UPDATING AND RE-MESHING VIRTUALLY DECOMPOSED MODELS

Benoit Lecallard[1], Christopher M. Tierney[1], Trevor T. Robinson[1], Cecil G. Armstrong[1], Declan C. Nolan[1], Alexander E. Sansom[2]

[1]*Queen's University Belfast, Belfast, U.K. t.robinson@qub.ac.uk*
[2]*Rolls-Royce Plc, Alexander.Sansom@rolls-royce.com*

ABSTRACT

Generating hexahedral meshes is often an expensive process, which limits the use of high-fidelity numerical simulation methods for design. Hexahedral meshes can be generated by decomposing a geometric model into simpler meshable regions, but robustly propagating design modifications to the decomposed representation makes any attempt to update the mesh very challenging. In this paper, a virtual topology workflow enabling automatic generation of hex-dominant meshes is extended to propagate parametric modifications and feature changes to the decomposition and resulting mesh. Geometric and topological modifications are identified and linked to the decomposition through virtual topology relationships. Modified regions are localized and reasoning on the virtual decomposition enables their definition and associated meshing strategy to be updated. Instead of starting the meshing process from the beginning, only modified cells are re-meshed. This provides an efficient and automated method to propagate design changes down to the analysis model.

Keywords: Automatic decomposition, mesh generation, mesh update, hexahedral, parametric models, virtual topology

1. INTRODUCTION

The increasing use of finite element analysis throughout a product lifecycle is limited by the ability to generate appropriate simulation models. This is especially true for the simulation of complex events, such as crash or large displacement analyses, where generating the hexahedral (hex) elements preferred for this task is very user-intensive work. Simulation-based design depends on the ability to quickly generate analysis models for many design variations to run an optimization procedure. This is incompatible with the manual analysis set up required in a typical hex meshing workflow. Analysis requirements are also prone to change, especially within coupled multi-physics analyses where analysis results from one domain dictate updates for another. For example, if a model is deformed by wear or thermal expansion and is used as an input for subsequent analysis these changes must be reflected in the downstream analysis model.

Many tools have been developed during the last decades in an attempt to automate hex meshing with various results [1]. They include direct methods such as Whisker Weaving [2] and Plastering [3], as well as indirect methods such as tet-combination [4]. Decomposition-based methods partitioning the design geometry into smaller sub-regions for which a simple meshing strategy can be found are the most widely used methods. These meshing strategies include mapping [5], sub-mapping [6] and sweeping [7], where a quad mesh of a source face is swept in order to generate 3D hex elements. Even if these tools fail to tackle generic geometries, they can be integrated in an incremental decomposition workflow to significantly alleviate the workload of generating meshes. Virtual topology-based decomposition, coupled with cellular modelling for meshing workflows, has shown promising results and flexibility [8]–[10]. The benefits of using virtual topology (VT) for generating meshes without altering the CAD model definition were first presented by Sheffer et al. [11]. In the context of this work, a cellular model is a decomposition of space into cells of analysis significance. The interfaces in the model are robustly captured and are cells in their own right [12]. This structure means that the links between the decomposed virtual representations and the design model can be robustly maintained [13]. In addition, as a model is decomposed for meshing, the cellular representation maintains connections between the subset domains that enable both the automation of downstream meshing and the localization of modified cells after design changes.

After design changes, the mesh needs to be updated to remain an accurate representation of the model. In the Computer-Aided Design (CAD) environment, updating the

design is straightforward, either by changing parameters associated to the model or by adding/removing features in the construction tree. However, applying the same design modification to geometrically identical models that are constructed with different feature orders may produce unexpected differences that may be inadvertently propagated to the analysis model.

Direct geometric editing (synchronous technology) is also available, where the user can interactively manipulate geometric entities without requiring access to the construction tree, allowing modifications of the model outside of the design environment. However, the analysis model is often constructed in a separate Computer-Aided Engineering (CAE) package, where such manipulations will break the link to the original CAD model, and CAD-CAE integration is a major bottleneck toward automation[14]. As a result, the mesh cannot be as easily updated as the construction tree and parameters are lost during the transfer across packages. The user is often tasked to update the CAD model before exporting it to the meshing environment where many pre-processing activities, such as clean-up operations, carried out to create the previous analysis model must be repeated. Furthermore, most automated tools for hex-mesh generation [2]–[4] do not offer the ability to efficiently update the mesh and require the whole decomposition process to be repeated. One challenge is that decomposition tools are often used to partition the design model geometrically to create a decomposed representation fit for hex meshing. These geometric partitions usually break the links between the design model and the decomposed representation, meaning that even if design changes can be identified there are no relationships that can be exploited in order to robustly update the decomposition. The major challenges to automatically updating a hex-dominant mesh created from a decomposition via the approach proposed herein are to:

- Identify the geometric and/or topological changes resulting from a design update.
- Reflect design amendments on the analysis model by exploiting the virtual topology relationships stored when generating the initial decomposition.
- Update the decomposition used for meshing and ensure it remains valid.
- Minimize the computational expense by re-using as many existing elements as possible.
- Maintain mesh quality after update.

This work proposes to extend a virtual topology decomposition workflow to address this problem, by using an integrated cellular model to reflect parametric and feature changes on the mesh. This paper first describes the virtual topology framework used for automatic decomposition and hex-dominant meshing. Then, handling feature and parametric perturbations for re-meshing is presented. This is done by first localizing the modifications, then updating the analysis topology and finally updating the mesh locally. The main contribution is the implementation of a cellular mesh and interface management to enable mesh updates even when the topology of the model is significantly altered. Considerations on meshing strategies are presented to help update the decomposition and the mesh.

2. RELATED WORK

Mesh update has been a topic of research for many years and includes several domains such as mesh morphing, mesh adaption and re-meshing. Mesh morphing [15] consists of mapping an initial mesh onto a new geometry which is similar to the initial geometry, either to account for the model deformation, or to re-use an existing mesh on a similar geometry (e.g. a design update). The mapping of nodes requires knowledge of both the entities mapping between the geometries and the node to entity associativity in the original mesh. The mesh topology must remain constant to ensure a correct mapping.

Mesh adaption [16] consists of modifying a mesh against known quantities. It can be an iterative process to minimize the simulation error while solving the computational problem iteratively. Parametric modification in an optimization loop can also drive mesh adaption procedures. The mesh can be locally refined or coarsened, with or without connectivity (or number of elements) modifications. Sheffer and Üngör have proposed a dual representation using both the boundary and a parametric representation to link design modifications and mesh adaption procedures [17]. In particular, the history of virtual operations applied for simplification is retained and automatically mapped on the updated design model before meshing. Mesh updating is done by moving the elements to the new geometry, and then adjusting the mesh quality by using techniques such as whisker-sheet operations. Feature displacements have been investigated recently by Shen et al. [18], using mesh deformation and the mesh is refined using density fields extracted from the initial mesh. However, adaption approaches are limited to simple parametric perturbations, as the mesh topology needs to be consistent to be mapped.

Parametric perturbations can induce topological modifications on a model. To this extent, Van Der Meiden and Bronsvoort have defined a method to relate the range of parameters to topological entities, therefore identifying critical parameters of interest [19]. Sun et al. have proposed a method using virtual topology to deform a surface mesh in the presence of simple topological perturbations [20].

Re-meshing is employed whenever mesh generation methods need to be re-applied either locally or globally to the model. For example, fully re-meshing a model can be avoided after feature insertion if the feature is meshed with new elements and connected to the existing mesh. Smit and Bronsvoort have successfully implemented a cellular modelling-based approach to tetrahedral re-meshing [21]. After capturing the feature differences between two models and their interactions, all the valid original nodes are mapped to the new model, and new elements are created to fill the gaps. However, in the absence of any link or equivalence between elements and geometric cells, all the nodes need to be classified and processed for re-meshing, which can be slow for very large meshes.

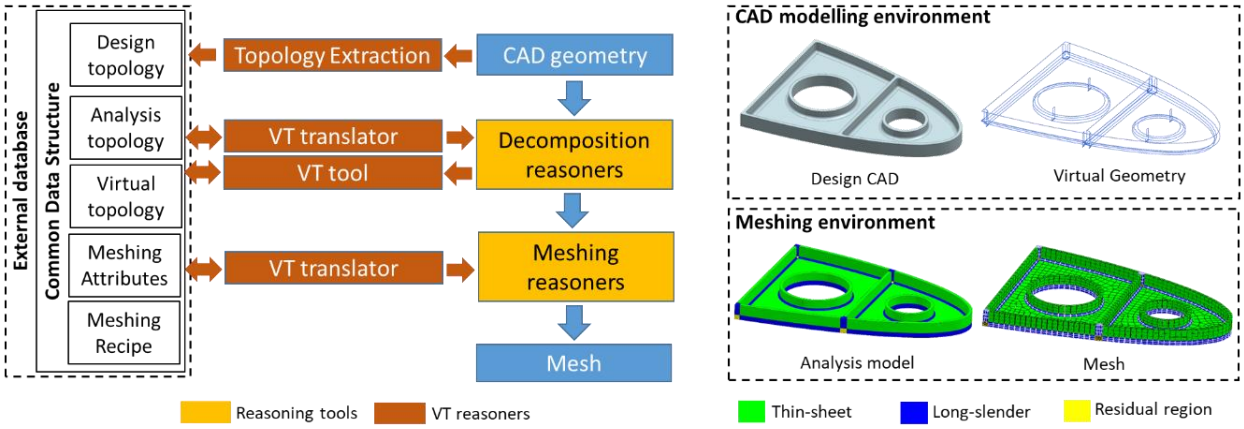

Figure 1. Virtual topology-based workflow for automatic decomposition and meshing.

In this paper, hex-dominant re-meshing is addressed using a cellular representation of a virtually decomposed model. Only the sub-regions that need to be modified to accommodate the design change are identified and re-meshed, while maintaining the constraints of a structured mesh. This allows design variations to be automatically propagated throughout a virtual topology workflow and reflected on the final mesh (Figure 4).

3. VIRTUAL TOPOLOGY DECOMPOSITION AND MESHING

This section describes a virtual topology-based workflow that integrates various reasoners for automatic decomposition and meshing (see Figure 1).

