Titre: Title:	Détection, visualisation et communication de défauts dans les modèles géométriques et les maillages
Auteur: Author:	François Petit
Date:	2013
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Petit, F. (2013). Détection, visualisation et communication de défauts dans les modèles géométriques et les maillages [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1085/

UNIVERSITÉ DE MONTRÉAL

DÉTECTION, VISUALISATION ET COMMUNICATION DE DÉFAUTS DANS LES MODÈLES GÉOMÉTRIQUES ET LES MAILLAGES

FRANÇOIS PETIT
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AVRIL 2013

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

DÉTECTION, VISUALISATION ET COMMUNICATION DE DÉFAUTS DANS LES MODÈLES GÉOMÉTRIQUES ET LES MAILLAGES

présenté par : PETIT François
en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury d'examen constitué de :

Mme CHERIET Farida, Ph.D., présidente
M. GUIBAULT François, Ph.D., membre et directeur de recherche
M. LABBÉ Paul, Ph.D., membre

À mes parents.....

REMERCIEMENTS

Avant de rentrer dans le vif du sujet, je souhaiterais remercier ici tous ceux qui m'ont permis de mener à bien ce projet de recherche, qui restera une expérience mémorable tout au long de ma vie.

Bien évidemment, je voudrais remercier mon directeur de recherche, Monsieur François Guibault, qui a non seulement rendu ce projet possible financièrement, mais qui m'a permis de découvrir le monde de la recherche scientifique grâce à ses conseils, son support et sa disponibilité permanents tout au long de ce travail.

Je voudrais aussi remercier mes collègues au laboratoire, et en particulier Ying Zhang et Julien Leray qui m'ont beaucoup aidé pour orienter mes travaux vers des outils utiles aux utilisateurs.

Je voudrais enfin remercier Madame Farida Cheriet et Monsieur Paul Labbé qui ont accepté de faire partie du jury lors de la présentation finale de mon travail.

RÉSUMÉ

Ce mémoire traite de problèmes rencontrés dans le cadre des premières étapes de la conception de pièces en lien avec la mécanique des fluides. En particulier, on s'attardera sur la méthodologie utilisée afin de détecter et communiquer des erreurs que peuvent rencontrer les concepteurs de pièces mécaniques, et en particulier de composantes de turbines hydroélectriques, lors du design préliminaire et des premières simulations numériques.

En effet, lors des premières phases de design, il est souvent nécessaire d'obtenir un modèle mathématique complet de la pièce conçue afin de pouvoir évaluer son comportement final sans avoir à en réaliser un prototype. Ces modèles sont le plus souvent réalisés selon les normes de la représentation frontière (Boundary REPresentation, BREP) où les volumes ne sont définis que par les surfaces qui les délimitent.

Cette méthode de modélisation, associée aux courbes paramétriques (en général de type Non Uniform Rational B-Spline (NURBS)), permet de modéliser des géométries extrêmement complexes tout en utilisant des entités mathématiques (courbes, surfaces) assez simples. Cela permettra, par la suite, de réduire les temps de calcul nécessaires pour les simulations ou les rendus graphiques.

Cette simplification des entités utilisées entraîne par contre des approximations, notamment au niveau des jonctions entre entités de même dimension (par exemple, deux surfaces jointives peuvent ne pas se suivre exactement le long de l'arête commune). Afin d'assurer l'étanchéité des volumes, différentes solutions sont utilisées. Celle qui a été choisie au sein de la série de logiciels basés sur la librairie Pirate est l'utilisation d'une topologie afin de spécifier quelles sont les entités coïncidentes. Mais ces solutions peuvent présenter des erreurs, par exemple si deux entités déclarées correspondantes sont en réalité géométriquement éloignées. Ces erreurs, si non corrigées, se retrouvent généralement dans les maillages générés à partir de ces modèles BREP, et donc ont une influence sur la qualité des simulations qui peuvent être effectuées ensuite.

Le premier aspect de ce mémoire traite des moyens qui ont été mis en place au sein du logiciel TopoVisu afin de détecter et de visualiser les erreurs présentes au sein des modèles BREP et des maillages utilisés lors de l'étape de conception.

Le second aspect de ce mémoire traite d'une des origines principales des erreurs citées précédemment : la conservation et le transfert des données. C'est à ce niveau que de nombreuses erreurs apparaissent. En effet, le passage d'un format de fichiers à un autre peut s'avérer extrêmement néfaste pour un modèle, et cela peut poser problème si on change d'application, mais aussi si on passe à une autre version d'une même application. La solution proposée ici

est l'exportation native vers un format de données ayant un bon niveau de maintenance et d'utilisation qui permettra de favoriser sa pérennité. Le format qui a été choisi pour cette fonction est le Portable Document Format, avec inclusion de modèles en trois dimensions (PDF 3D). Les éléments 3D sont inclus dans le format Product Representation Compact (PRC), car ce format est compatible avec la norme ISO 10303 (Standard for the Exchange of Product model data, STEP) qui est au cœur de beaucoup de modélisations BREP, dont celle de la librairie Pirate.

ABSTRACT

This master thesis addresses some of the problems encountered during the first design stages of engineered object related to fluid mechanics. More precisely, we will focus on the problems that can be encountered in the early modeling and simulations of hydroelectric turbines.

During the first steps of design, a mathematical model of the engineered item is realized in order to evaluate its properties in simulations, and without building a prototype. The methodology used to build those models is generally based on the boundary representation (BREP) rules : Volumes are defined through the faces that delimit them; faces are defined through a geometric support (a surface) and through the edges that delimit them and so on.

BREP models generally use parametric curves and surfaces (and Non Uniform Rational B-Spline (NURBS) in most cases). This type of models allows a great complexity in geometries while keeping reasonnably simple mathematical entities, and thus reasonnably low computing time in simulations or graphic rendering.

But the simplification of models implied by BREP results in approximations, which lead to small gaps or overlaps near the border of the entities (e.g. between two faces of a shell). In order to solve those small errors, the Pirate library and its software suite use topology to identify which entities are connected. But this isn't sufficient, if two surfaces are declared connected, but are separated by a non negligible gap, errors can appear later in the design process, often with bad quality meshes and simulation results.

The beginning of this master thesis will treat the methods used to find and highlight the errors in BREP models and meshes, and particularly how they have been implemented in TopoVisu.

The second main aspect of this work is to treat the errors at their source. Most of the errors in BREP models and meshes are generated when data is transferred, either to another CAD application, or to a more recent version of the same software. The solution proposed in TopoVisu is to natively export data to a highly maintained, widespread file format, which has a good chance to stay viable in the future. The format we chose is the Portable Document Format (PDF) with 3D models included in Product Representation Compact (PRC) format. This choice is based upon the ISO-10303 standard (Standard for the Exchange of Product model data, STEP) compatibility of the PDF 3D format, this standard being at the heart of most CAD engines, including the Pirate library.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

LISTE DES SIGLES ET ABRÉVIATIONS

STEP	STandard for the Exchange of Product model data
IGES	Initial Graphics Exchange Specification
BREP	Boundary REPresentation
CSG	Constructive Solid Geometry
CAD	Computer Aided Design
NURBS	Non Uniform Rational B-Splines
CFD	Computational Fluid Dynamics
PDF	Portable Document Format
PRC	Product Representation Compact
PLM	Product Lifecycle Management
SPR	Small Polyhedron Reconnection
MAGNU	(Laboratoire de) MAillage et Géométrie NUmérique

CHAPITRE 1

INTRODUCTION

Dans l'industrie manufacturière, l'innovation passe en général par des améliorations successives de pièces. Ce processus itératif d'amélioration est appliqué depuis de nombreuses années, tout d'abord sous la forme de prototypes testés en laboratoire, puis de nos jours sous la forme de modèles numériques et de simulations.

Toutes les grandes entreprises aéronautiques, automobiles, hydroélectriques, etc. utilisent des logiciels de conception assistée par ordinateur (Computer Aided Design) afin de créer des versions numériques de leurs produits sur lesquels des simulations seront effectuées. Des prototypes matériels sont généralement utilisés à la suite des simulations, mais l'étape à laquelle s'intéresse ce mémoire est celle des modèles numériques et des simulations.

Dans un premier temps, les concepts importants au cours de ce travail vont être définis.

1.1 Définitions et concepts de base

Les modèles utilisés pour la conception de pièces sont en général d'un de deux types : de type modélisation solide par construction(Constructive Solid Geometry), ou de type représentation frontière (Boundary REPresentations).

1.1.1 Modèles CSG

Le premier type de représentation se base sur des formes simples (cubes, sphères, tores...) qui sont combinées à l'aide d'opérations booléennes (union, différence, intersection). Pour comprendre plus aisément le fonctionnement de ce type de modèles, on peut représenter ceux-ci par des arbres (voir la figure 1.1) où chaque feuille est une des formes élémentaires, chaque nœud est une opération booléenne, et la racine de l'arbre est la pièce finale.

1.1.2 Modèles BREP

Dans le cadre plus précis de la mécanique, les modèles sont souvent de type représentations frontière. Ces modèles se basent sur un principe simple : chaque entité est définie à partir de ses bordures. En réalité, chaque entité topologique est définie par une entité géométrique déterminant sa forme, et par les entités topologiques de dimension inférieure qui la délimite.

La plupart des implémentations des modèles BREP se basent sur la norme ISO-10303 (1994), qui a pour objectif de rendre possibles les échanges de données. Ces implémentations

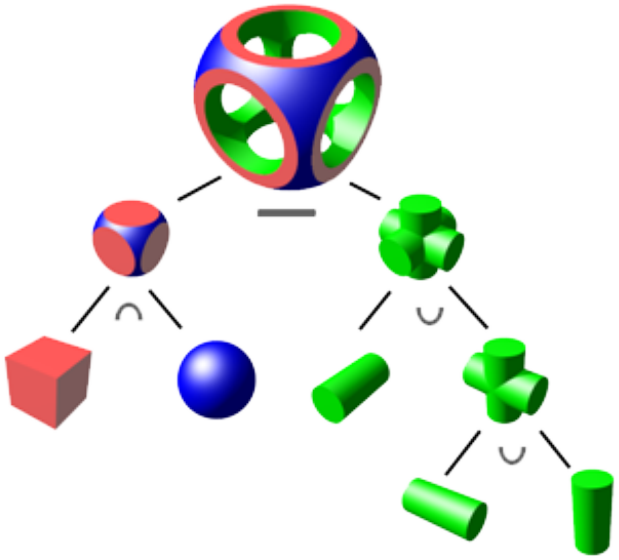

Figure 1.1 Exemple d'arbre de représentation d'une modélisation solide par construction (image tirée de Wikipedia (2013))

présentent des classes assez similaires, pouvant être réparties en deux catégories : les entités géométriques et les entités topologiques. Celles-ci sont décrites ci-dessous, dans le cas d'un modèle 3D.

Liste des entités géométriques :

Point

Entité de dimension 0, possédant 3 coordonnées.

Courbe

Entité de dimension 1, en général une courbe paramétrique (si l'on fournit une valeur de paramètre dans l'intervalle de définition, on obtient un point 3D).

Surface

Entité de dimension 2, en général une surface paramétrique (si l'on fournit un couple de paramètres dans le double intervalle de définition, on obtient un point 3D).

Espace

Entité de dimension 3, n'existe que par les entités topologiques qui le délimitent(voir la liste suivante).

Liste des entités topologiques :

Sommet
: Entité de dimension 0, associée à un ou plusieurs points, pour signaler leur unicité théorique.

Arête
: Entité de dimension 1, associée à une courbe définissant sa forme, et à deux sommets (placés sur la courbe) la délimitant.

Arête associée
: Entité délimitant une face et associée à une arête, afin de signaler leur correspondance théorique.

Boucle
: Entité encapsulant plusieurs arêtes associées formant un contour fermé (les sommets de début et de fin des arêtes doivent se correspondre).

Face
: Entité de dimension 2, associée à une surface définissant sa forme, et délimitée par une ou plusieurs boucles (les boucles suivantes décrivent des trous dans la surface).

Face associée
: Entité délimitant un volume et associée à une face, afin de signaler leur correspondance théorique.

Coquille
: Entité encapsulant plusieurs faces pour former une surface fermée (sans bords libres). C'est l'équivalent de la boucle, en trois dimensions.

Volume
: Entité de dimension 3, représentant l'espace délimité par ses coquilles (la première coquille délimite l'extérieur du volume, les suivantes forment des trous dans le volume).

Dans certaines implémentations, certaines de ces entités peuvent n'exister que sous forme de structures de données dans l'entité qui les encapsule (les boucles dans les faces et les coquilles dans le volume, principalement) ; mais la totalité des fonctionnalités que présentent ces entités est disponible dans chaque logiciel utilisant les modèles BREP.

Une autre distinction peut être effectuée au sein des modèles BREP. Les modèles peuvent être restreints ou non aux variétés, une variété de dimension n, où n désigne un nombre entier, étant un espace topologique localement euclidien. Initialement, seuls les modèles basés sur les variétés étaient intégrés dans les modèles BREP, c'est-à-dire les objets réels, ayant la propriété suivante : en tout point de la surface, on peut placer une sphère suffisamment petite qui sera

séparée en deux par cette surface, une partie à l'intérieur de l'objet, l'autre à l'extérieur. Par la suite, les objets non basés sur les variétés ont été ajoutés (objets surfaciques, ou comportant des faces internes par exemple), ce qui permet une plus grande diversité dans les modélisations. On pourra par exemple utiliser une face interne pour séparer la partie fluide et la partie solide d'une pièce en fusion, les phénomènes présents étant très différents au sein des deux milieux.

1.1.3 Courbes paramétriques

Les courbes (et surfaces) paramétriques (d'équation (x(t),y(t))) sont très utilisées dans ce contexte, car elles permettent de générer des formes de manière très libre, tout en gardant des fonctions d'assez bas degré (3 et 5, principalement), ce qui permet de conserver des temps de calcul relativement courts lors des simulations. Les différents paramètres permettent d'ajuster les formes, ce qui permet des raffinements de design, soit à la main, soit à l'aide d'algorithmes spécialisés. Cette automatisation possible de l'optimisation est un des gros avantages de ces modèles, cela permet d'accélérer le processus de conception en le rendant autonome.

Deux autres types de courbes sont parfois utilisées pour les représentations mathématiques de pièces : les courbes implicites (d'équation f(x,y) = 0) et les courbes explicites (d'équation y = f(x)). L'inconvénient majeur de la première catégorie est sa difficulté d'utilisation. Si les courbes implicites permettent de créer des formes aussi complexes que les courbes paramétriques, les paramètres ajustables qui interviennent dans l'expression des fonctions qui les définissent ont une influence beaucoup moins claire et directe que les constantes des courbes paramétriques. Les courbes explicites sont très peu utilisées, car elles limitent beaucoup les possibilités de formes, et peuvent être transformées en courbes paramétriques facilement (y = f(x) devient (x,y) = (t,f(t))).

Au sein des courbes paramétriques, les courbes les plus utilisées sont les NURBS (Non Uniform Rational B-Splines). Ceci est dû à leur propriété de contrôle local : les paramètres qui définissent une courbe ont une faible zone d'influence, et ainsi, lorsqu'on modifie un de ces paramètres, on n'a pas besoin de recalculer la totalité du modèle, uniquement la partie qui est affectée par la modification. Cet avantage est une des raisons pour lesquelles les NURBS sont choisies dans la plupart des modélisations BREP (et dans la norme STEP en particulier). Une autre raison est que les autres types de représentations paramétriques utilisés précédemment (Splines cubiques, courbes de Bézier, B-Splines) peuvent être définis comme des cas particuliers de NURBS.

1.1.4 Maillages

Le second aspect très important lors de la conception d'une pièce mécanique est le maillage prenant la forme du produit. C'est sur celui-ci que s'effectuera la simulation permettant d'évaluer le comportement final de l'objet qui a été créé au travers des modèles BREP. Un maillage, au sens large, est l'outil mathématique utilisé pour obtenir une solution numérique à des équations trop complexes pour être résolues analytiquement. Celui-ci consiste à discrétiser l'espace de définition des équations, et à ne les résoudre qu'en un nombre fini de points (les nœuds) en se basant sur les interactions entre eux. Ces maillages sont générés de manière procédurale à partir des modèles mathématiques, et peuvent être de différentes natures.

Les maillages structurés sont les maillages les plus simples à utiliser. Ils consistent en la déformation d'un motif de base (en général des juxtapositions de rectangles ou d'hexaèdres) pour qu'il remplisse au mieux l'espace à analyser (celui-ci pouvant être la pièce étudiée, afin d'évaluer les contraintes internes, ou son complémentaire, afin d'étudier les écoulements fluides générés, par exemple). L'image 1.2 illustre ce que peut être un maillage structuré.

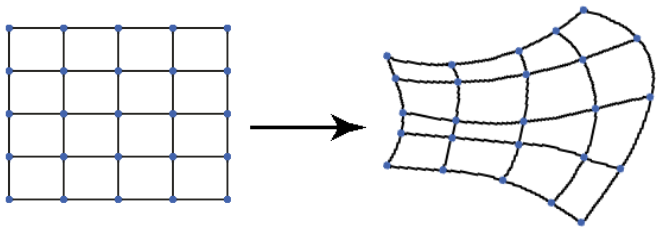

Figure 1.2 Illustration d'un maillage structuré 2D, avec le motif initial (à gauche) et la surface maillée (à droite).

Les maillages non structurés sont des maillages que l'on construit à partir d'une cellule de base (en général un triangle ou un tétraèdre) que l'on va répéter et déformer jusqu'à couvrir la totalité de l'espace. La différence majeure avec un maillage structuré est que le motif n'est plus régulier. La connectivité n'étant plus intrinsèque au motif, il va falloir la stocker, en plus des positions des nœuds. L'intérêt de ce type de maillage est la plus grande flexibilité dans la position des cellules, ce qui peut être plus adapté à certaines géométries (voir image 1.3).

Le dernier type de maillage, appelé maillages mixtes ou hybrides, regroupe les maillages dont une partie est maillée de façon structurée, et une autre de façon non structurée. Cette dernière catégorie est très utile pour utiliser les avantages des deux méthodes précédentes suivant les caractéristiques locales de la géométrie. L'image 1.4 montre un exemple d'utilisation de maillage mixte.

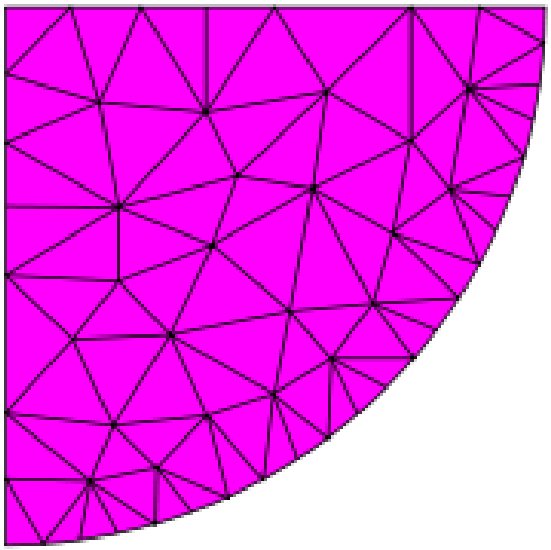

Figure 1.3 Illustration d'un maillage non structuré 2D d'un quart de disque.

1.1.5 Librairie Pirate

Avant de rentrer dans le vif du sujet, il est nécessaire de décrire l'environnement de travail. La librairie Pirate est une librairie informatique, codée en C++, implémentant les classes nécessaires à la construction de modèles BREP (basés sur la norme ISO-10303 (1994)) et de maillages (selon la norme Computationnal fluid dynamics General Notation System, CGNS). Sur cette librairie est basée une suite de logiciels (TopoVisu, Vérificateur, Maille) utilisée au cours de la conception de pièces hydroélectriques. Les fonctionnalités de chacun des logiciels seront détaillées au chapitre 3.

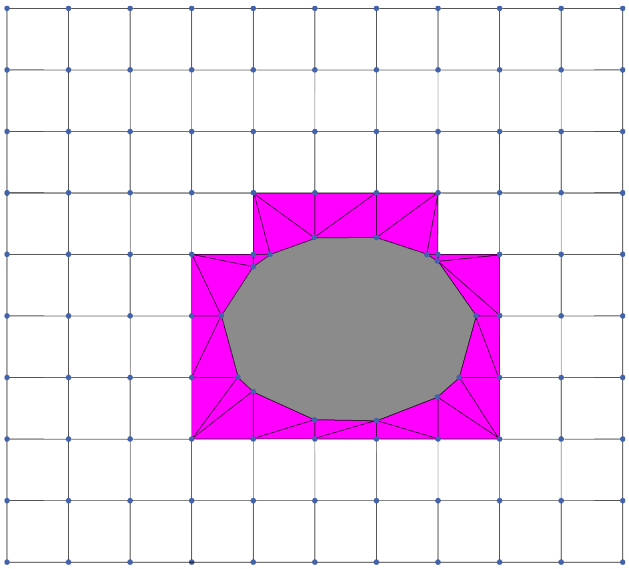

Figure 1.4 Illustration du maillage mixte d'un rectangle possédant un trou (en gris), le maillage non structuré est en rose.

1.2 Éléments de la problématique

La complexité des formes nécessaire à un comportement optimal de la pièce rend obligatoire un compromis entre une représentation fidèle de ces pièces et de bonnes performances de calcul. En effet, plutôt que de rendre les modèles extrêmement complexes au niveau mathématique en utilisant des courbes et surfaces de haut degré, afin que les différentes entités se superposent là où elles sont censées se correspondre (deux faces adjacentes sont censées avoir une arête identique), on utilise des tolérances qui vont permettre l'étanchéité des volumes, surfaces, ou courbes. Ces tolérances sont choisies par le concepteur, ou bien sont directement définies dans les logiciels de conception assistée par ordinateur. On va considérer qu'un modèle est étanche à une tolérance ϵ près si et seulement si toutes les entités de ce modèle valident la propriété suivante : toute sphère de rayon ϵ centrée sur la bordure (au sens BREP) de l'entité intersecte tout autre entité se raccordant à cette bordure.

Bien que cela permette de diminuer drastiquement la complexité des fonctions présentes dans le modèle, et donc les temps de calcul, l'inconvénient majeur de cette utilisation de tolérances est que celles-ci ne sont pas intrinsèques au modèle, elles doivent être spécifiées en plus de la géométrie. Et lors des transferts d'informations au sein de la chaine de conception, il arrive fréquemment que ces tolérances soient perdues ou modifiées. Retrouver ces tolérances n'est pas toujours simple. Certains cas (voir 1.5) posent de réels problèmes pour retrouver la connectivité entre les différentes entités lorsque les dimensions des géométries sont de l'ordre de celles des tolérances.

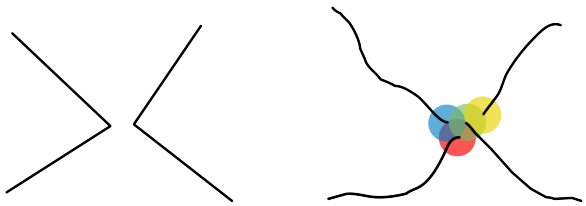

Figure 1.5 Cas idéal (à gauche) et représentation avec tolérances (à droite) d'une même situation topologique.

Plusieurs raisons existent à cette perte de données. Tout d'abord, aucun des grands logiciels de conception ne travaille dans un format standard de fichiers ; chacun a son format propriétaire. Les outils de conversion existent, mais ils peuvent introduire des erreurs à cause de non-compatibilités entre les formats. Ensuite, les différents services d'une entreprise (conception, simulation, production ...), qui peuvent utiliser le même logiciel afin de s'affranchir de ces problèmes de conversion, n'utilisent pas les mêmes configurations de tolérances. Le ser-

vice de conception va avoir tendance à être le plus précis possible afin que la pièce ait un comportement optimal, tandis que le service de production va essayer de garder de grandes tolérances, afin de réduire les coûts de production.

De même, des simulations basées sur un modèle non précis donneront des résultats plus éloignés de la réalité. De plus, ces simulations doivent en général passer par un maillage préalable du modèle numérique. On passe d'un volume défini par quelques faces à un grand nombre de cellules, permettant un calcul de type 'éléments finis'. Ce maillage aussi peut poser des problèmes, suivant les tolérances et les formes utilisées lors de la conception, les différentes cellules pourront être déformées, et ces non-conformités se retrouveront dans la justesse et la précision des résultats. Étant donné que ces différentes entités ne peuvent pas travailler avec les mêmes conventions, différentes solutions sont explorées.

Souvent, les erreurs dues aux conversions sont trouvées lors de l'étape de génération du maillage. C'est pour cela qu'il existe de nombreux outils de corrections de ces maillages. Certains algorithmes, par exemple, existent pour changer un maillage localement, sans en changer les bordures. On peut ainsi changer les paramètres de maillage dans une zone où les cellules sont déformées tout en pouvant relier ce nouveau maillage local à l'ancien.

Ce genre d'algorithmes de correction est particulièrement utile lorsque l'on n'a pas accès au modèle initial (ou lorsque l'on ne peut pas le modifier).

Dans le cadre du laboratoire de l'École Polytechnique de Montréal, le modèle peut être corrigé par les mêmes personnes qui effectuent la simulation. L'objectif du présent travail n'est donc pas simplement d'implémenter des méthodes de corrections déjà existantes, mais plutôt de créer un outil de diagnostic des modèles. Cet aspect d'identification automatique des erreurs, en amont de la correction, peut aussi grandement faciliter le travail des concepteurs, et donc accélérer le processus de création d'une pièce, ainsi que la qualité finale de celle-ci. C'est cette hypothèse que nous allons tenter de vérifier au cours de ce travail.

Mais détecter des erreurs n'est pas suffisant pour assurer une bonne qualité de produit. Pour s'assurer que le résultat final soit au niveau attendu par les concepteurs, il faut s'assurer que les problèmes détectés soient correctement perçus. Un environnement de visualisation pourrait permettre aux concepteurs de mieux percevoir les caractéristiques et les zones à risques des pièces qu'ils conçoivent. C'est pour cela qu'il semble qu'un environnement graphique peut fournir une fonctionnalité essentielle lors de la phase d'analyse d'un modèle 3D.

