7e Conférence Internationale de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 – Paris- France
« Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination.».

CARACTERISATION D'INFORMATIONS SEMANTIQUES EXTRAITES DE LA MAQUETTE NUMERIQUE ET RAISONNEMENTS ASSOCIES

V. CHEUTET

LISMMA Supméca Paris
3 rue Fernand Hainaut
93407 Saint-Ouen cedex
vincent.cheutet@supmeca.fr

F. CAMPANICO, J.C. LEON

Laboratoire G-SCOP
Avenue Félix Viallet
38000 Grenoble
Frederic.campanico@laposte.net
Jean-Claude.leon@g-scop.inpg.fr

L. FINE

EADS Innovation Works
12 rue Pasteur, BP 76
92152 Suresnes
Lionel.fine@eads.net

RESUME : *Sur la base de modèles industriels et d'applications logicielles préexistantes, l'objectif du travail proposé est d'évaluer de nouveaux concepts et de nouvelles technologies mettant en œuvre des outils de recherche, de tri et de structuration des informations décrivant un assemblage et donc un produit ou bien un sous-ensemble d'un produit. En particulier, l'utilisation d'ontologies et des moteurs de recherche associés semble aujourd'hui très prometteuse pour formuler et adresser des requêtes portant simultanément sur des informations technologiques et sur des caractéristiques de forme des interfaces entre composants d'un assemblage.*

MOTS-CLES : *informations sémantiques, ontologie, moteur d'inférence, assemblage, maquette numérique.*

1. INTRODUCTION

La maquette numérique d'un produit, vue comme ensemble de modèles participant aux représentations numériques de ce dernier, touche de plus en plus d'activités du processus de développement de produits. En particulier, elle est très fortement utilisée pour aider les concepteurs à générer et manipuler des assemblages décrivant un produit. La diversité et la complexité croissantes des produits, tant par rapport au nombre de composants, à la diversité des formes et des informations technologiques qui sont attachées aux produits, entraînent des besoins importants de gestion et de configuration des assemblages du produit.

A cette fin, les simulations d'Assemblage/ Désassemblage (A/D) du produit prennent toutes leurs importances pour intégrer les contraintes spécifiques à ces opérations. Cependant, pour réaliser des simulations d'A/D efficaces en termes de coût et de qualité, ces simulations ne doivent pas se baser uniquement sur une représentation purement géométrique de l'assemblage. En effet, (Léon et al., 2001) ont pu démontrer que l'ajout d'informations sémantiques et technologiques en amont de la simulation permet de mieux guider la sélection des différentes procédures d'A/D, par rapport à une approche purement géométrique, et ainsi d'obtenir des résultats et des gammes d'assemblage plus significatives. Ces informations technologiques ont également pour effet de réduire la combinatoire propre au séquençage des opérations d'A/D.

De ce fait, les mécanismes permettant d'ajouter et d'exploiter des informations technologiques constituent un enjeu industriel important. Ceci nécessite des traitements numériques adéquats pour extraire, transformer, trier et structurer ces données.

Or, si une maquette numérique est très couramment basée sur une modélisation dite d'assemblage, cette représentation est très limitée car elle est purement descriptive et ne possède pas ou peu d'informations intrinsèques comme l'indique ce qui suit.

Le concept d'entité caractéristique (ou *feature*) s'est développé depuis les années 90 afin d'associer à une représentation numérique des informations sémantiques (Shah and Mantyla, 1995). Ces entités ont pour objectif d'ajouter les informations qui guideront les simulations au cours du cycle de vie du produit, comme par exemple les entités de fabrication (GAMA, 1998). Si ces travaux ont trouvé un très fort écho dans la communauté scientifique, ils restent malheureusement peu concrétisés dans le milieu industriel, du fait de leur manque de déploiement dans les logiciels de CAO (Conception Assistée par Ordinateur). Leur utilisation reste le plus souvent cantonnée au domaine de la fabrication, ce qui les rend peu ou pas utilisable en dehors de ce contexte, comme par exemple dans le cadre de simulation physique qui demande des features adaptés.

Dans le même temps, l'activité de conception est collective et implique de multiples compétences attachées à des groupes d'acteurs contribuant au processus de conception et de développement de produits. Le terme de conception concourante réfère à une organisation de l'activité de conception bien établie (Alting, 1993) (Jo et al., 1993). Ce type d'organisation a renforcé les besoins de prendre en compte les multiples vues produit et les multiples représentations numériques

du produit dans son cycle de vie (Rosenman and Gero, 1996) (MG-IT, 1999). Par conséquent, une multitude d'informations peut être, dans le principe, attachée à la maquette numérique, mais l'information nécessaire ne l'est pas toujours, ou de manière incomplète et son formalisme n'est pas toujours adaptée au métier qui en doit l'utiliser.

Nous proposons dans cet article une approche pragmatique, basée sur l'existant dans le milieu industriel, tant en terme d'outils que de méthodologies. En particulier, nous baserons notre approche sur les hypothèses associées à un cas industriel aéronautique (en collaboration avec EADS IW) : un assemblage complexe, en particulier comme on peut le trouver en aéronautique, est un ensemble de composants, uniquement positionnés en 3D par rapport à un ou des repères de référence, et avec très peu d'informations sémantiques attachées (noms, couleurs, …). En particulier, cette maquette comporte très peu d'information relative à la constitution des l'assemblage : pas d'information cinématique, pas de contrainte de positionnement relatif entre les pièces…

Notre approche évaluera, dans ce contexte, l'emploi d'ontologies pour structurer et transformer les informations attachées à la maquette numérique. Une ontologie est un ensemble structuré de concepts. En intelligence artificielle, la définition communément admise d'une ontologie est énoncée par (Gruber, 1993) : « une ontologie est une spécification explicite d'une conceptualisation d'un domaine ». Cette définition a été par la suite précisée pour devenir : « une ontologie est une spécification formelle et explicite d'une conceptualisation partagée » (Studer, 1998).

Les ontologies sont un moyen pour modéliser les objets de connaissances sur un certain domaine d'intérêt, en lui associant le plus souvent une structure de réseau sémantique. Les outils de développement d'ontologies permettent le partage et la réutilisation des connaissances et la formulation du processus de résolution du problème (Mizoguchi et al., 2000). Ces dernières années, différents travaux ont été proposés pour intégrer les connaissances métiers à différentes étapes du processus de développement de produits à travers l'utilisation d'ontologies, offrant ainsi un environnement de conception sémantique (Kitamura and Mizoguchi, 2004) (Brunetti and Grimm, 2005) (Cheutet et al., 2006). Ainsi, les pré-requis et les caractéristiques du produit établis par les utilisateurs ou choisis par le concepteur peuvent être récupérés et pris en compte durant les différentes étapes de développement de produits. Même les modifications de la forme du produit réalisées durant le processus de conception peuvent être guidées par les informations intégrées dans un sous-ensemble de l'ontologie et, par conséquent, les acteurs successifs du processus de développement du produit peuvent les exploiter pour accélérer le processus de modélisation.