The process takes a CAD model as an input and has two main outputs. The first is a topological description of the decomposed model called analysis topology, contained in a Common Data Structure (CDS). The second is a mesh in a CAE environment that is exported to a neutral format file. The original topology is first extracted from the CAD to the CDS, and a series of virtual topology split operations are applied to create the analysis topology. This analysis topology contains the virtual decomposition and is linked to the original design topology through virtual topology relationships stored in the CDS. Once the model has been virtually decomposed, a mesh can be automatically generated by using the meshing recipe reasoner (section 3.4) and the mesh reasoner (section 3.5). Virtual topology allows more freedom for preparing a model for meshing and provides much more flexibility since the actual geometry is not modified. Instead of storing partitioning surfaces, only the method to construct them is stored (along with virtual geometry curves, curves that are not connected to the B-Rep design model but exist in the CAD environment, see section 3.3) and passed to the meshing environment to generate the mesh.

3.1 Design topology extraction to CDS

The role of the common data structure (CDS) is to convey information between the different steps and packages involved in the automated virtual workflow. It is based on an external SQL relational database, similar to the one presented by Tierney et al. [22], [23]. The relations relevant to virtual topology decomposition and meshing are shown in Figure 1. Other data that does not fit within the CDS is transferred using neutral formats. For example, virtual geometry can be transferred using STEP or Parasolid format. Meshes can be transferred using formatted text files, such as bulk data files that contain nodal information and element connectivity. The CDS contains the links to connect these various representations.

The CDS is initialized by querying all the entities contained in the CAD model through a topology extraction tool (Figure 1). Each entity is assigned a unique identifier when added to the database, which is linked to the name and/or tag attribute from the CAD system. Geometric attributes such as coordinates for vertices or mid-points for edges are also stored to aid tracking entities. Higher dimension entities can also be identified from their bounding entities.

The topological definition of the CAD model is extracted and stored in a design topology relation, which is duplicated in an analysis topology relation. Both topology relations contain a cellular representation of the model, with each topologic cell defined recursively from lower dimension elements forming their boundaries, along with the relative orientation between the bounded cell and its boundary.

To enable virtual topology manipulations, a virtual topology relation stores the history of the virtual operations applied on the design topology to create the decomposition in the analysis topology. The virtual topology relation contains the link between the virtual entities (subset or superset depending on the virtual split or merge operation) and their host entities.

3.2 Decomposition reasoners

Regions suitable for hex mesh generation are identified using a sequence of decomposition reasoners within the proposed virtual topology workflow. A decomposition reasoner encapsulates an algorithm to identify regions suitable for a specific meshing method (e.g. sweeping, mapping, template…) in a generic way. Implementation specific routines are handled outside of the reasoner, which

can be seen as a black box from the point of view of the process. The input is a design model with its topology extracted in the CDS. The output is splitting information which is used by the virtual topology tool to partition the domain, and a meshing strategy attribute specific to the reasoner. The power of this virtual topology approach is that multiple decomposition reasoners (sweep, multi-sweep, 3D block, 2D block…) can be integrated seamlessly within one workflow

All the geometric queries of the reasoners are made through a VT translator tool, which uses information stored in the CDS and the combination of the CAD and virtual geometry to query the analysis topology model and not the design model. Hence, the reasoner can operate on a model that has been virtually decomposed beforehand. This ensures that reasoners can be used one after another and in any order, but also allows virtual defeaturing to be carried out before the decomposition and not just as a final step before meshing. This ability to operate in the presence of virtual topology is critical for a robust analysis workflow.

After the reasoners have identified which region to extract, the splitting information is created. This is done by identifying which entities need to be partitioned, and what existing entities can be used to do so. Necessary geometric information such as points or curves to complete the definition of a split are also created by the decomposition reasoners.

3.3 Virtual Topology reasoners

Virtual topology uncouples the topological definition of a model from the geometrical one. This enables virtual generation of a meshed analysis model without altering the design model. Two VT reasoners are used in this process. The VT translator transfers geometric information between the real CAD model and the virtually decomposed model in the CDS. The VT tool manipulates topological representations in the CDS to clean or decompose a model, by applying operators as described in [8], [11]. These operators ensure that the analysis topology remains valid after topological manipulation, with the relative orientation of the virtual entities and the modifications properly recorded and updated in the CDS.

The VT translator processes the geometrical splitting information from the decomposition reasoner to create in the CAD environment all the curves required to define partitioning faces. These curves are referred to as virtual geometry, since they only exist as a layer of geometry independent from the design model in the CAD environment. Virtual geometry curves are used as an input by the VT tool to virtually partition the model to create the analysis topology. This analysis topology is linked to the design topology by a series of VT relationships resulting from the application of VT operators, stored in the CDS.

Virtual topology requires definition of virtual entities to formalize the relationship between virtual entities and their host entities (if any) [11]:

- Parasite entities: entities that did not exist in the original topology but lie on an existing entity of higher dimension (i.e. an edge lying on the face it splits or a face that lies in the interior of a body).
- Subset entities: subsets of host entities that are split by a parasite entity of lower dimension.
- Orphan entities: entity without host (e.g. an edge in the interior of a body bounding only parasite faces).

A virtual split operation uses a parasite entity on a host entity of higher dimension to create subset entities. For example in Figure 2(a), a face *f1* which is a bounded portion of a geometric surface can be split by adding a parasite edge *e1* which is a bounded portion of a curve. The parasite edge lies on the host shape to divide it into two subset faces *f2* and *f3*, but a single surface definition remains.

A virtual merge operation on the other hand groups multiple host entities into one by removing lower dimensional entities common to the hosts at their interfaces. For example, in Figure 2(b), a vertex *v2* bounding only two edges can be removed to generate one superset edge *e4*. A merge operation is required to update a modified decomposition since it enables the recombination of adjacent cells locally, without having to undo the whole decomposition operation. It can also be used to simplify and clean the model definition to facilitate mesh generation.

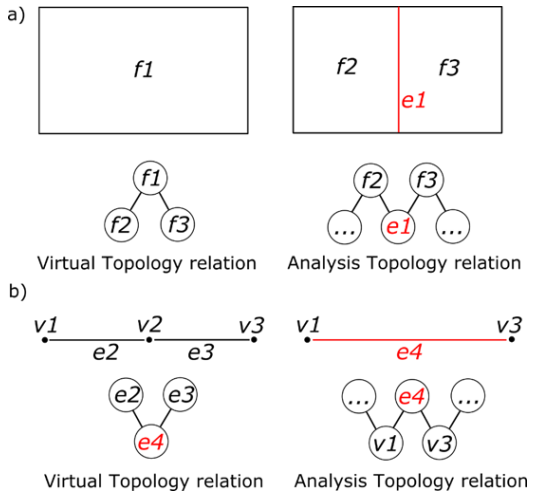

Figure 2. (a) The face *f1* is virtually split by inserting the parasite edge *e1*, and (b) edges *e2* and *e3* are merged into *e4* by virtually removing *v2*. Red entities are virtual geometry.

Virtual parasites, subsets and orphan edges are superimposed on the CAD design model as virtual geometry. All their links with the analysis model only exist in the CDS, and their purpose is to provide geometric information that does not exist in the original design model. They are created by a decomposition reasoner and can be easily transferred between packages using a neutral format such as STEP. Once added in the CDS, they can be used to aid virtual topology manipulations.

3.4 Meshing recipe reasoner

The meshing recipe reasoner is used to translate the different meshing strategies identified by the decomposition

reasoners into compatible mesh controls for hex-dominant mesh generation. The input is a CDS containing the analysis topology of the decomposed model along with the meshing strategy identified for each volume cell, and geometric information, such as aspect ratios of regions and curve lengths, previously extracted from the CAD. The reasoner outputs optimized division numbers for every curve and meshing methods for faces to the CDS.

Different decomposition reasoners can be applied to the model which will result in different meshing strategies with different priorities. The specific reasoners used in this work sought to create an anisotropic hex meshing recipe which was conformal throughout.

Sweeping strategies are converted into constraints on the number of elements following the methods applied in [9]. Source faces of sweepable regions can either be paved or mapped, while wall faces must be mapped meshed. Figure 3 shows how the meshing constraints propagate through the model, and the resulting mesh. Soft or hard goals on division numbers are applied on each edge of the model. A hard goal ensures a fixed division number will be applied (e.g. number of elements through thickness), while soft goals are optimized to meet the constraints. Constraints are checked to remove overly constraining mapping equalities. All the necessary geometric information is contained in the CDS, hence this reasoner is package independent.

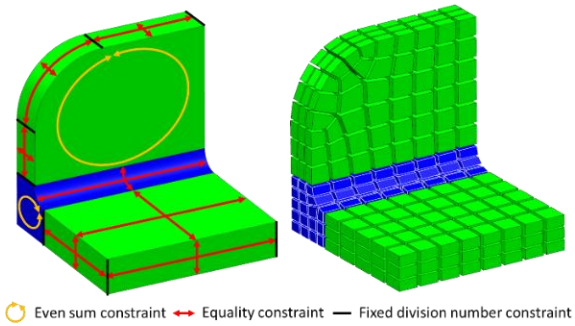

Figure 3. Flow of meshing constraints and associated mesh.

The LPSolve [24] package is used to optimize each individual number of elements on curves, by implementing a revised simplex algorithm. As a result, the mesh is fully constrained, which ensures order independence during the meshing step, and guarantees a conformal mesh will be obtained at interfaces.

3.5 Mesh reasoner

Once the meshing recipe has been generated, the mesh can be generated in a CAE environment using the meshing reasoner. The input to the mesh reasoner is the CAD model and the CDS containing the meshing recipe. The output is a conformal mesh.

The virtual partitioning surfaces are explicitly rebuilt from the virtual geometry and used to split the geometry of the model, hence becoming interfaces between sub-regions. Depending on the package used, the model is transferred to the meshing environment before or after the geometric decomposition which will create all the analysis topology entities. Mesh densities contained in the meshing recipe are applied on each curve.

All the interfaces are checked and meshed first to ensure a conformal mesh is obtained. The 2D surface meshes of all the interfaces are stored in a common neutral format file, with elements grouped by interface identifiers. This step is required to enable mesh manipulation later, but it also offers the possibility of the 3D meshes being generated on the individual cells in parallel. All the source faces of sweepable regions are meshed first, and hex elements are created by sweeping. Residual regions, where there is no known hex-meshing strategy identified by the reasoners, are tet-meshed at the end, after a layer of pyramid elements has been inserted to conform to the quad mesh of the interfaces. If the decomposition reasoners have identified hex-meshing strategies other than sweeping, the relevant meshing algorithms can be applied by the meshing reasoner.

The mesh is then exported in a neutral format file such as a Nastran input deck. This format enables the mesh to be transferred into different meshing packages, and to be edited simply by editing the mesh file.

