



Science Arts & Métiers (SAM)

is an open access repository that collects the work of Arts et Métiers Institute of Technology researchers and makes it freely available over the web where possible.

This is an author-deposited version published in: <https://sam.ensam.eu>
Handle ID: <http://hdl.handle.net/10985/16981>

To cite this version :

Marc DANIEL, Dorian DÉCRITEAU, Jean-Philippe PERNOT - Towards a declarative modeling approach built on top of a CAD modeler - Computer-Aided Design and Applications - Vol. 13, n°6, p.737-746 - 2016

Any correspondence concerning this service should be sent to the repository

Administrator : scienceouverte@ensam.eu



Towards a declarative modeling approach built on top of a CAD modeler

Dorian Decriteau^a , Jean-Philippe Pernot^a  and Marc Daniel^b 

^aArts et Métiers ParisTech, LSIS Laboratory UMR CNRS 7296; ^bAix-Marseille Université, LSIS Laboratory UMR CNRS 7296

ABSTRACT

Today's CAD modelers are very efficient in processing 3D shapes of CAD models by means of B-Rep modeling operators such as pad, pocket, shaft, groove, hole, fillet and so on. At a lower description level, those modeling operators are based on Euler operators acting directly on the faces, edges and vertices of the B-Rep models. Using such a top-down approach, the designers do not have to work on low-level geometric entities, but rather manipulate so-called structural and detail features to shape directly the CAD models. However, there is still a gap between the shapes the designers have in mind and the way they have to decompose them in a succession of modeling steps. This paper proposes a new declarative modeling approach to design industrial shapes allowing the designers to interact with a CAD software at a more conceptual level. The designers enter a high-level description of the expected shapes that is then transformed through scripts into traditional CAD operators successively called to create the shapes. Compared to the traditional feature-based approaches, our declarative modeling approach is closer to the way designers think. It saves time while keeping all the advantages of existing efficient CAD modelers. This new approach aims at quickly creating drafts rather than final shapes. Those drafts can then be modified using classical CAD software in which our new approach is fully embedded. This approach is a first step towards a declarative CAD modeler.

KEYWORDS

Declarative modeling; semantic description; encapsulating CAD operators; shape design

1. Introduction

Today, industrial CAD software rely on an incremental B-Rep (Boundary Representation) modeling paradigm where volume modeling is performed iteratively using planar sketched contours subjected to mainly extrusion or revolution operations, as generative parameters, and to either material addition or removal, as shape sculpting parameters. The construction tree structure is based on reference planes containing the sketched contours and primitive shapes defined from the generative and shape sculpting parameters [3]. At a lower description level, those modeling operators are based on Euler operators acting directly on the faces, edges and vertices of the B-Rep models. In this way, the designers do not manipulate low-level geometric entities, but rather manipulate so-called structural and detail features to shape directly the CAD models [13].

However, even if CAD modelers provide operators (e.g. pad, pocket, shaft, groove, hole, fillet) to get rid of the direct use and manipulation of canonical surfaces and NURBS [10], working with a CAD modeler is almost procedural with a lot of intermediate operations required to obtain the desired shape of an object. Actually, all those

intermediate operations are time-consuming and generate complex construction trees that are not particularly needed to describe the final shape. Moreover, using such a procedural approach, the designers have to make a mental gymnastic to break down the object body into several basic shapes linked to the different operators of the CAD software. Thus, to model a complex shape, a lot of operations have to be done, even if a featuring approach is used. This is even truer when dealing with free form objects for which the notion of free form features is much less adopted in current industrial practices [9].

Clearly, an approach closer to the designers' way of thinking is missing and there is still a gap between the ideas designers have in mind and the available tools and operators used to model them. Ideally, it would be more convenient to enter a semantic description of the shape, the CAD modeler being in charge of generating it.

This is the aim of the approach proposed in this paper. More precisely, in order to stay compatible with existing CAD modelers widely used in the industry, but also to take full advantage of their efficient geometric modeling kernels and features, our attempt was to define a high-level declarative modeling approach implemented

in the form of a plugin built on top of these modelers. From an initial high-level description, the plugin generates the CAD model and its building tree, which can be immediately integrated and classically manipulated within the CAD modelers. The main idea is to encapsulate several operators and/or features to answer to a partial description of the shape. Since it is built on top of actual CAD modelers, such a declarative modeling approach can be integrated within the Product Development Process (PDP) and the traditional way of manipulating CAD models obviously remains accessible to more experimented users/designers. This top-down approach is illustrated on Fig. 1. At the top level, within our plugin, designers manipulate a semantic description that is transformed into a procedural description, i.e. a sequence of traditional CAD functions and operators (also called features) which use the traditional Euler operators to act on the low-level geometric entities (faces, edges, vertices) defining the underlying B-Rep model.

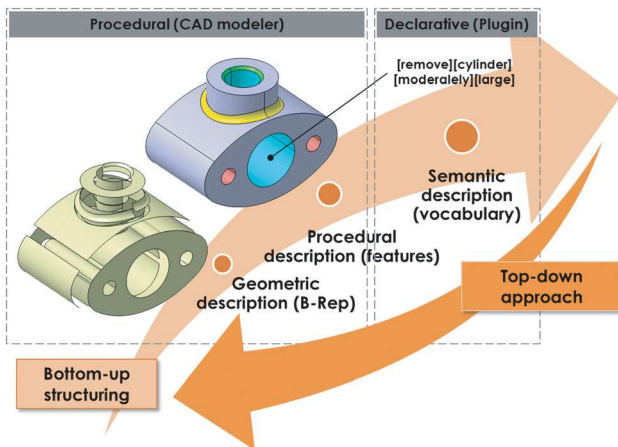


Figure 1. Declarative modeling approach built on top of a procedural CAD modeler.