Dans notre contexte, cette technologie émergente peut s'adapter à nos besoins car elle s'appuie sur des structures de données conformes aux analyses de Drieux et al. (2007) avec une approche de type graphe. Pour un assemblage très complexe, cette technologie permet d'élaborer facilement des requêtes nécessaires pour vérifier et/ou ajouter des informations à un assemblage et d'appliquer des inférences pour caractériser les propriétés de cet assemblage. A la connaissance des auteurs, ces travaux sont les premiers à introduire les outils d'inférence dans le cadre de la mécanique.

L'article est structuré de la manière suivante. La section 2 analysera les modèles d'assemblage présents dans les logiciels de CAO et dans le milieu industriel et justifiera notre approche. Les sections suivantes présenteront les étapes importantes de cette démarche : la section 3 présentera brièvement l'algorithme de détection des contacts utilisé, la section 4 développera l'ontologie mise en place dans notre scénario et la section 5 présentera les premiers résultats de raisonnements possibles à l'aide du moteur d'inférence proposé. Enfin, la section 6 conclura et présentera les perspectives de ces travaux.

2. ANALYSE DES MODELES D'ASSEMBLAGE

Pour la plupart des logiciels présents dans le milieu industriel, un assemblage est défini comme étant un ensemble composé :

- De représentations géométriques des composants constituants le produit,
- De mises en position des composants entre eux, par contraintes de positionnement ou par matrice de position dans une scène globale,
- D'informations attachées aux composants, un nom par exemple, sous la forme de chaîne de caractères, une couleur, …,
- D'une structure arborescente de ces composants.

Les trois derniers aspects sont fortement dépendants de l'utilisateur et les fonctions des logiciels ne produisent pas de solution homogène pour définir un assemblage. En particulier, le positionnement de deux composants entre eux peut aussi bien être défini par des contraintes entre différents éléments intrinsèques des composants (coaxialité entre deux axes, coïncidence de surfaces, parallélisme de deux plans, etc.) ou par un positionnement de chaque composant dans une scène de référence (en définissant une matrice de positon pour chaque composant). C'est cette dernière solution qui est utilisée dans le milieu aéronautique, pour les maquettes de grands ensembles. Néanmoins, aucune des deux approches proposées ne repose uniquement sur la définition des surfaces de contact entre composants, alors que cette interface est intrinsèque à une liaison fonctionnelle.

Par conséquent, les outils logiciels actuels permettent uniquement de décrire un assemblage des modèles

géométriques de composants, un arbre et un ensemble d'attributs. Mais ils ne permettent pas de définir un assemblage à partir d'un ensemble de propriétés intrinsèques et, par conséquent, ne peuvent donc pas être considérés comme des modeleurs d'assemblage. Dressons un parallèle pour illustrer ce propos : dans un logiciel de visualisation (par exemple Deep Exploratoin de Right Hemisphere) où un objet apparaît à l'écran et peut être interprété comme un volume, il est décrit uniquement par un ensemble de triangles et en lui assignant un attribut « cube » (par exemple) qui renforce la perception « volumique » de l'objet mais ne donne aucune garantie à l'utilisateur. Dans le contexte d'un modeleur CAO par exemple, celui-ci associe à ce même composant entre autres propriétés, celle de « surface fermée », qu'il peut vérifier à l'aide d'opérateurs spécifiques et des propriétés correspondantes. Ainsi, la perception d'un volume est issue des propriétés sur lequel s'appuie le modeleur. Notre approche est un premier pas vers ce concept de modeleur d'assemblage, dans la création de propriétés caractérisant un assemblage.

Pour ces raisons, et avec notre hypothèse sur les informations présentent dans la maquette numérique, notre approche se base sur les informations dont on dispose effectivement et qui sont intrinsèques au concept représenté, c'est-à-dire à partir des interfaces entre les composants de l'assemblage, on en déduit des propriétés sur les composants.

Notre proposition est d'apporter ces informations sémantiques et technologiques par un processus d'explicitation d'information contenues intrinsèquement et implicitement dans la forme des composants de l'assemblage, leur positionnement relatif et des raisonnements à partir de ces informations. En effet, la forme d'un objet porte une sémantique qui lui est propre et qui permet à un être humain de l'identifier ou d'en identifier une sous-partie (AIM@SHAPE). Dans le cadre d'un assemblage mécanique, cette sémantique est d'autant plus forte que les composants peuvent être normalisés et posséder des informations sémantiques importantes pour les simulations A/D. De plus, les composants ne sont isolés mais placés dans un environnement contextualisé que constitue un assemblage.

Pour extraire des informations de la forme du produit, il faut que le même type de produit soit toujours représenté de la même manière au sein des différents assemblages traités. Malheureusement, il n'existe pas de norme dans la représentation des composants et de leurs interfaces, à la différence du dessin technique. Ainsi, un filetage a, sur un dessin technique par exemple, une représentation normalisée spécifique qui permet de reconnaître directement ce type d'interface dans un assemblage. Dans le même temps, ce même type d'interface n'a pas de représentations numériques 3D normalisées et peut donc changer d'une maquette à l'autre. Néanmoins, nous faisons l'hypothèse de l'existence de représentations conventionnelles des interfaces de composants normalisés au sein d'une entreprise. Ces représentations peuvent être très différentes d'une entreprise à une autre et notre environnement doit pouvoir être flexible afin de s'adapter, ce que nous permet l'utilisation d'ontologies.

Néanmoins, d'autres informations peuvent se révéler nécessaires pour établir ces propriétés, comme par exemple les nomenclatures, le matériau du composant, le composant représentant l'entrée de la puissance dans un assemblage, etc. mais ceci n'a pas fait l'objet d'une étude précise à l'heure actuelle.

Il est néanmoins important de déterminer quel est le niveau minimal d'information à introduire (ou devant exister) dans la maquette pour obtenir les propriétés nécessaires, et ceci afin de limiter le travail fastidieux d'ajout d'informations sémantiques dans la maquette numérique par un utilisateur, de pouvoir maintenir automatiquement les propriétés d'un assemblage, et de supporter l'utilisateur dans ses tâches d'analyse, d'interprétation et d'utilisation d'assemblage mécanique pour des applications pratiques telles que l'analyse cinématique, la simulation comportementale (structure, thermique, dynamique rapide).

Les différentes étapes du flux d'informations sémantiques dans ce contexte s'énumèrent :

- Etape 1 : Explicitation des informations déjà présentes dans le modèle numérique de l'assemblage fourni par un environnement CAO, comme l'arbre d'assemblage, le nom des composants, les informations technologiques déjà présentes, …,
- Etape 2 : Extraction des informations des interfaces entre composants par un algorithme géométrique spécifique,
- Etape 3 : Exportation des informations au sujet des interfaces dans l'ontologie, pour une étape d'explicitation et de structuration des informations,
- Etape 4 : Traitement des requêtes, établies par l'utilisateur ou bien prédéfinies, qui permettent de rechercher certaines informations ou bien d'en générer de nouvelles qui seront elles-mêmes ajoutées à l'ontologie développée.