3.6 Integrated workflow

The choice and sequence of decomposition reasoner to apply is left to the user, while pre-defined workflows can be identified for specific classes of geometries. This sequence will define which meshing methods will be used, since the same region could be identified by different reasoners for different hex-meshing methods. An example of a virtual decomposition workflow for automatic meshing is shown in Figure 1. It includes a thin-sheet decomposition reasoner for identifying thin regions which can be sweep-meshed through their thickness and a long-slender decomposition reasoner for identifying regions with one large dimension suitable for sweeping. Models of thin-walled components are suitable for thin-sheet extraction, where regions with one small dimension compared to the other two offer a simple sweep-meshing strategy [25]. Pairs of large parallel faces are discretized and imprinted one onto another in order to calculate their intersection in the parametric space. The result is then projected back on the boundary representation to identify appropriate partitioning geometry, which will be used to create the virtual geometry and the virtual split operations for sweep meshing through the thickness. This integrated virtual topology workflow effectively demonstrates multi-sweeping in thin-walled components, with explicit interfaces in the decomposed cellular model facilitating multi-directional sweeping.

Truss-like structures, or thin-walled structures with their thin sheet removed, feature a lot of long-slender regions. A similar approach to Sun's method [26] is used to extract such regions. Long edges with a large aspect ratio relative to the width of the faces they bound are identified and grouped into loops. These loops are then used to find loops of mappable faces, which verify the conditions for sweep meshing. Cap faces are identified as a loop of virtual edges. There may also be an offset applied if the geometry is prone to the existence of skewed elements. This virtual geometry is then used to help virtually split the analysis model.

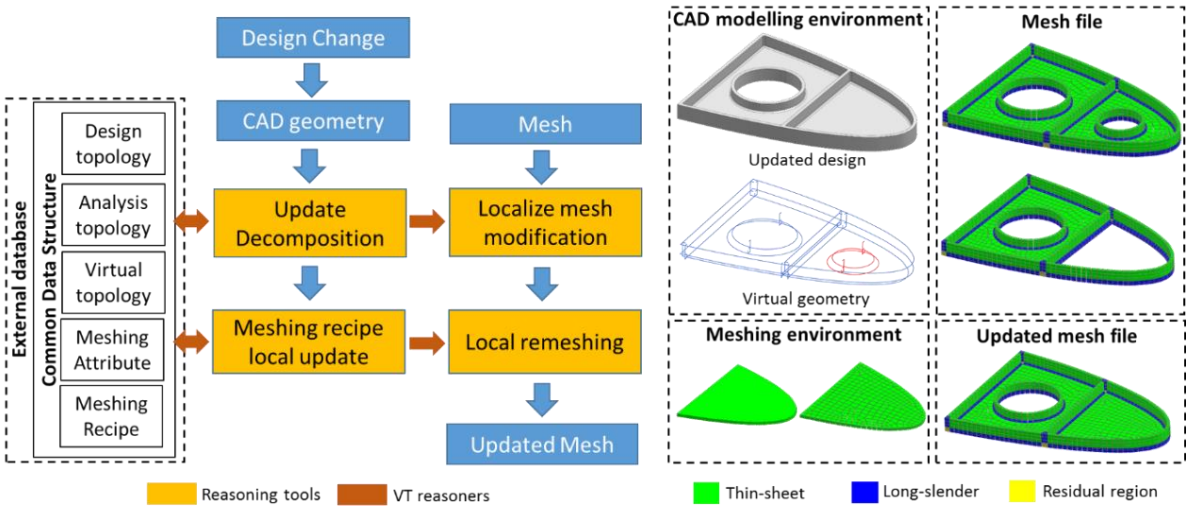

Figure 4. Workflow for updating the decomposition and mesh after design change.

Region attributes such as whether a region is thin-sheet or long-slender are stored in the mesh recipe relation of the CDS. This relation, along with the cellular model of the analysis topology, informs the reasoner tools and enables automatic identification of the meshing recipe. The meshing recipe is then stored in the CDS, describing face and edge meshing constraints in terms of size or number of elements.

Other decomposition reasoners have also been developed to identify simple sweepable regions or to decompose models into axisymmetric regions and repeated cyclic sectors, providing a minimal meshable representation [27].

4. UPDATING THE DECOMPOSITION

Figure 4 shows how the virtual workflow described in the previous section can be extended to handle design modifications to update the decomposition and ultimately the mesh. This section describes first how design modifications are identified by comparing the new design with the one stored in the CDS. Then, the constraints stemming from the hex-meshing strategies assigned to regions guide the update of the virtual geometry and the analysis topology. This reasoner takes a CAD model with a design change and the CDS associated with the previous version of the design as an input, and outputs an updated CDS for the new design (with updated virtual geometry), which can be used to update the mesh.

Modifications of the design can have various effects on the boundary representation of a model, especially for decomposed models where the number of boundary entities is increased. Figure 5 shows an example of a model decomposed for sweep-meshing undergoing various design modifications. Any design changes on a model can be classified into the following types:

- Topology only modifications, where the boundary representation is modified but not the shape. For example, introducing imprints on a face subdivides the face but the underlying surface geometry remains the same.
- Geometric only modifications, e.g. Figure 5(c) where only the geometry of the design is modified by changing the part length. All topology remains unchanged.
- Geometry and topology modifications, e.g. where new features, such as bosses, fillets etc. are added or removed from a model, Figure 5 (d), or where a parametric perturbation results in an additional topology change.

In order to update the decomposition, it is necessary to propagate the aforementioned modifications to the analysis topology. More specifically, the parasite entities used to virtually decompose the model must be modified (if necessary) alongside the virtual geometry in order to enable the mesh to be updated. In this work, design modifications can affect:

- Only the analysis model geometry. In this case it is necessary to determine if the virtual geometry needs to be updated, e.g. in Figure 5 (c) where the change in part length L requires the invalid virtual geometry (dashed red lines) to be morphed to the new model boundary.
- Both the analysis topology and virtual geometry, e.g. feature modifications will trigger geometric and topological modifications to propagate to the analysis model, such as removing the fillet in Figure 5 (d).

If the parametric perturbation has modified the design topology, then the analysis topology is also modified. However, it is possible the topological connectivity of parasite entities can be modified without changing the design topology. For example, the thickness *t* of the bottom pad is increased in Figure 5 (e), resulting in parasite entities whose configuration is now altered. The two parasite faces were disconnected in the original decomposition. However, in the updated decomposition, Figure 5 (e), they now share a common edge (in dashed bold). These changes can be subtle but will have a profound impact on updating the mappings required to update the mesh automatically.

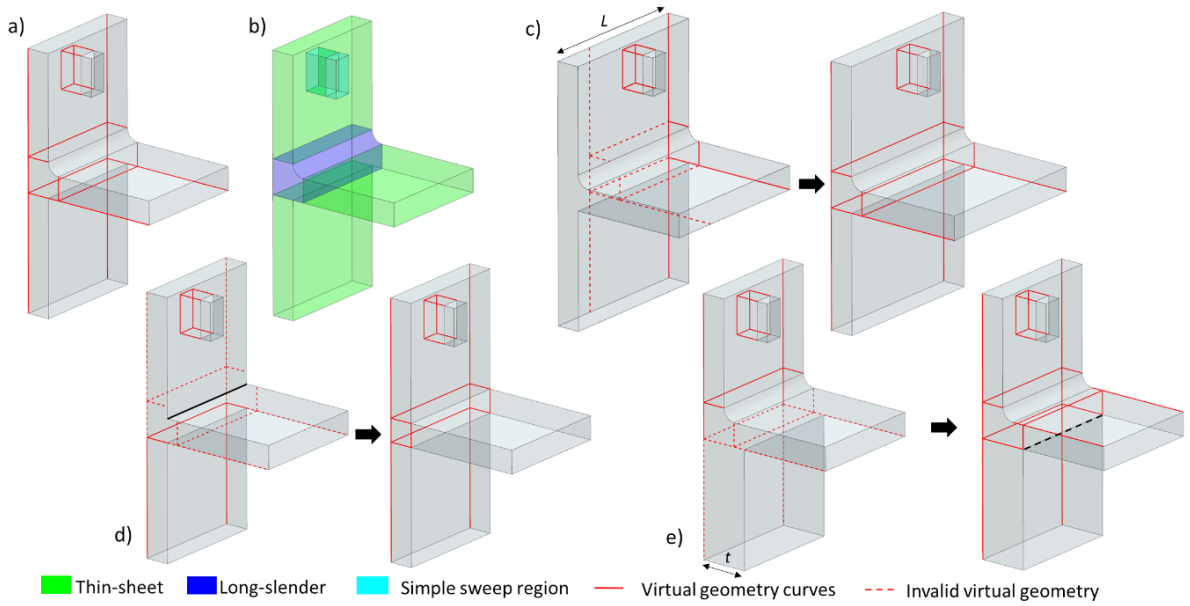

Figure 5. (a) initial decomposition, (b) corresponding meshing strategies, (c) geometric only change, (d) topological modification and (e) only the analysis topology is modified, one edge has an invalid projection.