Finally, it is clear that the output of this declarative modeling approach is not a final CAD model instantiated with accurate numerical values, but rather the output is to be considered as a first draft quickly obtained. As a matter of fact, the description must not be tedious and can remain incomplete so as to leave the possibility to refine the description during the next steps of the PDP [3]. Those forthcoming manipulations are made possible thanks to the fact that the output of this generative process is a B-Rep model defined by a traditional building tree with features and parameters on which the designers can still come back to make the final CAD model adapted to industrial constraints. Of course, handling incompletely defined shapes may generate not expected but valuable solutions. It is also a good mean to take

into account the uncertainties the designers have when defining complex shapes.

The paper presents the results provided by the first version of our plugin. It is organized as follows. Section 2 discusses the related work and section 3 introduces the proposed framework. The different modeling functions are described in section 4 and illustrated in section 5. Section 6 concludes this paper.

2. Related work

As suggested on Fig. 1, at a low level, CAD modelers generally consider purely geometric description using curves and surfaces represented by means of B-splines and NURBS. Invented more than 40 years ago, these mathematical models are well known and their fundamental concepts can be found in several reference books [4],[10]. Since the expected shapes are generally complex, the designer often has to decompose them into elementary shapes themselves subdivided into several surfaces. Each elementary surface is defined by means of a network of control points, weights and knot sequences. In addition, these surfaces must most often be trimmed to overcome the topological constraints of the mathematical models [4]. Finally, the elementary surfaces are assembled together to produce a manifold solid, i.e. a B-Rep representation expressing the relationships between the vertices, the edges and the faces of the topological model as described in [5]. However, interacting at this very low level is restricted to experts and generally at a final step of the modeling process or to address specific aesthetic issues. Thus, several attempts have been made to try to overcome the limits inherent to the manipulation of low-level geometric entities.

Feature-based modeling introduced in [14] falls into this category of higher-level approaches. By using features to build their CAD models, designers do not anymore act at a low level but rather on shape primitives that can be parameterized and pre-defined. Actually, features are used to give a meaning and to manipulate directly a set of geometric entities. Depending on the complexity of the shapes to be represented, different approaches and associated features can be used: form features, semi free form features, free form features or fully free form features [9]. If form features are now well known and implemented in most of the existing CAD systems, it is not true for free form features which are not yet fully available in commercial solutions. Sometimes, features can be directly associated to manufacturing information so that these information can be retrieved in downstream applications [11]. In this way, an overall CAD system can be fully automated, however, the idea of using manufacturing features to design a part has its own drawback: the

features used to design the part do not necessarily represent the best way to manufacture it. Therefore, it is the designer's responsibility to evaluate all methods that can produce the targeted shape. Furthermore, manufacturing features are not the most natural way of designing a part. In other words, there is no uniqueness in the way a 3D shape can be described and modeled within CAD software. Thus, even if features are a real improvement in the way designers interact with the CAD models, there is still a gap between the available tools and functionalities, and the way designers think in 3D.

Declarative modeling appeared in the early nineties' and aims at constructing objects by means of a set of properties rather than by entering geometrical information like point coordinates. It is thus mainly based on semantics and not on mathematical properties so that this approach is closer to the way designers think. In their work, Desmontils and Lucas have proposed a synthesis of their first experiments and have described this modeling process in three steps [2]:

- First, the designer has to make a description of the shape by using an adapted vocabulary. This description must be transformed into a set of constraints that can be solved. The description can be adapted to any given trade assuming that a list of synonymous exists.
- Second, the modeling software models the shape from the description by exploring through adapted and specific algorithms the space of solutions. As already introduced, this implicitly suggests that such an approach is more devoted to explore a set of solutions than to solve well-constrained or over-constrained problems.
- Third, the designer has to browse between the different produced solutions and choose one.

It is evidently easier to study objects belonging to discrete spaces of solutions since these sets can be explored (or described) and the tree of solutions can be pruned with the given properties. Spaces of solutions depending on real values are more difficult to study. A first insight of what can be done in mechanics was proposed in [1]. Then, declarative modeling has been studied on curves [12] and surfaces [7]. Actually, declarative modeling is one form of the generic variational design approach. The latter has been introduced in CAD in [8]. Different attempts have been made to develop this approach. Like for example in [6], the associated work quickly concentrates on the important problem of constraints modeling and how to split the set of constraints into smaller independent sets. However, this approach was ambitious and it has been slowed down by the huge

amount of work mandatory to redevelop all the basic operators and primitives required to generate shapes from descriptions. Effectively, at that time, the proposed approach has been designed starting from scratch without considering the existence of efficient and robust geometric kernels accessible through traditional CAD software.

As a conclusion, in this paper, the idea is to combine the advantages of the traditional B-Rep and feature-based modeling approaches with the advantages of a more advanced and high-level declarative modeling approach. As a consequence, this combination takes astutely advantage of today's robust commercial geometric modeling kernels without redeveloping everything starting from scratch. At the end, the proposed approach can be seen as a plugin of a CAD software that transforms a high-level description into a building tree gathering together all the operations and functions used to get the final 3D shapes. This procedure is more suitable to quickly obtain a draft or a sketch rather than defining a definitive complex shape. Depending on the needs, the output can then be further processed using the traditional CAD functionalities.

