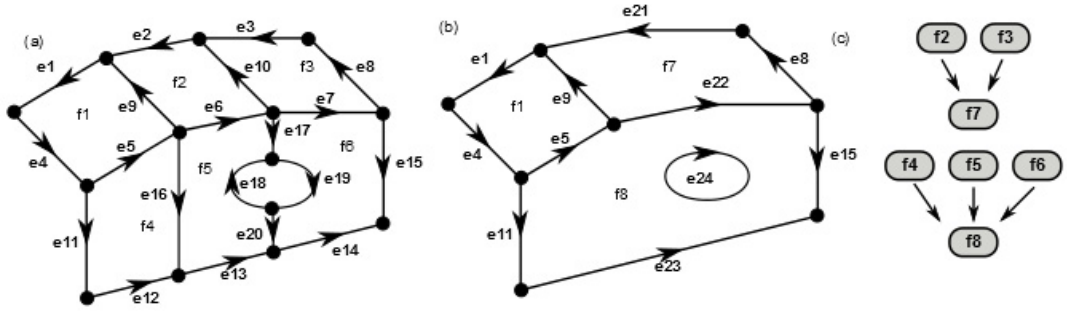

Fig. 3. (a) Host faces to be merged; (b) resulting analysis topology; (c) virtual dependencies of face cells.

Virtual superset faces can be defined by specifying either the faces to be merged or the edges to be ignored (i.e. the common edges between faces to be merged). Either approach is sufficient when creating virtual superset faces one-by-one. However, this is not the case when attempting to create many at once, which is useful when attempting to automate geometry clean-up procedures. Consider Fig. 3 where the objective is to generate two superset faces ‘f7’ and ‘f8’, Fig. 3 (b), by merging faces ‘f2’, ‘f3’, ‘f4’, ‘f5’ and ‘f6’, Fig 3 (a). Simply providing the faces to be merged as the input leaves it impossible to interpret the intent of the analyst and generate the two superset face, ‘f7’ and ‘f8’. Instead it is likely the tool will attempt to create one face. However, if the input provided is the edges to be ignored, ‘e10’, ‘e16’, ‘e17’ and ‘e20’, then by finding the bounded face of each edge, {f2, f3}, {f4, f5} and {f5, f6} respectively, the supersets are formed by recursively cycling through each face pair, and grouping common face pairs. This also enforces the condition that faces to be merged must be adjacent. From this it can be seen that superset faces ‘f7’ and ‘f8’ comprising of host faces {f2, f3} and {f4, f5, f6} are identified. The virtual dependencies for the superset faces are shown in Fig. 3 (c).

In order to create the topology of the superset faces, associated bounding entity topologies must also be merged. Since the edges to be ignored are used as the input their bounding vertices may also be ignored. The bounded edge pairs are returned for each vertex to be ignored and grouped into host edge sets as per the description for the face pairs above. Once the host edge sets are identified the merge operation for edges is used to create the superset edges. It should be noted that certain vertices are not ignored if they bound an edge which does not form the boundary of the superset face or is not ignored as part of another merge operation, e.g. the vertex bounding edge ‘e16’ bounds more than two edges (it is common to edges ‘e9’, ‘e5’ and ‘e6’). It is important that this is not ignored as edge ‘e9’ bounds face ‘f1’ and thus remains in the resulting analysis topology, Fig. 3 (b). Edge loop types are liable to be altered within the analysis topology. For example, edges ‘e18’ and ‘e19’ are classed as outer loop edges of faces ‘f5’ and ‘f6’. However, after host faces ‘f5’ and ‘f6’ are merged the common edges ‘e17’ and ‘e20’ are removed from the analysis topology and superset edge ‘e24’ becomes an inner loop periodic edge as it originally existed as two edges ‘e18’ and ‘19’ are merged through the removal of ‘e17’ and ‘e20’ and their bounding vertices. These situations are recognized when there are multiple unconnected common edges between the host faces that were merged, e.g. edges ‘e17’ and ‘e20’ from Fig. 3 (a).

An edge is considered positive relative to its bounded face if the face is on the left-hand-side of the forward pointing edge tangent direction when viewed from the outward pointing face normal. For example, edges ‘e1’, ‘e4’, ‘e5’ and ‘e9’ are all positive relative to face f1. However, edge ‘e5’ is negative relative to its other bounded face ‘f4’. This orientation convention is transferred to maintain a valid topological description within the analysis topology, Fig. 3 (b).

3.2. Virtual split operations

Virtual subset entities represent a partial section of a host entity. Parasite entities are Virtual bounding entities used to partition a higher dimensional topological entity into virtual subset entities, e.g. a parasite face used to partition a volume cell into simpler subset volumes for meshing.

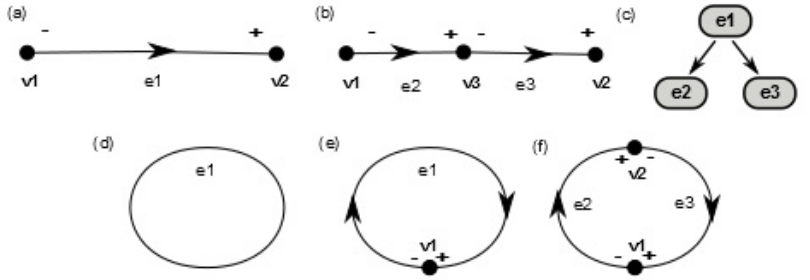

Fig. 4. Edge splitting operations: (a) Host edge to be split; (b) resulting analysis topology; (c) virtual subset dependencies; (d) periodic edge to be partitioned; (e) parasite vertex introduced to periodic edge; (f) periodic edge portioned with second parasite vertex.

