Proceedings of the Sixteenth International Conference on
Computational Structures Technology
Edited by: P. Iványi, J. Kruis and B.H.V. Topping
Civil-Comp Conferences, Volume 14, Paper 8.2
Civil-Comp Press, Edinburgh, United Kingdom, 2026
ISSN: 2753-3239, doi: 10.4203/ccc.14.8.2

The Automation of Building Information Modeling (BIM) Methodology in Structural Design of Concrete Walls

M. Cangussu Da Silva

Fumec, Belo Horizonte, Brasil

Abstract

The adoption of digital platforms like Building Information Modeling (BIM) has become increasingly prevalent in the Architecture, Engineering, and Construction (AEC) industry, facilitating information sharing among stakeholders and specializations. This advancement has introduced automation through visual programming languages, eliminating repetitive tasks and enabling interaction with modeling technologies such as Autodesk Revit. These capabilities create new, efficient workflows. This article explores the automation of BIM processes using visual programming languages like Dynamo and Python, focusing on structural projects in Autodesk Revit. The main goal is to optimize workflows, enhancing collaboration and efficiency while showcasing the potential of technology in structural design. A case study on routines for cast-in-place concrete walls serves as the practical application. Initially, the research identifies processes for automation, leveraging Dynamo—a visual programming plugin integrated into Revit. Methodologies assess routine development, usage, documentation, and their benefits. Specific algorithms automate tasks like importing data from AutoCAD files and modeling reinforced concrete walls in Revit. This reduces errors, simplifies processes, and boosts productivity. Demonstrations of these routines include QR codes linked to videos showcasing the automations' functionality. The study highlights how algorithms in BIM software significantly improve efficiency, collaboration, and quality in structural projects. By advancing the automation of BIM processes, this work contributes to the broader adoption of BIM methodology, promoting innovation and modernization in the construction industry.

Keywords: automation, BIM, dynamo, python, concrete walls, processes.

1 Introduction

The construction industry is mentioned on several occasions as being significantly more important than other industrial sectors, even though it is characterized as having the greatest productive inefficiency. In order to raise the sector's productivity indicators, given the high level of industrial competitiveness, it is extremely important to apply new technologies, focusing on automation, quality and process reliability. In this way, albeit belatedly, the BIM (Building Information Modeling) methodology has been proposed and is the current protagonist in the application of new technologies in construction.

Eastman *et al*. [1] are responsible for defining the BIM concept as the creation of a precise digital model of a building in a parameterized way, with the aim of simulating all the stages of construction. As a result, it becomes possible to check for possible problems and practical interferences, even during the design phase, to ensure savings in time and construction costs, with possible applications also in the building's use and maintenance stages.

The use of tools and software using the BIM concept, as defined above, does not necessarily result in faster project delivery, although its benefits are perpetuated throughout the life of the projected building. For this reason, in order to ensure that deadlines are met, the automation of processes becomes necessary in the environment in which these tools are implemented and used, as it can be carried out using various high-level programming languages or visual programming, in order to help carry out repetitive activities and obtain new functionalities.

The BIM methodology, with a focus on application in construction projects, is listed as a tool for identifying inconsistencies, promoting optimization of the structures designed and integration with other project disciplines. In this way, it can be used in favor of the compatibility process, to reduce costs and reduce existing rework during the execution and preparation of projects. Among the existing structural models in which the BIM methodology is used, this paper focuses on the cast-in-place concrete wall system.

The cast-in-place concrete wall system began to be used in Brazil in the 1970s and 1980s, but its widespread use only began in the 2000s, with the creation of the "*Minha Casa Minha Vida Program*" by the federal government, created with the aim of reducing the housing deficit, which is recognized as one of the country's main socioeconomic problems. The program promoted the growth of the construction process, due to its ability to meet the demand for quality buildings in the short term (MONGE *et al.* [2]).

With the growing use of concrete walls in Brazil, the need arose to create a national technical standard so as to standardize and establish technical requirements in relation to this system. NBR 16055 was therefore published in May 2012 [3] by the Brazilian Association of Technical Standards (ABNT). There are several discussions regarding the current state of the art, while, for example, Farias [4] reports on the scarcity of bibliographical references on the subject and the need for new studies and publications on this construction process.

As previously described, this work proposes the development of routines and algorithms that can significantly aid automation during the preparation of structural projects, in order to highlight the importance of productivity gains in the academic field and in the work routine of structural design offices. As described above, a reinforced concrete wall project was used to validate and apply the routines and algorithms proposed here.

2 Materials and Experimental Program

2.1 Autodesk Revit Automations

To achieve the objectives of this work, the first stage focused on the development of scripts in Dynamo, created to automate the structural design of concrete walls by means of computer routines that perform processing based on inputs entered by users. In a second stage, the routines were tested during the automation actions listed as objectives and demonstration videos were produced to show how they were used.
The combination of Autodesk Revit software and the Dynamo and Python programming languages was adopted to ensure the automation of structural features. Figure 1 illustrates the steps taken to meet the general and specific objectives.

Figure 1: Methodology for developing automation.

In order to automate processes and provide significant assistance to structural designers, routines were created in a visual programming language. For this stage, the Dynamo software was used through the Python programming language, by inserting lines of code.

Algorithms were developed in Dynamo and applied directly to Autodesk software, including Autodesk Revit. The proposed routines will be composed of linear processes, data and information, as shown in Figure 2. The execution flowchart began with the data needed to execute the routine, followed by the processing and development of processes, as well as the design of models and actions within the software.

Figure 2: Steps for data processing.

3 Results and Discussions

3.1 The Use of Dynamo and Python Programming Languages

During the development of the routines, integration between the Dynamo and Python languages was proposed in order to optimize and expand the automation possibilities. This integration came about as a natural solution to overcome the limitations of each tool individually, taking advantage of their respective advantages in a complementary way.

Thus, after Stage 1, mentioned in Figure 1, the actions required and carried out by designers were observed and compiled, as to classify which of these were more easily reproduced via Python and, conversely, which could only be carried out using Dynamo.

In general, steps that involved human intelligence, such as selecting the first element in a list, storing dimensions and distances between elements, categorizing elements, among others, were primarily done using Python. In the opposite scenario, actions that were essentially execution activities, such as inserting elements, rotating geometries, among other actions carried out via clicking, were provided in the foreground with Dynamo.

The main reason for choosing Dynamo as the main tool for executing actions within Autodesk Revit was its intuitive visual interface, which facilitates the creation of scripts and workflows, exactly one of the objectives of this study. In this way, Dynamo allows direct manipulation of Revit elements, just as users do in their routine activities without the application of automation.