3. EXTRACTION D'INFORMATIONS SEMANTIQUES LIEES AUX INTERFACES ENTRE COMPOSANTS

Plusieurs algorithmes de détection des contacts existent dans la littérature et à travers certains logiciels. Parmi eux, nous avons choisi les travaux de (Iacob et al., 2007). Ces travaux contribuent à un environnement de simulation pour les analyses d'A/D, intégré dans le logiciel Simpoly développé au sein du laboratoire G-SCOP, dans lequel s'intègre l'identification automatique des contacts dans un modèle 3D du produit,

environnement qui est adapté à nos besoins. Toutefois, les développements actuels sont limités à des conditions de contact qui forment un sous-ensemble des modèles d'interfaces possibles entre composants.

3.1. Une représentation mixte de l'assemblage

Cet environnement contribue à la préparation de processus de simulation adaptés aux deux types majeurs de représentations de formes qui sont utilisés dans le processus de développement du produits, i.e. les représentations B-Rep NURBS (provenant de la CAO) et les polyèdres (utilisés par diverses catégories de simulations) en même temps. Cette représentation mixte du produit permet de traiter efficacement les configurations où les représentations 3D d'un assemblage jouent un rôle clé.

L'environnement proposé se base sur une représentation mixte de la forme de composants/assemblages au niveau de l'interface de la vue produit dédiée à la simulation d'A/D. Dans ce cas, le principe est de considérer que la représentation de référence d'un composant/assemblage, i.e. la représentation maître, est le modèle polyédrique, le modèle B-Rep NURBS étant esclave. La figure 1 présente comment la représentation mixte peut prendre en compte les informations sémantiques attachées aux modèles NURBS d'entrée, comme par exemple la nature de la forme locale d'un objet (plan, cylindre, …) et les informations associées (normale, axe, …). Il est intéressant de noter que ces données sont uniquement de type géométrique et sont disponibles dans les fichiers STEP, générés par les environnements de CAO actuels.

Ce changement de représentation de référence se justifie par le fait que tous les opérateurs de simplification de forme s'appliquent sur le modèle polyédrique et que des changements généraux de forme peuvent apparaître dans le processus de préparation de la simulation. Une telle représentation mixte aide à caractériser la cohérence des deux représentations simultanément, pour tirer avantage de ces deux représentations à chaque étape du processus de développement du produit.

Pour lier ensemble les représentations de la forme de composants/assemblages, cet environnement utilise le concept de HLT (High Level Topology, i.e. Topologie de Haut Niveau) (Hamri et al., 2006). L'objectif principal de la HLT est de permettre la représentation intrinsèque d'informations sémantiques attachées à l'objet considéré.

Dans cette représentation mixte, les éléments de la HLT (HLT-sommet, 'polyedges' et partitions) sont définis à partir des éléments de la représentation polyédrique sous-jacente. Ainsi, un sommet de la HLT coïncide avec un sommet du polyèdre par exemple. De la même manière, les entités de la HLT sont connectées à certaines des entités du modèle B-Rep NURBS décrivant la topologie de ce modèle.

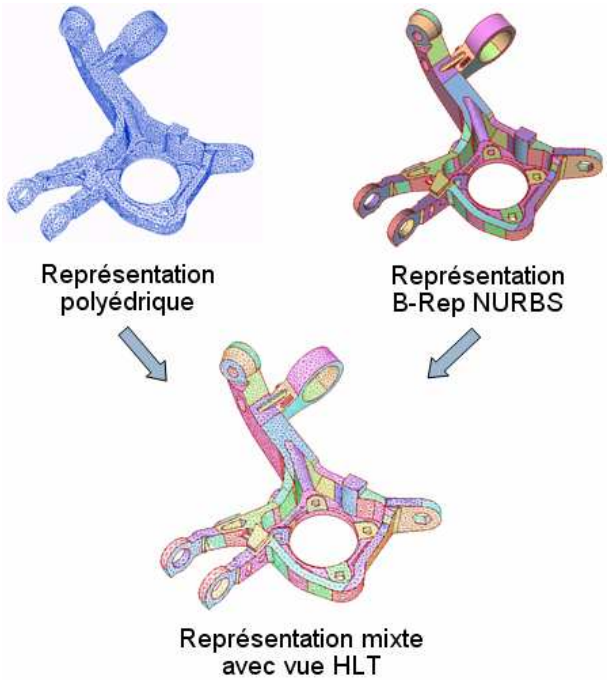

Figure 1. Description schématique de la représentation mixte (Hamri et al., 2006).

3.2. La détection des contacts

La détection des contacts dans un assemblage tire totalement parti de la représentation mixte propre à cet environnement.

Au tout début du processus, une étape de préparation des modèles des composants est réalisée. A partir du modèle STEP, correspondant à la représentation B-Rep NURBS, la représentation polyédrique est obtenue par un algorithme de tessellation contrôlé par les informations de la HLT, informations provenant de la représentation NURBS.

Une fois le modèle polyédrique obtenu, la HLT doit être adaptée aux besoins de la simulation recherchée. En effet, la topologie du modèle B-Rep n'est pas la même que la topologie des surfaces de contact car les systèmes de CAO représentent les surfaces fonctionnelles par un ensemble de carreaux, et ainsi, une surface cylindrique est très souvent représentée par deux ou plus parties de surfaces cylindriques. Il est donc nécessaire de préparer la HLT pour générer les partitions maximales sur la frontière du composant et rendre intrinsèque à la forme du composant sa topologie. Par conséquent, des opérateurs sont appliqués pour assembler les partitions ayant les mêmes paramètres pour les quatre types de surfaces ci-dessous :

- Plan : toutes les surfaces planes localisées dans le même plan et ayant la même normale,
- Cylindre : toutes les surfaces cylindriques ayant les mêmes rayons, axe et normale,
- Cône : toutes les surfaces coniques ayant les mêmes angles, axes, apex et normales,

- Sphère : toutes les surfaces sphériques ayant les mêmes centres, rayon et normale.

A la fin de cette opération, une Liste des Partitions Assemblées (LPA) est ajoutée dans la structure de données HLT. Cette liste est ensuite utilisée par l'opérateur d'identification des contacts.

Actuellement, l'opérateur de détection de contacts ne peut identifier que quatre types de contacts : Appui Plan (APP), Pivot Glissant (PVG), Pivot Glissant Unidirectionnel (PGU) et Rotule (RTL).

Dans un premier temps, à partir des composants de l'assemblage sélectionnés, l'opérateur génère automatiquement une Liste de Corps en intersection (LCI). A cette fin, la boîte englobante de chaque composant est utilisée pour tester l'intersection entre les composants et accélérer ainsi le processus.

