A Persistent Naming of Shells

David Marcheix[†]
LISI (Laboratory of Applied Computer Science)
National Engineering School for Mechanics and Aerotechnics

Abstract

Nowadays, many commercial CAD systems support history-based, constraint-based and feature-based modeling. Unfortunately, most systems fail during the re-evaluation phase when various kind of topological changes occur. This issue is known as "persistent naming" which refers to the problem of identifying entities in an initial parametric model and matching them in the re-evaluated model. Most works in this domain focus on the persistent naming of atomic entities such as vertices, edges or faces. But very few of them consider the persistent naming of aggregates like shells (any set of faces). We propose in this paper a complete framework for identifying and matching any kind of entities based on their underlying topology, and particularly shells. The identifying method is based on the invariant structure of each class of form features (a hierarchical structure of shells) and on its topological evolution (an historical structure of faces). The matching method compares the initial and the re-evaluated topological histories, and computes two measures of topological similarity between any couple of entities occurring in both models.

The naming and matching method has been implemented and integrated in a prototype of commercial CAD Software (Topsolid).

Key Words: CAD, Parametric design, Persistent naming.

1. Introduction

Static solid modeling systems (B-rep, CSG, etc.) largely used in the Computer Aided Design (CAD) area are more and more replaced by dynamic modeling systems (known as history-based, constraint-based and feature-based modelers) which allow both to express and to record conceptual designs and "design intents". These dynamic modeling systems are often gathered under the term of parametric modeling systems [15]. A parametric model is composed of a representation of an object, of a set of parameters (characterizing the object) and of a list of constraints (equations or functions) applied to the object. By extension, a parametric modeler is a system for geometric design which preserves not only the explicit geometry of the designed object (called *parametric object* or *current instance*), but also the set of constructive gestures used to design it (called *design process* or *parametric specification*).

This two-fold data structure enables rapid modifying by re-evaluation. However, when re-evaluation leads to topological modifications, references (between entities) used in the constructive gestures are difficult

[†] Corresponding author:
marcheix@ensma.fr

to match in the new context, giving results different from those expected. A persistent naming system, robust regarding some topological modification, proves useful to preserve, from a re-evaluation to another, references between topological entities. It is the problem known as “persistent naming” or “topological naming” [5][9]. There are two major situations in which persistent naming is need. First when parametric models change inside one CAD system and second when parametric models are exchanged to another CAD system. In the context of parametric data exchange, however, a full persistent naming solution is not essential during transfer because there are no parametric changes [14]. On the contrary, the main goal is to construct a model in the target system that is as similar as possible to the model in the source system, and specific problems can arise principally because constructive gestures can be defined differently in the two systems. Then, in this paper we focus on the persistent naming problem when parametric models change inside one CAD system.

This paper is structured as follows. In section two, we discuss some pre-existing works and we give a detailed account of the major issues about naming in parametric modeling. The third and fourth sections introduce an alternative approach and a matching method which enable to address theses different issues.

2. Previous work and Issues

Although parametric modeling has developed and expanded for more than one decade, both in research centers and in the industry, probably due to commercial competition only a rather small number of publications were issued in the field of persistent naming. A detailed survey of naming methods could be found in [11]. One important precursory work in this domain is that of Hoffmann and Juan [7]. Over the same period of time several authors have analyzed the internal structure of parametric data models, proposing some editable representations [7][10][16], studying the persistent naming in a context of decomposed pointsets [13], discussing their underlying mathematical structures [12], describing the problems, either of the semantics of modeling operations [3][6] or of constraint management [4]. Several naming scheme and persistent naming mechanisms based on geometric or topological characterization of ambiguous entities have also been proposed [6][9][17]. A significant work has been proposed by Chen [6]. He suggests a model that is composed of two representations. For the first one, he uses an editable representation, called Erep [7], which is an unevaluated, high-level, generative, textual representation, independent of any underlying core modeler. It abstracts the design operations, contains the parametric specification and stores all entities by name. The second representation, evaluated and modeler dependent, contains the geometry (the current instance). The link between these two representations is obtained by a name schema which establishes the link between the geometric entities of the geometric model and the generic names (persistent) of the unevaluated model. Chen defines a precise structure for naming invariant entities, vertices, edges and faces (before the interaction with the actual geometry) which belong to a specific class of form features build by sweep operations (extrusions or revolutions). In this precise context, every entity in a sweep is named by reference to the corresponding source entity of the swept 2D contour and the constructive gesture. He also proposes an identification technique for contingent entities based on compositions of topological contexts (more or less extended topological neighborhoods) and on feature orientations. Each contingent entity is described by the smallest unambiguous topological context [5][6]. Chen has mainly studied the naming problem at the construction stage of the parametric model, and has given only very few indications about matching method used during the reevaluation process. Moreover, the identification method is fairly voluminous, since all the entities seem to have to be identified, even if they are not referenced by any constructive gesture. In particular, is it the only method which gives names and records explicitly the history of all invariant entities (vertex, edge, face). However, the suggested model represents two major contributions in this domain : on the one hand, some concepts of topological identification which will be used thereafter per many of other approaches, and on the other hand a very precise study of cases of ambiguity.

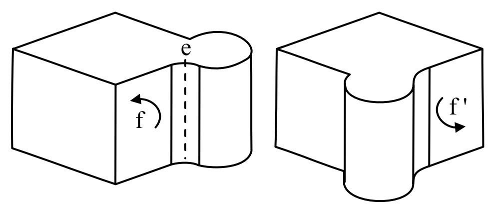


Fig 1 : Original and modified solid's Brep

Raghothama and Shapiro [12] provide what is believed to be the first formal analysis of the nature of this problem and suggest a solution that can be guaranteed to work within certain well-defined limitations. They use the structure of a B-rep as a cell complex to control and evaluate the effects of changes in parameter. In this way, they introduced boundary representation deformations in their approach to the problem, which they related to the question of how to define families of objects in parametric modelling. Then, the main problem to address is to be able to determine which objects belong to a family, defined by a prototype model, and which objects do not. For that, they define BR-invariance and BR-variance in order to characterize continuity in term of cells complex. Basically they argue that as long as a continuous boundary deformation is possible from the prototype model to the new instance, the latter is considered to be a member of the family. However, as they also admit, definition of continuous deformations, BR-invariance and BR-variance seem to be too restrictive when the model's topology changes radically, and operations like splitting and merging of model entities would have to be dealt with, in order to allow topological changes such as the elimination of holes, but this has not been elaborated. Moreover, the notion of parametric family must depend on the semantics of constructive operations used to create the object. For example, using the Fig 1 introduced in [12], the two objects can be regarded as belonging to the same or different families according to the meaning of the blend constructive gesture : if the blend is achieved either on the intersecting edges between the cylinder and the face f or on the oriented edge e according to position and orientation of this edge.

Clearly, it may not be possible to provide a complete solution, since this does not appear to exist in any practical system at present. Most of the previous studies discussed parametric modeling in terms of creation (but not or in a succinct way re-evaluation), and none deals with the whole range of major issues introduced in the following of this section. However, some mechanism must be provided that will give correct results in most cases and also, of course, be compatible with the approaches taken by the CAD system developers.

The main problem for parametric re-evaluation is to characterize geometric and topological entities of a parametric model. Characterizing entities consists in giving them a name at design time and "finding them" again at re-evaluation time (i.e. matching entities of the initial model with entities of the re-evaluated model.)

Let us take the example of Fig 2 to illustrate this problem. The initial model is designed by means of a parametric specification containing four successive constructive gestures. The fourth one consists of rounding edge *e*. If the initial model is saved after this fourth step, the current instance no longer contains edge *e*: it was removed by the rounding function. Thus, the rounding function, which has edge *e* as input parameter, cannot any longer be represented in the parametric specification part of the model. Therefore, "names" are needed to represent entities referenced in the parametric specification whether or not they exist in the current instance in order to be able to re-evaluate these entities.

Moreover each constructive gesture creates a number of entities which have to be distinguished and therefore named, to be referenced by further constructive gestures, even if the same number of entities exists in all possible re-evaluations (no topological change). Therefore, each entity should be named in a non-ambiguous and unique way at design time.