However, although Dynamo is a suitable language for the applications proposed in this study, it has some limitations when it comes to carrying out complex and specific operations, such as advanced manipulation of lists, vectors, selecting the position of elements and storing data for later manipulation. Therefore, to handle more elaborate

operations, such as manipulating large volumes of data, advanced mathematical calculations or tasks that require greater logical control, Python was the choice.
The hybrid approach proposed in this methodological step makes it possible to combine the simplicity and accessibility of Dynamo with the power and flexibility of Python, to create a robust and efficient solution for automation in Revit.

3.2 Process Automation

During the first stage, the architecture for data entry was developed by integrating the user with Autodesk Revit software. When executing the desired routine, the user was asked to fill in a personalized form. This data is usually variables of the type *integer*, *decimal*, *project element*, *true* or *false*, among others. Once this initial data has been obtained, the data will be processed in visual programming using Dynamo with the help of a few lines of Python code. It should also be noted that processing is only possible if users fill in the data correctly.
The processing presented resulted in the creation of actions in the software through the use of the routine developed to facilitate repetitive tasks in the model. In designing the routines, methodologies were applied to organize the algorithms in order to facilitate refactoring in different scenarios and maintenance.
After proposing, registering, developing and using the routines, an analysis was made of the advantages they provide, with the aim of discussing the relevance of implementing automation processes in structural design using the BIM methodology. The purpose of the proposed routines is to automate models and processes inherent to structural projects prepared in Revit software, such as the automatic creation of walls from a .DWG file, the insertion of lateral reinforcements in openings, welded mesh in walls, stirrups in beams and columns and the automatic elevation of walls, elements that are complementary to the structural model.

3.3 Access Menu in Autodesk Revit

In order to gather and compile the routines created during this work, a methodology was evaluated that would make it easier for users to access the proposed automations in the native Autodesk Revit interface.
This methodology was implemented with the help of the NonicaTab plugin, available on the Autodesk App Store. This plugin allows routines previously developed in the Dynamo language to be brought together in Autodesk Revit's native interface, as well as allowing users to manage *inputs* and *outputs* in the interface.
Once the plugins had been chosen, each of the routines developed was given a custom icon, designed in a size of 32 x 32 px, to meet the NonicaTab's requirements. In addition, the files with a dyn extension were renamed to also meet the required layout, so that the file name was inserted in such a way as to meet size and readability requirements, making it easier to understand what the automation inserted in each one is about. Therefore, the positioning and organization in the Nonica plugin make for excellent usability and user experience, so that the proposed new menu does not contrast with the others and with the groups present in the software, as it specifies the appropriate identification names for the menu and the routines tab.

3.4 Replicability Guide

This topic aims to provide a guide for replicating the automated routines developed during this study. Although the codes may seem complex at first glance, all the routines were created following a specific methodology that will be explained below, ensuring the replicability of the processes and understanding of the steps involved.
The first stage consists of selecting the project routine to be automated. At this stage, it is recommended to select activities that are made up of repetitive steps and therefore consume a great deal of work effort and time on the part of the designers of concrete wall projects in BIM software. After selection, the second stage consists of describing the applied activities in a language algorithm. For this description, it is recommended to insert the steps in a table, also indicating which steps involve the designer's intelligence or are mostly mechanical. An example description is shown in Table 1.

Stage no.	**Description**	**Was intelligence applied? (Yes/No)**	**Did the Dynamo + Python algorithm perform the step successfully?**
1	Open Autodesk Revit and load the project where the reinforced concrete walls will be added.	No	Yes
2	In the “Architecture” or “Structures” tab, select the “Wall” tool.	No	Yes
3	In the properties panel, choose the type of wall that will be used. If necessary, create a reinforced concrete wall type by specifying the material properties and the desired thicknesses.	No	Yes
4	Adjust the wall properties, such as the height, base level and top level. Make sure that the thickness and material are defined as reinforced concrete.	No	Yes
5	In the plan view, draw the wall by clicking to define the start and end points.	No	Yes
6	Select the wall created and go to the “Structures” tab. Click on “Add Reinforcement” to insert the steel elements into the wall. Choose the type of reinforcement (steel bars) and set the parameters such as spacing, bar diameter and concrete cover.	No	Yes
7	Adjust the frame parameters as necessary. This may include defining anchors, bends and other structural reinforcement specifications.	No	Yes

8	Review the wall and frame to ensure that everything complies with the design specifications. Make fine adjustments as necessary.	No	Yes
9	Save the project to ensure that all changes are stored.	No	Yes

Table 1: Step-by-step manual for creating concrete walls in Autodesk Revit.

Once the framework containing the step-by-step process carried out by a structural designer has been drawn up, the next step is to transform each stage into computer routines. For activities involving the application of intelligence, Python is recommended. Otherwise, the step can be carried out using Dynamo programming boxes.

After inserting the actions in one of the programming languages used, or by integrating the two, the next stage is to test the proposed algorithm in Autodesk Revit, in order to check that each stage of the manual development has been successfully completed, as shown in the column in Table 02.

It is worth noting that the successful development of computer routines involves numerous refactoring steps, i.e. correcting any steps that were not successfully completed. For this reason, it is extremely important that the planning and completion of Chart Z is carried out by a professional who has detailed knowledge of the stages that need to be carried out, so it is not recommended to automate a process that has not been completely mastered by the designers who are looking to implement the facilitation of the process via programming languages.

4 Case Study

In this chapter, in order to test the speed of the automation model developed and the tools involved, an application is presented in a practical case of construction processes for *cast-in-place* reinforced concrete walls. The structure was modeled in Revit using routines developed using the Dynamo and Phyton programming languages.

The routines proposed were the creation of load-bearing walls from a .DWG file (a file prepared in CAD software), stirrups in beams and columns, the automatic creation and lifting of walls and horizontal reinforcements in openings and the creation of welded mesh *in cast* -in-place concrete walls, as described below.

4.1 Routines for Creating Load-Bearing Walls from a .DWG File

In order to enable the automated creation of walls in Autodesk Revit *software*, based on a pre-existing file in .DWG format, generated from one of the software programs that use the CAD methodology, the proposed routine has the input data shown in Table 2.

Input data	Need	Nature
File directory	Identify the file that will be used as a model	File path
Views	Point out the views into which the dwg file will be imported	Project View
Scale	Define the scale at which the DWG plan will be imported where z = 5 (import in centimeters)	Int
Import color	Define the DWG plan import color x = 2 (black and white import)	Int
Location of imported plant	Define the location at which the plan will be imported in y = 2 (centralized import)	Int
Levels	Indicate the start and end levels of the walls created	Project Level

Table 2: Nature and necessity of the input data for the routine for creating structural walls from a .DWG file.