Split operations are described for possible edge configurations. The first configuration is an edge bounded by two vertices, Fig. 4 (a). In this scenario the addition of parasite vertex ‘v3’ to the boundary of host edge e1 results in two subset edges ‘e2’ and ‘e3’. The orientation of the parasite ‘v3’ is assigned opposite to the original vertices bounding each virtual edge, e.g. where subset edge ‘e2’ is bounded by negative vertex ‘v1’ this dictates that parasite ‘v3’ is positive relative to subset ‘e2’. The second configuration is a periodic edge bounded by zero vertices, Fig. 4 (d). Here the addition of a single parasite vertex ‘v1’ is insufficient to partition the host edge ‘e1’ as the boundary traversal arrives back at the initial vertex ‘v1’, Fig. 4 (e). The parasite vertex is simply added to the topology of the edge and assigned both a positive and negative orientation. If another operation in the analysis model preparation requires a second parasite vertex to be added to the edge topology then two subset edges result, Fig. 4 (f). The orientation of the parasite vertices relative to the subset edge is determined by their parametric value within the host edge. Periodic edge configurations are extensible to periodic surfaces, such as cylinders. This approach is able to robustly handle any configuration of periodic edges and faces within the virtual representation. Links to any topological CAD architecture are maintained as bounding entities are often introduced to periodic entities to

simplify the tracking of the topological connectivity. Here a periodic edge or face can be bounded by zero, one, two or n entities and can also be created for analysis purposes (‘e24’ Fig. 3 (b)) to reduce the propagation of splits through the analysis model.

Face partitioning is carried out by introducing a lower dimensional parasite edge into the host face topology. Due to the possible presence of multiple edge loops a wider range of configurations need considered for face partitioning. Fig. 5 (a) and (b) describe a situation where a single parasite edge is sufficient to partition the face. Firstly the bounding vertices of the parasite edge are used to partition the bounding edges of the host face, e.g. ‘v5’ and ‘v6’ partition host edges ‘e2’ and ‘e4’ into virtual subset edges {e6, e7} and {e8, e9}respectively. Then a face traversal procedure is utilized to determine whether a parasite edge partitions the host face. The traversal process begins from the first bounding vertex ‘v5’ of the parasite edge ‘e10’. From this vertex the host face topology is traversed in the positive direction until either the input vertex ‘v5’ or the second bounding vertex ‘v6’ of the parasite edge is met. If the start vertex is met then the face has not been partitioned whereas if the second vertex is met then the face has been partitioned. The edges traversed, edges ‘e6’, ‘e1’ and ‘e8’ form the boundary of the first virtual subset face along with the final traversal over the parasite boundary, which in this case is simply the parasite edge itself. The orientation of the parasite edge is inherited from the previous subset edge in the traversal direction. This is decided by identifying the common vertex and comparing its orientation relative to each edge. If the vertex has the same orientation relative to each edge then the parasite edge is assigned an opposite orientation to the subset edge. If the vertex orientations differ between edges then the parasite edge inherits the orientation of the subset edge. Therefore, edge ‘e10’ is assigned positive relative to subset face ‘f2’ and negative relative to subset ‘f3’. The boundary of the second subset face is defined from the remaining edges bounding the host face, excluding inner loop edges. Inner loop edges, such as ‘e5’, are assigned to the boundary of a virtual subset face by determining if the inner edge lies within the outer boundary parameters space of each subset. This returns the boundary of two subset faces ‘f2’ and ‘f3’ to be {e6, e1, e8, e10} and {e7, e3, e9, e10, e5} respectively.

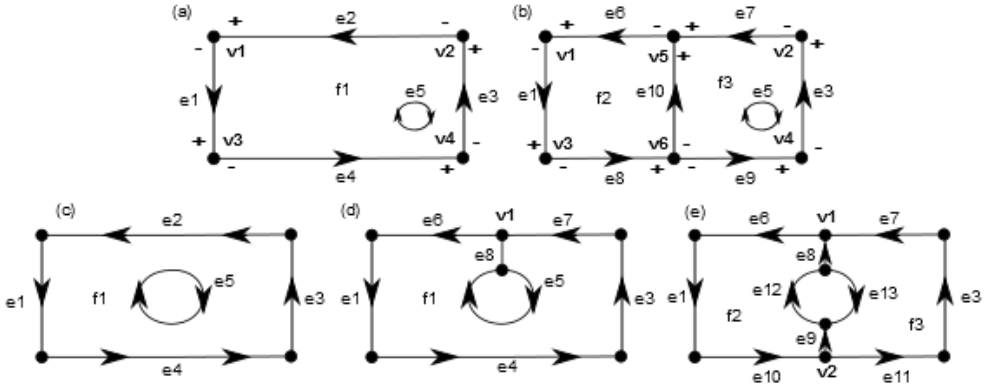

Fig. 5. Face splitting operations: (a) Host face to be split; (b) resulting analysis topology; (c) host face to be split; (d) parasite edge connected to an inner edge; (e) second parasite edge to partition the host face.