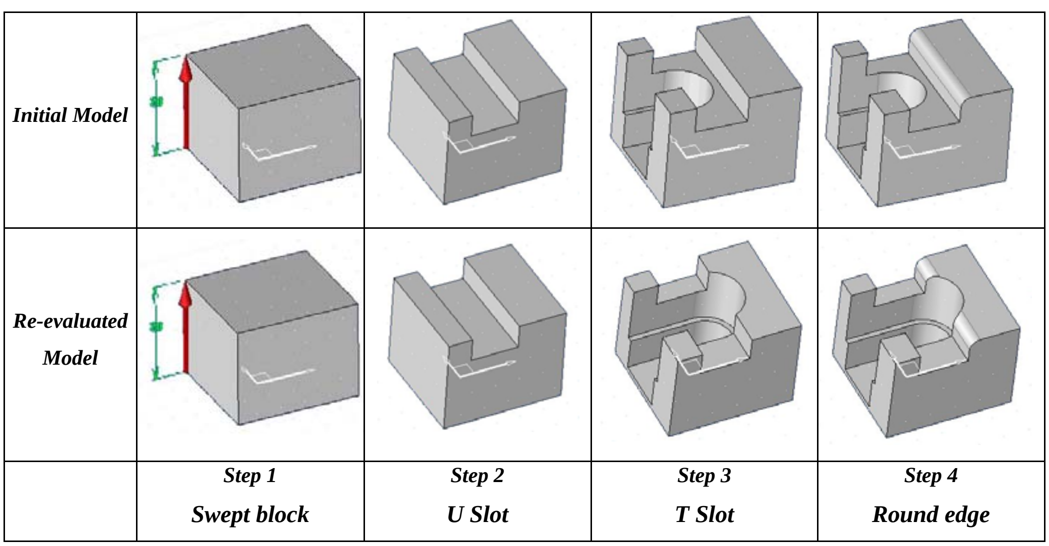


Fig 2: Naming problem

The problem is even more complex for parametric models of which the entities and the number of entities change from one evaluation to another. Let us return to the previous example, but this time in the re-evaluated model. We notice that, at step 3, the edge *e* has been split into edges e_1 and e_2. Thus at step 4 the problem is to determine which edge(s) has(ve) to be rounded. The problem is to identify, i.e. to match, edge *e* with edges e_1 and e_2 despite topology changes. Thus, when re-evaluation leads to topology changes a new issue is to match two different structures. The naming mechanism should be powerful enough to perform a robust matching during re-evaluation. For example, the innovative Kripac's model [9] introduces an interesting graph structure for identification of any topological entities based on face history (creations, splits, merges and deletions of faces), but it does not allow to record that a selected mapping was only approximated as it uses a discrete metrics. That strongly induces the later mappings introducing "piece loss" problem during

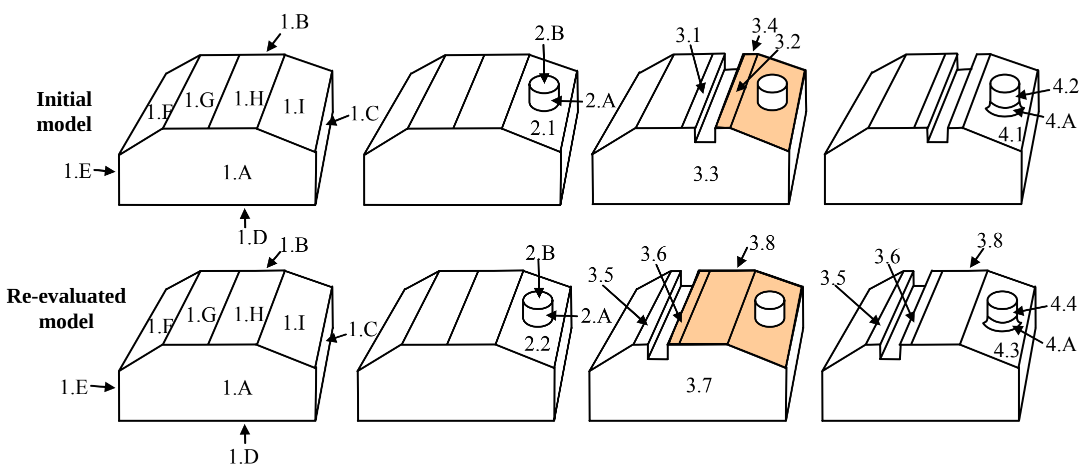


Fig 3 : Re-evaluation of a parametric model consisting of four construction steps: construction of a block by extrusion of the face 1.A, addition of a cylindrical protrusion, subtraction of a slot and then rounding of the edge between a shell and the protrusion

re-evaluation and would deserve to be taken into account. Indeed, the matching algorithm consists on a backward-forward search in the graph structure and a *cross-analysis*. More precisely, starting from a given face *f*, a backward search is done in the graph, until reaching an ancestor face already matched. Then, starting from this common face, a forward search is done in the graph, to find the mapping of *f*. The matching calculated on the ancestor face is done approximately. Therefore, it is possible not to analyze all faces that should be analyzed. Fig 4 illustrates this problem. The matching of face 3.2 is done at the re-evaluation's third step. The face 2.1, the ancestor face of face 3.2, has been matched with face 2.3 at the re-evaluation's second step. Then, the cross-analysis to find the mapping of face 3.2 is done only between faces coming from 2.1 and faces coming from 2.3. In particular, in this example, only faces 3.1 and 3.2 will be crossed with face 2.3. The algorithm "looses" the face 3.5 which can be considered part of face 3.2. Moreover, this matching algorithm is very sensitive to the subdivision of the topological neighborhood. It tries to match new faces with the old ones by analyzing the topological neighborhoods. The cross-analysis consists of finding the longest common face cycle in the topological neighborhoods. Unfortunately, as shown in [11], this cross-analysis is not relevant in case of irregular subdivision of the topological neighborhoods.

The last important issue relates to the persistent naming of aggregates. Most works in the persistent naming domain focus on the persistent naming of atomic entities such as vertices, edges or faces. But very few of them consider the persistent naming of aggregates. Particularly, the persistent naming of shells (a set of connected faces) currently represents an important requirement for CAD.

In fact, the problem here is to have a model able to reference in the parametric specification (i.e. to name in a persistent way), entities corresponding to various semantics and able to characterize various user design intents. This could be done particularly by naming high level abstractions like shells. For example, the designer might want to express that a feature is applied on a shell corresponding to a slot bottom rather than on a particular face. To illustrate this problem, let us take the example of Fig 3, but modifying a little the re-evaluation process by moving the position of the cylindrical protrusion to the face 1.F. According to whether the rounding function, at step four, was expressed between the cylinder and face 2.1 (case 1) or between the cylinder and the top shell composed of faces <1.F, 1.G, 1.H, 1.I> (case 2), the re-evaluated model is different because the "design intent" is different. In the first case one obtains a model where the round disappeared since it cannot be made any more between the cylinder and face 2.1. In the second case one obtains a model where the round always exists since it could be made between the cylinder and the top shell. Actually, in most of CAD systems it is possible to name some shells (also named "smart collectors", "selection intend", etc.). But either the domain of referenceable shells is considerably restricted to some particular shells (i.e. usually only the complete 2D manifold boundary of the part or the feature), either the re-evaluation process leads to pertinence and stability problems of the naming mechanism. Let's take again the example in Fig 3. The shell used and referenced by the last constructive gesture is the shell S^{ini} composed of faces <3.2, 2.1> (grayed shell). It represents one of the two parts appeared at step 3 as a result of a split of an initial shell corresponding to the top of the object created at step 1 and then possessing a user meaning. During the re-evaluation process, the position of the slot is changed and the topology of the object too. In order to be able to re-evaluate the fourth constructive gesture (rounding operation) and since the topology is now different, it is necessary to calculate in the re-evaluated model the shell corresponding to S^{ini} (i.e. to calculate the matching of S^{ini}). To solve this problem, a classical approach consists to characterized S^{ini} by the set of faces contained in this shell (S^{ini}={3.2, 2.1}). Then, using both a mapping mechanism for faces and this shell characterization it is possible to compute the shell $S^{réev}$={1.H, 2.2} corresponding to S^{ini} in the re-evaluated model because, at step 4, faces 3.2 and 2.1 are mapped respectively to faces 1.H and 2.2. Unfortunately, $S^{réev}$ is not suitable with the user design intent who wanted to designate one of the two shells appeared at step 3 by split of the initial shell corresponding to the top of the object created at step 1. Obviously, the shell $S^{réev}$ should be composed of the set of faces {1.H, 2.2, 3.6}. Then, the problem is how to represent and establish correspondence between shells in the presence of topology-changing operations.