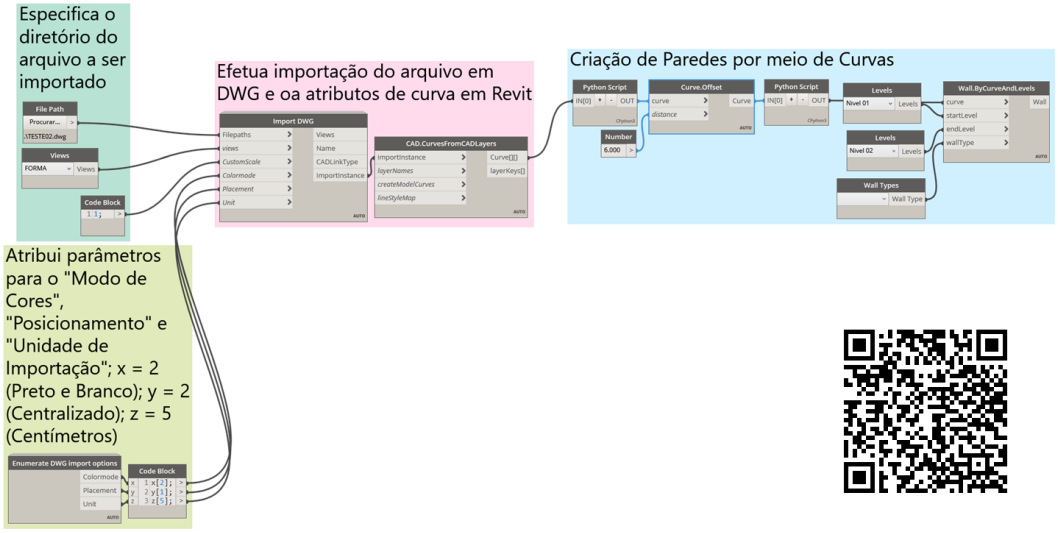

Figure 3: Routine for creating walls from a DWG file.

As shown in Figure 3, the stages highlighted in green are composed, firstly, of *nodes* responsible for storing the directory path of the .DWG file to be imported, as well as the *view* and scale at which the project originating from the CAD methodology should be imported into the Autodesk Revit *software* file.
In the step highlighted in Figure 3, the pink color transforms the lines in the imported .DWG file into curves recognized by Autodesk Revit. The blue one results in the detection of curves that represent the axis of the walls, in order to indicate to the Autodesk Revit software where the walls in the BIM methodology should be created, i.e. from the axis of the walls imported in the .DWG file, as shown in Figure 4.

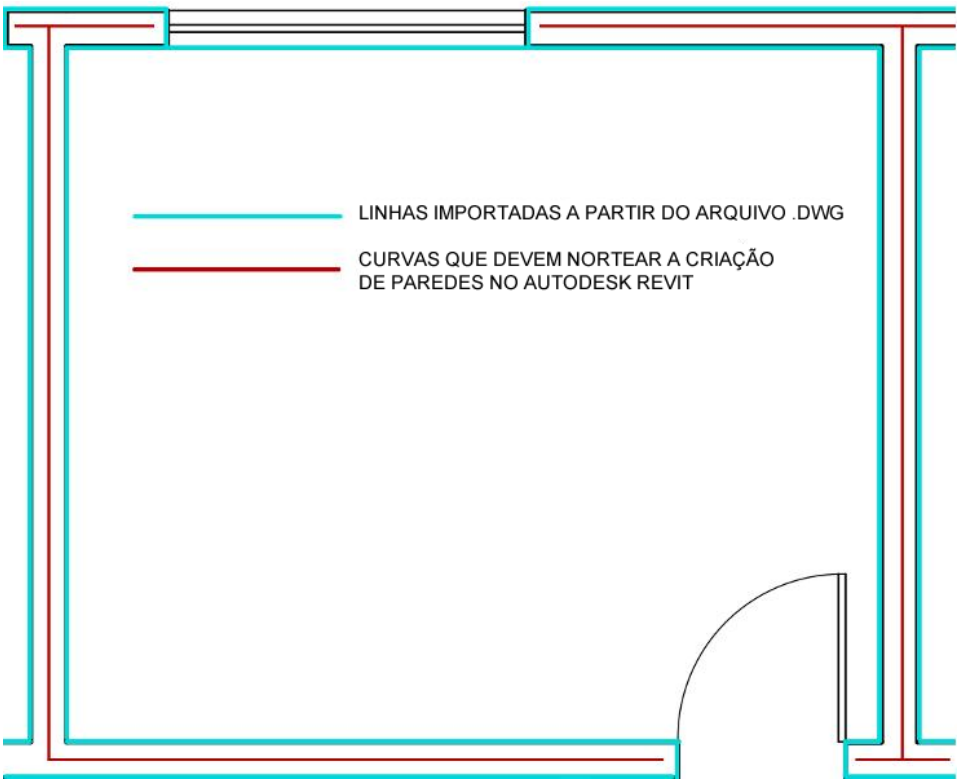

Figure 4: Description of curves and lines to create walls using automated routines.

Also, in the stage highlighted in blue, a node developed in the Python language, as shown in Figure 5, is responsible for collecting the length of each of the curves generated from the lines in the imported file, as well as selecting the half of the curves with the longest length, i.e. selecting the external curves that make up the wall

```python
def calculate_line_length(line):
    # Calcular o comprimento da linha
    if isinstance(line, Line):
        start_point = line.StartPoint
        end_point = line.EndPoint
        return start_point.DistanceTo(end_point)
    return 0

def flatten_list(lst):
    """ Achata uma lista de listas """
    flat_list = []
    for item in lst:
        if isinstance(item, list):
            flat_list.extend(flatten_list(item))
        else:
            flat_list.append(item)
    return flat_list

def select_largest_half(lines):
    if not lines:
        return "Lista de entrada vazia"

    # Achatar a lista, caso haja sublistas de linhas
    flat_lines = flatten_list(lines)

    # Calcular o comprimento de cada linha
    lengths = [calculate_line_length(line) for line in flat_lines]

    # Criar pares de (comprimento, linha)
    length_line_pairs = list(zip(lengths, flat_lines))

    # Ordenar os pares pelo comprimento em ordem decrescente
    length_line_pairs.sort(key=lambda x: x[0], reverse=True)

    # Selecionar a metade superior
    half_length = len(flat_lines) // 2
    selected_lines = [pair[1] for pair in length_line_pairs[:half_length]]

    return selected_lines

# Atribua a sua saída para a variável OUT.
OUT = select_largest_half(IN[0])
```

Figure 5: Python code responsible for selecting the largest curves from the list of curves created.

From the output provided by the Python node mentioned above, the largest curves, i.e. the outer lines of the walls, undergo an operation called offset, which changes their position and length in a pre-established dimension. In this way, the external curves are altered to make up the smaller ones, located in an internal position at a distance set at half the thickness of the constructed wall. After creating the curves as shown in Figure 5, the walls are created using internal curves established from the Wall.ByCurvesAndLevels node.