Le dernier aspect qui sera traité dans ce mémoire est le problème de la communication des données. Ce problème est loin d'être trivial, car aujourd'hui, il existe différents types de modèles, de nombreux formats de fichiers et de nombreux logiciels pour dessiner un produit. Mais rien ne garantit que ces différentes méthodes soient équivalentes. Il est donc très difficile

de réussir à convertir un modèle défini dans un standard vers un autre format, et ce sans aucune corruption de données. Et s' il est si difficile de transmettre des informations d'une division d'une entreprise à une autre, il risque d'être encore plus compliqué de s'assurer que les modèles créés aujourd'hui soient encore accessibles dans 20 ans, lorsque les méthodes et les outils de modélisation auront évolué.

Il est extrêmement difficile de trouver une solution à ce problème de transmission des données, car ce n'est pas uniquement un problème technique. Il dépend énormément des habitudes futures de la communauté scientifique, ainsi que des entreprises qui conçoivent les logiciels de CAO. Ce que l'on propose ici est donc de réduire les problèmes de communication dans le contexte du développement de pièces mécaniques. Cette approche est en lien direct avec la détection d'erreurs, puisque c'est sur celles-ci qu'il va être important de communiquer. Mais comme indiqué précédemment, ce problème de communication est différent dans chaque contexte de recherche, c'est pourquoi ce travail va se concentrer sur l'amélioration de la communication autour des données du laboratoire de l'École Polytechnique de Montréal.

1.3 Objectifs de recherche

Les objectifs principaux de recherche présentent deux aspects : le premier sera la détection d'erreurs, le second sera la communication autour de ces erreurs.

Il s'agira donc dans un premier temps de faire le point sur les outils de diagnostic existants, puis de proposer des améliorations permettant de résoudre les difficultés présentées précédemment. Ces améliorations pourront se situer au niveau de l'interface du logiciel, afin que l'utilisation des outils existants puisse se faire plus efficacement. Mais d'autres outils peuvent aussi être proposés afin d'améliorer la visualisation des modèles 3D et des maillages.

Il s'agira aussi de proposer des solutions afin de faciliter la communication tant au sein d'un même service de conception, qu'entre différents centres de recherche. Il faudra donc identifier un format aisément transmissible, mais qui peut aussi être conservé sur une période de temps suffisamment longue en comparaison à la durée de vie moyenne d'un produit. Cet aspect se concentrera principalement sur la possibilité d'ouvrir les données propriétaires d'un laboratoire à d'autres centres de recherche. Le point de départ est donc restreint pour rendre ce travail réalisable dans le contexte d'une maîtrise, mais l'auditoire visé devra être le plus large possible.

1.4 Plan du mémoire

Ce mémoire se découpe en trois chapitres principaux. Le chapitre 2 fait le point sur les problèmes présents dans la phase de design préliminaire dans le contexte de la mécanique.

Au cours de cette étude bibliographique, les différentes solutions décrites dans la littérature seront étudiées, et certaines seront identifiées pour servir d'inspiration à la suite du travail.

Le chapitre suivant (chapitre 3) s'attarde sur le cas particulier de la détection d'erreurs au laboratoire MAGNU de l'École Polytechnique. Une liste d'erreurs sera établie et les solutions appliquées pour les détecter seront détaillées et illustrées par des cas tests.

Le chapitre 4 du mémoire se concentre sur le problème de la transmission des données dans le cadre de la conception. En particulier, le format choisi, le PDF 3D, sera décrit, ainsi que sa mise en application au sein des outils développés.

CHAPITRE 2

REVUE DE LITTÉRATURE

Cette section a pour objectif principal d'expliquer les choix technologiques effectués au sein de la suite de logiciels Pirate, en se basant sur les travaux d'autres laboratoires. Les solutions présentées sont regroupées en fonction du problème traité. Les modèles CSG seront discutés dans un premier temps, puis les problèmes et solutions dans les modèles BREP seront analysés. Nous continuerons avec les maillages, puis les problèmes de conservations et de transmissions des données. Enfin, des technologies plus éloignées des problèmes traités dans cette maitrise, mais pouvant servir de source d'inspiration, seront étudiées.

2.1 Les modèles CSG

Les modèles CSG ont l'avantage principal d'être très compréhensibles humainement, les éléments de base sont simples, et les fonctions d'assemblage aussi. Ces modèles permettent aussi de retrouver partiellement le raisonnement derrière un modèle, grâce à l'ordre d'assemblage des différentes parties. Le problème principal de ces modèles est la longueur des temps de calcul nécessaires pour obtenir un rendu après modification d'un modèle. En effet, si l'on modifie un paramètre d'une feuille profonde de l'arbre de construction, le nombre d'opérations booléennes entre volumes à recalculer peut devenir extrêmement important. Paviot *et al.* (2012) démontre bien que l'ajout de nouvelles formes de base (en l'occurrence, des sphères) se produit en temps exponentiel. Cette méthode n'est donc utilisable que pour des formes simples. C'est ce qui est fait dans l'article en question, puisque seules les enveloppes des composants des centrales électriques sont modélisées pour les besoins de leur agencement.

2.2 Problèmes des modèles BREP

Comme signalé en introduction, la majorité des problèmes des modèles BREP vient d'un manque d'informations, principalement au niveau des jonctions entre les différentes entités. L'article Matsuki (2010) signale bien que ce manque d'information est autant dû à un problème de stockage de données qu'à des différences de tolérances au sein de la chaine de conception. Pour régler ces problèmes, on peut ajouter des informations aux modèles. La librairie Pirate définie des entités sommets, arêtes et faces, qui seront uniques à chaque frontière. Le côté commun à deux faces sera défini par deux "arêtes associées", qui seront donc associées à l'arête définie préalablement. Toutes ces entités forment la topologie des modèles,

et sont inspirées de la norme ISO 10303 (plus couramment appelée STEP, pour STandard for the Exchange of Product model data, Cf. ISO-10303 (1994)).

Une autre solution qui peut aider est la réduction du nombre d'entités définies afin de réduire les risques d'erreurs lors des transmissions et conversions de données. Par exemple dans l'article Gopalsamy *et al.* (2004), qui utilise un modèle basé sur celui présenté par Chew *et al.* (2002), il est choisi de ne stocker les entités qu'à l'aide d'une unique équation donnant sa définition dans l'entité de dimension supérieure. Par exemple, une courbe définissant une arête ne sera exprimée que dans la face qu'elle délimite. Cette méthode permet de réduire les problèmes au sein d'une même face, mais va ralentir les temps de calcul par la suite, puisqu'on n'aura pas d'expression directe pour passer de l'espace 3D aux coordonnées d'un sommet par exemple.

Ces solutions sont parfois clairement incompatibles et ne sont pas intégrées aux différents formats d'échange de données. Ainsi, chaque groupe industriel ou de recherche va avoir tendance à retenir la solution qui convient le mieux à son application, ce qui ne facilite pas les transferts de données (Cf Mezentsev et Woehler (1999)) et donne lieu à une diversité importante de formats (un point sur ceux-ci a été fait dans Steves et Frechette (2003)) qui rend la création d'un outil unique de conversion impossible. De plus, les différentes tolérances utilisées pour un modèle ne sont pas toujours transmises au logiciel de maillage (la conception géométrique et la simulation sont en général effectuées séparément, même si les deux sont interdépendantes), ce qui peut changer les propriétés d'étanchéité.

Afin de se prémunir contre tous les problèmes de géométrie et de topologie, une solution est d'intégrer au processus de maillage une vérification du modèle BREP. Cela peut se faire de plusieurs manières. Dans l'article Steinbrenner *et al.* (2000), il semble que le projet JULIUS ait opté pour une génération interactive du maillage. Ainsi, si un problème se présente, l'utilisateur peut corriger ou éclairer le programme sur la réalité du produit, corrigeant directement les erreurs. Cette approche est intéressante, mais est contraire au principe du traitement par lots utilisé pour les maillages de tailles importantes.

L'article Yang *et al.* (2005) décrit les fonctions principales d'un système de vérification, ainsi que l'implémentation que les rédacteurs ont effectuée. L'idée est de créer une liste de toutes les erreurs possibles (écart entre une arête et un sommet, entre deux arêtes associées, etc.), et de vérifier pour toutes les entités que les erreurs n'apparaissent pas. Cette méthode, bien que longue, peut détecter une grande majorité des problèmes, et s'automatise aisément. En suivant la même idée, Tanaka et Kishinami (2006) propose une méthode de validation de qualité pour les modèles BREP. Afin de garder un maximum d'objectivité et d'indépendance dans la description de cette méthode, il utilise le langage formel EXPRESS (défini dans la norme ISO 10303) tant pour décrire les modèles que les tests effectués. Comme tout modèle

STEP peut être défini dans ce langage, une généralisation de cette méthode est envisageable. Ces outils de vérifications de modèles topologiques ont pour objectif de déceler les problèmes qui pourrait n'être détectés que lors des simulations, bien plus tard dans le processus de conception (et donc bien plus couteux à corriger). On peut donc détecter les erreurs dans les modèles avant de les utiliser et l'automatisation du processus permet de rendre le traitement par lots possible. La limite principale des deux approches précédentes est la supposition que l'on peut créer un registre de la totalité des erreurs qui peuvent se produire. De plus, une fois les erreurs détectées, il est nécessaire de faire intervenir un spécialiste afin de corriger celles-ci.

On remarque donc que chacune des technologies présentées ici possède ses avantages et ses inconvénients, et il est donc nécessaire de choisir une approche qui convient à chaque contexte de travail. Malheureusement, c'est dans cette diversité des solutions que réside une source du problème de communication, et de génération d'erreurs.

2.3 Solutions au niveau des maillages

Dans le domaine des simulations, la génération du modèle géométrique et topologique n'est pas le seul point de difficulté. En effet, la qualité du maillage est, elle aussi, très importante vis-à-vis de la qualité des résultats obtenus. Les traitements effectués précédemment afin de "réparer" le modèle BREP facilitent la génération d'un maillage correcte, mais parfois, la forme même des pièces conduit à un maillage de moindre qualité.

Afin de mesurer la qualité d'un maillage, on peut s'intéresser à un certain nombre de défauts, qu'il va falloir corriger. Ces défauts peuvent être une connectivité (nombre d'arêtes reliées à un nœud du maillage) trop importante, des cellules dont un paramètre (angle, longueur de côté) est trop grand par rapport aux autres, une variation du volume trop importante au passage d'une cellule à l'autre, etc.

Pour éviter ce genre de déformations, l'article Gopalsamy *et al.* (2004) propose un formalisme XML pour fournir plus d'information au logiciel de maillage. Plusieurs classes sont définies, et permettent au concepteur de signaler les zones à risques dans le modèle, et de choisir quelle méthode de maillage est la plus adaptée à chacune de ces zones.

Cette idée qu'un modèle BREP seul n'est pas toujours suffisant pour obtenir des simulations de bonne qualité se retrouve dans l'article Chong *et al.* (2007). Dans cet article, l'idée principale est de ne pas chercher à corriger les erreurs du modèle, mais de mailler directement le modèle incorrect, et de corriger le maillage ensuite. Pour corriger celui-ci, les auteurs ont développé des algorithmes de détection et de correction automatiques pour les trous, les recouvrements, les joints en T et les éléments déformés. L'inconvénient de ce processus, comme

le signale l'auteur, est que l'on n'obtient jamais un modèle BREP de qualité, alors que celui-ci peut être nécessaire aux étapes suivantes du processus industriel (pour la production assistée par ordinateur, par exemple).

En s'inspirant de cette idée, les auteurs de l'article Busaryev *et al.* (2009) décident de prendre en compte la présence des défauts du modèle dans une étape préalable au maillage. Pour cela, toutes les frontières vont être recouvertes de "boules de protection", et les boules vont être fusionnées lorsqu'elles sont trop proches les unes des autres, ce qui résulte en un modèle BREP étanche. Le rayon des boules permet de fixer la taille maximale des détails qui vont être effacés. L'innovation de cette technique, en comparaison avec la méthode de réparation de modèles basée sur les tolérances, est que le modèle n'est pas la seule information transmise à l'algorithme de maillage, les boules sont aussi utilisées afin de définir les points initiaux dans l'algorithme de maillage. Même si l'on obtient bien une méthode pour mailler des modèles BREP défectueux, et qu'on obtient aussi un modèle de bonne qualité, on conserve la limitation de la méthode de réparation basée sur les tolérances (dont parle l'article Segal (1990), par exemple) : les défauts à effacer doivent avoir une taille inférieure aux détails de la géométrie souhaitée, sinon les détails de taille comparable aux défauts seront aussi lissés par la réparation.

D'autres articles proposent d'effectuer une vérification du maillage suite à sa génération. La vérification proposée dans Camarero *et al.* (2008) se base sur certaines propriétés des cellules. Ces propriétés sont la longueur des côtés, le rapport d'aspect, la surface (pour les cellules 2D), le volume et le covolume (pour les cellules 3D). Les tests effectués permettent de valider ou d'invalider un maillage, on peut ensuite le recalculer avec d'autres paramètres ou appliquer des méthodes de réparations.

Les méthodes les plus classiques sont des changements locaux de connectivité, ce qui peut par exemple, homogénéiser la taille des cellules (Cf 2.1).

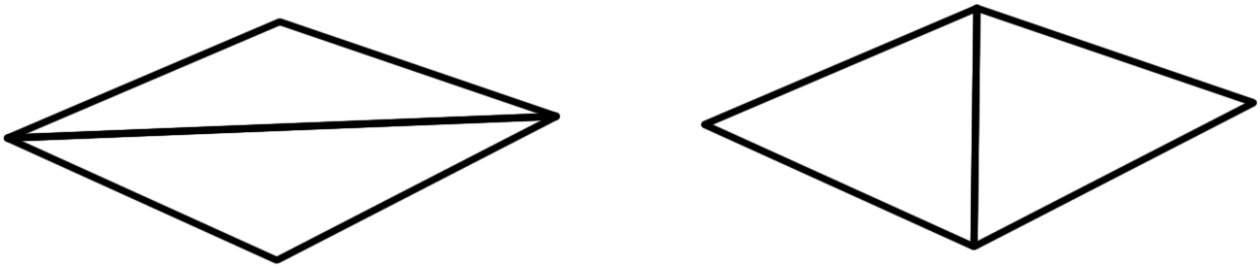

Figure 2.1 Illustration du changement local 2-2 flip

Pour avoir de meilleurs résultats, l'article Liu et Sun (2006) propose d'augmenter le nombre de cellules modifiées par la correction. Il décrit une méthode (Small Polyhedron

Reconnection ou SPR) pour créer un optimum dans les informations de connectivité autour des cellules qui posent problème. Cela permet d'obtenir un meilleur maillage en ne modifiant que localement celui-ci, ce qui réduit les temps de calcul, et évite de créer d'autres problèmes en d'autres endroits par un nouveau calcul global.

Dans la même optique de changement local, l'article Bunin (2006) propose de définir une zone modifiable autour d'un nœud défectueux, et de recalculer le maillage dans cette zone. Contrairement à la méthode SPR, les nœuds sont déplacés, ce qui permet des changements plus importants, mais qui nécessite une plus grande liberté. Ainsi, cette méthode est beaucoup plus efficace hors des bords du maillage, où les positions des nœuds sont beaucoup plus contraintes.

Mais la qualité d'un maillage n'est pas le seul problème, il faut aussi s'intéresser à sa justesse, à son adéquation au modèle initial. C'est ce que fait Frey et Borouchaki (1999) au niveau des surfaces, entités critiques dans les modèles mécaniques usuels.

Ces différents articles permettent d'obtenir un maillage de qualité, soit directement à partir d'un modèle BREP (défectueux ou annoté), soit à partir d'un maillage de mauvaise qualité. Ainsi, on peut obtenir des résultats de simulations justes, et avec une erreur suffisamment faible.

2.4 Transmission et conservation des données

Les articles présentés précédemment permettent, après avoir appliqué les méthodes décrites, d'obtenir des modèles et des maillages corrects. L'une des problématiques qui persistent est de s'assurer que les données obtenues conservent leur bonne qualité. En effet, comme le signal Matsuki (2010), des erreurs peuvent être créées lors de la transmission de données, que ce soit vers un autre service de la même entreprise (portabilité spatiale), ou sur une version future d'un logiciel utilisé par les mêmes personnes (portabilité temporelle).

L'article Sum *et al.* (1996) présente une idée pour résoudre le problème de la portabilité spatiale. L'objectif du travail présenté est de poser les bases d'une structure de données permettant de réunir les informations nécessaires à tous les aspects de la conception d'un produit. En pratique, une base de données est générée, et il est possible de présenter les informations présentes dans celle-ci de différents "points de vue", ces points de vue correspondent aux différents formats natifs des applications utilisées. Ainsi, les conversions entre les différents formats se font au sein d'une structure spécialisée, ce qui évite les erreurs. Le problème avec un tel système, c'est qu'il suppose qu'une conversion sans erreurs, si elle est prévue dès le départ, est possible. Or, rien ne permet actuellement de le garantir. Comment s'assurer qu'un produit modifié par un logiciel qui ne traite pas les tolérances reste valide

selon les tolérances fixées par un autre logiciel, et qui sont toujours attachées au produit dans la base de données ?

Gerbino (2003) essaye de préciser l'origine des erreurs lors de la transmission de données afin de proposer une solution en traitant le problème à sa source. La conclusion principale de l'étude présentée est que les erreurs proviennent en général des différences au sein des moteurs de modélisation 3D. Il est beaucoup plus difficile de transmettre les données entre deux logiciels dont les principes fondamentaux sont différents, plutôt qu'entre deux logiciels ou seules les implémentations diffèrent. De plus, entre deux logiciels utilisant le même moteur de modélisation, on peut encore avoir des problèmes de précision (ou de tolérances) lorsqu'on exporte les données d'un format à un autre. La solution utopique évoquée dans l'article est l'utilisation d'un unique logiciel de conception assistée par ordinateur dans le monde. Cette solution n'est bien évidemment pas réalisable à l'heure actuelle, c'est pourquoi tant de groupes de travail cautionnent et proposent des solutions partielles qui consisteront à réparer les modèles transmis en détectant et corrigeant les erreurs générées. Butlin et Stops (1996), Krause *et al.* (1997), Chand (2001), Yang et Han (2006) et Marchandise *et al.* (2011) sont autant d'exemples qui montrent que le problème de la gestion des erreurs au sein des modèles est présent tout au long de l'évolution des techniques de simulation informatiques. Ce problème risque de ne pas être réglé dans un futur proche, car, comme le signale Szykman *et al.* (2001), bien que le marché des logiciels de CAO soit dirigé par les besoins des entreprises de conception, un éditeur de logiciel qui fournit un moyen d'exporter ses données vers un format standard devient plus vulnérable à la concurrence.

Un autre aspect important concernant la conservation des données est la portabilité temporelle. L'article Regli *et al.* (2011) s'attarde sur ce problème en posant la question de la quantité de données à conserver. Qu'est-ce qui est nécessaire à la préservation de toutes les informations, qu'est-ce qui est suffisant ? Les auteurs de l'article signalent qu'il est préférable d'avoir les données de pièces sous au moins trois formats : un format propriétaire, qui permet de travailler avec l'application de conception dans son format natif, un format standard, comme STEP, qui permet de transmettre les informations à des personnes extérieures à l'entreprise initiale et qui a plus de chances d'être encore lisible dans les années futures car plus strict et documenté, et enfin, un format de visualisation, plus facile à générer et à transmettre. Bien que cela donne une idée des types de données qu'il faudrait être capable de générer, les conclusions de l'article sont claires, ces trois formats n'assurent en rien que les données ne seront pas perdues. Le meilleur moyen de conserver les données des produits conçus reste de conserver la totalité des fichiers produits, dans toutes les versions produites. Mais cette approche entre évidemment en conflit avec les coûts d'archivage de données, qui peuvent devenir particulièrement importants pour de grosses entreprises industrielles.

Kheddouci (2010) se pose le même problème de conservation des données dans le temps, mais non pas pour la qualité des données, mais du point de vue légal. En effet, aujourd'hui, les seules archives qui peuvent et doivent être conservées pour raisons juridiques sont sous forme de plans imprimés. Ceci s'explique par le fait que le papier est le seul support qui a une longévité suffisante pour couvrir toute la durée de vie de certains produits. On peut par exemple penser au bombardier Boeing B-52 Stratofortress, qui a volé pour la première fois en 1952, et dont l'Air Force prévoit qu'il vole encore en 2040, soit une durée de vie de presque un siècle. Mais les plans papier ne sont aujourd'hui presque plus utilisés lors de la conception des produits, tout le processus est devenu numérique. La problématique proposée par Kheddouci (2010) est de trouver un support qui garantit la conservation des données. Il propose de conserver les modèles sous format PRC inclus dans des documents PDF. Ce choix se justifie par l'adéquation du PRC à la norme ISO 10303 (STEP), qui est la plus utilisée pour décrire les modèles BREP, et par le fait que les documents PDF sont extrêmement répandus dans le monde, et qu'il y a donc de fortes chances pour que ce format soit encore maintenu dans les années à venir.

Les différents problèmes de communications présentés précédemment sont loin d'être triviaux, ou ils auraient déjà été réglés au sein des différents centres de recherche. Et pourtant, ces problèmes sont étudiés activement actuellement, partout dans le monde. On peut citer pour exemple la conférence internationale sur la gestion du cycle de vie des produits (Product Lifecycle Management International Conference Information Processing (IFIP) Working Group 5.1 (2013)) où de nombreux chercheurs viennent présenter leurs travaux sur ces problèmes de communications, et de gestion autour des produits et des données qui leur sont associées. C'est au cours de l'édition 2012 de cette conférence que le travail présenté dans ce mémoire a été présenté. L'article rédigé pour l'occasion est présent en annexe A.

2.5 Choix du format

Afin de pouvoir communiquer les données concernant les modèles BREP et leurs erreurs, il faut s'assurer de les transmettre dans un format qui sera facilement interprétable par le destinataire. Mais comme nous souhaitons que ces données soient aisément accessibles à la fois aujourd'hui et dans le futur, nous voulons nous assurer que le format choisi sera maintenu. Il est évidemment impossible de s'assurer complètement qu'un format sera toujours utilisé, mais on peut favoriser cette caractéristique en utilisant un format qui est déjà très utilisé aujourd'hui.

Afin de faciliter la communication, le format choisi doit pouvoir être utilisé librement et gratuitement. Dans cette optique, deux choix principaux s'offrent à nous. Premièrement, il

existe des standards, gratuits et libres, du stockage de modèles BREP. On peut par exemple citer Initial Graphics Exchange Specification(IGES), qui est de moins en moins utilisé, ou STEP, son remplaçant.

Le format IGES est un format de fichier publié par le National Institute of Standards and Technology, en 1980 (Nagel *et al.* (1980)). Il a vu le jour suite à une initiative de plusieurs utilisateurs et vendeurs de logiciels de CAO qui souhaitaient obtenir un format d'échange commun. Ce format a été mis à jour plusieurs fois depuis, notamment en 1983(Smith *et al.* (1983)) et 1986 (Smith *et al.* (1986)). L'un des avantages du format IGES est qu'il est un format ASCII (American Standard Code for Information Interchange), non binaire. Il peut donc être lu assez aisément par un humain.

STEP a été initialement développé à partir de 1984 comme successeur au format IGES (ISO TC184 / SC4 resolution 1, Gaithersburg - July 1984). Mais ce standard va plus loin, et ne se contente pas de décrire un format de fichier pour des modèles BREP (bien que ce soit cette partie du standard qui nous intéresse ici). Le standard ISO 10303 se charge de définir des normes de descriptions, d'implémentations, de tests et de biens d'autres aspects des technologies assistées par ordinateur (xAO) afin que la totalité des procédés soit intégrée et que des barrières ne se dressent pas entre les différentes étapes de la création d'un nouveau produit. Dans la continuité du format IGES, le format de description de modèles BREP de STEP peut être lu par un humain, ce qui peut s'avérer un avantage lors de l'échange de données.

Mais ces formats ont un défaut majeur : ils n'ont pas de logiciels associés qui permettent d'ouvrir simplement les fichiers. Les visualisations 3D sont les seules solutions possibles pour un humain de réellement interpréter une pièce très complexe. Les seules données numériques sont en général loin d'être suffisantes pour un individu moyen. De plus, comme signalé dans Gerbino *et al.* (1997), les spécifications du format IGES contiennent de nombreux points sujets à l'interprétation, qui produisent autant d'implémentations différentes au sein des outils de conversions, ce qui nuit énormément à la fiabilité de ce format comme méthode de transfert de données.

C'est pourquoi nous nous sommes orientés vers l'autre solution, le PDF (avec intégration de modèles 3D). Ce format est déjà extrêmement répandu, autant dans un contexte professionnel que personnel. Une grande majorité des ordinateurs possède un moyen de visualiser les documents dans ce format. De plus, le PDF 3D est un format qui se répand de plus en plus comme un standard de visualisation et de communication à propos de modèles 3D. C'est pourquoi nous nous sommes orientés vers ce format d'exportation.

Pour créer un fichier PDF 3D, plusieurs choix doivent être faits. Dans un premier temps, il faut savoir que les données 3D ne sont pas intégrées directement dans le fichier PDF, elles

sont enregistrées dans un fichier binaire qui peut ensuite être intégré dans le PDF. Ce fichier binaire peut être de deux formats : U3D (Universal 3D) ou PRC (Product Representation Compact, Adobe Systems Incorporated (2012)). Ces deux formats permettent de stocker des informations sur des modèles 3D. Le premier est principalement basé sur des représentations polygonales héritées des standards de l'affichage sur ordinateurs (OpenGL, Direct3D) alors que le second a l'énorme avantage d'être compatible avec la norme ISO 10303 (STEP), qui est au cœur de la plupart des modélisations BREP. On conserve donc la logique d'unification de cette norme, tout en ayant des visuels pouvant être ouverts sur n'importe quel ordinateur, sans installation de logiciel payant. C'est pourquoi c'est ce format qui a été choisi pour l'exportation.

