REPRESENTATION PARADIGMS FOR MASONRY MODULATION IN BIM TOOLS

Ari MONTEIRO
MSc. Candidate, University of São Paulo,
| ari.monteiro@poli.usp.br |
| http://lattes.cnpq.br/0929461339415273 |

Rita Cristina FERREIRA
DWG arquitetura e sistemas s/c, MSC.
| rita@dwg.arq.br |
| http://lattes.cnpq.br/8816822206656848 |

Eduardo Toledo SANTOS
Ph.D., Professor at the University of São Paulo
| eduardo.toledo@poli.usp.br |
| http://lattes.cnpq.br/8615127367466231 |

ABSTRACT

New CAD tools which support the BIM (Building Information Modeling) concept are based on 3D modeling of buildings by instancing objects from component families. Some of these objects have detailed geometry while others are restricted to their outer boundaries. For example, in the "wall object" this representation usually is limited to its external faces, and a list of layers is used to represent its internal composition (core and finishes). This representation makes the file light, favoring the application performance. However, in designs that require a higher detail level than this solution supports, like the Masonry Design for Production (MDP), a complete 3D representation of wall components is important. In this context, a question arises: how to represent the wall elements in a way that fulfills the MDP requirements and, at the same time, do not degrade the handling performance of the BIM model? This work presents some approaches to representing masonry modulation: an explicit one (object families) and others where the representation is implicit (array / generative modeling). A comparative analysis is presented, highlighting their pros and cons like implementation complexity, ease of use and performance impact on the application. The feasibility of each method was studied on Autodesk Revit® Architecture 2009 / 2010 considering generation of 2D views and quantity take off tasks, as well as memory and CPU consumption for each approach, enabling decision making at the implementation level.

Keywords: Masonry Design for Production, parameterization, BIM.

1. INTRODUCTION

The AEC objects available in BIM tools have varying levels of detail. Some of these objects have very a detailed geometry, while others are restricted to their volume, without further details of its composition.

An example is the *wall* object whose graphical representation is limited to its external dimensions. The composition of the wall is represented as a list of layers that define the characteristics of the core and of any coatings.

This standard representation results in lighter files, which favors the application performance. However, to meet the requirements of designs for production like those of the Masonry Design for Production - MDP, a greater level of detail than that offered in BIM tools available on the market today is required.

The requirement for a more complete 3D representation of these elements in MDP comes from the intense coordination activity between subsystems present in this kind of design and the need to plan the composition of courses, defining the sequences of blocks and joints in each one. Unfortunately, increasing the level of detail of 3D objects can lead to a decrease in the application performance since even a medium-sized wall is composed of hundreds of blocks.

According to Eastman (2006), the definition of a building model at the construction level, i.e. with a high level of detail, is a complicated undertaking that requires the definition and management of millions of objects.

Considering that context, this study aims to answer the following question: how to represent the wall components in order to meet the requirements of MDP and, at the same time, not degrading the handling performance of the BIM model?

To this end, the requirements for representing the elements of a wall in the context of MDP (sections 2 and 3) and two alternative representations for these elements (section 4) were examined. In section 5 is proposed using a shape grammar to assist the implementation of one of these alternative representations.

Application experiments were performed (section 6) for the explicit representation alternatives proposals using Revit® Architecture (versions 2009 and 2010) which is

a BIM product provided by Autodesk Inc. (AUTODESK, 2008a). After these experiments, the results were analyzed (section 7) with the objective of evaluating (section 8) which of these alternatives is the most appropriate for MDP.

2. MASONRY DESIGN

The MDP started in Brazil in the late 1980's. One of the initiatives was from a research project from EPUSP (Escola Politécnica da Universidade de São Paulo) in partnership with Encol (a construction company) (SILVA, 2003, p. 42-46).

The focus of MDP is to streamline the production processes and coordination of subsystems which masonry interfaces, such as structural, electrical and plumbing, coatings and other subsystems.

The nature of this kind of design leads the designer to check the interference between the various systems that make up a building, and demands the production of an accurate documentation for execution.

Masonry consists in laying of stones, bricks, blocks or other components together with or without mortar (CORONA, LEMOS, 1972, p. 37), including several other elements such as steel reinforcement, grouting, lintels, sills, etc. (CHING, 1999).

According to Silva (2003, p. 96), the basic elements of a masonry wall are the masonry units (blocks) and mortar joints. In this article, we identified that these elements can be grouped into three specific categories:

- Basic elements: blocks, stones, bricks;
- Bonding elements: joints (which can be filled or not with bonding material);
- Structuring elements: lintels, sills, masonry reinforcement meshes, steel reinforcement bars, etc.

The basic elements represent the majority of the composition of the wall and these parts are connected by joints. The function of the joints is to provide stability to the basic elements and to guarantee performance factors, such as acoustic and thermal insulation, waterproofing, etc. The function of the structuring elements is to provide adequate performance to the wall in the connections with other

subsystems and/or components such as doors, windows, beams, slabs, columns, other walls, etc.