In this way, the algorithm identifies the thickness of the wall using the lines in the CAD file and creates curves in the center of the walls using a node that already exists in the Dynamo visual programming language libraries. As a result, the walls generated are converted into load-bearing walls, as shown in Figure 6, in order to meet the requirements and be used in structural projects, as recommended in the methodological stage.

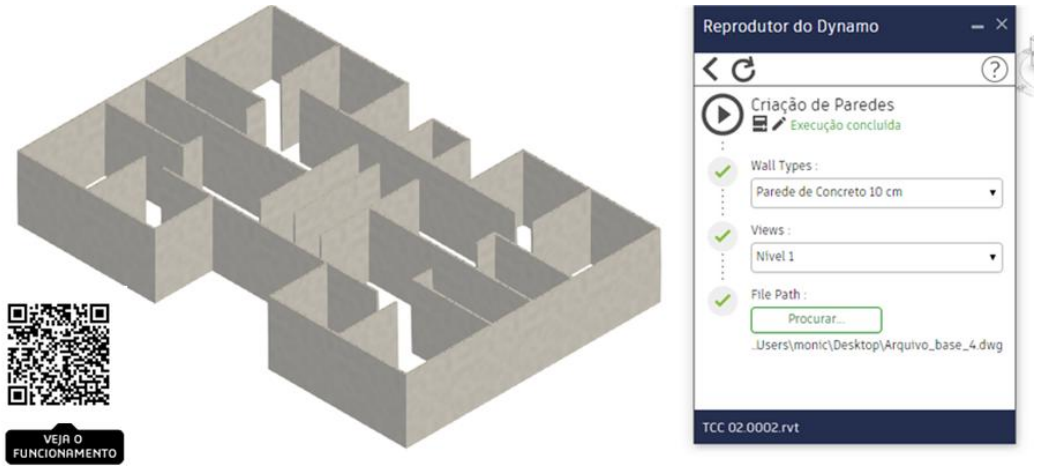

Figure 6: Routine Wall Creation Processing.

4.2 Routines for Creating Stirrups in Beams and Columns

Structural projects in cast-in-place concrete walls often require the use of reinforced concrete structural elements, such as beams and columns. This implementation, in turn, aims to reduce the limitations of this structural model and, therefore, there is a need to create a routine that automates the creation of stirrups.

To enable the automated creation of stirrups in reinforced concrete structural elements, beams and columns in Autodesk Revit software, the proposed routine has the input data shown in Table 3.

Input data	**Need**	**Nature**
Reinforcement host element	Determine the element that will house the reinforcement	Model Element
Armor type	Determine the nominal diameter and steel grade of the reinforcement to be inserted	Element type
Bend	Determine the bending of the reinforcement	Element type
Spacing	Determine the spacing between the inserted reinforcements, if the number of reinforcements exceeds one.	Double

Table 3: Nature and necessity of the input data for the routine for inserting stirrups in beams and columns.

According to Figure 8, the steps indicated in green begin with the creation of nodes responsible for transforming the perimeter of the smallest dimension of the element into curves. Subsequently, a 3 cm offset is made to obtain the curve of the stirrup contour, as shown in Figure 7.

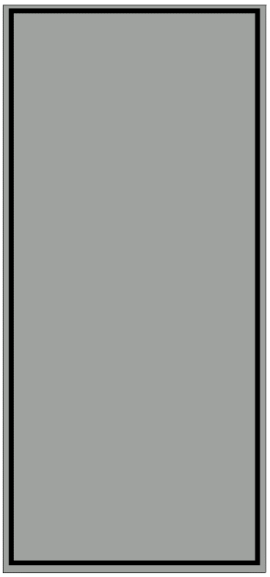

Figure 7: Demonstration of the offset of the internal rectangle in a concrete beam.

In the phase also shown in Figure 8, colored in pink, the nodes convert the “length” instance into a vector, which will be used to distribute the stirrups.
While all the lines that make up the perimeter of the element's smallest dimension have been transformed into curves, the next stage of the algorithm, highlighted in blue, still in Figure 8 and Figure 9, is responsible for transforming these curves into the dimensions of the reinforcement, using the following data provided by the user: *style, type, dimension and orientation of the reinforcement*, as well as the *type of bend* indicated for the stirrups.

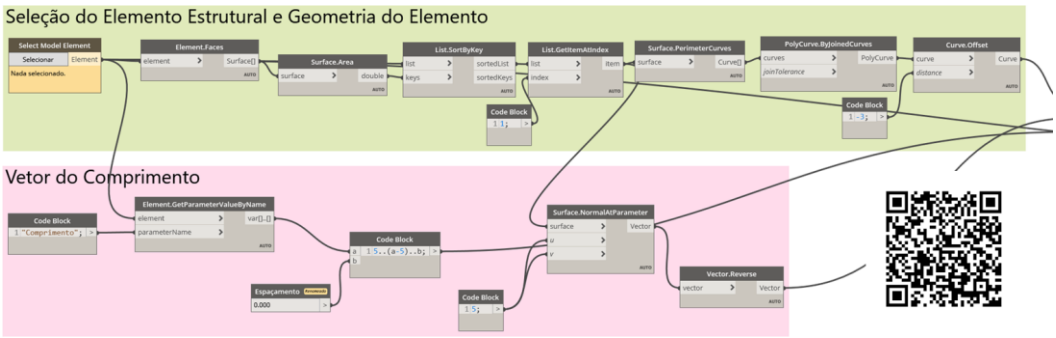

Figure 8: Routine for creating stirrups in beams and columns.

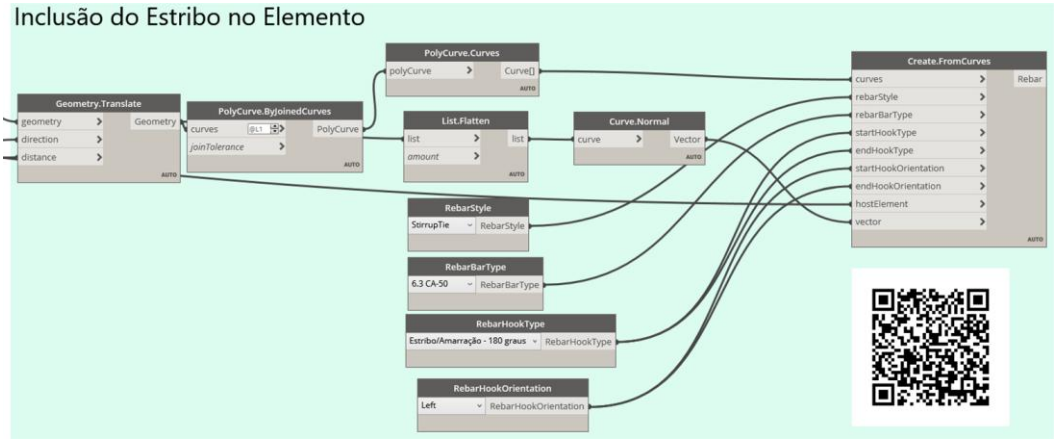

Figure 9: Dynamo algorithm for creating stirrups in beams and columns.