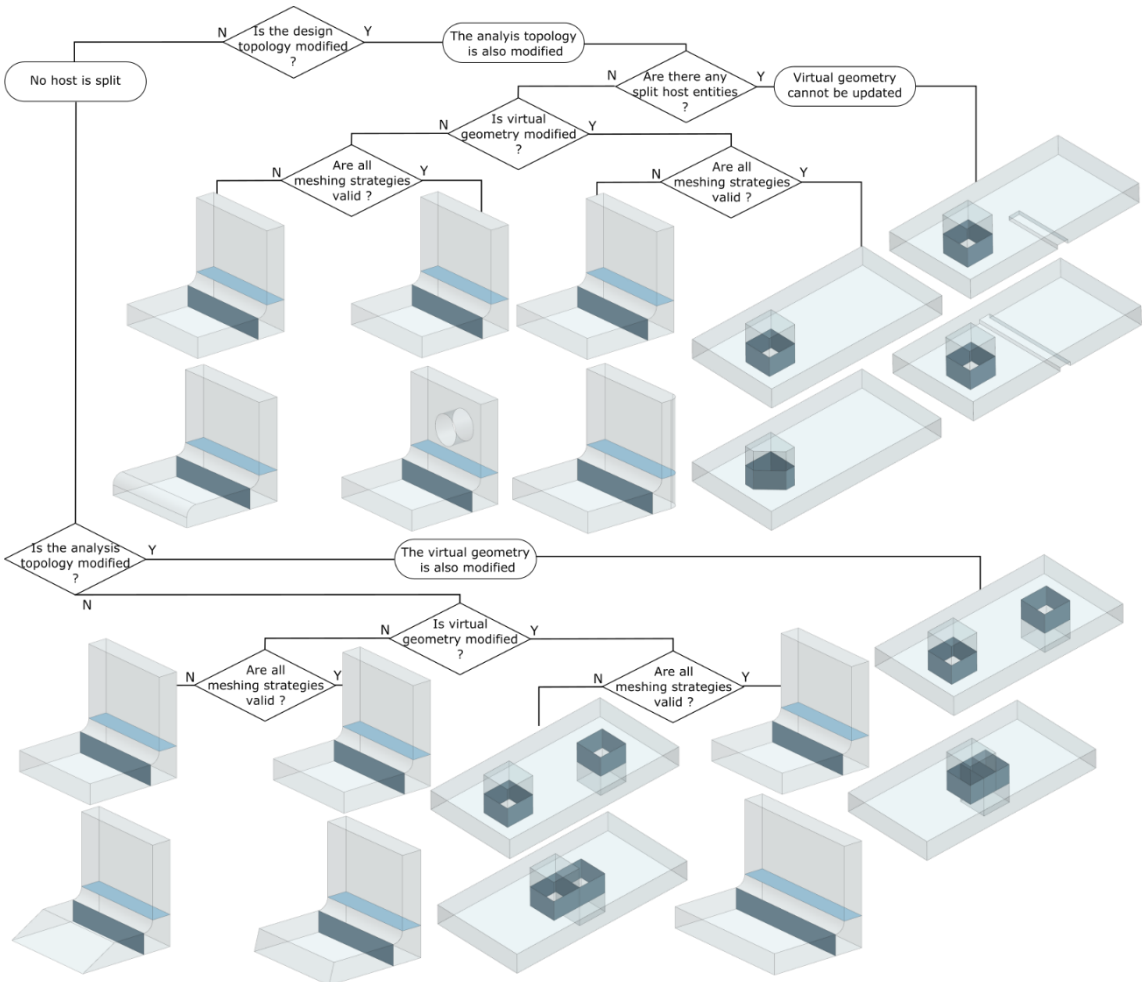

Figure 6. Analysis topology before and after design modification for various configurations.

Figure 6 shows procedures used in this work to determine the classification for the various geometric and topological configurations for decomposition update that can arise upon design modification (virtual faces shown in dark grey for visualization). This structure has been determined to be the most suitable for the mesh types being used in this paper, however a different structure or ordering may be better suited to different models, or the requirements of different analysts. Although only design changes involving geometrical modifications are used to illustrate the workflow, topological only modifications are handled in the same way. Some configurations are easy to update, e.g. for a purely geometric update where both the boundary topology and virtual geometry have not changed. However, the top right configuration is very challenging to update, as the bottom host face on which the boss edges were projected has become two unconnected faces due to the extension of the pocket. This is related to the persistent naming problem [28], where parametric modifications trigger topology changes that modify the underlying geometry.

The workflow in Figure 9 describes the method used to identify the aforementioned design changes and to update a virtually decomposed model after such design changes. Topological and geometrical modifications are identified from the CAD model. After the design modifications have been identified at the design topology level, analysis model modifications need to be identified. This is done by checking if the virtual decomposition history can be mapped on the new design, by checking if all the virtual splitting entities still lie within their hosts. Mapping constraints inferred from hex-meshing strategies are checked to ensure they are still valid and can inform the update of projected virtual geometry. Finally, all the candidate bodies for re-meshing are identified.

4.1 Tracking parametric and feature modifications

The CDS contains a representation of both the analysis topology and the original design topology independently from the CAD environment and also stores the virtual topology relationships required to transform one into the other. The original topology in the CDS is used to identify and classify both geometric and topological modifications after the CAD model has been updated. The VT relationships provide the link to map the changes in design to the analysis topology.

This section will describe how changes to the model in the CAD environment are propagated to the original topology in the CDS and then to the analysis topology describing the decomposition. The key point is that all entities in the original topology and analysis topology are linked to those in the CAD/CAE environment through two different attributes:

1) Name attributes attached to entities in the CAD environment. Any unique identifier offered by the CAD system (name, tag, color ...) can be used, provided that is can be assigned to an entity, queried and will persist between different modelling sessions.

2) Geometric attributes defining unique geometric identifiers of entities in the CAD environment, e.g. the center point of the edges, as well as the coordinates of its end vertices.

Both attributes are necessary, as structured interrogation of them allows the geometric and topological modifications to the design to be determined as outlined in the following sections. Once modified entities have been identified, each entity is mapped to an entity in the analysis topology through a series of VT relationships and topological queries. This enables the modifications to be identified and the entities of the analysis topology to be classified.

This classification is done from lower dimension entities to higher dimension ones, since any modification on the boundaries of an entity will propagate to the entity, while an entity can be modified without having its boundaries modified. While some CAD packages offer the ability to attach name attributes to vertices, other packages have not implemented this capability. Coordinates used as geometric attributes are not enough to classify vertices in the absence of name attribute, as a design change can move a vertex to the location of a different vertex that is also modified. However, the matching of the geometric attribute for edges includes checking the coordinates of both the mid-point and the bounding vertices. Therefore, the edge classification is based on the vertex classification, but not only as the mid-point factors as well. In this implementation, edges are classified first, so that vertex classification can be guided by the bounded edges classification.

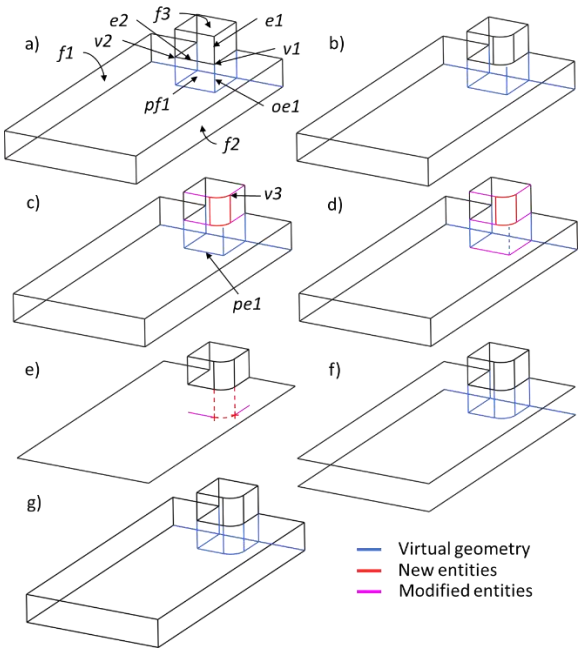

Figure 7. a) Original decomposition, b) the decomposition is not updated after a fillet is added, c) original entities classification, d) analysis entities classification, e) open design loops are closed and new virtual entities are identified, f) open analysis loops are closed and g) updated decomposition.

4.1.1 Edge classification

After parametric or feature modification, all the original edge entities in the CAD model are compared to the ones stored in the CDS and classified as follows:

- Original edge: If both the name and geometric attributes match, then it is assumed that the edge has not been modified (black edges Figure 7(c)).
- Modified edges: if the name is matched but not the geometric attributes, it is assumed that the edge has moved and is modified (purple edges in Figure 7(c)).
- New edges: Edge from the CAD whose name is not matched in the CDS is assumed to be a new edge (red edges in Figure 7(c)).
- Deprecated edges: Edges from the design topology in the CDS not matched to entities in the updated CAD model are assumed to be deprecated (edge *e1* in Figure 7(a)).

After adding the fillet, the two faces bounded by *e1* are bounded by two new edges, but some CAD systems will create just one new edge and reuse *e1* as the name attribute for the other. In previous workflows, this would prevent the automatic update of analysis models. Since *e1* was bounding two faces in the original model that are not connected in the updated model, *e1* must be classified as deprecated to remove any variability in the name attribute assignment.

New edges in the CAD model are grouped by the faces they bound, by querying the CAD model topology. This will be used later to simplify the identification of candidate for updating face topologies. Face tags that do not exist in the database help identify new CAD faces.

At this stage in the process, only the modifications to the design model have been identified. If only the geometric attributes of edges have changed, the geometry has changed and is updated as described in section 4.2. Otherwise, new or missing edges indicates that the original topology has changed. Further classification is required to identify modified, new and deprecated parasite entities in order to update the analysis topology.

Analysis topology modifications are inferred by their connectivity with original entities. For example in Figure 7, the parasite face *pf1* connected to the modified edge *e2* is classified as modified, and the mesh-mapping constraint on *pf1* indicates that the opposing edge *pe1* of the edge *e2* is also modified (purple in Figure 7(d)). This enables meshing constraints to be easily propagated to the updated topology.

4.1.2 Vertex classification

To accommodate the absence of name identifier on vertices in the geometric kernel used in this work, the classification of vertices is helped by the classification of the edges they bound. Vertices are classified as follow:

- Original vertex: vertices that bound original edges, or modified edges if there are matching geometric attributes and topological connectivity (vertex *v2* in Figure 7(a)).
- Modified vertex: vertex bounding modified edges with consistent topological connectivity.
- New vertex: vertices bounding a new edge that are not original or modified vertices (vertex *v3* in Figure 7(c)).
- Deprecated vertex: vertices bounding deprecated edges or modified edges (vertex *v1* in Figure 7(a)).

For each deprecated original edge, the list of edges connected to its bounding vertices is queried from the CDS. If these edges still exist and share the vertex in the new CAD design, the vertex still exists, otherwise it is deprecated. In this last case, a new vertex is created for each connected edge, and connected edges are stored along with their relationship to the old and new vertex to update the design topology at a later stage.

Analysis topology edges connected to the vertex are also checked and their relationship is stored. In the case where the vertex was bounding a parasite or orphan edge aligned with the sweep direction of a region, the eventual parasite or orphan edge connected is identified as deprecated. Its end vertices are stored to attempt to identify any new parasite edges. This is because these edges link the source and target faces of swept regions, and topological modification on one face can help identify analysis topology modification on the other face. For example, the orphan edge *oe1* in Figure 7(a) is classified as deprecated since the vertex *v1* is deprecated, hence the parasite edges lying on face *f2* are classified as modified.

4.1.3 Face classification

Topologically modified faces in both the original design and analysis topology are identified based on their bounding entities, since their geometric definition is more expensive to query. Identification of the geometric modifications of faces is kept for a later stage of the process, as it will be used to assess the validity of meshing strategies. Faces are classified as follows:

- Original topology faces: Faces with all their bounding edges classified as original or modified.
- Modified topology faces: Faces bounded by deprecated edges or disconnected modified edges (face *f3* in Figure 7(a)). Hence the boundary definition is incomplete in the CDS and will need to be updated.
- New faces: Face with a new name attribute, bounded by new edges. These are identified after updating the original topology in the CDS.
- Deprecated faces: Faces with less than two edges not deprecated.

