COMPUTER-AIDED DESIGN & APPLICATIONS, 2018
VOL. 15, NO. 3, 300–317
https://doi.org/10.1080/16864360.2017.1397882

From CAD assemblies toward knowledge-based assemblies using an intrinsic knowledge-based assembly model

Harold Vilmart [a], Jean-Claude Léon [a] and Federico Ulliana [b]

[a]Grenoble University, LJK laboratory, Inria team IMAGINE; [b]Montpellier University, LIRMM laboratory, Inria team GRAPHIK

ABSTRACT
CAD assemblies are essentially reduced to a set of components, often solids, and a user-defined tree structure. Given the generation process of CAD assembly models where the user may instantiate several times a given component, some solids may occur more than once. If this tree structure incorporates some functional information, this is not mandatory and it cannot be regarded as a reliable functional description. Similarly, component occurrences may not always end up being simple copies of a given solid. To this end, we introduce the concept of intrinsic assembly model and describe and illustrate an associated set of geometry processing operators that can produce an intrinsic shape descriptor of assembly components and extract assembly structure using symmetries, alignments, . . . As a complement, it is described how this intrinsic model can become an intrinsic knowledge-based assembly model. Some geometric concepts are mapped to symbolic information using ontology and new structural assembly information is derived using inferences. All these automated processes and mappings enforce the consistency of the proposed model. Illustrative examples show that this model is a first basis toward a functional description of an assembly where new inference rules can be added to express and characterize functional information. A website http://3dassblyanlysis.gforge.inria.fr/3d/ gives a public access to a knowledge-based assembly example (available with IE and Firefox navigators).

KEYWORDS
Assembly; ontologies; CAD models; symmetry analysis; repetitions; shape similarity; knowledge modeling; B-Rep models; RDF; inference

1. Introduction

CAD assembly models reduce to sets of B-Rep models, often solids, without explicit geometric relationships between them, i.e., each component is located using its reference frame position with respect to a sub-assembly reference frame or to the assembly reference frame. When exported through STEP files, these files can contain the assembly tree that structures the assembly and dependencies between solids and sets of solids, i.e., sub-assemblies that express the occurrence of components or sub-assemblies through the whole assembly. Here, occurrences designate components or sub-assemblies that share the same shape. However, this assembly structure derives from the generation process of a CAD assembly, which is a user-driven process. Indeed, the generation of sub-assemblies is up to the user's interpretation of a product and there can be no unique solution to group components to form sub-assemblies (see Figure 1). Furthermore, dependencies between components expressing occurrences of components, either as set into a CAD assembly or as described into an assembly STEP file, is also subjected to user's interactions. Hence, creating these occurrences is error prone [18]. If it is straightforward for a user to instantiate many times the same component when it is a product-specific component, standard components extracted from libraries like TraceParts [41] can be extracted multiple times, possibly by different engineers, thus generating different components sharing the same shape. All these observations show that any assembly model possibly contains an inconsistent or incomplete structure.

From a complementary standpoint, it can be observed that 3D CAD models can be inserted into knowledge-based approaches to structure design knowledge that interact with 3D shapes [10]. Given the difficulty to extract and formalize design knowledge, knowledge-based approaches have found industrial applications often limited to routine design and modifications where the 3D shapes are not evolving significantly [19] because they are constrained by CAD modeler dependencies. Knowledge-based approaches are recognized as means to describe functional information and able to set up reasoning processes that perform at a functional level. Functional information is often described as symbolic

CONTACT Harold Vilmart Harold.Vilmart@inria.fr; Jean-Claude Léon Jean-Claude.Leon@grenoble-inp.fr; Federico Ulliana Federico.Ulliana@lirmm.fr

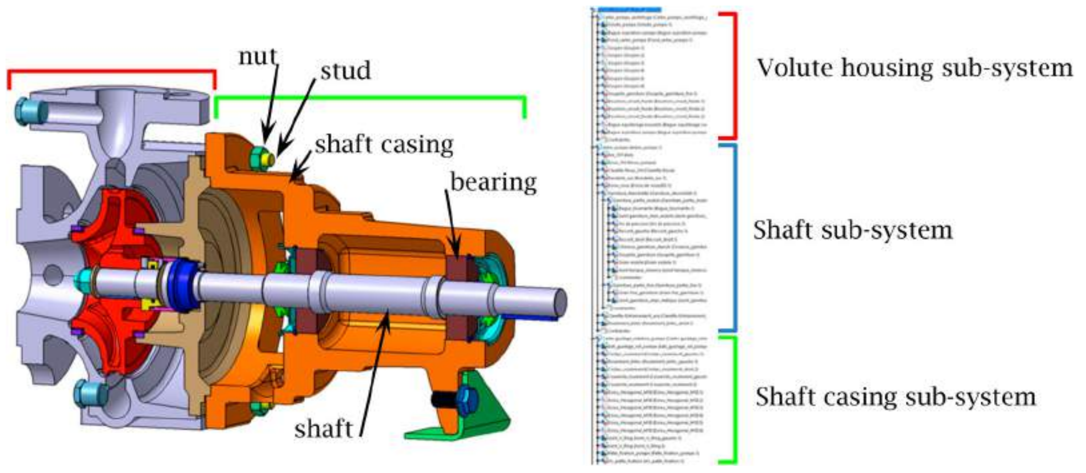

Figure 1. An example of assembly tree structure of a hydraulic pump. Colored brackets indicate the location of sub-systems in the tree structure and in 3D as well. The studs are part of the volute housing sub-system and the nuts can be part of this subsystem or the shaft casing sub-system as well. The bearings are part of the shaft sub-system but could have been placed in the shaft casing sub-system.

information devoted to inference mechanisms or querying systems [6, 29, 34, 37, 42].

Here, the purpose is the proposition and implementation of an assembly model that is intrinsic, i.e., the CAD assembly tree defined by a designer can express some of its functional aspects only.

This intrinsic model is defined by a set of intrinsic geometric information, i.e., independent of the surface parameterization, the modeling process of each component and the CAD assembly tree. It contains also symbolic information as part of an ontology where the geometric properties associated with the intrinsic geometric assembly model produces elementary facts that populate the knowledge base connected to the CAD model.

These geometric properties are obtained algorithmically; hence the knowledge base is populated automatically. From this set of elementary facts, the inference engine part of our software architecture is equipped with inference rules. Triggering these rules enables the generation of new, higher level facts related to the assembly structure that relates to the functional structure of the assembly. These facts enrich the knowledge base so that it can be queried in accordance with the ontology set up. The resulting architecture becomes a foundation to evolve toward knowledge-based assemblies whose interests have been already highlighted in areas such as design purposes [10] and the preparation of finite element models of assemblies [7]. Similarly, the concept of assembly model appears when searching assemblies in databases [21, 22, 45], defining assembly/disassembly processes [15, 16, 30, 32, 44]. As a result, we can observe that there are strong interests in setting up an assembly model covering more than the assembly structure and in addressing assembly knowledge to raise assembly information up to the level of functions. This is the current objective of the ongoing research work.

The description of the proposed contribution is organized as follows. Section 2 reviews prior work regarding assembly models as well as knowledge based approaches. Section 3 introduces the concept of intrinsic geometric assembly model. Section 4 describes the software architecture set up to develop the assembly model from a geometric point of view as well as a knowledge based one. Section 5 introduces the main features of the proposed assembly model but it is not intended to go into their detailed algorithmic description to preserve the conciseness of the paper. Also, it describes features of the knowledge-based of the assembly model. Section 6 shows results and performances of the proposed assembly model that can relate to various applications.

2. Related works

From a 3D geometry standpoint, CAD assembly models have been enriched with geometric interfaces between components [7, 11, 15, 16, 21, 30, 34, 37, 45] to provide a subset of the intrinsic assembly model targeted. Some of these contributions, however, differ from each other regarding the categories of interfaces extracted. Given the common practice of engineers, Shahwan et al. [34] highlighted three main interface categories, i.e., contact, interferences, clearances, taken into account in [7, 16, 21, 30, 34]. Zhang et al. [45] approach relies on a surface analysis using a point sampling process that cannot straightforwardly be extended to cope with interferences that characterize volumes. Hsu et al. [15] refer to contacts only, leaving the user interactively annotating these interfaces, which is adequate to add some technological information but does not lead to an intrinsic assembly model. Chen et al. [11] refer to interfaces but don't give details about the categories handled and how an assembly is processed to extract this information. Swain et al. [37], use some threshold to locate face pairs but do not state the effective criterion to conclude about parallelism, threaded areas, and rely on standalone part features to characterize joints without referring to their common imprint. Generally, clearances and contacts are separated based on a user-defined threshold. This is a first approach that requires enhancement to become more generic and robust. Similarly, it appears important to obtain precise geometric information about geometric interfaces to obtain an effective assembly model that can be either used for different applications [7] or characterize constitutive inconsistencies.

Now, it is also important that the geometric interfaces produce invariant information under equivalent patch decompositions. As an example, for an interface describing a surface contact, the targeted quantity is invariant when the interface stays embedded in the same surface while its patch decomposition is changing. Given the approach of Zhang et al. [45], the graph structure derived from the contact areas will be influenced by different patch decompositions, other approaches [15, 16, 30, 37] don't address this issue while [7, 21] incorporate processing that refer to the concept of maximal faces introduced in Li et al. [6, 20]. A full topology is needed similarly to [20] to extract geometric and topological information characterizing the various geometric interfaces.

If geometric interfaces are extracted in [7, 11, 21, 30, 34, 45], a graph representing the relationships between interfaces and characterizing the assembly structure is derived in [7, 11, 34] though Chen et al. [11] neither mention categories of interfaces processed and how the graph is effectively derived. In all configurations, contacts between components are restricted to surface contacts leaving linear and punctual contacts unaddressed.