The mortar joints can be classified according to their position and role in the modulation: a) bed joints (base joint, wall-slab joint, and intermediate); and b) head joints (dry and filled).

Silva (2003, p. 92) enumerates a number of Brazilian technical standards (NBR-5731/82, NBR-5706/97, NBR-10837/89, among others) used as the basis for MDP, but also states that the practices are a blending of these standards with the experience of the designers in this area.

3. RULES FOR MASONRY MODULATION

Masonry modulation (Figure 1) is a complex activity involving several rules and design variables. The masonry modulation process in MDP is divided into three main activities. They are: a) horizontal modulation; b) wall bonding; and c) vertical modulation.

Horizontal modulation consists in the optimized distribution of block family modules along the length of the wall. This process initially aims to generate the first two courses of masonry modulation.

Wall bonding is an activity that defines how the walls are connected. In ABCI (1990), it may be seen various methods of bonding, such as bonding by use of masonry reinforcement meshes and by wall interlocking. The vertical modulation consists in replicating the horizontal modulation pattern in the wall height direction.

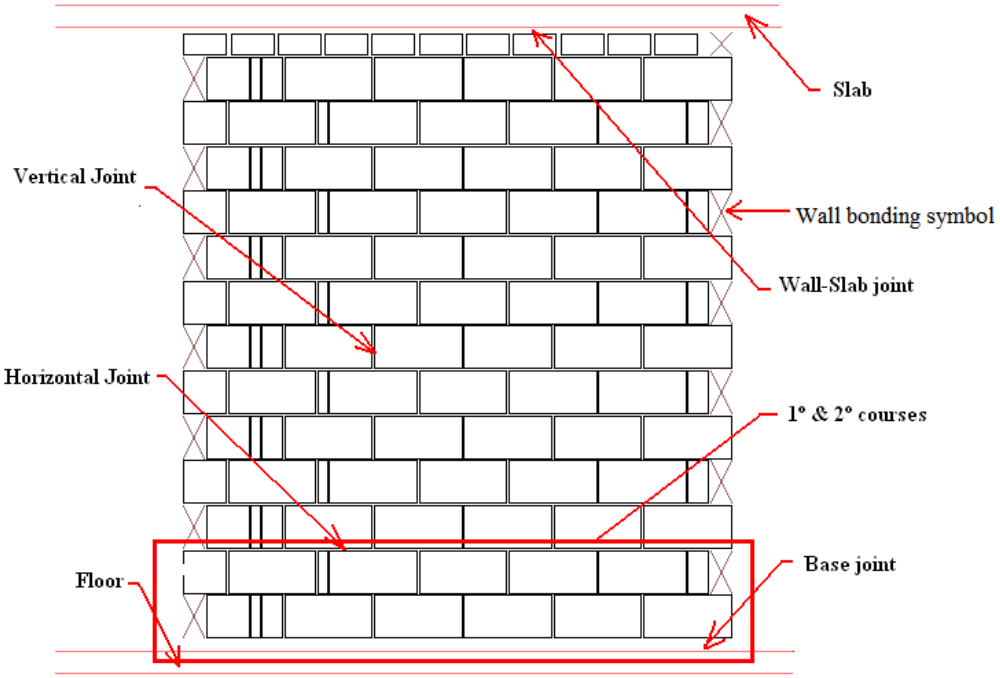

Figure 1 – Basic elements of masonry modulation.

In all these cases, the designer should pay attention to the resolution of any interference among the masonry and other subsystems, guided by the following basic rules:

Horizontal modulation

- One course can start with any module available in the family of blocks; however, it is more rational to start with full blocks;
- The vertical joints can be of two types: dry or filled;
- In a single course, there can be both dry and filled vertical joints;
- Regardless of the type of vertical joint chosen, the first two joints at both ends of the course must be filled;
- Dry joints may have thickness ranging from 0.3 cm to 0.7 cm;
- Filled joints may have thickness ranging from 0.8 cm to 1.2 cm;
- Begin the calculations setting the thickness for dry joints to 0.5 cm and 1.0 cm for filled joints, in order to avoid the use of compensation parts or fillers;

- It should be avoided that two consecutive vertical joints stay aligned, that is, blocks must be staggered;
- If the joints calculation generates aligned head joints and/or residues smaller than the smallest module available in the block family, one must redistribute this residue in the thicknesses of the joints in the course;
- If, after the implementation of the above rule, an optimal solution could not be found, compensation parts and/or fillers should be used. An alternative to this rule is to redefine the tolerance used for each type of joint and to recalculate the course.