Dans un second temps, en utilisant la LCI créée lors de la phase précédente, quatre listes de contacts possibles (LpC) sont créées pour chaque type de surface. Ils comprennent :

- Plan : surfaces coplanaires avec des normales extérieures matière colinéaires et opposées,
- Cylindre : surfaces cylindriques avec même axe, même rayon et des normales extérieures matière opposées,
- Cône : surfaces coniques avec même angle, même axe, même sommet et des normales extérieures matière opposées,
- Sphère : surfaces sphériques avec même centre, même rayon et des normales extérieures matière opposées.

A ce moment, ayant toutes les informations nécessaires dans la structure de données HLT, l'identification des contacts est réalisée. Pour chaque type de contact, les surfaces contenues dans LpC qui appartiennent à des composants de LCI sont testées et une liste est créée :

- Appui plan : surfaces planes avec une zone commune non vide (LcAPP),
- Pivot Glissant : surfaces cylindriques partageant une zone commune (LcPVG),
- Pivot glissant unidirectionnel : surfaces coniques partageant une zone commune (LcPGU),
- Rotule : surfaces sphériques de LpC et de LCI, sachant qu'il ne peut pas y avoir plus de deux surfaces sphériques ayant les mêmes paramètres (LcRTL).

Le processus a été testé sur le modèle de la figure 2, représentant un moteur électrique. Sur ce modèle, 21 contacts « Pivot Glissant » et 14 contacts « Appui plan » ont été identifiés. En particulier, le corps principal du moteur (body) possède 3 contacts APP, un avec le chapeau avant (front cover), un avec le chapeau arrière (back cover) et le dernier avec le stator.

4. EXPLICITATION ET STRUCTURATION DES PARAMETRES DE CONTACT DANS UNE ONTOLOGIE

Au moins deux langages permettent de décrire une ontologie, RDF et OWL (Vacher et al., 2006). Notre ontologie se base sur OWL qui est un langage à balises fondée sur le XML Pour la création de l'ontologie, nous utiliserons le logiciel Protégé qui permet l'édition d'ontologies et qui est développé par l'université de Stanford en open source (Protege).

4.1. Description de l'ontologie

Nous avons créé une ontologie vide, i.e. sans instance, que nous instancierons ensuite pour chaque assemblage considéré. Seuls les éléments nécessaires aux requêtes ont été modélisés dans l'ontologie, afin de ne pas surcharger l'ontologie.

Dans cette section, les classes seront écrites en gras et italique (figure 3), tandis que les propriétés seront écrites en italique et souligné (figure 4).

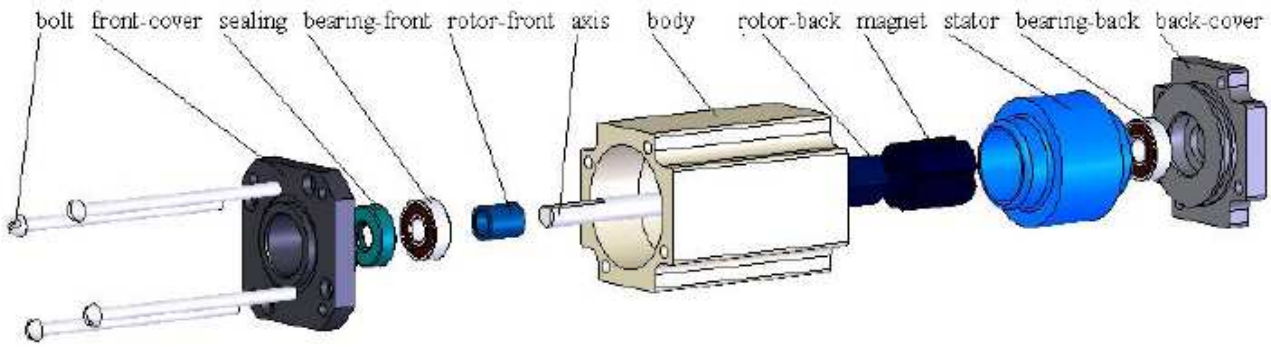

Figure 2. Vue explosée d'un moteur électrique (courtoisie Dynetic System). Courtoisie AIM@SHAPE repository pour la mise à disposition des modèles (http://shapes.aim-at-shape.net/)

4.1.1. Description des classes

Puisque l'étude porte sur un assemblage, les premiers éléments de l'ontologie nous permettent de modéliser les concepts de composants dans la classe ***Component***, et ceux d'assemblages dans ***Assembly***.

Au niveau des relations entre deux composants d'un assemblage, deux configurations ont été modélisées :

- Les deux composants peuvent être en contact et dans ce cas, les informations extraites sont modélisées dans la classe ***Contact***,
- Les deux composants sont en interférence, i.e. ils s'interpénètrent et, dans ce cas, la classe ***Interference*** donnera l'information adéquate.

Un certain nombre d'informations de nature purement géométrique ont aussi été modélisées dans l'ontologie. Elles contribuent à la définition de l'interface entre la présentation 3D des composants et celle de l'ontologie qui doit permettre la formulation de requêtes et d'inférences pour traiter des informations à caractère sémantique.

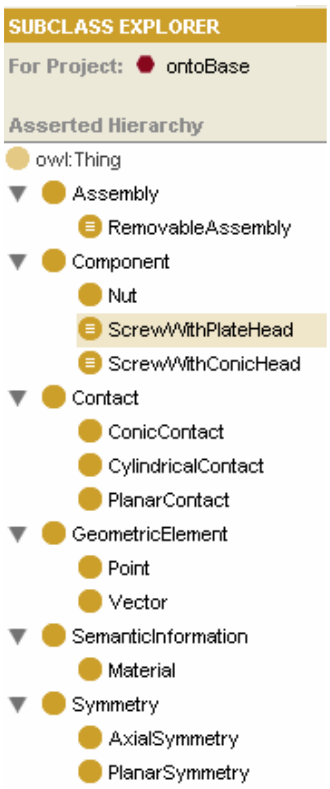

Figure 3. Description des classes de l'ontologie

Ainsi, la classe ***GeometricElement***, qui est liée à la classe ***Contact***, permet de caractériser la zone de contact sans la décrire totalement, en connaissant par exemple la normale à la zone de contact pour un contact plan sur plan ou l'axe dans un contact cylindrique. Elle est composée d'une classe ***Vector*** et d'une classe ***Point*** dans la configuration actuelle.

De la même manière, la classe ***Symmetry*** donne une information sur la symétrie d'un composant, en particulier une symétrie axiale ou plane.

Toutes les autres informations sur les composants, qui sont de type sémantique, sont modélisées dans la classe ***SemanticInformation***. Cette classe se spécialise avec la classe ***Material***, les autres informations sémantiques (par exemple les contraintes de maillage) étant stockées au niveau de la classe ***SemanticInformation***.