Thus, the algorithm begins by identifying the perimeter in the smallest dimension of the structural element, which is then transformed into a curve. From this curve, the dimensions of the stirrup are obtained, and then the reinforcement is generated with the diameter indicated by the user, as shown in Figure 10.

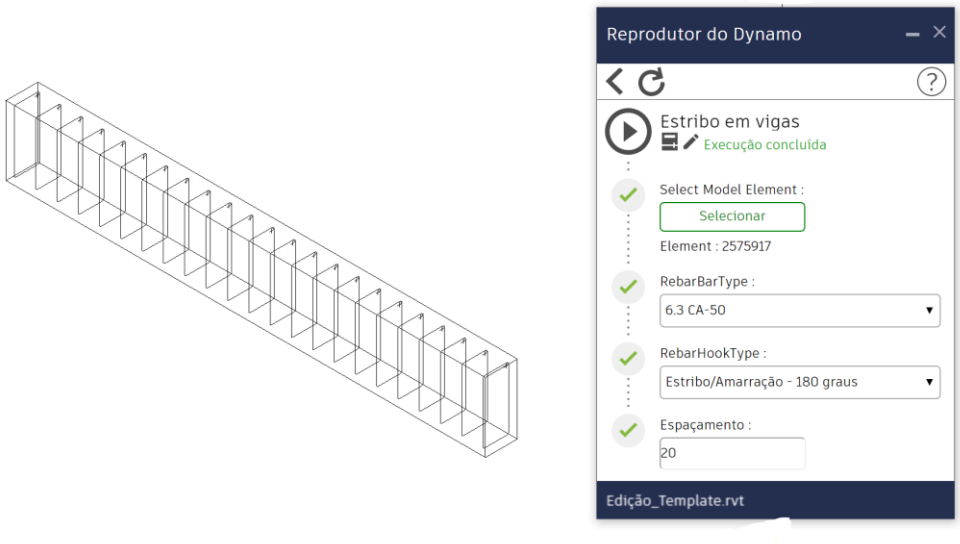

Figure 10: Routine for creating stirrups in beams, with data and output input.

4.3 Routines for Creating Automatic Wall Elevations

In order to enable the automated creation of wall elevations in Autodesk Revit software and to facilitate the detailing actions carried out by structural designers, the proposed routine has the input data shown in Table 4

Input data	Need	Nature
Elevation	Select the type of elevation to be created	View Type Family
Ambientes	Select the environments in which the views will be generated	Model Elements by Category

Table 4: Nature and necessity of the input data for the Automatic Wall Elevation routine.

Using the input data entered and specified by the user, the conclusion of the routine, shown in Figure 11, requires the selection of the environments in which the elevations will be created. After that, the algorithm for the elevation views of the environments indicated by the user is created, to allow the creation of complex details with high productivity.

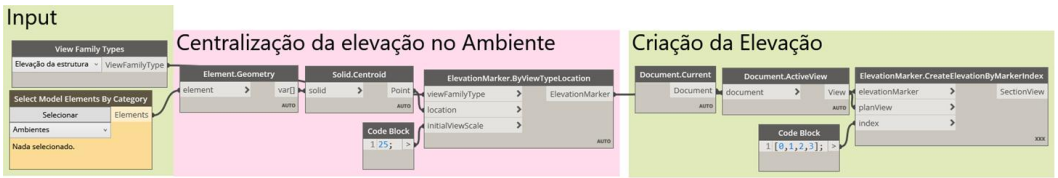

Figure 11: Wall Elevation Routine Processing.

The processing of routine data is presented in Figure 12 and Figure 13, which shows how the use of routines allows the designer to perform more complex and detailed detailing, thus facilitating the execution steps:

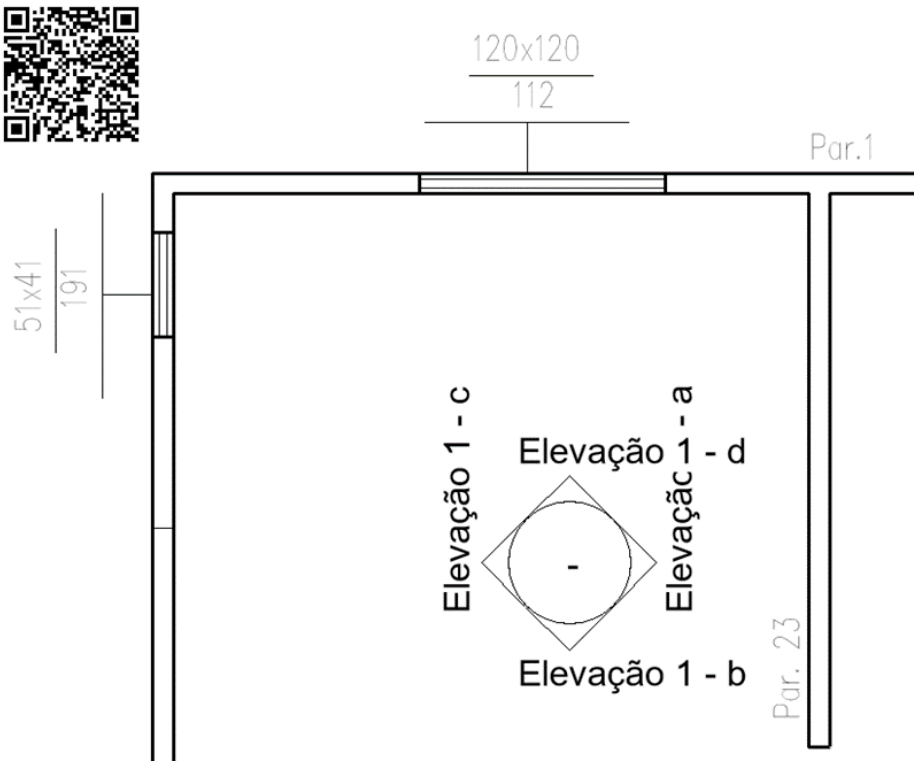

Figure 12: Presentation of the automatic lifting routine symbology in the plan.