3. Declarative modeling framework and semantic description module

To be able to move from a mental image the designer has in mind to a draft CAD model generated in a CAD environment, the proposed framework is composed of three modules to be used as a sketcher in the early designed phases of a PDP (Fig. 2):

- **the semantic description module** in which the designer describes the shapes he/she wants to generate using a dedicated vocabulary and grammar;
- **the generic shape modeling module** which transforms this semantic description into a succession of generic shape modeling functions and operators. This step can be seen as a transformation from a semantic description to a more geometry-oriented description. At this stage, this new description is still generic and do not rely on a specific CAD modeler;
- **the specific CAD modeling module** which transforms this set of generic shape modeling operators and functions into a set of CAD modeling functions and operators. During this last stage, the CAD model is generated using a succession of traditional CAD modeling functions and operators that may differ from a CAD modeler to another. The output is a draft CAD model that can then be used and modified during the next steps of a PDP.

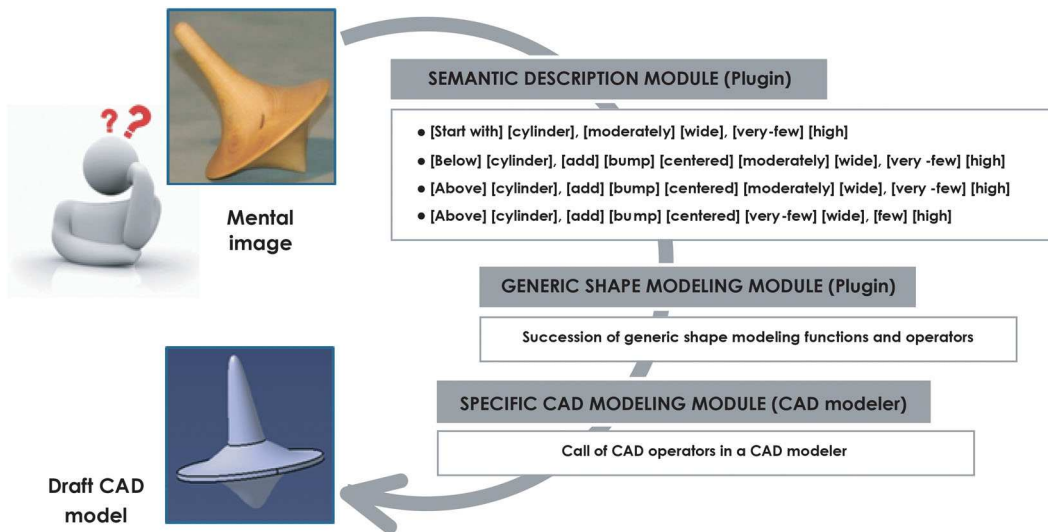


Figure 2. Modular declarative modeling framework as part of a PDP.

The first challenge of this approach is to be able to provide a semantic description of a part. Different tests have been performed to analyze the way existing parts can be described. From a set of predefined descriptions, designers were first invited to model the corresponding parts in a CAD environment. This process proves that, starting from a simple description of a part, a designer can construct it with a CAD software. This study also revealed that describing a part requires some training. It is not necessarily easy but possible. Actually, the goal was not to work on a rich and exhaustive text description but rather to highlight the feasibility of such a top-down approach. Thus, the vocabulary and associated grammar have been voluntarily reduced to a set as simple as possible. This may also explain the difficulties designers had in describing more complex shapes. Assuming a vocabulary and a grammar defined to describe the shapes themselves (see the next section for a complete definition of this vocabulary for the slice, bump and bend operators), the main problem remains the specification of the localization and relative dimensions of those shapes.

Even if at the end the shapes are positioned with respect to an absolute reference frame, always visible during the description phase, the localization of the shapes is performed using a dedicated vocabulary that can either consider an absolute or a relative positioning. The first step of our plugin is to create the master solid. To this aim, the designer has to choose a template into a list composed of parallelepipeds, cylinders, spheres and so on. To position the solids and shapes, the designer can use absolute positioning words like **[above]**, **[below]**, **[to the right]**, **[to the left]**, **[fore]** and **[back]** that should be understood with respect to the absolute reference frame.

Table 1. Correspondences between the positioning vocabulary and the direction of the reference frame.

Positioning vocabulary	Reference frame directions
[above]	Z+
[below]	Z-
[to the right]	Y+
[to the left]	Y-
[fore]	X+
[back]	X-

Table 1 summarizes the correspondences between this positioning vocabulary and the directions of the absolute reference frame. He/she can also use the same wording while further describing it to enable relative positioning. For example, the sentence **[above] [parallelepiped 1]**, **[on the right hand size]** can be used to position a new shape with respect to **[parallelepiped 1]**. To help the user to make his/her description, all the names of the shapes already designed are written on the model displayed in the 3D viewer.

For the dimensioning of the shapes, the user has to complete his/her description with some adjectives called “quantifiers”. The dimensioning of the master solid is to be considered as specific since the quantifiers are necessarily absolute values. The list of relative quantifiers is composed of the following adjectives: **[extremely-few]**, **[very-few]**, **[few]**, **[moderately]**, **[rather]**, **[very]** and **[extremely]**. For example, a parallelepiped can be dimensioned as **[parallelepiped][moderately][wide]**, **[very][long]** and **[very-few][high]**. One must remember that the purpose is to create a draft and not a final fully constrained CAD model. Thus, the proportions between dimensions are more important than the real size values. To do so, the relative dimensioning is performed with respect to an Order-of-Magnitude (OoM) on which

factors are applied depending on the vocabulary that is used. The final size can be obtained by any homothetic transformation if required. Table 2 summarizes these factors that have been tuned so as to have different ones for straight lines and for radii.