Chacune des classes possède des sous-classes qui spécialisent certaines informations pré-existantes ou bien permettent d'ajouter d'autres informations à un assemblage. Par exemple, les sous-classes de la classe ***Component*** sont décrites en fonction de leurs contacts avec leur environnement, leurs symétries, et leurs matériaux. Pour ***Assembly***, la classe ***RemovableAssembly*** regroupe les sous-assemblages qui possèdent au moins une vis, i.e. une instance de la classe ***Screw*** (sous-classe de ***Component***). Une taxonomie partielle des contacts a aussi été modélisée, avec les classes ***CylindricContact***, ***PlanarContact*** et ***ConicContact***, permettant de spécialiser chacun des contacts et surtout d'associer les éléments géométriques adéquats.

4.1.2. Description des propriétés

Afin de caractériser les classes de notre ontologie et de les lier entre elles, un ensemble de propriétés de type objet a été défini.

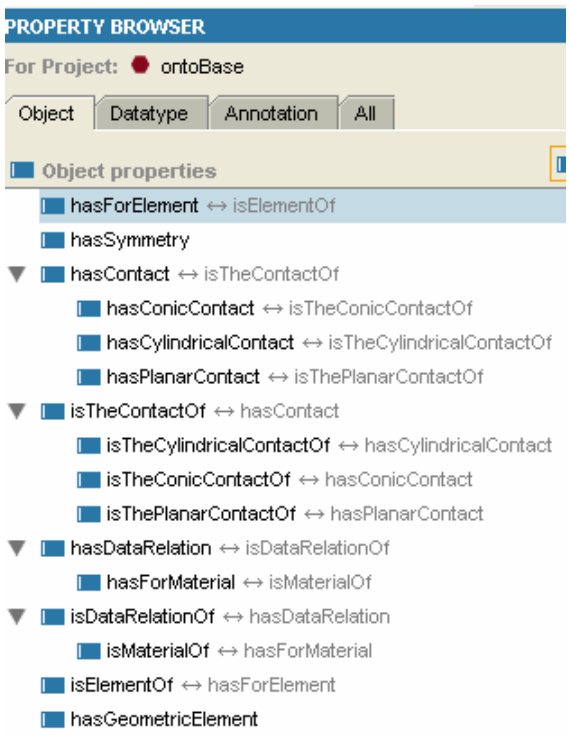

Figure 4. Description des propriétés objets de l'ontologie.

On attache les informations au composant, qu'elles soient géométriques ou sémantiques, à travers plusieurs propriétés. Ainsi, la classe ***Component*** est liée :

- à la classe ***Symmetry*** via la relation *hasSymetry*,
- à la classe ***SemanticInformation*** via la relation *hasDataRelation*;
- à la classe ***Contact*** via la relation *hasContact*, cette relation étant elle-même subdivisée en 3 sous-propriétés pour caractériser le type de contact : *hasConicContact*, *hasCylindricContact*, *hasPlanarContact*;

La relation *hasGeometricElement* lie les classes ***Symmetry*** et ***Contact*** à la classe ***GeometricElement***, afin de caractériser et de « spécialiser » le type de contact et de symétrie.

La propriété *hasForElement* va de la classe ***Assembly*** vers les classes ***Component*** et ***Assembly***. Elle permet de spécifier qu'un assemblage est constitué soit par un composant, soit par un sous-assemblage, et ainsi permet d'avoir une notion de parenté entre un assemblage et des composants. Cette relation est transitive et permet donc de retrouver tous les niveaux de l'assemblage.

Chacune des propriétés décrites ci-dessus possède une relation inverse, afin de pouvoir lire et exploiter les informations dans les deux sens (figure 6).

Certaines classes ont besoin d'avoir des données alphanumériques, c'est pourquoi on a créé des propriétés de données pour ces classes :

- *hasReference* est une propriété (de type string) de la classe ***Component***, qui permet de donner à chaque composant un nom ou une référence complémentaire provenant de la nomenclature, …,
- les instances de la classe ***Material***, dans notre contexte, ne sont caractérisées que par un nom et par une couleur de visualisation donnée en *code RGB via les propriétés hasValueRed*, *hasValueGreen*, *hasValueBlue*, qui n'admettent qu'une seule donnée de type int,
- *hasData* est une propriété de la classe ***SemanticInformation***, elle est de type string et permet d'entrer un champ de caractères spécialisant l'information. Seul, ce niveau de spécialisation est pour le moment retenu, vu le scénario choisi,
- *hasXValue*, *hasYValue*, *hasZValue* sont des propriétés de la classe ***Point*** et ***Vector***. Chaque propriété ne possède qu'une seule valeur de type float et permet de quantifier l'élément géométrique,
- *hasGeometricRepresentation* est une propriété des classes ***Component*** et ***Assembly***, elle est de type String et permet de stocker le nom et l'emplacement de la représentation géométrique du composant et de l'assemblage.

4.2. Création des instances

Comme indiqué précédemment, l'ontologie présentée est initialement vide, et la création des instances est réalisée à travers deux étapes principales. Dans un premier temps, la maquette numérique contenant l'assemblage est analysée. Les instances représentant les différents composants de l'assemblage et l'arbre d'assemblage sont créés dans les classes ***Component*** et ***Assembly***. Si des informations sémantiques sont déjà attachées à la maquette numérique, tant au niveau d'un composant que d'un sous-assemblage, elles peuvent être ajoutées à l'ontologie, sous réserve qu'elles correspondent à des classes prédéterminées. Ainsi, des instances de la classe ***Material*** ou ***SemanticInformation*** peuvent être créées et liées aux instances de ***Component*** ou d'***Assembly***.

Pour ajouter l'ensemble de ces informations dans l'ontologie, nous n'ouvrons pas une session Protégé pour créer les instances mais nous les insérons directement dans le fichier OWL. Pour cela, nous utilisons les propriétés du langage, proche du langage XML, en utilisant un parseur XML (DOM) pour insérer différents item de l'ontologie selon localisation dans la structure de l'ontologie et avec les informations appropriées.

Pour les informations liées aux contacts entre les composants, la recherche des contacts réalisée dans le logiciel Simpoly retourne une liste de l'ensemble des liaisons qu'il a pu trouver. Pour chaque liaison, cette liste contient les couples de partitions qui sont en contact et, par transitivité, les informations sur les normales au contact ou les points de contact. Une étape d'exportation est alors nécessaire pour ajouter ces informations dans l'ontologie et cette étape se base sur l'utilisation du même parseur.

Pour les contacts cylindriques, on exporte les noms des deux composants et les coordonnées de l'axe et d'un point de chaque contact. Pour les contacts planaires, on exporte aussi les noms des deux composants et les coordonnées de la normale de chaque contact.

Par la suite, certaines requêtes voudront comparer deux contacts pour savoir si les normales ou les axes des contacts sont identiques. Malheureusement, les outils d'inférence utilisés dans ce processus ne peuvent pas réaliser des calculs sur les valeurs des données présentes dans l'ontologie durant la phase d'inférence, ils ne peuvent donc pas évaluer une colinéarité entre deux vecteurs, en particulier. Il est cependant possible de vérifier si deux instances de la classe ***PlanarContact*** sont liées à la même instance de la classe ***Vector*** par la propriété *hasGeometricElement*. Par conséquent, il a été décidé que le logiciel Simpoly réaliserait un prétraitement pour ne fournir qu'un seul vecteur représentatif de l'ensemble des vecteurs colinéaires, tant pour une normale à un contact plan que pour un axe de contact cylindrique ou conique.

