A Persistent Naming System Based on Graph Transformation Rules

Marcheix David
LIAS, ENSMA
France, 86360, Futuroscope
david.marcheix@ensma.fr

Cardot Anaïs
XLIM Lab.
France, 86360, Futuroscope
anais.cardot@univ-poitiers.fr

Skapin Xavier
XLIM Lab.
France, 86360, Futuroscope
skapin@xlim.fr

Dieudonné-Glad Nadine
HeRMA Lab.
France, 86000, Poitiers
nadine.dieudonne.glad@univ-poitiers.fr

ABSTRACT

3D modeling for Archaeology requires to easily model scenes by letting users evaluate a parametric specification of archaeology-oriented gestures, then modify and reevaluate the specification to produce various restitution hypotheses. But the current modeling tools that support reevaluation mechanisms are not dedicated to Archaeology. The *Jerboa* library, based on graph transformations rules, is well suited for creating operations fitting the needs of archaeologists. But it does not any support reevaluation mechanism and especially the *persistent naming system*, that is used to identify the entities of the initial model and match them with entities of the reevaluated model. In this paper, we extend *Jerboa* with a new application-independent persistent naming model, which is more general and homogeneous than other solutions found in the literature and is the first one to handle parametric specification edition.

Keywords

Parametric Specification; Persistent Naming; Graph Transformation Rules; Generalized Maps.

1 INTRODUCTION

Digital Humanities, and 3D modeling tools in particular, have profoundly modified the discipline of archaeology in several ways. They enrich the patrimonial description and significantly improve its understanding by the public. Modeling ancient buildings in 3D usually borrows from: (1) *Computer Vision*, requiring buildings in good condition for 3D replication and/or completion [GBS14]; (2) *Geometry Modeling* based on fragmentary data, which requires the definition of several restitution hypotheses, and the availability of a tool to test these hypotheses quickly and simply. Our work is set in this latter context.

Procedural generation grammars is a commonly used process for creating several variants of the same building [HMV09], [QB15], but requires some rich information corpus information to produce grammars. Moreover, the same tool cannot be used for very different case studies with many specific features. Therefore, archaeologists usually use more "conventional" 3D tools such as *CityEngine*^{TM}or *Blender*^{TM}. Unfortunately, those tools do not comply with inherently incomplete archaeological data [Wit13], [Ver10]. In particular, they cannot easily model a display of several reconstruction hypotheses, each of them matching the observed data. The central problem of testing reconstruction hypotheses on a 3D view basis leads to limited interpretations of the Past, all the more for protohistory, for which remains are scarce.

To overcome these limitations, we use the *Graph Transformation Rules* formalism [EEPT06] through a *Java* library called *Jerboa* [JER], and designed to assist the development of application-specific modelers. Rule-based languages form a standard approach for geometric modeling, from plant growth with the seminal L-Systems [PH89], to numerous applications such as buildings [HMV09]. Unlike most approaches, Jerboa is independent from any application domain and avoids any hand-coding of operations, except rule writing. It allows rapid development of new operations to automatically check the consistency of different objects properties. All applications developed with Jerboa library share the same topological model called *Generalized maps* (or "*G-Maps*") [Lie91], describing a particular class of labeled graphs.

But Jerboa does not support the rapid production of restitution hypotheses, i.e. the mechanisms of *reevaluation* inherent to parametric systems used in CAD domain. Reevaluation allows to modify any part of an object construction history and to replay this history to produce a new result. A *parametric system* is a two-fold data structure composed of a geometric model defining the explicit geometry of the designed object (called *parametric object*), and a mechanism able to reevaluate it when some parameters are changed (called *parametric specification*)[Kri95]. The geometric model is usually a topological-based one. Most current parametric modeling systems are known as "history-based" because the parametric specification may be regarded as a history of modeling functions (or *constructive gestures*), which are attached via their parameters to topological entities defined in previous states of the model. Such an approach requires to define how to ensure the persistence of the referenced entities and to avoid systems failure during the reevaluation phase when various kinds of topological changes occur. This issue, known as *persistent naming*, should enable both unambiguous identification of initial model entities and consistent matching between initial and reevaluated model entities.

