INCREMENTAL DECOMPOSITION FOR HEX-MESHING IN CAD USING VIRTUAL TOPOLOGY

Benoit Lecallard[1], Trevor T. Robinson[1], Cecil G. Armstrong[1], Declan C. Nolan[1], Harsha Ramesh[2]

[1]*Queen's University Belfast, Belfast, United Kingdom. b.lecallard@qub.ac.uk*
[2]*Rolls-Royce plc, Derby, United Kingdom Harsha.Ramesh@rolls-royce.com*

ABSTRACT

This paper presents methods for preparing a geometry model for finite element mesh generation in a Mechanical Computer-Aided Design (MCAD) environment. It works by creating a new representation of the model through the application of virtual topology operators. The resulting "analysis topology" description is used to abstract the analysis model, enabling automated tools and experts to apply an incremental strategy to decompose the model for meshing, without modifying the original CAD model. This work also demonstrated how virtual topology enables the integration of multiple model decomposition tools to expand the capabilities of the hosting CAD environment, providing support for more meshing strategies and more freedom in how they are applied, while bridging the gap between the CAD and analysis models. Herein, the virtual topology operators used to decompose the model are checked and propagated based on the required mesh constraints to ensure the resulting mesh is conformal at the interfaces. Finally, the methods required to decompose the original CAD model using the analysis topology description and "virtual geometry curves" are presented, enabling downstream automation of the mesh.

Keywords: mesh generation, analysis topology, virtual topology, CAD

1. INTRODUCTION AND RELATED WORK

Generating a good quality mesh is a major bottleneck in most finite element analysis workflows. The generation of high-quality hexahedral (Hex) element meshes remains a highly skilled and user intensive task, which often requires the use of dedicated CAE packages into which the original CAD geometry needs to be transferred from the CAD environment. Hex elements are preferred over alternatives (e.g. tetrahedral elements) when simulating highly non-linear events, using explicit analysis codes and for accurate contact capture between deformable bodies. A comprehensive survey by Sarrate et al. [1] highlights a wide range of approaches to hex meshing, as well as the benefits of using this element type. Decomposition-based approaches are widely used and involve partitioning the geometry of the model to be meshed into sub-regions with specific topological and shape characteristics which can be meshed using hex-meshing algorithms like mapping and sweeping.

With decomposition and meshing accounting for more than 50% of the time taken for the entire simulation task [2], automating aspects of the hex meshing task is a well-researched ambition. Whilst the push toward fully automated hex-mesh generation for arbitrary domains has yet to yield a generic solution, it has resulted in many automated tools that are applicable to specific classes of geometry. These tools use either divide and conquer paradigms to recursively extract simple regions [3]–[6], or use intermediate constructs to capture the flow of elements and identify partitions [7]–[10]. An extension of this is to apply the same divide and conquer paradigms in an integrated incremental decomposition workflow, where simpler tools alleviate the task of the more complex and computationally expensive ones. While analysts would greatly benefit from combining existing tools, their integration is challenging as standards for geometry exchange are not tailored for analysis models. Dedicated meshing packages such as CUBIT[11] already integrate various automated methods, but still resort to the judgment of the user to select the best partitioning strategy. These packages are also limited by the need to transfer the geometry from a CAD environment, and the difficulty to add additional decomposition methods. The shortcomings of fully automatic tools are also recognized in [12], where the benefits of semi-automated decomposition workflows are demonstrated using a manual sketch-based decomposition method enhanced by geometric reasoning [13].

When a model is decomposed into sub-regions, a conformal mesh is required at the interfaces to successfully connect their respective meshes. The constraints of conformal hex meshing and structured mesh implications are reviewed by Blacker [14]. Previous work on generating conformal meshes using sweeping is described in [15] and [16]. Even though both are mesh-based methods tailored for one type of decomposition, they highlight the importance of interface management for conformal meshing.

The benefit of using virtual topology for pre-processing a model for meshing has been presented by Sheffer et al. [17]. The concept involves creating virtual topology entities by applying virtual topology operators to the entities in the original CAD model, which are therefore based upon but do not alter the underlying CAD definition. To date it has mostly been used for correcting minor "defects" (e.g. to merge a sliver face with a larger adjacent face), with implementations focused on the final steps of the analysis model preparation process. Extending the use of virtual topology to the entire pre-processing stages would facilitate the integration of different automated tools, as the need to exchange geometry (e.g. decomposed CAD) and/or pre-processing operations (e.g. split operation) is replaced by the need to exchange virtual topology operations. White [4] used virtual decomposition to automate hex mesh generation, where surface nodes of an initial mesh are reassigned to a virtual sub-region.

More recently, Tierney et al. used virtual topology operators to generate an "analysis topology" based on the outputs of a decomposition algorithm [18]. The concept of analysis topology enables to streamline pre-processing tasks, by adding flexibility to the decomposition while exposing all the necessary information to manage interfaces and automate decomposition and meshing. However, the implementation in that work was limited by the need to edit automated tools to work using virtual topology, and the a-posteriori identification of meshing strategies preventing further decomposition in the absence of a mechanism to maintain a conformal mesh at interfaces. Finally, generating a mesh from a virtually decomposed model requires either new meshing tools or robust geometrical decomposition capabilities for compatibility with existing meshing tools.

This work builds on the analysis topology concept to enable incremental decomposition of CAD models for automatic hex meshing. The main contributions include introducing a method to integrate virtual topology with both existing tools and manual operations and a method to manage and exploit meshing strategies to propagate splits automatically. Finally, a method to ensure that the virtual topology decomposition can be applied geometrically for compatibility with downstream meshing is presented.

2. INCREMENTAL DECOMPOSITION FOR MESHING

Incrementally decomposing a model for meshing involves identifying and extracting individual regions of the geometry to which known meshing algorithms can be applied. Once a meshing strategy has been identified for each sub-region of the domain, each can be meshed in a piecewise manner.

2.1 Structured meshing requirements

The quality of a hex mesh is directly related to the geometry of its elements and their connectivity. In a regular mesh each interior node should connect exactly 4 quad elements or 8 hex elements. To accommodate complex shapes while retaining the quality of individual elements, nodes must sometimes connect an irregular number of elements, which introduces "singularities" into the structure of the mesh.

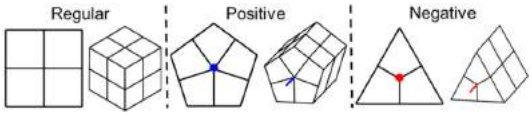

Figure 1. Mesh singularities.

Definition: A **mesh singularity** is a collection of one or more irregular nodes**.** It can be either positive (more than the regular number of connected elements), or negative (less than the regular number of connected elements), as shown in Figure 1.