Table 2. Correspondences between quantifiers and factors for relative dimensioning.

For the straight line size:	For the radius-like size:
• [Extremely-few] → OoM × 1/10	• [Extremely-few] → OoM × 1/20
• [Very-few] → OoM × 1/5	• [Very-few] → OoM × 1/10
• [Few] → OoM × 1/2	• [Few] → OoM × 1/5
• [Moderately] → OoM × 1	• [Moderately] → OoM × 1/2
• [Rather] → OoM × 2	• [Rather] → OoM × 1
• [Very] → OoM × 5	• [Very] → OoM × 2
• [Extremely] → OoM × 10	• [Extremely] → OoM × 5

4. From a semantic description model to a DRAFT CAD model

Once the semantic description is available from the first module (section 3), the generic shape modeling module transforms it into a set of generic shape modeling functions and operators. This transformation step is quite generic in the sense that **it does not rely on a specific CAD modeler**. This is not anymore true when considering the last module that transforms this generic shape modeling description into a set of specific CAD modeling functions and operators. Those two transformations are explained in this section and examples are given in section 5.

In this paper, three main operators are introduced as a basis to define shapes: the slice, the bump and the bending operators. They all use the localization operator to position the resulting shape.

4.1. From a semantic description to a generic geometric description (module 2)

4.1.1. Localization operator

The localization operator aims at positioning a reference plane and a reference point with respect to

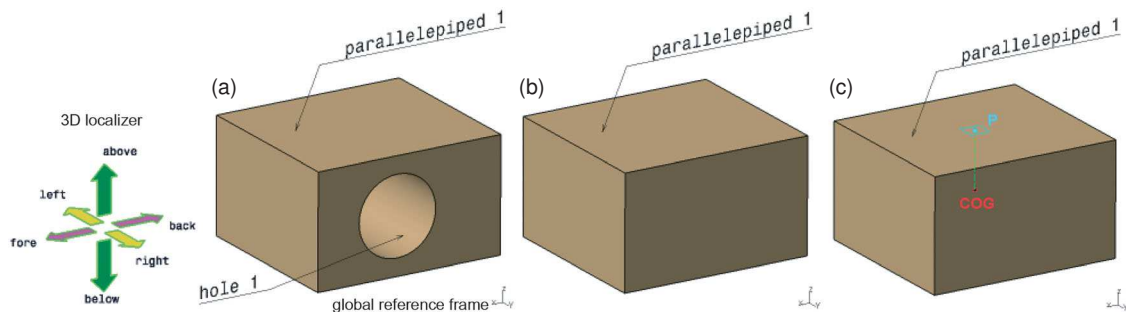


Figure 3. Example of localization.

directional information. For example, starting from a semantic description such as **[above] [parallelepiped 1]**, the generic description module (module 2) defines the following generic steps to be adapted to the CAD modeler in a later stage (module 3):

LO1. **Deactivate** all the shapes except **[parallelepiped 1]** (Fig. 3.a and 3.b);

LO2. **Compute** the position of the center of gravity (COG) and project it onto the outer skin using **[above]** as a direction of projection (namely **Z+** in the present case as presented in Table 1). **Create** a reference point P at this location (Fig. 3.c);

LO3. **Create** a reference plane going through P having **Z+** as normal (Fig. 3.c);

4.1.2. Slice operator

The slice operator consists in removing a slice of an object according to a reference plane, defining the half-space to be kept and the removal direction, and a given width of the slice. This is a simple operator in the sense that it does not make use of many geometric functions and operators. Starting from a semantic description such as **[above] [parallelepiped 1]**, **[remove] [slice] [moderately] [high]**, the generic geometric description module defines the following generic steps:

SL1. **Call** the localization operator with **[above] [parallelepiped 1]** as parameters to **create** a reference plane and a reference point P (section 4.1.1 and Figs. 4.a and 4.b);

SL2. **Transform** the **[moderately] [high]** quantifiers in a numerical value characterizing the width of the slice to be removed, i.e. OoM × 1 according to Table 2 and with OoM the user-specified Order-of-Magnitude. **Remove** the slice in the direction opposite to **[above]**, i.e. **Z-** in the present case as defined in Table 1 (Fig. 4.c).

4.1.3. Bump operator

The bump operator consists in generating a bump on an object according to a reference plane and numerical parameters describing the extent and size of the bump. Starting from a semantic description such as **[above] [parallelepiped 1]**, **[add] [bump]**, **[moderately] [wide]**,

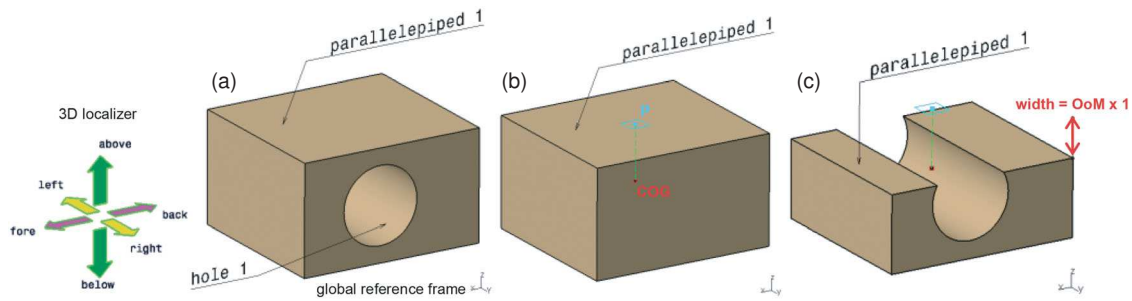


Figure 4. Example of a slice operator.