Wall bonding

- In bonding by wall interlocking, a wall gets into the other, alternating blocks at the ends of the courses;
- For the bonding by masonry reinforcement meshes, a wall is attached to another wall, orthogonal to it, with a 1.0 cm vertical joint and, for every two courses starting from the 2nd course, masonry reinforcement meshes are placed and sized according to the wall thickness.

Vertical modulation

- The horizontal joints can have thickness ranging from 0.8 to 1.2 cm, but ideally a 1.0cm thickness is adopted;
- The base joints and wall-slab joints should have thicknesses ranging from 2.0 to 4.0 cm;
- For the initial vertical modulation calculation, a thickness of 1.0 cm for bed joints and 3.0 cm for base and wall-slab joints are used avoiding the use of compensation parts and/or fillers, if possible;
- One should avoid the use of compensation parts and/or fillers in the last (top) course;
- If the vertical modulation calculation generates residues smaller than the smallest module available in the family of blocks, one must redistribute this residue in the joints of the course;

- If, after the implementation of the above rule, an optimal solution could not be found, one must use compensation parts and/or fillers. An alternative to this rule is the redefinition of the tolerance used for each type of joint and recalculation of the course.

4. ALTERNATIVES FOR REPRESENTATION

Two alternatives were proposed to represent the elements of MDP: one explicit and another implicit. In explicit representation, all components are modeled using the object family concept available in Revit® software, for example.

On the other hand, in the implicit representation these same elements are modeled using generative modeling techniques. Generative modeling is a procedural modeling technique which uses a set of rules for creating 3D models. Through these rules, algorithms can be defined to represent, implicitly, geometric models.

Revit® allows, using its standard features, the implicit representation of some objects. However, these resources proved to be too limited to meet the requirements for representation of the basic elements of MDP.

In the following topics are presented the details of the two alternative representations proposed in this paper.

4.1. Explicit representation

To implement this representation, we used the object family concept. A family is a group of elements (2D/3D) which has a set of common properties (parameters) and a graphical representation. The parameters of each element of a family can take different values. These variations within the family are called types or family types (AUTODESK, 2008b; AUTODESK, 2009b).

Before a family can be used, it must first be loaded into the project. Once loaded, family types can be instantiated in the project through a specific Revit® command. Thereafter, each instance can have its parameters changed to meet its specific design requirements. In Autodesk (2009c), 3 types of families are described:

System families

- Define the basic elements of a construction: walls, roofs, ceilings, floors.

Elements that define the system settings, such as layers, grids, sheet layouts and viewports are also covered by system families;

- System families are predefined and one cannot change their basic definitions (i.e., creation of new parameters). The only customization allowed is adding new types in an existing family;
- Generally, it is not required that other objects are in place to instantiate their types, i.e., they are usually not hosted in other elements.

Loadable families

- They are used to define building components that are normally purchased, manufactured or installed, or annotation elements;
- These families are created in external files (with the file extension *.rfa*) which must be loaded in the project.

In-Place families

- Define elements considered specific to a particular project;
- The geometry of objects built with in-place families can be linked to other objects in the design (walls, slabs, roofs, etc.);
- When the reference objects are changed, these changes are propagated to the objects of the in-place family;
- In-place families cannot be shared with other projects. They are always created in the context of the current project;
- It is not recommended the creation of many in-place families in a project, because this practice can degrade Revit's performance.

The kind of family chosen to represent the elements of MDP was the loadable family, because with this family it is possible to set custom parameters and reuse these families in other projects.

To meet the requirements of MDP, the following families of objects are needed: block modules, lintels, sills, reinforcement meshes and frames. To simplify the problem, this work only focuses on the representation of modules of blocks used to

compose the modulation of a wall. The joints between the blocks were represented as parameters of the family, instead of 3D elements.

The first step in the implementation of this alternative was the definition of the family of blocks that would be modeled in Revit®. As the idea was just to represent any family of blocks, it was decided to model the concrete blocks of a traditional Brazilian manufacturer, whose specifications were collected on its website (GRESCA, 2008).

After the definition of the block family, we moved to the modeling of this family in Revit®. We considered only the external dimensions of the blocks. The holes were not represented. Although Revit® allows the complete block representation, these simplifications were adopted as they do not impact much the results.

To create a family, the software has several templates. The choice of the template file depends on how the family types should interact with other elements of the design (AUTODESK, 2008c; AUTODESK, 2009c).

In the specific case of wall blocks, the idea is they being hosted on wall objects. Among the templates available, the one that met these requirements was the *Metric Generic Model wall based.rft*. This type of model is used to create families of objects whose types can be instantiated only within the walls. When starting a new family project using this template, an example wall is loaded. This wall serves as a reference for modeling the new family.

Another interesting feature when using this template is that some dimensions of the object defined in the family can be adjusted to suit the dimensions of the wall. For this to happen, during the modeling process of the family the user must restrict the geometry of the new object to the faces of the reference wall. In this case, the parts of the geometry restricted to the reference wall should not be dimensioned, otherwise these parts will not follow the updating of the wall dimensions in the design.