Persistent naming is a much-debated problem in CAD domain [Kri95] [Bab10][XJHY16], but has never been investigated in conjunction with graph transformation rules. Our approach enables: (1) to extend the persistent naming scope to modeling systems based on such graph transformation rules; (2) to extend Jerboa by including the working mechanisms of parametric systems. We address naming problems through a very precise characterization of the basic elements forming the model and propose a naming mechanism both general (independent of the model dimension) and homogeneous (independent of the entity dimension), for which only the entities actually used in the parametric specification are followed. Unlike others methods, this follow-up is performed only during reevaluation (and not also during initial evaluation), in order to optimize both time and memory consuming. Moreover, beyond static reevaluation with only parameter modifications, we explore how to carry out *parametric specification edition* (i.e. adding or deleting constructive gestures).

In Section 2, we present the G-maps model, the graph transformation rules and our contribution to persistent naming. In Section 3, we detail the different parts of our works, from the persistent naming system to the complete edition of a parametric specification using bulletin boards and history records. We conclude in Section 4 and propose some perspectives.

2 MAIN CONCEPTS

2.1 Generalized maps

As stated above, Jerboa is based on *G-Maps*, which intuitively represent the decomposition of n-dimensional objects according to the successive dimensions of their boundaries, the different parts being linked by relationships noted α_i. For example, the 2D object in Figure 1(a) is split into faces linked by α_2 (blue line, Figure 1(b)); face sides are split into edges linked by α_1 (red lines, Figure 1(c)); and ends of edges are linked by α_0 (black lines, Figure 1(d)). A G-Map is therefore a graph whose nodes are called *darts* (represented as green disks in Figure 1(e)) and arcs represent various α_i. Entities are described as specific set of darts linked by dimension-specific α_i: vertices (dim 0), edges (dim 1) and faces (dim 2) are respectively defined as set of darts linked by [α_1, α_2], [α_0, α_2] and [α_0, α_1].

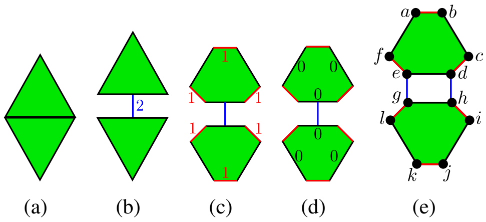

Figure 1: Modeling 2D objects using G-Maps.

We call *orbit type* the set $\{\alpha_i, ..., \alpha_n\}$ describing any entity, denoted as $\langle i...n \rangle$: orbit type "Vertex" (resp. "Edge", "Face") shown in Figure 1 is thus denoted as $\langle 12 \rangle$ (resp. $\langle 02 \rangle$, $\langle 01 \rangle$). We call *orbit* the association of a dart with an orbit type to designate a specific entity. For example, on Figure 1(e), darts {a,b,c,d,e,f} represent a face, {f,e,g,l} a vertex, and {h,i}, the restricted corner of a face. Entities used in our parametric specifications are expressed as orbits. They can be fully characterized by their type and a selection of their darts.

2.2 Graph transformation rules

Jerboa is based on topological rules of graph transformation [BALB14]. Each modeling operation is formally defined as a rule applied to a G-Map. Jerboa ensures by design that the topological consistency of the G-Map is maintained after each rule application.

Rules are made up of two parts separated by a left-to-right arrow. The left (resp. right) part, which describes the pattern to be filtered (resp. the rewritten pattern), represents the model before (resp. after) application. Patterns are defined by the orbit types of the rule nodes. For example, the Vertex Insertion rule is illustrated in Figure 2. The left node n_0 carries the orbit type $\langle 02 \rangle$, and thus filters the edge associated with this node.