Following the requirements mentioned in the introduction to produce an intrinsic assembly model, the assembly tree cannot be regarded as reliable to characterize component occurrences as acknowledged by Iyer et al. [18]. This means component occurrences must be extracted using geometry processing. This extraction process is equivalent to find shape similarities. According to the classification of shape retrieval methods of Tangelder et al. [38], graph-based methods are most frequent for CAD models. Indeed, their B-Rep description produces a graph representation that can be efficient to compare components. Other methods rely on facetted representations of 3D objects. These methods however, are not relevant in the present context because they incorporate an approximation, i.e., the facettisation, generating a deviation from the original model that is orders of magnitude larger than the accuracy of a CAD modeler. Consequently, components having dimensions differing by more than the CAD modeler accuracy could be regarded as identical, which would contradict the concept of component occurrences as they are generated using isometric transformations of components. Zhang et al. [45] use a shape descriptor of components to discover common structures. Components are B-Rep models but the shape descriptor uses a sampling process that can hardly be adjusted to stay compatible with the accuracy of the CAD modeler even though the quantities defining the shape descriptor are curvature-based ones [23, 36] that are independent of the boundary decomposition into patches as well as the patch parameterization. Lupinetti et al. [21, 22] use a shape layer attached to components as part of their assembly descriptor. Without referring to the assembly tree structure, the identification of occurrences of components is obtained through repeated queries to obtain effective occurrences of components. Other approaches characterizing the similarity of components compare graphs [13, 39] but these approaches refer to approximate comparisons and they do not take into account the effect of symmetry properties over the posing effect of objects or regions. As a consequence, there is no effective answer to the problem of finding components occurrences in a CAD assembly model.

In accordance to the above review, there is no intrinsic assembly model that can be robustly derived from CAD assemblies and process a large range of interfaces between the components. Occurrences of components are derived either from an assembly tree or using repeated shape matching, showing the possible improvements required to obtain an effective intrinsic assembly model.

From a knowledge representation standpoint, knowledge-based assembly models are often addressed from a top-down approach [10, 20, 24] and require user input to annotate components or sub-assemblies, which is not robust and is often associated with the early design phases where product function specification and reasoning is at play [40]. Complementary approaches, closer to bottom-up approaches relate 3D shapes to engineering knowledge [5, 15, 19, 27, 34, 37, 44] or, more generally shape semantics [6, 29]. Most of these contributions fit into the field of knowledge based engineering [19].

Knowledge based systems coupled to CAD modelers give access to 3D geometry processing like Knowledge Fusion in NX [5, 19] to be able to monitor part modeling processes interacting with engineering knowledge. Consequently, this is based on a set of geometric entities and operators as available in CAD modelers and the requirements of independence of the boundary decomposition of components mentioned above cannot be satisfied. More generally, the side effects of construction processes are still present and another limitation holds in the fact that the set of geometric entities accessible is closed and cannot be extended because the code is proprietary. Often, inserting new knowledge processing capabilities requires a distributed architecture [4, 27, 34, 42] as illustrated by Moitra et al. [27] to generate manufacturing features extraction rules, Shahwan et al. [34], Ulliana et al. [42], to generate functional information.

Knowledge representation frameworks are commonly specified with description logics languages like the lightweight DL-Lite [9], or the expressive Horn-SHIQ [12], or, more generally, rule-based languages representing fragments of first-order logic like Datalog± [8] and existential rules [3]. Each knowledge representation language has a distinctive set of modeling features and expressivity. This led to the implementation of a number of reasoners each one optimized for a specific language [7, 14, 35].

Learning-based approaches using neural networks or other techniques, rely essentially on input data with a wide range of datasets. Here, there is no well-defined representative set that can be used as basis for such approaches. Consequently, these approaches are not in the scope of the proposed approach.

In order to set up an intrinsic assembly model and extract robustly generic information about component occurrences and geometric interfaces between components, learning-based approaches are not suited and proprietary knowledge-based engineering systems are not open enough to insert new geometry processing and new concepts. Here, we promote robust information through the effective use of the CAD assembly model and real knowledge modeling. Throughout the following sections, the purpose is not to go into the details of each feature of the proposed architecture and functions to keep the paper as concise as possible. Rather, it is aimed at highlighting the key features of each function that effectively contribute to the availability of an intrinsic assembly model that can be enriched with functionally related knowledge.

3. Concept of intrinsic assembly model

The two previous sections have referred to the concept of intrinsic assembly model. The purpose, here, is the description of this concept. It is assumed that:

Every component of an assembly is described as a B-Rep model, which is the commonly available description either in commercial CAD modelers or in STEP files [17];
The boundary of each component has faces embedded in canonical surfaces, i.e. plane, cylinder, cone, sphere, and torus. Most mechanical components satisfy this constraint [26] and some functions described afterward have extensions to free-form surfaces. This issue is essentially left for future work.

The major features of an intrinsic assembly model are listed as:

1. The boundary description of every component must be:
   - Independent of its modeling process. This means the multiple construction processes of a given solid M must produce the same boundary decomposition. On the one hand, Fig. 2a shows an example where a pocket feature is defined with a contour embedding a symmetry property (see Fig. 2b). As a result, the corresponding decomposition of the contour (orange and magenta subsets in Fig. 2b) generates an edge E that is not intrinsic to the shape of M. On the other hand, if the same pocket contour is generated without this explicit symmetry decomposition, E is not going to appear in the boundary of M. Concepts of maximal faces [6,

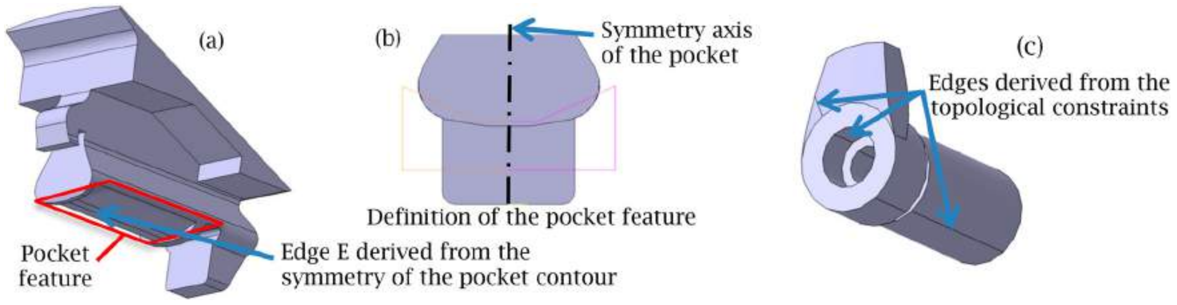
(a)
Pocket feature
Edge E derived from the symmetry of the pocket contour
(b)
Symmetry axis of the pocket
Definition of the pocket feature
(c)
Edges derived from the topological constraints

Figure 2. (a) B-Rep of a mechanical component having a face decomposition illustrating the influence of an effective symmetry property in the contour of a pocket. (b) Contour of the pocket bearing the symmetry property, (c) edges part of a boundary decomposition of M that derives from the homeomorphism condition required to apply the generalized Euler theorem.

20] or specific treatments [21, 39] can be used to obtain sets of faces becoming intrinsic to the shape of M;

- Independent of the topological constraints of geometric modelers. Indeed, geometric modelers are subjected to conditions so the generalized Euler theorem must be applicable to M [25] to perform shape transformations during a construction process. One of them is expressed by the fact that the decomposition of the boundary of M must produce sub-domains that are homeomorphic to a plane. This ends up as a constraint for surfaces of revolution where a circle in a sketch may generate two half cylindrical faces in M (see Fig. 2c), depending on the modeler considered, e.g. CATIA V5, or only one surface in M that is split along one generatrix, e.g., SALOME, or one cylindrical surface that is bounded by two closed edges only, e.g., Siemens NX. In the latter case, however, the Euler theorem is not readily applicable. At least, one such edge must appear because the concept of homeomorphism relates to the existence of an embedding of a parametric plane into each surface defining a face of M, e.g. OpenCascade, SALOME, and the location of this edge derives from the underlying parameterization of the surface of revolution. Removing these edges ends up with a topological boundary description that becomes independent with respect to any parameterization of the underlying surface as well as the homeomorphism condition. This is achieved with [6, 20] but there is no detail in [21, 39] regarding this equivalence.

However, this independence is not fully achieved with the face merging process only. It is mandatory to carry on applying the same analysis at the level of edges and this is achieved in Li et al. [20] where an edge merging operator performs similarly. Altogether, specific treatments are also applied to vertices to generate a boundary decomposition that meets the independence requirements. This is achieved using three hypergraphs [20] forming a data structure intrinsic to the shape of M. This data structure is not superseding the ubiquitous B-Rep CAD one, rather it is a new one that is connected to the B-Rep CAD one (see Fig. 3a);

2. Geometric properties extracted from components that can be part of subsequent modeling operations must be obtained at the level of accuracy of the modeler. This is often expressed as a linear tolerance, ϵ (mm), used to connect faces, edges, etc. A typical value of ϵ is 10^{-3} mm for commercial modelers. Subsequent modeling operations refer to geometry processing that may be applied to components to extract geometric properties at the level of the assembly. As an example, reflective symmetry properties of a component [20] can derive symmetry planes that must be located with respect to M within the tolerance ϵ to ensure that these planes can be used to cut M if necessary and that any operation equivalent to the inverse operation produces a valid object. This example generalizes to state that the geometry processing operators, o_i, $i \in \{1, \cdots, n\}$, must form a *closed set*, $\mathcal{O}$, i.e., $\forall o_i \in \mathcal{O}, \forall M, N, \cdots$, representing components (solids) or other geometric entities consistent with respect to ϵ and forming the operands of o_i, then $(M, N, \cdots) \xrightarrow{o_i} (P, Q, \cdots)$ where $(P, Q, \cdots)$ is the set of geometric entities output by o_i that are consistent with respect to ϵ;

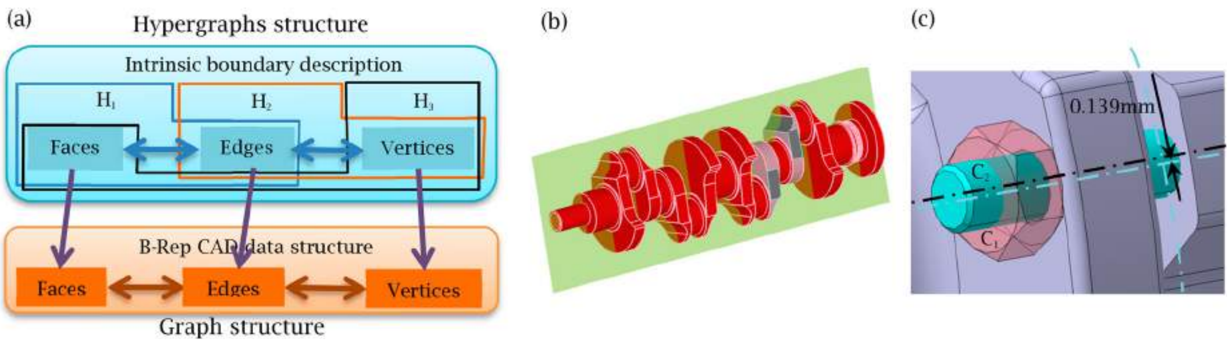

Figure 3. (a) Intrinsic boundary description data structure connected with the CAD B-Rep data structure. H_1, H_2, H_3 are hypergraphs expressing adjacencies between (faces, edges), (edges, vertices), (faces, vertices), respectively. (b) Component highlighting inconsistencies of feature position or shape based on a symmetry analysis. Pink and gray areas indicate regions inconsistent with respect to the symmetry plane (green). (c) Inconsistent relative position of C_1 wrt. C_2 due to differences of hole axes positions. Axes of C_1 and C_2 are located on circles having different radii.