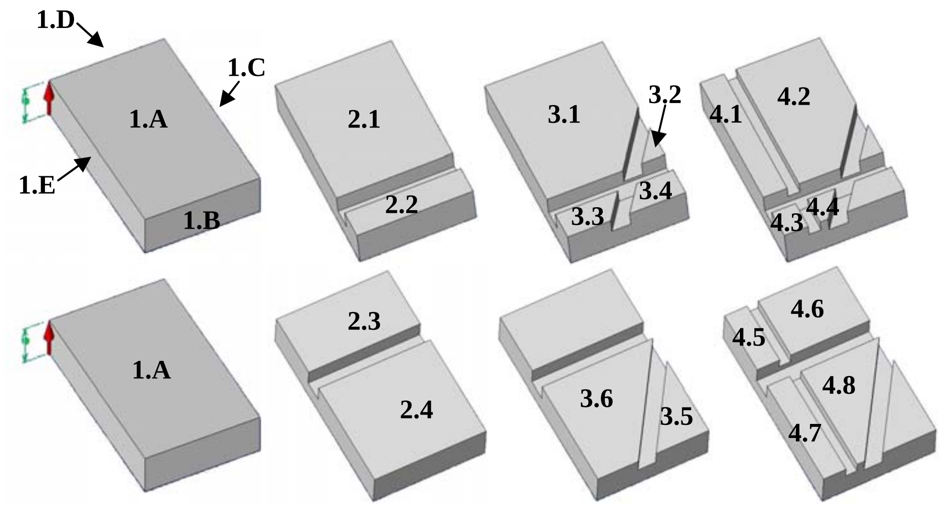


Fig 4: "Loss of faces" during matching (Is 3.2 matched with 3.5 ?)

3. Principles of the proposed approach

The global architecture of the model consists of a three layers architecture :

- The *parametric specification layer* contains the description of the modeling operations (feature-based operation, Boolean operations, chamfering operations, blending operations, 2D contour and sweeping operations, etc.) that defined the parametric object; these operations refer to the layer of names.
- The *name layer* allows representing geometric or topological entities that existed at some time during the parametric design process; these names can reference geometric or topological entities if they still exist in the geometrical layer.
- The *geometrical layer* contains the geometric and topological entities that constitute the current instance; it can also contain geometric and topological entities that existed in some previous steps of a parametric design to support UNDO functions.

To define robust names allowing solving the issues introduced in section 2, we have proposed to distinguish two types of geometric and topological entities [1]:

- *Invariant entities*. An invariant entity is a geometric or topological entity that can be, completely and unambiguously, characterized by the structure of a constructive gesture and its input parameters, independently of involved values. In Fig 2, invariant entities include the end face of the swept block, the lateral shell of the horizontal slot with its begin and end faces (that can, or not exist), etc.. To characterize, i.e. to "name", such entities, information models are to be defined that relate these entities to constructive gestures and to their input parameters.

- *Contingent entities*. Beside those invariant entities, there exist entities that depend on the context of a constructive gesture. We call contingent entity a geometric or topological entity that results from an interaction between the pre-existing geometric model and invariant entities resulting from a particular constructive gesture. For example, in Fig 4, the number of lateral faces generated by the second slot (step 3) in the initial and in the re-evaluated model is not identical. A naming mechanism is also required to define how to name these contingent entities.

The proposed method enables to identify, in a single and non-ambiguous way, both invariant and contingent entities using a direct acyclic graph (dag), called *shell graph*, consisting of a hierarchical structure of shells above of a historical structure of faces (i.e. dag of faces).

The hierarchical structure of the shell graph is defined recursively in the following way: A node represents a shell (a connected set of faces) and an oriented link from node *a* to node *b* means that the shell *b* is include in the shell *a*. Let *f* be a face in the geometric model existing or having existed at a previous construction step in the geometric model. On the lowest level of the hierarchical structure a node represents a particular shell corresponding to a face f_i. This node is also called *elementary-shell*, because a face can be considerate as an elementary-shell. It is associated to a label equal to 0 corresponding to a hierarchical level of the elementary-shell in the graph and it is noted S^0.

Then, a higher hierarchical level shell (also called *non elementary-shell*) is defined recursively by the connected set $S^p = \{S_1^{q_1}, S_2^{q_2}, \dots, S_n^{q_n}\}_{n \geq 1, q_i \leq p, p = 1 + \max(q_i)_{i=1..n}}$ where p is the hierarchical level of the shell in the graph. This shell is represented in the graph by a node noted S^p connected through oriented links to nodes $S_1^{q_1}, S_2^{q_2}, \dots, S_n^{q_n}$.

Let a node S^p be connected through oriented links to a node S^q (according to the previous definition, we must have p>0 and p>q). This means that the corresponding shell S^q is include in the corresponding higher hierarchical level shell S^p. S^p is called *super-shell* for S^q and S^q is called *sub-shell* for S^p.

The graph is then structured in shells, sub-shells, etc., in order to be able to apprehend the geometry at various levels of granularity, and thus to express different semantics. An example of this hierarchical structure is given in Fig 5. For example, *1.G, 1.H, 1.I, 1.F* are elementary-shells (level 0). They are sub-shells of the *1,top* (level 1) and *1,top* is the super-shell of these elementary-shells. *1,block* (level 3) is the super-shell of *1,front* (level 1), *1,back* (level 1) and *1,lateral* (level 2).

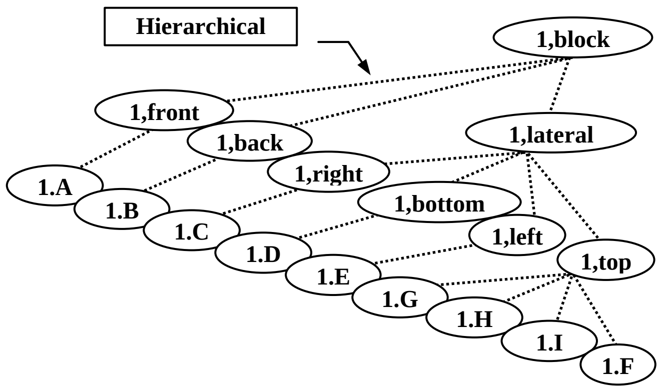


Fig 5 : Invariant hierarchical shell structure corresponding to the swept block in Fig 3

Such a definition allows representation of connected shells (a shell is a composition of connected elementary or non elementary-sub-shells), and overlapping shell (an elementary or non elementary-shell could belong to different non elementary-super-shells).

In the following of this paper, we use equivalently the term face or elementary-shell.

The historical structure of the shell graph is defined in the following way: The goal is to follow shells evolution (split, merge, destruction, modification) in order to be able, during model design, to identify the involved shells, then, during re-evaluation, to identify the effective shells (in the current instance) corresponding to the referenced shell. We only represent explicitly the elementary-shell (i.e. face) history in the graph because it is the only historical information necessary to solve the persistent naming problem of shells (see sections 4 and 5). Each node represents a face, which exists or has existed in the model. All the faces without outgoing historical links exist in the geometry. Invariant shells have no historical ancestor. They define entries in the shell graph. At a construction step *t*, an invariant non elementary-shell referenced in a constructive gesture consists of all the faces belonging to this shell and existing at step *t* in the current instance (i.e. the historical leave faces at step *t* of the faces (i.e. elementary sub-shells) contained in this non elementary-shell). For example, at the fourth step of the initial construction process, the non elementary-shell <1, top> in Fig 8 consists of faces <1.F, 1.G, 3.1, 3.2, 2.1> in the current instance.

Fig 9 illustrates a part of the face graph corresponding to the object presented in Fig 3. The top face evolution is described by *historical links*. In this example, the top face of the block is split differently during the construction and the re-evaluation process then the initial and the re-evaluated graphs are not the same.

4. Naming of entities

It is necessary to associate a characterization to each graph's node in order to identify in a persistent way the shells during construction and re-evaluation processes.

The identification of a shell is based on unchanging elements that characterize it in a unique way. In a parametric model, what never changes is the construction process (we consider the modification of the construction process as an edited model and not as a model re-evaluation). Therefore, shell characterization is done by means of the construction step number (creation order) and by means of additional information which identifies each shell in a unique way. So the characterization of a shell is as follows : **<StepId, Additional information>**. The problem is to define the necessary and sufficient additional information that enables both characterization of shells in a unique way within each construction step and persistent re-evaluation.

As stated before, we consider that each constructive gesture can be subdivided in two steps. Firstly, the creation of the feature where all shells must be named (for example the elementary-shell 1.A in Fig 4), and secondly, the feature positioning within the existing geometry. This interaction with the existing geometry leads to modification and deletion of existing shells and to creation of new contingent shells. These contingent shells must also be named (for example the elementary-shell 2.1 in Fig 4). Therefore, there are two types of naming to implement: one for invariant shells and another for contingent shells. A preliminary study of only elementary shells matching can be found in [2].

4.1 Naming of invariant shells

4.1.1 Naming of invariant elementary-shells (faces)

According to the feature taxonomies proposed in [1][8], invariant faces present in the graph are generated by four types of features (primitive, transition, extrusion and revolution). For the two first cases, the invariant naming of the generated faces are ensured by a unique topological traversal of the object (see [1]).