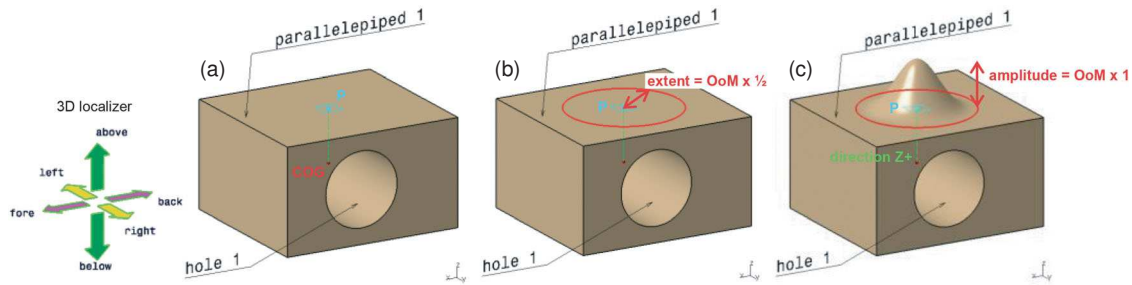


Figure 5. Example of a bump operator.

[moderately] [high], the generic geometric description module defines the following generic steps:

BU1. **Call** the localization operator with [above] [parallelepiped 1] as parameters to **create** the center of gravity COG, a reference plane and a reference point P. **Activate** shapes deactivated to compute COG (Fig. 5.a);

BU2. **Draw** a circle in the reference plane, centered on P and having [moderately] [wide] as a radius, i.e. $OoM \times \frac{1}{2}$ in the present case (Fig. 5.b);

BU3. **Deform** the parallelepiped so as to [add] [bump] with the previously defined circle as a limiting curve, P as deformation center, Z+ as deformation direction and $OoM \times 1$ the amplitude of the deformation as defined by the [moderately] [high] quantifiers (Fig. 5.c).

It can be noticed that this operator could easily be extended to a [hollow] operator while simply changing the deformation direction to Z-.

4.1.4. Bending operator

The bending operator consists in bending an object along an axis and according to a parameter characterizing the amplitude of the bend. Starting from a semantic description such as [add to] [parallelepiped 1], [bend] [moderately] [concave], [around above-below axis], the generic geometric description module defines the following generic steps:

BE1. **Determine** the direction around which the bending is performed. If the shape is a cylinder, then this direction corresponds to the axis of the cylinder, otherwise

the axis is determined from the [around above-below axis] description, i.e. the Z axis in the present case (Fig. 6.a);

BE2. **Extract** the contour C and C' of the faces which are [above] and [below] (Fig. 6.a). Between the two contours, **generate** a third contour C'' with a size characterized by the descriptor [moderately] [concave], i.e. $OoM \times \frac{1}{2}$ in the present case (Fig. 6.b);

BE3. **Deform** the outer skin of the object to bend it [around above-below axis] and so that the deformed skin goes through the three contours C, C' and C'' (Fig. 6.c).

It can be noticed that the operator could easily be extended to a [shrink] operator while changing the description [moderately] [concave] to [moderately] [convex]. In this case, the quantifier [moderately] should be inverted, i.e. $OoM \times 1/(\frac{1}{2}) = OoM \times 2$ in the present case.

4.2. From a generic geometric description to a specific draft CAD model (module 3)

After the semantic description has been transformed into a generic geometric description, a draft CAD model can be generated while adapting this intermediate description to a specific CAD modeler. Actually, this module 3 can be seen as an API (Application Protocol Interface) between the high-level semantic language and the functions and operators available in a given CAD software. Of course, it exists as many API as available CAD

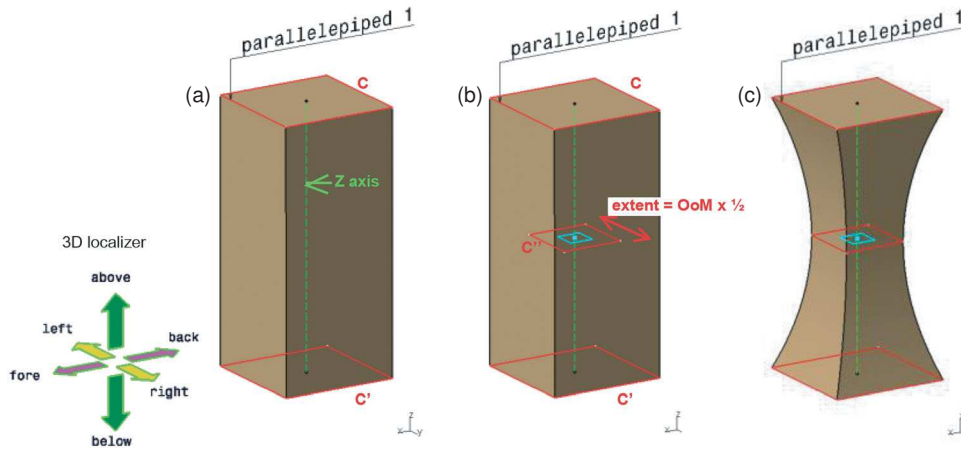


Figure 6. The bending operator.

software. In this work, CATIA V5 has been used as the CAD software for which a specific API as been developed at it is presented in this section.