3. The product structure description must be independent of the CAD assembly tree to avoid incorporating user-dependent and/or company-dependent content, which is not intrinsic to an assembly model;
4. Rather, the product structure must be based on the geometric interfaces between components [34]. The corresponding intrinsic description is a graph structure rather than a tree structure. Identical components must be characterized independently of their names to generate a robust identification [18]. This identification process should be performed as an operator $o_k \in \mathcal{O}$, i.e., o_k must enrich the closed set of CAD modeler operators. Similarly, the identification of groups of components G_c and other geometric properties of an assembly must be generated within the tolerance ϵ of the modeler to enable the re-use of G_c or any other component M as input of any o_i. Any group identification operator o_G, or other assembly property generator, o_k, must belong to $\mathcal{O}$. If o_G operates on sets of components, i.e., solids, it does not mean that o_G is somewhat different from common CAD modeling operators, e.g., feature-based operators like extrusion or pocket. Indeed, G_c, from a geometric standpoint, can be seen as a single solid with multiple components, as it can be described using the generalized Euler operator [25], thus categorizing o_G and similar operators in the range of solid modeling operators. Now, some of these operators may produce so-called cellular models [28] or, more generally, non-manifold models that are needed to generate simulation models, e.g., for finite element (FE) simulations. As an example, two components M_1 and M_2 sharing a planar contact, i.e., a planar area of M_1 coincides with a planar area of M_2, can be transformed into a cellular model when one of these planar areas is removed and the remaining boundary of the corresponding component is connected to the other component. Common CAE software, e.g., Abaqus, ANSYS, SALOME, perform somewhat similar operations. Here again, these softwares incorporate solid modeling kernels forming a set $\mathcal{O}$ with respect to a tolerance ϵ;
5. Similarly to components, geometric interfaces supporting the graph structure describing the intrinsic assembly structure must bear an intrinsic boundary description. This means interfaces of type contact or interference must be identified using a tolerance ϵ so that the areas thus detected can be processed with operators belonging to $\mathcal{O}$. Within such a context, clearances cannot be seen as another category of contacts whose distance parameter would be some distance $\beta \gg \epsilon$. In this case, operators from $\mathcal{O}$ cannot be applied. If the operators described in the following sections to identify clearances refer to a user-defined threshold, it is a first simple approach leaving a more generic approach to future work;
6. The assembly model accuracy, i.e., the tolerance ϵ, must be similar to the tolerance value used for the geometry processing of assemblies at other dependent product development steps. This means the tolerance ϵ used to process the geometry of components or sets of components as mentioned in the previous items of the current list must be similar to the tolerance used to generate the solid model of each component, which characterizes the product definition phase performed at the engineering office and takes place prior to the setting of the assembly model currently described. A similar observation can be set up with regard to subsequent assembly processing when considering the preparation of assemblies for FE simulations, for example. When considering other similar dependencies along a product development process, the requirement for preserving the consistency of the tolerance ϵ can be propagated as needed. However, if this analysis favors the concept of an integrated software environment, this is not strictly necessary because B-Rep CAD models are not mandatory at all steps of a product development process.

The previous list has enumerated the key geometric and topological features of a so-called intrinsic assembly model. As a consequence, components lacking some of the geometric properties required to set up the intrinsic assembly model, or failing under some geometry processing with an operator $o_k \in \mathcal{O}$, may reveal geometric inconsistencies in the assembly model processed. Here, inconsistencies differ from those where components fail to satisfy the required properties for defining a solid. The latter are often encountered when solid models are exchanged among different software, S_1 and S_2, describing solids using tolerance values ϵ_1 and ϵ_2, respectively. Here, we refer to inconsistencies where components can be effectively processed as solids but fails to produce the required assembly model. Figure 3 shows two examples of these inconsistencies:

- A component failing to produce a desired global symmetry property (see Fig. 3b). This is highlighting a modeling inconsistency at level higher than ϵ when the user has located some features with respect to others;
- An assembly where the relative position of features belonging to different components is inconsistent and generates inconsistent geometric interfaces between these components (see Fig. 3c). Often, these

inconsistencies are not straightforwardly visible, i.e., not visible on screen but larger than ϵ, because they cannot derive from the built-in update processes available in CAD or CAE software.

To the experience of the authors, these inconsistencies are rather frequent in industrial assemblies as well as assemblies available on open websites and it appears to be a contribution of the concept of intrinsic assembly model to be able to highlight such inconsistencies that are otherwise difficult to find and to locate. It seems that among the contributions of a knowledge-based assembly model, the intrinsic geometric assembly model specified can be a first step to ease the consistency check of these models and locate more rapidly some sources of inconsistency. Based on the above geometric features characterizing an intrinsic assembly model, the following section focuses on the software architecture enabling geometry processing as well as symbolic information processing to model and process assembly knowledge. Here, this knowledge is addressed generically at a geometry level. Functional and other mechanical engineering information will be addressed elsewhere for sake of conciseness.

4. System architecture to process geometry as well as symbolic information

In order to evolve from CAD assembly models toward knowledge-based assemblies, it is mandatory to relate some 3D geometry of an assembly to a knowledge representation. The review of related works has pointed out that ontology-based systems provide us with enough flexibility to describe a wide variety of concepts as well as capabilities to be coupled with inference engines to process this knowledge.

The software architecture set up incorporates a CAD modeler, part of the SALOME software platform [33], developed by OpenCascade, EDF R&D and CEA. SALOME is devoted to the numerical simulation of physical phenomena and addresses the geometry generation of a simulation domain, the mesh generation and the numerical simulation and its post processing, similarly to ANSYS, Abaqus and other similar software. SALOME being an Open source software, it offers the capability to generate new entities more easily than using APIs of other commercial CAD or CAE software, which partly justifies our choice. SALOME architecture decomposes into modules among which GEOM is the geometric modeler (see Figure 4). This modeler has no specific capability to model and process assembly models, likewise other commercial CAE software. Effectively, the geometric modeler data structure of CAE software is strictly based on the concept of geometric object, i.e., a solid or a surface domain, possibly containing several disconnected components but there is no concept of assembly tree in such modules. Currently, the knowledge-based assembly model, named MyProductFabrica, is available through a specific SALOME module that can communicate and exchange geometry with GEOM.

To be able to describe, store, and process knowledge, a knowledge base, JENA, developed by Apache, is connected to MyProductFabrica to describe the ontology as RDF triplets stored in a triple store. A reasoner to process inferences described as RDF triplets, CoGUI, developed by GraphiK Inria team (see Figure 4) is connected to the knowledge base. Indeed, the reasoner can be substituted by other equivalent modules, e.g., the inference engine GRAAL, developed by GraphiK Inria team, or other inference engines to process RDF triplets. CoGUI, however, is not only a reasoner but it contains also conceptual graph editing capabilities. Equivalently, the architecture could use Protégé [31] rather than CoGUI.

Based on this architecture, a STEP file describing an assembly is input into MyProductFabrica and processed

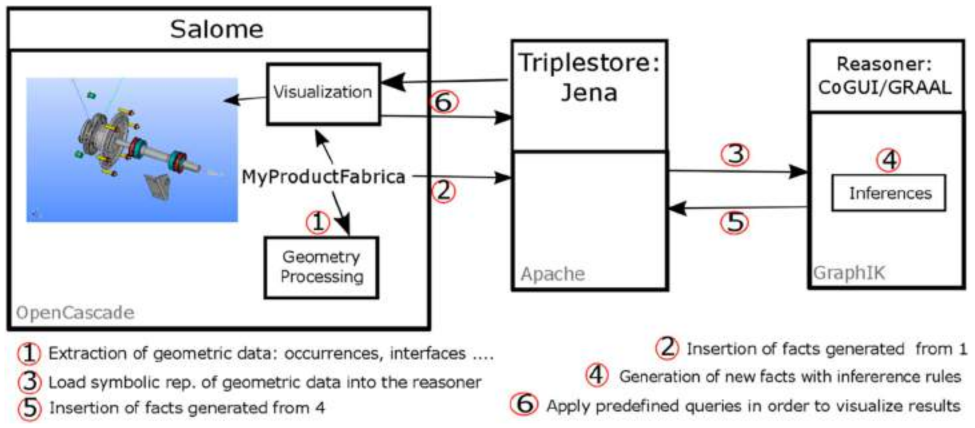
Salome
Visualization
MyProductFabrica
Geometry Processing
OpenCascade
Triplestore: Jena
Apache
Reasoner: CoGUI/GRAAL
Inferences
GraphIK
1
2
3
4
5
6
1 Extraction of geometric data: occurrences, interfaces ....
3 Load symbolic rep. of geometric data into the reasoner
5 Insertion of facts generated from 4
2 Insertion of facts generated from 1
4 Generation of new facts with infererence rules
6 Apply predefined queries in order to visualize results

Figure 4. Software architecture incorporating a new module in SALOME, MyProductFabrica, a knowledge base, JENA, and a reasoner CoGUI and interactions between these components.

as follows to illustrate how geometry processing and symbolic information are derived, feed the knowledge base, and produce new knowledge through inferences (see Figure 4):

- Geometry processing of each assembly component and of the assembly itself takes place in MyProductFabrica. The geometric information thus generated is stored in MyProductFabrica (step 1). A symbolic information can be attached to it to instantiate concepts of the ontology as RDF triplets and form a symbolic representation of some geometric concepts;
- Each symbolic information derived from a geometry processing algorithm generates a corresponding fact that is inserted in the knowledge base, JENA, as RDF triplets to populate the ontology (step 2);
- Symbolic information is loaded into the reasoner, CoGUI, (step 3) and processed using inference rules expressed as RDF triplets forming queries (step 4). The facts derived from these inferences are stored into the knowledge base (step 5). In simple configurations, inferences may not be mandatory and the knowledge base can be accessed directly by queries;
- The visualization of the results is achieved in MyProductFabrica, taking advantage of the connections set up between the geometric entities describing the assembly and the entities resulting from the geometry processing algorithms (step 6). Using these connections, the symbolic information produced by the queries can be expressed with the proper 3D geometric entities.