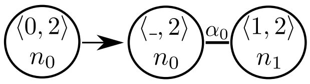

Figure 2: Vertex Insertion rule.

2.3 Persistent naming

Our method of persistent naming is grounded on both G-Maps and rewriting rules. Persistent naming allows to characterize the topological entities in a sufficiently robust way during the initial construction. Parameters of parametric specification operations are often topological references, so this mechanism is essential to produce a valid reevaluation.

Various naming methods have been proposed to try and solve this problem in a full and homogeneous way. Most methods ([Kri95][WN05] [XJHY16]) use faces as references to name all other entities, since in 3D, each entity can be characterized by an intersection of faces and some additional geometric information. However, these naming algorithms are *not generalizable in dimension n*. Moreover, the naming mechanism of any entity depends on its dimension, so *the naming is not truly homogeneous*. In addition, even though the design of persistent naming is well depicted in the literature, the way it can be used for reevaluation is not always precisely defined. Furthermore and despite memory overload, it is usually necessary to trace the evolution of many entities during the initial construction, in order to perform the match between entities when reevaluating, even though many of them will not be used.

Finally, based on the review of existing literature, no method explains how to deal with parametric specification *editing*, i.e. adding or deleting gestures between the initial evaluation and the reevaluation. We describe in the next section the various mechanisms that address these limitations.

3 REEVALUATION MECHANISMS

3.1 Parametric specification and edition

To reevaluate a sequence of constructive gestures, we record them in the form of a parametric specification beforehand. Each gesture corresponds to the call of a graph transformation rule as defined in Section 2.2. Let us consider the sequence of gestures performed in the initial specification shown in Figure 3. The specification cannot be limited to the simple recording of rule calls (physical id. of darts being inherently unstable from one reevaluation to another, they cannot be used directly). Darts should therefore be labeled persistently. The use of rules makes it possible, both in the initial evaluation and the reevaluation, to assign each dart a *Persistent Id*, denoted as PI_a, PI_b and so on.

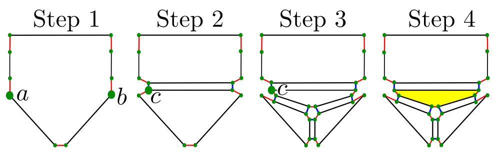

Figure 3: Initial specification.

Rules are defined for any filtered orbits, but only specific orbits are used by gestures as parameter entities. To identify each entity, we define their *Persistent Names*[1] *(PN)*, composed of a set of Persistent Ids to keep track of all gestures that have impacted that entity (see section 3.2.2). More precisely, $PN = \{PI\}.\langle o\rangle$, where $\{PI\}$ is a set of Persistent Ids of the representative darts of the orbit, and $\langle o\rangle$ is the orbit type of the entity.

The parametric specification shown in Figure 3 is: Step 1 : 1-PentagonCreation; Step 2 : 2-EdgeInsertion($PN1$, $PN2$); Step 3 : 3-Triangulation($PN3$); Step 4 : 4-Coloring($PN4$, Yellow), where $PN1$, …, $PN4$ are respectively the Persistent Names containing the Persistent Ids detailed in Table 1.

PN	PI	O. type	PN	PI	O. type
$PN1$	$\{PI_a\}$	$\langle 1\rangle$	$PN3$	$\{PI_c\}$	$\langle 01\rangle$
$PN2$	$\{PI_b\}$	$\langle 1\rangle$	$PN4$	$\{PI_c\}$	$\langle 01\rangle$

Table 1: Persistent Ids and orbit types related to gesture parameters of the initial specification.

To illustrate the behaviour of our persistent naming mechanism, we modify the initial specification by adding a Vertex Insertion operation (denoted as A-Step 1) between Step 2 and Step 3 (Figure 4). The reevaluation proceeds as follows.

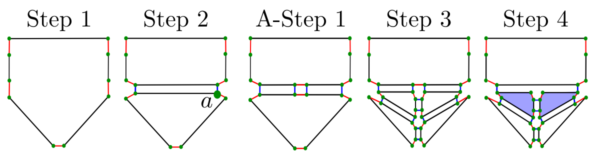

Figure 4: Specification reevaluation.