Considering this characteristic, the parameter of the block thickness was linked to the wall core thickness, leaving the parameters *length* and *height* of the block types defined by the family.

The problem encountered here was that, by using this family of blocks in a project that had walls with different thicknesses, the command to extract quantities would not make a distinction between blocks of the same type but with different thicknesses. For example, whole blocks in walls of 14 cm and 9 cm, would be counted together, when in fact it should not be.

The solution adopted in this study was to duplicate the family of blocks, considering the following differences between them:

- The names of the families contain the thickness of the wall to which they are applied;
- The types are named with the model of the block and all block dimensions (length x height x thickness).

Figure 2 shows the control parameters defined for a specific family of blocks for walls 14 cm-thick.

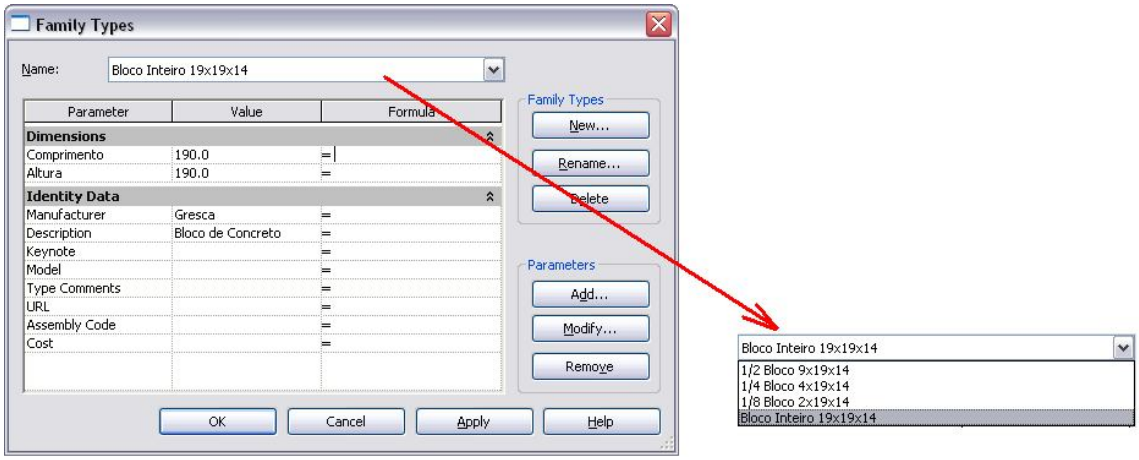

Figure 2 – Family types of Gresca blocks for walls with 14 cm of thickness.

Revit® also has a "nested family" concept. This concept allows the creation of families that include other families. Using this concept, we created a family called *Masonry*, inside which block family types were instantiated in the form of parameterized arrays.

Arrays are sets of elements that are repeated. Linear arrays are distributed on a straight line and radial / polar arrays on arcs. When creating arrays, the number of

elements is always one of the parameters. Other parameter is the spacing between the elements or the total length or angle. In the case of parametric arrays, the elements are always created associated and the array parameters can be edited so that the movement of an element of it automatically changes the spacing between the others, keeping it uniform. A change in the number of elements leads to their redistribution in the overall length, if this is another array parameter.

In the masonry family, we defined parameters to control the number of items in the parametric array as a function of the length and height of the wall. According to Ferreira (2007, p. 50), AutoLISP™ routines can be used to automate the modulation activity in AutoCAD®. Access to these routines allowed us to use some of the parameters needed in the design of the masonry family (Figure 3).

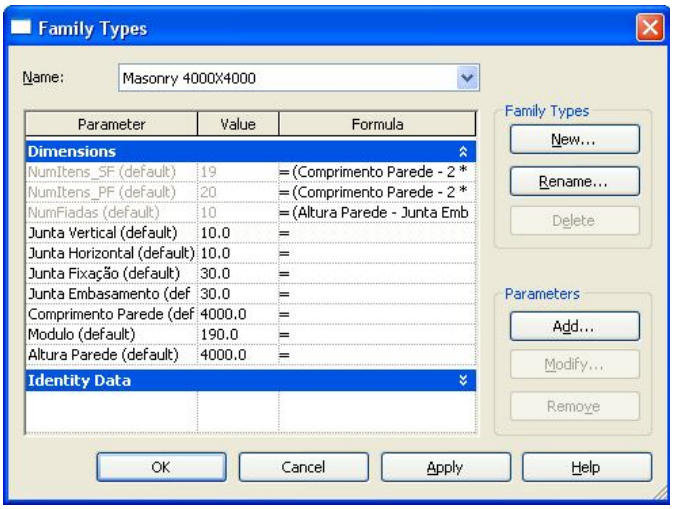

Figure 3 – Masonry family parameters.