In the case of extrusion, the *generator contour* is swept along a *director path*. Each resulting

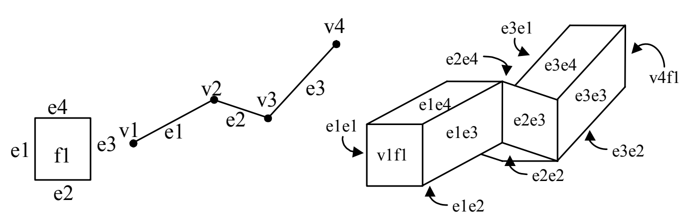


Fig 6: Invariant face naming.

topological entity corresponds to the cartesian product between a topological entity of the profile and a topological entity of the path. For example in Fig 6, face *e1e4* from the extruded object corresponds to the cartesian product between the director edge *e1* of the director path and the generator edge *e4* of the generator contour. In a similar way, the internal face *v2f1* comes from *v2* (director path) and *f1* (generator contour). Robust naming of each contour entity and of each path entity is fundamental to enable robust naming of faces in the graph.

Each invariant face is associated with an invariant stamp representing the additional information allowing to characterize each face in a unique way within each construction step. So, for each graph node corresponding to an invariant elementary-shell the following characterization is associated: <StepId, <InvariantStamp>> where StepId is the construction step of this face and InvariantStamp is <generator entity, director entity>.

4.1.2 Naming of invariant non elementary-shells

Invariant shells are entirely defined by the constructive gesture. Each constructive gesture must then be associated in advance to an invariant structure which allows non elementary-shell naming in a unique and unambiguous way. Fig 5 gives an example of a hierarchy of an invariant shell. In [1][8], an invariant hierarchical structure is defined for each form feature classified in the taxonomy. However, it is suitable to extend a little this taxonomy, in order to integrate invariant aggregates generated automatically by the system during the interaction between the object (part) and the feature. For example, in case of a protrusion feature, this is done by adding a new super-shell (aggr-prot) in the hierarchical structure of the graph between the protrusion and the face or the shell of the block on which this protrusion is fixed to (see Fig 7). An invariant aggregative shell can consist of either invariant or contingent sub-shells.

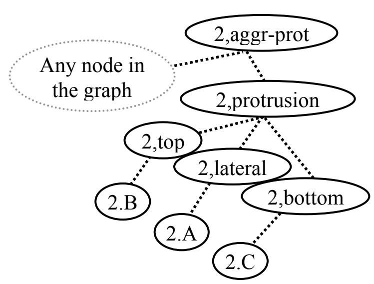


Fig 7: The invariant hierarchical shell structure of a cylindrical protrusion feature.

Then, in this way, each entity of the invariant structure can be associated with an unique *invariant stamp* defined in the taxonomy (top, lateral, protrusion, aggr-prot, etc.). This invariant stamp represents the additional information allowing to characterize each shell in a unique way within each construction step. So, for each graph node corresponding to an invariant non elementary-shell the following characterization is associated: <StepId, <InvariantStamp>> where StepId is the construction step of this non elementary-shell.

4.2 Naming of contingent shells

4.2.1 Naming of contingent elementary-shells (faces)

For contingent face, the additional information consists of an iterative number (an arbitrary but unique number for each construction step). This name (<StepId, <ItNumber>>) is sufficient to characterize in a unique way each entity during the construction process, but is not sufficient to allow an ulterior matching of contingent faces. Therefore, each contingent face in the graph is associated with information about his topological neighborhood (see section 5.2.1).

4.2.2 Naming of contingent non elementary-shells

Contingent non elementary-shells result from the evolution (split, merge, destruction, modification) of an invariant non elementary-shell. They appear in a conditional way because of interactions occurring during the construction process. So, contingent non elementary-shells possess necessarily an invariant ancestor non elementary-shell (several in case of fusion).

The only contingent non elementary-shells explicitly represented in the graph are the contingent non elementary-shells referenced in a constructive gesture. When a contingent non elementary-shell must be referenced by a constructive gesture, a new node corresponding to this shell and consisting of faces is generated in the graph. This shell can then in its turn being used in an ulterior constructive gesture to build, for example, an invariant aggregative shell. This node is associated with the following characterization :

<StepId, <InvariantShell, FaceList>>,

where *StepId* is the construction step of this shell, *InvariantShell* is the graph node corresponding to the invariant ancestor shell from where this contingent shell results, *FaceList* is the list of the faces contained in this shell at the StepId construction step.

As explained before, contingent non elementary-shells can be completely calculated and characterized at the time of their referencing in the parametric specification according to invariant and contingent elementary-shells (faces) and invariant non elementary-shells. Avoiding to calculate and to represent at each construction step the history of the various contingent non elementary-shells simplifies considerably both the time and the memory complexity of the model. Then, contingent non elementary-shells are calculated through a boundary traversal of the geometric model just when the user wishes to use it in a constructive gesture. The user selects a face F belonging to this shell, and the system must be able to calculate dynamically the list L_{cont} of non elementary-shells containing this face. The calculation of L_{cont} is achieved in two steps :

- firstly, the calculus of L_{inv}, the list of all invariant non elementary-shells containing F, by a backward (history of faces) and upward (hierarchy of faces) traversal of the shell graph from F. We can note that a face can have several direct historical predecessors if its history contains a fusion. In a same way, a shell can belong to different super-shells in case of overlapping super-shells (see for example shell <1, top> in Fig 8). Finally, a shell can belong to a contingent super-shell. In the example proposed in Fig 8, the selection and then the backward traversal from the face 3.2 at step four of the construction process generate the list L_{inv}={<1, top>, <1, latteral>, <1, block>, <2, aggr-prot>}.

- Secondly, for each shell S_{inv} belonging to L_{inv} a contingent non elementary-shell S_{cont} is calculated and added to L_{cont}. A contingent non elementary-shell is necessarily an historical descendant of an invariant ancestor non elementary-shell. So for each shell S_{inv} an incremental topological traversal from F of the geometric model at step StepId is performed and each encountered face is added to S_{cont} if it is a historical descendant of invariant faces (i.e. elementary-sub-shells) belonging to S_{inv}. In this way, S_{cont} represents the biggest connected component containing F and historical descendant of S_{inv}.