(1) 1-PentagonCreation is reevaluated the same way as in the initial evaluation (the related rule is applied). (2) 2-EdgeInsertion($PN1$, $PN2$). $PN1$ and $PN2$ will be used to find darts automatically, in order to call the corresponding rule. (3) Add-1-VertexInsertion($a.\langle 02\rangle$) adds a vertex on edge $a.\langle 02\rangle$ directly designated by the user during the reevaluation process. (4) 3-Triangulation($PN3$) is not modified. Using $PN3$, we find a dart representing the face and apply the related rule. (5) 4-Coloring($PN4$, Blue) is also reevaluated, finding the darts corresponding to $PN4$ but with a different color parameter. Due to Add-1-VertexInsertion, the initial face has been split, so the new coloring is applied to both sub-faces.

As shown above, determining the types of edition undergone by gestures is mandatory to apply the reevaluation. But to achieve the matching of entities, it is also required to determine how the Persistent Names of referenced orbits have evolved.

3.2 Orbit evolution

We consider the evolution of orbits for both initial evaluation and reevaluation. First, we define the different types of orbital evolutions that may happen (Section 3.2.1). Then, to match evaluation and reevaluation entities, we detail the structures of related Ids and Persistent Names (Section 3.2.2). Finally, we propose a structure allowing to follow the entities during the evaluation and a tree structure allowing to report the matching during the reevaluation (Sections 3.2.3 to 3.2.5).

3.2.1 Evolution types

We define the following types of orbit evolution, some of which are shown in Figures 3 and 4. (a) *Creation*: creates a new orbit. (b) *Deletion*: removes an orbit, so no constructive gesture can use it anymore. (c) *Fusion*: merges several orbits. (d) *Modification*: modifies the orbit without any splitting or merging. (e) *NoEffect*: does not affect the orbit. (f) *Split*: splits the orbit.

3.2.2 Persistent naming

The Persistent Id (*PI*) of a dart is set at the time of dart creation, and then modified each time the dart is rewritten by rules. Each *PI* consists of the various operation numbers and rule nodes that have created or rewritten the related dart. For example, dart c of the initial set (Figure 3) is created by instantiating node n_2 of the rule defining 2-EdgeInsertion (Figure 5): "$2 - n_2$" is thus a part of PI_c. But n_2 itself is the rewriting of node n_0 located on the left side of the rule, which is associated with dart a in the initial set. Since a has been created by instantiating node n_7 of the rule defining 1-PentagonCreation, PI_c is defined as $\{1 - n_7; 2 - n_2\}$.

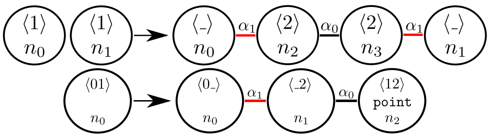

Figure 5: Transformation rules. Top: Edge insertion. Bottom: Triangulation.

The *PN* (see Section 3.1) is used as a parameter of the operations. Thus, 3-Triangulation, which tessellates the face adjacent to dart c, has face Persistent Name $PN3 = \{\{1 - n_7; 2 - n_2\}\}.\langle 01 \rangle$ as topological parameter. 4-Coloring is also applied to the face adjacent to c. However, $PN3$ is different from $PN4$ because the face (and therefore c) has been affected by triangulation: $PN4 = \{\{1 - n_7; 2 - n_2; 3 - n_0\}\}.\langle 01 \rangle$.

3.2.3 Rule bulletin boards

Following orbit evolution over several steps of the specification requires to follow evolution depending on each gesture. We use structures called *bulletin boards* for that purpose. Bulletin boards are essential to any monitoring system, but have been very little detailed in the literature.