Faces with an incomplete boundary definition are marked as open loops and stored as a sequence of edges, with all vertices bounding only one edge marked as open ends. Open loops appear when an original or a virtual edge is deprecated. Two modified edges bounded by a common deprecated vertex also identify an open loop, and the vertex is stored as a double open-end until it is replaced by the new vertices generated at each end of the modified edges. Furthermore, a face that has all its bounding edges deprecated is also considered deprecated. However, new parasite faces cannot be identified at this stage since the new parasite edges will

be identified when updating the design topology (see section 4.2).

New faces bounded by an existing or modified edge indicates that the face has been subdivided as a result of the design modification, similar to the persistent naming problem described previously (top right case in Figure 6). By looking at the classification of the edges that were connected to the edge that is matched, the other sub-faces can be identified, and split edges are re-classified to account for the fact that their bounding entities have been modified. Faces that are merged as a result of parametric modification are processed like other topology-modified faces, and the boundary entities of the deprecated face are used as candidates to complete the boundary definition of the modified one. The example in Figure 8 features both split and merged configurations where *f1* and *f3* are merged while *f2* becomes two faces.

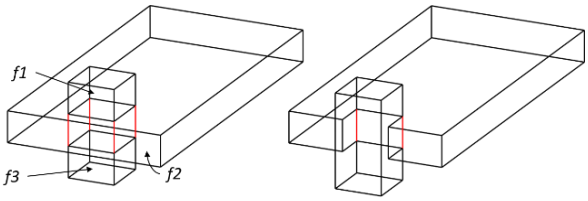

Figure 8. Face definition can be split or merged by parametric modifications.

4.1.4 Body classification

Original body classification is derived from the classification of its bounding entities. Analysis body classification is based on the virtual topology relationship along with original bodies and meshing strategies checks described in section 4.4. They are classified as follow:

- Original body: Bodies with all their faces, edges and vertices classified as original.
- Modified body: Analysis bodies bounded by modified entities that can be re-meshed.
- Modified topology body: body with deprecated faces
- Invalid body: Modified body with an invalid meshing strategy.
- New body: New original body with a name attribute not referenced in the CDS.
- Deprecated body: Bodies with all their bounding entities removed.

Modified volume subsets in the analysis topology can be preliminarily identified from the open loop faces in their boundaries. Further identification is done during the update of the analysis topology. Invalid bodies describe a set of modified bodies for which the meshing strategy has become invalid and is identified at a later stage (see section 4.4). These bodies are hence deprecated in the decomposition and will be merged to roll back the decomposition, enabling a local decomposition update to be performed if necessary.

4.2 Original topology update

Once all the entities in the original topology have been classified, they can be updated in the CDS to match the new design contained in the CAD model, according to Figure 9.

Deprecated and new entities in the CAD model indicate that the design topology has been modified. In that case, all the deprecated entities are removed from the CDS. Deprecated end vertices are matched against existing vertices, and a new vertex entity is created if no valid vertex is found. Bounding/bounded relationships and relative orientations are updated in the original topology relation. New edges and their vertices that are not matched by any existing vertex are added in the entity relation, and the bounding/bounded relation between edges and vertices is added to both topology relations.

In the case where all the entities are matched, without any deprecated or new entities, only the geometry has changed. The design topology can be updated by simply updating the geometrical attributes in the CDS. Mid-points of the modified original edges are updated in the entity relation of the CDS, along with the coordinates of modified vertices. Updating the topology before the geometry attributes enables new entities to be sorted and avoids having the same geometric attribute for multiple entities.

At this point, only the edges and vertices of the design topology are updated. Edges and vertices of the analysis topology need to be updated (in the next section) before updating the original face topology, since analysis topology faces are updated simultaneously with the original faces. The reason is that some new virtual edges can be identified when closing loop of edges on the face (see section 4.3.2).

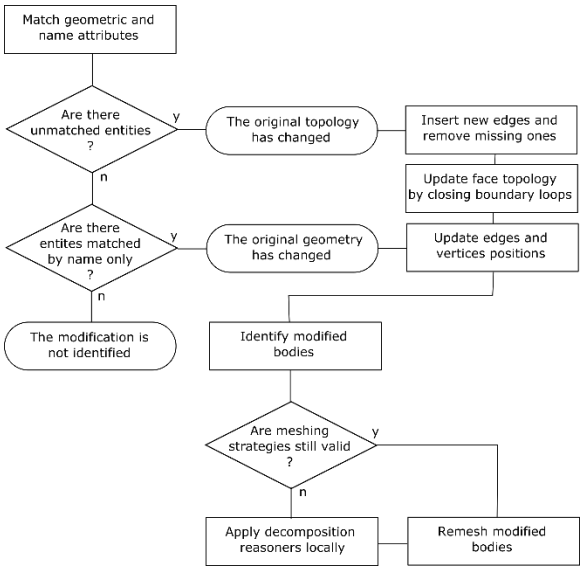

Figure 9. Geometry and topology update workflow.

4.3 Virtual geometry and analysis topology update

The next step after the design modifications have been identified and classified is to update the virtual geometry, and the topological connectivity of modified entities in the analysis topology, within the CDS. Deprecated virtual geometry vertices and edges are removed from the topological description.

4.3.1 Virtual geometry update

Modified original edges in the analysis topology have already been updated when updating the design topology. Here, the virtual geometry subset and parasite edges are updated by querying the entities that have been modified and propagating the changes to the host entities on which they rely.

In situations where parasite edges have been modified their virtual geometry needs to be updated. If a parasite edge is bounded by a vertex in the original topology, then a one-way projection is sufficient to capture the update virtual geometry. End vertices which were projected to create parasite edges need to be re-projected, e.g. in Figure 10(a), vertices A and B are created by projecting existing vertices to the original face *f1* and edge *e1* respectively.

A more complicated scenario arises when a modified parasite/orphan edge is bounded by two parasite vertices. In Figure 10(b), which corresponds to the model in Figure 14, the vertices C and D are the result of the reciprocal projection between the original edges *e2* and *e3*, hence a two-way projection is required. Point containment and angles are used to check whether the projection has succeeded, and the pair of vertices are stored to identify new potential parasite or orphan entities. If the projection has failed, parasite faces bounding the parasite edges bounded by the vertices are classified as deprecated, and the connected bodies become invalid.

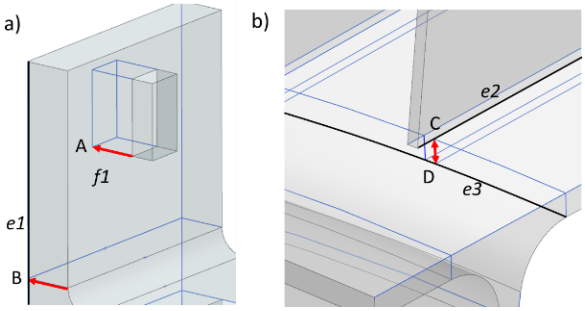

Figure 10. Projection of parasite vertices.

All the parasite vertices lying on modified edges and faces are also re-projected in order to update their coordinates. This checks the validity of subset edges and faces. Virtual geometry is used to assess the point containment of parasite vertices splitting parasite edges. Parasite face validity is inferred from the validity of their boundary entities.

4.3.2 Modified face topology update

At this stage, all the necessary information is available to update the topology of faces with an incomplete boundary definition or open loops. Design topology and analysis topology faces are updated at the same time, as updating the design topology to match the CAD topology will guide the update of the analysis topology. Suitable candidates to close open loops in the design topology are found in the list of new boundary edges in the updated design. Candidates for the original and parasite loops of the analysis topology include new boundary edges and new virtual geometry edges. Subset faces are updated using the relationships previously identified between edges and their bounded faces.

Gaps in open loops are filled iteratively by adding candidate edges at the open-end vertices and updating the open end until another open end is found. The process terminates when there are no more open ends, and all the edges belong to closed loops. Since this method can handle several disconnected gaps in the boundary definition of the loop, both inner and outer loops can be processed. In particular, the outer loop can absorb a previous inner loop if a feature on the face is moved to its boundary (see Figure 16 for example, where the rod is moved to the boundary of the middle rib).

In the case of analysis topology, mapping information from the attached meshing strategy is used to identify matching loops between source and target faces, and identify what entities can be projected to complete the opposite loops. This highlights the importance of the traceability between design topology, analysis topology and meshing attributes. For example, if a new edge is added in the loop of a source face for a thin-sheet region, this edge can be projected to identify a new parasite edge on the target face (Figure 7(e)). If the projection succeeds, then a parasite face is created, otherwise the thin-sheet meshing strategy needs to be reassessed.

Once all the open loops have been closed (Figure 7(f)), leftover virtual geometry entities are traversed to identify the smallest loops and infer new parasite faces. The analysis bodies can then be updated, and a valid topological representation of the analysis model is obtained (Figure 7(g)).

4.4 Updating meshing strategies

Since the main objective of the decomposition is to identify meshing strategies for hex meshing, checks are implemented after the analysis topology update to make sure the model can still be meshed automatically. Angles are checked to make sure no skew elements will be introduced. Wall faces of swept regions, such as face *f3* in Figure 12(a), also need to remain mappable for the sweep mesh generation to be successful. The CDS contains the geometrical and topological information to perform these tests.

If the hex meshing strategy becomes invalid, the body is classified as residual and connected residual regions are merged into a single superset. Since virtual topology is used for the decomposition, merging is done by altering entity connectivity and orientation in the analysis topology contained in the CDS, as described in [8]. Merging retains

all the correct imprints from other neighbor cells that would be lost by undoing the split operations, hence the meshing attributes of the neighbor cells remain valid. In Figure 11(a), a plate with two opposite bosses is decomposed into a thin-sheet and two general sweepable regions. After parametric modifications, the two blocks are merged into a single residual region while the thin-sheet strategy of the plate remains valid, Figure 11(b).

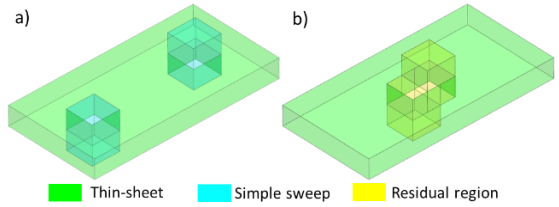

Figure 11: 2 volume cells are merged into one residual region after parametric perturbation.