The parameters set to control the number of array elements are parametric value-driven formulas that, in turn, use the other parameters of the masonry family.

The wall length and wall height parameters were created because it was not found, in the documentation available for Revit® 2009, a way to reference these parameters in the formulas (AUTODESK, 2008b; AUTODESK, 2008c; AUTODESK, 2009c). We also investigated this feature in the documentation of Revit® 2010 and confirmed that this limitation still exists in the most recent version.

Ideally, when instantiating a masonry family type on a wall, it should be possible

to automatically capture the values corresponding to its length and height. However, to use this family, the designer should assign values to these (length and height) and the other parameters after the instantiation of the type.

When types of a nested family are instantiated, its internal elements cannot be accessed. In this case, the blocks cannot be accounted for by the quantity extraction command (AUTODESK, 2009c).

However, Revit® also has the concept of shared families. This concept allows one to enable access to the combined nested families. The family of blocks created has been adapted to incorporate the concept of shared families, using the command *Settings > Family Category and Parameters*.

4.2 Implicit representation

The implicit representation proposes the blocks, joints and other elements of the wall not being modeled. Instead of modeling these elements independently, the idea of this approach is to create a special wall family.

This new family would incorporate the representation of the blocks and joints by means of a parametric array. The representation of the structure elements would be left to independent families that have their types instantiated on the walls of this special family. This special wall family would also automatically regenerate the parametric array after the instantiation of a type belonging to an interfacing family (lintels, sills, frames).

In Revit®, wall objects are defined as special families called system families. Families of this kind cannot be created and its definition is a "black box", that is, there is no way for accessing its implementation (AUTODESK, 2008c). The maximum customization allowed for these families is the creation of new types, which always reflect the settings (parameters) of the original family (AUTODESK, 2009c).

Consulting the documentation available for the management of Revit® 2010 families, it was found that even in the new version it is not possible to create new system families. The *Application Programming Interface* documentation (API) was also checked and it appears that there are no ways to create new system families

using the programming interface either.

The creation of new types within a wall family allows the creation of new layers (Figure 4) with its features, but does not allow the creation of new parameters that would be useful for the implementation of a parametric array.

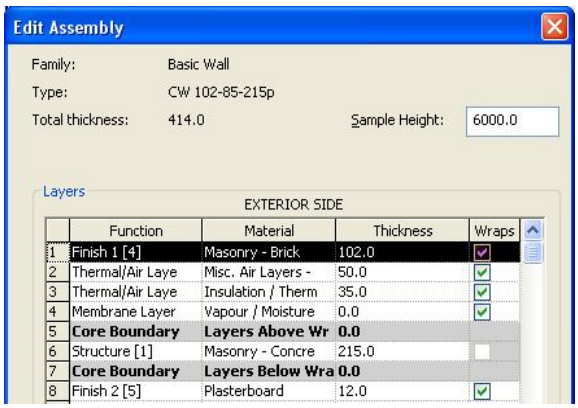

Figure 4 – Layer definitions in one family type of Basic Wall family.

In defining the layers of a wall, blocks can be represented using a hatch. Even today there are families with these characteristics (AUTODESK, 2008b), but it was not possible to represent the various modules of a block family using a hatch. In addition, a modulation can take many different configurations depending on the interface elements and elements belonging to other subsystems, such as HVAC elements, for example.

Considering the limitations presented for the design of a particular wall family, we decided to evaluate the use of generative modeling techniques as an approach to solve this problem.

The use of generative modeling supposes the definition of shape vocabularies and rules to represent these shapes. The set determined by the representation rules and the shape vocabulary is called a shape grammar.

Using loadable families, the resources available on the Revit® API and the concept of grammars, we intend to implement a special family that will have the conventional walls as host objects.

The idea is that this family represents the modulation of masonry and stores the

variables used to generate this modulation. A compact expression stored in a property of this family would represent the designed modulation. The modulation rules translate this expression in a 2D graphical representation (texture / symbol), according to the needed visualization.

The results of processing this expression would be: a) a texture applied to the faces of the object wall; b) a symbolic representation of the block courses in plan and elevation views; c) the quantity of blocks and mortar volume consumed by the mortar joints, being both values stored in custom wall properties.

Additionally, it should be possible to update this texture and quantities when the dimensions of the wall are changed by user. It is believed that this behavior can be implemented linking identifiers of the host wall and of the customized wall, through the resources of the Revit® API.

5. A GRAMMAR TO REPRESENT MODULATIONS

The idea of using a compact expression to represent the modulation of a wall leads to the need to formally define what elements - symbols (shapes) and reserved words - should compose this expression or, in other words, what should be the syntax of this expression. To define this syntax, a grammar can be used. Grammars are formalisms used to define symbolic or visual languages.

Our initial studies on the specification of shape grammars allowed the outline of a specialized grammar for representing masonry modulations.