Quad (2D) and hex (3D) meshing algorithms impose strict requirements on the presence of singularities, which in turn impose constraints on the shapes that can be processed, as the number of singularities is directly linked to the shape. These requirements are as follows:

- **Mapping** (quad): The mesh is generated by mapping the template of a unit square onto a local surface parametrization [19]. As such, no singularities will occur and the face must be 4-sided. It also implies that opposite pairs of edges need to have the same number of divisions. A sub-mapping variant is also possible for non-rectangular faces where all the edges can be grouped into two sets of opposite groups.
- **Paving** (quad): The mesh is generated by inserting rows of quad elements from the boundaries towards the interior [20]. There are no specific requirements on the structure of the mesh and singularities can be present. This means there is no constraint on the shape of the face. However, the algorithm may introduce pairs of singularities that cancel each other. It also requires that the sum of division numbers on each loop of edges must be even.
- **Mapping** (Hex): The mesh is generated by mapping the template of a unit cube on the local i-j-k parametrization. This requires the shape to have a cube-like topology, with 6 logical faces and 12 logical edges, and no singularities can be present. It implies that all bounding faces are mapped meshed, with the associated constraints on singularities and edges divisions.
- **Sweeping** (hex): Hex elements are generated by sweeping quad elements on a **source face** to a **target face**. This means all lateral faces (so-called **wall faces**) connecting the source and the target have mapped mesh structures, and hence no singularities can exist on wall faces. Also, corresponding edges at opposite ends of the sweep must have the same number of divisions. There is no mesh structure requirement on the source face, which can be either

mapped or paved, and therefore singularities can be channeled from the source to target face.

This work focuses on two types of shapes suitable for hex meshing with these algorithms:

- **Block** shapes, with a cube-like topology that can be map (Hex) meshed with 6 faces mappable (quad).
- **Sweepable** shapes, with a loop of mappable (quad) wall faces in the sweep direction.

Directly identifying a block decomposition for an arbitrary geometry is difficult, as there should be no singularities in the blocks. This means all singularities need to be located at the edges bounding the interfaces between blocks. Sweepable regions are less constraining as they can accommodate singularities along the sweep direction and are therefore easier to identify. There is a strong correlation between the two types, as a block can be swept meshed in any of three directions, and sweepable regions can have a mappable source face and therefore satisfy block constraints. It is therefore easier to identify first a semi-structured mesh by identifying sweepable regions, and then decomposing their source faces to constrain the singularities and achieve a more structured block decomposition.

2.2 Reasoners

Manually identifying and extracting block and sweepable regions can be a very tedious task for geometries which include many details. Various automated tools or reasoners have been developed to facilitate this task by extracting regions based on specific geometric and topological characteristics. These characteristics define in turn a meshing strategy which specifies how the regions should be meshed. This information is required as the type of the shape (block or sweepable) does not contain sizing information and can change.

Definition: A **decomposition reasoner** refers to an algorithm that queries the model to identify regions that can be assigned a specific hex-meshing strategy and provides the topological and geometrical information to create the partitioning entities necessary to extract such regions.

Definition: A **meshing strategy** describes the type of element (e.g. Hex or Mixed-Tet) along with sizing information, symmetries and anisotropic element shape metric properties of the region.

The simplest reasoners are tools that identify regions that are already blocks or are sweepable, by checking that the topology and geometry match the requirements of that region type (described previously). Other reasoners use shape properties such as concavities and symmetries to help breaking down a model into simpler regions. For example, aero-engine models are mostly axisymmetric with cyclic patterns that repeat around the circumference. Using a dedicated reasoner based on [21], axisymmetric regions and regions that can be meshed using cyclic symmetries can be identified. The associated meshing strategy stores any repetition pattern, to ensure a compatible mesh between each occurrence. Other reasoners exploit local anisotropy of the shapes to identify sweepable regions. For example, thin-walled regions with two large dimensions compared to the third can be meshed by applying a mesh to a larger face and sweeping through the small thickness. A thin-sheet reasoner based on Sun's implementation [22] identifies and extracts thin regions by manipulating pairs of opposing faces from the CAD geometry. The associated meshing strategy stores the aspect ratio of the shape and the thickness, which can then be used to infer a target element size as described in [23]. Similarly, truss-like structures, or models which have had their thin-sheet regions removed, can have many long regions with a nearly constant cross-section topology, that are also appropriate for hex-meshing by sweeping. These can be identified by a long-slender reasoner that processes loops of nearly parallel long edges, as described by Sun [5]. These reasoners can greatly reduce the number of DOFs of the mesh, as the anisotropy of the region can be used to stretch the hex elements and reduce their number. More complex decomposition reasoners can also make use of other types of information, such as temporary constructs (frame-fields, medial-object), functional and adjacency information if available, or AI methods.

Each reasoner has its strengths and weaknesses in terms of speed, accuracy and class of shapes supported. More than one may be required to achieve a full hex mesh for a complex shape. Therefore, an efficient incremental decomposition workflow requires the integration of a diverse range of decomposition reasoners. To be of maximum benefit these need to work in any order, without any dependencies on the preceding reasoners or the package where the CAD model is hosted. Preparing a CAD model for meshing can also include de-featuring and dimensional reduction operations, which can be identified and applied using dedicated automated reasoners which are not covered in this paper.

2.3 Challenges

Since many meshing workflows start from a geometry that has been created in a feature-based CAD environment, and to maintain the associativity with the design history in the model, the ability to decompose the model for meshing within the CAD system is an attractive solution. However, there are several challenges to doing so, primarily because CAD packages have not been developed for the purposes of decomposing a model for meshing.

First, creating a split operation in CAD may create unexpected geometrical defects such as sliver faces and result in non-watertight models due to trimming errors [24]. Secondly, automating the decomposition and downstream meshing requires a robust tracking of B-Rep entities, which is challenging due to persistent naming issues inherent to CAD packages [25]. Then, incrementally decomposing the CAD model will append a sequence of split operations to the feature tree of the model, and any edit further up in the tree may produce unexpected results further down, including the splits. Finally, most commercial CAD environments rely on a manifold boundary representation scheme, meaning that two bodies cannot share a same face, edge or vertex. Hence, two identical faces are created within the CAD system at the interface between two bodies after a split operation.

Even when a CAD system is used to help prepare a geometry model for meshing, a transfer to a dedicated CAE package is usually still required for meshing. After doing so the

decomposition will be converted to a non-manifold representation which ensures the resulting mesh is conformal at interfaces between regions. It is therefore important to ensure that incremental decomposition will produce a usable collection of bodies that can be re-assembled in a CAE package for meshing.

Another challenge comes from the incremental decomposition principle itself. Identifying simple regions first means all of the complexity of the meshing task will be pushed to the last regions of the geometry to be processed. This can become problematic as these regions may harbor complex arrangements of singularities. Where these exit through an interface, they make any hex meshing strategy in connected regions invalid. Therefore, special care must be taken when chaining reasoners, as structure modification can propagate throughout the decomposition.

2.4 Proposed workflow

Since most of the challenges of incrementally decomposing a model in CAD come from the application of the successive split operations, the idea in this work is to identify regions to which a known meshing strategy can be applied, store the required partitioning strategy, and then query the partitioning strategy to identify the next regions to process. This is enabled by virtual topology split operators that will topologically partition the model without altering the CAD representation, as described in the next section. Each region in the model for which a meshing strategy has not yet been identified is classed a "residual region". Eventually, once all the reasoning is done and a suitable virtual topology decomposition is available, the model can be decomposed within the CAD system to be used for meshing. Should any residual regions remain at the end of the process a tet-mesh can be applied to them, with a layer of pyramid elements at interfaces with hex-meshed regions, to produce a mixed mesh. To be successful, this workflow requires a simple way of integrating existing reasoners with virtual topology, so they can identify suitable regions in presence of virtual topology and define virtual topology splits. The meshing strategies identified by the reasoners need to be robustly managed to remain valid after further decomposition of neighbor regions. Finally, the ability to robustly convert a virtual decomposition into a CAD decomposition is required to ensure the virtual decomposition is usable.