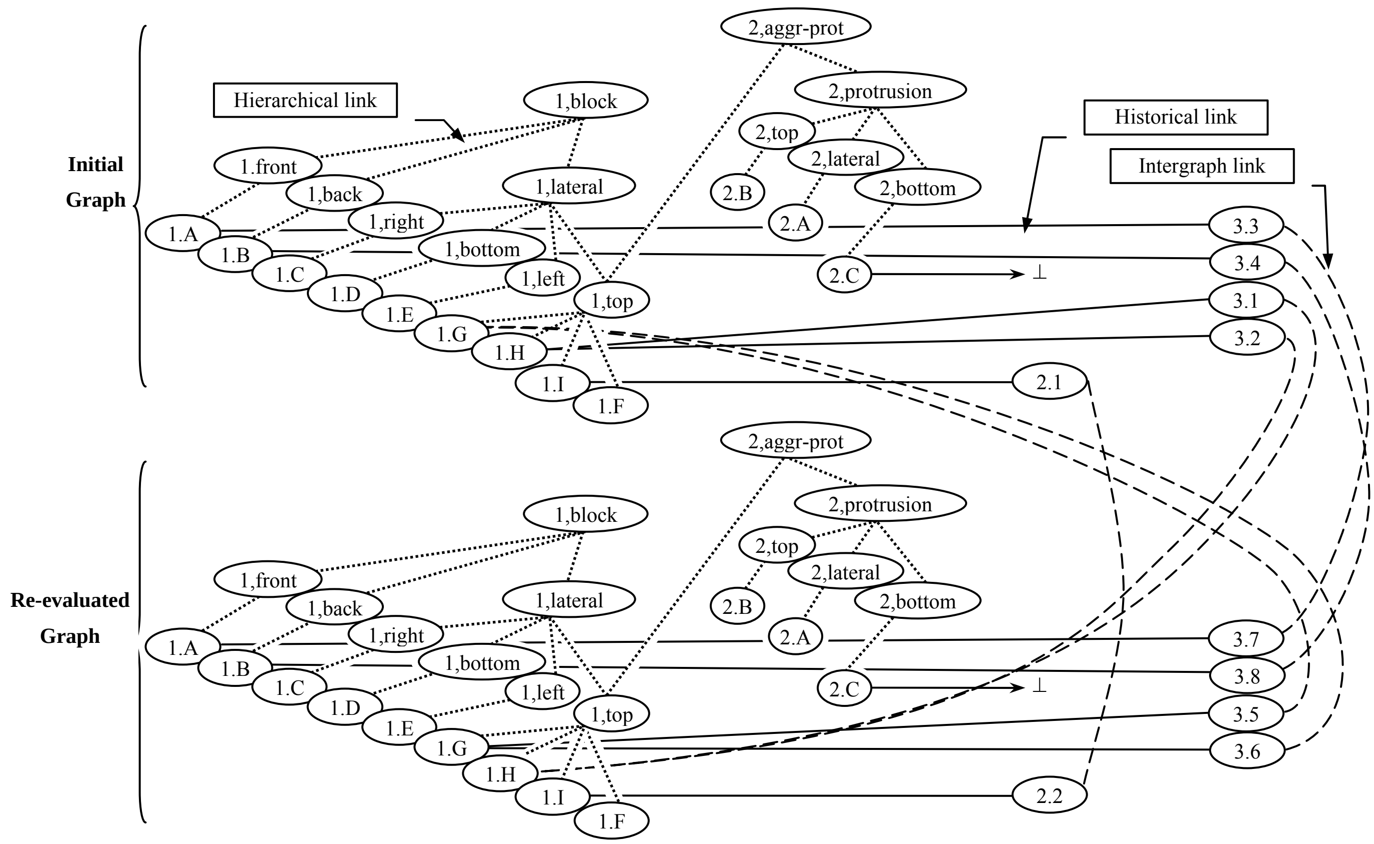


Fig 8 : Part of the shell graph corresponding to the parametric model in Fig 3 for step 1, 2 and 3. The cylindrical protrusion feature is positioned on the shell <1, top>. To make the graph readable, intergraph links between the invariant structure and the nodes corresponding to the slot feature are not represented.

For example, let us continue with the selection of the face 3.2, at step four of the construction process, in Fig 8. For each invariant shell in L_{inv}, a contingent shell is generated dynamically through construction of connected components using an incremental topological traversal from face 3.2. For example, the first invariant shell <1, top> in L_{inv} generates the contingent shell <4, <1, top>, {3.2, 2.1}>. Indeed, the incremental traversal from face 3.2 is achieved on the geometric model following the topological adjacencies. Each encountered face is both used in its turn to continue the incremental traversal and added to the contingent shell, if it is an historical descendant of elementary-sub-shell (i.e. faces) belonging to the <1, top>. The encountered face belonging to <1, top> is only the face 2.1. Using in a similar way all invariant shells belonging to L_{inv}, different contingent shells are generated and added respectively in the list L_{cont}={<4, <1, top>, {3.2, 2.1}>, <4, <1, latteral>, {3.2, 2.1, 1.C, 1.D, 1.E, 1.F, 1.G, 3.1}>, <4, <1, block>, {3.2, 2.1, 1.C, 1.D, 1.E, 1.F, 1.G, 3.1, 3.3, 3.4}>, <4, <2, aggr-prot>, {3.2, 2.1, 2.A, 2.B}>}.

5. Matching of entities

The matching of entities consists in associating *n* entities of the initial model with *m* entities of the re-evaluated model in order to decide if each of the *n* entities corresponds to one or several entities of the re-evaluated model, and conversely if each of the *m* entities corresponds to one or several entities of the initial model. The matching can be done using the geometry and/or the topological neighborhood of entities to be referenced. Topology use allows to get a robust matching method in relation to important geometric variations and small topological variations. However, in some particular cases, when the model contains non-linear entities, topological neighborhood, even extensive [7], are ambiguous and do not allow characterizing in a unique way an entity of this model. Thus, it would be proper to use an additional geometric mechanism (feature orientation, etc.) allowing to remove this ambiguity [6].

Matching quality is very relative and generally depends on operations on the one hand and on the semantics the designer wants to express on the other hand. Some semantics information can be available only in the parametric specification layer. When the parametric specification layer asked for a name, the responsibility of a generic naming layer is to provide information enough about this name in order to enable interpretation of this name according to the semantics of a particular operation accessible in the parametric specification layer. For example, suppose that a face F is referenced by two constructive gestures in the parametric specification : a painting of F and a build of a protrusion on F. A persistent name is asked by the parametric specification layer for F to the naming layer in order to store the construction process. A name called N_1 is then constructed and returned by the naming layer for F. Now, modifying some parameters of the construction process, suppose that the re-evaluation of the parametric specification split F in two faces. It seem obvious that N_1 must correspond to different entities according to the constructive gesture of the specification layer that uses this persistent name : for the painting operation the two faces should be painted, but for the protrusion operation only one face could be used.

According to the previous statements, and since matching an entity with another means that both entities are geometrically and topologically similar but not necessarily identical, our approach consists in defining a naming layer which measures the similarity between the set of potentially corresponding entities (called crossing process) and returns these measures to the parametric specification layer. The later will then use this information according to the semantics of a particular constructive operation to select the well matching.

To achieve this, a matching value between faces present in the initial graph (called *IG*) and faces present in the re-evaluated graph (called *RG*) is calculated. Others entities (non elementary-shell, edge, etc.) are mainly named according to this matching.

5.1 Matching of invariant shells

The matching method is similar for both invariant elementary-shells and invariant non elementary-shells. In both cases, the characterization <StepID, InvariantStamp> of invariant shells is associated to the constructive gesture. This characterization is unique and unambiguous during both construction and re-evaluation of the model. Then, the matching calculus is obvious. For example, in Fig 8 the invariant shell <1, top> can be found in the two invariant hierarchical structures.

5.2 Matching of contingent shells

5.2.1 Matching of contingent elementary-shells (faces)

At the re-evaluation step, we calculate topological similarity between *p* faces of IG and *q* faces of RG. Thus, we speak about *"crossing"* between two sets of faces based on each face topological neighborhood. For each face *F*, we note $\gamma_F=\{o_0,o_1,..o_n\}$ the circuit of oriented edges $(o_i)_{i=0..n}$ of the boundary of *F*. The crossing result is a set of inter-graphs relationships that can exist between faces of IG and faces of RG.

Let $\gamma_{F_{ig}}=\{o_0, o_1,.. o_n\}$ and $\gamma_{F_{rg}}=\{o_0', o_1',.. o_m'\}$ be the circuits associated with faces F_{ig} of IG and F_{rg} of RG. We define $\Gamma_{F_{ig}}$ and $\Gamma_{F_{rg}}$ the sets of the partial sub-paths of $\gamma_{F_{ig}}$ and $\gamma_{F_{rg}}$; a partial sub-path of a circuit is a sub-path of the circuit where some oriented edges have been deleted.

First, one could notice that actually an oriented edge cannot appear in two distinct face circuits in an oriented model. If an oriented edge appears in the circuit of face *F* and the circuit of face *G* then it means that *F* and *G* have opposite orientation: the model is not oriented. So, for each oriented edge *o*, there is a unique face of which circuit uses *o* and we call *neighbor adjacent face* of *o*, the adjacent face of the edge associated with *o* that does not use *o* in its circuit.

In order to quantify topological matching, we define the equivalence relation $\sim_{Adj}$ between face circuits γ and γ', defined by: $\gamma\sim_{Adj}\gamma' \Leftrightarrow \exists(o_i)_{i=0..n}$ and $(o_i')_{i=0..n}$ / $\gamma=o_0.. o_n$, $\gamma'=o_0'.. o_n'$ and $\forall i\in\{0..n\}$, the invariant ancestor face of the neighbor adjacent face of o_i is also the invariant ancestor face of the neighbor adjacent face of o_i'. In other words, when you come along γ and γ' and you consider only the invariant ancestor of neighbor adjacent faces, you get the same circular list of invariant faces around the faces of which circuits are γ and γ'.

Therefore, we can define, $\Gamma_{F\cap G}$ the set of elements of Γ_F that are equivalent to an element of Γ_G according to our relation. This way, $\Gamma_{F\cap G}$ contains all partial sub-paths of γF such as there is at least an element of γG of which circular list of adjacent faces, in terms of invariant faces, is identical. We propose to introduce a coefficient allowing to weight each edge influence in the topological neighborhood according to the edge length. Thus, we introduce three functions:

- π such that for each edge *e*, $\pi(e)$ is the length of *e*,
- Π such that for each circuit $\gamma=\{o_0, o_1,.. o_n\}$, $\Pi(\gamma)=\Sigma_{i=0..n}\pi(o_i)$,
- Θ, such that for each element γ of $\Gamma_{F\cap G}$, $\Theta(\gamma)=\max\{ \Sigma_{i=0..n} \min(\pi(o_i), \pi(o_i'))$ with $(o_i)_{i=0..n}$ and $(o_i')_{i=0..n}$ / $\gamma=o_0.. o_n$ and $o_0.. o_n \sim_{Adj} o_0'.. o_n'\}$.