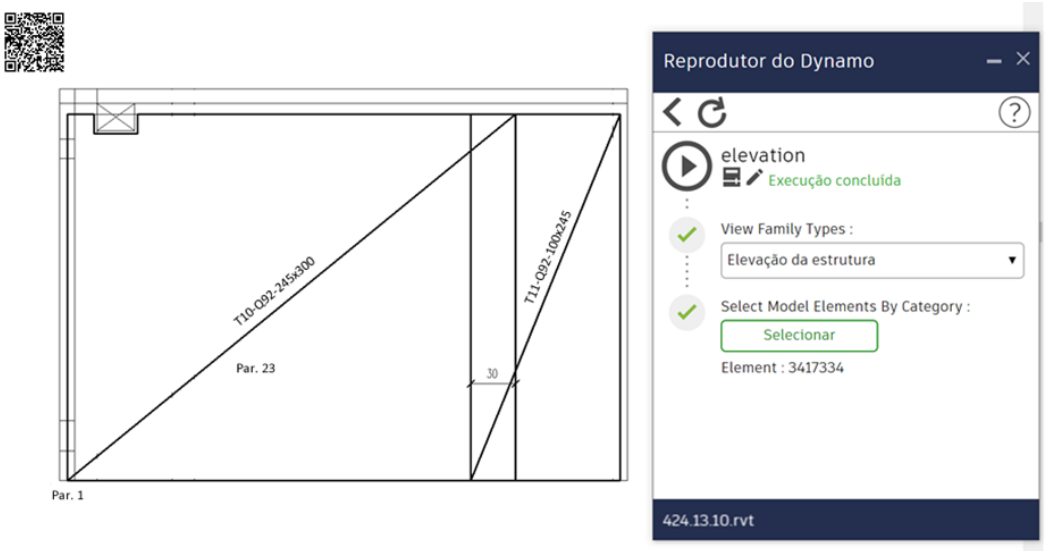

Figure 13: Demonstration of the automatic lifting routine menu.

4.4 Routines for Creating Screens in Walls

In order to enable the automation of the process of inserting ribbed welded screens in walls in the Autodesk Revit software, the algorithm used employed the Dynamo programming language. The input data is presented in Table 5.

Input data	Need	Nature
Host wall	Determine the element for insertion of the welded mesh sheet	Model Element
Screen designation	Inform the designation of the mesh to be inserted in the host wall	Element type
Screen location	Inform the location of the welded meshes	Element location
Screen distribution direction	Inform the distribution of the welded meshes	Lap Position

Table 5: Nature and necessity of input data for the routine for inserting screens into walls.

Using the input data, highlighted in green in Figure 21, the nodes previously programmed in the Dynamo language were created in the processing steps highlighted in pink in Figure 21, in order to recommend the creation of a vector, located in the center of the wall, responsible for including the screen along the entire length of the wall.

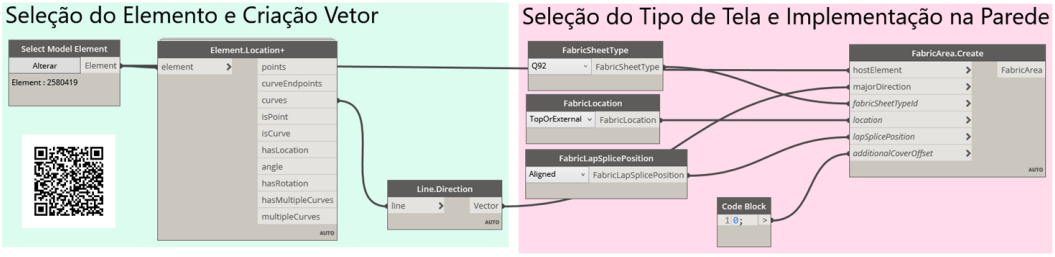

Figure 14: Routine for inserting structural screens in concrete walls.

The algorithm begins by identifying the perimeter of the wall where the screen will be distributed, in a step prior to the user indicating the wall itself. In addition to the type, orientation and direction of the screen distribution. Thus, after the previous selections, the algorithm creates the screen on the selected wall, as shown in Figure 15.

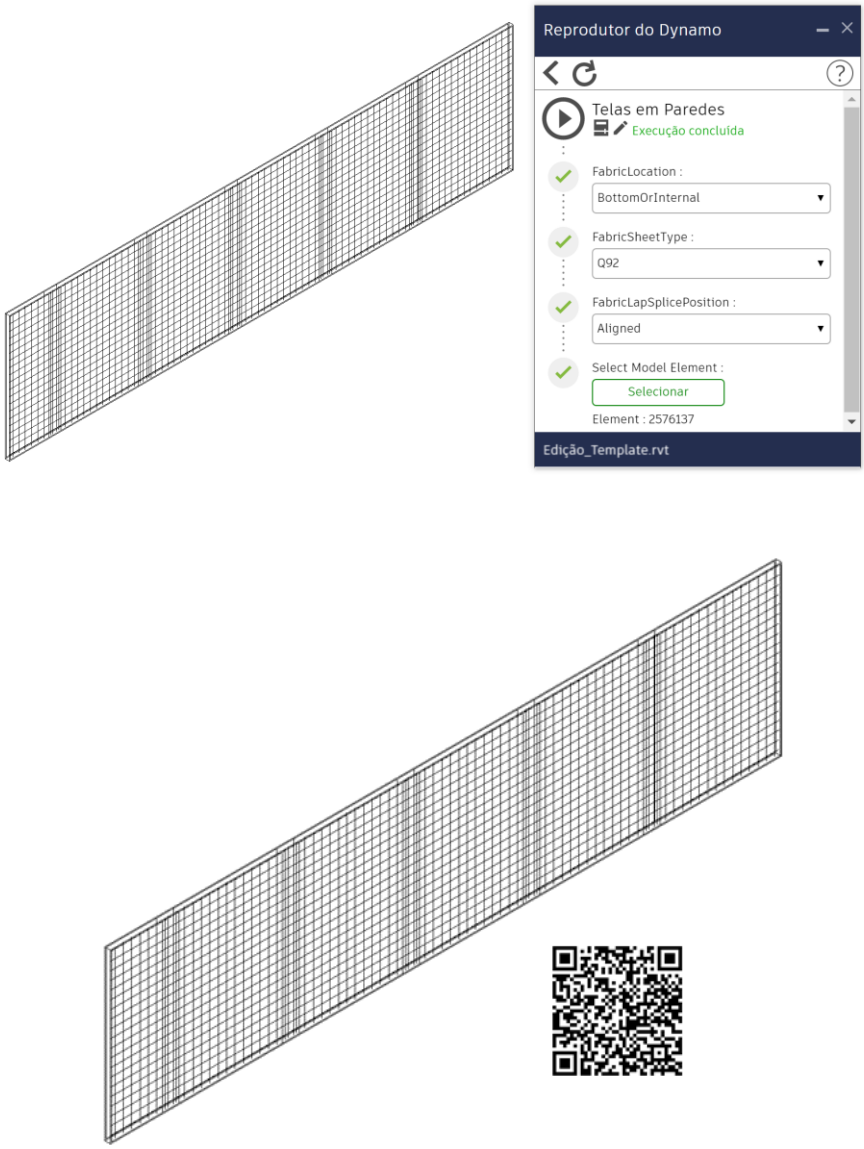

Figure 15: Performing screen routines on walls, with data and output input.