Our approach is rule-specific: a bulletin board is generated when the user creates a rule to account for the different types of evolution (Section 3.2.1). Figure 6 shows the bulletin board for Vertex Insertion operation. There is one box per orbit type. Inside each box, we describe the evolution types for the rewritten nodes. Let $\langle x \rangle$ be an orbit type: we gather the nodes of the right side of the rule, whose rewriting instantiates darts belonging to the same $\langle x \rangle$, then we search for the left-side nodes which have rewritten these darts, and for which orbit type. A tree is then created for each set: the root contains the nodes selected on the right side, and the leaves contain left-side nodes and the related orbit. The joining arc is labeled with the type of evolution carried out.

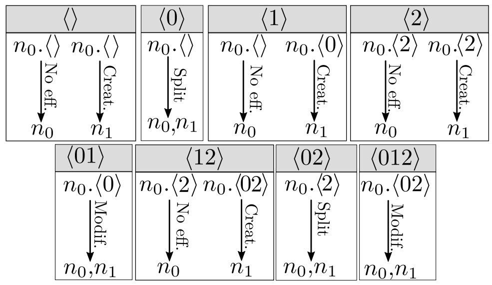

Figure 6: Vertex Insertion bulletin board.

As an example, consider the orbit type $\langle 12 \rangle$ in the bulletin board displayed in Figure 6. Figure 7 focuses on vertices (orbit type $\langle 12 \rangle$): two vertices are composed of darts instantiated by node n_0 in the right side of the rule whereas the central vertex is made of darts instantiated by node n_1. The two vertices are composed of all darts instantiated from $n_0.\langle 2 \rangle$ on the left side of the rule. The type of evolution of these vertices is no effect, because we simply have the same darts in the vertex before and after the rule is applied. A tree is thus created with root labeled "$n_0.\langle 2 \rangle$" and leaf labeled "n_0", linked by the "No Eff." arc. The central vertex is composed of all darts instantiated from $n_0.\langle 02 \rangle$ on the left side of the rule. This vertex did not exist before the rule is applied. A second tree is thus created, with root labeled "$n_0.\langle 02 \rangle$" and leaf labeled "n_1", linked by the "Creation" arc.

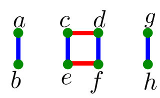

Figure 7: Topological view of edge vertices.

3.2.4 History Record

Bulletin boards are completed by *history records* to process the whole specification. History records analyze the successive bulletin boards of the rules that have impacted any dart. One carries out as many history records as there are *PI*. Let $PN = \{PI_a, PI_b, \ldots\}.\langle x \rangle$ be a Persistent Name. Let $PI_b = \{1 - n_i; \ldots; k - n_j\}$ be the Persistent Id of dart b. To create the history record of PI_b, we scan its contents in reverse order (from the most recent to the oldest). Therefore, we first consider $\langle x \rangle$ and $k - n_j$ (the last rewriting of dart b by the node n_j of the related rule set at step k). In the bulletin board of this rule, we retrieve the box corresponding to $\langle x \rangle$ and we select the (unique) tree whose child contains n_j. This process is then repeated by going back up each operation constituting PI_b, knowing that it retrieves, for the operation $(k-1)$, the box of the bulletin board corresponding to the orbit indicated at the root of the tree used for operation k.

To illustrate this point, let us create the history record for 4-Coloring applied to $PN4$ (see Figures 3 and 4), that has $\{PI_c\}$ as Persistent Id (see Table 1). The result is shown in Figure 8, with green or red arrows labeling the 6-step process. Before applying 4-Coloring, $PI_c = \{1 - n_7; 2 - n_2; 3 - n_0\}$ and $PN4 = \{PI_c\}.\langle 01 \rangle$, meaning that 4-Coloring is to be applied to orbit $\langle 01 \rangle$ (see the bottom of Figure8(a)). Assume the last element of PI_c (i.e. $3 - n_0$, that is 3-Triangulation applied to n_0) has been initially recovered. Step 1: we look at orbit type $\langle 01 \rangle$ in the Triangulation bulletin board, that is the last rule having impacted c before coloring. At this stage, c is rewritten by node n_0. Step 2: Figure 6 shows that, for orbit type $\langle 01 \rangle$, n_0 results from a split of $n_0.\langle 0 \rangle$. The related excerpt of the Triangulation bulletin board is shown in Figure 8(a).