$\Theta(\gamma)$ can be interpreted as the maximum common weight between γ and an equivalent element in Γ_G.

Finally, we define $\sigma=\max\{\Theta(\gamma),\gamma\in\Gamma_{F\cap G}\}$.

σ is the maximum sum of edge lengths that we can extract from the boundaries of F_{ig} and F_{rg} such as the edges appear in the same order in the boundaries of F_{ig} and F_{rg}.

We calculate two ratios: $\delta_0=\sigma/\Pi(\gamma_G)$ and $\delta_1=\sigma/\Pi(\gamma_F)$. δ_0 is the ratio of inclusion of $\gamma_{F_{ig}}$ in $\gamma_{F_{rg}}$ and δ_1 is the ratio of inclusion of $\gamma_{F_{rg}}$ in $\gamma_{F_{ig}}$. δ_0 and δ_1 range in interval [0,1] according to the similarity of both weighted topological neighborhoods. $\delta_0=\delta_1=1$ means that the two topological neighborhoods $\gamma_{F_{rg}}$ and $\gamma_{F_{ig}}$ are equals. $\delta_0=1$ and $\delta_1\in]0,1[$ means that $\gamma_{F_{rg}}$ is included in $\gamma_{F_{ig}}$. $\delta_0\in]0,1[$ and $\delta_1\in]0,1[$ means that $\gamma_{F_{rg}}$ and $\gamma_{F_{ig}}$ partially overlaps. $\delta_0=\delta_1=0$ means that the two topological neighborhoods $\gamma_{F_{rg}}$ and $\gamma_{F_{ig}}$ do not overlaps.

Initial graph faces **Re-evaluated graph faces**	***$F_{2.1}$*** 1.E SL$_A$ 1.C 1.D 7 5 7 5	***$F_{2.2}$*** 1.E 1.B 1.C SL$_B$ 1 5 1 5
$F_{2.3}$ 1.E SL$_A$ 1.C 1.D 3 5 3 5	1.E SL$_A$ 1.C 1.D 3 5 3 5 $\delta_0=16/16$, $\delta_1=16/24$	1.E 1.C 1 1 $\delta_0=2/16$, $\delta_1=2/12$
$F_{2.4}$ 1.E 1.B 1.C SL$_B$ 5 5 5 5	1.E 1.C 5 5 $\delta_0=10/20$, $\delta_1=10/24$	1.E 1.B 1.C SL$_B$ 1 5 1 5 $\delta_0=12/20$, $\delta_1=12/12$

Weighted topological neighborhood of *$F_{2.2}$*.

Length of edges

Partial sub-path of circuits of faces *$F_{2.2}$* and *$F_{2.4}$* maximizing σ (here σ=12).

Table 1 : illustrates one calculation step of δ0 and δ1.

If a face needs to be matched at step *i* of the re-evaluation process, faces to be use in a same crossing are leaves of IG and RG appeared up to step *i* and having the same invariant ancestor face. This approach resolves the problem of "piece loss" introduced in section 2 and illustrated in Fig 4. Let us observe on this example the calculation of δ_0 and δ_1 at step 2 (see Table 1). It is necessary to cross two faces of IG ($F_{2.1}$, $F_{2.2}$) with two faces of RG ($F_{2.3}$, $F_{2.4}$).

Previous calculations allow evaluating in an individual way probabilities δ_0 and δ_1 with mutual inclusion of F_{rg} and F_{ig} faces and so topological matching between both faces. This very local approach does not take into account topological matching of adjacent faces. Once δ_0 and δ_1 are calculated, we have to define a method allowing to evaluate in a global way similarity between crossed faces. This method consists in handling, in an iterative way, the whole table cells in decreasing order of matching possibilities. For that, we apply the following process:

- Find a cell which is not already "handled" of which sum $\delta_0+\delta_1$ is maximum (if there exist several cells giving this maximum sum, we take any cell of them). Let us suppose that this cell corresponds to the crossing of faces F_{rg} and F_{ig}.
- Decrement edge weights for edges in $\gamma_{F_{ig}}$ and $\gamma_{F_{rg}}$ according to the weight of corresponding oriented edges in the element $o \in \Gamma_{F \cap G}$ that lead to the maximum σ; actually, a temporary weight function replaces π that makes edges appear 'shortened' since some length is no more available for further cell computing.
- For cells which are not already handled, calculate numerators σ of δ_0 and δ_1 with remaining weights.
- Mark this cell as handled
- Iterate the process until all cells are marked.

Note that handling a cell of which coefficients δ_0 and δ_1 are equal to zero does not change anything for the table. Thus, when a cell has both coefficients equal to zero, it can be considered as handled. Note also that during the handling, coefficients δ_0 and δ_1 only decrease.

Observe the result of this method on Table 2. We can see, on the second loop table, that the grayed cell is selected because it is the maximum coefficient sum. Edge coefficients (through a temporary weight function) of $\gamma_{2.3}$ are zero because every edge length has been totally used. Edge coefficients of $\gamma_{2.1}$ are not totally equal to zero because some edge length has been partially used. Coefficients of the row and the column are recalculated. The result is zero for the row corresponding to face 2.3 because there is no more face that can be used on L to identify other faces. Coefficients being equal to zero, this cell is considered as already handled (dashed cells). At the third loop, only one cell has to be handled. No computing of coefficients δ_0 and δ_1 is needed because all cells of row and column are handled.

Loop 1	$F_{2.1}$ 1.E SL$_A$ 1.C 1.D 7 5 7 5	$F_{2.2}$ 1.E 1.B 1.C SL$_B$ 1 5 1 5
$F_{2.3}$ 1.E SL$_A$ 1.C 1.D 3 5 3 5	$\delta_0 = 16/16$ $\delta_1 = 16/24$	$\delta_0 = 2/16$ $\delta_1 = 2/12$
$F_{2.4}$ 1.E 1.B 1.C SL$_B$ 5 5 5 5	$\delta_0 = 10/20$ $\delta_1 = 10/24$	$\delta_0 = 12/20$ $\delta_1 = 12/12$

Loop 2	$F_{2.1}$ 1.E SL$_A$ 1.C 1.D 4 0 4 0	$F_{2.2}$ 1.E 1.B 1.C SL$_B$ 1 5 1 5
$F_{2.3}$ 1.E SL$_A$ 1.C 1.D 0 0 0 0	$\delta_0 = 16/16$ $\delta_1 = 16/24$	$\delta_0 = 0/16$ $\delta_1 = 0/12$
$F_{2.4}$ 1.E 1.B 1.C SL$_B$ 5 5 5 5	$\delta_0 = 8/20$ $\delta_1 = 8/24$	$\delta_0 = 12/20$ $\delta_1 = 12/12$

Loop 3	$F_{2.1}$ 1.E SL$_A$ 1.C 1.D 4 0 4 0	$F_{2.2}$ 1.E 1.B 1.C SL$_B$ 0 0 0 0
$F_{2.3}$ 1.E SL$_A$ 1.C 1.D 0 0 0 0	$\delta_0 = 16/16$ $\delta_1 = 16/24$	$\delta_0 = 0/16$ $\delta_1 = 0/12$
$F_{2.4}$ 1.E 1.B 1.C SL$_B$ 4 0 4 0	$\delta_0 = 8/20$ $\delta_1 = 8/24$	$\delta_0 = 12/20$ $\delta_1 = 12/12$

Loop 4	$F_{2.1}$ 1.E SL$_A$ 1.C 1.D 0 0 0 0	$F_{2.2}$ 1.E 1.B 1.C SL$_B$ 0 0 0 0
$F_{2.3}$ 1.E SL$_A$ 1.C 1.D 0 0 0 0	$\delta_0 = 16/16$ $\delta_1 = 16/24$	$\delta_0 = 0/16$ $\delta_1 = 0/12$
$F_{2.4}$ 1.E 1.B 1.C SL$_B$ 0 0 0 0	$\delta_0 = 8/20$ $\delta_1 = 8/24$	$\delta_0 = 12/20$ $\delta_1 = 12/12$

Table 2 : Global calculations of mutual inclusion between crossed faces, at the second re-evaluation step of Fig 4

Let us observe the evolution of different re-evaluation steps in Fig 4. Initially, at the first re-evaluation step, identification between invariant entities (see section 5.1) exists and is symbolized by the dotted link between face *1.A* of IG and RG (see Fig 9).