Another applicable configuration for face partitioning is shown in Fig. 5 (c-e). In this configuration the subset faces are created by partitioning a host face from its inner loop edge to its outer boundary. The use of a single parasite edge ‘e8’, Fig. 5 (d), does not partition the host face as the face traversal procedure returns to start vertex v1. After introducing a second parasite edge ‘e9’ the face outer boundary is now partitioned as the traversal process ends at vertex ‘v2’. The final step to confirm the partition is to traverse the parasite boundary. The traversal process continues over the inner loop in the positive direction defining a parasite loop of {e9, e12, e8}. This traversal process is possible by maintaining the correct orientations when partitioning the edges with the parasite vertices and is also used to assign the correct orientation to the parasite edges. This returns the boundary of two subset faces ‘f2’ and ‘f3’ to be {e6, e1, e10, e9, e12, e8} and {e7, e3, e11, e9, e13, e8} respectively, Fig. 5 (e). The nature of the traversal procedure enables multiple parasite edges to be used to partition a face. These parasite edges can be added to the face topology in any order with the process able to correctly determine if the host face has been partitioned.

Periodic edges can also be used to partition faces. One virtual subset face will be bounded by the original host outer boundary and the periodic parasite edge, which will be classed as an inner loop edge. The periodic parasite edge will form the outer boundary of the second virtual subset face.

The procedure required to virtually split a volume cell is illustrated using the simple example in Fig. 6, where a lower dimensional face is used as the parasite entity to partition the volume cell. The parasite face can either be a real geometric face, or a virtual face defined by its bounding entities. With either approach the initial step is to use the lower dimensional bounding edges and vertices, highlighted black in Fig. 6 (b), to partition the applicable faces and edges respectively. Once this is achieved, the objective is to determine if the volume cell has been partitioned. The orientation of the bounding edges of a parasite face is established using the orientations of their host. Doing this for the first bounding parasite edge enables the remaining edges to be traversed and assigned the correct orientation.

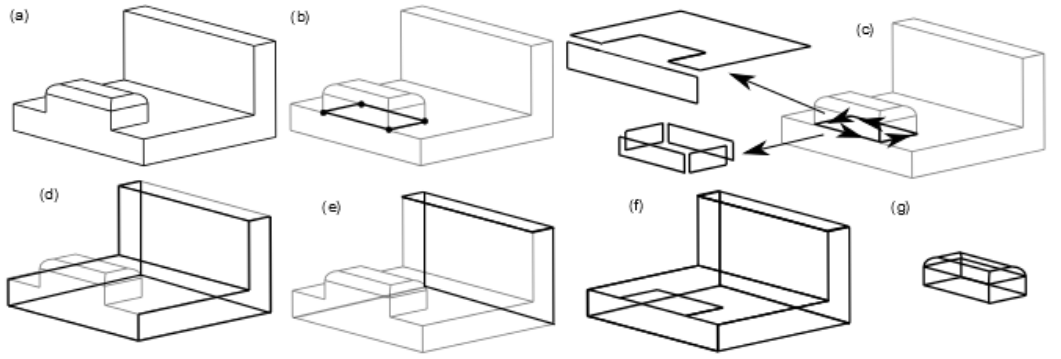

Fig. 6. Volume splitting operation: (a) Host volume to be split; (b) bounding vertices and edges of parasite face; (c) faces on one side of the parasite boundary; (d) first face propagation step; (e) second face propagation step; (f) subset volume; (g) subset volume.

Determining if the volume has been partitioned requires a face traversal process similar to the edge traversal process utilized when partitioning face entities. Here the bounding edges of the parasite faces are traversed in order to determine a face loop on either side of the parasite boundary, Fig. 6 (c). At the first bounding parasite edge both bounded faces are identified, excluding the parasite face itself. Each bounded face is the first face in a distinct parasite face loop. The remaining parasite edges are traversed in the correct order to identify additional loop faces. Loop faces share a common edge with an existing loop face, excluding the bounding parasite edges. The two parasite face loops identified are shown in Fig. 6 (c). One parasite face loop is selected to determine if the volume cell has been partitioned. Using this face loop an edge-face propagation approach is utilized to traverse the faces in the volume. To begin this process the bounding edges (excluding the edges bounding the parasite face) of all faces in the face loop are identified. The edge-face propagation method detects any faces, highlighted black in Fig. 6 (d), that are bounded by these edges. These faces are added to the face loop and the process is repeated with the new set of bounding face loop edges returning the faces highlighted black in Fig. 6 (e), which are appended to the face loop. If the final face loop does not contain any common faces with the second parasite face loop then the volume has been successfully partitioned with the face loop and the parasite face defining its boundary. The subset volume cells are shown in Fig. 6 (f) and (g) where the second subset volume boundary is defined as the residual faces in the host volume. Parasite face orientations are inherited from the bounding parasite edge orientations where the virtual face normal is specified as the normal between the oriented edges.