3. ANALYSIS TOPOLOGY

3.1 Virtual topology

Virtual topology uncouples the topological representation of a model from its geometrical representation in the B-Rep scheme [17], allowing manipulation of the topology without having to alter the underlying geometry of the model. It defines a set of entities and operators to carry out the operations associated with model pre-processing for meshing, and to formalize the relationships with the original host model.

Virtual topology entities do not require an explicit geometric definition and instead use a geometric definition inferred from their host entities, or which can be related to simple geometrical constructs (e.g. line between two points, least-square fitted surface, etc.). These are illustrated in Figure 2 (a), and include:

- **Parasite entities**: entities that do not exist in the topology of the original CAD model, but lie on an entity from the original CAD model of higher dimension (e.g., an edge lying on the face it splits).
- **Subset entities**: subsets of host entities that are split by a topological entity of lower dimension (e.g., faces obtained by partitioning a host face with a parasite edge).
- **Superset entities**: a superset of host entities that are merged together by ignoring their common boundary entities.
- **Orphan entities**: an entity without a host one dimension higher, and from which no geometry description can be inherited (i.e., an edge through volume).

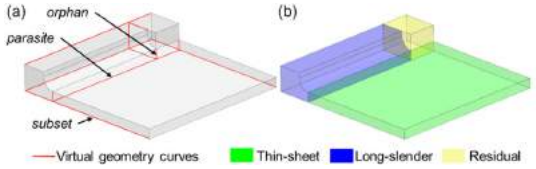

Figure 2. (a) virtual topology entities created after virtual decomposition and (b) equivalent geometric decomposition and meshing strategies.

Virtual topology operators relevant for an incremental decomposition workflow are the virtual topology split, where a host entity is split into several subsets by parasite entities, and the virtual topology merge where several entities of the same dimension are merged into a single superset by ignoring their common boundary entities.

3.2 Abstracting the analysis model

Definition: The **analysis model** is a transformed version of the design model that exists within a CAE environment, to which mesh, boundary conditions and loads are applied.

Implementing the decomposition in the CAD system using virtual topology operators means that only a topological description of the analysis model is created, known as the analysis topology.

Definition: The **analysis topology** is a representation of the boundary topology of the analysis model.

In this work, the analysis topology is a non-manifold cellular model, which means that all interfaces between cells are known and are considered cells in their own right. Meshing strategies can be attached to cells. The analysis topology is initialized by extracting the topological representation of the B-Rep from the original design model. It is external to any CAD package and can represent topological relationships not supported in many CAD environments, but which are required for conformal meshing. It is therefore capable of acting as the interface between different CAD and CAE

packages. However, while the analysis topology can be used to represent the topology of the model to be meshed, it does not contain sufficient information to be used for reasoning. To address this issue, "virtual geometry" is introduced.

Definition: virtual geometry entities are geometric representations of virtual entities that co-exist in the modelling space of the design model, but are not associated with its B-Rep.

Virtual geometry entities are used to perform geometric tests on the analysis topology and to visualize the virtual volume cells. Virtual geometry curves (in red in Figure 2 (a)) are combined with the existing edges of the CAD model that have not been virtually edited to define a wireframe representation of the volume cells. These curves help store the partitioning intent of decomposition reasoners and avoid deleting and re-creating curves. Whenever the actual CAD decomposition is required, virtual geometry curves are used to define virtual geometry surfaces that can partition the CAD model to generate the equivalent analysis model, Figure 2 (b).

3.3 Reasoning on the analysis topology

The use of an analysis topology implies that the current decomposition state of a model is not explicitly available and cannot be directly queried or decomposed. Additional steps are required to adapt the decomposition reasoners, which depend on the ability to integrate a reasoner. Figure 3 shows the integration of 6 different types of reasoners to interact with the analysis topology

3.3.1 Queries

Reasoners that are fully integrated with virtual topology can directly query the analysis topology. Geometrical queries are achieved by inheriting the geometric definition of host entities or by querying virtual geometry curves if no geometry is linked. Reasoners for extracting thin-sheets, long-slender and axisymmetric regions have been fully integrated with virtual topology, as described in [26]. Other reasoners that are not integrated with virtual topology require an explicit geometry description to work with, as modifying their implementation to work with virtual topology might be tedious, or not even possible. In that case, there is no need to commit the entire decomposition, only the subset regions of interest can be temporarily extracted from the CAD model, as detailed in section 5.1. The temporary region can then be processed in either another CAD session of the native CAD environment, or a different CAD environment after STEP export.

In the situation where a user wants to manually insert partitions by applying CAD split operations, an explicit geometry is also extracted. It is then enriched with interface and mesh singularity information from neighbors, to help the user understand the flow of elements and constraints stemming from the meshing strategies of neighbor regions. (see Figure 13 (c)).

3.3.2 Parasite wireframe

In the absence of any standard for exchanging virtual topology partitions (though one could easily be defined), the concept of a parasite wireframe is introduced to integrate the output of different reasoners, or manual intervention, around a common format. The purpose is to collect the minimal information required for applying virtual topology split operators that cannot be recovered by reasoning, to accompany the transfer of the geometry as a STEP file.

Definition: A **parasite wireframe** is a collection of vertices, curves and loops of curves that represent virtual topology parasite vertices, parasite edges and parasite faces respectively.

Additional information can also be included to reduce processing time, such as host entity information for each vertex and curve to establish the link with the model to decompose, the bounded/bounding relationship between vertices and curves and which operation can be used to re-create a face from the loop of curves (e.g., swept surface, fill surface). Since the objective is to apply a virtual topology split, and the final position of the nodes on these faces may eventually depend on a mesh smoothing algorithm, the exact geometry of the partition is not required. Hence, transferring the CAD curves only is sufficient and it is more flexible to transfer the scaffold required to define the cut faces than the cut faces themselves.

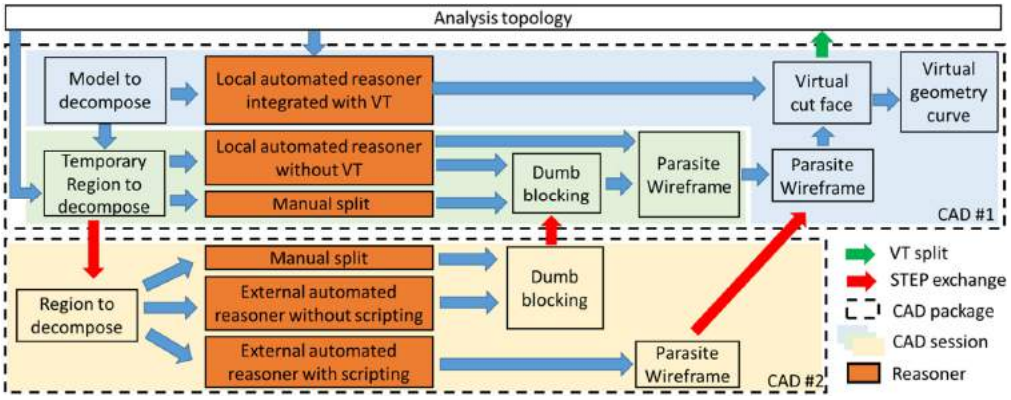

Figure 3. Integration of different reasoners with virtual topology.