According to Celani et al. (2008), the essential elements of a shape grammar, which must be defined in this order, are:

- Shape vocabulary – a finite set of primitive shapes that can be two- or three-dimensional;
- Spatial relations – a set of desired combinations between the primitive shapes of the shape vocabulary;
- Rules – from the spatial relations, transformation rules like A $\rightarrow$ B (when finding A, replace it by B) are defined. These rules can be classified into three groups: addition, subtraction and substitution;

- Starting shape - to initiate the application of the rules, one must select a starting shape, belonging to the shape vocabulary.

By following these guidelines, in addition to those presented in section 3 of this work, we defined the essential elements of a new visual language that we call MML (*Masonry Modulation Language*). Table 1 describes the elements of the grammar that generates MML.

The scope of MML is a description of the masonry modulation using a shape grammar that contains the rules and basic shapes of modulation (blocks and joints). The representation of interface elements, such as lintels, sills, reinforcement meshes and frames are outside of the scope of MML.

Element	Description
Shape Vocabulary	Block modules (full block, 1/2 block, 1/4 block, 1/8 block); Head joints (dry and filled); Bed joints (base, wall-slab and intermediate).
Spatial Relations	Horizontal positioning: block module + head joints; Vertical positioning: block module + bed joint; Horizontal positioning with module rotated to 90 degrees: rotated block module + head joints.
Rules	The transformation rules used in the MML were grouped into two categories: • Additive - used for the juxtaposition of block modules with head joints (horizontal modulation) and of the block courses with bed joints (vertical modulation); • Substitute - used for replacing modules not rotated by a rotated version (typical situation when completing a vertical modulation). Also are included in this category the rules for bonding of walls, because they involve the switch of block modules at the ends of courses.
Start Shape	Any block module can be used as a starting shape.

Table 1 – MML grammar elements.

6. APPLICATION EXPERIMENTS

The results presented in this section are limited to the solution for explicit representation, based on the use of families of objects. The solution to implicit representation has not been implemented yet, because it is still in specification phase.

Using the families described in the previous section, experiments were performed to evaluate which of the proposed families is the most appropriate for the MDP.

Table 2 describes the system configuration of the computer used to perform the application experiments.

Model	NOTEBOOK ACER ASPIRE 4520
Processor	AMD ATHLON X2 64 BITS
RAM Memory	2 GB
Virtual Memory	2 GB
Hard Disk	TOSHIBA MK 1646GSX 150 GB
Graphic Card Adapter	NVIDIA GEFORCE 7000M / NFORCE 610M 512 MB
Operational System	WINDOWS XP PROFESSIONAL SP3 32 BITS

Table 2 – Computer system configuration for the tests.

In these experiments, it was only considered the modulation of a "blind" wall, i.e., one without openings and structural elements. Although this type of wall is an exception in MDP, it was enough to draw some conclusions about the problem addressed in this article.

6.1 Experiment 1 – Block Family

In this experiment we used the family of blocks set up for the modulation of a wall. Below are described the steps in this experiment:

- Wall file:

  Creation of a new project file containing a wall of family *Basic Wall: Interior Blockwork 140*, 4 meters long and 4 meters high;

- Loading the family of blocks, previously created in the project;
- Instantiation of two whole blocks and a 1/2 block to define the starting elements of the first and second courses;
- Positioning of these blocks using the *Dimension* command;
- Replication of the blocks in the first course using the *Array* command with the option *Group and Associate* checked. This option allows to associate the array elements and add them together, making the array parametric;
- Replication of the blocks of the second course using the *Array* command with the option *Group and Associate* checked;
- Instantiation of a 1/2 block to finish the second course;

- Replication of the first and second courses, towards the height of the wall, using the *Array* command with the option *Group and Associate* checked.
- Copy of the wall file:

Creation of three copies of the wall file using the command *File> Save As*

- Master File:

Creating a new project to group the copies of the walls;

- Inserting references to the walls using the *File> Import/Link* Revit® command;
- Copies of the references in the project using the *Copy* and *Mirror* commands.
- Block quantification:

Extraction of block quantities using the *Schedule/Quantities* command in the *View Panel.*

- Plan view of first and second courses:

Extraction of plan views of first and second courses by using the Revit® commands to generate views (Figure 5).