The virtual decomposition process was described using a single parasite face to partition the volume cell. It is often the case that multiple parasite faces are required to create a subset volume. Fig. 7 shows a decomposition process using multiple parasite faces. Attempting to partition the host volume, Fig. 7 (a), with a single parasite face, Fig. 7 (b), does not generate subset volumes. This is due to the inner loop edges and faces whereby the bounding parasite edges do not partition the top, bottom and inner loop faces and as a result the face loops on either side of the parasite boundary share certain faces and the volume cell remains intact. However, the parasite face is added to the bounding topology of the host volume cell. The addition of a second parasite face, Fig. 7 (c), partitions the top, bottom and inner loop faces and generates two distinct face loops either side of the parasite boundary, partitioning

the volume, subset volume highlighted Fig. 7 (d). Non-manifold conditions are considered to inform the face-propagation process not to include non-manifold edges. This enables the use of multiple parasite faces. Fig. 7 (e) and (f) show the addition of other parasite faces to decompose the subset volumes further and create the o-grid decomposition that can be used to generate a hex mesh around the hole. Thus, virtual operations are able to be used for existing virtual entities. It is worth pointing out that manifold modelling environments cannot manage the non-manifold conditions due to the addition of a single parasite face that does not partition the volume, Fig. 7 (b), resulting in errors when attempting to directly modify the geometry.

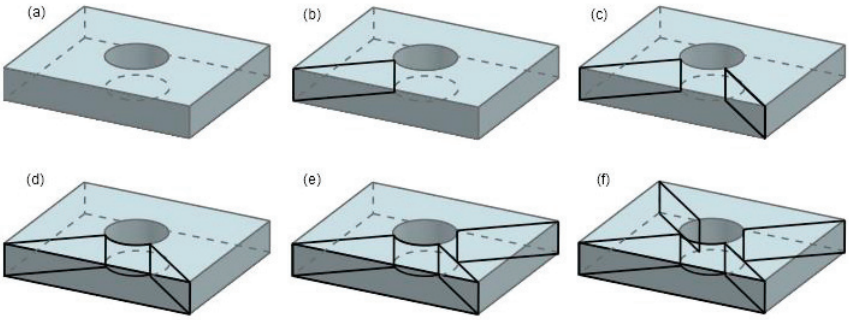

Fig. 7. (a) Host volume; (b) first parasite face; (c) second parasite face; (d) parasite face loop; (e) subset volume cell; (f) o-grid decomposition.

Based on other requirements, such as the extraction of thin regions for hex meshing (section 4.2), it may be necessary to use multiple connected parasite faces to decompose a volume cell, Fig. 8. As the virtual representation does not need to maintain a valid geometric definition it can directly represent non-manifold conditions like the presence of dangling faces or edges, Fig. 8 (b), within their respective parent volumes or faces. This enables multiple parasite faces to be used to decompose a volume cell as shown in Fig. 8. Parasite face loops are constructed by finding the common boundary between adjacent parasite faces, where no free edges exist in the loop, such as in Fig. 8 (d) and (c). Free edges are edges bounding a parasite face but no other face. Their presence informs the process that the volume has not been partitioned. Completing the parasite face loop enables the traversal process to begin to determine if the domain is partitioned, Fig. 8 (d). In this example two subset volumes are created representing a thin region, Fig. 8 (d) right, and a block topology type region, Fig. 8 (d) left.

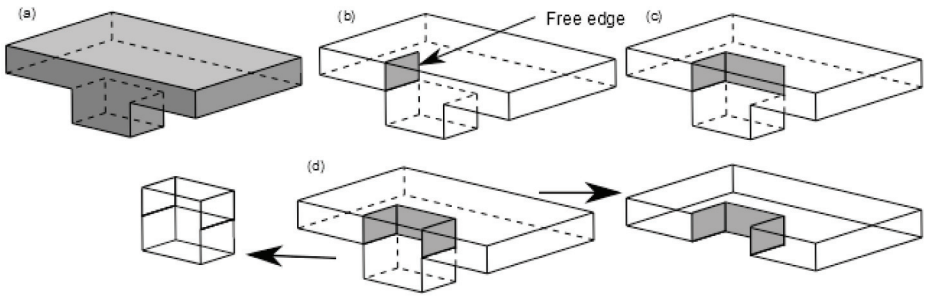

Fig. 8. (a) Host volume; (b) first parasite face; (c) second parasite face; (d) parasite face loop complete and volume decomposed into two subsets.

4. Virtual topology operations for meshing

In this section examples of geometry clean-up and decomposition are provided, using the virtual topology operations in the previous section. The work detailed in this section has been implemented as a C# plug-in for Siemens NX, where a generic data structure in the form of a relational database is used to store the model topology and associated information as described in [17]. Having a robust link to the NX model topology enables the NX manifold environment to exploit non-manifold conditions within the virtual representation. Extracting the initial model topology into this generic data structure enables it to be interrogated and manipulated to execute virtual

topology operations. The topological connectivity and relative orientations are extracted from a CAD model automatically along with all vertex and edge classifications which is used for geometry clean-up and to help identify if a specific meshing strategy can be assigned to a volume cell. Vertex classifications define the angle between the bounded edges meeting at each vertex bounding a face while edge classifications define the edge dihedral angle.