2.6 Autres travaux dans le domaine de la conception

Bien évidemment, il existe d'autres travaux qui visent à améliorer le processus de design de pièces mécaniques. Ces travaux peuvent utiliser des approches bien différentes de celle proposée dans ce mémoire, parfois tellement différentes qu'elles sont incompatibles avec le principe de fonctionnement du laboratoire MAGNU de Polytechnique.

Le travail présenté dans l'article Li *et al.* (2010) vise à réduire le travail demandé au concepteur. Pour cela, une méthode d'optimisation automatique est proposée. L'idée est de générer une base de données de modèles, et des fonctions d'évaluation de la justesse d'un produit. Ces fonctions d'évaluations sont une forme mathématique des règles qu'appliquent les concepteurs lors de leur création de design. On pourra ensuite évaluer la justesse d'un nouveau produit, et proposer des solutions similaires de meilleure qualité au concepteur, en résolvant un problème mathématique d'optimisation. Cette idée est particulièrement applicable lorsqu'on fait des produits de nature similaire, comme dans une entreprise automobile ou aéronautique où toutes les pièces d'un moteur n'ont pas besoin d'être réinventées.

Dans la même idée de réduction du travail de conception, Frey *et al.* (2009), puis Iraqi et Roucoules (2012) proposent une méthode pour réduire le temps consacré à la récupération d'informations. En effet, dans la plupart des implémentations de logiciels CAO, les informations enregistrées sont uniquement le modèle (sa géométrie). L'approche présentée par les auteurs est d'enregistrer la totalité du processus de création d'une pièce afin qu'un utilisateur futur du modèle puisse saisir le raisonnement derrière la conception. L'approche est de type 6W (who, what, where, when, why, how) et se base sur un principe de méta modèle. Ce modèle est une abstraction du modèle réel, et permet de stocker plus que l'information géométrique. L'intérêt majeur dans le cadre de la transmission des données est que l'on se rend indépendant des logiciels utilisés, et comme l'information conservée est plus exhaustive

que dans les formats natifs des applications classiques, il est possible de produire une ”vue” adéquate du projet, dans un format spécialisé pour une tâche donnée. On se rapproche ici de l'approche présentée dans Sum *et al.* (1996), et décrite en 2.4.

Pour ce qui est d'éviter la perte de données lors de transmissions de modèles, une approche pourrait être de créer une base de données comportant non seulement les données géométriques, mais aussi toutes les données concernant le comportement de la pièce et le raisonnement derrière la conception de celle-ci. Cela permettrait, s'il y a corruption du modèle, d'avoir la totalité des informations nécessaires à la réparation des données plus facilement. Le concept d'une telle base de données est ébauché dans Bohm *et al.* (2008) puis développée dans Bohm *et al.* (2010). Ces bases de données ne règlent pas directement les problèmes de changement de formats (même si l'on peut imaginer qu'un même produit soit enregistré sous différents formats afin de faciliter les transferts, les corrections dues à la conversion étant directement effectuées par le concepteur initial), mais elles pourraient aider à la reconstruction. Malheureusement, la lourdeur du remplissage de telles bases de données (autant initialement que lors de l'utilisation journalière ensuite) rend leur utilisation très difficile dans de petites structures.

Le travail présenté dans Sheffer et Üngör (2001) vise à réduire les temps de génération de maillage en évitant de mailler à nouveau les parties des modèles qui n'ont pas été modifiées. Bien que cette approche ne concerne pas la justesse des modèles, on peut imaginer que les parties des modèles qui n'ont pas été changées restent aussi juste qu'avant les modifications. Ainsi on pourrait réduire le nombre d'entités géométriques et topologiques à tester ainsi que la quantité de cellules à vérifier sur le maillage final.

D'autres travaux se consacrent à la reconstruction de parties absentes d'un modèle. Par exemple, Emelyanov *et al.* (2009) propose une solution basée sur la définition de champs de probabilité de présence des surfaces, en partant des données existantes. Le travail a été effectué dans le but de recréer des surfaces sur des nuages de points, lorsque les algorithmes classiques ne donnent pas de bons résultats. Bien évidemment, le contexte n'est plus celui de la conception / création, mais les problématiques de récupération de données géométriques corrompues restent communes.

La totalité des travaux présentés dans ce chapitre montre que la présence de défauts dans les modèles BREP et les maillages est un problème qui doit être résolu si l'on souhaite avoir des résultats de simulations de qualité. Une approche théoriquement simple peut être utilisée pour cela : il faut identifier ces défauts, les corriger, et s'assurer qu'ils ne se représentent pas. En pratique, il est beaucoup plus compliqué de trouver une solution universelle étant

donnée la diversité des erreurs rencontrées (la variété des articles en traitant en témoigne). Dans la suite, nous allons donc proposer notre solution qui s'oriente principalement sur la communication, afin que ces défauts ne se représentent pas, ou qu'ils puissent être traités efficacement.

CHAPITRE 3

DÉTECTION D'ERREURS

Dans ce chapitre, le premier aspect du travail, la détection d'erreurs, va être traité. Dans un premier temps, une brève mise en contexte va être effectuée, les différentes erreurs vont être listées, puis les outils mis en place pour les traiter seront détaillés.

3.1 Mise en contexte

Un nouveau design de produit commence par la création d'un modèle. Celle-ci n'est pas effectuée au laboratoire MAGNU de Polytechnique, elle est faite au sein de Andritz Hydro, l'entreprise avec laquelle le laboratoire a un partenariat. Par la suite, ce modèle est amélioré au sein du laboratoire à l'aide des logiciels présentés ci-après, et le modèle numérique final est enfin renvoyé à Andritz Hydro pour que le design détaillé puisse se poursuivre. La figure 3.1 présente une schématisation de ce processus de conception, l'encadré bleu représentant les étapes effectuées au laboratoire, et qui sont donc au cœur du présent travail. Le tableau 3.1 présente les logiciels utilisés, en précisant à quelles étapes chacun intervient.

Tableau 3.1 Logiciels utilisés dans le processus de conception.

Logiciel	Étape(s) d'intervention	Description
Vérificateur	Analyse du modèle	Le logiciel Vérificateur se charge de l'analyse du modèle, c'est à ce niveau que les erreurs (présentées dans la section suivant) doivent être détectées.
TopoVisu	Visualisation (modèles et maillages), analyse du maillage	TopoVisu assure la visualisation de la suite de logiciels basée sur la librairie Pirate, au niveau des modèles BREP, mais aussi des maillages. Les erreurs détectées à ces deux niveaux peuvent aussi être visualisées.
Maille	Création / Modification du maillage	Maille se charge de la création du maillage à partir du modèle BREP. Le maillage peut ensuite être modifié en changeant la méthode de maillage ou la taille des mailles par exemple.
Check_mesh	Analyse du maillage	Check_mesh assure l'analyse du maillage, en parallèle à TopoVisu. Il est plutôt prévu pour une utilisation dans le cadre d'analyses par lots.

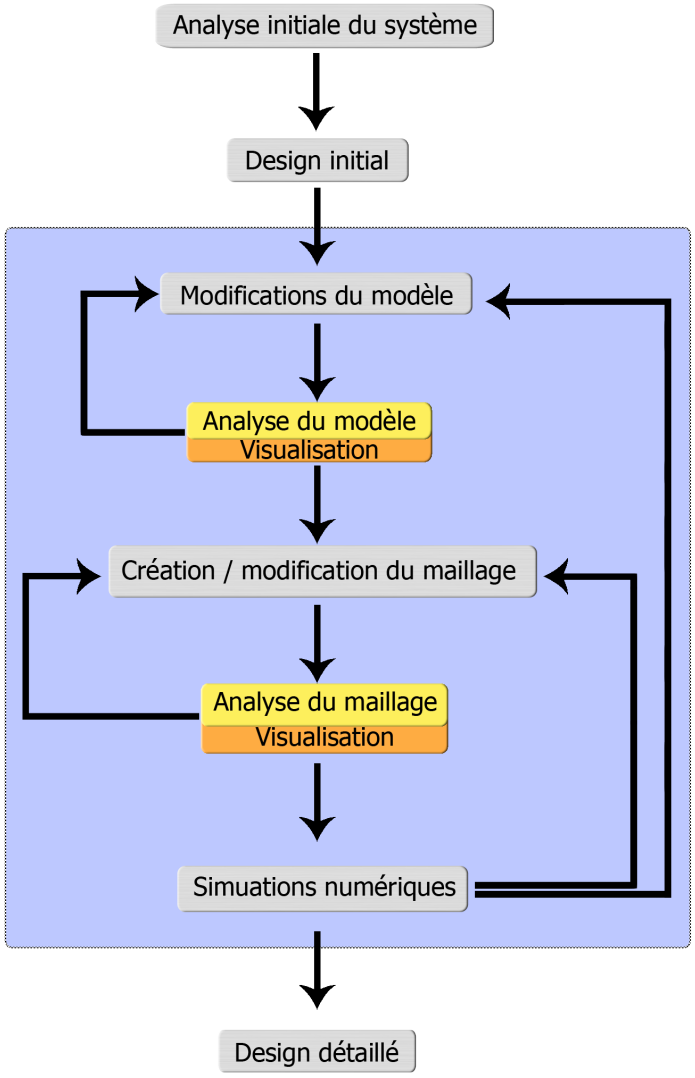

Figure 3.1 Schéma du processus de conception.

D'autres logiciels sont utilisés par la suite pour effectuer les simulations et observer les résultats obtenus, mais ceux-ci s'éloignent de notre objectif initial de la validation de modèles BREP et de maillages. En effet, à l'étape de la simulation, les modèles et les maillages doivent déjà être validés afin de garantir la fiabilité des résultats obtenus.

3.2 Présentation des erreurs traitées

Nous allons décrire ici les points qui doivent être vérifiés pour assurer la validité des modèles BREP et des maillages.

3.2.1 Erreurs des modèles BREP

Les articles Yang *et al.* (2005) et Tanaka et Kishinami (2006) proposent une classification similaire des erreurs pour les modèles BREP basés sur la norme STEP, cette classification peut donc s'appliquer à notre environnement de travail.

Nous reprenons ici la classification issue de l'article Yang *et al.* (2005) car celle-ci est indépendante de l'implémentation effectuée. Seuls les intitulés des erreurs ont été extraits des travaux de Yang *et al.* (2005), leur interprétation a dû être adaptée aux structures de données présentes dans l'implémentation Pirate des modèles BREP. Ces erreurs sont au nombre de 19 et sont détaillées ici :

1) Écart arête-sommet Sommets et extrémités des arêtes doivent se correspondre géométriquement.

2) Longueur des arêtes Toutes les arêtes se correspondant doivent avoir la même longueur.

3) Courbure des arêtes Toutes les arêtes se correspondant doivent avoir une forme similaire.

4) Écart face-arête Les arêtes délimitant une face doivent reposer sur celle-ci.

5) Consistance arête-boucle Les arêtes délimitant une face doivent former une boucle fermée, orientée correctement.

6) Écart/recouvrement arête-arête Les extrémités d'arêtes en contact doivent se correspondre.

7) Angle arête-arête Des angles trop aigus peuvent conduire à des modèles très difficilement analysables.

8) Proximité arête-arête Deux arêtes de faces juxtaposées doivent se correspondre.

9) Degré des faces Des entités de degré inutilement élevé peuvent ralentir tout traitement informatique (rendu graphique ou analyse numérique).

10) Écart/recouvrement face-face Deux faces juxtaposées doivent se correspondre le long de leur arête commune.

11) Angle face-face Des angles trop aigus peuvent conduire à des modèles très difficilement analysables.

12) Consistance face-face Les faces internes à un volume doivent se correspondre.

13) Nombre d'utilisations des arêtes Dans un volume, chaque arête doit être utilisée deux fois (une dans chaque direction) pour assurer sa fermeture.

14) Faces/surfaces nulles Les faces / surfaces ne doivent pas être dégénérées.

15) Longueur des courbes Les courbes doivent avoir la même longueur que les arêtes qui reposent dessus.

16) Courbure des courbes Les courbes doivent avoir une forme similaire aux arêtes qui reposent dessus.

17) Degré des surfaces Des entités de degré inutilement élevé peuvent ralentir tout traitement informatique (rendu graphique ou analyse numérique).

18) Écart/recouvrement surface-surface Les extrémités de surfaces en contact doivent se correspondre.

19) Angle surface-surface Des angles trop aigus peuvent conduire à des modèles très difficilement analysables.

3.2.2 Validation des maillages

On suppose ici que les méthodes utilisées pour la génération des maillages garantissent que le maillage est valide. Chaque arête est donc connectée à deux nœuds, tous les nœuds sont connectés par des arêtes, il n'y a pas de recouvrement, ni de nœuds en dehors de la géométrie, etc. Il reste donc à analyser la qualité du maillage produit, afin d'assurer des simulations de qualité par la suite. Cela passe par l'analyse des cellules, à la fois une à une, et de manière globale.

Les cellules de mauvaise qualité présentent ces défauts :

- Cellule trop allongée (arêtes trop longues / courtes selon un axe)
- Cellule déformée (angles trop grands / petits)
- Volume négatif

Les maillages peuvent aussi être de mauvaise qualité lorsque les grandeurs caractéristiques des cellules sont trop différentes d'une cellule à l'autre. Plus de détails peuvent être trouvés dans l'article Dompierre *et al.* (2005).

3.3 Description des solutions apportées dans les modèles BREP

Les solutions appliquées pour détecter chacune de ces erreurs vont être décrites dans les sous-sections suivantes. Certaines solutions règlent plusieurs problèmes cités précédemment.

3.3.1 Écart arête-sommet et écart/recouvrement arête-arête

On s'intéresse ici à la proximité géométrique de toutes les entités de dimension 0.

Détection

Dans l'implémentation faite pour la librairie Pirate, l'unicité géométrique d'un sommet est assurée par plusieurs entités topologiques. Les entités impliquées sont les sommets, les sommets associés et les extrémités des arêtes et des arêtes associées. Le logiciel Vérificateur implémente les tests qui se chargent de vérifier que toutes ces entités sont géométriquement éloignées d'une distance inférieure à un seuil. Au sein de la librairie Pirate, chaque test géométrique est en réalité double. On compare les valeurs calculées à deux seuils, un seuil d'avertissement, et un seuil d'erreur. Les valeurs par défaut ne sont utilisables que pour des géométries ayant une dimension adaptée, elles ne seront précisées ici que pour être comparées entre elles.

Les sommets et arêtes étant les références dans le modèle, c'est à elles que l'on compare les autres entités. On compare donc (les valeurs entre parenthèses sont les seuils par défaut) :

- Les sommets associés au sommet (Avert. : 0.1, Err. : 1)
- Les extrémités des arêtes associées au sommet (Avert. : 1.10^{-6}, Err. : 0.01)
- Les sommets associés à l'extrémité de l'arête (Avert. : 0.01, Err. : 1)
- Le sommet à l'extrémité de l'arête (Avert. : 0.01, Err. : 1)

On remarque que le seuil arête associée-sommet est très strict. Ceci s'explique par le fait que les sommets sont en général présents dans les zones de concentration de contraintes, au niveau des discontinuités des pièces. On s'assure donc que les surfaces y sont précises (par l'intermédiaire des arêtes associées, qui les délimitent).

Ainsi, toutes les arêtes et arêtes associées se terminant en un sommet s'y terminent bien, ou sont détectées comme erronées. Il n'y a plus d'écart entre une arête et son sommet, ni entre deux arêtes qui se correspondent.

Visualisation

Vérificateur doit être utilisé sur un fichier de type PIE, et son résultat est fourni sous la forme d'un fichier de même format. La seule modification est l'ajout d'annotations aux entités qui comportent des erreurs. Comme on peut le voir dans le résultat d'exécution présent en annexe B (il s'agit du résultat d'exécution sur le cas test 1 présenté en figure 3.2), ces annotations se présentent sous la forme ("TYPE_DE_L'ERREUR_TROUVÉE" "Erreur_ou_Avertissement identifiant_de_l'entité_erronée valeur_erronée_calculée seuil_d'avertissement seuil_d'erreur").

Vérificateur fournit aussi une trace d'exécution listant toutes les erreurs trouvées, pouvant être enregistrée dans un fichier en format texte. Ces deux formes de résultat sont nécessaires pour les deux modes d'utilisation de la suite de logiciels basée sur la librairie Pirate. Les annotations permettent à TopoVisu, le logiciel de visualisation interactive 3D, de mettre en

évidence les erreurs détectées, afin que l'utilisateur puisse les identifier plus rapidement. La sortie sous forme de texte permet d'obtenir la liste des erreurs séparément des données si un traitement par lots est adopté.

Cas test

Le cas test 1 consiste en une unique arête, et deux sommets la délimitant. Un des sommets a été déplacé, et ne correspond donc pas à l'extrémité de l'arête. La description Pirate de ce cas test peut être trouvée en figure 3.2, il décrit un champ (Coo) de coordonnées géométriques contenant deux positions ponctuelles, et une géométrie (cube). Cette géométrie contient une courbe (crb1) définie à partir des points du champ, deux sommets (1 et 2) définis sur la courbe aux valeurs de paramètre 0.2 et 1, et enfin une arête (3) définie à l'aide de la courbe et délimitée par les deux sommets. L'erreur va venir du fait que la position géométrique de l'arête est identique à celle de la courbe, elle n'a pas été réduite à l'intervalle $[0.2\,;\,1]$ qui aurait correspondu à la position des sommets.

La trace d'exécution (voir figure 3.3) liste les tests effectués, les valeurs des deux seuils (avertissement et erreur) utilisés, ainsi que les erreurs trouvées pour chaque test. Dans la figure 3.4, on peut voir le résultat des tests dans TopoVisu. On constate que l'erreur est attribuée à l'arête, qui est colorée en rouge (dans la vue 3D ainsi que dans la liste des entités). Le comportement ici est général, on attribut toujours l'annotation à l'entité de dimension minimum englobant la totalité des entités concernées par l'erreur.

```
Champ <double >Coo() =
{
  0 0 0
  1 0 0
};

Geometrie cube() =
{
  Courbe crb1( 2, [ <int>1 2 ], Coo%3 );

  Sommet 1( crb1( 0.2 ) );
  Sommet 2( crb1( 1 ) );

  Arete 3( crb1 ) = { 1 2 };
};
```

Figure 3.2 Données décrivant les entités Pirate du cas test 1.

```
-->Verifier Distance Sommets vs Aretes de "la geometrie "cube"
               Avertissement: 0.0001
               Erreur       : 0.01
// ATTENTION ERREUR : ---***---TOLERANCE_SOMMETS_ARETES----****---- :
// Verifiez la definition du (sommet=1, arete=3)


-->Verifier: Distance Sommets vs Aretes Associees de "la geometrie "cube"
               Avertissement: 1e-06
               Erreur       : 0.01
-->Verifier: Distance Sommets Associes Aretes de  "la geometrie "cube"
               Avertissement: 0.01
               Erreur       : 1
```

Figure 3.3 Journal d'exécution des tests de Vérificateur sur le cas test 1.

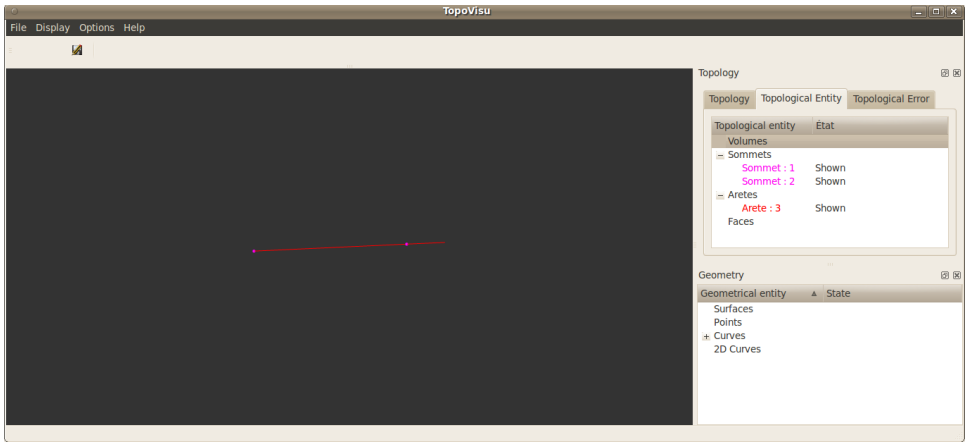

Figure 3.4 Visualisation des erreurs du cas test 1 sur TopoVisu.

3.3.2 Longueur des arêtes, écart face-arête, courbure des arêtes et proximité arête-arête

Le concept commun des erreurs traitées ici est la proximité géométrique des entités de dimension 1.

Détection

Dans la librairie Pirate, une arête est unique. Ce qu'il faut vérifier, c'est que les arêtes associées lui correspondent. Les tests existants se chargent donc de :

- comparer les longueurs d'une arête et de ses arêtes associées (Avert. : 0.01, Err. : 1).
- comparer différents points se correspondant le long de l'arête et des arêtes associées (Avert. : 0.0001, Err. : 0.01).

Les différents points sont échantillonnés à des ratios de longueurs d'arc identiques sur les arêtes et arêtes associées, ce qui permet de vérifier l'écart face-arête, la courbure identique des arêtes associées avec celle de l'arête et la proximité arête-arête. Les seuils du second test sont plus petits afin de prendre en compte l'échelle réduite à laquelle ce test est effectué, on travaille sur des portions d'arêtes, non sur les arêtes elles-mêmes.

Visualisation

De la même manière que précédemment, Vérificateur fournit un journal d'exécution et des annotations qui peuvent être visualisées dans TopoVisu. Les annotations sont ajoutées à la ou aux faces impliquées dans l'erreur détectée.

Cas test

Le cas test 2 (voir annexe C) consiste en un cube, dont une arête est trop courte. Cela entraîne donc une erreur au niveau de la comparaison des longueurs pour les deux faces qui s'y joignent, ainsi qu'une erreur de courbure car les points échantillonnés ne sont plus superposés. On peut observer ces erreurs détectées dans le journal de Vérificateur (Figure 3.5) et dans TopoVisu (Figure 3.6)

Dans le cas test 3 (voir annexe D), un trou a été créé au niveau d'une arête du cube, tout en conservant les longueurs des arêtes et arêtes associées identiques. Le journal (Figure 3.7) et la visualisation 3D (Figure 3.8) montrent bien que l'erreur a tout de même été détectée.

On peut donc détecter les différences de longueur sur une arête, les écarts face-arête, les problèmes de non-proximité arête-arête et les différences de courbure des arêtes.

```
-->Verifier:Difference Longueur Aretes vs Aretes Associees de "la geometrie "Cube"
               Avertissement: 0.01
               Erreur       : 1
// ATTENTION Avertissement : ---***---TOLERANCE_DIFFERENCE_LONGUEUR_ARETE_ARETE_ASSOCIEE----****---- :
// Verifiez la definition de l'arete associee (arete=9, surface=surface_1)

// ATTENTION Avertissement : ---***---TOLERANCE_DIFFERENCE_LONGUEUR_ARETE_ARETE_ASSOCIEE----****---- :
// Verifiez la definition de l'arete associee (arete=9, surface=surface_2)

-->Verifier: Distance point/point Arete vs Arete Associee de  "la geometrie"Cube"
               Avertissement: 0.0001
               Erreur       : 0.01
// ATTENTION ERREUR : ---***---TOLERANCE_DISTANCE_PP_ARETE_ARETE_ASSOCIEE----****---- : 0.2
// Verifier la definition de l'arete associee (arete=9 sur la face 21, surface=surface_1)


// ATTENTION ERREUR : ---***---TOLERANCE_DISTANCE_PP_ARETE_ARETE_ASSOCIEE----****---- : 0.2
// Verifier la definition de l'arete associee (arete=9 sur la face 22, surface=surface_2)


-->Verifier: Distance Sommets vs Sommets Associes de la geometrie Cube"
               Avertissement: 0.1
               Erreur       : 1
--> Verifier: Arete coincide avec Arete Associee de " la geometrie "Cube"
               Avertissement: 0.0001
               Erreur       : 0.01
// ATTENTION ERREUR : ---***---TOLERANCE_ARETES_ARETES_ASSOCIEES----****---- :
// Verifier la definition de (arete=9, surface=surface_1)


// ATTENTION ERREUR : ---***---TOLERANCE_ARETES_ARETES_ASSOCIEES----****---- :
// Verifier la definition de (arete=9, surface=surface_2)
```

Figure 3.5 Journal d'exécution des tests de Vérificateur sur le cas test 2.

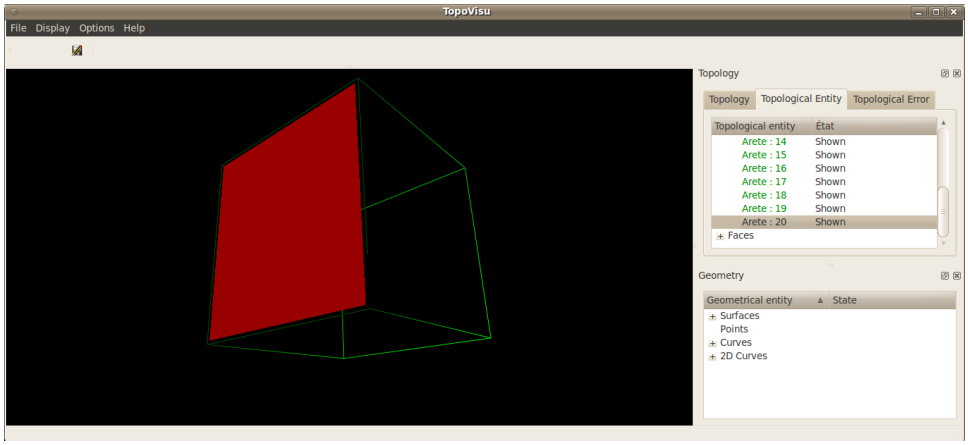

Figure 3.6 Visualisation des erreurs du cas test 2 sur TopoVisu.

```
-->Verifier Distance Sommets vs Aretes de "la geometrie "Cube"
               Avertissement: 0.01
               Erreur       : 1
-->Verifier: Distance Sommets vs Aretes Associees de "la geometrie "Cube"
               Avertissement: 1e-06
               Erreur       : 0.01
-->Verifier: Distance Sommets Associes Aretes de  "la geometrie "Cube"
               Avertissement: 0.01
               Erreur       : 1
-->Verifier:Difference Longueur Aretes vs Aretes Associees de "la geometrie "Cube"
               Avertissement: 0.01
               Erreur       : 1
-->Verifier: Distance point/point Arete vs Arete Associee de  "la geometrie"Cube"
               Avertissement: 0.0001
               Erreur       : 0.01
// ATTENTION ERREUR : ---***---TOLERANCE_DISTANCE_PP_ARETE_ARETE_ASSOCIEE----****---- : 0.25373
// Verifier la definition de l'arete associee (arete=9 sur la face 21, surface=surface_1)


-->Verifier: Distance Sommets vs Sommets Associes de la geometrie Cube"
               Avertissement: 0.1
               Erreur       : 1
```

Figure 3.7 Journal d'exécution des tests de Vérificateur sur le cas test 3.