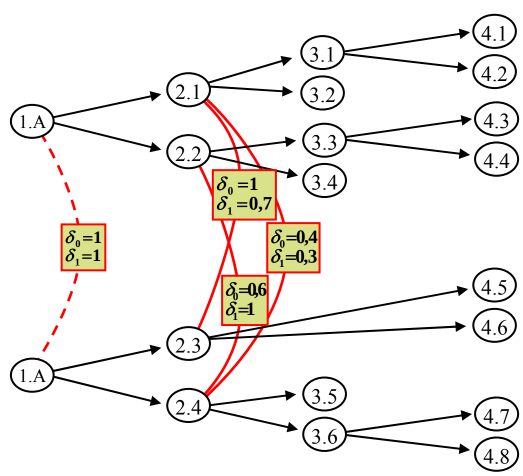


Fig 9: Cover links after re-evaluation at step 2

At the second re-evaluation step, face *1.A* is split into two new faces *2.3* and *2.4*. Face *1.A* of RG, father of both faces, is connected by a *cover link* (identification link in this particular case because it is an invariant face) to face *1.A* of IG of which leaves, appeared up to the second step, are faces *2.1* and *2.2*. Both faces *2.3* and *2.4* have to be crossed with faces *2.1* and *2.2*. Intergraph links obtained after the second re-evaluation step are represented in Fig 9 by tagged links between nodes *2.1, 2.2*and *2.3*, *2.4* containing values δ_0 and δ_1.

At the third re-evaluation step, leaves of *1.A* appeared up to the third step must be crossed (i.e. faces *3.1, 3.2, 3.3* and *3.4* must be crossed with *2.3, 3.5* and *3.6*).

Finally, the whole graph obtained after the fourth re-evaluation is shown in Fig 10.

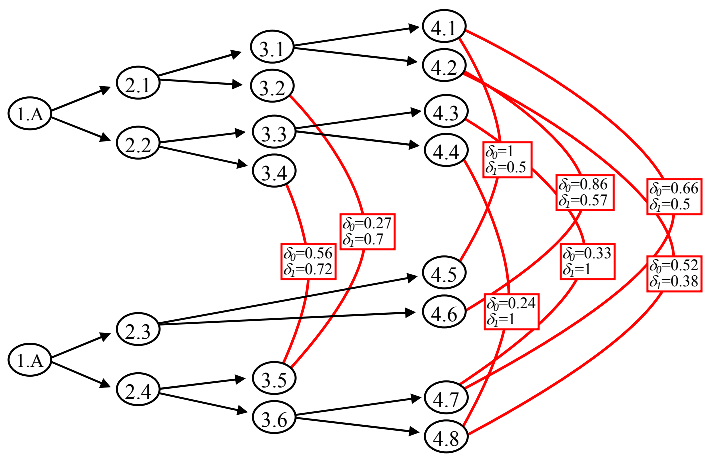


Fig 10: Construction of the initial and the re-evaluated graphs corresponding to the history of the invariant face 1.A in Fig 3 and calculation of the matching intergraph links at step 4.

5.2.2 Matching of contingent non elementary-shells

Matching a contingent non elementary-shell S^{ini} of the initial model consists in calculating a set Γ^{reev} of potentially corresponding non elementary-shells S_i^{reev} in the re-evaluated model and deciding which of these shells correspond effectively to S^{ini}. A shell corresponds potentially to S^{ini} if it belongs to the same invariant shell history and if it possesses a topological similarity with S^{ini}.

The matching of non elementary-shells can be done by using either geometry and/or the topological neighborhood of these shells or faces belonging to these shells. Calculation of matching having already been achieved for faces, it is less expensive to use the contents of non elementary-shells rather than topological neighborhood.

The proposed shell matching method breaks up into two main steps: firstly, the calculation of Γ^{reev} the set of potential corresponding shells and then the calculation of ratios measuring the probability of matching between S^{ini} and each shell S_i^{reev} belonging to Γ^{reev}. Similarly to faces, these ratios, called δ_0 and δ_1, are then stored in intergraph links between non elementary-shells.

5.2.2.1 Calculation of Γ^{reev}

If S^{ini} the initial non elementary-shell we try to match is contingent, then the calculus of Γ^{reev} requires the construction of all potential corresponding shells S_i^{reev} in the re-evaluated model. The contingent non elementary-shells are not explicitly represented in the B-rep model. Then, calculation of the non elementary-shells S_i^{reev} is performed through construction of connected components in a similar way to the one introduced in the section 4.2.2. The starting face of the incremental topological traversal of the re-evaluated geometric model is calculated through the matching intergraph links stored in the graph at face level for each face contained in the shell S^{ini}. So, a shell belongs to Γ^{reev} if at least one of its faces corresponds to the matching of a face belonging to S^{ini}.

In Fig 8, the re-evaluation of the contingent non elementary-shell <4, <1, top>, {3.2, 2.1}> generate the set Γ^{reev}={<4, <1, top>, {3.6, 1.H, 2.2}>} because there is intergraph links between faces 3.2 and 1.H and faces 2.1 and 2.2. Similarly, the non elementary-shell <4, <2, aggr-prot>, {3.2, 2.1, 2.A, 2.B} generate the set Γ^{reev}={<4, <2, aggr-prot>, {3.6, 1.H, 2.2, 2.A, 2.B}>}. The non elementary-shell <4, <1, top>, {1.F, 3.1, 1.G}> generate the set Γ^{reev}={<4, <1, top>, {1.F, 3.5}>, <4, <1, top>, {3.6, 1.H, 2.2}>} containing two shells. Indeed, the face 1.G, for example, is matched on both the faces 3.5 and 3.6, and the incremental traversal from these two faces generates two different connected shells.

5.2.2.2 Calculation of matching intergraph links

For each non elementary-shell S, we note Ψ_S={F1,..Fn} the set of faces belonging to S. If a faces F belonging to the initial graph is connected through an intergraph link to a face G belonging to the re-evaluated graph, then we note $F \approx G$ this relation. In order to quantify topological matching, we define the function $\Phi_{\Psi_S}(G)$=1 if $\exists\ F_i \in \Psi_S / G \approx F_i$, 0 otherwise.

Let Ψ_S^{ini}={F1,..Fn} and $\Psi_{S_i}^{reev}$={G1,..Gm} be the sets associated with the non elementary-shell S^{ini} of the initial graph and one non elementary-shell S_i^{reev} of Γ^{reev}. We can define two ratios: $\delta_0 = \dfrac{\sum_{k=1..n} \Phi_{\Psi_{S_i}^{reev}}(F_k)}{\left|\Psi_S^{ini}\right|}$ and $\delta_1 = \dfrac{\sum_{k=1..m} \Phi_{\Psi_S^{ini}}(G_k)}{\left|\Psi_{S_i}^{reev}\right|}$. δ_0 is the ratio of inclusion of S^{ini} in S_i^{reev} and δ_1 is the ratio of inclusion of S_i^{reev} in S^{ini}. δ_0 and δ_1 range in the interval [0,1] according to the similarity of the contents of the shells. These calculations allow evaluating in individual way probabilities δ_0 and δ_1 of mutual inclusion of S^{ini} and S_i^{reev} shells and so topological matching between both shells.

For example in Fig 8, the calculation of the matching intergraph links for the contingent shell <4, <1, top>, {3.2, 2.1}> and the set Γ^{reev}={<4, <1, top>, {3.6, 1.H, 2.2}>} gives δ_0=2/2 and δ_1=2/3.

The calculation of the matching intergraph links for the contingent shell <4, <2, aggr-prot>, {3.2, 2.1, 2.A, 2.B} and the set Γ^{reev}={<4, <2, aggr-prot>, {3.6, 1.H, 2.2, 2.A, 2.B}>} gives δ_0=4/4 and δ_1=4/5.

The calculation of the matching intergraph links for the contingent shell <4, <1, top>, {1.F, 3.1, 1.G}> and the set Γ^{reev}={<4, <1, top>, {1.F, 3.5}>, <4, <1, top>, {3.6, 1.H, 2.2}>} gives δ_0=2/3 and δ_1=2/2 for the matching between {1.F, 3.1, 1.G} and {1.F, 3.5} and δ_0=2/3 and δ_1=2/3 for the matching between {1.F, 3.1, 1.G} and {3.6, 1.H, 2.2}.