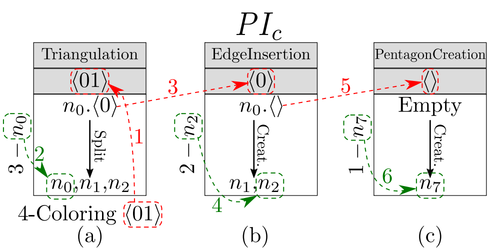

Figure 8: History record of $PN4$.

Step 3: using this orbit type $\langle 0 \rangle$ as an index in the bulletin board of the previous gesture recorded (i.e. 2-EdgeInsertion), we search among the trees related to this entry, the one which contains n_2, for the corresponding identifier in PI_c is $2 - n2$ (Step 4). We find a tree with root $n0.\langle\rangle$ (Figure 8(b)). We repeat the process once again: at Step 5, we go through the bulletin board associated with the previous recorded gesture (1-PentagonCreation). Using the orbit type $\langle\rangle$ as an entry, we search for the related tree which contains n_7, since the corresponding identifier $1 - n7$ (Step 6). The root of this tree has `Empty` as root (see Figure 8(c)), meaning that there is no previous gesture.

The history record of every Persistent Name is carried out in a similar way. As an example, Figures 9 show the history record related to $PN3$.

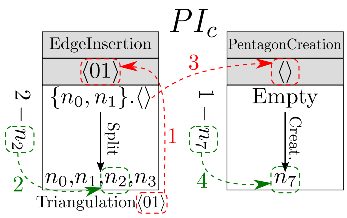

Figure 9: History records of $PN3$.

3.2.5 *Entity matching*

Performing reevaluation requires to match entities between both evaluation and reevaluation specifications. For each history record, a *matching tree* is built, with a Persistent Id as root and orbits as leaves. A matching tree allows to determine which darts of the reevaluation will be used for each orbit designated in the initial set.

For each constructive operation called during reevaluation, we focus on the type of edition which has impacted it. We refer to gestures shown in Figure 4 to describe various scenarios. Considering any gesture already present in the initial evaluation (e.g. gestures 1, 2, 3, 4), matching trees are updated in order to reevaluate this gesture. In case of adding a gesture (e.g., Add-1-VertexInsertion), the bulletin board of the related rule is used to update the matching trees according to the orbits impacted by this addition. In case of deletion, the impacted tree branches are not updated.

We now detail step by step this reevaluation for $PN3$ and $PN4$, as $PN1$ and $PN2$, which are used as parameters of edge insertion gestures, do not involve any particular issue during reevaluation: they use Persistent Ids which have been present since the beginning of the specification and have been impacted by only one gesture.

1-PentagonCreation reevaluation
Since this gesture has no parameter, it is reevaluated in the same way as the initial evaluation. Since the matching trees of $PN1$ to $PN4$ are all impacted by this gesture, they are updated. Figure 10 shows the model after applying the rule, and the impact on the matching trees of $PN3$ and $PN4$. History records shown in Figures 8 and 9 are scanned, one gesture after another, to match darts and orbits in the reevaluated model. Consider $PN3$ for instance: the history record of PI_c indicates that to process 1-PentagonCreation, one must find the newly created orbit type $\langle\rangle$, associated with the instance of node $n7$. A branch of the matching tree is thus created, related to the orbit found in the reevaluated model (a' is the dart instanciated by $n7$). Similarly, one matching tree is generated for $PN4$ using the history record in Figure 8(c).

2-EdgeInsertion reevaluation
Since this operation takes both $PN1$ and $PN2$ as parameters, we use the orbits found in their matching trees.