3.3.3 Output processing

Reasoners that are already fully integrated with virtual topology directly define virtual topology splits and produce the necessary virtual geometry curves. For other reasoners, with some level of scripting available, defining a simple parasite wireframe is straightforward. It can then be transferred back to the CAD session of the original CAD model and processed to define virtual topology splits. The curves contained in the parasite wireframe can be used directly as virtual geometry curves or can be reconstructed to get a better fitting with the CAD model. A user can also directly specify a parasite wireframe, although it can be tedious as the curves forming a loop of a face need to be grouped manually.

For reasoners that only output a CAD decomposition, or after the user is done splitting the region of interest, the dumb blocking that results is converted into a parasite wireframe using an automated routine. It first queries all the edges and faces of each block, to identify and match coincident entities stemming from the manifold nature of the splits. Then entities are classified as existing, subset or parasite entities by comparing them with the entities of the region of interest before splitting, that are matching the analysis topology. Only parasite entities are kept to define the parasite wireframe and their host entity is also recorded. This parasite wireframe is then transferred to the original CAD environment to define the virtual topology splits.

4. SPLIT PROPAGATION

In the analysis topology, each face of each body has its own meshing strategy assigned, which is inferred from the meshing strategy of the parent body or bodies in the case of an interface. Whenever the topology of a face is modified to accommodate imprints, either to decompose the face or because of further decomposing neighbor regions sharing the interface, the flow of elements or the net number of singularities on the face may change. This implies that decomposing a body to extract hex-meshable regions can invalidate the meshing strategies previously identified on adjacent regions. As a result, special care must be taken to maintain meshing strategies as the model is incrementally decomposed.

4.1 Imprints and interfaces

Each face of the analysis topology is assigned one of the following meshing strategies:

- unstructured triangular mesh: this only exists on or between residual regions.
- unstructured quad mesh (e.g., paved): on source and target faces of sweepable regions.
- structured quad mesh (e.g., mapped): faces of block regions, walls of sweepable regions.

Unless it is an interface with a hex meshed region, there is no limitation on partitioning the faces of residual regions. In the case of source and target faces, singularities can be channeled, therefore there is no limitation on partitioning these faces. However, doing so may transform a simple one-to-one sweep into a many-to-many sweep that is not supported by many meshing tools. The condition on mapped interfaces is the most stringent, as it implies that the result of a split/imprint on the face must be a collection of faces with the same mappable properties. Otherwise, the sweepable/block strategy of the bodies will become invalid and reprocessing will be needed.

The validity of an imprint on mapped interfaces is assessed by checking how it modifies the flow of elements associated with the interfaces. The direction of the flow of elements is only modified by the introduction of negative or positive singularities on the face, which either stem from a subset with a non-null net singularity number, or from the topology of the imprint itself. Figure 4 shows various imprints on a wall face of a swept region (which must be 4 sided). The imprints in Figure 4 (a), (b) and (c) do not perturb the flow of elements from top to bottom and left to right, so they are valid, and the body bounded by the face is still sweepable. The imprints in Figure 4 (d) introduce two triangular faces that would require negative singularities (in blue). In Figure 4 (e), while all the subset faces are 4 sided, the connectivity of the imprints introduce a negative singularity that redirects part of the top-down flow of elements to the left. Figure 4 (f) is inconclusive when considering the bottom subset as a logical rectangle, as all the subsets have 4 corners and are mappable.

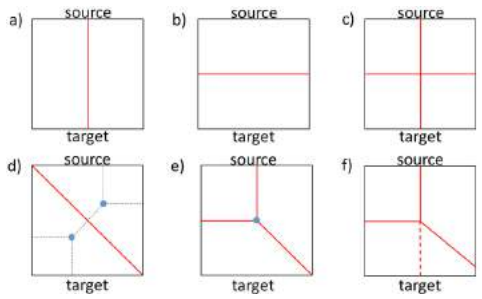

Figure 4. Valid imprints (a-c) do not modify the mesh flow, (d-e) introduce singularities making the sweep invalid, and (f) is inconclusive.

Even if all the imprints on all individual wall faces are valid and only result in mappable faces, sweepable regions require that the wall faces form a loop of mappable faces. This introduces an additional constraint on the flow of elements, which is assessed by solving the mapping constraint on the number of elements. In Figure 5, two mappable faces forming a loop receiving valid imprints are laid flat. In Figure 5 (a), solving the equality constraint on opposite edges yields *Ne2=Ne5=0* (where *Ne#* is the number of element edges on edge *e#*) which implies that the loop cannot be meshed unless the imprints are moved. On the other hand, the configuration in Figure 5 (b) is valid, but will result in elements being stretched on *e3* and compressed on *e4*.

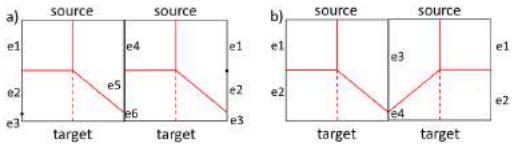

Figure 5. Loop of mappable faces with (a) invalid and (b) valid edge division balancing.

Block topology is a special case of sweepable regions with 3 pairs of opposite faces resulting in 3 possible sweeping axes, and all faces mappable. Therefore, the same approach for checking invalid imprints can be used. The only difference is that some invalid imprints can be handled by reclassifying the shape type from block to sweepable, provided there is still a loop of valid mappable faces.

If the imprints are valid, the decomposition of the face or volume can go ahead. If the face is an interface the question of the propagation of the split arises. For single imprints on wall faces aligned with the sweep as shown in Figure 4 (a), there is no need to propagate the imprint as all the wall faces of the sweep remain 4-sided.

4.2 Aligned split

The process of splitting a sweepable region by propagating imprints along the sweep direction is illustrated in Figure 6 (a), where a sweepable body has had its source face imprinted to match quad meshing requirements (in this case, imprints have been created by mid-point decomposition reasoner applied to the face). A new parasite wireframe is created to store the split information. The curves of the imprint are added along with their host face, and vertices are processed to identify host curves and merge coinciding ones. Wall edges are discretized and are used to trace discretized curves aligned with the sweep on wall faces and inside the volume, as shown in Figure 6 (b). Curves that are lying on a wall face are re-projected if an explicit surface is available, and all the curves are added to the parasite wireframe. Finally, the curves matching the imprint curves on the opposite target face are created by joining the last points of the newly created curves to match the topology of the imprint. This completes the parasite wireframe with one loop of curves identified for each imprint, producing 3 parasite faces, as shown in Figure 6 (c). The resulting analysis topology after virtual topology split and the equivalent geometric decomposition are shown in Figure 6 (d) and (e) respectively, with three simple sweepable regions without imprint generated.

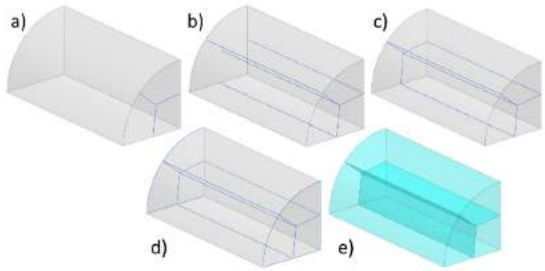

Figure 6. Imprints on the source face are propagated along the sweep direction to create virtual parasite faces splitting the sweepable region into 3 parallel sweepable regions.