Une fois ces étapes réalisées, l'ontologie dispose de toutes les informations nécessaires pour être questionnée à travers un premier niveau basique de requêtes.

5. RAISONNEMENTS SUR LES CONTACTS ENTRE COMPOSANTS

Racer (Renamed ABox and Concept Expression Reasoner) est un moteur d'inférences agissant sur des fichiers de type RDF ou OWL (Racer). Son fonctionnement est comparable à celui d'un serveur web : il s'exécute en tâche de fond et les communications se font par des requêtes HTTP. Le langage utilisé pour ses requêtes est appelé nRQL, acronyme de new Racer Query Language.

Pour l'envoi des requêtes, on utilise une interface RacerPorter qui se connecte à Racer.

5.1. Principe des requêtes

Traitons un exemple de requête pour mieux comprendre les possibilités et les limites d'un tel langage. Nous avons dans notre ontologie la classe ***Component***, composée de quatre instances différentes (Component_1, Component_2, Component_3, Component_4), chacune étant uniquement liée à un certain nombre de matériaux suivant le tableau 1.

Component_1	Material_1
Component_2	Material_2
Component_3	Material_1 Material_3
Component_4	Material_2 Material_3 Material_4

Tableau 1. Répartition des matériaux/composants.

La première requête soumise à Racer nous permet de retrouver toutes les instances de la classe Component :

```
(retrieve (?x) (?x |ontology.owl#Component|)).
```

Retrieve est un mot clé pour désigner qu'on veut un résultat. Ce qui suit entre parenthèse « (?x) » indique le nombre de réponses fournies à la fois, une dans le cas présent, qui sera stockée dans la variable x. Le reste est le corps de la requête. On retrouve à côté de la variable, le nom et l'emplacement de classe Component que l'on questionne. Le langage nRQL dispose d'un certain nombre d'outils pour créer des requêtes. Nous avons aussi des opérateurs logiques tels que or, and, union, neg, same-as. D'autres opérateurs permettent de questionner les attributs des instances (int, float, date) : at-least, exactly, at-most, égalité (=) et différence (<>).

Voici un autre exemple de requête qui utilise cette fois-ci un opérateur booléen : on veut récupérer les instances des classes Screw ou Nut.

```
(retrieve (?x)
    (?x (or |ontology.owl#Screw| |ontology.owl#Nut|))
)
```

Pour notre scénario, nous avons besoin de questionner l'ontologie sur le nombre de relations qu'une instance possède avec d'autres classes. Ainsi, on voudrait connaître toutes les instances de la classe ***Component*** qui ont au moins deux matériaux différents. Malheureusement, Racer ne nous permet pas de formuler de telles requêtes directement, nous avons du trouver une alternative. L'idée est de reformuler notre requête de la manière suivante : on cherche un composant qui a une relation avec un matériau et une autre relation avec un autre matériau. La requête s'écrit alors :

```
(retrieve (?comp)
    (and
        (?mat1 |ontology owl#Material|)
        (?mat2 |ontology.owl#Material|)
        (?comp |ontology.owl#Component|)
        (?comp ?mat1 |ontology.owl#isMaterialOf|)
        (?comp ?mat2 |ontology.owl#isMaterialOf|)
    )
)
```

La ligne (?comp ?mat1 |ontology.owl#isMaterialOf|) permet de traduire « une relation avec un matériau ». Racer nous donne alors le résultat attendu, i.e. Component_3 et Component_4.

Pour connaître les composants qui ont au plus un matériau, il suffirait normalement de prendre le complément de la requête, c'est-à-dire les composants qui ont au moins deux matériaux :

```
(retrieve (?comp)
    (neg
        (and
            (?mat1 |ontology owl#Material|)
            (?mat2 |ontology.owl#Material|)
            (?comp |ontology.owl#Component|)
            (?comp ?mat1 |ontology.owl#isMaterialOf|)
            (?comp ?mat2 |ontology.owl#isMaterialOf|)
        )
    )
)
```

Mais, là encore, Racer pose problème. En effet, la réponse théorique à une telle requête devrait être Component_1 et Component_2. Mais, au lieu de cela, la requête affiche toutes les instances de ***Component***.

Le problème vient du fait que Racer n'évalue pas le complément sur le résultat de l'opération booléenne 'and', mais il simplifie l'association des opérateurs 'neg' et 'and' en les remplaçant par un opérateur 'or'. La requête devient donc : trouver les composants qui ont un matériau ou un autre matériau. Or, dans notre cas tous

les composants ont au moins un matériau et Racer nous retourne toutes les instances de ***Component***.

Pour résoudre ce problème, il est nécessaire de séparer la requête en deux en récupérant d'abord les composants ayant au moins deux matériaux (R1) puis en établissant l'ensemble complémentaire dans la classe ***Component*** (R2) :

```
R1: (retrieve (?comp)
      (and
            (?mat1 |ontology owl#Material|)
            (?mat2 |ontology.owl#Material|)
            (?comp |ontology.owl#Component|)
            (?comp ?mat1 |ontology.owl#isMaterialOf|)
            (?comp ?mat2 |ontology.owl#isMaterialOf|)
      )
)

R2: (retrieve (?comp)
      (?comp |ontology.owl#Component|
      (and
            (<> |ontology.owl#Component_3|)
            (<> |ontology.owl#Component_4|)
      ))
)
```

5.2. Exemple de règle de caractérisation de propriétés d'un assemblage

Une fois l'appropriation du langage des requêtes faite, nous avons pu établir un ensemble des règles relatives à des propriétés d'interfaces entre composants d'un assemblage et s'appuyant sur des informations sémantiques liées aux contacts.

En particulier, nous avons approfondi la caractérisation de vis comme attribut fonctionnel de composants à partir des informations des différents contacts (plan, cylindrique, conique) ou interfaces avec leurs composants adjacents. Nous avons fait le choix de spécifier des règles permettant de caractériser une vis de manière nécessaire, i.e. les composants trouvés contiennent obligatoirement l'ensemble des vis, mais pas suffisante, i.e. parmi les composants non sélectionnés par la propriété, aucun d'entre eux ne peut correspondre à une vis. Ainsi, la propriété mise en évidence ne peut actuellement garantir l'unicité du processus de caractérisation de vis. Ce travail fera l'objet de développements ultérieurs.

Pour établir ces règles, il convient d'abord de faire des hypothèses sur la représentation d'une vis dans une maquette numérique afin de définir notre représentation conventionnelle utilisée comme référence. Dans un premier temps, nous avons émis comme hypothèse que la vis serait représentée idéalisée, i.e. sans son filetage, sans jeu représenté géométriquement et sans interférence c'est-à-dire que le diamètre extérieur de la vis est égal à tous les diamètres intérieurs des composants traversés par la vis (figure 5).