After a valid analysis topology has been recovered, reasoning tools for decomposition are 'locally' used to recover deprecated meshing strategies or process new residual regions. In the case where a region had a previous hex-meshing strategy that has been identified as invalid, the reasoner related to this particular meshing strategy is used first. This also enables to re-use information provided by tools external to the framework, such as face-pair information. In Figure 12(a-b), a plate with a boss is decomposed into a thin-sheet and a general sweepable region. Adding a fillet to a bounding edge of the source face of the thin sheet makes the sweeping strategy invalid as *f3* is not mappable anymore, Figure 12 (c). The face pair *f1-f2* initially used to identify the thin-sheet region is re-used to identify a thin-sheet and a new residual region, Figure 12 (d). The long-slender reasoner is automatically applied to classify this residual as sweepable, and the model is once again fully hex-meshable Figure 12 (e-f).

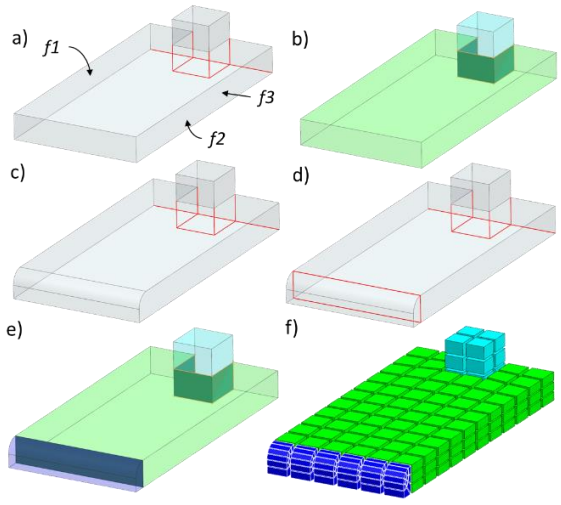

Figure 12. Thin-sheet strategy made invalid by a fillet insertion is recovered, and a new residual is processed to recover a fully hex-meshed model.

Alternatively, residual regions may become eligible for a hex-meshing strategy after a design change and can be re-assessed. In the case where several invalid bodies have been grouped in a single residual superset, the sequence of reasoners used to decompose the original design model can be re-used on the updated design. This ability to localize changes reduces the rework required to generate a valid analysis decomposition.

5. UPDATING THE MESH

Since all the modification have been localized, the mesh can be updated only where necessary, thus significantly reducing the expense of re-meshing. The original mesh was created from a decomposed model; hence every volume cell in the analysis topology is linked to a collection of mesh elements. Therefore, only the elements associated to a modified volume cells need to be altered. This collection of elements is referred to as a mesh collector. Upon first generation, all the interface meshes between sub-regions are stored in a separate mesh file, allowing interface information to be maintained when locally modifying the mesh. Management of interfaces in this manner also provides the ability to parallelize the meshing and re-meshing processes. Figure 13 shows an overview of the re-meshing process. Only the modified sub-regions are exported to the meshing environment, and a meshing recipe is automatically identified, taking into consideration constraints from the neighboring meshes that are not modified. Finally, the deprecated elements are replaced by the new ones directly in the input deck file of the solver, hence there is no need to load the whole mesh.

5.1 Meshing recipe update

The meshing recipe is updated to inform the re-meshing process of the modified regions. In simple cases with small deformations, the same meshing recipe can be reused. However, large modifications require the meshing division numbers and constraints to be adapted to the new geometry by locally updating the meshing recipe. To ensure compatibility with the rest of the mesh, the original recipe is interrogated to extract the controls at the interfaces with the modified regions. A new integer programming problem is created, using the number of elements division on interface curves as fixed constraints. Meshing constraints directing the flow of elements are recovered from the CDS and new constraints are added. The problem is solved using LPSolve [24] as before, and the resulting division numbers are used to update the meshing recipe. If no feasible solution can be found then the constraints are relaxed where possible, or a larger portion of the model needs to be re-meshed. Alternatively, further decomposition can be carried out on the modified volume cells in order to create the necessary transition zones to remove over constraints.

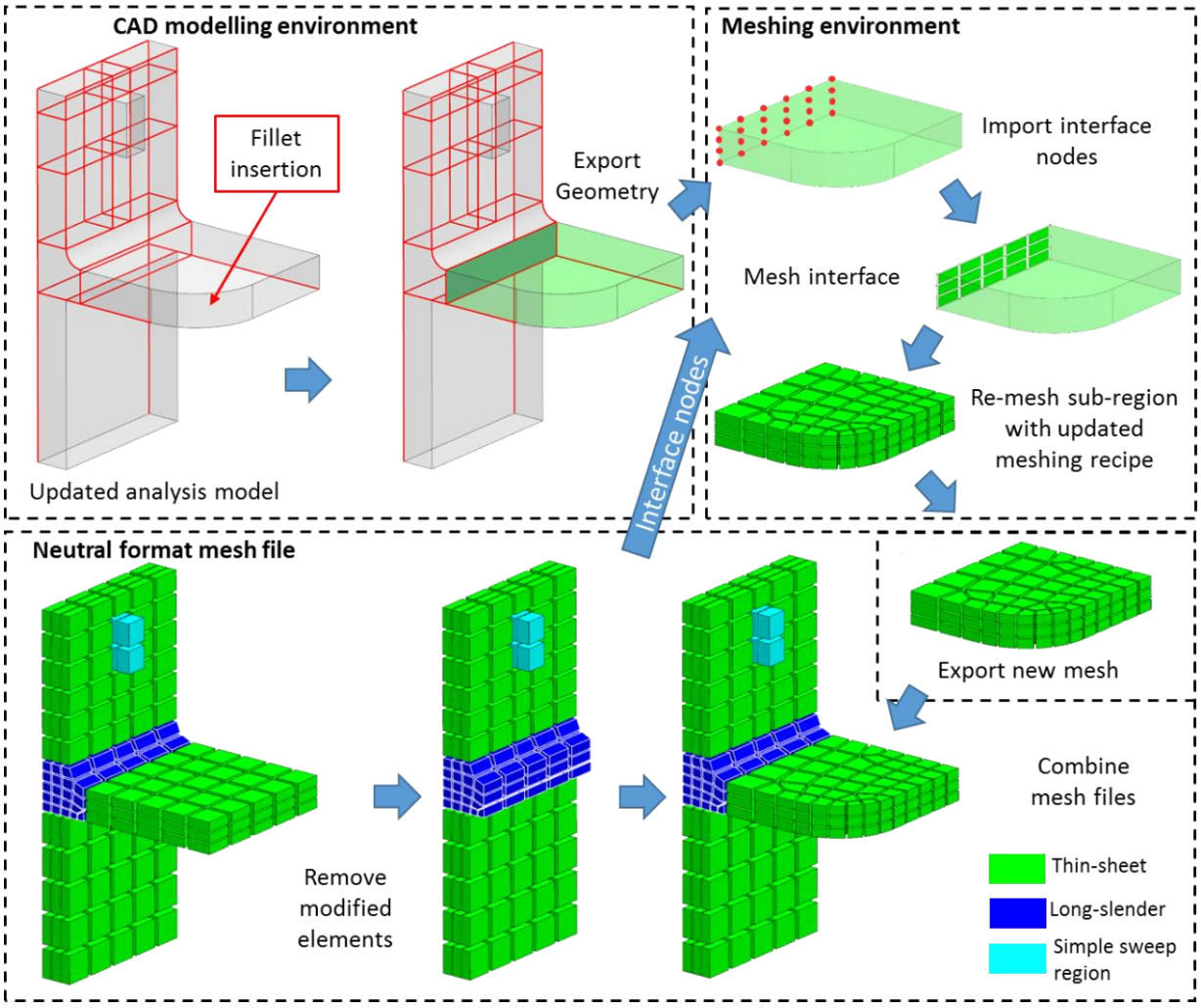

Figure 13. Local re-meshing workflow. Only the modified sub-regions are extracted from the virtual decomposition and re-meshed in the meshing environment, after the meshing recipe is locally updated and the interface nodes are recovered. Then the new mesh is re-assembled in the main mesh file

5.2 Local body extraction

Efficient re-meshing of a modified region is achieved by transferring only the modified regions to the meshing environment. This implies that any sub region can be geometrically extracted from the virtual decomposition in the analysis without having to decompose the whole model. This is possible through the robust connections that exist between the analysis topology and the geometric definitions in the CAD and CAE environments. To achieve this, all the interface entities can be identified from the analysis topology relation in the CDS. These parasite faces are created, grouped into connected sets and sewed together to make them usable for CAD geometry split. The result can then be extracted and exported, before the split is undone and the cutting faces deleted, to keep the CAD design model unaltered. Geometric modifications are only required for the mesh generation. The residual region in Figure 14 (c) can be extracted without having to remove thin-sheet and long slender regions first, with all the appropriate imprints from neighboring regions.

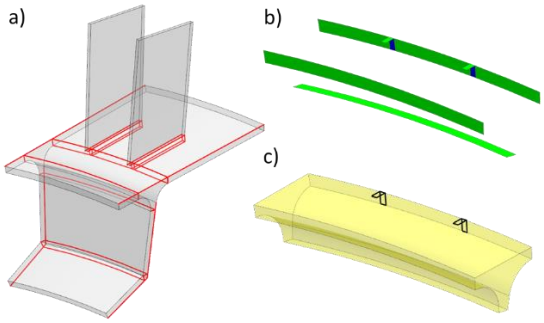

Figure 14. (a) virtually decomposed model, (b) parasite faces of the residual regions created and sewed, and (c) extracted residual with imprints.

5.3 Meshing sub-regions

All the nodes lying on the interfaces between the sub-region being re-meshed and the rest of the analysis model are recovered from the mesh files of the interfaces. These nodes

are imported into the meshing environment as frozen (locked in space) mesh points and associated with the corresponding face geometry in order to rebuild the surface mesh of the interface. The labels of interface nodes and elements are then updated to match the original mesh labelling. This ensures that the interface nodes will not be modified by the 3D meshing algorithm and that the new elements can be connected to the unmodified part of the analysis model mesh. If the re-meshing involves several connected bodies, all the new interface nodes generated will be added to the interface mesh file. Any deprecated nodes and elements are removed.

After the local mesh controls have been applied from the meshing recipe, the 3D elements are generated by the meshing reasoner described in section 2.2, and a mesh file containing all the new mesh for the concerned subsets is exported as a neutral mesh file. Nodes and elements indexing are automatically managed by setting the start index as the largest value in the current mesh file to ensure compatibility with the existing mesh and conformity at the interface.