This architecture is derived from previous work of Ulliana et al. [42] since the knowledge basis can incorporate other symbolic information, e.g., human anatomy, and bind ontologies together.

The above description of the elementary steps contributes to the enrichment of the initial assembly model to form a typical cycle transforming an assembly into a knowledge-based one. As an example, the extraction and comparison of geometric descriptors of components can generate symbolic information expressing the property 'C_1 *HasSameShape* C_2' between solids C_1 and C_2. The comparison of geometric descriptors being algorithmic, the facts inserted into the knowledge base may not conform to all their symbolic properties, e.g., 'C_2 *HasSameShape* C_1'. To ensure the consistency of the knowledge base, it is subjected to the so-called saturation process that accounts for some properties of facts, e.g., symmetry, transitivity, to generate new facts and ensure its consistency. Then, querying the knowledge base for occurrences of components produces a set of facts that identify components and, hence solids, that can be selected in MyProductFabrica and visualized in 3D.

Among the most important features of this architecture, it is important to point out that the consistency of the knowledge base always holds because new facts are always inserted automatically from MyProductFabrica or derived from CoGUI inferences and the saturation process derives automatically all the facts required to keep the knowledge base up to date. Consequently, the knowledge base can be regarded as an effective knowledge model of an assembly that is equipped with processes to maintain the consistency of this model under ranges of modifications.

5. Geometry and symbolic information processing

Prior work on automatically and functionally enriched assemblies [7, 34] has demonstrated the efficiency of structuring assemblies for simulation preparation purposes. Inferring functions was obtained through repetitive applications of rules to each component, resulting in a lengthy process. Consequently, geometry processing operators are important to take into account repetitive configurations in assemblies so that similar configurations between components can be identified and used to infer component functions.

5.1. Geometry processing operators

Here, the focus is placed on the high level description of the geometry processing operators and some symbolic information processing to produce a synthetic overview of interactions between 3D geometric entities and symbolic ones enabling the knowledge-based description of assemblies.

Based on Section 3, the following geometry processing operators described conform to most of the corresponding requirements. Among these requirements, the one addressing the consistency of the operators, i.e., the closed set $\mathcal{O}$ has been implemented, i.e., the operators set up are all consistent with the accuracy of the geometric modeler, GEOM, and they are aligned on the same tolerance $\epsilon = 10^{-3}$ mm.

The geometry processing algorithms perform the following major treatments. Each of them is not detailed for sake on conciseness and their details will be published elsewhere. They come as:

1. The generation of the maximal boundary decomposition of each a component to obtain a boundary that is intrinsic as described in Section 3. This concept derives from Boussuge et al. [6, 20] and

has been extended to be able to handle different boundary decompositions that all fulfill the conditions of Section 3 but may conform to additional criteria [6, 20], [6]. Figure 5 gives an illustration of this decomposition through two of the associated hypergraphs;

2. The symmetry analysis of each assembly component. It outputs a set of global symmetry planes (see Figures 3b and 6) with the algorithm of Li et al. [20] and uses the maximal boundary decomposition obtained previously. Consequently, the symmetry planes and/or axes obtained are intrinsic to the component shape. Further, the symmetry planes obtained can be organized in accordance with ten categories of symmetries to cover shapes of solids. Figure 6 illustrates four such categories. The ten categories of global reflective symmetries enumerate:
   a. No symmetry plane;
   b. One symmetry plane (see Figure 6a);
   c. Two symmetry planes. They are necessarily orthogonal (see Figure 6b);
   d. Three symmetry planes. They are necessarily orthogonal to each other (see Figure 6c);
   e. Discrete axi-symmetry. All the symmetry planes intersect along a common line defining the 'axis' of the solid (see Figure 6d). Simple examples of these shapes are solid pyramids with a regular polygonal basis. In that category, the number of symmetry planes can become arbitrarily large;
   f. Discrete axi-symmetry plus plane. In addition to the previous category, there exists one symmetry plane that is orthogonal to all the others. To fit into this category, such solids have four symmetry planes, at least;
   g. Axi-symmetry. The solid possesses an axis of symmetry that is equivalent to an infinite number of symmetry planes intersecting along its axis;
   h. Axi-symmetry plus plane. In addition to the previous category, there exists one symmetry plane that is orthogonal to the symmetry axis. Simple examples of these shapes are cylinders;
   i. Discrete central symmetry. All the symmetry planes intersect at a common point and that point is the apex of pyramids defined from the symmetry plane intersections. The solid angle of every such pyramid is constant. This is equivalent to have a finite number of discrete symmetry 'axes' sharing a common point;
   j. Central symmetry. Here, there are an infinite number of symmetry axes. The only solid benefiting this property is the sphere.

   Observing the above descriptions, it can be seen that these categories are independent of each other. This information is automatically attached to every solid of an assembly. A

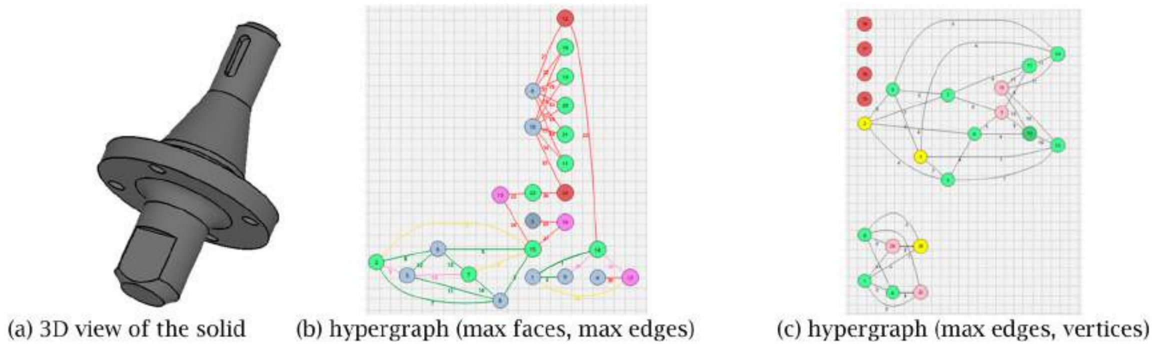
(a) 3D view of the solid
(b) hypergraph (max faces, max edges)
(c) hypergraph (max edges, vertices)

Figure 5. Example of boundary decomposition of a solid (a). (b) hypergraph describing the adjacency (max face, max edge). Node color indicates the category of surface (type of canonical surface). Arc color gives information about the underlying edge geometry. In case of solids this hypergraph reduces to a graph. (c) hypergraph describing (max face, max edge) adjacency. Node color indicates the geometry of the max edge. Hyperarcs are decomposed into binary arcs to conform to the Graphviz library capabilities.

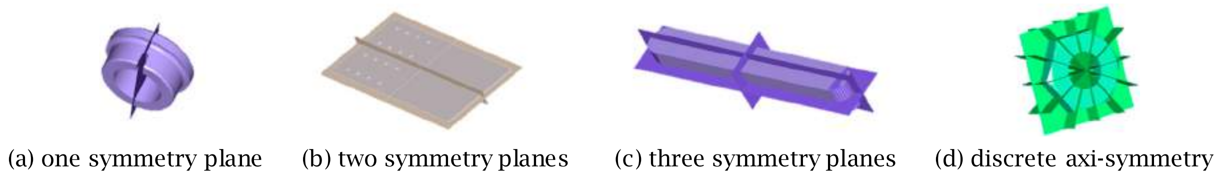

Figure 6. Global symmetry properties of components organized into categories. Examples with four independent categories: (d) is a nut.

corresponding demonstration of these properties will be published elsewhere. These categories of symmetries are valid for any solid whether bounded by canonical surfaces or free-form ones;

3. The generation of a geometric shape descriptor for each solid M. Here, some key features are given but a detailed demonstration will be published elsewhere. This descriptor is based on the maximal boundary decomposition of M (item 1), on its global symmetry properties (item 2) and its inertia properties. The descriptor is defined with the center of gravity, G, of M, which is a unique point, and a reference frame, R_G, defined from its principal axes of inertia. However, the principal axes of symmetry may not be unique depending on the global symmetry properties of M. To this end, the categories of global symmetry properties (items 2a-2j), S_G, are used to characterize the uniqueness of each principal axis of inertia. In connection with the global symmetry properties S_G, the hypergraphs H_i, $i \in \{1, 2, 3\}$, of maximal boundary decomposition are processed to reduce them to their minimal subsets H_{Ri}, $i \in \{1, 2, 3\}$. Thus, G, S_G, H_{Ri}, the geometric parameters of each canonical surface, and their spatial location in R_G, P_C, characterize the finite number of isometries that can reduce R_G to a unique and intrinsic reference frame, R_{GM}, of M. Altogether, (G, S_G, H_{Ri}, P_C, R_{GM}) is intrinsic to M and forms its descriptor. Figure 7a gives an example of the usage of this descriptor to identify occurrences of components in an assembly;
4. The extraction of identical components. Identical shape descriptors (see Figure 7a) produces occurrences of components or, more precisely, of solids. The proposed descriptor is able to remove the isometries between components related to their center of gravity as well as the isometries related to the global symmetry properties that define R_{GM}. This solves the posing problem [38] and approximate comparisons of their H_{Ri} and P_C [13, 39], or the use of discretizations [23, 38, 45], can now be avoided. Using the proposed descriptor, it is equivalent to consider that components have a distance zero [38]. Though these occurrences may appear in the STEP file and assembly tree describing the assembly, their names are neither robust nor intrinsic compared to the proposed shape descriptor;
5. The extraction of families of components. This is a weaker form of the previous geometric descriptor where the comparison uses S_G, H_{Ri}, but P_C is no longer used. Additionally, local symmetry properties [20] are added to characterize the desired set of shapes (see Figure 7b). These properties are equivalent to geometric properties of type parallelism, orthogonality among others. Components belonging to the same family are bound to the same topology, as defined by H_{Ri}. Though this is a first approach, it could be extended to cover topological variants [43] but it is already efficient to characterize components like screws, nuts, bearings, . . . ;
6. The symmetry properties of an assembly. Based on item 2, this symmetry analysis is extended to determine sets of components benefiting symmetry properties (reflection planes, symmetry axes). It uses the global symmetry properties of each component independently, the concept of intrinsic reference frame of each component (see item 3), and the concept of occurrence (see item 4) to propagate symmetry planes across components (see Figure 8a). Similarly to solids, categories of symmetries can be defined for assemblies;
7. The repetition properties of components. It is a generalization of the symmetry analysis [20] to characterize the spatial relationships among components. Linear and circular patterns are detected based on the intrinsic reference frame assigned to each component (see item 3) after occurrences of components are made available (see item 4). They form three sets of properties:

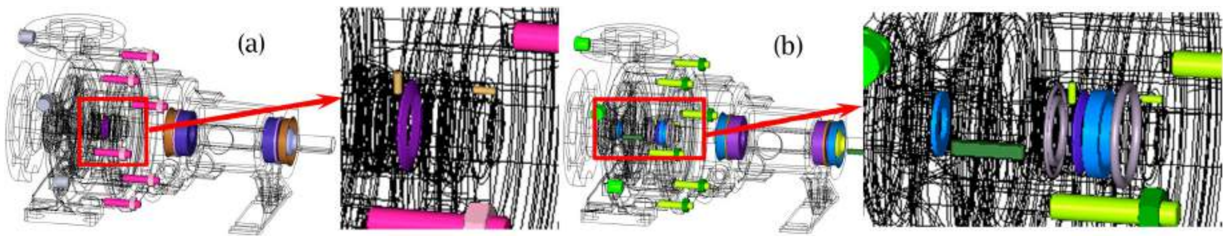

Figure 7. (a) Extraction of categories of identical components. Each color identifies a different category. (b) Extraction of families of components: Each color identifies a different family. The results have to be compared with (a) to distinguish the newly identified components.

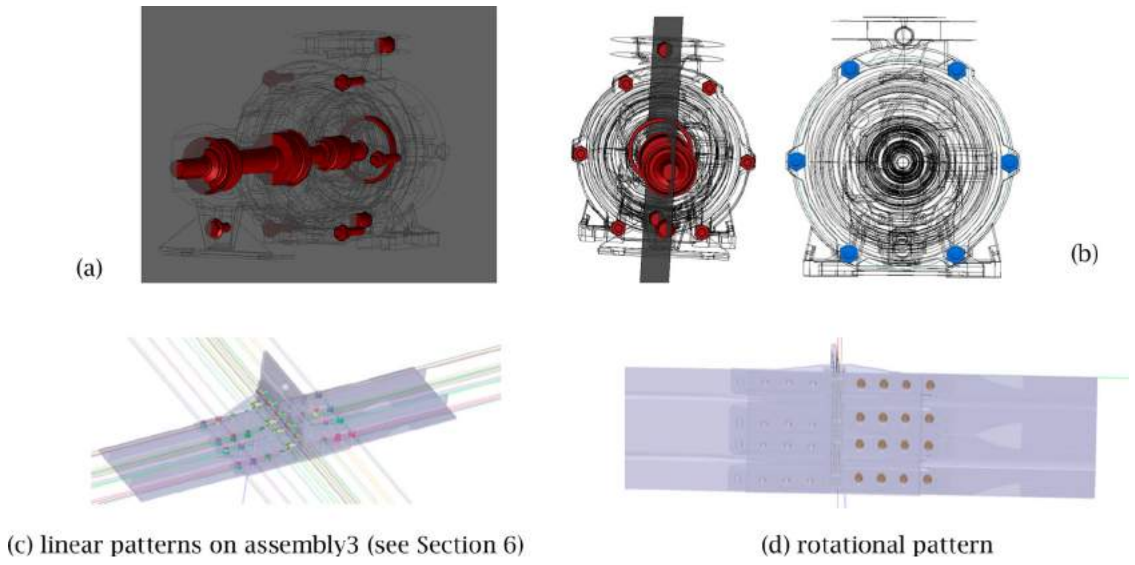

Figure 8. (a) Extraction of symmetry planes: A symmetry plane attached to a set of components (in red). The corresponding set of components is of maximal cardinality. The two views depict the same plane. (b, c, d) Extraction of repetitive patterns of single components. (b) rotational pattern related to nuts. Due to the global symmetry property of nuts (see Figure 6d), it is simultaneously of type radial as well as translational circular. (c) shows all the translational patterns. (d) shows an example of '2D linear' one.

a. Rectilinear repetition of components;
b. Rotational repetition of components that are radially distributed (see Figure 8b);
c. Rotational repetition of components that are translationally distributed according to a circular pattern (see Figure 8b). Indeed, this category of repetitions reflects the construction process of assembly models where a given component is instantiated several times, e.g., nuts connected to screws, while their relative position is not entirely constrained by their neighboring components;

Their cardinalities are: R_L, R_R, R_{TR}, respectively. This is an extension of Lupinetti et al. [21, 22] using information intrinsic to an assembly (see Figure 8b, c, d). These properties can be fairly combinatorial. In order to reduce the cardinality of R_L, R_R, R_{TR}, rectilinear repetitions are extracted when three components, at least, are aligned. Similarly, rotational repetitions are assigned a lower bound of four components;

8. The extraction of repeated sets of components whose relative positions are identical. This generalizes the concept of sub-assembly and these sets are called 'modules' (see Figure 9). More precisely, a module can be defined as a set of components, $C_M = \{C_i, \cdots, C_j, C_k\}$, each of them occurring twice, at least. Also, C_M characterizes the relative positions of its constitutive components, $C_i, \cdots, C_j, C_k$ and C_M forms a module if the assembly contains two such instances, at least. Extracting all the modules brings intrinsic information that participates to the definition of intrinsic sub-assemblies. However, it is important to point out that the extraction of all the modules is heavily combinatorial since, on one side, card(C_M) is unknown and should be maximal in the given assembly and, on the other side, the number of instances of C_M is also a result of the extraction and maximal with respect to the set of components forming the assembly. These two maxima are interdependent since the removal of a component in C_M can raise the number of instances. Presently, symmetry

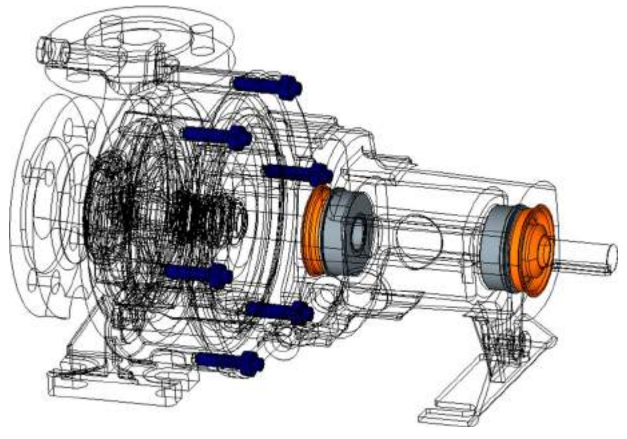

Figure 9. Extraction of repetitive patterns of sets of components forming 'modules'. Each color identifies a different set of components. The orange and grey modules effectively differ because clearances are not distributed symmetrically between occurrences of these two sets.

properties of the assembly (item 7) are used to speed up the extraction of modules;

9. The extraction of the graph of geometric interfaces between components similarly to Boussuge et al. [7, 34]. Currently, the graph has been extended to incorporate interferences, contacts and clearances (see Figure 10). This extraction is based on geometry processing operators currently available in CAD geometric modelers, thus ensuring the use of closed set of operators consistent with the tolerance ϵ. Clearances are derived from a user-defined distance, which is not intrinsic to the assembly but its generalization is left for future work. The geometric interfaces are structured into categories, e.g., contact areas are subdivided into categories: planar, cylindrical,, ... , that can be related to functional information (see Figure 13).
   Interfaces are processed to extract geometric properties, e.g., characterizing contact areas between canonical surfaces of revolution of angle $\pi <, > \pi$ or $= 2\pi$. Sub categories thus obtained are useful information for functional purposes or assembly process purposes.

All these geometric informations produce geometric properties and some of them can be described as symbolic information and concepts of the assembly ontology. The corresponding process flow is depicted in Figure 11. Symbolic information populates the knowledge base, which is currently restricted to generic geometric properties. Functional or other application dependent informations are part of ongoing work and will be described elsewhere.

5.2. Symbolic information generation and processing

Here, the issue is the characterization of geometric information, as made available from section 5.1 and through the geometry of each solid of the CAD assembly, that can become a symbolic information inserted into the knowledge base while staying connected to its geometry counterpart so that this symbolic information becomes part of the intrinsic model of the assembly. The tree structure possibly available is not addressed here since the information contained is neither robust nor intrinsic.

Somehow, the objective is the extension of the intrinsic model described in Section 3 to symbolic information. One justification for such an extension holds in the fact that it is aimed at describing and processing assemblies at a knowledge level and, more precisely, a functional knowledge level. Another justification derives from the fact that assembly knowledge has been barely explored and its structure is rather unclear, therefore using an algorithmic approach is not suitable since the programming effort may be severely reconsidered if data structures require significant modifications.

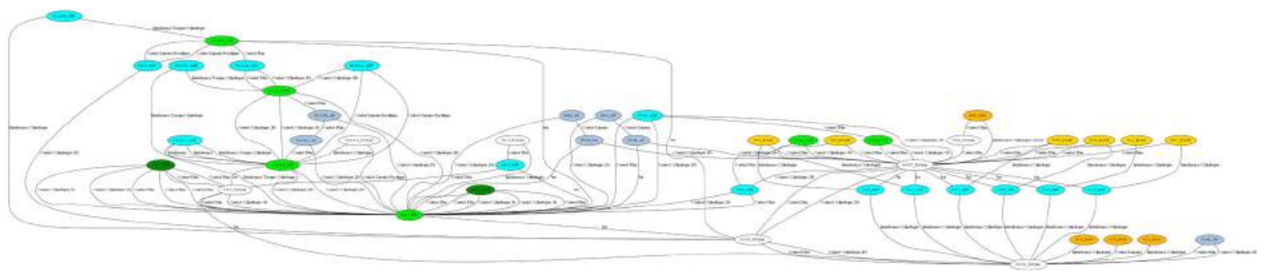

Figure 10. Graph of geometric interfaces between components. Nodes are solids of the assembly and arcs indicate the interfaces between them. The assembly processed is the hydraulic pump (see Figure 1).