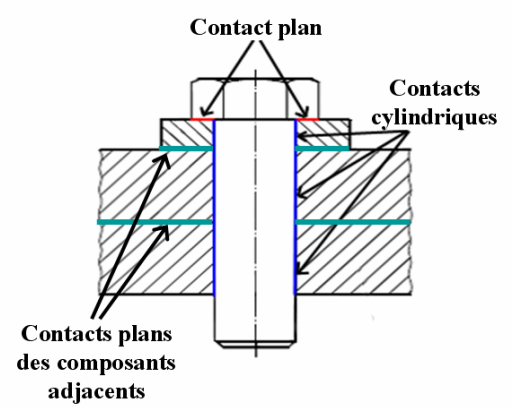

Figure 5. Hypothèses de représentation numérique d'une vis dans un assemblage.

Il convient de remarquer que ce choix correspond à des configurations défavorables pour la caractérisation de vis car elle les place dans des configurations proches de liaisons cinématiques de type pivot. Le choix de représentations conventionnelles alternatives, i.e. avec jeu et/ou interférences constituerait une classe d'interface plus spécifique au cas des liaisons vissées.

La Figure 6 montre l'intérêt de mettre en place un principe de caractérisation des vis à partir de leurs interfaces car la représentation conventionnelle proposée, basée sur des propriétés des interfaces, est applicable à un grand nombre de catégories de vis.

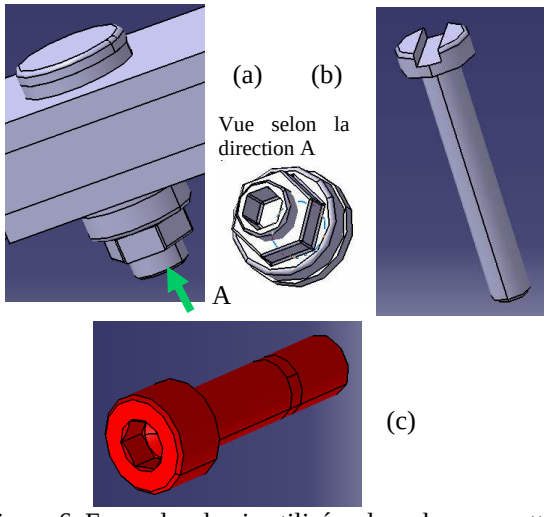

Figure 6. Exemples de vis utilisées dans des maquettes numériques d'assemblages. a) vis spéciale utilisée dans le secteur aérospatial (courtoisie EADS, LMT Cachan), b) vis à tête fendue, c) vis à tête six pans creux.

Sous cette hypothèse, une première règle peut s'écrire ainsi :

« Un composant ayant un seul contact plan et au moins deux contacts cylindriques est une vis ».

Après quelques tests sur des maquettes numériques différentes, on a constaté que cette règle n'était pas suffisamment caractéristique des vis : il existait des composants qui ne sont pas des vis mais qui étaient considérés comme tels avec cette règle, comme un chapeau par exemple. Ceci montre que les informations minimales des interfaces caractérisant une vis n'étaient pas suffisantes. Pour spécifier plus précisément les vis, nous avons enrichi cette règle en utilisant l'environnement du composant et les caractéristiques des contacts (normale pour un contact plan, axe pour un contact cylindrique). La règle devient la suivante (voir figure 5) :

« Un composant ayant un seul contact plan et ayant au moins deux contacts cylindriques dont les composants adjacents sont « empilés », i.e. ils doivent avoir au moins un contact plan en commun, et dont la normale au contact plan et les axes des contacts cylindriques sont les mêmes, est une vis »

Cette nouvelle règle a été testée sur plusieurs maquettes numériques qui satisfont la représentation conventionnelle définie. Nous l'avons en particulier testée sur l'exemple du moteur électrique déjà utilisé dans la section 3 (voir figure 2) et Racer a bien identifié les quatre vis présentes dans ce modèle (figure 7).

A ce jour, aucun contre-exemple, tant conceptuel que réel, n'a été trouvé pour infirmer que la règle proposée était nécessaire, i.e. tous les composants sélectionnés par cette règle étaient bien des vis, avec le choix des hypothèses de représentation conventionnelle des vis dans la maquette numérique. Néanmoins, cette règle n'est pas suffisante. La figure 8 illustre, sur une maquette numérique d'assemblage simple, l'empli de la représentation conventionnelle proposée. Dans ce cas, les deux composants en contact avec la vis n'ont pas de contact direct entre eux mais ont un contact avec la même pièce. Il est donc nécessaire de rajouter des règles complémentaires pour couvrir l'ensemble des configurations avec des vis. De même, si la représentation conventionnelle est modifiée, les règles seront-elles-aussi à modifier et à adapter au nouveau contexte.

Cette propriété permet donc dans une approche de type modeleur soit de nommer automatiquement le composant dans un assemblage (et par conséquent de lui donner une fonction), dans le cas où aucune information n'était attachée auparavant au composant, soit d'analyser la cohérence de l'assemblage vis-à-vis de la représentation conventionnelle de l'entreprise et des désignations des composants spécifiées par des utilisateurs. Cette information pourra ensuite être utilisée dans la simulation A/D pour permettre une élaboration des gammes plus efficaces en terme de coût et de qualité des résultats.

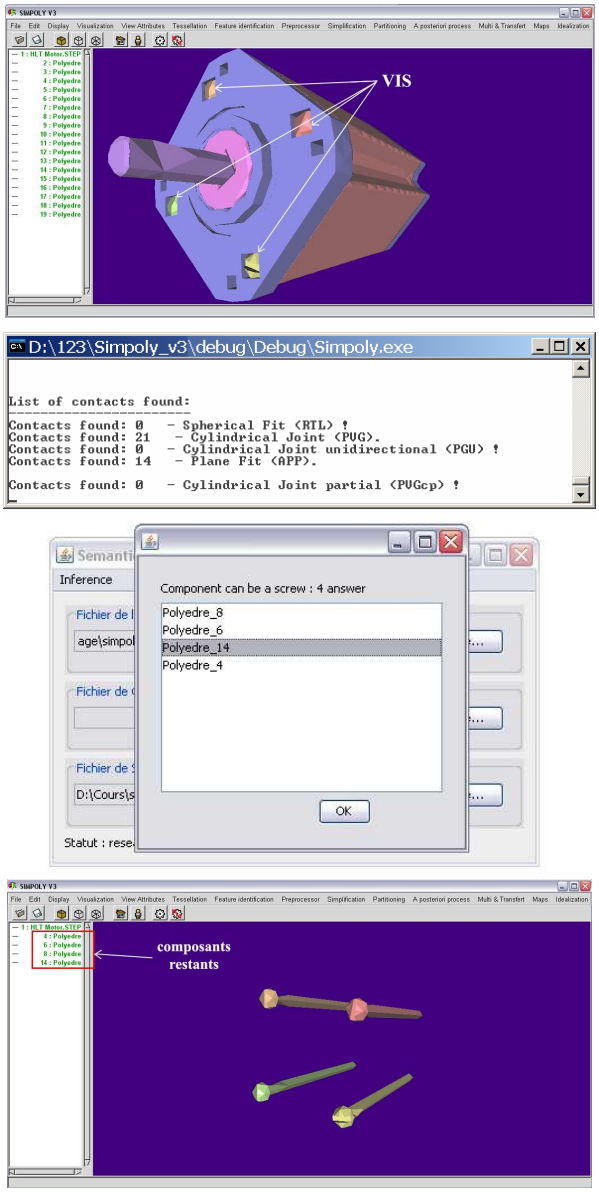

Figure 7. Illustration de la séquence d'opérations permettant de caractériser les vis contenues dans l'assemblage du moteur électrique.