4.1. Geometry clean-up example

Geometry clean-up operations attempt to remove features that can cause problems during mesh generation and cause poor quality elements to be formed. Clean-up decisions are based on geometric reasoning criteria (edge dihedral angle, edge length, face width, curvature etc.) adhering to predetermined mesh metrics (target element size, element angles etc.). Here a simple automated clean-up operation based upon edge dihedral angles is shown.

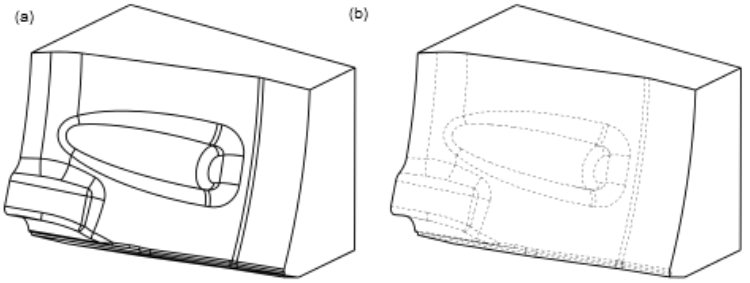

Fig. 9. (a) Original design topology; (b) analysis topology after ignoring edges with a specific dihedral angle.

Edges to ignore are selected if they have a dihedral angle within 10% of 180 degrees, meaning that nodes do not need to be place along these edges. For the model in Fig. 9 the selection returns 154 edges to be ignored from the original model topology, Fig. 9 (a), which are then provided as the input to the virtual merge operation described in Section 3.1. For the model in Fig. 9 the automated operation is executed in 5 seconds (on 64-Bit 2.8GHz Intel Core i7 CPU with 16GB RAM) and merges 59 faces and 28 edges to form 6 superset faces and 12 superset edges, Fig. 9 (b), where the ignored edges are faded out. The purpose of this example is to demonstrate the robustness of the virtual merge operations using a set of ignored edges as input. The generic nature of the procedure enables the input to be provided from any source, i.e. geometric reasoning operations built into an existing CAE package.

4.2. Thin-sheet region isolation through virtual decomposition

One approach for decomposing a complex domain for hex meshing is to isolate thin-sheet regions (large lateral dimensions relative to thickness) as separate volume cells [18, 19]. The hex meshing problem is simplified in thin-sheet regions where a quad mesh generated on a top or bottom face is swept through the thickness direction to create Hex elements. The thick-thin and virtual topology tools have been developed within Siemens NX. The thick-thin algorithm [19] uses the NX mid-surface tool to identify face-pairs, which are the opposing faces bounding a thin-sheet region. Face boundaries are discretized and to calculate intersection regions between the boundaries to define thin-sheet regions. Parasite faces are generated from the opposing boundaries and are used to partition the domain.

A simple example of the thin-sheet identification and virtual decomposition process is shown in Fig. 10. The edges on opposite sides of the thin-sheet region are defined as bounding edges of the parasite and provided by the thin-sheet identification tool as the input to the virtual decomposition, Fig. 10 (b). These parasite bounding edges are provided along with their host face in the original topology. Virtual topology relationships are used to identify the new virtual host from the original input host. As the decomposition progresses this allows the appropriate faces to be partitioned using the parasite bounding edges and the analysis topology to be maintained. The wall edges through the thickness of the thin-sheet region are automatically generated within the virtual decomposition tool, where their host entity is identified and partitioned if necessary, Fig. 10 (c). Wall edge hosts are defined as the common bounded face of the end vertices of the input bounding parasite edges. Once all lower host topologies are partitioned the volume cell is virtually decomposed, Fig. 10 (d), into a thin-sheet and residual region.

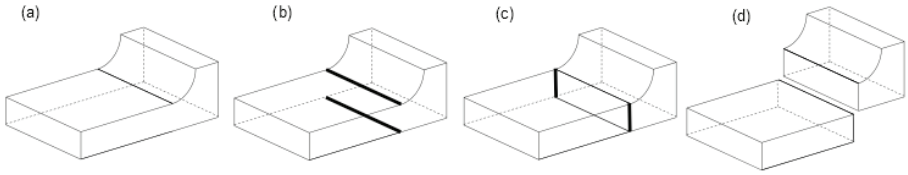

Fig. 10. (a) Original topology; (b) parasite face bounding edges highlighted; (c) wall edges highlighted; (d) subset volume cells.

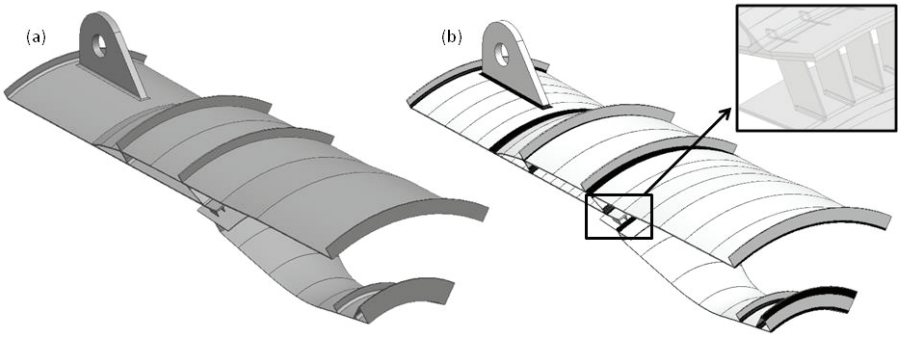

Fig. 11. (a) Original casing component; (b) thick-thin virtual decomposition.