4.3 Perpendicular split

Since sweepable regions are defined by a loop of mappable wall faces around the sweep direction, the propagation of imprints that are perpendicular to the sweep direction is achieved by exploiting mapping constraints to trace loops of curves. The resulting curves partition the loop of wall faces into two or more loops of mappable faces, effectively splitting the original sweep region into a chain of sweepable regions, as described in Figure 7. As for the propagation of aligned splits, a new parasite wireframe is first created and the imprint curves on wall faces (Figure 7 (a)) are added. Then, all coincident vertices are merged, and the parameter of each vertex lying on a wall edge is extracted. These parametric values are clustered within a tolerance range and new vertices are created for each cluster on wall edges without a vertex using the mean value of the cluster. In Figure 7 (b), vertices with parameters p_1 and p_2 are clustered, and a new vertex with parameter p_3 is created. Once all vertices are created, the loop of wall edges is traversed for each cluster, and vertices without existing parasite curves are joined by tracing a new curve on the wall face. The resulting loop of curves are added to the parasite wireframe and used to define a parasite face, as shown in Figure 7 (c). The resulting analysis topology after virtual topology split and the equivalent geometric decomposition are shown in Figure 7 (d) and (e) respectively, with a chain of two simple sweepable regions without imprint generated. This algorithm enables processing of multiple imprints on multiple wall faces and to propagate cuts on wall edges only.

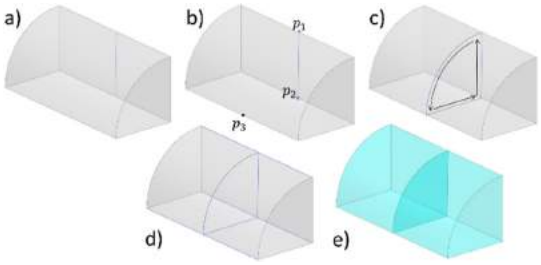

Figure 7. Perpendicular imprint on a wall face is traced around the loop of mappable wall faces to create a virtual parasite face splitting the sweepable region into 2 stacked sweepable regions.

4.4 Identifying propagation order

As the model is incrementally decomposed, the number of hex meshable regions increases throughout the process and their interaction becomes more complex. Since propagating imprints to partition sweepable bodies also produces new imprints on adjacent bodies, special care must be taken when propagating splits. If the meshing strategies assigned result in a valid mesh, propagating the imprints following the meshing constraints will also produce a valid mesh. As such, the order in which imprints are propagated in the sweep direction and perpendicular to it does not matter. However, since imprints on source faces can modify the number or position of singularity lines, it is better to propagate aligned splits first, to ensure proper channeling of the singularities.

In Figure 8 (a), the model is decomposed into one thin region and 4 sweeps. The source face of one sweepable region is decomposed resulting in the imprints in Figure 8 (b), which are first propagated to split the region (Figure 8 (c)) introducing both perpendicular and aligned imprints on neighbor sweeps. The imprints on the source faces are processed first (Figure 8 (d)), followed by the lateral propagation (Figure 8 (e)).

Eventually, the last sweep has compatible imprints on both its wall face and source face, which are propagated in the sweep direction, Figure 8 (f)).

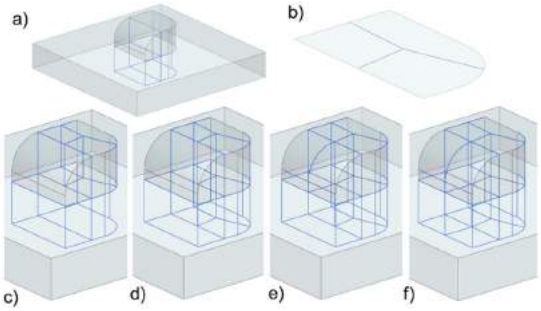

Figure 8. Imprints on the source face (b) are first propagated to the connected sweepable body (c) resulting in new imprints on neighbor regions that are recursively propagated (d-f) until no more splits can be found on sweepable bodies.

In some cases, additional meshing constraints stemming from symmetry properties and patterning can arise, where not only the topology but also that actual geometry must be matching between faces to reconnect everything. This is handled by applying the symmetry/patterning transform to the imprint curves before propagating them, to ensure they are correctly located.

5. DECOMPOSITION IN CAD

Once the incremental decomposition is complete with all the splits correctly propagated, and when no more hex meshable regions can be identified, the virtual volume cells can be extracted to generate a meshable analysis model. Rather than trying to apply a sequence of split operations matching the virtual topology operators applied, the model is decomposed by querying and using all the interfaces between bodies as cutting faces. This provides a more flexible way of partitioning the model that does not rely on the history of the decomposition process, while allowing a single region to be extracted in the model without having to perform the entire decomposition.

The final analysis model must be contained within a non-manifold CAE environment to ensure a conformal mesh is created at interfaces. The partitioning of the geometry can either be applied in a CAD environment or a CAE environment. In the first case, the virtual geometry curves are used to create the cutting surfaces, and the final blocking is exported to the destination meshing environment. In the second, virtual geometry entities are exported, and the model is decomposed by applying split operations through an API.

If the geometry decomposition is performed in a non-manifold environment the process is straightforward, and the topology of the resulting analysis model will exactly match the analysis topology. If the decomposition is carried out in a manifold CAD environment, the limitations from the manifold representation and the split capabilities of the CAD engine must be taken into consideration.

5.1 Split ordering

Extracting all the subset regions identified in a single split operation has a high chance of failing in current tools, even for reasonably simple splits such as decomposing a cube into 8 octants (Figure 9 (a) and (b)). For this reason, an incremental decomposition approach is preferred, extracting regions of interest one after the other. This however produces intermediate bodies that can exhibit invalid non-manifold touch configurations even though all the final extracted bodies would be valid manifolds. In Figure 9 (c), if the green octant is removed first, extracting the yellow octant would create a non-manifold edge on the intermediate body (in translucent grey), hence the extraction would fail. Similarly, in Figure 9 (d), extracting the green octant first followed by the blue would create a non-manifold vertex on the intermediate volume.

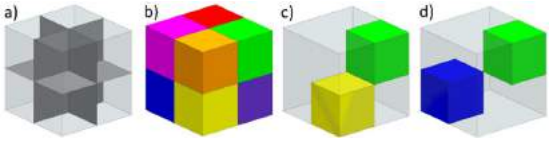

Figure 9. Invalid manifold condition on the intermediate body for different extraction order.

This issue is eliminated by prescribing a decomposition order that avoids invalid intermediate volumes and maintains the manifold condition at all times. The process starts by querying all the internal vertex and concave edge neighborhoods to initialize the list of connected volumes. If the neighborhood is complete, e.g., a vertex is fully surrounded by geometry, any touching body can be removed. If the neighborhood is incomplete, e.g., a concave edge, the touching faces that are not interfaces define a front, and only bodies bounded by faces on that front are valid candidates for extraction. For each volume to be removed, the relevant neighborhoods are checked to ensure no touching condition will be created. If the extraction is valid the body is added to the decomposition sequence and the neighborhoods are updated. Else the candidate bodies are re-ordered before the current bodies and assessed in turn.