When S^{ini} is an invariant shell then the coefficients δ_0 and δ_1 between S^{ini} and the single shell S_i^{reev} belonging to Γ^{reev} are equal to 1. Indeed, the shells being invariant, the matching is perfect and then it is not necessary to achieve the previous calculus. Then, the matching of an invariant shell S^{ini} at step *t* consists of all historical leaves at step *t* of the faces belonging to S_i^{reev}. For example, the invariant shell <2, aggr-prot> at step 3 consist of faces {2.A, 2.B, 3.1, 3.2, 2.1, 1.G, 1.F} in the initial model. During the re-evaluation, the invariant shell <2, aggr-prot> of the initial graph is matched to the invariant shell <2, aggr-prot> of the re-evaluated graph with coefficients δ_0 and δ_1 equal to 1. The faces belonging to <2, aggr-prot> in the re-evaluated graph are {2.A, 2.B, 2.C, 1.F, 1.G, 1.H, 1.I} and then the historical leaves of these faces at step 3 are {2.A, 2.B, 1.F, 3.5, 3.6, 1.H, 2.2}.

5.2.3 Other entity matching

The matching of faces being robust, other entities (loops, edges, vertices, etc.) can then be named in term of faces or sets of faces. The characterization of these entities can be carried out in a way similar to the one introduced by Chen [6]. For example, an edge will be characterized by its two adjacent faces, the ordered list of the adjacent faces at its ends, as well as an orientation according to the feature orientation making it possible to remove some topological ambiguities.

6. Implementation

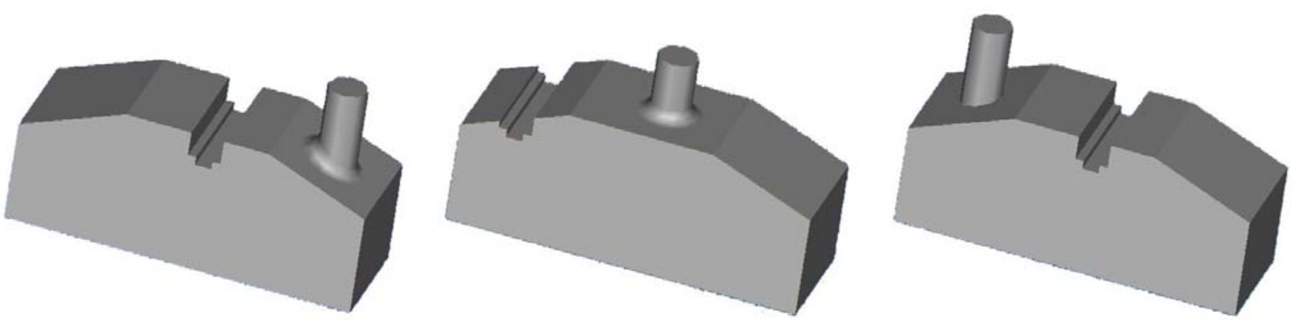
A package corresponding to the naming layer of the parametric model architecture has been implemented and integrated in a commercial CAD Software (Topsolid).

The API between the naming layer and the two others layers of the parametric model is reduced to about ten elementary functions which enable to generate a persistent name for a topological entity designated interactively by the user, to calculate the matching of a persistent name computing the intergraph links, and to construct the historical structure of faces using at each construction step a bulletin board generated by the geometric layer and containing the evolution appeared on faces.

The additional overhead in space mainly relates to the shell graph. Even if it is the price to pay in order to have the persistent naming of shells, this overhead is perceptible for important mechanical parts. However, the complexity of the graph does not grow tremendously with each operation because each operation impacts on average a limited number of faces. Moreover, the matching method for non elementary-shells avoid calculating and representing at each construction step the history of the various contingent non elementary-shells. That simplifies considerably both the time and the memory complexity of the model.

The additional overhead in time is not noticeable on a real mechanical part because on the one hand the calculus of measure of topological similarity is very limited (only faces coming from the same invariant ancestor), and on the other hand, geometrical calculus in CAD application represents a very important part of the time complexity in comparison with topological traversals of the model and the shell graph.

The objects presented in Fig 11 have been constructed using this prototype. The first object is build with fourth constructive gestures (creation of an initial bloc by sweeping of a sketch, creation of a cylindrical protrusion, creation of a slot, and then creation of a blend between the cylindrical protrusion and the right part of the top shell). In the second and third objects, the position of the slot and the protrusion are changed. The shell matching method enables to retrieve the well right part of the top shell in both case and then the re-evaluation can preserves or deletes the blend in a persistent way.



Object 1 **Object 2** **Object 3**

Fig 11 : An example of re-evaluation using persistent shell matching method

7. Conclusion

We proposed a mechanism of persistent naming associated with a hierarchical structure allowing tracing the historical evolution of easily identifiable invariants in each constructive gesture. For that, the suggested method defines a shell graph, consisting of a hierarchical structure of shells above of an historical structure of faces.

The shells characterization enables on the one hand to represent the different kinds of shells required in CAD during initial model construction and on the other hand to identify the effective shells matching to the referenced shells during model re-evaluation. A shell could be connected, overlapping (a face or a shell could belong to different super-shells), hierarchical (a shell could be a composition of shells).

The matching method for faces offers various advantages. First of all, it is a global method of topological matching in the sense that it involves two sets of faces in order to find the best possible matching for all faces. Then, it addresses the tracability loss problems. Finally, the use of weighted topological neighborhood of faces enables an efficient measurement of topological similarity.

The matching method for non elementary-shells avoid calculating and representing at each construction step the history of the various contingent non elementary-shells. That simplifies considerably both the time and the memory complexity of the model. It is based on an invariant aggregative structure and on a construction of connected components using an incremental topological traversal of the geometric model.

8. References

[1] Agbodan D., Marcheix D., Pierra G. (2000) "Persistent Naming for Parametric Models", *WSCG'2000*, pp.17-38.

[2] Agbodan D., Marcheix D., Pierra G., Thabaud C. (2003) "A topological Entity matching Technique for Geometric Parametric Model", *in proceedings of Shape Modelling'03*, Seoul, Korea, pp.235-244.

[3] Bidarra, R., Bronsvoort, W.F. (2000) Semantic feature modelling *in Computer-Aided Design* Vol. 32 pp. 201-225

[4] Bouma W., Fudos I., Hoffmann C.M., Cai J., Paige R. (1995) "Geometric constraint solver*", Computer-Aided Design*, vol. 27, n° 6, pp 487-501.

[5] Capolylas V., Chen X., Hoffman C.M. (1996) "Generic naming in generative, constraint-based design", *Computer-Aided Design*, Vol. 28, pp. 17-26.

[6] Chen X. (1995) "Representation, Evaluation and Editing of Feature-Based and Constraint-Based design.", *Ph.D. thesis*, Department of Computer Sciences, Purdue University, West Lafayette, Indiana.

[7] Hoffmann C.M., Juan R. (1993) “EREP: an editable high-level representation for geometric design and analysis”, *Technical Report CER-92-24*, Department of Computer Sciences, Purdue University, West Lafayette, Indiana.

[8] ISO 10303-224: 1999, “Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 224: Application protocol: Mechanical product definition for process planing using machining features”, ISO, Geneva, 1994.

[9] Kripac J. (1995) “A mechanism for persistently naming topological entities in history-based parametric solid models (Topological ID System)”, *Proceedings of Solid Modeling’95*, Salt Lake City, Utah USA, pp.21-30.

[10] Laakko T., Mäntylä M. (1996) “Incremental constraint modeling in a feature modeling system”, *Computer Graphics forum*, Vol.15, N°3, EUROGRAPHICS’96, Poitiers, France, pp.366-376.

[11] Marcheix D., Pierra G. (2002) " A Survey of the Persistent Naming Problem", Proceedings of Solid Modeling'02, Saarbrücken, Germany, pp 13-22.

[12] Raghothama S., Shapiro V. (1997) “Boundary Representation Variance in Parametric Solid Modeling”, *Report SAL 1997-1, Spatial Automation Laboratory*, University of Wisconsin-Madison.

[13] Rappoport A. (1997) "The generic Geometric Complex (GGC): a modeling scheme for families of decomposed pointsets". *Proceedings of Solid Modeling'97*, Atlanta.

[14] Rappoport A., Spitz S., Etzion M. (2005) "One-Dimensional Selections for Feature-Based Data Exchange", *Proceedings of Solid Modeling'05*, MIT.

[15] Shah J.J., Mäntylä M. (1995) “Parametric and feature-based CAD/CAM : Concepts, Techniques, Applications”, John Wiley and Sons Inc.

[16] Solano L., Brunet P. (1994) “Constructive Constraint-based model for parametric CAD systems”, *Computer-Aided Design*, Vol.26, N°8, pp.614-621.

[17] J. Wu, J., Zhang, T., Zhang, X., and Zhou, J. (2001) “A face based mechanism for naming, recording and retrieving topological entities”, Computer-Aided Design, Vol. 33, N° 1, pages 687-698.