Fig. 11 shows the thin-sheet virtual decomposition of an aero-casing component. 88% of the component is recognized as thin-sheet and highlighted grey, the residual regions highlighted black and virtual partitioning faces inset, Fig. 11 (b). The virtually decomposed representation is completely linked to the original component model and can be viewed and interacted with in the NX environment. Virtual edge geometry is created for subset, superset and parasite edges and for visualization purposes. Surface geometry is not required for any virtual faces.

4.3. Identification of meshing strategies in the presence of virtual topology

The analysis topology description enables the use of simple interrogations to determine if either a mapped, swept or mid-point subdivision meshing approach can be applied to a volume cell. The interrogation is automatically executed using SQL queries on the analysis topology, which is associated with the original topology in tables in a relational database. Topological queries are considered alongside criteria such as edge dihedral angles and vertex corner angles when identifying the different regions.

A mapped meshing scheme can be utilized for volumes with a logical block topology consisting of 6 faces, 12 edges and 8 corner vertices. Criteria for a volume cell to be map-able dictates all edges must be convex and all faces must be map-able. A face is considered map-able if it consists of four logical edges and four logical corner vertices where a single hex element will appear in the mesh pattern at each vertex and edge segment.. Determining whether a volume cell is map-able is straightforward since the edge dihedral and vertex corner angles are maintained throughout the virtual operations described. A structured hex mesh generated for map-able volume cells can use opposing edges information to assign division numbers.

Volume cells that cannot be map meshed may be applicable for mid-point subdivision meshing if the cell is convex and contains all tri-valent vertices, Fig. 12 (a). Unlike mapping methods this scheme requires the cell to be partitioned into n map-able blocks, where n is the number of vertices in the volume cell, Fig. 12 (c). This is an automatic procedure where the volume is decomposed using a series of four-sided faces bounded by edges generated from edge mid-points to face mid-points (black edges in Fig. 12 (b)) and from face-mid-points to volume center (dashed edges Fig.. 12 (b). In this work this step is carried out via the virtual decomposition. This decomposition contains mesh line singularities where an internal edge valence is not referenced by four hex elements and represents an unstructured portion of the mesh. Placement of these singularities is the root of the auto-Hex meshing problem and is used to help define Hex mesh topology and sizing amongst other properties. Using mid-point subdivision the line singularities are generated between the volume center and the mid-point of non-map-able faces.

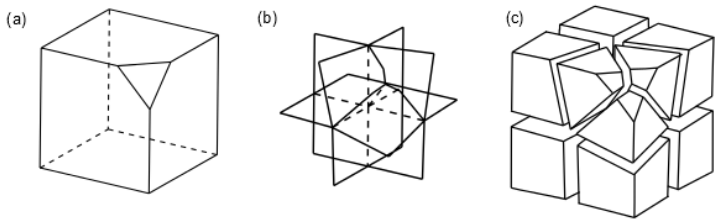

Fig. 12 (a) Volume cell being queried; (b) parasite faces; (c) block decomposition.

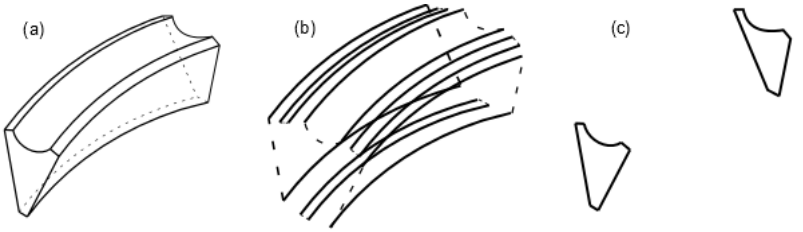

Fig. 13 (a) Volume cell being queried; (b) connected map-able faces; (c) source and target face.

For certain classes of geometry swept meshing schemes simplify the 3D Hex meshing problem into the 2D domain. Sweep-able volume cells are bounded by a loop of map-able wall faces bounded by a source and target face, Fig. 13. The detection of sweep-able volume begins by classifying the map-able faces bounding the volume, Fig. 13 (b). These map-able faces are grouped into loops of connected map-able faces through their common edges, highlighted black in Fig. 13 (b). For a closed loop of map-able faces each map-able face is bounded by two other map-able faces. The uncommon edges, dashed in Fig. 13 (b) between the closed loop of map-able faces are used to identify the source and target faces, which are each bounded by a distinct loop of uncommon edges, Fig. 13 (c), ensuring both source and target faces are topologically identical. Hex meshes can be generated on swept volume by meshing the source face with quadrilateral elements and sweeping towards the target face along the map-able wall faces. The line singularities in sweep-able domains run along the sweep direction.

The subset volume cells from the virtual thin-sheet decomposition from Fig. 11 were automatically interrogated to identify an appropriate meshing scheme. The results are shown in Fig. 14 with 77% of the original volume map-able and 21% sweep-able.The remaining 2% of the volume cannot be swept, mapped or meshed using mid-point subdivision and would require further manual decomposition to generate meshable volumes.