While this process results in a propagation of the partitioning front from the boundary, it also enables the extraction of a single region, by identifying the minimal number of regions that must be extracted first where the extraction would create an invalid intermediate volume. It also reduces the number of intermediate bodies, as these are difficult to manipulate since they do not match any volume cells in the analysis topology.

5.2 Cut definition

Once the order in which the regions need to be extracted is known, the sequence of split operations and cutting geometry required to perform the decomposition need to be generated. The cutting geometry is inferred from the interfaces between bodies recorded in the analysis topology. Virtual geometry curves are combined with the existing edges bounding each interface to generate a face by fitting a surface through the curves (in effect a fill surface operation). The resulting cutting faces are then clustered to match each successive split operation. This is achieved by querying all the interfaces of the

body to extract and removing the ones that have already been used. Adjacent faces with coincident edges are sewn together within each cluster.

shows the decomposition process for the model in Figure 9 (b). The first row shows the decomposition order identified, while the second row shows the different clusters of cutting faces generated for each split operation associated with this order. The third row shows the anticipated results from the incremental splitting, with all intermediate bodies being valid manifold representation in CAD.

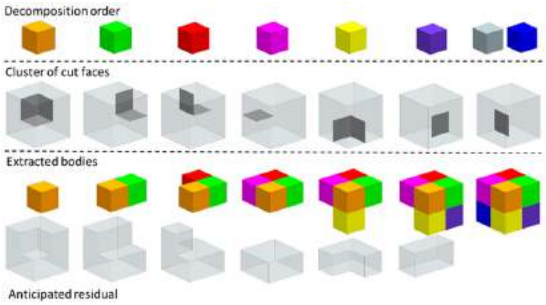

Figure 10. Cluster of faces identified for the extraction sequence. The intermediate body at each step is a valid manifold model.

5.3 Subset mapping

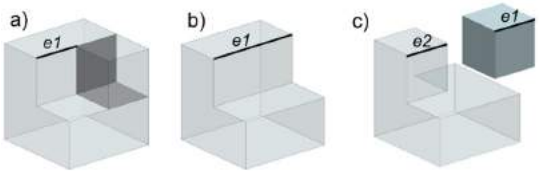

Figure 11. Persistent naming issue on edges.

When automatically applying the sequence of split operations in a CAD package, special care must be taken at each step to identify which bodies need to be split and to remap entities on the subset corresponding to the region to extract. The remapping consists in matching the B-Rep entities that have been generated by the split operation with their topological analogue that already exists in the analysis topology. This is critical to ensure downstream automation of meshing but is made difficult by the way many CAD modelers implement split operations and how they suffer from the persistent naming problem. In Figure 11 (b), a common CAD practice is to merge faces that have the same underlying surface. As a result, the bold edge *e1* is extended to bound the merged face. When the merged faces are split to recover imprints or to extract the next region, one subset inherit the attributes of the parent, which may not match the original entity. In Figure 11 (c), the edge *e1* as moved to the right following the split.

When it comes to linking the representation in the CAD system with the analysis topology description, since the topology of the region being extracted matches the analysis topology it can be identified by looking first for the CAD bodies that have the same topology. If several CAD bodies are identified, the coordinates of the mid-point of the edges can be used to match the correct subset. The re-mapping of the new CAD edges and faces is also recovered by matching the mid-point of edges.

When several intermediate bodies are created after a split, the host entity information is used to identify which one needs to be partitioned to extract the next region. All the faces and edges of the region to extract, that are subsets, are queried to get the list of host CAD entities. The intermediate CAD body that has the most matching CAD entities is then identified as the target for the splitting operation. If this test is not sufficient, point in volume methods are used to differentiate the bodies

Once all the regions have been extracted and remapped, a manifold collection of bodies will exist in the CAD environment, with all the coincident entities (e.g., bodies sharing a non-manifold interface in the analysis topology now have coincident faces in CAD) identified and labelled to automate the conversion to a non-manifold representation once transferred to a CAE package.

6. RESULTS

The incremental decomposition workflow is demonstrated within a virtual topology framework built around a relational database used to store the analysis topology, and the Siemens NX [27] CAD package, as described in [26]. In addition to various decomposition reasoners, the framework includes a meshing strategy reasoner to identify a meshing recipe from the meshing strategies. It uses integer programming to resolve mapping constraints and identify edge division numbers directly on the analysis topology. After the geometric decomposition is applied, another meshing reasoner is used to transfer the model to the NX CAE environment, recover the associativity with the analysis topology by merging coincident faces, and transfer the meshing recipe to automatically generate the mesh.

Within the current framework, fully automated workflows from the CAD model of the design to the mesh are only limited by the decomposition reasoners available and in identifying in which order they must be applied. In this work, this decision is left to the user, who applies the automated decomposition reasoners one after the other, and can also manually decompose the regions left by automated reasoners. Once satisfied with the analysis topology obtained, the user can adjust the meshing sizing parameters before the model is automatically decomposed geometrically and meshed. Further details on the virtual topology framework and automatic meshing are available in [26], and will be presented in a future paper.

6.1 Boss plate

Figure 12 presents different decompositions for a simple model of a plate with a boss that has fillets that introduce mesh singularities. All models are first processed using the thin-sheet decomposition reasoner, followed by a reasoner that identifies sweepable regions that are embedded in thin-sheets. In Figure 12 (a), a mid-point subdivision [28] reasoner is applied, resulting in a block decomposition but with all singularities meeting at the body mid-point. In Figure 12 (b), the residual is exported to CADFix [29] to use a reasoner based on the medial object.

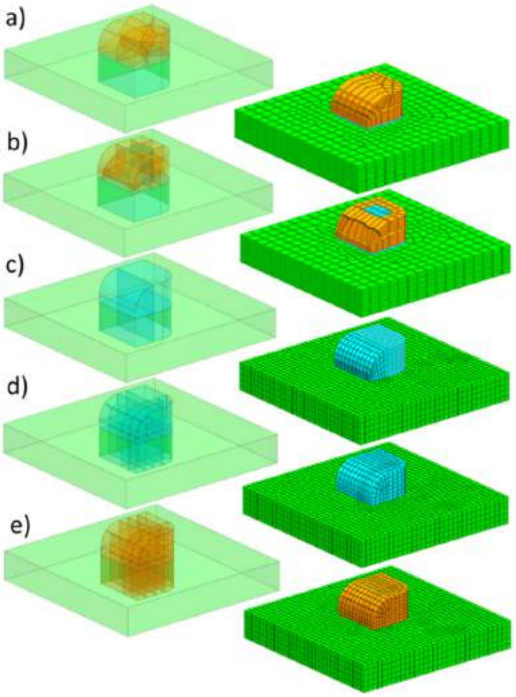

Figure 12. Different decompositions and meshes obtained for various combinations of decomposition reasoners and manual intervention.