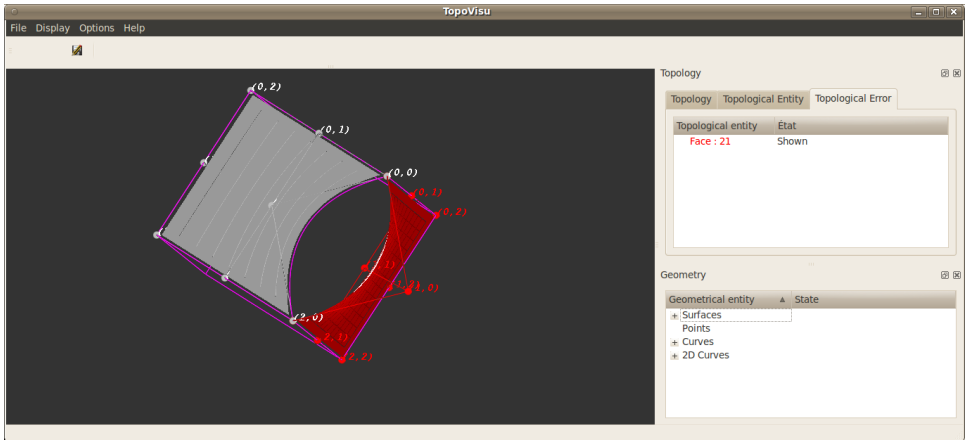

Figure 3.8 Visualisation des erreurs du cas test 3 sur TopoVisu.

3.3.3 Écart/recouvrement face-face et consistance arête-boucle

On va vérifier ici que les frontières des faces sont bien définies.

Détection

Une partie des vérifications nécessaires a déjà été présentée en section 3.3.2, on sait que les arêtes associées de deux faces juxtaposées sont proches d'une même arête. Il n'y a donc pas de trou ou de recouvrement entre celles-ci. Ici, on s'assure qu'une face est topologiquement correcte. C'est-à-dire que l'on vérifie si les arêtes associées d'une face reposent bien sur la surface qui supporte cette face et qu'elles forment bien des boucles fermées. Les seuils du premier test ont un niveau habituel de précision (Avert. : 0.01, Err. : 1). Pour le second aspect, on s'assure simplement que lorsqu'on parcourt une boucle, le sommet de fin de chaque arête correspond au sommet de début de l'arête suivante, les tests effectués précédemment (Cf section 3.3.1) nous garantissent alors la justesse de la géométrie.

Visualisation

Vérificateur fournit le journal d'exécution et les annotations pour TopoVisu. Les annotations sont ajoutées à la face détectée comme non topologiquement valide.

Cas test

Le cas test 4 contient une unique face carrée, dont on a raccourci une arête et l'arête associée correspondante. Le code de ce cas test est présent en annexe E. Le résultat fourni par Vérificateur peut être vu en figure 3.9, et l'affichage de TopoVisu en figure 3.10. On peut voir le type d'annotations ajoutées aux entités erronées dans la fenêtre "General Information" de TopoVisu, cela permet de préciser le type de l'erreur rencontrée.

```
-->Verifier: Distance Sommets Associes Aretes de  "la geometrie "gest"
               Avertissement: 0.01
               Erreur       : 1
-->Verifier:Difference Longueur Aretes vs Aretes Associees de "la geometrie "gest"
               Avertissement: 0.01
               Erreur       : 1
-->Verifier: Distance point/point Arete vs Arete Associee de  "la geometrie"gest"
               Avertissement: 0.0001
               Erreur       : 0.01
-->Verifier: Distance Sommets vs Sommets Associes de la geometrie gest"
               Avertissement: 0.1
               Erreur       : 1
--> Verifier: Arete coincide avec Arete Associee de " la geometrie "gest"
               Avertissement: 0.0001
               Erreur       : 0.01
 --> Verifier: Face topologiquement valide de  "la geometrie "gest"
               Avertissement: 0.01
               Erreur       : 1
//Warning: la boucle n'est pas topologiquement valide.
//Warning: les aretes sont mal utilisees dans la boucle.
//Warning: le sommet de fin de la (2)ieme arete de la boucle devrait correspondre au sommet de depart de la (3)ieme arete de la boucle
//Warning: la (2)ieme arete de la boucle est - 63(   - srf1( crb3p ) )
//Warning:   dont le sommet de fin est 3
//Warning: la (3)ieme arete de la boucle est - 62( srf1( crb4p ) )
//Warning:   dont le sommet de depart est 31
// ATTENTION : la face(101) n'est pas toologiquement valide
// ATTENTION : la (0)ieme boucle de la face n'est pas valide
//ATTENTION ERREUR: ---***---TOLERANCE_ARETES_FACES----****---- :
// Verifier la definition de la (face=101, surface=srf1)
```

Figure 3.9 Journal d'exécution des tests de Vérificateur sur le cas test 4.

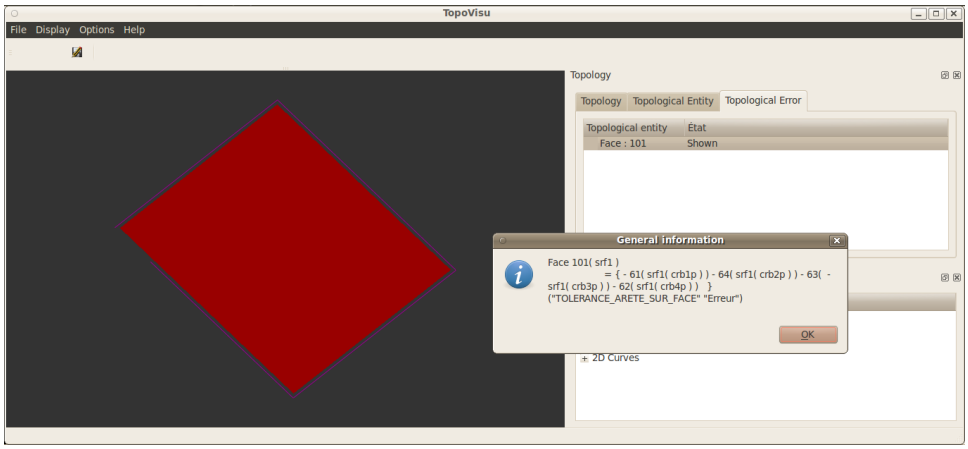

Figure 3.10 Visualisation des erreurs du cas test 4 sur TopoVisu.

3.3.4 Nombre d'utilisation des arêtes

Ce test permet de valider la fermeture des coquilles.

Détection

Ce type d'erreur est particulier aux modèles basés sur les variétés, il ne doit pas être pris en compte dans le cas des modèles BREP au sens plus large. Malgré le fait que Pirate supporte à la fois les modèles basés ou non sur les variétés, un test existe pour détecter ces erreurs. Le test est assez simple, on parcourt toutes les arêtes de chaque volume, et on s'assure que chacune est parcourue une fois dans chaque sens.

Dans les modèles qui ne sont pas des variétés de dimension 3, les arêtes peuvent être utilisées un nombre quelconque de fois (une seule fois pour les objets infiniment fins, ou

plusieurs suivant le nombre de faces qui s'y rejoignent). Il n'est donc pas possible d'effectuer un test sur le nombre d'utilisations des arêtes.

Visualisation

De la même manière que dans les cas précédents, Vérificateur fournit le journal et les annotations interprétables.

Cas test

Le cas test 5 (en annexe F) se compose d'un cube, dont on a retiré une face. On peut observer le résultat des tests sur les figures 3.11 et 3.12.

```
-->Verifier: Distance point/point Arete vs Arete Associee de  "la geometrie"gest"
                Avertissement: 0.0001
                Erreur       : 0.01
-->Verifier: Distance Sommets vs Sommets Associes de la geometrie gest"
                Avertissement: 0.1
                Erreur       : 1
--> Verifier: Arete coincide avec Arete Associee de " la geometrie "gest"
                Avertissement: 0.0001
                Erreur       : 0.01
 --> Verifier: Face topologiquement valide de  "la geometrie "gest"
                Avertissement: 0.01
                Erreur       : 1
--> Verifier: Le volume est ferme pour la geometrie "gest"
                Erreur       : si une arete n'est pas parcourue dans les deux sens
                ERREUR!!!:L'arete 9 est utilisee 1 fois
                ERREUR!!!:L'arete 10 est utilisee 1 fois
                ERREUR!!!:L'arete 11 est utilisee 1 fois
                ERREUR!!!:L'arete 12 est utilisee 1 fois
// ATTENTION  : GEOMETRIE NON VALIDE
```

Figure 3.11 Journal d'exécution des tests de Vérificateur sur le cas test 5.

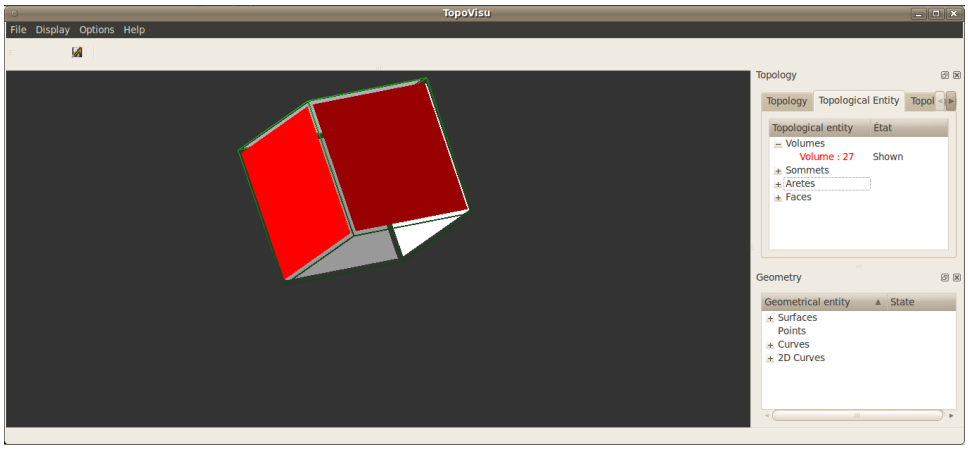

Figure 3.12 Visualisation des erreurs du cas test 5 sur TopoVisu.

3.3.5 Longueur des courbes, courbure des courbes et écart/recouvrement surface-surface

Dans la bibliothèque Pirate, les entités géométriques et leur position dans l'espace sont deux concepts séparés. Les courbes et les surfaces ne jouent donc aucun rôle dans le modèle, elles ne servent qu'à positionner géométriquement les entités topologiques (arêtes et faces) à leur création. On ne vérifie donc pas leurs propriétés directement, elles sont vérifiées uniquement au travers des entités topologiques.

3.3.6 Angle arête-arête, degré des faces, angle face-face, faces/surfaces nulles, degré des surfaces et angle surface-surface

Les erreurs qui concernent les angles arête-arête, le degré des faces et des surfaces, les angles face-face ou surface-surface et les faces/surfaces dégénérées sont considérées comme pouvant être des choix du concepteur et ne sont donc pas détectées dans notre implémentation. Elles pourront avoir une influence sur le processus de maillage par la suite, car c'est dans ces zones que la densité de maillage risque de varier si besoin.

3.3.7 Consistance face-face

Pour ce test, deux situations sont à distinguer. Dans le cas des modèles basés sur les variétés, il ne peut s'agir que d'un assemblage de deux volumes en contact, un volume ne pouvant utiliser la même face deux fois, ni dans la même coquille, ni dans deux coquilles séparées. La librairie Pirate ne contient pas de classe modélisant les assemblages, le test de correspondance de faces n'est donc pas implémenté. De plus, ces deux faces peuvent ne pas se correspondre exactement, et n'être qu'en contact imparfait, on ne peut donc pas vérifier leur correspondance exacte systématiquement.

Pour le cas des modèles qui ne sont pas des variétés 3D, la manière de modéliser une face interne consiste à attribuer deux faces associées à la face interne définie. Contrairement aux arêtes associées, les faces associées n'ont pas de positions 3D propres dans l'implémentation Pirate, car il n'y a pas de notion d'espace. Leur position est uniquement celle de la face à laquelle elles sont associées, l'information est unique. Deux volumes partageant une même face se correspondront donc nécessairement sur cette face, puisque la manière de les déclarer joints et de déclarer la position de la face dans chaque volume est une unique opération.

3.4 Description des solutions de validation de maillages

Détection

Dans la suite de logiciels Pirate, il existe deux méthodes de détection complémentaires des erreurs dans les maillages. En effet, étant donné le nombre de cellules pouvant former un maillage, annoter chaque cellule défectueuse de la même manière qu'on annote une entité topologique risque de rendre le maillage très lourd en mémoire. Les algorithmes d'analyses s'accomplissent donc à deux endroits : dans Check_mesh, pour créer le rapport d'analyse statistique correspondant à un traitement par lots, et dans TopoVisu afin de visualiser plus en détail chaque cellule défectueuse.

Dans Check_mesh, on calcule les grandeurs suivantes, et on les compare à deux seuils (avertissement et erreur) :

- Le volume / l'aire des cellules
- Les angles des cellules
- Le ratio d'aspect des cellules

Dans TopoVisu, on compare les grandeurs suivantes à un seuil modifiable par l'utilisateur :

- Le volume / l'aire des cellules
- Les angles des cellules
- Le ratio d'aspect des cellules

Dans les deux logiciels, les tests sont identiques. Pour une cellule tétraédrique, le ratio d'aspect est : $\frac{12*(3*Volume)^{2/3}}{\sum_{i=1}^{6}(L_i)^2}$ où les L_i sont les longueurs des côtés du tétraèdre. L'angle calculé est l'angle dièdre dont l'expression peut être trouvée dans Dompierre *et al.* (2005), à l'équation (7). Pour les autres types de cellules (hexaèdres, prismes), on calcule ces valeurs pour les sous-tétraèdres formés sur les nœuds des cellules. Le résultat final est le minimum des valeurs des tétraèdres multiplié par un facteur. Ce facteur permet de rendre compte du fait qu'une cellule parfaite n'est pas formée de tétraèdres parfaits.

Visualisation

Check_mesh fournit une vision d'ensemble des résultats, sous la forme d'un rapport statistique. Il conserve le nombre de cellules étant détectées par chaque seuil, ainsi que les positions du minimum et du maximum de chaque grandeur évaluée.

TopoVisu permet de voir les cellules détectées par les seuils en colorant celles-ci sur la vue 3D. Le seuil modifiable de manière interactive permet de suivre l'évolution d'un paramètre sur l'ensemble du maillage.

Cas test

Le cas test 6 présente un maillage dont certaines cellules sont erronées (une a un volume négatif, vingt ont un angle aigu et six ont un mauvais ratio d'aspect). La figure 3.13 présente le rapport d'erreur produit par Check_mesh. Le résultat de l'activation des seuils dans TopoVisu est présenté dans la figure 3.14.

```
================================================================================
 Mesh ID:           Turbgeom
                    96582 vertices, 4032 3D-elements
================================================================================

ERROR: Some checks failed, the mesh is invalid.
Mesh summary:
Mesure: "volume"
  Min volume : -0.00278139 at location 1170 (1170:[9 12 7]) in zone1354
  Max volume : 9.48101 at location 1861 (1861:[14 1 12]) in zone1354
  Warning volume Count : 1 / 4032       ( threshold :  1e-12 )
  Error volume Count : 1 / 4032 ( threshold :  0 )

Mesure: "angle"
  Min angle : 0.0358449 at location 1002 (1002:[9 12 6]) in zone1354
  Max angle : 179.998 at location 1506 (1506:[9 12 9]) in zone1354
  Warning angle Count : 20 / 4032       ( threshold :  5 )

Mesure: "form"
  Min form : 0.00104076 at location 1337 (1337:[8 12 8]) in zone1354
  Max form : 0.969269 at location 3967 (3967:[6 8 24]) in zone1354
  Warning form Count : 6 / 4032 ( threshold :  0.01 )
```

Figure 3.13 Journal d'exécution des tests de Check_mesh sur le cas test 6.

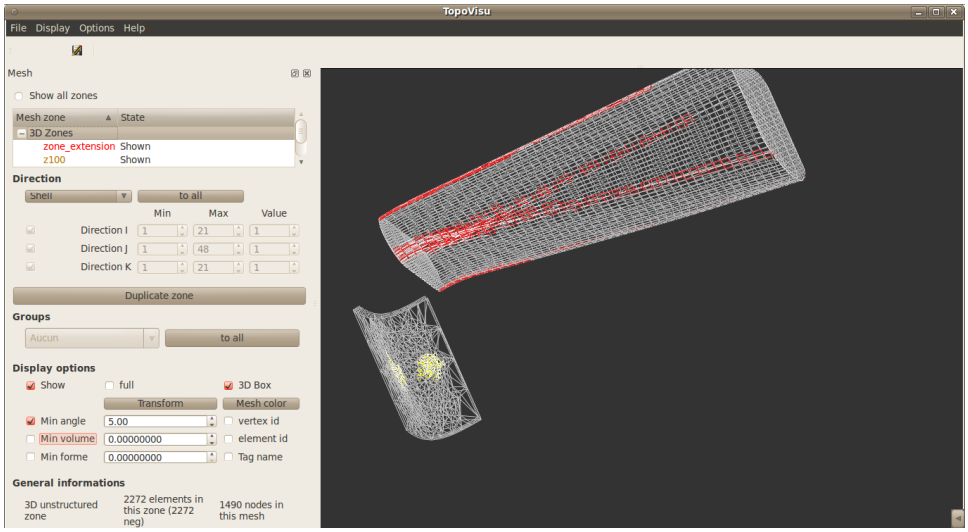

Figure 3.14 Visualisation des erreurs du maillage du cas test 6 sur TopoVisu.

CHAPITRE 4

CONSERVATION ET COMMUNICATION DES DONNÉES 3D

Une fois les erreurs trouvées dans les modèles et les maillages, il faut en général communiquer ces erreurs à d'autres personnes, soit pour archivage, soit pour résoudre les problèmes à une autre étape de la conception. Comme signalé à la section 2.5, le format choisi pour cela est le PDF 3D, avec la composante PRC. Ce chapitre commence donc par une description de ce format, qui regroupe un point sur les spécifications, sur l'implémentation partielle qui a servie de base au projet, ainsi que sur les correspondances qui ont été effectuées entre ce format et le format natif des applications Pirate (PIE). La suite du chapitre détaille l'aspect technique du travail effectué.

4.1 Description du standard PDF 3D - PRC

Le format appelé PDF 3D présente deux composantes, la première est la partie PDF, largement utilisée dans le monde pour la transmission de documents. Cet aspect est assez secondaire dans le travail réalisé ici, il n'a pas été utilisé en détail, il ne sert que de capsule permettant au second aspect, les modèles 3D, d'être interprété. Comme signalé précédemment (Cf 2.5), les modèles 3D sont importés au format PRC afin de conserver la structure des modèles, héritée de la norme STEP. Ce format est intégré sous forme de fichiers binaires au sein du fichier PDF. La technique utilisée ici est l'utilisation d'une bibliothèque [1] lors de la compilation du fichier LaTeX permettant l'obtention du PDF.

4.1.1 Description des spécifications

Selon les spécifications publiées par Adobe Systems, les modèles PRC sont créés en sérialisant les informations concernant chaque entité du modèle, puis en compressant les données. Les spécifications se contentent de fournir les fonctions permettant de sérialiser les informations de chaque entité. Étant donné que chacune de ces fonctions fait appelle à une fonction identique de la classe parente, il est possible de reconstruire l'arbre d'héritage des fonctions de proche en proche. C'est ce travail qui a été commencé par les développeurs du logiciel Asymptote, dont on a extrait et complété la bibliothèque pour répondre à nos besoins.

1. La bibliothèque "movie15" Grahn (2012)

4.1.2 Asymptote et sa bibliothèque PRC

Asymptote est un logiciel de tracé de fonctions mathématiques. La partie qui nous intéresse ici est la bibliothèque développée pour exporter ces représentations géométriques vers le PDF 3D. Étant donné l'utilisation de la bibliothèque, et la densité des spécifications du format, seule une partie des classes a été entièrement développée (principalement les classes de géométrie). Il s'agit donc d'une implémentation partielle des spécifications, qu'il a fallu compléter pour atteindre notre but : l'exportation des modèles BREP.

Le diagramme des classes présent en figure 4.1.2 donne une idée de la structure des classes de cette bibliothèque. Toutes les classes héritant de PRCBaseGeometry concernent l'implémentation des classes géométriques, les classes héritant de PRCBaseTopology implémentent la topologie du modèle (ces classes sont détaillées dans le diagramme de la figure 4.3 car elles sont au cœur du présent travail), enfin les classes héritant de ContentPRCBase sont des classes-outils, permettant par exemple de gérer les paramètres d'affichage des objets 3D.

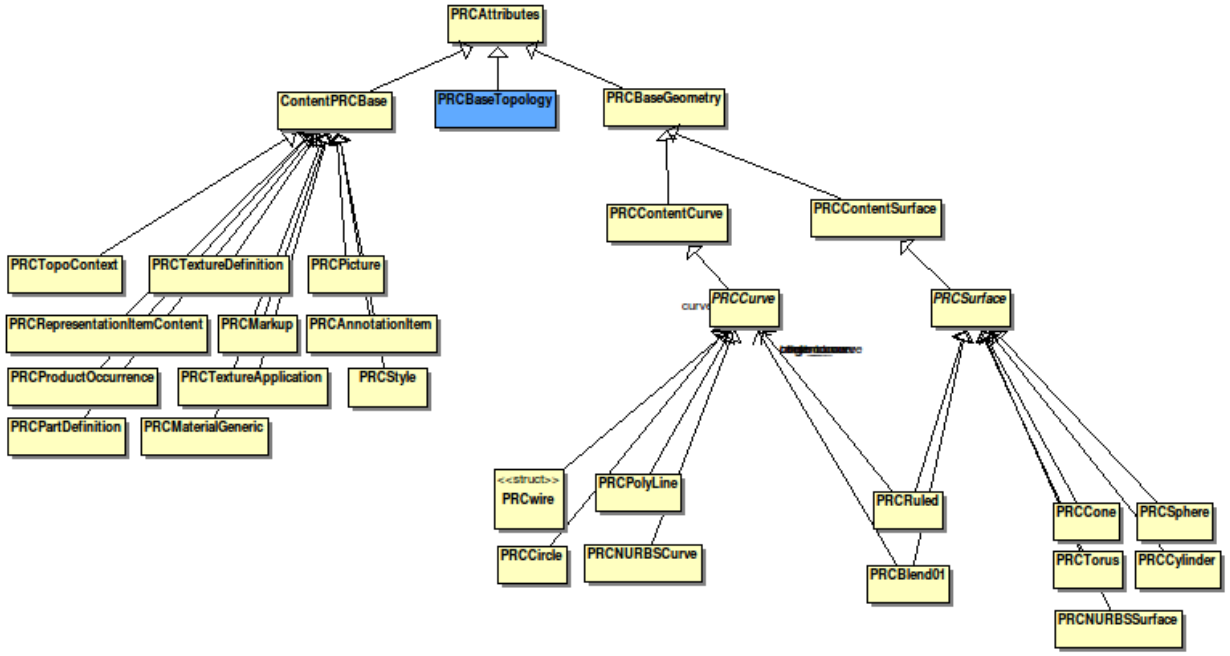

Figure 4.1 Diagramme de classes de la structure de la bibliothèque PRC.

Un autre ensemble de classes (dont on peut avoir un aperçu sur le diagramme 4.1.2) se charge de gérer la structure du fichier PRC créé. Les propriétés des objets géométriques et topologiques sont transmises par sérialisation dans un objet de type PRCbitStream, qui assure le lien entre les deux arbres.

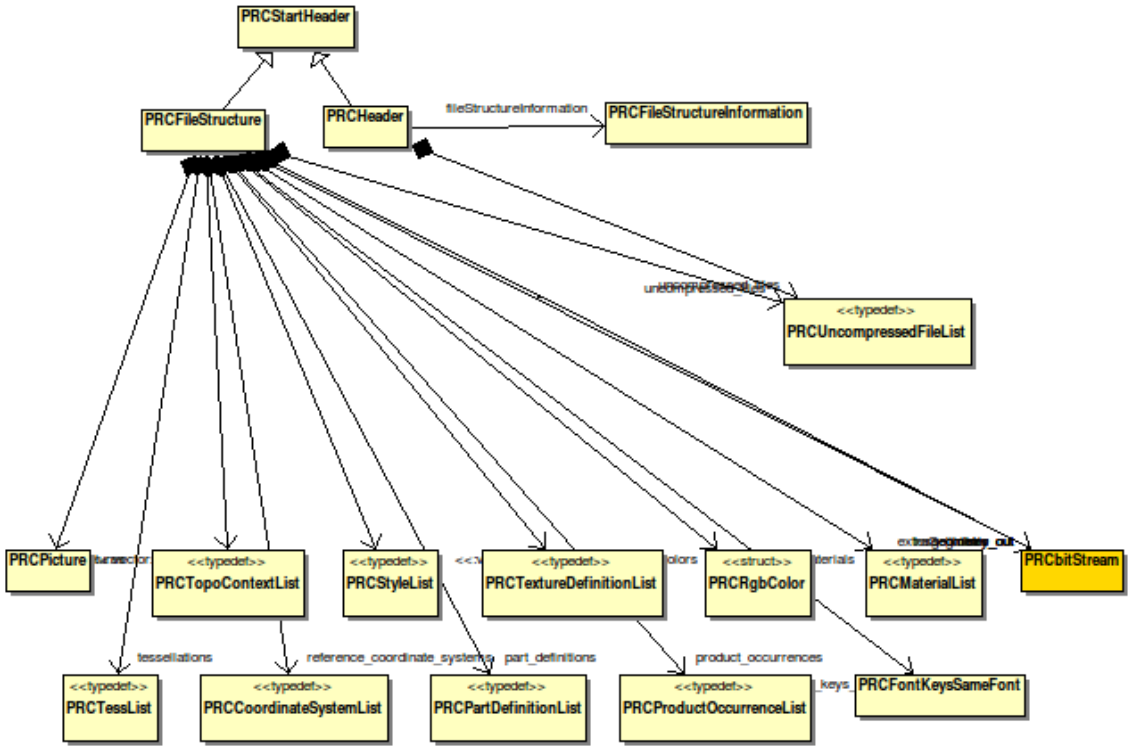

Figure 4.2 Diagramme de classes gérant la structure du fichier PRC.