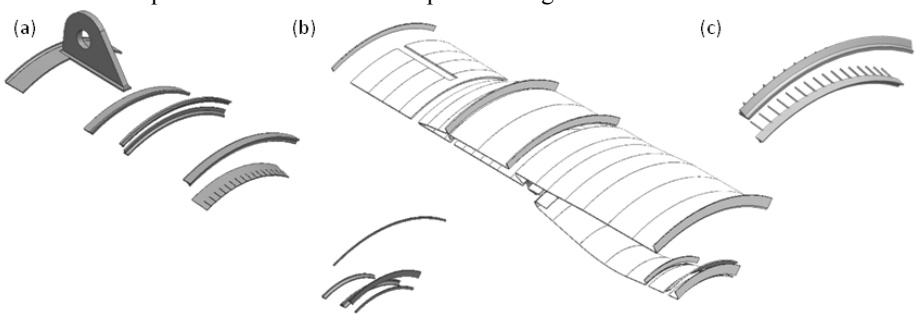

Fig. 14. Identification of meshing strategies: (a) Sweep-able; (b) block topology; (c) no meshing strategy.

5. Discussion

This work has shown how the virtual topology model can be linked to but stored separately from the original topology and is referred to as the analysis topology. Maintaining the virtual representation in this manner enables multiple analysis topologies to reference a single base topology. Creation of the analysis topology enables a non-manifold representation to be created without the constraint of needing a non-manifold modeling package, which

maintains the links between adjacent sub-cells within the virtual decomposition. For example, the implementation within Siemens NX described here makes use of the non-manifold common faces to assign mesh mating conditions.

Section 4 demonstrates the virtual topology operations for realistic geometric models, where virtual merge operations are used for geometry clean-up and virtual splits are used for thin-sheet removal. A major benefit is that the virtual representation is completely linked to the original model topology. Therefore, virtual operations can be updated after small parametric design changes. This is in contrast to situations where real geometric modifications have been made, where the links to the original model are lost making it difficult to update after even the smallest modifications. The virtual merge operations are described for single components but need extended to deal with multi-component assemblies. The difficulty is to determine the overlap and gaps between components to create exact interfaces between within the specific analysis context. This might be achieved by generating a tolerant virtual representation to treat gaps and overlaps. Further work has to be carried out to fully exploit the virtual topology within current toolsets to enable mesh generation algorithms to operate in the presence of virtual topology [20]. For example, using the virtual merge operations to merge certain faces, i.e. with the same underlying surface definition that are bounded by edges with a dihedral angle close to 180 degrees, prior to implementing the thin-sheet removal tool would reduce the complexity of the identification and partitioning process. The same proportion of the model would remain recognized as thin-sheet regions due to the upfront merging operations essentially treating adjacent thin-sheet regions as one and reducing the number of partitions required.

The thin-sheet virtual decomposition was automatically interrogated to determine the meshing strategy of cells, where an extra 10% of the model was assigned a meshing scheme. Over 85% of the thin-sheet regions were identified as map-able and 15% as sweep-able. Since the mesh scheme identification algorithms consider only topological characteristics of a volume along with vertex and edge classifications there will be instances where geometric reasoning needs to be implemented to take into account properties such as varying edge dihedral angles or surface curvature. However, in this work it is assumed that since the model has been initially interrogated during the thin-sheet identification process, the thin-sheet regions and their adjacent regions can accurately be assessed using the topological approach undertaken. Manual virtual decomposition methods can be utilized for the residual regions assigned no meshing strategy. The virtual topology tool will accept two bounding edges of a parasite face as the manual input and will automate the partitioning of all lower bounding topologies and the addition of the remaining edges bounding the parasite face, as described in Fig. 10. This releases the analyst from substantial effort required to generate partitioning geometry. In addition the link between the virtual decomposition to the underlying geometry is automatically maintained unlike manual blocking tools such as ICEM CFD where associations between the block topology entities to the original model entities needs manually specified as the blocking is overlaid on the geometry.

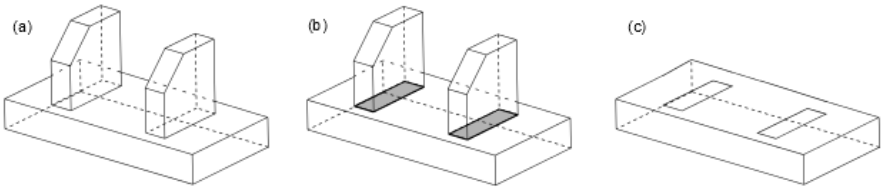

Fig. 15. (a) Original topology; (b) parasite faces highlighted; (c) block topology.

Regions may also be considered as map-able where a logical block face may be represented by several faces. Such a scenario can be seen in the subset volume in Fig. 15 (c) which can be treated as a map-able region since there are no mesh singularities. This removes the need to propagate the block pattern through the virtual decomposition and reduces the burden on the analyst. The subset volume is essentially a sweepable block where the block faces with the internal imprints is meshed first. Therefore, these imprinted regions may exist on source and/or target faces of sweep-able regions. These ideas need extended to treat 1-n and n-n sweeps.