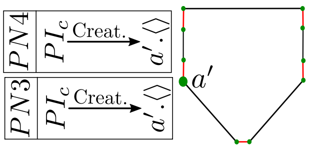

Figure 10: Matching trees and model after the reevaluation of 1-PentagonCreation.

Matching trees of *PN*3 and *PN*4 are updated as in the previous step (as shown in Figures 8 and 9, their respective history record contains the operation prefix "2−"). Figure 11 shows the model after applying the rule, and the impact on matching trees.

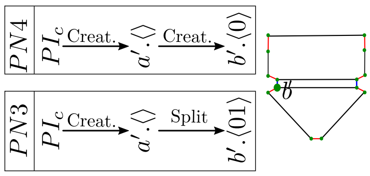

Figure 11: After 2-EdgeInsertion.

Add-1**-VertexInsertion addition**

This new insertion gesture (relatively to the initial evaluation) requires to trace its impact on orbits currently traced with matching trees.

To determine which parts of the bulletin board related to this rule are relevant, we examine the current leaves of matching trees of *PN*3 and *PN*4 (Figure 11): $b'.\langle 01\rangle$ and $b'.\langle 0\rangle$, resp. The leaves are impacted by this gesture, so we use the trees of the bulletin board of the rule which have $\langle 01\rangle$ and $\langle 0\rangle$ as roots to trace this impact. These trees are displayed in Figure 6. Orbit type $\langle 01\rangle$ undergoes a modification impacting $n0$ and $n1$. The instantiation of any of those nodes can be chosen (here we keep b', i.e. the respective instances of $n0$). Regarding $\langle 0\rangle$, there is a split impacting $n0$ and $n1$. Once again, we choose the instantiation of any node, but since it is an added split, we must trace one dart for each resulting half-edge (we keep both b' and c', i.e. the instance of $n0$ for each half-edge). Matching trees are updated accordingly, as shown in Figure 12.

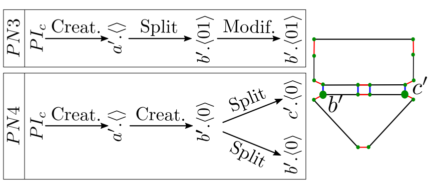

Figure 12: After Add-1-VertexInsertion.

3**-Triangulation reevaluation**

This gesture is reevaluated on *PN*3. Since *PN*3 is no longer used afterwards, its matching tree can now be removed. Only *PN*4 is impacted by 3-Triangulation (see Figure 8(a)). The result of this reevaluation is displayed in Figure 13.

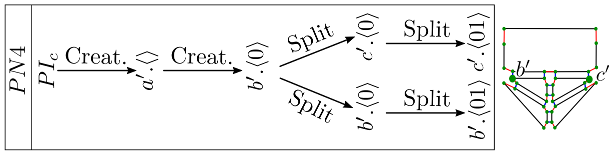

Figure 13: After 3-Triangulation.

4**-Coloring reevaluation**

Since the color parameter has been modified during reevaluation, 4-Coloring sets the color of the face designated by *PN*4 to blue. Its matching tree indicates that the orbit used in the initial evaluation now corresponds to both $b'.\langle 01\rangle$ and $c'.\langle 01\rangle$ (see Figure 13). Each face is therefore colored. Since *PN*4 is no longer used afterwards, its matching tree is removed.

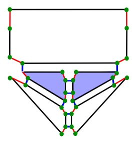

Figure 14: After 4-Coloring.

The final result is shown in Figure 14.

4 CONCLUSION

We propose a new persistent naming system and an entity matching algorithm combining the strong points of graph transformation rules and G-Maps to create and reevaluate new models using parametric specification. Those tools lay the foundation to develop 3D modeling operations dedicated to specific domains such as Archaeology.