4.1.3 Lien entre Pirate et PRC

Les classes géométriques et topologiques implémentées dans les deux bibliothèques (Pirate et PRC) ne sont pas identiques. Il a donc fallu créer une correspondance entre les deux implémentations.

Correspondance des classes géométriques

Au niveau des classes géométriques, le lien a été très facile à effectuer. Les seules entités à exporter sont les suivantes :

1. Points
2. Courbes NURBS
3. Surfaces NURBS

Ces entités ont toutes une implémentation exacte dans les deux formats.
Dans la bibliothèque Pirate :

1. PG_POINT

2. PG_CRB_NURBS
3. PG_SRF_NURBS

Dans la bibliothèque PRC :

1. PRCVector3D
2. PRCNURBSCurve
3. PRCNURBSSurface

Une fois l'analogie conceptuelle effectuée, il a été assez simple ce créer des fonctions de conversion. Il est tout de même intéressant de noter que certains choix d'implémentation dans les structures de données peuvent compliquer la conversion. On notera par exemple l'inversion nécessaire sur les deux dimensions du tableau stockant les points de contrôles des surfaces lors du passage d'une bibliothèque à l'autre.

Correspondance des classes topologiques

Au niveau de la topologie, les classes ne se correspondent pas exactement, il a donc fallu identifier celles qui se correspondent, et s'assurer que toutes les informations sont transmises. Pour observer ces correspondances, le plus simple est de comparer les deux diagrammes de classes représentant la topologie dans les deux bibliothèques (figures 4.3 et 4.4). Les classes ont été placées de manière similaire, et la suite va revenir sur les différences principales entre les deux implémentations.

La première différence est l'absence du concept de face associée dans l'implémentation PRC. Ce concept est présent dans la bibliothèque Pirate pour conserver le principe de "vérité" détenue par les entités non associées (sommets, arêtes et faces) à tous les niveaux de dimension. PRC peut s'en passer aisément, puisqu'au sein d'un volume, chaque face ne peut être utilisée qu'une seule fois, et qu'il est possible de créer deux volumes sur une unique face. Les faces associées ne sont que des interfaces dans la librairie Pirate, cette interface n'existe plus en PRC. Cette suppression n'est possible qu'à cause de l'absence de notion d'espace dans les modèles BREP.

On remarque aussi qu'un plus grand nombre d'entités a besoin d'être déclaré comme "topologie" dans la bibliothèque PRC. Cela est principalement dû à la nature linéaire du fichier, toutes les entités doivent hériter d'une des trois classes principales (ContentPRCBase, PRCBaseTopology, PRCBaseGeometry) pour présenter le marqueur correspondant au début de leur série de bits.

Enfin, on remarque que Pirate différencie les concepts d'entité géométrique et de position géométrique. Ainsi, les différentes entités topologiques n'ont pas de lien direct avec la géométrie sous-jacente qui leur sert de support, uniquement avec la position de ce support.

Cela permet aussi de changer l'origine de la position (un sommet va pouvoir être défini en un point évalué d'une courbe lors de sa création). La bibliothèque PRC préfère unifier les deux notions de position et d'entité géométrique (pour créer un sommet sur une courbe, il va falloir créer le point d'évaluation en plus de la courbe).

4.2 Implémentation

4.2.1 Création des classes nécessaires

La première étape de la réalisation a été la création des classes manquantes dans la bibliothèque d'Asymptote, nécessaires à l'exportation des modèles BREP. Les classes qui existaient déjà étaient celles qui étaient nécessaires à la représentation de surfaces, Asymptote étant un logiciel de représentation de fonctions mathématiques.

Les classes créées sont basées sur les variables nécessaires au sein des fonctions "serialize**NomDeLaClasse**" décrites dans les spécifications. Les classes qui ont été définies sont donc celles qui concernent la topologie des modèles : sommets (PRCUniqueVertex), arêtes (PRCEdge), arêtes associées (PRCCoEdge) et boucles (PRCLoop). Les faces (PRCFace) ont aussi dû être réécrites partiellement, afin de prendre en compte les liens avec les autres composantes de la topologie. Précédemment, elles ne servaient que de capsules aux surfaces, ces dernières ne pouvant pas être intégrées directement dans un modèle PRC. Un diagramme présentant les liens entre ces différentes classes est présent sur la figure 4.3.

Il a ensuite fallu créer le convertisseur entre le format Pirate et le format PRC. Pour cela, une nouvelle classe a été créée : PRCMaker. Cette classe contient une fonction membre pour chaque entité exportable qui s'assure de récupérer toutes les données nécessaires à cette exportation (principalement les caractéristiques mathématiques des entités) dans les objets Pirate. Ces fonctions appellent ensuite d'autres qui se chargent de construire les entités PRC. Ces dernières ont été modifiées ou définies dans la bibliothèque d'Asymptote, et renvoient des pointeurs vers les entités PRC créées, ce qui permet de recréer l'arbre de topologie en format PRC au sein de PRCMaker. La totalité de l'algorithme de sauvegarde peut être exécutée sur une géométrie à l'aide de la fonction générale "ajouterGeometrieStructure" qui se charge de créer la structure PRC à partir du membre "body" (de type PRCBrepData) de PRCMaker, qui sert de racine à l'arbre topologique. La méthode "PRCMaker : :enregistrerStructure" permet ensuite de créer le fichier PRC.

Précédemment à la création des classes de topologie au sein de la bibliothèque extraite du logiciel Asymptote, une tentative d'exportation de la géométrie des modèles Pirate avait été effectuée. Les fonctions nécessaires à cette exportation sont toujours présentes dans PRCMaker, et peuvent être appelées sur une géométrie par la fonction "ajouterGeometrie" qui se

charge de créer chacun des éléments dans une structure PRC. Cette structure est définie au sein de la bibliothèque d'Asymptote, et peut être enregistrée en un fichier PRC en utilisant la méthode "PRCMaker : :enregistrer".

4.2.2 Intégration au sein de TopoVisu

Ces nouvelles classes créées ont ensuite dû être interfacées avec celles déjà existantes au sein de TopoVisu. Pour cela, une nouvelle classe (PRCMaker) a été créée. Elle se charge d'effectuer la conversion depuis le format natif de TopoVisu et de la bibliothèque Pirate, vers le format PRC. L'intérêt principal de créer une classe indépendante pour l'exportation en PRC est d'avoir un programme modulable, facilement entretenu, et dont les composants sont indépendants.

Il est nécessaire d'intégrer les nouvelles fonctionnalités dans le programme déjà existant. Étant donné qu'un module de sauvegarde existait déjà dans TopoVisu, nous avons décidé de greffer la fonctionnalité d'exportation à ce widget. Ainsi, au niveau de l'interface, un simple menu déroulant a été ajouté à la fenêtre de sauvegarde. Ce menu permet de choisir le format de sauvegarde. Pour le moment, sept options sont disponibles, mais on peut imaginer que d'autres possibilités seront disponibles à l'avenir, suivant les besoins des concepteurs. Le premier format disponible est évidemment le format natif de l'application. Il est le choix par défaut, ce qui permet à un utilisateur habitué à l'ancienne version de TopoVisu d'obtenir une sauvegarde dans le format qu'il a l'habitude d'utiliser s'il ne remarque pas le menu déroulant.

Les trois options suivantes sont l'enregistrement des géométries du modèle en PDF, de la totalité du modèle , ou les deux (géométrie et modèle complet) en PDF. L'intérêt de pouvoir exporter les modèles avec et sans la topologie est de pouvoir observer les différences entre les deux, car l'interprétation de la topologie par le logiciel d'ouverture de PDF peut changer la géométrie elle-même. Par exemple, la différence entre les figures 3D [2] 4.5 et 4.6 est l'exportation ou non de la topologie, le modèle initial contenant une arête erronée (trop courte de 20% en comparaison à la géométrie). De plus, le modèle apparait comme un seul "bloc" dans l'arborescence de visualisation. Sauvegarder la géométrie séparément permet d'observer certaines parties indépendamment.

Enfin, les trois dernières options sont identiques aux trois précédentes, mais se contentent de générer le fichier PRC, et ne l'intègrent pas à un PDF. Ceci est particulièrement utile si l'utilisateur de TopoVisu ne souhaite qu'intégrer le modèle 3D dans un rapport, et non l'avoir seul dans un fichier PDF séparé.

La différence principale entre les trois premières options et les trois suivantes sont l'utili-

2. Pour visualiser le contenu, cliquer dans le cadre pour l'activer, puis cliquer droit → options de pièce → contenu, pour recadrer la caméra sur les objets à afficher.

sation de LaTeX pour générer un fichier PDF. Ce fichier PDF ne contient qu'un grand cadre noir qui limite la fenêtre de visualisation du modèle 3D (le cadre permet de voir la fenêtre de visualisation, qui est blanche si non activée).

Le script LaTeX utilisé pour générer le fichier PDF (Cf Figure 4.7) fait appel à différentes bibliothèques propres au langage. Celles-ci permettent d'inclure le fichier PRC dans le PDF. Les commandes utilisées peuvent être réutilisées dans un autre script LaTeX (un rapport par exemple) afin d'y intégrer le modèle 3D. Pour réutiliser ce code source dans un autre document LaTeX, il suffit de :

- S'assurer que les deux bibliothèques (”movie15” et ”hyperref”) sont chargées dans l'entête du document,
- Placer la commande ”includemovie”, avec les paramètres souhaités pour la taille de la fenêtre (en bleu sur la figure 4.7) et le fichier à intégrer (en vert), où l'on souhaite que la fenêtre de visualisation apparaisse,
- (Optionnel) Utiliser les commandes ”fbox” et ”noindent” pour encadrer au plus proche la fenêtre de visualisation.

La totalité des objets 3D présents dans ce mémoire a été intégrée en utilisant cette méthode.

4.3 Justification des choix de conception

Lors de l'extension des classes de la bibliothèque Asymptote, certains choix de conception ont été effectués. L'objectif ici est d'expliquer le raisonnement derrière ceux-ci. Une grande quantité de fonctions initialement présente dans la bibliothèque extraite d'Asymptote sont encore présentes dans la version finale de TopoVisu, même si elles ne sont pas utilisées dans ce contexte. Ce choix permet de conserver la compatibilité de notre bibliothèque modifiée avec celle utilisée dans Asymptote. À terme, l'objectif est que les modifications effectuées dans la bibliothèque d'Asymptote puissent être réintégrées dans le logiciel, et utilisées par des développeurs tiers.

Un autre choix a été de réunir la totalité des fonctions qui assurent la conversion du format natif de TopoVisu (.PIE) vers le format PRC dans une seule classe. Ces fonctions auraient pu être séparées dans différentes classes en fonction des objets mathématiques (volumes, surfaces, courbes, ou points) afin d'avoir une plus grande modularité. La raison initiale du regroupement en une classe était de réussir à limiter la redondance d'information en retenant dans une structure de données (de type map) les entités qui ont déjà été rencontrées. Par exemple, deux faces se rejoignant sur la même arête devraient avoir un pointeur vers le même objet arête, et non deux différents. Ceci était accompli en associant le nouvel objet PRC créé

à l'identifiant unique de l'objet PIE (cet identifiant servant de clé dans la map).

Cette conception est malheureusement incompatible avec la bibliothèque importée d'Asymptote. Les destructeurs des classes de degrés supérieurs font appel aux destructeurs des classes de degrés inférieurs qui les délimitent. Ainsi, si deux classes d'un niveau supérieur ont une limite identique, le destructeur est appelé deux fois, et le programme génère une erreur de segmentation. Encore une fois, la correction de ce comportement risque de rendre la bibliothèque de génération de fichiers PRC incompatibles avec le logiciel Asymptote initial, ce qui est incompatible avec notre objectif initial de facilitation de la communication.

4.4 Cas tests - Exemples de PDF 3D

Cette section vise à présenter certains modèles géométriques avec ou sans topologie. Les modèles vont être divisés en deux catégories. Les premiers sont des cas tests créés afin de montrer certaines fonctionnalités des modèles BREP, et leur bonne conservation à l'exportation en PDF 3D. La seconde catégorie regroupe des modèles de pièces réelles.

On rappelle que la méthode pour activer les contenus 3D et avoir une bonne visualisation de ceux-ci est la suivante :

- cliquer dans le cadre pour l'activer
- puis, pour recadrer l'image :
  - cliquer droit
  - → "options de pièce"
  - → "contenu"

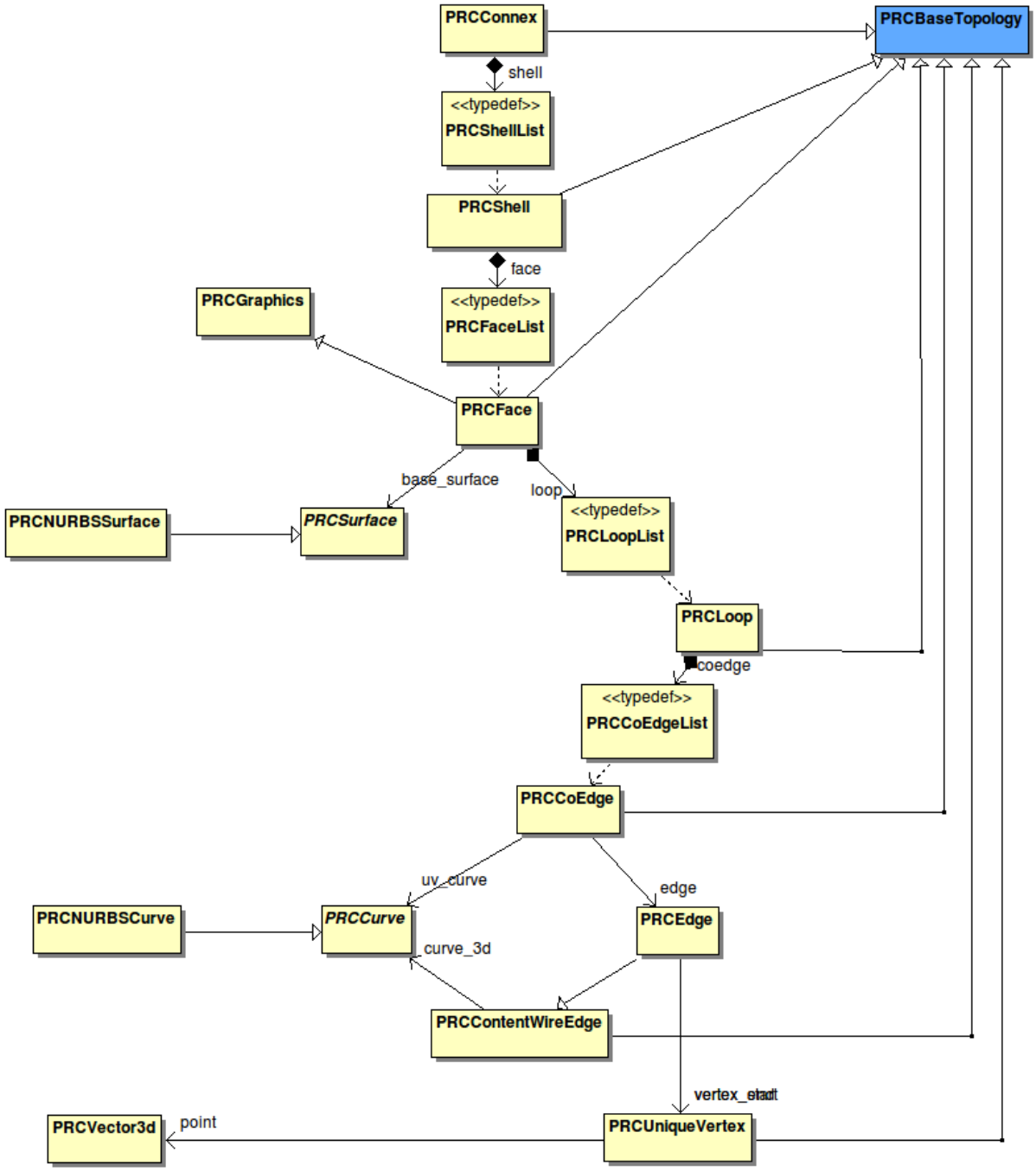

Figure 4.3 Diagramme de classes topologiques de la bibliothèque PRC.

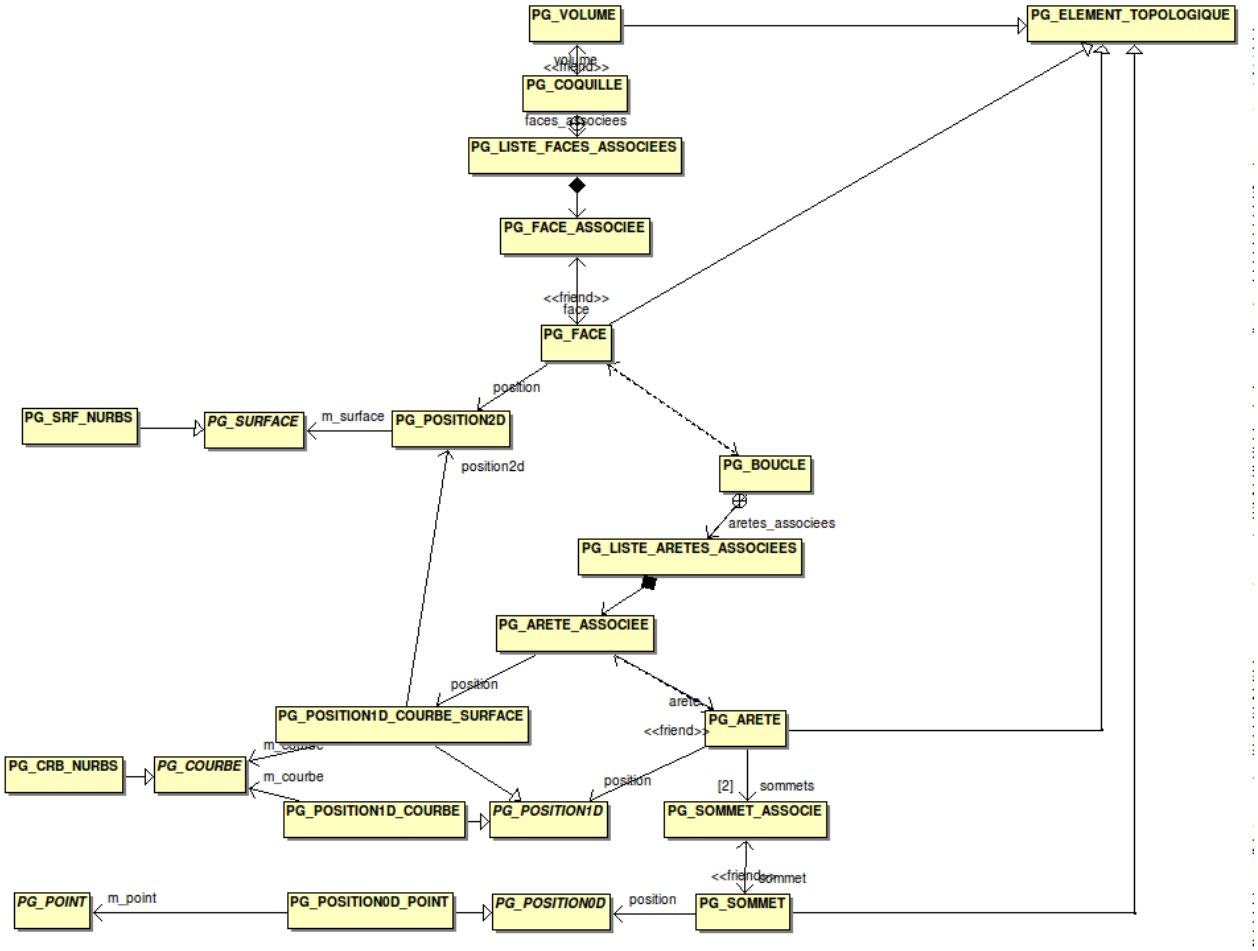

Figure 4.4 Diagramme de classes topologiques de la bibliothèque Pirate.

Figure 4.5 Vue 3D d'un cube ayant une arête plus courte que son côté géométrique, avec sa topologie.

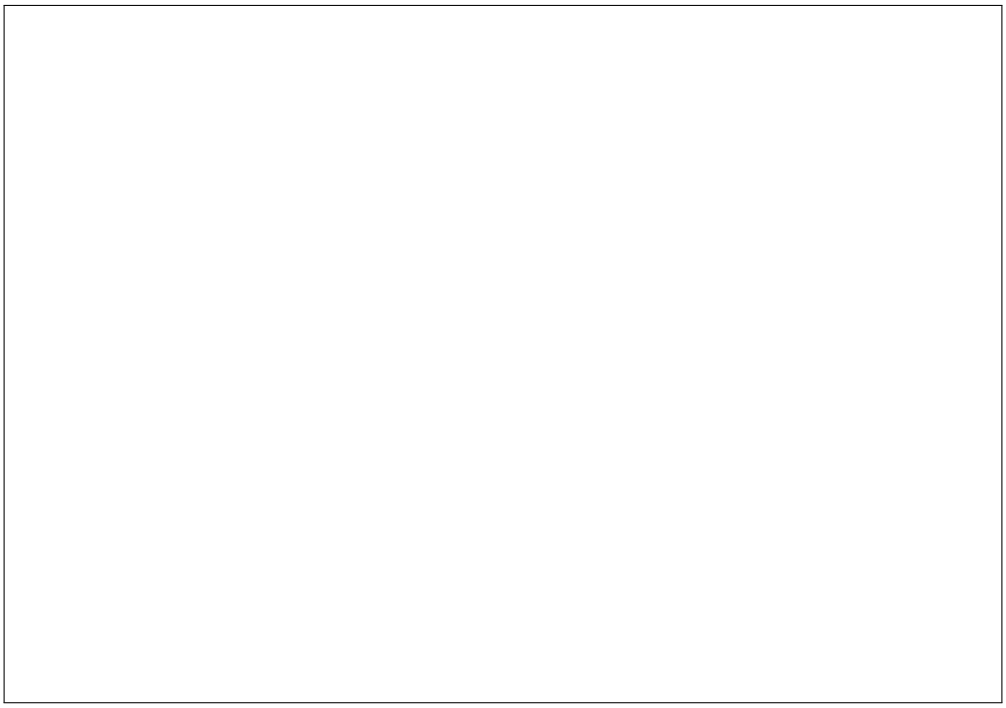

Figure 4.6 Vue 3D d'un cube ayant une arête plus courte que son côté géométrique, sans sa topologie.

```
%% Entête du document
\documentclass [a4paper,12pt] {article}
\usepackage [3D] {movie15}
\usepackage{hyperref}
\begin {document}

%% Définition du cadre et inclusion du fichier PRC
\fbox{
\noindent
\includemovie{140mm}{190mm}{ Modèle_3D.PRC }
}

%% Fin du document
\end {document}
```

Figure 4.7 Code LaTeX intégré à TopoVisu pour générer le fichier PDF (la taille de la fenêtre est en bleu, et le fichier à intégrer en vert)

4.4.1 Modèles simples

Surface rationnelle

On veut vérifier que les courbes et surfaces rationnelles peuvent bien être exportées. Les exemples les plus classiques, testés dans le modèle 4.8, sont les cas du cercle et de la sphère.

Figure 4.8 Exemple d'un cercle et de la sphère générée par sa révolution.

Découpage dans une surface 2D

Cet exemple vise à montrer le bon fonctionnement des découpages dans les surfaces. La figure 4.9 montre une unique surface carrée, que l'on a limitée par deux boucles (une extérieure, l'autre intérieure). Seule la topologie est à l'origine du trou central. Après recadrage, la surface peut se retrouver perpendiculaire au point de vue, il faut donc la faire tourner.