4.5 Routines for Creating Horizontal Reinforcements in Openings

In cases where the openings have horizontal dimensions greater than 40 cm, standard NBR 16055 [3] establishes the need for horizontal reinforcement on the upper and lower faces of the opening, with minimum area requirements of 0.5 cm².
In order to automate the process of inserting longitudinal reinforcement into openings in cast-in-place concrete walls in the Autodesk Revit software, the algorithm used the Dynamo programming language. The input data is shown in Table 6.

Input data	Need	Nature
Window	Determine the window element for inserting reinforcement	Model Element
Reinforcement Type	Inform the type of reinforcement to be inserted in the host wall	Element type

Table 6: Nature and necessity of input data for the horizontal reinforcement routine in openings.

The proposed routine starts by selecting the window that will receive the reinforcement and collecting the points that make up the ends of this element, internally and externally, as shown in the orange dots in Figure 14. From the end points, the Python routine creates the points used to obtain the curve that determines the shape of the reinforcement, at the bottom of the opening, to form a structural reinforcement.

It should be noted that the points that make up the reinforcement, highlighted in cyan in Figure 16, are inserted at a midpoint between the external and internal points of the window element, i.e. on the center line of the total wall thickness. This ensures that the reinforcement is inserted in the axis of the wall.

```python
import clr
clr.AddReference('RevitAPI')
from Autodesk.Revit.DB import *

# Função para converter unidades de pés para centímetros
def feet_to_cm(value_in_feet):
    return value_in_feet * 30.48

# Função para extrair as dimensões da janela do tipo de família e converter para centímetros
def get_window_dimensions_from_family_type_in_cm(window):
    # Acessa o tipo da família da janela
    familyType = window.Document.GetElement(window.GetTypeId())

    # Inicializa as variáveis de dimensão
    width = height = 0.0

    # Acessa e atribui as dimensões, se disponíveis
    if familyType:
        width_param = familyType.get_Parameter(BuiltInParameter.WINDOW_WIDTH)
        height_param = familyType.get_Parameter(BuiltInParameter.WINDOW_HEIGHT)

        # Verifica se os parâmetros são válidos e extrai os valores já convertidos para centímetros
        if width_param:
            width = feet_to_cm(width_param.AsDouble())
        if height_param:
            height = feet_to_cm(height_param.AsDouble())

    # Retorna as dimensões em centímetros
    return (width, height)

# Substitua IN[0] pelo elemento da janela selecionado
window = UnwrapElement(IN[0])
dimensions_cm = get_window_dimensions_from_family_type_in_cm(window)

# Define a saída do script para as dimensões obtidas em centímetros
OUT = dimensions_cm
```

Executar IronPython2

Figure 17: Python code for the routine for inserting structural reinforcements in windows.

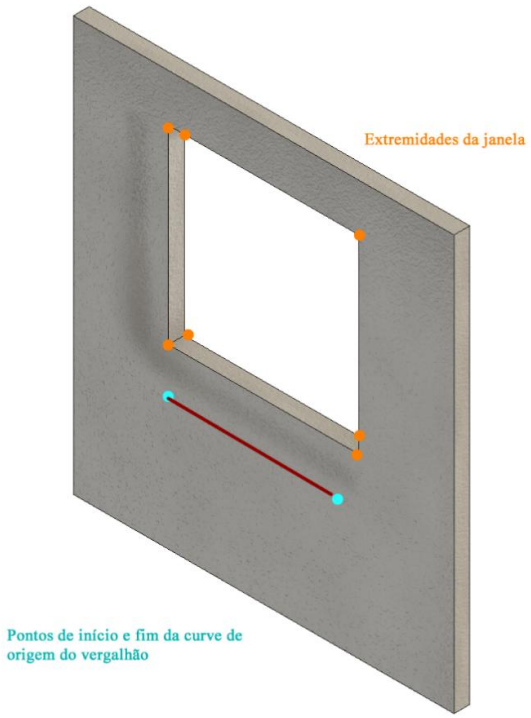

Figure 16: Description of the points of interest in the routine for inserting structural reinforcements in windows.

```python
import clr
clr.AddReference('RevitAPI')
from Autodesk.Revit.DB import *

# Função para converter unidades de pés para centímetros
def feet_to_cm(value_in_feet):
    return value_in_feet * 30.48

# Função para extrair as dimensões da janela do tipo de família e converter para centímetros
def get_window_dimensions_from_family_type_in_cm(window):
    # Acessa o tipo da família da janela
    familyType = window.Document.GetElement(window.GetTypeId())

    # Inicializa as variáveis de dimensão
    width = height = 0.0

    # Acessa e atribui as dimensões, se disponíveis
    if familyType:
        width_param = familyType.get_Parameter
        (BuiltInParameter.WINDOW_WIDTH)
        height_param = familyType.get_Parameter
        (BuiltInParameter.WINDOW_HEIGHT)

        # Verifica se os parâmetros são válidos e extrai os valores já convertidos para centímetros
        if width_param:
            width = feet_to_cm(width_param.AsDouble())
        if height_param:
            height = feet_to_cm(height_param.AsDouble())

    # Retorna as dimensões em centímetros
    return (width, height)

# Substitua IN[0] pelo elemento da janela selecionado
window = UnwrapElement(IN[0])
dimensions_cm = get_window_dimensions_from_family_type_in_cm(window)

# Define a saída do script para as dimensões obtidas em centímetros
OUT = dimensions_cm
```

Executar IronPython2

Figure 17: Python code for the routine for inserting structural reinforcements in windows.

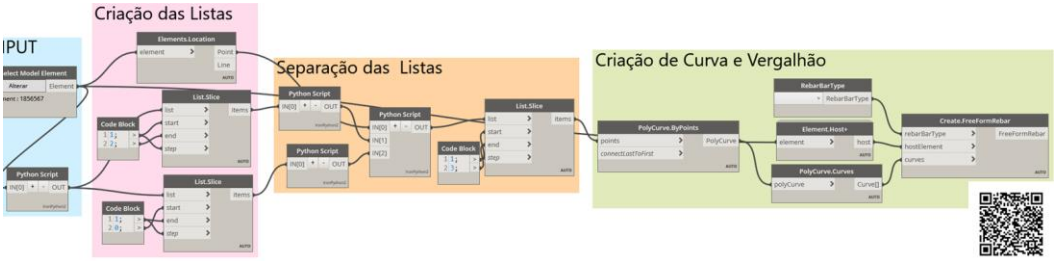

Figure 18: Dynamo algorithm in the routine for inserting structural reinforcements in windows.

The routine shown in Figure 18 is displayed to the user as shown in Figure 19, so that the inputs for executing the actions described can be selected. Thus, after selection, Revit automatically creates the reinforcement, without the user needing to execute complex curve drawing routines, host element selection, among other requirements necessary to execute this action manually, that is, without the help of the routines developed in this work.