5.4 Mesh manipulations

After the modified bodies have been re-meshed, the main mesh file needs to be updated. Since all the elements are grouped by sub-region into mesh collectors, the neutral mesh file can be re-written to include new nodes and elements. The updated mesh file is first created by copying the headers until the section containing the nodes is reached. Nodes are read and copied to the new file, removing deprecated nodes and updating the modified ones. All the new nodes are inserted at the end. Then mesh collectors of each of the bodies are transferred, with elements and their nodal connectivity replaced in the case where bodies are re-meshed. New collectors are copied from the mesh file containing new subset meshes, and finally the material properties section is updated and copied.

RESULTS AND DISCUSSION

The proposed method is run automatically on a large number of test models to ensure that the decomposition and meshing update can handle many configurations. Figure 15 shows the resulting meshes after a design change on a L-shaped bracket. Modifications to the L-shaped bracket include parametric changes by modifying wall thicknesses, and more advanced topological changes by adding fillets etc. to the model. The initial hex mesh in Figure 15 (a) is automatically updated to fully conforming hex meshes after all design modifications. In particular, thin-sheet decompositions are updated without having to interrogate the face pairs again, which is a costly part of the initial process.

The model in Figure 16 is taken from [21] in order to assess the performance of the method presented here on a complex design modification. The model is automatically decomposed in 12 seconds, and the meshing requires 16 seconds. After perturbation of the parameter ‘d’ in Figure 16(c), the analysis topology and virtual geometry are updated in 3.8 seconds, and the model is re-meshed in 11 seconds, recovering a full hex mesh. This corresponds to a 47% gain of time compared to running the automatic meshing procedure form the beginning.

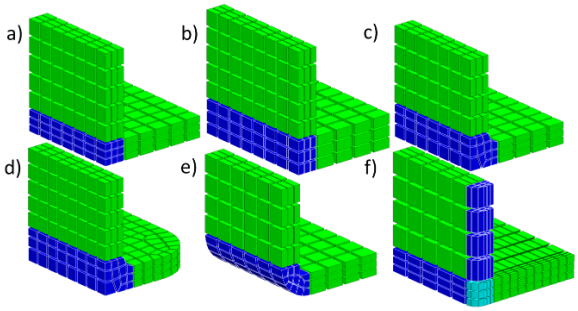

Figure 15. L-bracket re-meshed after various design modifications.

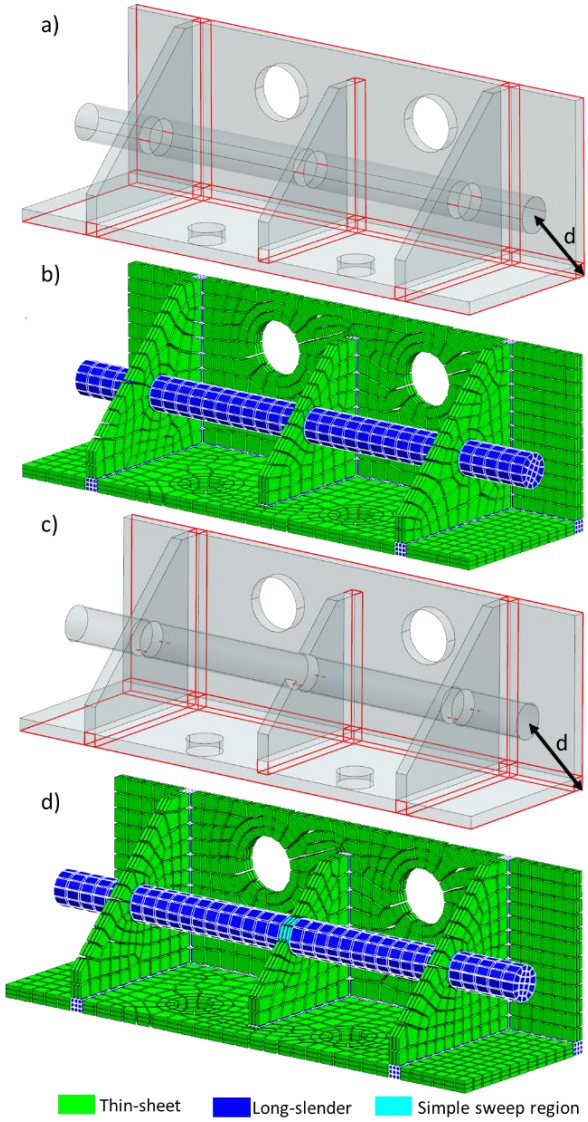

Figure 16. Re-meshing all hex model. (a) Virtually decomposed model, (b) all-hex mesh, (c) parameter d is increased, and (d) updated mesh.

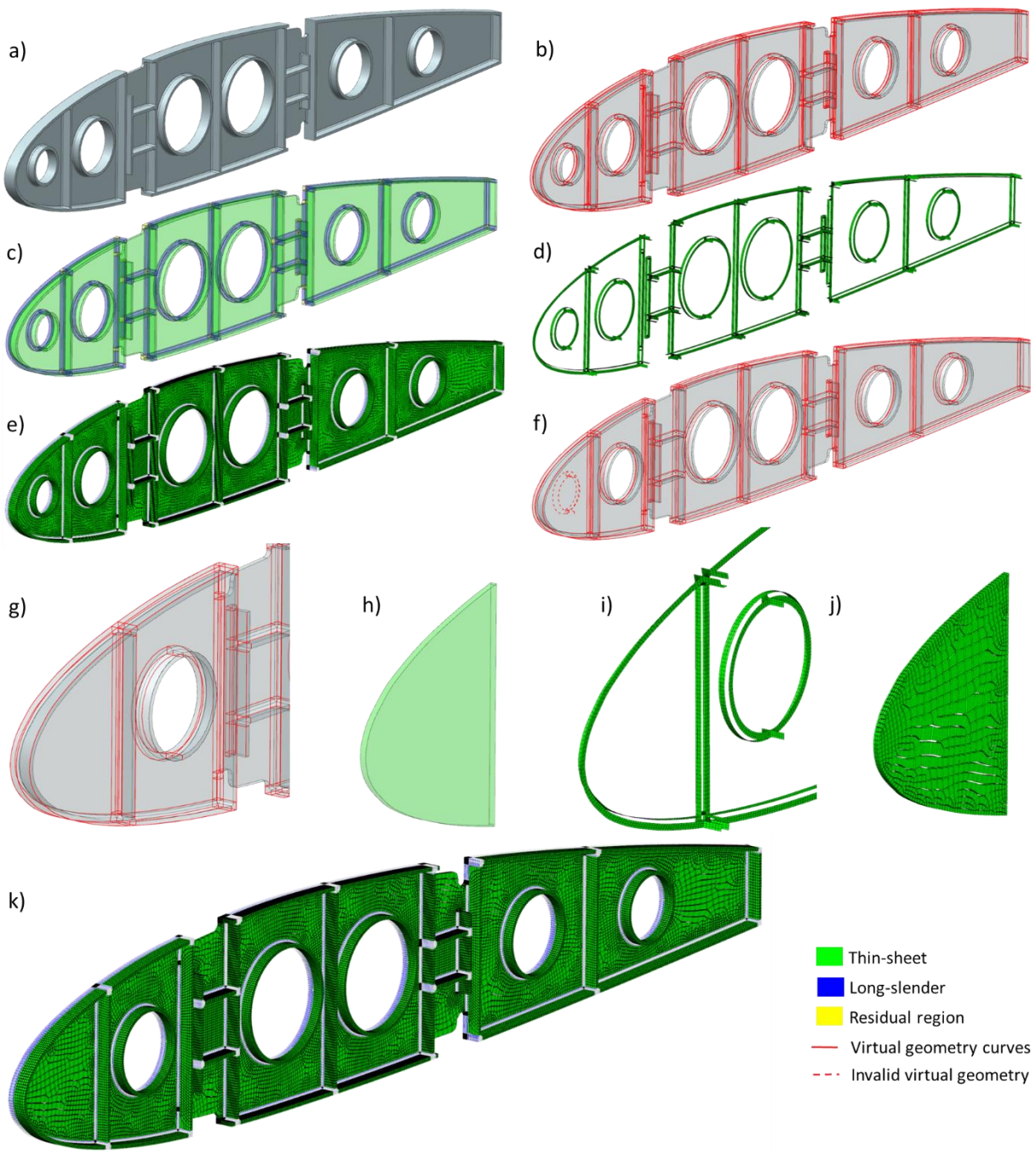

Figure 17. Automatic meshing and re-meshing of a rib model. (a) Original design, (b) decomposed analysis topology, (c) equivalent geometric decomposition, (d) interfaces are meshed first, (e) resulting mesh, (f) design modification with topological changes, (g) updated analysis topology, (h) modified cell extraction, (i) updated interfaces meshes and interfaces nodes import, (j) re-meshed cell and (k) updated mesh.

While the proposed method offers significant time reduction compared to a process involving manual decomposition and meshing for large models, the re-meshing process is not as beneficial for small analysis models, where a design modification can require re-meshing of most of the sub-regions.

An example of a larger model is given in Figure 17. Successive reasoners are used to decompose the model within 2 minutes (on a windows workstation with a 3.7 GHz Intel Xeon E5-1630 CPU with 32GB RAM) in Figure 17(b), and the geometric decomposition is generated in 1 minute, Figure 17(c). The mesh is automatically generated within 5 minutes in Figure 17(e), with 235,000 hexahedra and 78,000

tetrahedra. The interface mesh file (Figure 17(d)), contains 19,800 quad elements allowing individual manipulation of each of the 174 meshed sub-regions. After changing the design by removing a hole and its protrusion near the leading edge, the virtual decomposition is updated in 3.5 seconds (Figure 17(g)), and the model is locally re-meshed (Figure 17(h-k)) in 19.4 seconds. 7850 elements of the previous mesh are replaced by 2430 new hexes. The normal re-meshing in the CAE environment for this modification takes approximately 6 minutes, which is only for re-meshing. This does not include the manual time required to update the decomposition and link it to the meshed model.

A critical scenario for this approach arise when an original entity hosting many subsets is modified. In Figure 18, all the projections need to be updated to ensure they are still valid after perturbation of the bottom face; hence the modification propagate to a large portion of the analysis model. Although applying a front propagation technique that stops once the modified subsets have been updated is future work, the current implementation still outperforms any manual intervention.

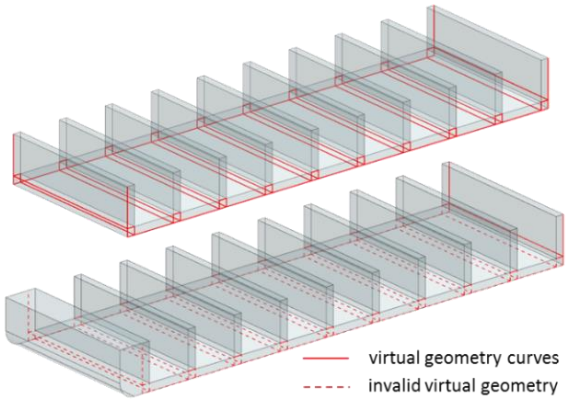

Figure 18. All the projection must be checked after the bottom face is modified.