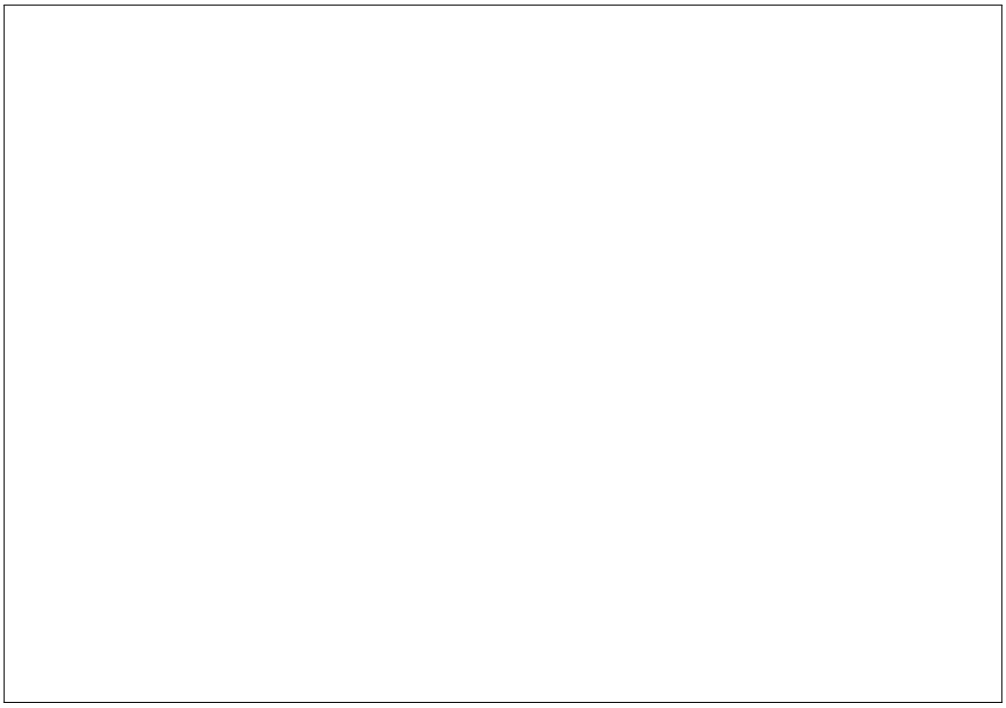

Figure 4.9 Exemple d'un trou dans une surface.

Trou dans un volume 3D

Ici, l'objectif est de voir le bon fonctionnement des trous lorsqu'un volume est limité par deux coquilles. Pour bien visualiser la figure 4.10, il est conseillé de passer la méthode d'affichage en fil de fer.

Pour cela, il faut :

– cliquer droit

– → "options d'affichage"

– → "Mode de rendu du modèle"

– → "Filaire"

On pourra remarquer que les surfaces complexes sont maillées au sein du moteur de rendu d'Adobe Reader. Ceci n'est fait que pour accélérer le rendu graphique, les surfaces sont stockées de manière exacte dans le format PRC.

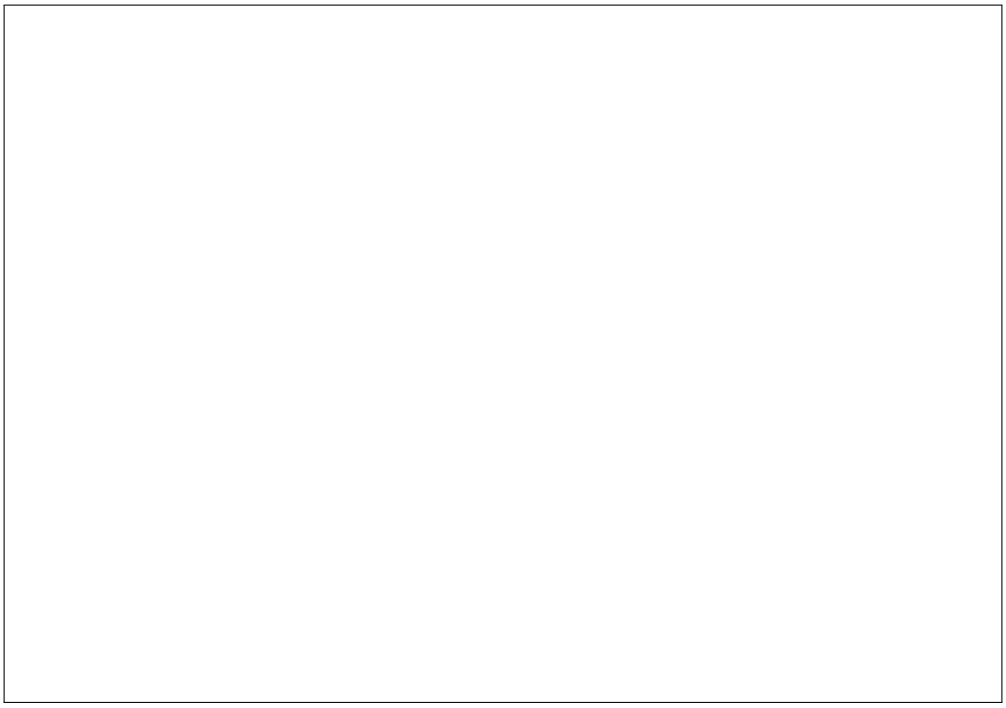

Figure 4.10 Exemple d'un trou dans un volume.

4.4.2 Modèles de pièces réelles

Ces modèles sont des pièces réelles, étudiées au sein du laboratoire MAGNU de l'École Polytechnique. Par conséquent, certains de ces modèles sont lourds en mémoire, et il est déconseillé d'activer le contenu 3D sur des ordinateurs de faible puissance. De plus, le temps de chargement des fenêtres 3D peut être très variable, et aller jusqu'à plusieurs minutes. Il est aussi conseillé de désactiver les contenus 3D inutiles (cliquer droit → "désactiver le contenu") avant d'en activer un nouveau afin de libérer l'espace en mémoire.

Pour faciliter la lecture de ce mémoire, des captures d'écran de l'affichage 3D seront ajoutées à chaque objet 3D.

Géométrie d'un modèle complexe "GAMM3"

Le modèle 4.11 présente un exemple de modèle 3D plus complexe. C'est sa géométrie qui est exportée ici, on obtient donc les différentes surfaces, mais celles-ci ne sont pas limitées par la topologie.

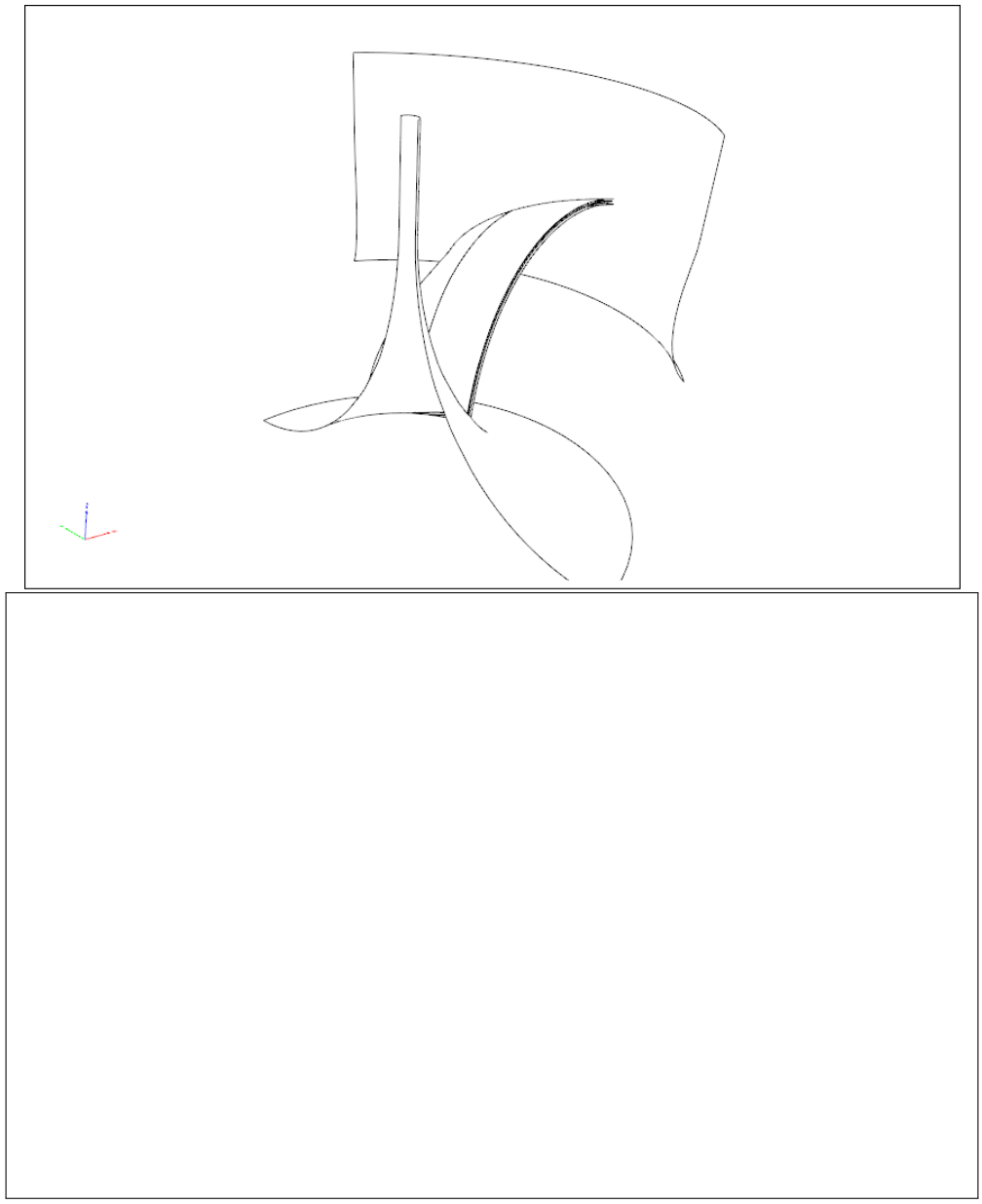

Figure 4.11 Image et contenu 3D de la géométrie d'un modèle complexe "GAMM3".

Géométrie et topologie d'un modèle d'aspirateur "0pier"de type "bulb"

Le modèle 4.12 présente un exemple de modèle BREP complet. On pourra remarquer que le modèle apparait comme un seul nœud au sein du logiciel de visualisation Adobe Reader.

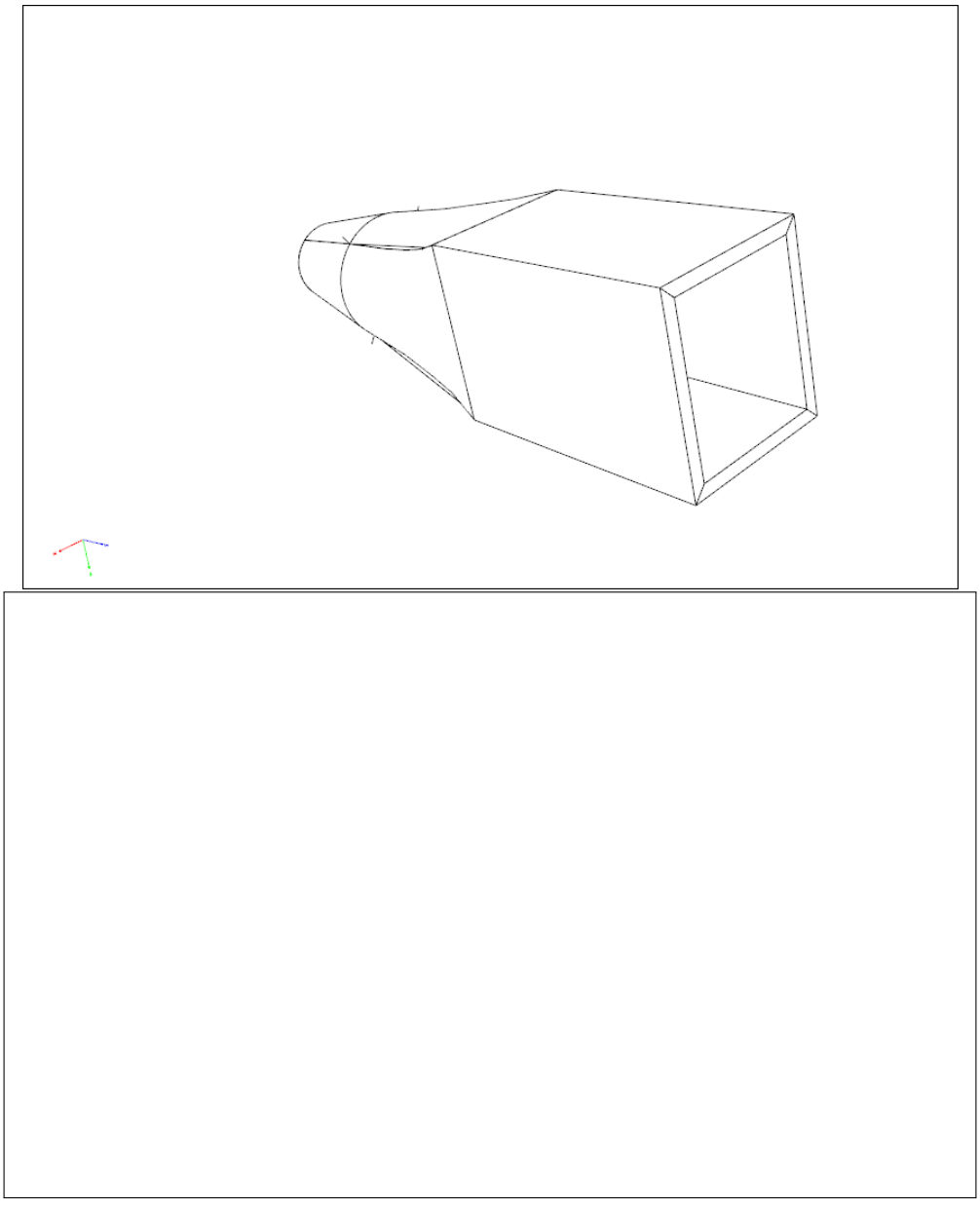

Figure 4.12 Image et contenu 3D de la géométrie d'un modèle d'aspirateur "0pier"de type "bulb".

4.4.3 Exemple de maillage, avec erreurs

L'exemple de la figure 4.13 a pour but de montrer l'utilisation de l'exportation pour les maillages. Les critères d'angle et de volume minimum ont été définis comme extrêmement stricts afin de détecter une quantité de cellules bien visible.

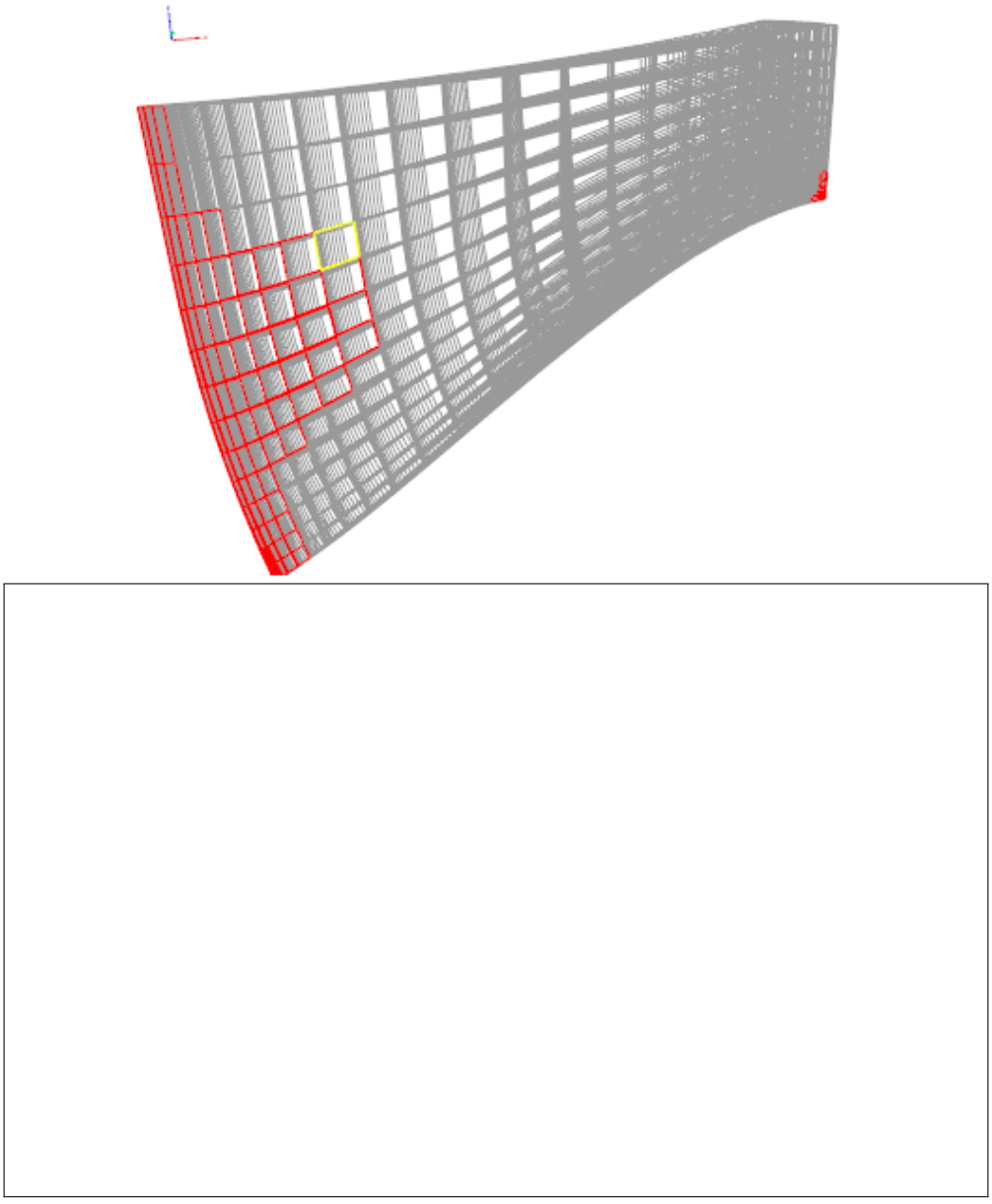

Figure 4.13 Image et contenu 3D d'un maillage, avec des cellules critiques colorées (volume en rouge et angle en jaune).

4.5 Discussion des résultats

On remarque que pour exporter des géométries, le PDF 3D est extrêmement intéressant. On peut exporter ces géométries de manière exacte, et les visualiser très aisément. Si on ajoute la topologie, par contre, on ne peut pas regarder le modèle dans ses détails. En réalité, Adobe Reader ne permet pas de détailler les différents composants d'un modèle géométrique, mais cette limitation a pu être contournée pour les géométries sans topologie en ajoutant chacune des entités dans un sous-modèle PRC séparé. Pour les modèles complets, par contre, il est nécessaire que les entités liées par la topologie soient dans un même modèle, d'où l'impossibilité de détailler les différents éléments qui composent ces modèles. Il est tout de même utile de pouvoir utiliser les modèles avec topologie comme illustrations dans un rapport, et d'avoir la géométrie séparément pour observer les détails.

Un autre point majeur à étudier si l'on souhaite utiliser le PDF 3D comme format d'exportation de données est la taille des différents fichiers générés. Pour cela, nous allons comparer différents objets mécaniques, sous différents formats : Pirate, PRC et PDF3D. Les résultats sont retranscrits dans le tableau 4.1.

Les exemples utilisés pour ces tests de taille de fichier sont les suivants :

Cube : Volume cubique unitaire, avec faces, arêtes et sommets, similaire aux figures 4.5 et 4.6.

Aspirateur "0pier bulb" : Aspirateur ayant une extrémité circulaire et une extrémité carrée, invariant par rotation de 90 degrés sur son axe.

Aspirateur "0pier elbow" : Aspirateur ayant une extrémité circulaire et une extrémité rectangulaire, les deux extrémités sont dans deux plans orthogonaux.

Maillage 1 : Zone de maillage visible sur la figure 4.13.

Roue Francis "GAMM 3" : Section correspondant à une pale d'une roue Francis, avec les volumes de fluide qui l'encadrent. Les surfaces extérieures de la roue et de la pale peuvent être vues sur la figure 4.11.

Lame "GAMM 3" : Extrait du modèle précédent ne contenant que le volume de la pale.

Roue "runner_mo" : Section similaire à la roue Francis, mais sur un autre modèle de turbine.

Plusieurs choses intéressantes sont à signaler. Tout d'abord, pour les modèles géométriques, le passage de PRC à PDF 3D s'accompagne d'un coût fixe (de l'ordre de 10 Ko), dû à l'encapsulage du fichier binaire par la page, qui permet de le visionner dans Adobe Reader. De plus, pour ces modèles, on remarque que le PRC et le PDF 3D sont plus efficaces que le format Pirate. Cela peut s'expliquer par le fait que les informations sont enregistrées de

Tableau 4.1 Tableau présentant l'espace mémoire requis pour des modèles dans différents formats

Modèle	Type de données	Taille PIE	Taille PRC	Taille PDF 3D
Cube	Géométrie	1.67 Ko	1.09 Ko	10.8 Ko
Cube	Modèle complet	3.19 Ko	2.16 Ko	11.9 Ko
Aspirateur "0pier bulb"	Géométrie	1 952 Ko	1 304 Ko	1 314 Ko
Aspirateur "0pier bulb"	Modèle complet	1 978 Ko	4 096 Ko	4 107 Ko
Aspirateur "0pier elbow"	Géométrie	4 656 Ko	3 478 Ko	3 488 Ko
Aspirateur "0pier elbow"	Modèle complet	4 691 Ko	11 001 Ko	11 014 Ko
Aspirateur "2pier"	Géométrie	14 956 Ko	10 970 Ko	10 983 Ko
Aspirateur "2pier"	Modèle complet	15 197 Ko	60 473 Ko	60 498 Ko
Roue Francis "GAMM 3"	Géométrie	20 113 Ko	7 174 Ko	7 186 Ko
Roue Francis "GAMM 3"	Modèle complet	20 152 Ko	27 274 Ko	27 291 Ko
Maillage 1	Maillage	3 306 Ko	585 Ko	575 Ko
Lame "GAMM 3"	Maillage	5 849 Ko	4 468 Ko	4 299 Ko
Roue Francis "GAMM 3"	Maillage	8 654 Ko	24 036 Ko	22 849 Ko
Roue "runner_mo"	Maillage	13 507 Ko	58 547 Ko	55 786 Ko

façon plus efficace dans les fichiers PRC, sa nature binaire permettant une utilisation plus efficace de la mémoire, et utilisant moins de bits inutiles lors de la description des entités.

On remarque ensuite que le constat précédent semble totalement faux pour les modèles avec topologie et les maillages. Pour les modèles avec topologies, on fait face à une limitation de l'implémentation Asymptote du standard PDF 3D. En effet, le système de référencement n'est pas implémenté au sein de cette bibliothèque, il est donc nécessaire de redéfinir les entités à chaque fois qu'elles sont référencées. On a donc une redondance d'information très importante, c'est un problème majeur qui a un coût en mémoire non négligeable et qui atteint à la justesse du modèle.

Pour les maillages, le problème est différent. Actuellement, un maillage est exporté comme une géométrie, chaque arête étant représentée comme un segment de degré 1. Comme signalé précédemment, les éléments de géométrie sont ajoutés dans des sous-modèles. On obtient donc un maillage où les coordonnées des nœuds sont enregistrées au sein de chaque arête, et où on ajoute de l'information (pour garantir la structure de donnée) à chaque élément simple qu'est une arête. On se retrouve donc avec des maillages de taille plus importante en PRC et PDF 3D, le stockage n'étant pas optimisé pour ce genre de structure. Ceci ne se révèle exact que pour les maillages complexes, les maillages plus simples semblent être stockés de manière plus efficace. Cette variation de comportement est difficile à prévoir car elle implique un algorithme de compression intrinsèque à la création du fichier PRC, et lié à sa nature binaire.

CHAPITRE 5

DISCUSSION GÉNÉRALE

Le travail effectué peut se présenter sous deux axes principaux, la détection et la communication.

Comme on a pu le constater en chapitre 3, la détection d'erreurs est suffisante pour le bon déroulement du processus de conception. En effet, les problèmes ont été identifiés dans la littérature et des solutions pour les détecter ont été mises en place. Seuls les critères correspondant à une notion de qualité des entités géométriques ne sont pas traités actuellement. C'est sur ce point que des améliorations pourraient être faites, notamment en définissant des critères précis de qualité, et en implémentant ceux-ci dans des tests automatisés.

La communication (traitée en chapitre 3) autour de ces erreurs, et des modèles et maillages eux-mêmes, a été facilité par la possibilité d'exporter les données utiles sous format PDF 3D. Ce format remplit bien la fonction pour laquelle il a été choisi, soit une visualisation aisée, accessible et exacte des entités géométriques et topologiques formant les modèles BREP et les maillages. Ses limitations actuelles ne sont pas liées au format en lui-même, mais plutôt à l'implémentation partielle présente au sein de la bibliothèque Pirate. Comme signalé en partie 4.5, certains problèmes de duplication de données, ainsi que l'impossibilité d'effectuer la conversion depuis le format PDF 3D vers le format Pirate, limitent encore l'utilisation du PDF comme support d'échange complet.

Malgré les limitations présentes, et sur lesquelles on reviendra en conclusion, les objectifs initiaux ont été atteints. Les erreurs pouvant être présentes au sein des modèles BREP sont détectées, les déformations de maillages sont localisées, et la communication autour de ces problèmes a été facilitée.

CHAPITRE 6

CONCLUSION

L'objectif de ce projet de maîtrise était de faciliter le travail des concepteurs de pièces mécaniques. Le focus principal a été placé sur la détection d'erreurs au sein des modèles BREP et des maillages, ainsi que sur la limitation de ces erreurs en facilitant la communication entre les différents acteurs de la conception. Nous allons donc faire le point sur les travaux effectués, leurs limitations et les améliorations qui peuvent être effectuées dans le futur.

6.1 Synthèse des travaux

La première étape a été de faire le point sur les problèmes existants. Ces problèmes étant liés aux technologies utilisées, une étude des solutions existantes a été effectuée simultanément en établissant un état de l'art. Les points qui sont apparus comme prédominants ont été la création d'erreurs lors des changements de format produits lors de la transmission de données. Ces erreurs doivent être traitées avant de continuer le processus de conception si l'on souhaite obtenir un résultat de qualité. Ces erreurs sont produites principalement à deux niveaux : lorsque les données doivent être passées d'un format à un autre car deux formats différents sont utilisés à deux niveaux différents de la chaîne de conception (on inclut ici deux logiciels de CAO différents utilisés, mais aussi la génération de maillages à partir de modèles BREP), et lorsque des données sont réutilisées après de longues périodes de temps, sur des versions différentes des logiciels. Pour les traiter, deux améliorations principales ont été effectuées. La première a été d'améliorer les différentes interfaces de visualisation d'erreurs afin de faciliter leur identification et leur correction par les concepteurs. Le second ajout au logiciel TopoVisu a été l'exportation des données vers un format non propriétaire très répandu : le PDF 3D. Ceci a été fait en implémentant les classes de topologie (en accord avec la norme STEP) dans une librairie de modèles BREP, en accord avec les spécifications du format PRC.