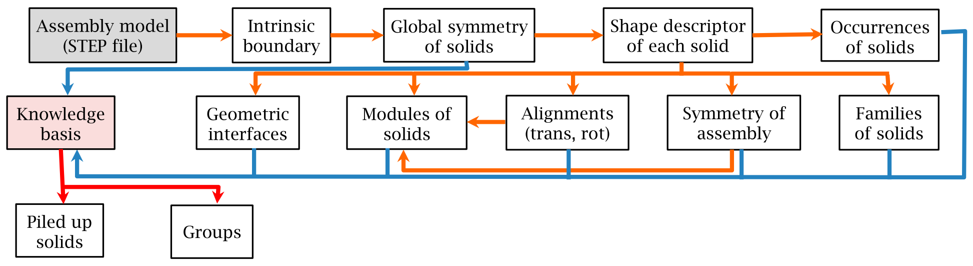

Figure 11. Process flow generating an intrinsic knowledge-based assembly. Orange arrows indicate geometry processing dependencies, blue ones express the extraction of symbolic information from geometry, red ones characterize dependencies of symbolic information.

Ontology-based approaches appear less sensitive to such evolutions. Lastly, it appears that large CAD assemblies are not always easy to process interactively, e.g., selecting interactively hundreds or thousands of components is fairly tedious and error prone, rather than querying the assembly model using the knowledge base.

Here, the model of symbolic information is ontology-based. Consequently, some geometric entities can be mapped to concepts as part of it. The most important feature is the automated mapping between the geometry processing (Section 5.1) and the entities, i.e., the concepts, of the ontology-based model of the symbolic environment. Indeed, it is a key aspect because the geometry processing operators or, more generally, the closed set $\mathcal{O}$ (see Section 3) forms an automation producing consistent geometric entities. This consistency must be extended to the symbolic environment and the automated mapping is part of this extension. This means populating the knowledge base and generating new symbolic information must be achieved automatically to obtain a consistent knowledge-based assembly model. This justifies the use of an inference engine to automate the generation of new symbolic information.

For the sake of conciseness the content of the ontology is not described. The focus is placed on the inference mechanism.

Firstly, concepts are identified that do not require numerical treatments though they enable geometric reasoning to extend the knowledge assembly model through the generation of new symbolic information. In a first place, let us consider the concept of occurrence. A symbolic extension that relates to geometry is the concept of component that is related to geometry with its corresponding solid. Then, the geometry processing of solids, i.e., components, using its geometric descriptor enables the extraction of occurrences. A component C_i is mapped to '*ComponentOfIndustrialProduct(C_i)*' whose solid model, M_i, is mapped to '*Solid(M_i)*' and occurrences are expressed with the property '*AreGeometricOccurrences*' and related to components with '*AreGeometricOccurrences(C_i, C_j)*'.

Similarly, all the geometric informations extracted and described at section 5.1 have their symbolic counterpart generated in the knowledge base (see Figure 11). The knowledge base is automatically fed with facts belonging to the intrinsic assembly model. As examples, the knowledge base contains:

- The global symmetry properties of each solid model M_i of component C_i, e.g., '*IsAxiSymetric(M_i)*';
- A geometric interface I between components (C_i, C_j) is a property of each component based on its solid model M_i with, e.g., '*HasGeometricInterface(M_i, I)*'. The geometric assembly model contains the geometric interface graph and, somehow, the property '*HasGeometricInterface*' describes the same graph topology in the knowledge base. This is complementary since the geometric interface graph is processed for quantitative and structural purposes, i.e., subdivision of the graph into cycles or dangling connections, . . . , whereas symbolic information is essentially bound to adjacency relationships, i.e., '*HasGeometricInterface*';
- A taxonomy of geometric interfaces in each component [34]. This is a property of geometric interfaces, I, e.g., '*PlanarContact(I)*'. This property is mapped to the lowest level of function that is being set up in the knowledge based assembly model [34], e.g., '*PlanarSupport(F)*' that can be generated using the following rule: $\forall M_i, I$. *Solid(M_i) PlanarContact(I) HasGeometricInterface(M_i, I)* $\rightarrow \exists F$ *HasFunction(M_i, F) PlanarSupport(F)*, where F is an instance of function.

From this knowledge base, a set of inference rules has been developed under first order logic form:

$$\forall \hat{X}, \hat{Y}(Hypothesis(\hat{X}, \hat{Y}) \rightarrow \exists\, \hat{Z}.Conclusion(\hat{X}, \hat{Z}))$$

Here, *Hypothesis* and *Conclusion* are sets of atoms defined over sets of variables $\hat{X}, \hat{Y}, \hat{Z}$. The formal semantics of the existential rules is based on the concept of homomorphism [3].

Other inference rules can be illustrated to produce new facts and derive higher-level structural information. These inferences enable the extraction of '*Groups*' and '*Piled up*' components. These two concepts are interesting because they are geometric subsets of functional properties. A *Group* is a set of solids S_M, $\{M_i, M_j, \cdots, M_k\} \subset S_M$ representing components $\{C_i, C_j, \cdots, C_k\}$, occurrence of each other, that share a same type of interface with a single component T. The corresponding rule writes:

$$\forall M_i, M_j, T, I_p, I_q.Solid(M_i)Solid(M_j)Solid(T)Interface(I_p)$$
$$Interface(I_q)AreGeometricOccurrences(M_i, M_j)$$

$$HasGeometricInterface(M_i, I_p)$$
$$HasGeometricInterface(M_j, I_q)$$
$$HasGeometricInterface(T, I_p)$$
$$HasGeometricInterface(T, I_q) \rightarrow Group(M_i, M_j)$$

As such, this rule finds groups of cardinality two. To extract generic configurations, it is applied in cooperation with a transitivity rule enabling to produce groups of arbitrary cardinality. Typically, groups appear in the case of nuts and studs assembling a housing (see Figure 12b). This is interesting for assembly / disassembly simulation [32] to generate parallel sequences. In Rejneri et al. [32], they were interactively defined but can be automatically identified now. These groups convey functional information since the components involved share a geometric interface and characterize the repetition of this information. This is important to speed up the inference of functional designations of components [7, 34] because groups express that identical components sharing identical geometric interfaces bear the same functional meaning.

A set of '*PiledUp*' components $\{C_i, C_j, \cdots, C_k\}$ represented by their solids $\{M_i, M_j, \cdots, M_k\} \subset S_M$ is a set of identical components sharing geometric interfaces of same type (see Figure 12b). The inference process is based on a first rule involving any two of its components stated as:

$$\forall M_i, M_j, I_p.Solid(M_i)Solid(M_j)Interface(I_p)$$
$$AreGeometricOccurrences(M_i, M_j)$$

$$HasGeometricInterface(M_i, I_p)$$
$$HasGeometricInterface(M_j, I_p)$$
$$\rightarrow ElementaryPiledUp(M_i, M_j)$$

Then, to extend the cardinality of this set to an arbitrary number of components, two complementary rules are set up based on the following properties: (i) if two solids form an '*ElementaryPiledUp*' set, then they belong to the same '*PiledUp*' set, (ii) the previous property is transitive. These properties write:

$$\forall M_i, M_j.ElementaryPiledUp(M_i, M_j)$$
$$\rightarrow PiledUp(M_i, M_j)$$

$$\forall M_i, M_j, M_k.PiledUp(M_i, M_j)PiledUp(M_j, M_k)$$
$$\rightarrow PiledUp(M_i, M_k)$$

This property occurs is the case of Belleville washers (see Figure 12b). Piled up components express functional information related to the elastic behavior of the Belleville washers because these washers are elementary spring elements with rather high stiffness compared to common helical springs. Here, the discovery of the '*PiledUp*' set is a hint toward the discovery of the spring function and shows how the number of piled up Belleville washers linearly modulates the stiffness of the '*PiledUp*' set.

Indeed, *Groups* and *PiledUp* sets are examples of assembly structures that can contribute to the functional definition of sub-assemblies and become part of the intrinsic knowledge-based assembly model. *Groups* and *PiledUp* sets illustrate also how new symbolic information can be derived solely from the assembly model input. Additionally, it is important to note that every time an inference is performed, a saturation of the knowledge base takes place to ensure the consistency of the facts derived from this inference. This ensures the consistency of the intrinsic knowledge-based assembly model with:

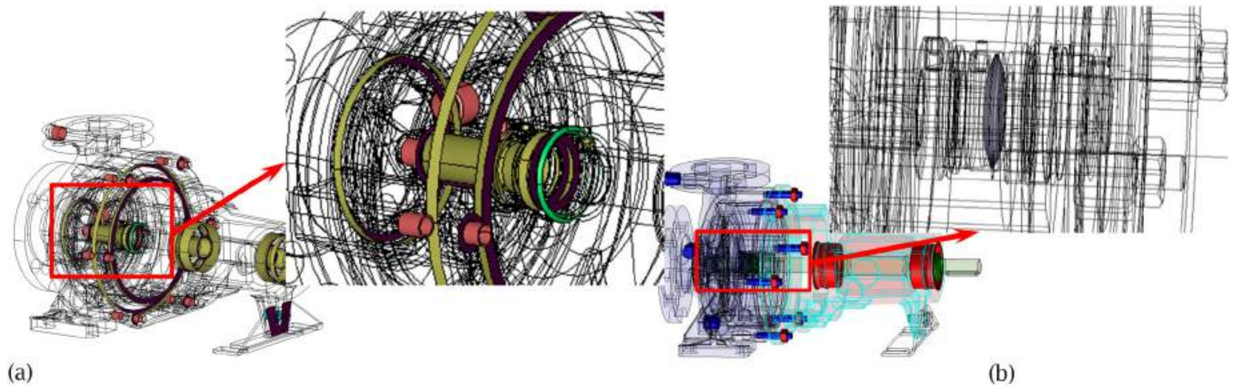

Figure 12. (a) geometric properties from interfaces between components. A subset of the color code indicates: cylindrical interferences (pink), planar contacts (violet), cylindrical contact (beige). (b) Examples of higher-level structures derived from inference rules: *Groups* and *PiledUp* components. Groups of components: The colors of components indicate different sets of Groups. Inside the same color, different components shapes indicate different Groups. The common component involved in each Group is represented with transparency. The pale violet one relates to the blue components. The pink one relates to the red components. PiledUp components are two Belleville washers (in gray).