Considering the implementation point of view, the localization, slice, bump and bending operators have been transformed into a set of VBA macros that can be called to apply directly on an existing model a predefined set of CATIA functions and operators controlled by parameters directly linked to the semantic description. Thus, the output is a CAD model defined by a building tree that can be used in the later stages of the PDP.

4.2.1. Localization macro

This macro uses the Part Design and Generative Shape Design modules of CATIA V5 and needs a solid model as input. The correspondences between the generic shape description introduced in section 4.1.1 and the CATIA V5 functions and operators is given in Table 3:

Table 3. Correspondences between the localization generic description and CATIA V5 macro.

Steps	Localization generic description	CATIA V5 functions/operators
L01	Deactivate	Deactivate
L02	Compute position of COG	Inertia measuring
	Project on the outer skin and create a reference point	Disassemble the solid , project COG and create a Point
L03	Create a reference plane	Plane with point and normal

4.2.2. Slice macro

This macro uses the Part Design module of CATIA V5 and needs a solid model as input. The correspondences between the generic shape description introduced in

section 4.1.2 and the CATIA V5 functions and operators is given in Table 4:

Table 4. Correspondences between the slice generic description and CATIA V5 macro.

Steps	Slice generic description	CATIA V5 functions/operators
SL1	Localization	See macro in 4.2.1
SL2	Remove the slice of width $OoM \times 1$ in the direction Z-	Create a sketch in the reference plane defined in SL1 Create a rectangular contour of size $OoM \times 100$ Generate a pocket of depth $OoM \times 1$

4.2.3. Bump macro

This macro uses the Part Design and Generative Shape Design modules of CATIA V5 and needs a solid model as input. The correspondences between the generic shape description introduced in section 4.1.2 and the CATIA V5 functions and operators is given in Table 5:











Table 5. Correspondences between the bump generic description and CATIA V5 macro.

Steps	Bump generic description	CATIA V5 functions/operators
BU1	Localization	See macro in 4.2.1
	Activate all shapes	Activate
BU2	Draw a circle of radius $OoM \times \frac{1}{2}$ in the reference plane	Create a sketch in the reference plane defined in BU1 Create a circle centered on P and having $OoM \times \frac{1}{2}$ as radius
BU3	Deform the parallelepiped in Z+ direction with the circle as a limiting line and $OoM \times 1$ as an amplitude	Disassemble the solid Create a bump with the previously defined circle as a limiting curve, P as deformation center, Z+ as deformation direction and $OoM \times 1$ as amplitude Remove the previous surface Assemble the set of surfaces Fill the close surface to get a solid

4.2.4. Bending macro

This macro uses the Part Design and Generative Shape Design modules of CATIA V5 and needs a solid model as input. The correspondences between the generic shape description introduced in section 4.1.2 and the CATIA V5 functions and operators is given in Table 6:

Table 6. Correspondences between the bend generic description and CATIA V5 macro.

Steps	Bending generic description	CATIA V5 functions/operators
BE1	Determine the direction around which the bending is performed	Simple test on the type of solid used as input. If it is a cylinder, then its axis is used, otherwise the Z axis is used
BE2	Extract the contour C and C'	Create sketch on the faces  Project the faces on the sketches to get the two contours C and C'  Create new plane between the two faces  Create a sketch  Create a circle or a rectangle with a size of  OoM $\times \frac{1}{2}$ 
BE3	Deform the outer skin to create the bend	Disassemble the solid  Create a loft through the three contours C, C' and C''  Assemble the faces  Fill the close surface to get a solid 

5. Results

To illustrate the proposed approach, three examples are introduced in this section. Starting from a semantic description entered by a sequence of menus, the developed algorithm identifies which VBA macros are to be used and which parameters are to be set up. Of course, one can imagine creating a user-friendlier interface but it was not the objective of this research. Due to the vocabulary and grammar considered, the examples are rather simple but allow us to illustrate our approach, actually different than the classical uses of CAD modelers. The reader must keep in mind that the results are obtained by a very low number of clicks.

5.1. Example on a mechanical part

In this example, the designer wants to create a mechanical part which looks like the one represented on Fig. 7.a. Actually, in this example, the designer has been asked to describe an initial CAD model using our vocabulary and grammar. Here is the result of this first step within the semantic description module whereas the

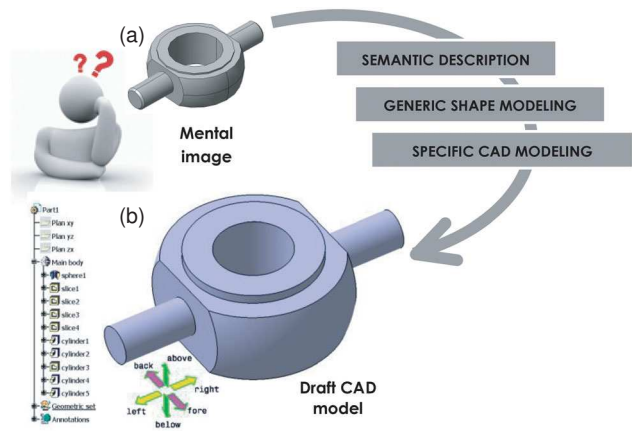


Figure 7. Initial CAD model (a) and the reconstructed one using the declarative modeler (b).

reconstructed part and building tree are presented on Fig. 7.b:

- [Start with] [sphere 1] [moderately][voluminous]
- [Above] [sphere 1], [remove] [slice 1] [very-few] [high]
- [Below] [sphere 1], [remove] [slice 2] [very-few] [high]
- [To the left of] [sphere 1], [remove] [slice 3] [extremely-few] [high]
- [To the right of] [sphere 1], [remove] [slice 4] [extremely-few] [high]
- [Above] [slice 1], [add] [cylinder 1] [few] [wide], [extremely-few] [high]
- [Below] [slice 2], [add] [cylinder 2] [few] [wide], [extremely-few] [high]
- [Above] [sphere 1], [remove] [cylinder 3] [very-few] [wide] [through all]
- [To the left of] [slice 3], [add] [cylinder 4] [extremely-few] [wide], [few] [high]
- [To the right of] [slice 4], [add] [cylinder 5] [extremely-few] [wide], [few] [high]

It can be noticed that the proposed description is not unique and several ones are possible. For such a part, a symmetry operator could have been interesting but it has not yet been integrated in our approach. If the solution is not suitable, the user can redo his/her description.

As it is visible from the building tree, the designer should create and instantiate 10 sketches and use as many operators. With our approach, the designer has to write 10 lines of description (or only 6 when the symmetry function will be implemented) using intuitive menus. Even for a basic shape like this one, it is quicker to use the declarative modeler if the user wants to make a first draft. Of course, this CAD model can then be modified during the next steps of the PDP.

5.2. Example of a spinning top

In this example, the designer was interested in generating a draft CAD model of a spinning top as imagined on Fig. 8.a. Here is a possible description :

- [Start with] [cylinder 1] [moderately] [wide], [very-few] [high]
- [Below] [cylinder 1], [add] [bump 1] [centered], [moderately] [wide], [very-few] [high]
- [Above] [cylinder 1], [add] [bump 2] [centered], [moderately] [wide], [very-few] [high]
- [Above] [cylinder 1], [add] [bump 3] [centered], [very-few] [wide], [few] [high]

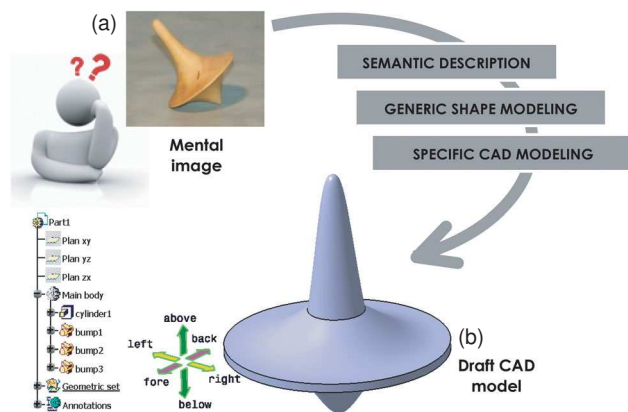


Figure 8. A spinning top (a) and its draft CAD model obtained with the declarative modeler (b).

In this example, creating each bump on a classical modeler would be time-consuming since a lot of intermediate construction entities have to be created. With the declarative modeler, the description is made in few minutes and it is really easy to obtain this draft CAD model. One can notice that this part could have been designed using a revolution. But in this case, the difficulty would have been to describe the sketch using a dedicated vocabulary and this has not been yet considered.

5.3. Example of a plastic bottle

The description of the plastic bottle represented on Fig. 9.a is made step by step like for the spinning top. Intuitively, the user should use some repetition of item but it has not been yet implemented. The resulting CAD model is displayed on Fig. 9.b. The result is a draft but it is easy modifiable by the user and it takes just few minutes to describe and generate it. It is really quick to make a first draft and everybody can easily use this descriptive modeler to create shapes without being an expert

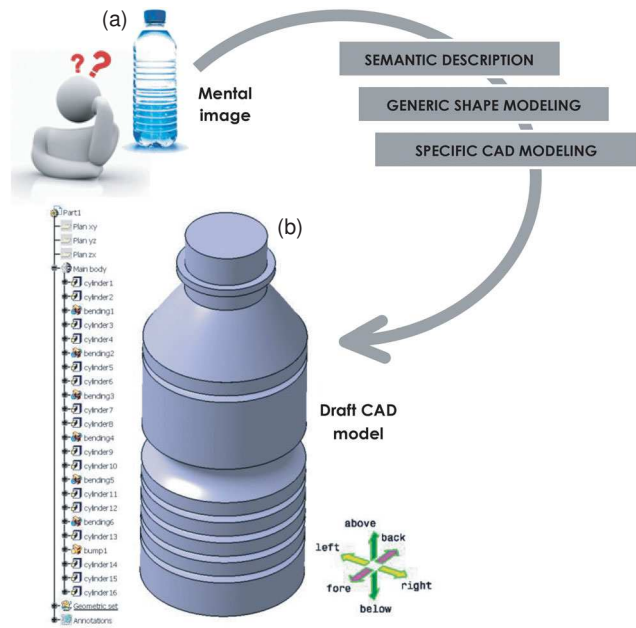


Figure 9. A bottle of water (a) and the draft model obtained with the declarative modeler (b).

in CAD modeling. This example is for us the opportunity to suggest that it can be created differently than using a revolution surface for which anyway the profile curve must be defined. Applying a similar approach on the sketch with another grammar could also be possible.

Here is the partial description of the plastic bottle:

- [Start with] [cylinder 1] [moderately] [wide], [few] [high]
- [Above] [cylinder 1], [add] [cylinder 2] [moderately] [wide], [extremely-few] [high].
- [Bend] [extremely-few] [cylinder 2]
- [Above] [bending 1], [add] [cylinder 3], [moderately] [wide], [very-few] [high]
- ...