6. Conclusion

This paper has described a generic topological approach for executing virtual topology operations. Merge and split operations have been described in detail for all topological entities including the partitioning of volume cells.

These procedures maintain a valid topological description within the virtual. This topological consistency enables boundary traversals to be used to virtually decompose the domain of interest where virtual operations are executed successfully on both original and virtual entities. A robust hierarchical virtual structure enables the analyst to prepare the analysis model in any desired fashion. Virtual merge and split operations have been demonstrated on realistic geometric components and can be utilized in either manual or automated workflows where the partitioning instructions are delivered from an upstream tool, such as the thin-sheet identification tool.

The analysis topology description is managed alongside the original model topology facilitating complete associativity between the representations and connects the generic virtual representation to existing CAD packages. This enables model interrogations and modifications to be carried out on the analysis topology without affecting the original topology. To this end a process has been described where volume cells are automatically interrogated to determine whether a specific meshing scheme can be assigned to it. Substantial work needs to be done to develop more tools to operate in the presence of virtual topology, especially meshing algorithms for meshing virtual cells.

Acknowledgements

The authors wish to acknowledge the financial support provided by Innovate UK: the work reported herein was funded via GHandI (TSB 101372), a UK Centre for Aerodynamics project. Figs 11 and 14 created using a Rolls-Royce model provided through the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 234344 (www.crescendo-fp7.eu).

References

[1] Cottrell, J.; Hughes, T.; Bazilevs, Y.: Isogeometric analysis: Toward integration of CAD and FEA, Wiley, 2009

[2] Sheffer, A.; Blacker, T.; Clements, J.; Bercovier, M.: Virtual Topology Operators for Meshing, in 6th International Meshing Roundtable, Sandia National Laboratories, 1997. http://dx.doi.org/10.1142/S0218195900000188

[3] Nolan, D.C.; Tierney, C.M.; Armstrong, C.G.; Robinson, T.T.: Defining simulation intent, Computer-aided design, 2015, 59, 50-63. http://dx.doi.org/10.1016/j.cad.2014.08.030

[4] Foucault, G.; Cuilliere, J-C; Francois, V.; Leon, J-C; Maranzana, R: Adaption of CAD model topology for finite element analysis, Computer-Aided Design, 40(2), 2008, 176-196.

[5] ABAQUS CAE, http://www.3ds.com/products-services/simulia/products/abaqus/abaquscae/, Dassault Systemes.

[6] Ansys Workbench, http://www.ansys.com/Products/ANSYS+15.0+Release+Highlights/ANSYS+Workbench.

[7] CUBIT, Sandia National Laboratories, https://cubit.sandia.gov/.

[8] ICEM CFD, Ansys, http://www.ansys.com/Products/Other+Products/ANSYS+ICEM+CFD.

[9] NX, Siemens, http://www.plm.automation.siemens.com/en_gb/products/nx/.

[10] Tierney, C.M.; Nolan, D.C.; Robinson, T.T.; Armstrong, C.G.: Using mesh-geometry relationships to transfer analysis models between CAE tools, Engineering with Computers, 2014. http://dx.doi.org/10.1007/s00366-014-0377-7

[11] Mukherjee, N.: An art gallery approach to submap meshing, Procedia Engineering, 82, 2014, 313-324.

[12] Sheffer, A.; Etzion, M.; Clements, J.; Bercovier, M.: Hexahedral mesh generation using the embedded voronoi graph, Engineering with computers, 15(3), 1999, 248-262.

[13] GAMBIT, Fluent, http://www.arl.hpc.mil/software/description.html?sw=Gambit.

[14] Sheffer, A.; Blacker, T.; Bercovier, M.: Towards the solution of fundamental issues in CAD-FEM integration, 4th World Congress on Computational Mechanics, 1998.

[15] White, D.R.; Mingwu, L.; Benzlet, S.E.; Sjaardema, G.D.: Automated hexahedral mesh generation by virtual decomposition, 4th International Meshing Roundtable, Sandia National laboratories, 1995, 165-176.

[16] Chong, C.S.; Kumar, A.S.; Lee, K.H.: Automatic solid decomposition and reduction for non-manifold geometric model generation, Computer-Aided Design, 36, 2004, 1357-1369.

[17] Tierney, C.M.; Nolan, D.C.; Robinson, T.T.; Armstrong, C.G.: Managing equivalent representations of design and analysis models, Computer-Aided Design and Applications, 2014, 11(2), 193-205. http://dx.doi.org/10.1080/16864360.2014.846091

[18] Robinson, T. T.; Armstrong, C. G.; Fairey, R.: Automated mixed dimensional modelling from 2d and 3d cad models, Finite Elements in Analysis and Design, 47(2), 2011, 151-165. http://dx.doi.org/10.1016/j.finel.2010.08.010

[19] Sun, L.; Armstrong, C. G.; Robinson, T. T.; Tierney, C. M.: "Automatic Thick Thin Decomposition of Complex Body for Hex Dominant Meshing," Paper presented at 4th Aircraft Structural Design Conference, 2014.

[20] Foucault, G.; Cuilliere, J-C; Francois, V.; Leon, J-C; Maranzana, R: Generalizing the advancing front method to composite surfaces in the context of meshing constraints topology, Computer-Aided Design, 45(11), 2013, 1408-1425.