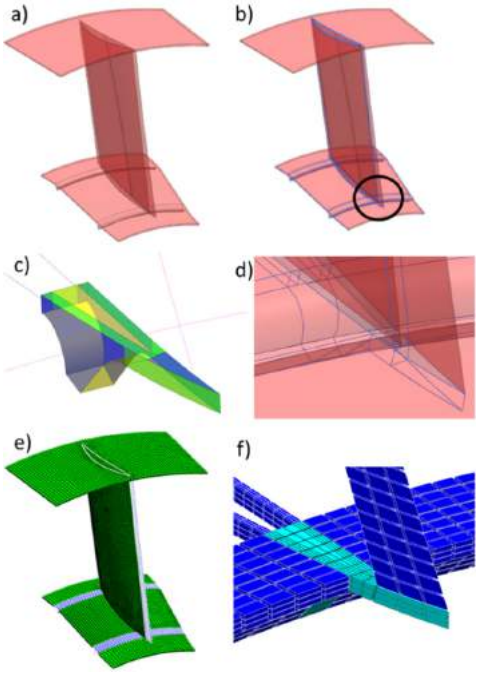

Figure 13. (a) Crescendo vane model, (b) result of automated decomposition, (c) residual for manual processing, (d) manual split converted to virtual split and (e-f) resulting automated mesh.

In Figure 12 (c), the user has specified a cutting plane to create two sweepable regions that channel the two singularity lines. In Figure 12 (d), 4 cutting planes are manually specified to extract sweepable regions, followed by automatic mid-point subdivision of the source faces to constrain the location of the singularity lines. In Figure 12 (e), the same manual decomposition is used but cube-shaped sweeps are re-classified as blocks, resulting in a full blocking of the residual.

6.2 Crescendo vane

Figure 13 shows the manual processing of a vane geometry to achieve a full hex mesh. After extracting symmetries and applying thin-sheet and long-slender tools (Figure 13 (b)), a complex residual region is left at the root of the leading edge. This is extracted (Figure 13 (c)) and manually partitioned into three sweepable regions, one to channel the singularity lines from the left and right long-slender regions, one to channel the singularity coming from the sharp leading edge, and one in between to channel the singularity coming from the yellow triangular face through the thickness. The operation is then converted into virtual splits resulting in Figure 13 (d), and the model can be automatically decomposed and meshed as seen in Figure 13 (e) and (f).

7. DISCUSSION

The robustness of the incremental decomposition based on virtual topology depends on several aspects. The robustness of the decomposition reasoners is not critical, as checks are carried out after the regions have been extracted to ensure they are suitable for hex meshing. Failed reasoners can either be re-applied with different parameters or another reasoner can be used. As a result, prototype reasoners can be added without jeopardizing the entire decomposition process. The variety of decomposition reasoners available is more important, as some reasoners might define hex-meshing strategies resulting in poor element quality, and regions that are not covered by any reasoner will either need manual decomposition or receive a tet mesh. The current limitation comes from the ability to apply virtual topology split operations, as the workflow requires the application of many operations successively and any failed operation will make any subsequent split invalid.

Beyond facilitating the integration of the different tools, having a pre-processing workflow based on virtual topology makes this approach compatible with traditional applications of virtual topology for de-featuring and geometry clean-up. The update of the decomposition is also made simpler. Sub-regions can be recombined by the virtual topology merge operator without rolling back the entire decomposition (e.g. the decomposition in Figure 12 (a) can be obtained from the one in Figure 12 (b)). The constraints associated with the meshing strategies can also be used to automatically propagate CAD design updates to the decomposition [30].

One can argue that using virtual geometry curves and extracting explicit regions for some reasoners is incompatible with the notion of virtual topology. Even simple geometries can result in complex block decompositions (see Figure 14 (a)) and inferring cutting geometry solely from the topological

requirements would create skewed angles and potentially inverted geometry that are easily avoided using virtual geometry curves. Extracting temporary geometric regions is the only realistic way currently available for a user to interact with the analysis topology and allows integration of a wide range of off-the-shelf tools within a given CAD environment.

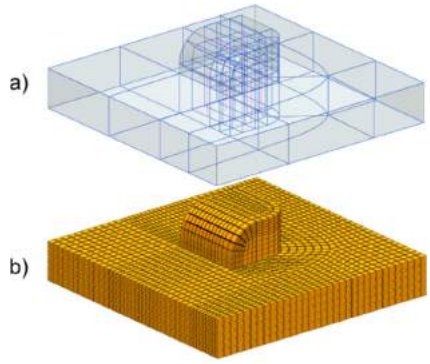

Figure 14. (a) Fully blocked model and (b) mesh file generated directly from CAD using virtual entities.

The analysis topology is essential to maintain the meshing strategies at interfaces in manifold environment and ensure that the final decomposition is suitable for meshing. Meshing strategies translate downstream meshing constraints into constraints on the decomposition that are available from within a CAD environment. The process of propagating imprints and decomposing source faces of sweepable regions may create more subset regions than necessary for achieving a good quality hex mesh, but this results in sub-regions that are simpler to mesh and compatible with a wider range of meshing tools. Eventually, a model that has been fully decomposed into block regions is in itself a very coarse hex mesh. It can be refined and meshed directly from the virtual decomposition in CAD, without having to commit the geometric decomposition and transfer to a CAE environment. In Figure 14 (b), all edges are discretized as per the meshing recipe and nodal positions on surface and inside the volume are identified using transfinite interpolation [31], before being written to a Nastran deck input file to define a mesh. On the other hand, sweepable regions with paved source faces can accommodate pair of singularities that cancel each other, redirecting the flow of the elements. This offers more freedom for node location and avoids the propagation of small element size from small details to the entire mesh.

Automatically propagating imprints is also beneficial for semi-automated decompositions workflows, as automatic partition of neighbor regions reduces the amount of work for the operator. Manual intervention can also unlock regions that are suitable for automatic processing, hence the impact of user input is maximized and no time is wasted carrying out repetitive decomposition tasks.

8. CONCLUSION

A method to integrate various automated decomposition reasoners in a single incremental decomposition workflow has been presented. All the split operations are applied using virtual topology to build an analysis topology that stores and maintains interface information and meshing strategies. This analysis topology along with virtual geometry curves are used to abstract the actual analysis model, enabling reasoners and manual users to operate on a model that is equivalent to the analysis model before it is created. Each reasoner identifies and extracts sweepable and block regions, and the meshing strategies associated with each region are used to propagate splits across interfaces to ensure everything remains hex-meshable. With all the necessary information for mesh automation available, the CAD model is decomposed to create an analysis model that can be exported to a meshing tool.

9. FUTURE WORK

Future research directions include:

- Extending the range of decomposition reasoners, in particular frame-field based methods that also focus on the handling of singularity lines.
- Integrating automatic de-featuring reasoners to remove small fillets and holes using virtual topology.
- Further investigating virtual topology meshing capabilities, including sub-mapping and paving.

AKNOWLEDGMENTS

The authors wish to acknowledge the financial support provided by Innovate UK through the COLIBRI (ref 113296) project. We also thank Rolls-Royce for permission to publish this paper.

REFERENCES

[1] J. Sarrate, E. Ruiz-Gironés, and X. Roca, "Unstructured and Semi-Structured Hexahedral Mesh Generation Methods," *Comput. Technol. Rev.*, vol. 10, pp. 35–64, 2014.

[2] S. J. Owen *et al.*, "An Immersive Topology Environment for Meshing," in *Proceedings of the 16th International Meshing Roundtable*, Berlin, Heidelberg: Springer, 2008, pp. 553–577.

[3] Y. Lu, R. Gadh, and T. J. Tautges, "Feature based hex meshing methodology: feature recognition and volume decomposition," *Comput. Des.*, vol. 33, no. 3, pp. 221–232, Mar. 2001, doi: 10.1016/S0010-4485(00)00122-6.