D'un point de vue plus large, on peut considérer que les travaux effectués vont permettre une meilleure communication des données générées dans le laboratoire (on inclut ici les modèles produits, ainsi que les erreurs rencontrées et le moyen de les corriger) vers le reste de la communauté scientifique, et comme le format choisi tend à devenir un standard de transmission d'informations dans de nombreux milieux professionnels, on ne restreint pas le public potentiel.

6.2 Limitations de la solution proposée

Le travail effectué n'est évidemment pas parfait, certaines limitations pourront être corrigées par un développement futur, alors que d'autres sont dues aux limites des technologies utilisées.

6.2.1 Limitations au niveau de la détection d'erreurs

Une des limitations ici reste la nécessité de l'intervention des concepteurs pour corriger les erreurs détectées. La correction automatique des modèles BREP serait un idéal, mais la diversité des modèles rend très difficile la décision de la correction à appliquer. Il faudrait pour cela intégrer l'expertise des concepteurs au sein du logiciel de correction et l'on se rapproche donc d'un problème d'intelligence artificielle, qui n'est pas le propos ici.

6.2.2 Limitations du PDF 3D

La première limitation est intrinsèque au format PRC. Dans ce format, seuls certains éléments ont une composante d'objet graphique. Ainsi, il est impossible d'afficher les arêtes et les sommets. Ils doivent être affichés au travers des faces dont ils font partie.

Un autre problème est l'impossibilité d'afficher les différentes parties d'un modèle BREP complet (les différents volumes par exemple) séparément dans le logiciel de visualisation Adobe Reader. Cela est dû au fait que l'arbre de représentation présent dans ce logiciel est moins développé (moins de niveaux) que celui présent dans TopoVisu. Il aurait été possible de contourner cette limitation en séparant les volumes à un niveau plus haut des classes du format PRC, mais cela nuirait à la justesse des modèles et n'a donc pas été implémenté.

6.3 Améliorations futures

Bien évidemment, dans un contexte aussi complexe que la conception de pièces mécaniques, de nombreuses améliorations peuvent être apportées à la suite de logiciels basée sur la librairie Pirate. On ne discutera ici que les améliorations qui ont un lien direct avec le travail effectué.

Le point le plus urgent à améliorer est la gestion du référencement au sein du PRC, afin de se débarrasser de la duplication de données lors de l'exportation de modèles BREP complets.

Étant donné que l'exportation vers le PDF est un outil de communication, il pourrait être intéressant de pouvoir ajouter du texte sur la page générée, afin de créer un rapport rapide directement au sein du logiciel. Pour le moment, LaTeX doit être utilisé afin de créer un tel rapport en incorporant le fichier PRC fourni par l'exportation dans TopoVisu.

Il pourrait aussi être utile de continuer le développement de la librairie d'exportation vers le format PRC, afin de pouvoir ajouter des outils de visualisation plus complets au sein du PDF (comme des plans de coupe, des angles de vue prédéfinis, etc.). Cela pourrait grandement faciliter la communication au niveau des erreurs.

Une amélioration majeure qui pourrait être extrêmement intéressante à intégrer au logiciel TopoVisu serait l'importation de modèles PRC intégrés à des PDF 3D. Cela permettrait d'utiliser le PDF 3D comme un format d'échange complet avec d'autres logiciels de CAO. La difficulté, ici, est la création de l'analyseur syntaxique permettant de localiser la composante PRC au sein du PDF puis d'interpréter celle-ci, et ce quelque soit la manière dont est intégré le PRC au sein du PDF, quelques soient les options, et les composantes de ces deux éléments.

Enfin, dans une vision à plus long terme, on pourrait imaginer transformer la bibliothèque d'écriture de PDF 3D de TopoVisu en une librairie libre d'écriture de PRC, indépendante de notre suite de logiciels. On ne fournirait plus uniquement des fichiers aisément transmissibles, mais aussi le moyen de les créer.

RÉFÉRENCES

ADOBE SYSTEMS INCORPORATED (2012). PRC format specifications. `http://livedocs.adobe.com/acrobat_sdk/9/Acrobat9_HTMLHelp/API_References/PRCReference/PRC_Format_Specification`.

BOHM, M., HAAPALA, K., POPPA, K., STONE, R. et TUMER, I. (2010). Integrating life cycle assessment into the conceptual phase of design using a design repository. *Journal of mechanical design*, 132.

BOHM, M., STONE, R., SIMPSON, T. et STEVA, E. (2008). Introduction of a data schema to support a design repository. *Computer-Aided Design*, 40, 801–811.

BUNIN, G. (2006). Non-local topological clean-up. *Proceedings of the 15th International Meshing Roundtable*. Springer, 3–20.

BUSARYEV, O., DEY, T. et LEVINE, J. (2009). Repairing and meshing imperfect shapes with delaunay refinement. *SPM*. vol. 9, 25–33.

BUTLIN, G. et STOPS, C. (1996). Cad data repair. *5th International Meshing Roundtable*. Citeseer, 7–12.

CAMARERO, R., COURCHESNE, O. et GUIBAULT, F. (2008). Hybrid mesh validation and visualization. *Proceedings of Numerical geometry, grid generation and scientific computing*.

CHAND, P. (2001). Detecting translation errors in cad surfaces and preparing geometries for mesh generation. *Proceedings of the 10 th International Meshing Roundtable*, 363–371.

CHEW, L. P., VAVASIS, S., GOPALSAMY, S., PAUL, L., STEPHEN, C., GOPALSAMY, V. S., YU, T. et SONI, B. (2002). A concise representation of geometry suitable for mesh generation.

CHONG, C., SENTHIL KUMAR, A. et LEE, H. (2007). Automatic mesh-healing technique for model repair and finite element model generation. *Finite Elements in Analysis and Design*, 43, 1109–1119.

DOMPIERRE, J., VALLET, M.-G., LABBÉ, P. et GUIBAULT, F. (2005). An analysis of simplex shape measures for anisotropic meshes. *Computer methods in applied mechanics and engineering*, 194, 4895–4914.

EMELYANOV, A., ASTAKHOV, Y. et KLIMENKO, S. (2009). General concept of repairing cad-models. *2009 International Conference on CyberWorlds*. IEEE, 108–113.

FREY, E., OSTROSI, E., ROUCOULES, L. et GOMES, S. (2009). Multi-domain product modeling : From requirements to cad and simulation tools. *Proceedings of the 17th International Conference on Engineering Design (ICED'09), Vol. 5*. 477–488.

FREY, P. et BOROUCHAKI, H. (1999). Surface mesh quality evaluation. *International journal for numerical methods in engineering*, 45, 101–118.

GERBINO, S. (2003). Tools for the interoperability among cad systems. *Associazione Nazionale Disegno di Macchine*.

GERBINO, S., CROCETTA, S. et DI MARTINO, C. (1997). Data exchange in cad systems : limits, solutions, perspectives. *Proceedings of X ADM Conference, Florence, Italy*. 17–19.

GOPALSAMY, S., ROSS, D. et SHIH, A. (2004). Api for grid generation over topological models. *The 13th International Meshing Roundtable, Williamsburg, Virginia, USA*, 221–230.

GRAHN, A. (2012). The movie15 package. Disponible à l'adresse `http://ctan.open-source-solution.org/macros/latex/contrib/movie15/doc/movie15.pdf`.

INFORMATION PROCESSING (IFIP) WORKING GROUP 5.1 (2013). PLM international conference website. `http://www.plm-conference.org/`.

IRAQI, M. et ROUCOULES, L. (2012). From functional analysis to cad modelling based on knowledge transformation driven by the design process.

ISO-10303 (1994). Industrial automation systems and integration – product data representation and exchange. `http://www.iso.org/iso/home.html`.

KHEDDOUCI, F. (2010). *L'archivage à long terme de la maquette numérique 3D annotée*. Mémoire de maîtrise, École de technologie supérieure.

KRAUSE, F., STIEL, C. et LÜDDEMANN, J. (1997). Processing of cad-data conversion, verification and repair. *Proceedings of the fourth ACM symposium on Solid modeling and applications*. ACM, 248–254.

LI, D., SÓBESTER, A. et KEANE, A. (2010). A knowledge-based geometry repair system for robust parametric cad models.

LIU, J. et SUN, S. (2006). Small polyhedron reconnection : A new way to eliminate poorly-shaped tetrahedra. *Proceedings of the 15th International Meshing Roundtable*. Springer, 241–257.

MARCHANDISE, E., PIRET, C. et REMACLE, J. (2011). Cad and mesh repair with radial basis functions. *Journal of Computational Physics*.

MATSUKI, N. (2010). Problems in current cad systems. *International Journal of Product Lifecycle Management*, 4, 326–330.

MEZENTSEV, A. et WOEHLER, T. (1999). Methods and algorithms of automated cad repair for incremental surface meshing. *Proceedings 8th International Meshing Roundtable*. Citeseer, 299–309.

NAGEL, R., BRAITHWAITE, W. et KENNICOTT, P. (1980). Initial graphics exchange specification iges, version 1.0. *National Bureau of Standards*.

PAVIOT, T., FORTINEAU, V., LAMOURI, S., LOUIS-SIDNEY, L. *ET AL.* (2012). A modeling language for 3d process plant layout representation, exchange and visualization.

REGLI, W., KOPENA, J. et GRAUER, M. (2011). On the long-term retention of geometry-centric digital engineering artifacts. *Computer-Aided Design*, 43, 820–837.

SEGAL, M. (1990). Using tolerances to guarantee valid polyhedral modeling results. *ACM SIGGRAPH Computer Graphics*, 24, 105–114.

SHEFFER, A. et ÜNGÖR, A. (2001). Efficient adaptive meshing of parametric models. *Proceedings of the sixth ACM symposium on Solid modeling and applications*. ACM, 59–70.

SMITH, B., BRAUNER, K., KENNICOTT, P., LIEWALD, M. et WELLINGTON, J. (1983). Initial graphics exchange specification(iges) version 2. 0. *NTIS, SPRINGFIELD, VA(USA), 1983, 341*.

SMITH, B., WELLINGTON, J. et NATIONAL BUREAU OF STANDARDS. WASHINGTON, D. (1986). Initial graphics exchange specification (iges) ; version 3.0.

STEINBRENNER, J., WYMAN, N. et CHAWNER, J. (2000). Fast surface meshing on imperfect cad models. *9th International Meshing Roundtable*. Citeseer, 33–42.

STEVES, M. et FRECHETTE, S. (2003). Viewing technologies for cad models.

SUM, S., KOCH, D., NYEN, C., DOMAZET, D. et SAN, L. (1996). Development of a framework system for tool integration in a product information archive. *Computers in industry*, 30, 225–232.

SZYKMAN, S., FENVES, S., KEIROUZ, W. et SHOOTER, S. (2001). A foundation for interoperability in next-generation product development systems. *Computer-Aided Design*, 33, 545–559.

TANAKA, F. et KISHINAMI, T. (2006). Step-based quality diagnosis of shape data of product models for collaborative e-engineering. *Computers in Industry*, 57, 245–260.

WIKIPEDIA (2013). Image csg_tree.png. `http://commons.wikimedia.org/wiki/File:Csg_tree.png/`.

YANG, J. et HAN, S. (2006). Repairing cad model errors based on the design history. *Computer-Aided Design*, 38, 627–640.

YANG, J., HAN, S. et PARK, S. (2005). A method for verification of cad model errors. *Journal of Engineering Design*, 16, 337–352.

ANNEXE A

Article : A BREP Model and Mesh Errors Detecting Tool : TopoVisu

Article présenté à la conférence Product Lifecycle Management 2012, et publié dans IFIP Advances in Information and Communication Technology, Vol. 388

A BREP model and mesh errors detecting tool: TopoVisu

François Petit and François Guibault

École Polytechnique de Montréal, Génie informatique,
2500, chemin de Polytechnique, Montréal (Québec), H3T 1J4 , Canada
{francois.petit,francois.guibault}@polymtl.ca
http://www.polymtl.ca/

Abstract. This paper aims to describe the methodology behind the computer aided design tool TopoVisu. The objective of this tool is to support analysts and designers in their work, mainly by helping detect errors in models and meshes. But the error detection is not the only feature of TopoVisu, it also provides a visualization environment to cope with data portability issues. The paper briefly describes the type of problems that may be detected through the tests implemented in TopoVisu, and the PDF 3D exporting tool implemented as a mean of documentation and communication of the models and their validation.

Keywords: Computer aided design, Boundary representation, meshes, errors detection, errors documentation.

1 Introduction

Computer aided engineering and design involve processes that can be decomposed in several steps. First comes the system analysis, then the preliminary design, during which a boundary representation (BREP) model is created for analysis and optimization purposes. In order to carry out analysis, several models with various levels of fidelity may be used. For high fidelity analysis, the geometric models must be discretized, meshes are therefore generated, and simulations are run on the meshes. According to the simulation results, optimizations can later be carried out on the BREP model to obtain the final numerical model that will be used in the subsequent phases of detailed design.

Along those steps, some errors can be generated, either by designer mistakes or by conversion among the software products used in the design chain. The less demanding way to correct these errors would be to correct them when they are made, or very soon afterwards. Otherwise, an error in design could sometimes be corrected only after several simulations. And the later an error is found, the harder (and costlier) it is to correct it. That is why there is a need for active detection of problems, mainly during the preliminary design phase.

This paper presents a software system that aims to help designers detect CAD and mesh generation errors early. Its main purpose is to propose a visualization environment for BREP models and meshes, to detect and highlight errors and

to allow documenting these errors and save the diagnostics in order to ease communication.

Among various verifications that are performed, the software can, for instance, measure gaps between coinciding topological entities, volumes of mesh cells, size of edges, etc. and compares them to tolerances in order to determine if a given geometric model is valid and watertight or if a mesh is of sufficient quality to allow simulation. When errors are detected, geometric or mesh entities are highlighted (by coloration and listing) in the graphic interactive interface. It's also possible to label each entity if a problem appears. Finally, the user can export his model, with its visualization parameters, to an Adobe PDF 3D file, which can be read easily on any computer with Adobe Reader installed.

The paper first discusses the verification processes that are carried out at three levels: 1) geometric, 2) topological, and 3) meshing, along with example problems for each. The paper also discusses visualization strategies used to help designers understand the problems detected, and how the visual cues may be saved and manipulated in the application-neutral format, PDF 3D.

The resulting software can be considered as a BREP model and mesh debugger, and is able to export its data to an open file format, which can ease analysis and communication during computer aided engineering.

2 Problem review

According to Matsuki [9], problems in CAD systems can be classified in two categories, quality of models and durability of data. The first category regroups problems at different stages of the model. Some are present in the BREP model, others have to be solved on the mesh.

2.1 Issues concerning the quality of models

BREP model issues. The BREP model issues can be summarized in one word: water-tightness. As the main principle for BREP models is to define an entity through its boundaries, it is necessary to have closed surfaces to delimit the volumes. Most of the problems consist in two entities that do not coincide while they should. For example, two connected surfaces can be slightly different at their common edge (see Fig. 1). In this case, the shell defining the volume of the turbine will be closed through the creation of edges on each surface, and declaring them as corresponding.

With this example problem, as with every manifold BREP model, it is possible to create a surface joining the existing borders, but such an approach increases the computing time and model complexity drastically. What's more, models used in TopoVisu are based on the STEP [7] concept of BREP models, so they can contain non-manifold entities (eg. internal faces in volumes), and thus attempting to connect several surfaces along a given edge worsens the problem. For these reasons, other solutions are used, which may also be applied

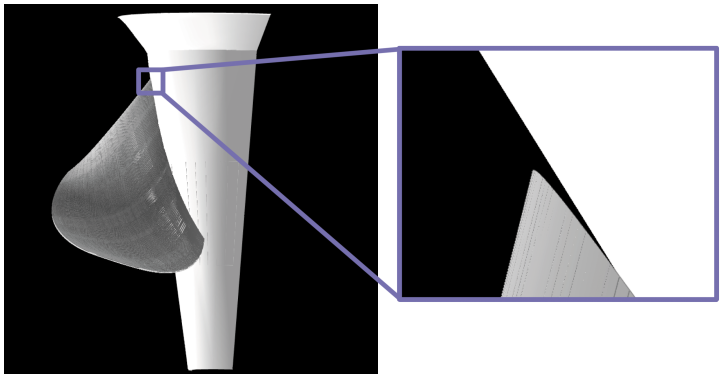

Fig. 1. Example of water-tightness problem: there is a small gap between the turbine blade and the hub.

in the more restrictive domain of manifold models. The existing solutions that inspired the development of TopoVisu follow.

It can be relatively simple to add information to the geometrical information. For example, most BREP models contain topological information. Entities (Vertices, edges, faces, ...) are used to mark up the corresponding geometrical objects. These entities are the core of the topology defined in the STEP [7] standard. This allows to verify the theoretical water-tightness, but if the surfaces do not match where they should, the quality of the mesh generated afterwards will be impacted.

To ease the adjustment of borders, it's possible to use a simplified model, such as the one proposed by [5]. In this model, the entities of lower dimensions (curves, points for 3D) are only defined as sub-objects (and one scalar equation) of higher dimension objects. This allows the suppression of edges and vertices by themselves but, as a drawback, it's not possible to access an explicit expression for these entities. So there can again be a loss in computing speed.

Segal [11] proposed to use tolerances to detect what entities should be merged. This solution, although useful in most cases, is not flawless. Geometric models of industrial complexity often include regions or parts which are too narrow to be united/separated correctly using only tolerances (see Fig. 2 for an example).

The approach proposed in complete topological models, such as those proposed in STEP [7], by Tanaka and Kishinami [14], and TopoVisu, is to add the topological information to the geometry, and verify model coherency through tests based on tolerances. That way, one can know beforehand what entities are connected, and the tolerances are only used to ensure that the geometric reality is consistent with the topological information.

This solution allows to have a BREP model of quality before mesh generation. But the mesh can still have problems due, for example, to a narrow geometry. To ensure that particularities of a geometry does not result in a simulation of poor quality, a mesh checking step is generally added before simulations are run.

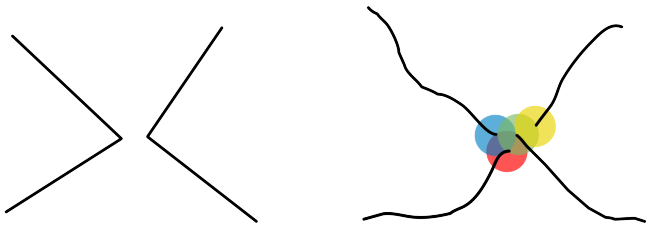

Fig. 2. Ideal case (on the left) and ambiguous case described with tolerances of a unique topological situation.

Meshing issues. Concerning the mesh generation, different strategies can be used to avoid errors. One approach is to have a BREP model as precise and with as much information as possible. Some examples of this strategy follow.

Gopalsamy, Ross and Shih [5] chose to add information to the BREP model in the mesh generation as a XML file. This file contains information about critical areas of the model, as the meshing method to use on each area.

Steinbrenner, Wyman and Chawner [12] presents the interactive mesh generation process they developed for the JULIUS project. The main idea is to make the program ask for help or directions according to its needs for each product. The main problem with this method is that it is not compatible with batch-processing.

Another approach to mesh generation is to consider that the BREP model contain errors, and that the meshing algorithm will have to ignore them first. Chong, Senthil Kumar and Lee [4] decided to begin by meshing the incorrect BREP model, and to solve its problems on the resulting mesh. The difficulty with this approach is that they never have a correct BREP model, while correctness may later on be required, for instance for manufacturing purposes.

Busaryev, Dey and Levine [2] present an algorithm based on the merging of areas with small defects (gaps, overlaps, and intersections) through protecting balls, directly on the BREP model, and then to mesh the model using these protecting balls. But with this method, there is a risk to have similar problems to the approach using only tolerances to close a BREP model.

What differentiate the approach presented in this paper is that the BREP models are created and used in a unique, integrated format. This approach ensures that no error can be generated once the BREP model has been checked. Moreover, meshes need to be generated in batch mode, so no interactive method can be used. Thus meshes need to be checked afterwards, prior to using them in simulations (which is generally also a batch process). As verification of the mesh is not performed during its generation, the need for a checking tool arises : TopoVisu, a BREP model and mesh debugger, is needed.

2.2 Issues concerning portability and durability of data

Another issue is that even if a product is in a correct state, either a good quality mesh or BREP model, it can be deteriorated through communication. As the different steps of the design process do not use the same tools, software and file formats, errors can be generated, and information can be lost during conversions. That is why Sum, Koch, Nyen, Domazet and San [13] defined an archive framework adapted to the design tools used by their team. Their objective was to unite every version of a product while still being able to produce program-specific "views" of the product. This archive would handle the specificities of each file format.

Short-term errors are not the only ones designers have to face. Another issue is the conservation of product data. Regli, Kopena and Grauer [10] try to determine what data is necessary and sufficient to ensure that product data will be valid, and well interpreted in the future. The answer is far from trivial as it is not safe to assume that any software or file format will still be in use in 20 years (and 20 years is actually quite short-term when compared to the life span of some planes or buildings). Their conclusion is that it is really difficult to estimate what is sufficient, so it is easier to just keep everything possible. They also advise to keep records of product data in different forms, such as a native file format, a standard file format and a visualization file format.

Our approach is similar as our models can be saved as ".pie" (our native file format) and they can be exported as STEP files, and as PDF 3D files, which correspond to the standard and visualization aspects. To keep a good portability of the tool, the meshes are described in a CGNS-compatible file format.

3 Tests run in TopoVisu

The main objective for handling errors during the design phase is to detect the errors as soon as possible. TopoVisu detects those errors by running tests at various steps of the design phase. Some tests are run on the geometry and the topology, while others are run later, when the mesh is generated. Although both series of tests check different criteria, they are similar in their process. The idea is to evaluate each entity of the model or the mesh, and then to display the results (either as a list of faulty entities, or as statistics) to the user.

3.1 Geometry and topology testing

The quality of the initial model is really important for the validity of the mesh that will be generated from it. As long as the mathematical expression of the surfaces and curves are valid, the main problems of a BREP model come from the topology. That's why the tests on the model have to consider both the geometry and the topology.

The verification process is quite straightforward. The topological entities can be represented as a tree, an abstract root node is parent to all volumes, each

volume is parent to the shells that bound it, each shell is parent to faces and so on. Each entity has a list of the other entities it interacts with (for example, each co-edge has a list of other co-edges built on the same edge). Thus, to verify the model, the entity hierarchy is recursively traversed and tests are run on each entity to verify model properties based on neighborhood dependencies.

Table 1 presents the geometry and topology tests that are implemented in TopoVisu, they are classified according to the dimension of the geometric entities concerned. This table summarizes the description of the tests detailed in [3].

Table 1. Table presenting the tests run on geometry and topology in TopoVisu.

Dimension - Name	Test	Other entities involved	Description of the tests
0 - Vertex	Position	Co-vetices	Each vertex is compared to all the other vertices that should coincide with it (co-vertices).
	Position	Edges ending on the vertex	The extremity of each edge is compared to the position of the vertex
	Position	Edges ending on the vertex, co-vertices	The extremity of each edge is compared to the positions of all the co-vertices
1 - Edge	Length	Co-edges	The length of the corresponding edges are compared
	Coincidence	Co-edges	Coincidence among co-edges is checked: points along the edges are evaluated, and corresponding positions on each co-edge are compared
2 - Face	Coincidence	Edges	Verify that each edge delimiting the surface lies on that surface
	Loop closure	Edges	Verify that the edges delimiting the surface form a closed loop
	Loop closure	Co-edges	Verify that the co-edges form a closed loop

Fig. 3 shows two examples of detected errors. The problematic elements are colored in red and listed in the "Topological errors" box. The errors are assigned to the topological element of the lowest dimension containing all the entities implicated in the error.

3.2 Mesh testing

At the end of the mesh generation process, the validity of the mesh must be assessed. In a valid mesh, each edge connects two nodes, each node is connected by edges, there is no folding, no node outside the geometry, no hanging nodes, etc. What designers have to check is the quality of this mesh. Cells that are too

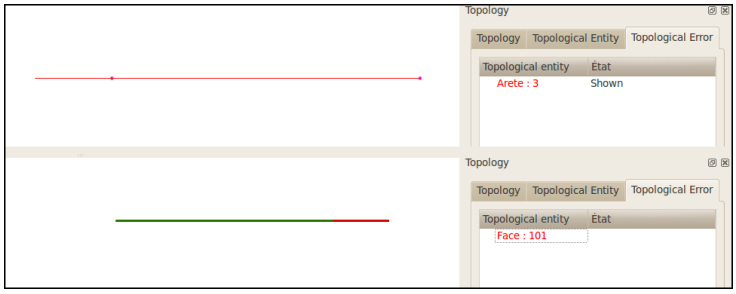

Fig. 3. Examples of topological errors: a vertex not corresponding to the end of an edge (top), and corresponding edge and co-edge with different lengths (bottom).

long, or angles that are too sharp may directly affect convergence during the subsequent simulation phases. To maximize the quality, each cell is evaluated through some of its geometric characteristics (which are calculated beforehand for each cell). Table 2 presents the mesh tests that are implemented in TopoVisu, for 2D and 3D cells. Each threshold mentioned is actually a pair of values which includes a warning and an error threshold.

Table 2. Table presenting the tests run on mesh in TopoVisu

Dimension	Entity concerned	Description of the test
2D	The length of each edge	Lengths are compared to low and high thresholds, and between each other, to avoid sudden change of scale.
	The area of each cell	Areas are compared to low and high thresholds, and between each other, to avoid sudden change of scale.
	The aspect ratio of each cell	Aspect ratio is compared to low and high thresholds, to limit error in the following simulation.
3D	The length of each edge	Lengths are compared to low and high thresholds, and between each other, to avoid sudden change of scale.
	The volume of each cell	Areas are compared to low and high thresholds, and between each other, to avoid sudden change of scale.
	The aspect ratio of each cell	Aspect ratio is compared to low and high thresholds, to limit error in the following simulation.