6. CONCLUSION ET PERSPECTIVES

Ces travaux ont montré un premier niveau de faisabilité pour l'utilisation de technologies de type ontologie pour mettre en œuvre des outils de recherche, de tri et de structuration des informations décrivant un assemblage avec, en particulier, l'utilisation d'ontologies et d'un moteur d'inférences. Ces travaux ont été utilisés pour intégrer des règles métiers dans l'insertion d'informations sémantiques permettant de compléter les simulations d'A/D et à terme de créer un modeleur d'assemblage.

Deux pistes sont envisagées pour le futur. La première piste est d'évaluer la robustesse et la durée des traitements de ce type d'approches face à des maquettes

numériques d'assemblage industrielles, où le nombre de composants peut dépasser très rapidement le millier.

La seconde est d'intégrer cette approche dans un contexte plus complet, où plusieurs points de vue interagissent sur une maquette numérique et où l'exploitation de données sémantiques permettrait une meilleure communication entre les différents acteurs.

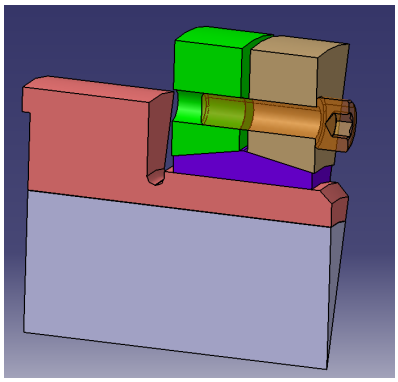

Figure 8. Exemple de maquette numérique d'assemblages comportant un montage de vis conforme à la représentation conventionnelle définie à la figure 5 (courtoisie LMT Cachan).

REMERCIEMENTS

Ces travaux s'intègrent dans le partenariat actuel du laboratoire G-SCOP avec EADS-IW et s'inscrivent dans le contexte du réseau européen d'excellence AIM@SHAPE, Contrat de la Commission Européenne IST 506766.

REFERENCES

AIM@SHAPE: Advanced and Innovative Models And Tools for the development of Semantic-based systems for Handling, Acquiring, and Processing knowledge Embedded in multidimensional digital objects, *European Network of Excellence*, Key Action: 2.3.1.7 Semantic-based knowledge systems, VI Framework, URL: http://www.aim-at-shape.net.

Alting L., 1993. Life-cycle design of products: a new opportunity for manufacturing enterprises. *Concurrent engineering: automation, tools and techniques*, Wiley Inter Science.

Brunetti G. and Grimm S., 2005. Feature Ontologies for the Explicit Representation of Shape Semantics. *International Journal of Computer Applications in Technology*, vol. 23, pp. 192-202.

Cheutet V., Leon J.C., Catalano C.E., Giannini F., Monti M., Falcidieno B., 2006. Preserving car stylists' design intent through an ontology. Proceedings of 6th Int. Conf. on Integrated Design and Manufacturing in Mechanical Engineering, Grenoble, France.

Drieux G., Léon J.C., Guillaume F., Chevassus N., Fine L., Poulat A., 2007. *Interfacing product views through a mixed shape representation. Part 2: Model processing description*, Int. Journal on Interactive Design and Manufacturing, Vol. 1(2), pp.55-126.

Groupe GAMA, 1998. *Modélisation par entités*, Conception de produits mécaniques, Hermès, Paris.

Gruber T.R., 1993. A Translation Approach to Portable Ontology Specification. *Knowledge Acquisition*, vol. 5, pp. 199-220.

Hamri O., Léon J.C., Giannini F., Falcidieno B., Poulat A., Fine L., 2006. Interfacing product views through a mixed shape representation. Part 1: Data structures and operators. *Research in Interactive Design*, Springer Verlag, vol. 2, pp. 72-86.

Iacob R., Mitrouchev P., Léon J.C., 2007. A simulation framework for assembly/disassembly process modeling. *Proceedings of the ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2007)*, Las Vegas, USA.

Jo H.H., Parsei H.R., Sullivan W.G., 1993. *Principles of concurrent engineering, Concurrent engineering, contemporary issues and modern design tools*. edited by H.R. Parsei and W.G. Sullivan, Chapman & Hall.

Kitamura Y. and Mizoguchi R., 2004. Ontology-based systematization of functional knowledge. *Journal of Engineering Design*, Vol. 15, No.4, pp. 327-351.

Léon J.C., Rejneri N., Debarbouillé G., 2001. Assembly/disassembly simulation early during a design process. *Proceedings of the ASME 2001 International Design Engineering Technical Conferences*, Pittsburg, USA, p. 1-9.

MG-IT (collective name of the working group of Pôle Productique Rhône Alpes), head: J-C. Léon, 1999. Multi-views and multi-representations design framework applied to a preliminary design phase, *Integrated Design and Manufacturing in Mechanical Engineering*, Kluwer, pp 553-560.

Mizoguchi R., Kozaki K., Sano T. and Kitamura Y., 2000. Construction and deployment of a plant ontology. *Proceedings of 12th International Conference EKAW2000*, pp.113-128.

Protégé web page: http://protege.stanford.edu/

Racer: http://www.racer-systems.com/index.phtml

Rosenman M.A. and Gero J.S., 1996. Modelling multiple views of design objects in a collaborative CAD environment. *CAD Journal (Special Issue on AI in Design)*, Vol. 28, n°3, 207-216.

Shah J.J., Mantyla M., 1995. *Parametric and feature-based CAD/CAM*, Wiley-Interscience Publication, John Wiley Sons, Inc.

Studer B.F., 1998.Knowledge Engineering: Principles and Methods. *Data and Knowledge Engineering*, Vol. 25, pp. 161-197.

Vacher A., Deshayes L., Brissaud D., Tichkiewitch S., 2006. Toward the use of ontologies for scientific knowledge modelling and integration in production community. Proceedings of 6th Int. Conf. on Integrated Design and Manufacturing in Mechanical Engineering, Grenoble, France.