Our approach specifically addresses the issue of naming in the context of parametric specification edition (adding and deleting gestures). The naming mechanisms make it possible to name all types of entities in an homogeneous and general way, whatever the dimension of the model. We define unambiguously *Persistent Identifiers* of darts and *Persistent Names* of entities, using the information returned by transformation rules. We follow the evolution of a limited number of entities during both evaluation and reevaluation, in order to achieve matching. Only entities that are actually referenced in the parametric specification are traced, and only during the reevaluation phase. This allows us to hope for space and time savings, but a comparative study will have to be carried out in future works.

To follow entity evolution after applying a gesture, we use the bulletin board associated with the rule defining this gesture. At this time, rule bulletin boards have to be designed by the (human) rule designer; it would be interesting to generate them automatically. We also intend to study the effect of changing the order of operations in the parametric specification, since this feature has never been proposed in the literature.

Finally, the full integration of the archaeological dimension into our works will require to study *spatio-temporal reevaluation*, i.e. a reevaluation that, with parameters such as dates, would or would not reevaluate some gestures, in order to give an account of the state of a building through the ages.

5 REFERENCES

[Bab10] Baba-Ali, M. Système de nomination hiérarchique pour les systèmes paramétriques. PhD thesis, 2010. http://nuxeo.edel.univ-poitiers.fr/nuxeo/site/esupversions/16648b94-bfdc-48c0-8ed9-a8d16d5c336c.

[BALB14] Belhaouari, H., Arnould, A., Le Gall, P., and Bellet, T. Jerboa: A Graph Transformation Library for Topology-Based Geometric Modeling, in Int. Conf. on Graph Transformation (ICGT), pp. 269–284, 2014.

[EEPT06] Ehrig, H., Ehrig, K., Prange, U. and Taentzer, G. Fundamentals of Algebraic Graph Transformation, Monographs in Theoretical Computer Science, An EATCS Series Springer, 2006.

[GBS14] Gomes L., Bellon O. P. R., Silva L. 3D reconstruction methods for digital preservation of cultural heritage: A survey. Pattern Recognition Letters 50 (Dec.), pp. 3–14, 2014.

[HMV09] Haegler S., Müller P., Van Gool L. Procedural modeling for digital cultural heritage. Journal on Image and Video Processing - Special issue on image and video processing for cultural heritage, 2009 (Feb).

[JER] http://xlim-sic.labo.univ-poitiers.fr/jerboa/

[Kri95] Kripac, J. A mechanism for persistently naming topological entities in history based parametric solid models. Proc. of the 3rd ACM symposium on Solid Modeling and Applications (SMA'95), pp. 21–30, 1995.

[Lie91] Lienhardt, P. Topological models for boundary representation: a comparison with n-dimensional generalized maps. Computer-Aided Design, 23:1, 1991.

[WN05] Wang, Y. and Nnaji, B.O. Geometry-based semantic id for persistent and interoperable reference in feature-based parametric modeling. Computer Aided Design, vol. 37, pp. 1080–1093, 2005.

[PH89] Prusinkiewicz, P. and Hanan, J. Lindenmayer Systems, Fractals, and Plants, 1989.

[PLH90] Prusinkiewicz, P., Lindenmayer, A. and Hanan, J. The algorithmic beauty of plants, Virtual laboratory, Springer-Verlag, 1990.

[QB15] Quattrini R. and Baleani E. Theoretical background and historical analysis for 3D reconstruction model. Villa Thiene at Cicogna. Journal of Cultural Heritage 16, pp. 119-125, 2015.

[Ver10] Vergnieux R. L'usage scientifique des modéles 3D en archèologie. De la validation Ã la simulation . Proc. of ARQUEOLÓGICA 2.0 Symposium, in Issue 3 of the Int. Journal Virtual Archaeology Review (VAR), 2010.

[Wit13] Wittur J. Computer-Generated 3D-Visualisations in archaeology. Between added value and deception. British Archaeological Reports (BAR) Int. Series, 2013.

[XJHY16] Xue-Yao G., Jia-Qi L, Hao G. and Yun-Feng G. Name and maintain topological faces in rotating and scanning features. Int. Journal of Grid and Distributed Computing, 9:3, pp. 21–26, 2016.