The cells that do not pass the tests are tagged as containing errors, and are colored in red. In addition, some thresholds (such as volume, angle, etc.) can be adjusted by the user to match stricter requirements from clients. In that case, more cells are highlighted, as shown in Fig. 4. At the end of the series of tests, a report is generated. This report notifies the user of the number of cells in the warning and error ranges, as well as the positions of the cells having extreme values for each criterion.

4 GUI Design

The main purpose of the proposed debugger is to support the designers and analysts in their work. A tool is really of help when it is instinctive in its use. That's why one of the first steps of development was to create a graphics environment to support the visualization of the models.

The core of the application is based on Qt, a multi-platform object-oriented framework. This framework proposes a widget-based development, which allows adding new features as the functionality of the tool needs to evolve. As Qt is a widespread framework, it provides a familiar look to the user and minimizes portability issues. Qt's familiar look was maintained in the creation of menus and sub-windows as can be seen in Fig. 4.

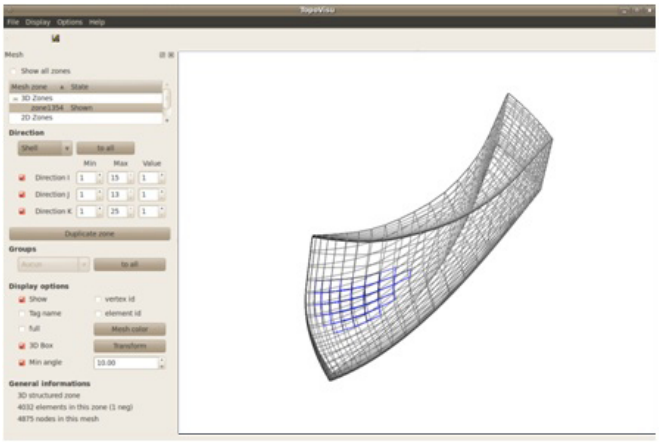

Fig. 4. Screenshot of the TopoVisu interface, with angle threshold activated. Cells with too sharp angles are colored in blue.

For the visualization of the three dimensional model, the widely used cross-platform OpenGL API was adopted. The philosophy of TopoVisu is to separate visualization from actual mathematical data. To achieve this, independent visual objects are constructed, which are then linked to the mathematical objects used in the BREP representation. Two symmetric structures are obtained, one containing the visualization information, and the other containing the BREP model. Each item of each list can be linked to its corresponding item in the other list.

Complex models can be hard to fathom as a whole. TopoVisu provides some ways to help its users have a better visual on BREP models and meshes. First, each graphical entity has its own color. Although initial colors are based on the type of entity they are attributed to (faces are blue, edges are green, etc.), it is possible to change the color of each element separately. This is particularly

useful in order to focus on a particular area. It's also possible to hide parts of the model if it is of no use for the user's task. Calculating the bounding boxes of the visible elements allows a precise focus of the camera.

Concerning meshes, several display modes are available. Users can choose to only display the frame of the meshed volume, or to add the cells on the external shell, or to display the whole mesh. As mentioned in 3.2, cells can be colored according to certain characteristics.

5 Export to PDF3D

As explained in [9] and [10], durability of CAD data is an unsolved issue. Nowadays, most legally accepted archives are on paper, and according to [8], some companies are attempting to move away from this strictly paper-based approach. This orientation isn't surprising given that paper plans are not used anymore during the design process.

The main problem in that area is to ensure the longevity of the data. One solution is to use a non application-specific format that is already used on a large scale. With such a format, we can be sure that support will exist for a long time, and that's why we decided to export our own data in the same format: the Adobe PDF file format.

In this file format, it is possible to insert 3D data in two types of formats: U3D and PRC. While the first format aims for three dimensional graphical scenes (so it contains geometry in an approximated form), the second can store geometry and topology in their exact mathematical form. That's why we chose the PRC embedded in PDF as a favorable transmission format, the embedding allowing any user with the freeware Adobe Reader to open the file and see the geometric model. It's also possible to manipulate the model in three dimensions in order to see every angle.

To create our PDF3D - PRC exporter, we decided to use an existing C/C++ component. This component was extracted from the open source software Asymptote [6]. This library only allows the creation of geometry (the visualization aspect of TopoVisu), the topology management still had to be implemented. It was developed according to the PRC file format specifications [1].

The exporting tool offers the possibility to save all or part of the geometry of the current open project. It saves all items with their current color to allow the highlight of a specific part. This can be particularly useful for a good visualization in a report for example.

6 Conclusion

The development of TopoVisu is still in progress and is probably to stay that way, in order to keep up with all the changes that come in the design industry. But the debugger is usable, and used, right now. The error detection tools are really helpful to designers. It would be nearly impossible to check models entirely by hand.

One advantage of our method is that the product has a good quality at both BREP model and mesh steps. This allows to reuse the model if meshing technology evolve drastically in the future, as opposed to the methods that consider the model to be faulty. Another strong point of TopoVisu is its strong visualization environment. The software is interactive, easy to use, and easy to learn. The results of the tests can be seen, and that is very helpful.

The software does not only help to design new products, it also allows to document and communicate easily on the designs. All the file formats we use are easily convertible to standards (STEP for BREP models, CGNS for meshes and PDF 3D - PRC for reports), as they follow the same methodology. Our objective, in the long run, is to be able to create easily shareable archives of BREP models and meshes, with highlighted errors and points of interest.

References

1. Adobe Systems Incorporated: PRC format specifications (Jan 2012), `http://livedocs.adobe.com/acrobat_sdk/9/Acrobat9_HTMLHelp/API_References/PRCReference/PRC_Format_Specification`
2. Busaryev, O., Dey, T., Levine, J.: Repairing and meshing imperfect shapes with delaunay refinement. In: SPM. vol. 9, pp. 25–33 (2009)
3. Camarero, R., Courchesne, O., Guibault, F.: Hybrid mesh validation and visualization. Proceedings of Numerical geometry, grid generation and scientific computing
4. Chong, C., Senthil Kumar, A., Lee, H.: Automatic mesh-healing technique for model repair and finite element model generation. Finite Elements in Analysis and Design 43(15), 1109–1119 (2007)
5. Gopalsamy, S., Ross, D., Shih, A.: API for grid generation over topological models. The 13th International Meshing Roundtable, Williamsburg, Virginia, USA pp. 221–230 (2004)
6. Hammerlindl, A., Bowman, J., Prince, T.: Asymptote sources website (Jan 2012), `http://asymptote.sourceforge.net`
7. ISO-10303: Industrial automation systems and integration – product data representation and exchange (1994), `http://www.iso.org/iso/home.html`
8. Kheddouci, F.: L'archivage à long terme de la maquette numérique 3D annotée. Master's thesis, École de technologie supérieure (2010)
9. Matsuki, N.: Problems in current CAD systems. International Journal of Product Lifecycle Management 4(4), 326–330 (2010)
10. Regli, W., Kopena, J., Grauer, M.: On the long-term retention of geometry-centric digital engineering artifacts. Computer-Aided Design 43(7), 820–837 (2011)
11. Segal, M.: Using tolerances to guarantee valid polyhedral modeling results. ACM SIGGRAPH Computer Graphics 24(4), 105–114 (1990)
12. Steinbrenner, J., Wyman, N., Chawner, J.: Fast surface meshing on imperfect CAD models. In: 9th International Meshing Roundtable. pp. 33–42. Citeseer (2000)
13. Sum, S., Koch, D., Nyen, C., Domazet, D., San, L.: Development of a framework system for tool integration in a product information archive. Computers in industry 30(3), 225–232 (1996)
14. Tanaka, F., Kishinami, T.: Step-based quality diagnosis of shape data of product models for collaborative e-engineering. Computers in Industry 57(3), 245–260 (2006)

ANNEXE B

Résultat d'exécution de Vérificateur sur le cas test 1 (fichier PIE)

```
/** @file     testcase1.pie

    @date     Wed May 18 18:49:37 2011

    Commande :  verificateur.exe -o output1.pie testcase1.pie   */
Champ<double> Coo() = {  0 0 0 1 0 0};

Geometrie cube() = {

   Courbe crb1( 2, 2, [<double>
 0 0 0
 1 0 0
 ]%3 );

   Sommet 1( crb1( 0.2 ) ) ;
   Sommet 2( crb1( 1 ) ) ;

   Arete 3( crb1 ) = { 1 2 } ("TOLERANCE_SOMMET_SUR_ARETE" "Erreur 1 0.04
   0.0001 0.01") ;
} ;
```

ANNEXE C

Code source du cas test 2 (fichier PIE)

```
/** @file     cube_top.pie
    @brief    Fichier ecrit par le programme /home/p302223/Pirate/
          RepTest_opgeomPirate/CubeTopologique/dbgLinux_3.2.1/main.exe.
    @author   Mohammed Khachan login: p302223
    @command  /home/p302223/Pirate/RepTest_opgeomPirate/
          CubeTopologique/dbgLinux_3.2.1/main.exe - -o cube_top.pie
    @date     Fri Nov  7 12:20:56 2003
*/


Geometrie Cube() = {

   Courbe rationnelle courb_1( 2, [<int>  1 2 ], [<double>
 0.2 0 0 1
 1 0 0 1
 0 0 1 1
 1 0 1 1
 0 1 0 1
 1 1 0 1
 0 1 1 1
 1 1 1 1
 ]%4 );
   Courbe rationnelle courb_2( 2, [<int>  2 4 ], [<double>
 0 0 0 1
 1 0 0 1
 0 0 1 1
 1 0 1 1
 0 1 0 1
 1 1 0 1
```

```
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_3( 2, [<int>  4 3 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_4( 2, [<int>  3 1 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_5( 2, [<int>  5 6 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_6( 2, [<int>  6 8 ], [<double>
0 0 0 1
1 0 0 1
```

```
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_7( 2, [<int>  8 7 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_8( 2, [<int>  7 5 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_9( 2, [<int>  1 5 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
```

```
]%4 );
  Courbe rationnelle courb_10( 2, [<int>  2 6 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_11( 2, [<int>  4 8 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_12( 2, [<int>  3 7 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe umin( 2, 2, [<double>
0 0
1 0
]%2 );
  Courbe umax( 2, 2, [<double>
```

```
0 1
1 1
]%2 );
  Courbe vmin( 2, 2, [<double>
0 0
0 1
]%2 );
  Courbe vmax( 2, 2, [<double>
1 0
1 1
]%2 );

  Surface rationnelle surface_1( 2, 2, 2, [<int>  1 2 3 4 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Surface rationnelle surface_2( 2, 2, 2, [<int>  1 2 5 6 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Surface rationnelle surface_3( 2, 2, 2, [<int>  2 4 6 8 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
```

```
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Surface rationnelle surface_4( 2, 2, 2, [<int>  3 4 7 8 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Surface rationnelle surface_5( 2, 2, 2, [<int>  1 3 5 7 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Surface rationnelle surface_6( 2, 2, 2, [<int>  5 6 7 8 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
```

```
   Sommet 1( courb_1( 0 ) ) ;
   Sommet 2( courb_2( 0 ) ) ;
   Sommet 3( courb_4( 0 ) ) ;
   Sommet 4( courb_3( 0 ) ) ;
   Sommet 5( courb_5( 0 ) ) ;
   Sommet 6( courb_6( 0 ) ) ;
   Sommet 7( courb_8( 0 ) ) ;
   Sommet 8( courb_7( 0 ) ) ;

   Arete 9( courb_1 ) = { 1 2 };
   Arete 10( courb_2 ) = { 2 4 };
   Arete 11( courb_3 ) = { 4 3 } ;
   Arete 12( courb_4 ) = { 3 1 } ;
   Arete 13( courb_5 ) = { 5 6 } ;
   Arete 14( courb_6 ) = { 6 8 } ;
   Arete 15( courb_7 ) = { 8 7 } ;
   Arete 16( courb_8 ) = { 7 5 } ;
   Arete 17( courb_9 ) = { 1 5 } ;
   Arete 18( courb_10 ) = { 2 6 } ;
   Arete 19( courb_11 ) = { 4 8 } ;
   Arete 20( courb_12 ) = { 3 7 } ;



   Face 21( surface_1 )
= { 9( surface_1( umin ) ) 10( surface_1( vmax ) )
11(  - surface_1( umax ) ) 12(  - surface_1( vmin ) )   } ;
   Face 22( surface_2 )
= { 9( surface_2( umin ) ) 18( surface_2( vmax ) )
- 13(  - surface_2( umax ) ) - 17(  - surface_2( vmin ) )   } ;
   Face 23( surface_3 )
= { 10( surface_3( umin ) ) 19( surface_3( vmax ) )
- 14(  - surface_3( umax ) ) - 18(  - surface_3( vmin ) )   } ;
   Face 24( surface_4 )
= { - 11( surface_4( umin ) ) 19( surface_4( vmax ) )
```

```
15(  - surface_4( umax ) ) - 20(  - surface_4( vmin ) )   } ;
   Face 25( surface_5 )
= { - 12( surface_5( umin ) ) 20( surface_5( vmax ) )
16(  - surface_5( umax ) ) - 17(  - surface_5( vmin ) )   } ;
   Face 26( surface_6 )
= { 13( surface_6( umin ) ) 14( surface_6( vmax ) )
15(  - surface_6( umax ) ) 16(  - surface_6( vmin ) )   } ;

   Volume 27()
= { 21 - 22 - 23 24 25 - 26   } ;
};
Proprietes
{
  Geometrie Cube("VERIFIER_DISTANCES_ARETE_VS_ARETE_ASSOCIEE" "");

};
```

ANNEXE D

Code source du cas test 3 (fichier PIE)

```
Geometrie Cube() = {

  Courbe rationnelle courb_1b( 2, [<int>  1 2 ], [<double>
 0 0 0 1
 1 0 0 1
 0 0 1 1
 1 0 1 1
 0 1 0 1
 1 1 0 1
 0 1 1 1
 1 1 1 1
 ]%4 );


   Courbe rationnelle courb_1( 3, 3, [<double>
 0 0 0 1
 0.5 0.5 0 1
 1 0 0 1
 ]%4 );


  Courbe rationnelle courb_2( 2, [<int>  2 4 ], [<double>
 0 0 0 1
 1 0 0 1
 0 0 1 1
 1 0 1 1
 0 1 0 1
 1 1 0 1
 0 1 1 1
 1 1 1 1
```

```
]%4 );
  Courbe rationnelle courb_3( 2, [<int>  4 3 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_4( 2, [<int>  3 1 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_5( 2, [<int>  5 6 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_6( 2, [<int>  6 8 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
```

```
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_7( 2, [<int>  8 7 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_8( 2, [<int>  7 5 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_9( 2, [<int>  1 5 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_10( 2, [<int>  2 6 ], [<double>
```

```
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_11( 2, [<int>  4 8 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_12( 2, [<int>  3 7 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Courbe umin( 2, 2, [<double>
0 0
1 0
]%2 );
  Courbe umax( 2, 2, [<double>
0 1
1 1
```

```
]%2 );
  Courbe vmin( 2, 2, [<double>
0 0
0 1
]%2 );
  Courbe vmax( 2, 2, [<double>
1 0
1 1
]%2 );


   Surface rationnelle surface_1( 3, 3, 3, 3, [<double>
0 0 0 1
0.5 -0.5 0 1
1 0 0 1

0 0 0.5 1
0.5 0 0.5 1
1 0 0.5 1

0 0 1 1
0.5 0 1 1
1 0 1 1
]%4 );

    Surface rationnelle surface_2( 3, 3, 3, 3, [<double>
0 0 0 1
0.5 0.5 0 1
1 0 0 1

0 0.5 0 1
0.5 0.5 0 1
1 0.5 0 1

0 1 0 1
0.5 1 0 1
```

```
1 1 0 1
]%4 );

  Surface rationnelle surface_3( 2, 2, 2, [<int>  2 4 6 8 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Surface rationnelle surface_4( 2, 2, 2, [<int>  3 4 7 8 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Surface rationnelle surface_5( 2, 2, 2, [<int>  1 3 5 7 ], [<double>
0 0 0 1
1 0 0 1
0 0 1 1
1 0 1 1
0 1 0 1
1 1 0 1
0 1 1 1
1 1 1 1
]%4 );
  Surface rationnelle surface_6( 2, 2, 2, [<int>  5 6 7 8 ], [<double>
0 0 0 1
1 0 0 1
```

```
 0 0 1 1
 1 0 1 1
 0 1 0 1
 1 1 0 1
 0 1 1 1
 1 1 1 1
 ]%4 );

   Sommet 1( courb_1( 0 ) ) ;
   Sommet 2( courb_2( 0 ) ) ;
   Sommet 3( courb_4( 0 ) ) ;
   Sommet 4( courb_3( 0 ) ) ;
   Sommet 5( courb_5( 0 ) ) ;
   Sommet 6( courb_6( 0 ) ) ;
   Sommet 7( courb_8( 0 ) ) ;
   Sommet 8( courb_7( 0 ) ) ;

   Arete 9( courb_1 ) = { 1 2 };
   Arete 10( courb_2 ) = { 2 4 };
   Arete 11( courb_3 ) = { 4 3 } ;
   Arete 12( courb_4 ) = { 3 1 } ;
   Arete 13( courb_5 ) = { 5 6 } ;
   Arete 14( courb_6 ) = { 6 8 } ;
   Arete 15( courb_7 ) = { 8 7 } ;
   Arete 16( courb_8 ) = { 7 5 } ;
   Arete 17( courb_9 ) = { 1 5 } ;
   Arete 18( courb_10 ) = { 2 6 } ;
   Arete 19( courb_11 ) = { 4 8 } ;
   Arete 20( courb_12 ) = { 3 7 } ;



   Face 21( surface_1 )
= { 9( surface_1( umin ) ) 10( surface_1( vmax ) )
11(  - surface_1( umax ) ) 12(  - surface_1( vmin ) )   } ;
   Face 22( surface_2 )
```

```
= { 9( surface_2( umin ) ) 18( surface_2( vmax ) )
- 13(  - surface_2( umax ) ) - 17(  - surface_2( vmin ) )   } ;
   Face 23( surface_3 )
= { 10( surface_3( umin ) ) 19( surface_3( vmax ) )
- 14(  - surface_3( umax ) ) - 18(  - surface_3( vmin ) )   } ;
   Face 24( surface_4 )
= { - 11( surface_4( umin ) ) 19( surface_4( vmax ) )
15(  - surface_4( umax ) ) - 20(  - surface_4( vmin ) )   } ;
   Face 25( surface_5 )
= { - 12( surface_5( umin ) ) 20( surface_5( vmax ) )
16(  - surface_5( umax ) ) - 17(  - surface_5( vmin ) )   } ;
   Face 26( surface_6 )
= { 13( surface_6( umin ) ) 14( surface_6( vmax ) )
15(  - surface_6( umax ) ) 16(  - surface_6( vmin ) )   } ;

   Volume 27()
= { 21 - 22 - 23 24 25 - 26   } ;
};
```

ANNEXE E

Code source du cas test 4 (fichier PIE)

```
Geometrie gest() = {

  Courbe crb4( 2, 2, [<double>
 0 1 0
 0 0 0
 ]%3 );
  Courbe crb2( 2, 2, [<double>
 1 0 0
 1 0.8 0
 ]%3 );
  Courbe crb1( 2, 2, [<double>
 0 0 0
 1 0 0
 ]%3 );
  Courbe crb3( 2, 2, [<double>
 1 1 0
 0 1 0
 ]%3 );


  Courbe crb1p( 2, 2, [<double>
 0 0
 1 0
 ]%2 );
  Courbe crb2p( 2, 2, [<double>
 1 0
 1 1
 ]%2 );
  Courbe crb3p( 2, 2, [<double>
 0 1
```

```
 1 1
 ]%2 );
   Courbe crb4p( 2, 2, [<double>
 0 0.8
 0 0
 ]%2 );



   Surface srf1( 2, 2, 2, 2, [<double>
 1 0 0
 0 0 0
 1 1 0
 0 1 0
 ]%3 );

   Sommet 4( crb4( 0 ) ) ;
   Sommet 2( crb2( 0 ) ) ;
   Sommet 1( crb1( 0 ) ) ;
   Sommet 31( crb2( 1 ) ) ;
   Sommet 3( crb3( 0 ) ) ;

   Arete 61( crb1 ) = { 1 2 } ;
   Arete 64( crb4 ) = { 4 1 } ;
   Arete 63( crb3 ) = { 3 4 } ;
   Arete 62( crb2 ) = { 2 31 } ;

   Face 101( srf1 )
= { - 61( srf1( crb1p ) ) - 64( srf1( crb2p ) )
- 63(  - srf1( crb3p ) ) - 62( srf1( crb4p ) )   } ;

};
```

ANNEXE F

Code source du cas test 5 (fichier PIE)

```
Geometrie gest() = {

   Courbe rationnelle courb_1( 2, 2, [<double>
 0 0 0 1
 1 0 0 1
 ]%4 );
   Courbe rationnelle courb_2( 2, 2, [<double>
 1 0 0 1
 1 0 1 1
 ]%4 );
   Courbe umin( 2, 2, [<double>
 0 0
 1 0
 ]%2 );
   Courbe rationnelle courb_3( 2, 2, [<double>
 1 0 1 1
 0 0 1 1
 ]%4 );
   Courbe vmax( 2, 2, [<double>
 1 0
 1 1
 ]%2 );
   Courbe rationnelle courb_4( 2, 2, [<double>
 0 0 1 1
 0 0 0 1
 ]%4 );
   Courbe umax( 2, 2, [<double>
 0 1
 1 1
 ]%2 );
```

```
  Courbe vmin( 2, 2, [<double>
0 0
0 1
]%2 );
  Courbe rationnelle courb_6( 2, 2, [<double>
1 1 0 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_10( 2, 2, [<double>
1 0 0 1
1 1 0 1
]%4 );
  Courbe rationnelle courb_5( 2, 2, [<double>
0 1 0 1
1 1 0 1
]%4 );
  Courbe rationnelle courb_9( 2, 2, [<double>
0 0 0 1
0 1 0 1
]%4 );
  Courbe rationnelle courb_7( 2, 2, [<double>
1 1 1 1
0 1 1 1
]%4 );
  Courbe rationnelle courb_11( 2, 2, [<double>
1 0 1 1
1 1 1 1
]%4 );
  Courbe rationnelle courb_8( 2, 2, [<double>
0 1 1 1
0 1 0 1
]%4 );
  Courbe rationnelle courb_12( 2, 2, [<double>
0 0 1 1
0 1 1 1
]%4 );
```

```
  Surface rationnelle surface_1( 3, 3, 3, 3, [<double>
0 0 0 1
0.5 0 0 1
1 0 0 1

0 0 0.5 1
0.5 -1 0.5 1
1 0 0.5 1

0 0 1 1
0.5 0 1 1
1 0 1 1
]%4 );
  Surface rationnelle surface_2( 2, 2, 2, 2, [<double>
0 0 0 1
1 0 0 1
0 1 0 1
1 1 0 1
]%4 );
  Surface rationnelle surface_3( 2, 2, 2, 2, [<double>
1 0 0 1
1 0 1 1
1 1 0 1
1 1 1 1
]%4 );
  Surface rationnelle surface_4( 2, 2, 2, 2, [<double>
0 0 1 1
1 0 1 1
0 1 1 1
1 1 1 1
]%4 );
  Surface rationnelle surface_5( 2, 2, 2, 2, [<double>
0 0 0 1
0 0 1 1
0 1 0 1
```

```
 0 1 1 1
 ]%4 );
   Surface rationnelle surface_6( 2, 2, 2, 2, [<double>
 0 1 0 1
 1 1 0 1
 0 1 1 1
 1 1 1 1
 ]%4 );

   Sommet 1( courb_1( 0 ) ) ;
   Sommet 2( courb_2( 0 ) ) ;
   Sommet 4( courb_3( 0 ) ) ;
   Sommet 3( courb_4( 0 ) ) ;
   Sommet 6( courb_6( 0 ) ) ;
   Sommet 5( courb_5( 0 ) ) ;
   Sommet 8( courb_7( 0 ) ) ;
   Sommet 7( courb_8( 0 ) ) ;

   Arete 9( courb_1 ) = { 1 2 } ;
   Arete 10( courb_2 ) = { 2 4 } ;
   Arete 11( courb_3 ) = { 4 3 } ;
   Arete 12( courb_4 ) = { 3 1 } ;
   Arete 18( courb_10 ) = { 2 6 } ;
   Arete 13( courb_5 ) = { 5 6 } ;
   Arete 17( courb_9 ) = { 1 5 } ;
   Arete 19( courb_11 ) = { 4 8 } ;
   Arete 14( courb_6 ) = { 6 8 } ;
   Arete 15( courb_7 ) = { 8 7 } ;
   Arete 20( courb_12 ) = { 3 7 } ;
   Arete 16( courb_8 ) = { 7 5 } ;

   Face 21( surface_1 )
= { 9( surface_1( umin ) ) 10( surface_1( vmax ) )
11(  - surface_1( umax ) ) 12(  - surface_1( vmin ) )   } ;
   Face 22( surface_2 )
= { 9( surface_2( umin ) ) 18( surface_2( vmax ) )
```

```
- 13(  - surface_2( umax ) ) - 17(  - surface_2( vmin ) )   } ;
   Face 23( surface_3 )
= { 10( surface_3( umin ) ) 19( surface_3( vmax ) )
- 14(  - surface_3( umax ) ) - 18(  - surface_3( vmin ) )   } ;
   Face 24( surface_4 )
= { - 11( surface_4( umin ) ) 19( surface_4( vmax ) )
15(  - surface_4( umax ) ) - 20(  - surface_4( vmin ) )   } ;
   Face 25( surface_5 )
= { - 12( surface_5( umin ) ) 20( surface_5( vmax ) )
16(  - surface_5( umax ) ) - 17(  - surface_5( vmin ) )   } ;
   Face 26( surface_6 )
= { 13( surface_6( umin ) ) 14( surface_6( vmax ) )
15(  - surface_6( umax ) ) 16(  - surface_6( vmin ) )   } ;

   Volume 27()
= { - 22 - 23 24 25 - 26   } ;
};
```