- The consistency of the intrinsic assembly geometric model described in Sections 3 and 5.1;
- The consistency of the taxonomies derived from the geometric entities;
- The consistency of the knowledge base that is partly achieved with saturation processes during inferences and existential rules acting as constraints not described here for sake of conciseness;
- The automation of all the geometry and symbolic information processing.

Currently, the intrinsic knowledge-based assembly model contains an ontology described with 103 concepts, 41 relations, 10 inference rules, and 21 constraints.

6. Results and discussion

The proposed approach has been applied to various assemblies. The assemblies selected are:

A1. A hydraulic pump that has been used to illustrate most of the geometry and knowledge processing described in Section 5 (see Figures 1, 8, 9, 10, 13);
A2. An electric stapler (see Figure 13a);
A3. A subset of wing – aircraft body junction (see Figure 8c, d);
A4. A tooling equipment used to support an aircraft wing (see Figure 13b).

They are ordered by increasing number of elementary components. A2 and A3 contain approximately the same number of components but differ in terms structure, as depicted in Table 2.

Table 1 gives a breakdown of timings (geometry processing and symbolic information processing). The tests have been performed on a PC with Intel 8 cores 64 bits processors @3.4Ghz, 16 Gb RAM.

The analysis of the geometry processing results shows that the algorithmic complexity can have a significant influence over the processing time. This is particularly illustrated when computing the reflective symmetry planes of very large assemblies (Assembly A4). It should be pointed out that the Module computation times reported have been obtained using an algorithm with low combinatorial complexity. An algorithm with an exhaustive list of Modules incorporates a much larger combinatorial complexity. As an example, the time reported for A3 evolves from 0.21s in Table 1 to 18s when referring to the largest combinatorial complexity. A2 and A3, though having nearly the same amount of solids, exhibit significant differences of processing time for some treatments due to the differences of:

- Shapes of the components;
- Number of solids contributing to groups of occurrences, i.e., structural differences.

Anyhow, these treatments stay efficient compared to the computation of the geometric interfaces because they use the intrinsic reference frame of each solid. The corresponding treatments become independent of the complexity of each component. Computing geometric interfaces requires further investigations in Open Cascade

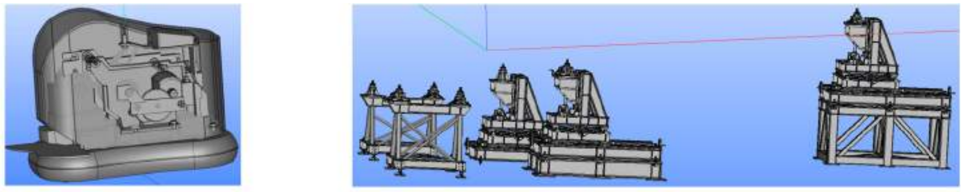
(a) electric stapler (b) tooling equipment (courtesy Airbus IW)

Figure 13. Test examples of assemblies.

Table 1. Assembly geometry processing. Meanings of the columns: 'Decomp.' (decomposition of the assembly into solids), 'Treat. Sol.' (treatments of a solid up to the computation of its geometry descriptor), 'Occur.' (computation of the occurrences), 'Ass. Sym.' (reflective symmetry planes of the assembly), 'Repeat.' (computation of linear & rotational repetitions), 'Modules' (computation of the modules), 'Total' (time for all the previous treatments), 'Interfaces' (computation time of all the geometric interfaces). *indicates that the computation of geometric interfaces is partial.

Ass. ID	Decomp.	Treat. Sol.	Occur.	Ass. Sym.	Repeat.	Modules	Total	Interfaces
A1	$3\ 10^{-3}$ s	10.7 s	0.23 s	1.06 s	0.9 s	$4\ 10^{-3}$ s	12.94 s	14mn 10s
A2	$3\ 10^{-3}$ s	40.23 s	1.14 s	12.31 s	8.21 s	0.019 s	61.94 s	7mn 16s
A3		14.25 s	1.39 s	22.75 s	2.71 s	0.21 s	41.33 s	2mn 7s
A4	$15\ 10^{-3}$ s	11mn 17s	4mn 12s	100 mn	4mn 24s	12mn 41s	137mn	213mn*

Table 2. Assembly structure parameters. Columns meaning: number of solids in the assembly, number of groups of identical solids and total amount of solids in these groups (for #Occur and #Family), number of repetitions (linear and rotational) and number of solids they involve, number of modules and number of solids into these modules, number of 'PiledUp' configs, and number of groups and number of solids they involve (possibly with duplicates).

Ass. ID	#solids	#Occur.	Family	#Repeat.	Modules	PiledUp	Groups
A1	47	9gr, #27	11gr, #33	1, #12	3gr, #20	1	8, #24
A2	124	9gr,#70	12gr,#80	8, #54	2gr, #54	2	41, #141
A3	148	14gr, #141	8gr, #141	324, #61	13gr, #44	1	9, #202
A4	1793	214gr, #1786	334gr, #1791	165, #1068	253gr, #1222	N/A	N/A

Table 3. Assembly knowledge processing performances. The columns designate: number of facts (triplets) inserted in the knowledge base, time to insert them, time for the knowledge base saturation.

Ass. ID	#triplets	Insertion	Saturation
A1	2593	3 s	5.5s
A2	12947	19 s	20mn 4s
A3	20148	45 s	21mn 56.5s
A4	153469	8 mn	N/A

library to characterize the robustness of its operators because the timing seems sensitive to some component configurations and, in the case of A4, didn't produce all the geometric interfaces. Altogether, Table 2 shows the results characterizing the structure of each assembly and highlights the importance of the concepts described in Section 5.

In the scope of knowledge processing, it has to be pointed out that assembly A4 exhibits partial results because the saturation could not be performed due to technical limitations of the inference engines. Consequently, the timing for saturation is not available, as well as the PiledUp sets of solids and the Groups. Also, the knowledge processing time is significant compared to the geometry processing one, though it is performed only once to ensure the consistency of the knowledge and geometric assembly model. The size of the queries appears as a current issue too due to its connection with the query execution plan as set up by JENA. Typically query execution time can range from tenth of seconds to several seconds depending on the complexity of the query. Another important issue holds in the capacity of the inference engines to handle large datasets, as pointed out by assembly A4. These limits are currently analyzed and processed.

The knowledge base and queries characterizing the treatments described at Section 5 are made publicly available from the website (IE and Firefox navigators): http://3dassblyanlysis.gforge.inria.fr/3d/ for the hydraulic pump where the 3D viewer uses a facetted representation of the assembly.

The results obtained validate the proposed approach and the concept of intrinsic knowledge-based assembly model. Because this concept is fairly generic, it can find applications into a wide scope from finite element model preparation as illustrated in Boussuge et al. [7], capitalization and model retrieval as illustrated in [11, 21, 22, 45] to contribute to a descriptor of assemblies and bring some functional meaning, and assembly / disassembly simulation where the geometric interfaces can bring automatically more precise information about extraction directions [16, 30], help defining assembly sequences [15, 16, 32, 44]. Future work involves the extension of geometric operators to improve their efficiency, especially for modules and develop new ones to handle a larger range of interfaces between components. From the symbolic information point of view, the current limitations encountered regarding the queries and the size of the knowledge base during inferences are addressed to achieve the scalability of the proposed approach. The extension of the reasoning mechanisms is ongoing with the identification of bearings, O-ring seals, fasteners, ... , as functional designations.

7. Conclusion

The proposed approach structures CAD assembly models based on intrinsic information. The content of the assembly structure is available as 3D entities in the CAD modeler SALOME and it is tightly connected to the knowledge base JENA where it can be queried through the assembly ontology. The initial assembly is now transformed into a first level of knowledge-based assembly where low-level functional information is made available for a wide range of applications.

The concept of intrinsic knowledge-based assembly model has proved its efficiency through assembly structural information that would hardly be available otherwise; the availability of a geometric descriptor of CAD components, and the knowledge processing that can be regarded as a first level of approach to spatial reasoning in a symbolic environment.

The overall consistency of the model has been highlighted though the formalism of the mixed model is still to be developed.

Acknowledgments

The authors would like to thank the EEC through the ERC-EXPRESSIVE grant that partially contributed to support this research. Also, the authors thank EDF Lab that partially contributed to support this research.

ORCID

Harold Vilmart http://orcid.org/0000-0002-7849-8211
Jean-Claude Léon http://orcid.org/0000-0003-3337-081X
Federico Ulliana http://orcid.org/0000-0002-9192-9573

References

[1] Attene, M.; Robbiano, F.; Spagnuolo, M.; Falcidieno, B: Characterization of 3D shape parts for semantic annotation, *Computer-Aided Design*, 41, 2009, 756–763. https://doi.org/10.1016/j.cad.2009.01.003

[2] Baget, J.-F.; Leclère, M.; Mugnier, M.-L.; Rocher, S.; Sipieter, C.: Graal: A Toolkit for Query Answering with Existential Rules, Proc. 9th Int. Web Rule Symp. (RuleML), Berlin, August 2015, 328–344. https://doi.org/10.1007/978-3-319-21542-6_21

[3] Baget, J.; Leclère, M.; Mugnier, M.; Salvat, E.: On rules with existential variables: Walking the decidability line, *Artificial Intelligence,* 175(9–10), 2011, 1620–1654. https://doi.org/10.1016/j.artint.2011.03.002

[4] Barbau, R.; Krima, S.; Rachuri, S.; Narayanan, A.; Fiorentini, X.; Foufou, S.; Sriram, R.D.: Ontostep: enriching product model data using ontologies, *Computer-Aided Design*, 44 (6), 2012, 575–590. https://doi.org/10.1016/j.cad.2012.01.008

[5] Bernal, Z. F. A.; Rojo, V. A.; Tornero, M. J.: Application of NX Knowledge Fusion module for the Design Automation of an Automotive Painting Defects Inspection Tunnel, *Computer-Aided Design & Applications*, 9(5), 2012, 655–664. https://doi.org/10.3722/cadaps.2012.655-664