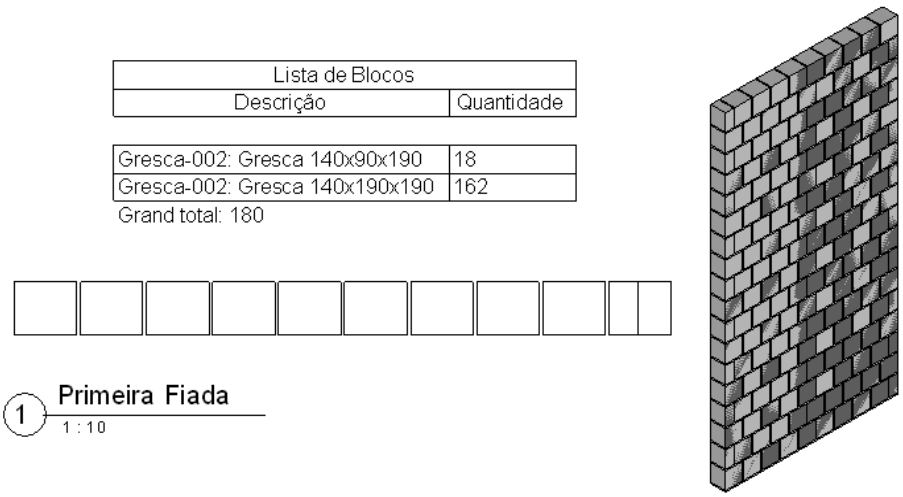

Figure 5 – First course view and block quantitative extracted from the 3D model of one wall built with the developed block family.

6.2 Experiment 2 – Masonry Family

In this experiment we used the masonry family (shared and nested) to perform the modulation of a wall. This family contains instances of the family of blocks arranged in the form of a parametric array. Below are described the steps in this experiment:

- Wall file:

  Creation of a new project file containing a wall of family *Basic Wall: Interior Blockwork 140*, 4 meters long and 4 meters high;;

- Loading of masonry family in the project;
- Instantiation of the standard type of the Masonry family;
- Edition the parameters of the type instantiated (joints, wall length and height) for regeneration of the parametric array.
- Copy of the wall file:

  Same as in experiment 1.

- Master File:

  Same as in experiment 1.

- Block quantification:

  Same as in experiment 1.

- Plan views of first and second courses:

  Same as in experiment 1.

Also, it was studied an alternative for creating a family to represent courses, rather than the complete wall modulation. The objective was to determine whether the use of this family would make the course maintenance more practical. The AutoLISP™ routines used by Ferreira (2007) have rules for distribution of blocks in a course, but it was not possible to incorporate these rules by using formulas in the definition of the proposed family, because the use of formulas in Revit® is too simple compared to the resources available in AutoLISP™. Revit® does not allow programming new commands using formulas in families.

The rules contained in the routines used by Ferreira (2007) not only calculated the joints, but also selected the block modules to suit the courses. In those AutoLISP™ routines, the parametric array was assembled in run time to adapt to different wall dimensions. In the case of Revit®, the same array is preset in the Masonry family.

7. ANALYSIS

7.1 Experiment 1 – Block Family

The files generated in this experiment used Revit® standard commands for the distribution of the blocks (*Array* and *Dimension*). Positioning and editing of each block resulted very flexible. In each course, it was possible to modify the number of items in the array and replace the block modules when necessary.

When the *Array* command is used, Revit® creates a group of objects if the option *Group and Associate* is enabled (AUTODESK, 2008b). After the command, one can change the number of elements without ungrouping the array, but if one needs to replace a block within the array, the ungrouping needs to be done, losing the parameterization of the array.

Regarding response time in view commands (*Rotate, Zoom, Pan*), it was found that with the wall file the Revit® performance was good. But when viewing the master file containing 16 walls, with 4 of these as references and the others as copies, the regeneration performance fell slightly relative to the previous case. The size of files generated in this experiment were also recorded (Table 3).

Description	Disk space	RAM Memory space
Gresca block family	196 KB	47 KB
Wall file (14x400x400 cm)	676 KB	75 KB
Master file with 16 walls	1988 KB	117 KB

Table 3 – File sizes in experiment 1.

7.2 Experiment 2 – Masonry Family

In this experiment, the positioning of the blocks was automated by the parameterized array embedded in the default family type. The default position of

the modulation in the Masonry family was executed with the *Dimension* Revit® command.

After instantiating the default type of the Masonry family, the designer needs to update the parameters of the instance to regenerate the parametric array. It was found that the time for regeneration of the array was very high, taking, in some cases, several minutes, depending on the size of the wall (length and height).

In the viewing commands (*Rotate, Zoom, Pan*) the response time was faster when handling the master file compared with the master file generated in experiment 1. Revit® optimized the viewing of the walls with the Masonry family. The sizes of files generated in experiment 2 are listed in Table 4.

Description	Disk space	RAM Memory space
Gresca block family	196 KB	47 KB
Masonry family	504 KB	67 KB
Wall file (14x400x400 cm)	1952 KB	75 KB
Master file with 16 walls	684 KB	103 KB

Table 4 – File sizes in experiment 2.

8. CONCLUSIONS

In MDP, the masonry can take various configurations depending on the specific needs of integration with the subsystems or components (lintels, sills etc.) it interfaces. To enable the coordination among different subsystems, the representation of the elements of a wall becomes crucial.