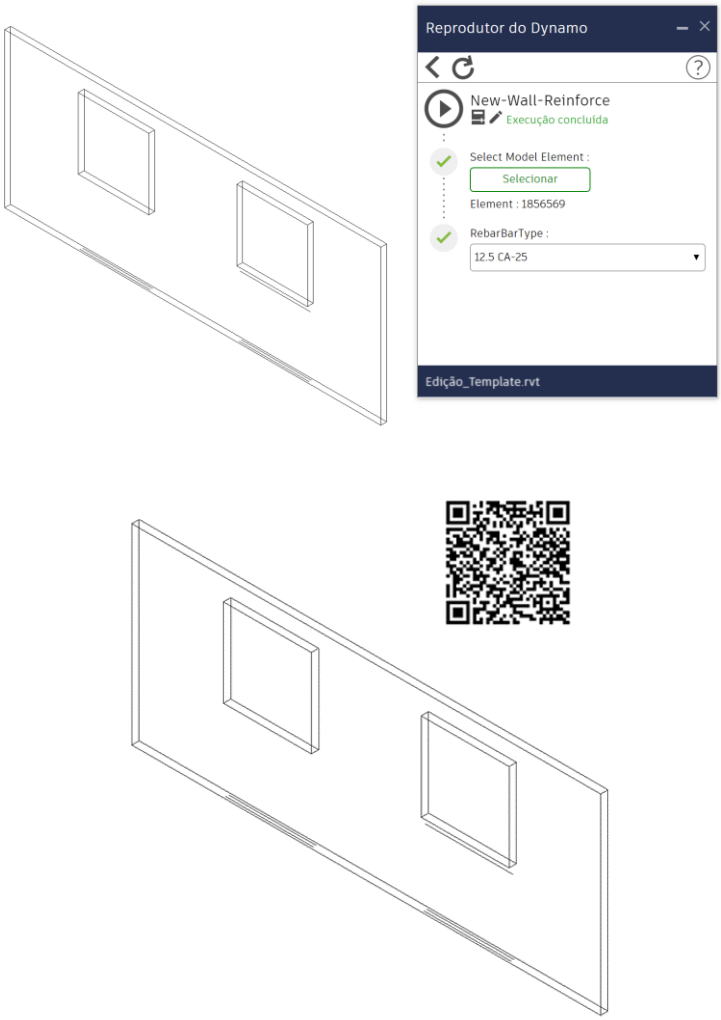

Figure 19: Execution of reinforcement in windows, with data and output input.

4.6 Access Menu

As described in the objectives and methodologies assigned to facilitated access to the developed routines, the menu created through the NonicaTab PRO plugin is located next to the categories already present in the Autodesk Revit software. The icons created are shown in Figure 20.

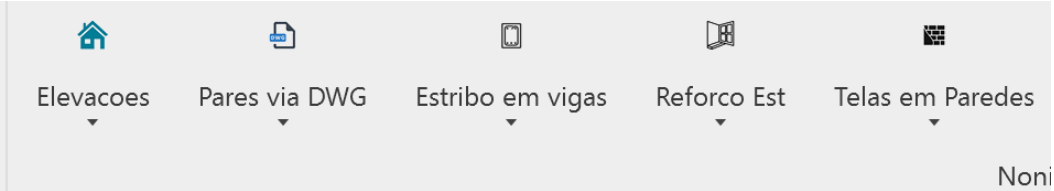

Figure 20: Routines menu.

Thus, as shown in Figure 15, each of the proposed routines was given the following intuitive and simplified names: elevations; walls via DWG; stirrups in beams; structural reinforcement and screens in walls. In addition, each routine has been given a personalized icon so that users can familiarize themselves with its visual characteristics.

It is possible to argue, from the perspective of the results obtained, that the objective of bringing together and organizing the routines created was achieved, so that they all adapted to insertion in the Nonica plugin and, therefore, constituted the menu displayed in a similar way to the other Autodesk Revit functionalities, inserted in the native interface of this software.

It is worth noting that as the number of routines proposed in this study does not exceed the number of positions available for insertion into menus, it was not necessary to group the routines by subject or objective. In this way, the routines are displayed independently, although the plugin used also has the option of grouping the routines into large groups, which was not necessary in the implementation described in this study.

5 Conclusions

This work shows that, although the BIM methodology is a powerful tool in the architecture engineering and construction (AEC) industry, the automation of processes within this methodology is extremely important in terms of time, quality and standardization of projects.

Despite the time invested in creating routines such as those presented in this work, the benefits and return on application are remarkable. The algorithms developed result in optimization with a consequent gain in productivity, quality and standardization of *cast-in-place* concrete wall projects.

The routines developed aim to automate the execution of repetitive tasks in structural projects, guaranteeing speed and precision. This was achieved by taking advantage of the inherent advantages of the BIM methodology and maximizing the benefits of project execution and analysis in software such as Autodesk Revit.

During the course of this work, it was also possible to conclude that organizing the routines to carry out refactorings and alterations is an extremely important action,

because they can be easily adapted to the different realities of designers working in structural design offices.
In addition, the proposed routines need to be organized and brought together in an Autodesk Revit menu so that they can be executed easily, in order to provide an incentive and visual invitation for designers to intensify their use of these new tools. Like Autodesk Revit itself, they require a certain amount of training and a period of adaptation in the places where they are implemented, as the changes modify the design of projects already experienced by professionals and students.
It should be noted that the use of the languages presented is a valuable opportunity for project automation, not just in structural design, but in all fields of study in the construction industry.
It is worth considering that this work aims to help the public interested in optimizing the process of producing virtual construction models and to encourage researchers to develop new processes, create artifacts and contribute to the continuous improvement of this field of research.

References

[1] C. Eastman, P. Teicholz, R. Sacks, and K. Liston, *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors*, 2nd ed. Hoboken, NJ, USA: Wiley, 2008. https://doi.org/10.1002/9780470261309.

[2] R. Monge, A. V. Mayor, and J. B. R. Silva, "A construção de um sistema de sucesso," Concr. Constr., no. 90, p. 42, Apr.–Jun. 2018. Accessed: Nov. 13, 2024. [Online]. Available: https://ibracon.org.br/Site_revista/Concreto_Construcoes/ebook/edicao90/files/assets/basic-html/page42.html

[3] Associação Brasileira de Normas Técnicas (ABNT), *NBR 16055*, 2012.

[4] R. S. Farias, "Análise estrutural de edifícios de paredes de concreto com a incorporação da interação solo-estrutura e das ações evolutivas," M.S. thesis, Dept. Eng. Estrutural, Univ. São Paulo, São Carlos, SP, Brazil, 2018. [Online]. Available: http://www.teses.usp.br/teses/disponiveis/18/18134/tde-19022019-104219

Author contributions: MCS: wrote the full article and developed the codes; EAO: conducted the review and provided guidance for the work.