[6] Boussuge, F.; Léon, J.-C.; Hahmann, S.; Fine, L.: Extraction of generative processes from B-Rep shapes and application to idealization transformations, *Computer-Aided Design*, 46, 2014, 79–89. https://doi.org/10.1016/j.cad.2013.08.020

[7] Boussuge, F.; Shahwan, A.; Léon, J.-C.; Hahmann, S.; Foucault, G.; Fine, L.: Template-based Geometric Transformations of a Functionally Enriched DMU into FE Assembly Models, *Computer-Aided Design & Applications*, 11(4), 2014, 436–44. https://doi.org/10.1080/16864360.2014.881187

[8] Calì, A.; Gottlob, G.; Pieris, A.: Towards more expressive ontology languages: The query answering problem, *Artificial Intelligence*, 193, 2012, 87–128. https://doi.org/10.1016/j.artint.2012.08.002

[9] Calvanese, D.; Giacomo, G. D.; Lembo, D.; Lenzerini, M.; Rosati, R.: Tractable Reasoning and Efficient Query Answering in Description Logics: The DL- Lite Family, *J. of Autom. Reasoning*, 39(3), 2007, 385–429. https://doi.org/10.1007/s10817-007-9078-x

[10] Chandrasegaran, S. K.; Ramani, K.; Sriram, R. D.; Horváth, I.; Bernard, A.; Harikf, R. F.; Gao, W.: The evolution, challenges, and future of knowledge representation in product design systems, *Computer-Aided Design*, 45, 2013. https://doi.org/10.1016/j.cad.2012.08.006

[11] Chen, X.; Gao, S.; Guo, S.; Bai, J.: A flexible assembly retrieval approach for model reuse, *Computer-Aided Design*, 44(10), 2012, 1033–48. https://doi.org/10.1016/j.cad.2010.12.008

[12] Eiter, T.; Ortiz, M.; Simkus, M.; Tran, T.; Xiao, G.: Query Rewriting for Horn-SHIQ Plus Rules, Proc. 26th AAAI Conf., Toronto, July 22–26, 2012, 726–733.

[13] El-Mehalawia, M.; Miller, R. A.: A database system of mechanical components based on geometric and topological similarity. Part II: indexing, *retrieval, matching, and similarity assessment, Computer-Aided Design*, 35, 2003, 95–105. https://doi.org/10.1016/S0010-4485(01)00178-6

[14] Haarslev, V.; Hidde, K.; Möller, R.; Wessel, M.: The RacerPro knowledge representation and reasoning system, *Semantic Web*, 3(3), 2012, 267–277.

[15] Hsu, Y.-Y.; Tai, P.-H.; Wang, M.-W.; Chen, W.-C.: A knowledge-based engineering system for assembly sequence planning, *Int. Journal of Advanced Manufacturing Technology*, 55(5), 2011, 763–782. https://doi.org/10.1007/s00170-010-3093-5

[16] Iacob, R.; Mitrouchev, P.; Léon, J.-C.: Assembly simulation incorporating component mobility modeling based on functional surfaces, *Int. J. Interact Des. Manuf.*, 5(2), 2011, 119–132. https://doi.org/10.1007/s12008-011-0120-1

[17] ISO 10303: Industrial automation systems and integration – Product data representation and exchange, Standard, International Organization for Standardization, Geneva, Switzerland, 2003.

[18] Iyer, N.; Jayanti, S.; Lou, K.; Kalyanaraman, Y.; Ramani, K.: Three-dimensional shape searching: state-of-the-art review and future trends, *Computer-Aided Design*, 37, 2005, 509–530. https://doi.org/10.1016/j.cad.2004.07.002

[19] LaRocca, G.: Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design, *Advanced Engineering Informatics*, 26, 2012, 159–179. https://doi.org/10.1016/j.aei.2012.02.002

[20] Li, K.; Foucault, G.; Léon, J.-C.; Trlin, M.: Fast global and partial reflective symmetry analyses using boundary surfaces of mechanical components, *Computer-Aided Design*, 53, 2014, 70–89. https://doi.org/10.1016/j.cad.2014.03.005

[21] Lupinetti, K.; Giannini, F.; Monti, M.; Pernot, J.-P.: CAD assembly descriptors for knowledge capitalization and model retrieval, *Proc. Int. Conf, Tools and Methods for Competitive Engineering*, 2016, 587–598.

[22] Lupinetti, K.; Chiang, L.; Giannini, F.; Monti, M.; Pernot, J.-P.: Regular patterns of repeated elements in CAD assembly model retrieval, *Computer-Aided Design & Applications*, 14, 2017, 1–12. https://doi.org/10.1080/16864360.2016.1257193

[23] Ma, L.; Huang, Z.D.; Wang, Y.W.: Automatic discovery of common design structures in CAD models, *Computers & Graphics*, 34, 2010, 545–55. https://doi.org/10.1016/j.cag.2010.06.002

[24] Ma, Y.-S.; Britton, G. A.; Tor, S. B.; Jin, L. Y.: Associative assembly design features: concept, *implementation and application, Int. J Adv. Manufacturing Technology*, 32, 2007, 434–444. https://doi.org/10.1007/s00170-005-0371-8

[25] Mäntyla, M.: *Introduction to Solid Modeling*, W. H. Freeman & Co. New York, NY, USA, 1988.

[26] Mills, B.; Langbein, F.; Marshall, A.; Martin, R.: Estimate of frequencies of geometric regularities for use in reverse engineering of simple mechanical components, Tech. Report GVG 2001–1, Dept. Computer Science, Cardiff University, 2001.

[27] Moitra, A.; Palla, R.; Rangarajan, A.: Automated Capture and Execution of Manufacturability Rules Using Inductive Logic Programming, *Proc. of the Twenty-Eighth AAAI Conference on Innovative Applications (IAAI-16)*, 2016, 4028–4034.

[28] Nolan, D. C.; Tierney, C. M.; Armstrong, C. G.; Robinson, T. T.: Defining Simulation Intent, *Computer-Aided Design*, 59, 2015, 50–63. https://doi.org/10.1016/j.cad.2014.08.030

[29] Palombi, O.; Ulliana, F.; Favier, V.; Léon, J.-C.; Rousset, M.-C.: My Corporis Fabrica: an ontology-based tool for reasoning and querying on complex anatomical models, *Journal of Biomedical Semantics, BioMed Central*, 5(20), 2014, 1–13.

[30] Popescu, D.; Iacob, R.: Disassembly method based on connection interface and mobility operator concepts, *Int. Journal of Advanced Manufacturing Technology*, 69(5), 2013, 1511–1525. https://doi.org/10.1007/s00170-013-5092-9

[31] Protégé: A free, open-source ontology editor and framework for building intelligent systems, http://protege.stanford.edu/, (last accessed: 03/24/2017).

[32] Rejneri, N.; Léon, J.-C.; Débarbouillé, G.: Early assembly/disassembly simulation during a design process, Proc. Conf. Int. ASME, Design for Manufacturing Conference, Pittsburg, 9–12 September, 2001.

[33] SALOME: The Open source Integration platform for Numerical Simulation, http://www.salome-platform.org, (last accessed 03/20/2017).

[34] Shahwan, A.; Léon, J.C.; Foucault, G.; Trlin, M.; Palombi, O.: Qualitative behavioral reasoning from components' interfaces to components' functions for DMU adaption to FE analyses, *Computer-Aided Design*, 45 (2), 2013, 383–394. https://doi.org/10.1016/j.cad.2012.10.021

[35] Sirin, E.; Parsia, B.; Cuenca Grau, B.; Kalyanpur, A.; Katz, Y.: Pellet: A practical OWL-DL reasoner, *J. of Web Semantics*, 5(2), 2007, 51–53. https://doi.org/10.1016/j.websem.2007.03.004

[36] Sonthi, R.; Kunjur, G.; Gadh, R.: Shape feature determination using the curvature region representation, Proc. of the Fourth Symposium on Solid Modeling and Applications (SMA '97), Atlanta, GA, USA, May 14–16, 1997, 285–296.

[37] Swain, A.K.; Sen, D.; Gurumoorthy, B.: Extended liaison as an interface between product and process model in assembly, *Robotics and Computer-Integrated Manufacturing*, 30 (5), 2014, 527–545. https://doi.org/10.1016/j.rcim.2014.02.005

[38] Tangelder, J. W.; Veltkamp, R. C.: A survey of content based 3D shape retrieval methods, *Multi- media tools and applications*, 39(3), 2008, 441–471. https://doi.org/10.1007/s11042-007-0181-0

[39] Tao, S.; Huang, Z.; Ma, L., Guo, S.; Wang, S.; Xie, Y.: Partial retrieval of CAD models based on local surface region decomposition, *Computer-Aided Design*, 45, 2013, 1239–1252. https://doi.org/10.1016/j.cad.2013.05.008

[40] Tomiyama, T.; Van Beek, T. J.; Alvarez Cabrera, A. A.; Komoto, H.; D'Amelio, V.: Making function modeling practically usable, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 27, 2013, 301–309. https://doi.org/10.1017/S0890060413000309

[41] TraceParts: library of standard components delivering 3D CAD models, http://www.traceparts.com/fr/, last accessed: 03/09/2017.

[42] Ulliana, F.; Léon, J.-C.; Palombi, O.; Rousset, M.-C.; Faure, F.: Combining 3D Models and Functions through Ontologies to Describe Man-made Products and Virtual Humans: Toward a Common Framework, *Computer-Aided Design & Applications*, 12(2), 2014, 166–18. https://doi.org/10.1080/16864360.2014.962429

[43] van der Meiden, H. A.; Bronsvoort, W. F.: Solving topological constraints for declarative families of objects, *Computer-Aided Design*, (8), 2007, 652–662. https://doi.org/10.1016/j.cad.2007.05.013

[44] Vigano, R.; Osorio Gomez, G.: Assembly planning with automated retrieval of assembly sequences from CAD model information, *Assembly Automation*, 32(4), 2012, 347–360. https://doi.org/10.1108/01445151211262410

[45] Zhang, J.; Xu, Z.; Li, Y.; Jiang, S.; Wei, N.: Generic face adjacency graph for automatic common design structure discovery in assembly models, *Computer-Aided Design*, 45(8–9), 2013, 1138–1151. https://doi.org/10.1016/j.cad.2013.04.003