[4] D. White, L. Mingwu, and S. Benzley, "Automated hexahedral mesh generation by virtual decomposition," in *Proceedings of the 4th International Meshing Roundtable*, 1995, pp. 165–176.

[5] L. Sun, C. M. Tierney, C. G. Armstrong, and T. T. Robinson, "An enhanced approach to automatic decomposition of thin-walled components for hexahedral-dominant meshing," *Eng. Comput.*, vol. 34, no. 3, pp. 431–447, Nov. 2018, doi: 10.1007/s00366-017-0550-x.

[6] H. Wu, S. Gao, R. Wang, and J. Chen, "Fuzzy clustering based pseudo-swept volume decomposition for hexahedral meshing," *Comput. Des.*, vol. 96, pp. 42–58, Mar. 2018, doi: 10.1016/J.CAD.2017.10.001.

[7] X. Roca and J. Sarrate, "Local dual contributions: Representing dual surfaces for block meshing," *Int. J. Numer. Mehtods Eng.*, vol. 83, pp. 709–740, 2010, doi: 10.1002/nme.2852.

[8] N. Kowalski, F. Ledoux, M. L. Staten, and S. J. Owen, "Fun sheet matching: towards automatic block decomposition for hexahedral meshes," *Eng. Comput.*, vol. 28, no. 3, pp. 241–253, Jul. 2012, doi: 10.1007/s00366-010-0207-5.

[9] R. Wang, C. Shen, J. Chen, H. Wu, and S. Gao, "Sheet operation based block decomposition of solid models for hex meshing," *Comput. Des.*, vol. 85, pp. 123–137, Apr. 2017, doi: 10.1016/J.CAD.2016.07.016.

[10] M. A. Price and C. G. Armstrong, "Hexahedral Mesh Generation by Medial Surface Subdivision: Part II. Solids with Flat and Concave Edges," *Int. J. Numer. Methods Eng.*, vol. 40, no. 1, pp. 111–136, 1997.

[11] "Sandia National Laboratories: index." https://cubit.sandia.gov/ (accessed Aug. 03, 2021).

[12] J. H.-C. Lu, W. R. Quadros, and K. Shimada, "Evaluation of user-guided semi-automatic decomposition tool for hexahedral mesh generation," *J. Comput. Des. Eng.*, vol. 4, no. 4, pp. 330–338, Oct. 2017, doi: 10.1016/J.JCDE.2017.05.001.

[13] J. H.-C. Lu, I. Song, W. R. Quadros, and K. Shimada, "Volumetric Decomposition via Medial Object and Pen-Based User Interface for Hexahedral Mesh Generation," *Proc. 20th Int. Meshing Roundtable, IMR 2011*, pp. 179–196, 2011, doi: 10.1007/978-3-642-24734-7_10.

[14] T. Blacker, "Automated Conformal Hexahedral Meshing Constraints, Challenges and Opportunities," *Eng. Comput.*, vol. 17, no. 3, pp. 201–210, Oct. 2001, doi: 10.1007/PL00013384.

[15] K. Miyoshi and T. Blacker, "Hexahedral Mesh Generation Using Multi-Axis Cooper Algorithm Cubit Mesh Generation," in *Proceedings of the 9th International Meshing Roundtable*, 2000, pp. 89–97.

[16] H. Wu, S. Gao, R. Wang, and M. Ding, "A global approach to multi-axis swept mesh generation," *Procedia Eng.*, vol. 203, pp. 414–426, Jan. 2017, doi: 10.1016/J.PROENG.2017.09.817.

[17] A. Sheffer, M. Bercovier, T. Blacker, and J. Clemets, "Virtual Topology Operators for Meshing," *Int. J. Comput. Geom. Appl.*, vol. 10, no. 03, pp. 309–331, Jun. 2000, doi: 10.1142/s0218195900000188.

[18] C. M. Tierney, L. Sun, T. T. Robinson, and C. G. Armstrong, "Using virtual topology operations to generate analysis topology," *Comput. Des.*, vol. 85, pp. 154–167, 2017, doi: 10.1016/j.cad.2016.07.015.

[19] K. Ho-Le, "Finite element mesh generation methods: a review and classification," *Comput. Des.*, vol. 20, no. 1, pp. 27–38, 1988, doi: 10.1016/0010-4485(88)90138-8.

[20] T. D. Blacker and M. B. Stephenson, "Paving: A new approach to automated quadrilateral mesh generation," *Int. J. Numer. Methods Eng.*, vol. 32, no. 4, pp. 811–847, 1991, doi: 10.1002/nme.1620320410.

[21] C. M. Tierney *et al.*, "Efficient Symmetry-Based Decomposition for Meshing Quasi-Axisymmetric Assemblies," *Comput. Des. Appl.*, vol. 16, no. 3, pp. 478–495, 2019, doi: 10.14733/cadaps.2019.478-495.

[22] L. Sun, C. M. Tierney, C. G. Armstrong, and T. T. Robinson, "Decomposing complex thin-walled CAD models for hexahedral-dominant meshing," *Comput. Aided Des.*, vol. 103, pp. 118–131, Dec. 2018, doi: 10.1016/j.cad.2017.11.004.

[23] B. Lecallard *et al.*, "Automatic Hexahedral-Dominant Meshing for Decomposed Geometries of Complex Components," *Comput. Des. Appl.*, vol. 16, no. 5, pp. 846–863, 2019, doi: 10.14733/cadaps.2019.846-863.

[24] N. J. Taylor and R. Haimes, "Geometry modelling: Underlying concepts and requirements for computational simulation (invited)," *2018 Fluid Dyn. Conf.*, 2018, doi: 10.2514/6.2018-3402.

[25] J. Kripac, "A mechanism for persistently naming topological entities in history-based parametric solid models," *Comput. Des.*, vol. 29, no. 2, pp. 113–122, Feb. 1997, doi: 10.1016/S0010-4485(96)00040-1.

[26] B. Lecallard, "Virtual topology based hex-dominant meshing and re-meshing," PhD thesis, Queen's University Belfast, 2020.

[27] "NX | Siemens Digital Industries Software." https://www.plm.automation.siemens.com/global/fr/products/nx/ (accessed Aug. 03, 2021).

[28] T. S. Li, C. G. Armstrong, and R. M. McKeag, "Quad mesh generation for k-sided faces and hex mesh generation for trivalent polyhedra," *Finite Elem. Anal. Des.*, vol. 26, no. 4, pp. 279–301, Aug. 1997, doi: 10.1016/S0168-874X(96)00085-6.

[29] "ITI - International TechneGroup | CADfix." https://www.iti-global.com/cadfix (accessed Mar. 06, 2019).

[30] B. Lecallard, C. M. Tierney, T. T. Robinson, C. G. Armstrong, D. C. Nolan, and A. E. Sansom, "Updating and Re-meshing Virtually Decomposed Models," in *Proceedings of the 28th International Meshing Roundtable*, 2019, pp. 50–67.

[31] L. E. Eriksson, "Generation of Boundary-Conforming Grids Around Wing-Body Configurations Using Transfinite Interpolation," *Aiaa J.*, vol. 20, no. 10, pp. 1313–1320, 1982, doi: 10.2514/3.7980.