6. Conclusion

The proposed approach proved to be relevant to generate rapidly CAD models from user-specified semantic descriptions. The proposed declarative modeler is composed of three modules. Users do not manipulate directly the functions and operators of the CAD software. They manipulate a vocabulary used to create sentences (semantic description module), which are then transformed in an intermediate generic modeling description (generic modeling module) finally used to drive several VBA macros that call the functions and operators of the CAD software (specific CAD modeling module). The intermediate module has been designed so as to be able to

easily adapt the macros to multiple CAD environments. In the future, it is foreseen to use this intermediate representation as a CAD-less representation used all along the design process and shared between the actors of the PDP whatever the CAD environments they use.

Some limits have been identified. First, even if the declarative modeler is intuitive, it still requires an adaptation phase and a beginner could have difficulties the first time he/she uses the system. The description is not unique. However, for this first work, a rigid framework based on menus to enter the description has been developed even if a different way to handle the user-interface could be more adapted. This issue has not been yet addressed. Our approach is really new in CAD. The paper proposes the first results we obtained and we do not claim that the plugin we describe corresponds to a final version. The current examples are simple and permit to illustrate the method. They can be easily produced by a classical modeler but not faster (by example by comparing the number of clicks). It is easy to understand that the vocabulary and the grammar can be easily extended allowing to handle more complex situations.

Anyway, this approach offers a gain in time and the generated model can easily be updated with classical CAD operators since the resulting model is directly embedded in the CAD environment and is defined with a traditional building tree. This model can also be exported in another CAD environment using the classical IGES or STEP file formats.

Finally, entering such a description means that some semantic information are inserted into the CAD modeler. Today, those information are solely used to create the CAD model but one can imagine that it could be used in any other steps of the PDP. In the future, this interesting possibility will evidently require large modifications of actual CAD modelers to give rise to a new generation of modeling environments.

Acknowledgments

The authors would like to thank the two French Institutes CARNOT ARTS and STAR for their supports to this research project.

ORCID

Dorian Decriteau  <http://orcid.org/0000-0001-6050-9050>

Jean-Philippe Pernot  <http://orcid.org/0000-0002-9061-2937>

Marc Daniel  <http://orcid.org/0000-0003-0560-7306>

References

- [1] Daniel, M.; Lucas M.: Towards Declarative Geometric Modeling in Mechanics, in “IDMME’96 - Integrated Design and Manufacturing”, KLUWER Publ., P. Chedmail, J-C. Bocquet, D. Dornfeld eds., 1997, pp. 427–436.
- [2] Desmontils, E.; Lucas, M.: Les modeleurs déclaratifs, *Revue de CFAO et d’informatique graphique*, 10(6), 1995, 559–585.
- [3] Falcidieno, B.; Giannini, F.; Léon, J-C.; Pernot; J-P.: Processing free form objects within a Product Development Process framework, in ASME-Press, *Advances in Computers and Information in Engineering Research*, J. G. Michopoulos, C. J.J. Paredis, D. W. Rosen, and J. M. Vance (Eds.), 1, 2014, 317–344.
- [4] Farin, G.: *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*, Academic Press, fourth edition 1997.
- [5] Hoffmann, C.: *Geometric and Solid Modeling: An Introduction*, Morgan Kaufmann, 1989.
- [6] Hoffmann, C.; Juan, R.: Erep: An editable, high-level representation for geometric design and analysis, *Proceeding of IFIP TC5/WG5.2 Working Conference on Geometric Modeling for Product Realization*, pp. 129–164, North-Holland Publishing Co., 1992.
- [7] La Gréca, R.; Daniel, M.: A Declarative System to Design Surfaces, *WSCG 2006 Conference*, ISBN: 80-86943-03-8, 2006, 17–24, Pzen, Czech Republic.
- [8] Lin, V.C.; Gossard, D.C.; Light, R.A.: Variational Geometry in Computer Aided Design, *ACM Computer Graphics*, 15(3), 1981, 171–177.
- [9] Pernot, J-P.; Falcidieno, B.; Giannini, F.; Léon, J-C.: Incorporating free-form features in aesthetic and engineering product design: State-of-the-art report, *Computers in Industry*, 59(6), 2008, 626–637. <http://dx.doi.org/10.1016/j.compind.2008.03.004>
- [10] Piegl, L.; Tiller W.: *The NURBS Book*, Springer Verlag, 1995. <http://dx.doi.org/10.1007/978-3-642-97385-7>
- [11] Regli, W.C.: *Geometric algorithms for recognition of features from solid models*, PhD dissertation, Univ. Maryland, College Park MD. 1995.
- [12] Rossignol, V.; Daniel, M.: A Declarative Modeler for B-spline Curves, in *Curves and Surfaces Design*, Saint-Malo 1999, P.J. Laurent, P. Sablonnière, L.L. Schumaker (eds.), Vanderbilt University Press 2000), 353–262.
- [13] Shah, J.J.; M. Mäntylä M.: *Parametric and Feature-Based CAD/CAM*, Wiley-Interscience Publication, John Wiley Sons, Inc., 1995.
- [14] Shah, J.J.; Rogers, M.T.: Expert form feature modeling shell, *Computer Aided Design*, 20(9), 1988, 515–524. [http://dx.doi.org/10.1016/0010-4485\(88\)90041-3](http://dx.doi.org/10.1016/0010-4485(88)90041-3)