In the event of rigid body motion, the whole decomposition is modified. In the absence of topological modification, the analysis model is easily updated. However, all bodies are modified, hence the model needs to be fully re-meshed in the current implementation. This could be easily handled by any mesh deformation algorithm in future work.

One important challenge when re-meshing is to ensure a good quality mesh will be re-generated, as moving features can create small faces that will drive down the size of elements and increase computation time. In Figure 19, virtual geometry is moved in close proximity to original entities. Detection of close entities is carried out using the medial axis while taking into account the virtual geometry, since the proximity is a result of the decomposition. This information will be used in future work to decide merging operations and define whether topological updates are required to the parasite topologies.

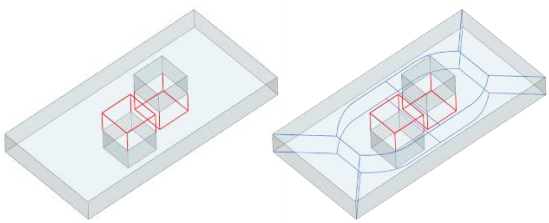

Figure 19. Medial axis (in blue) on thin sheet source and target faces to identify close entities.

Whilst the automated workflow in this paper has been demonstrated around the use of thin-sheet and long-slender decomposition reasoners, the same process is valid for any decomposition reasoner that introduces meshing strategy which creates mapping constraints between entities. For example, if a reasoner identifying regions suitable for transfinite meshing methods were applied, the mapping between opposite pairs of faces and edges would help to propagate the design modification. The reason is that meshing constraints are used to guide the update of the decomposition, and the proposed approach is capable of automatically updating these meshing attributes to match design modifications.

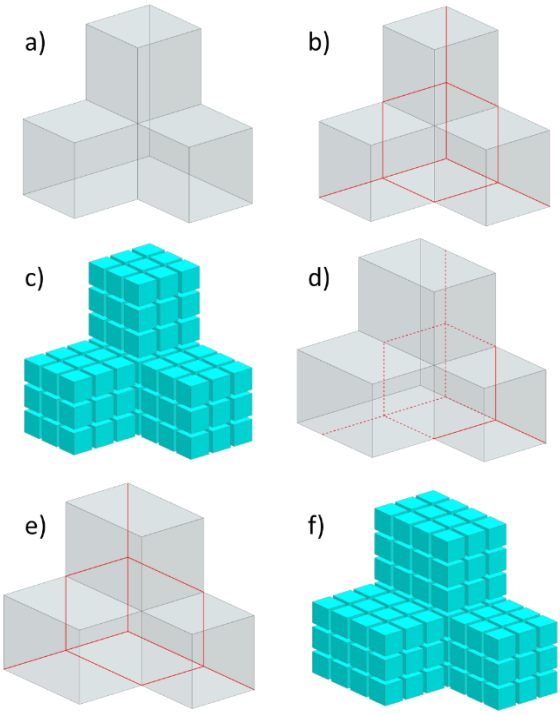

Figure 20. Manually defined block can be updated. (a) original model, (b) sweepable blocks defined manually, (c) automatic mesh, (d) parametric modification, (e) automatically updated decomposition and (f) updated mesh.

Virtual block decomposition defined manually can also be updated if meshing strategies have also been defined by the user (Figure 20). Limiting the update to only those blocks that have been modified provides analysts with the assurance they will not have to perform a block decomposition for all

subsequent design changes. This enables an analyst to focus only on the regions of interest that have been modified. This is important in the context of large analysis models where a manual block decomposition could be performed on a modified region with all interfaces, meshing constraints and re-meshing automatically handled by the work presented here.

The identification of modified regions relies on the postulate that all the splitting entities connectivity can be traced back to original entities, hence they can be identified by manipulating the topological graphs of the models and the virtual topology relation. If the modifications cannot be processed with the method described here, the model can still be automatically meshed using the virtual topology workflow presented from the beginning. The user-defined parameters such as the sequence of the reasoners and the target aspect ratios are automatically recovered and re-used. However, if a bump was to appear on a face without moving any of the points sampled for the identification of geometrical modification nor changing the topology, the current method would be unable to identify this modification and no modification would be made to match the new design. Alternative shape descriptors could be used to identify these localized geometric updates and feed this information into the workflow.

CONCLUSION

A virtual topology workflow has been extended to handle design changes in automatic meshing workflows, by using the cellular description of the mesh to manipulate and update it. Significant time reduction is achieved by automatically controlling the re-meshing process, especially since only parts of the mesh are loaded and updated. This approach has achieved the following objectives:

- The model can be re-meshed after large parametric modifications or feature changes, and all design modifications are propagated to the analysis model.
- The mesh structure is maintained as much as possible by taking into account hexahedral meshing constraints.
- The mesh update process is fully automated and more efficient than traditional approaches.

FUTURE WORK

The presented method is limited by the variety of reasoners applied in the work, and further work is required to improve and validate the proposed approach. This includes:

- Implementing more decomposition reasoners. Only reasoners that identifies region hex-meshable with a one-to-one sweeping method have been used. Reasoners dedicated to the identification of mappable cube-like region can be used within the proposed method, as the meshing constraints are similar and can be used to guide the update of the decomposition.
- Many-to-many sweeping is currently not handled since the mesh reasoner is limited to simple one-to-one sweeps. Many-to-many sweeping introduces different constraints on entities than mapping or sweeping techniques, and therefore will require additional reasoning to help the update of the decomposition.
- The decomposition reasoners used provide good results on thin models, but introducing different blocking tools could enhance further the method. Special care will be required if singularity lines are introduced, since the mapping of entities through meshing constraints will become more difficult.

ACKNOWLEDGMENTS

The authors wish to acknowledge the financial support provided by Innovate UK via GEMinIDS (project 113088), a UK Centre for Aerodynamics project. The authors acknowledge Rolls-Royce for granting 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] T. Tautges, T. Blacker, and S. Mitchell, "The whisker weaving algorithm: A connectivity-based method for constructing all-hexahedral finite element meshes," *Int. J. Numer. Methods Eng.*, vol. 39, no. 19, pp. 3327–3349, 1996.

[3] T. Blacker and R. Meyers, "Seams and wedges in plastering: a 3-D hexahedral mesh generation algorithm," *Eng. Comput.*, vol. 9, no. 2, pp. 83–93, 1993.

[4] G. F. Carey, "Hexing the Tet," *Commun. Numer. Methods Eng.*, vol. 18, no. 3, pp. 223–227, Jan. 2002.

[5] 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.

[6] E. Ruiz-Gironés and J. Sarrate, "Generation of structured meshes in multiply connected surfaces using submapping," *Adv. Eng. Softw.*, vol. 41, no. 2, pp. 379–387, Feb. 2010.

[7] D. R. White and T. J. Tautges, "Automatic scheme selection for toolkit hex meshing," *Int. J. Numer. Methods Eng.*, vol. 49, pp. 127–144, 2000.

[8] 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.

[9] 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.

[10] D. C. Nolan, C. M. Tierney, C. G. Armstrong, and T. T. Robinson, “Defining Simulation Intent,” *Comput. Des.*, vol. 59, pp. 50–63, 2015.

[11] 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. 2003.

[12] R. Bidarrat, K. Jan de Kraker, and W. F. Bronsvoort, “Representation and management of feature information in a cellular model,” *Comput. Des.*, vol. 30, no. 4, pp. 301–313, 1998.

[13] C. M. Tierney, “Managing equivalent representations of design and analysis models,” Queen’s University Belfast, 2014.

[14] G. P. Gujarathi and Y.-S. Ma, “Parametric CAD/CAE integration using a common data model,” *J. Manuf. Syst.*, vol. 30, pp. 118–132, 2011.

[15] M. L. Staten, S. J. Owen, S. M. Shontz, A. G. Salinger, and T. S. Coffey, “A Comparison of Mesh Morphing Methods for 3D Shape Optimization,” in *Proceedings of the 20th International Meshing Roundtable*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 293–311.

[16] P. Knupp, “Applications of mesh smoothing: copy, morph, and sweep on unstructured quadrilateral meshes,” *Int. J. Numer. Methods Eng.*, vol. 45, no. 1, pp. 37–45, 1999.

[17] A. Sheffer and A. Üngör, “Efficient Adaptive Meshing of Parametric Models,” *J. Comput. Inf. Sci. Eng.*, vol. 1, no. 4, p. 366, Dec. 2001.

[18] C. Shen, R. Wang, S. Gao, and H. Maehama, “An approach to feature moving of hexahedral mesh,” *Comput. Des.*, vol. 107, pp. 12–22, Feb. 2019.

[19] H. A. Van Der Meiden and W. F. Bronsvoort, “Tracking topological changes in parametric models,” *Comput. Aided Geom. Des.*, vol. 27, pp. 281–293, 2010.

[20] L. Sun, T. T. Robinson, C. G. Armstrong, S. Marques, and W. Yao, “Surface Mesh Deformation in CAD-based Shape Optimization,” in *AIAA Scitech 2019 Forum*, 2019, p. 2360.

[21] M. Sypkens Smit and W. F. Bronsvoort, “Efficient tetrahedral remeshing of feature models for finite element analysis,” *Eng. Comput.*, vol. 25, no. 4, pp. 327–344, Nov. 2009.

[22] C. M. Tierney, D. C. Nolan, T. T. Robinson, and C. G. Armstrong, “Managing Equivalent Representations of Design and Analysis Models,” *Comput. Aided. Des. Appl.*, vol. 11, no. 2, pp. 193–205, Mar. 2014.

[23] C. Tierney, F. Boussuge, and D. C. Nolan, “Tolerant geometric extraction of fluid domains to assist CFD analyses of aero-engines,” in *AIAA Scitech 2019 Forum*, 2019, p. 1720.

[24] “LPSolve Mixed Integer Linear Programming.” [Online]. Available: http://lpsolve.sourceforge.net/5.5/. [Accessed: 25-Mar-2019].

[25] 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.

[26] 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.*, pp. 1–17, Nov. 2017.

[27] F. Boussuge, C. M. Tierney, T. T. Robinson, and C. G. Armstrong, “Symmetry-based decomposition for meshing quasi-axisymmetric components,” *Procedia Eng.*, vol. 203, pp. 375–387, 2017.

[28] 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.