To meet this MDP requirement, our path was to adopt an explicit representation of the elements of the wall using families to represent each element of the modulation: concrete blocks, lintels, sills, reinforcement meshes and frames. Although this approach naturally degrades application performance, it was the one that offered more flexibility for the masonry designers.

The use of a parametric array to represent a modulation proved to be an interesting solution to automate the process of distribution of blocks using Revit® standard

features. However, this solution did not offer flexibility in the editing of courses and automatic resolution of the various modulation possibilities available to a particular family of blocks.

During the coordination process, the designer often needs to change the distribution of blocks on the wall. Thus, even being practical, the parametric array will always be broken in this process. This MDP requisite ultimately makes the modulation a dynamic factor, difficult to implement by a parametric array.

Still regarding the use of the parametric arrays, we stumbled on the technical difficulty of incorporating the rules of distribution of blocks used by Ferreira (2007) in her AutoLISP™ routines. The rules contained in these AutoLISP™ routines selected the block modules and optimally calculated head joints as a function of wall length. The formula feature within the definition of parameters in the families gave no support for this type of customization (AUTODESK, 2008b).

Regarding the implementation of the implicit representation of the elements of a wall, we studied the custom pattern feature (custom hatches). It was found that it is possible to create new hatch patterns and apply them on the external sides of the walls to simulate a cladding. However, the controls for parameterization of hatches are very simple, and are limited to controlling the spacing and angle of these elements. It is also possible to use a description language for defining new hatch patterns, but such language is also not capable of describing the whole complexity of a modulation.

The limitations encountered in the implementation of the implicit representation using standard Revit® resources leaded to the evaluation of the use of generative modeling techniques as an approach to solve this problem.

The conclusions of this study are limited to the scope of resources available in Revit® Architecture 2010, which does not rule out a similar survey with other BIM tools in order to verify which of the software available in the market is suitable for MDP.

As future works we will complete the detailed MML specifications in order to include the treatment of other rules of modulation and the subsequent implementation of this visual language in a BIM tool.

REFERENCES

ABCI. Associação Brasileira da Construção Industrializada. **Manual técnico de alvenaria**. São Paulo: Projeto/PW editores, 1990.

AUTODESK. **Building Information Modeling:** The Power of BIM. 2008a. Available at: <http://www.autodesk.com/bim>. Accessed: 25 mar. 2008a.

AUTODESK. **Revit Architecture 2009**: User's Guide. 2008b.

AUTODESK. **Melhores práticas para a criação de componentes paramétricos (famílias) com o Autodesk Revit**. 2008c. 15p. Available at: <http://www.autodesk.com/bim>. Accessed: 15 out. 2008c.

AUTODESK. **Revit Architecture 2010**: User's Guide. 2009a.

AUTODESK. **Revit Architecture 2009**: Families Guide – Metric Tutorials. 2009b. 812p. Available at: <http://www.autodesk.com/revitarchitecture-documentation>. Accessed: 05 jan. 2009.

AUTODESK. **Revit Architecture 2010**: Families Guide – Metric Tutorials. 2009c. 812p. Available at: <http://www.autodesk.com/revitarchitecture-documentation>. Accessed: 15 set. 2009.

CELANI, G.; CYPRIANO D.; GODOY G.; VAZ C.E. **A gramática da forma como metodologia de análise e síntese em arquitetura**, 19p, 2008. Available at:

<http://www.ufpel.edu.br/faurb/prograu/documentos/artigo2-sustentabilidade.pdf>. Acessado em: 25/09/09.

CHING, F. D. K. **Dicionário Visual de Arquitetura**. São Paulo: Ed Martins Fontes, 1999.

CORONA, E.; LEMOS, C. **Dicionário da Arquitetura Brasileira**. São Paulo: Edart, 1972.

EASTMAN, C. **New Opportunities for IT Research in Construction**. In: SMITH, I.F.C. EG-ICE 2006/ LNAI 4200. Berlin:Springer, 2006. p. 163-174.

FERREIRA, R. C.. **Uso do CAD 3D na compatibilização espacial em projetos de produção de vedações verticais em edificações**. 2007. 160 p. Dissertação (Mestrado). Escola Politécnica da Universidade de São Paulo, São Paulo, 2007.

GRESCA. **Vedação 39 cm**. 2008. Available at: <http://www.ceramicagresca.com.br>. Accessed: 20 set. 2008.

SCHEER, S.; AYRES FILHO, C.; AZUMA, F.; BEBER, M. **CAD-BIM requirements for masonry design process of concrete blocks.** In: CIB W78 INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY IN CONSTRUCTION, 25., 2008, Santiago. **Proceedings**... Santiago:Universidad de Talca/Stanford University, 2008, p. 40-47.

SILVA, M.M.A. **Diretrizes para o projeto de alvenarias de vedação**. 2003, 167p. Dissertação (Mestrado). Escola Politécnica da Universidade de São Paulo, 2003.