ES10679

An MEP Engineer's Guide to Dynamo

Andrew Duncan
Arup, BIM Manager

Andrei Capraru
Arup, Mechanical Engineer

Learning Objectives

- Discover enough about Dynamo to get started
- Learn how to link Revit Space information to Revit families for use in calculations
- Learn how to use Dynamo as an engine for performing MEP calculations such as determining heating loads or flow rates
- Learn how to pull data from Excel into Revit via Dynamo to inform calculations

Description

People often associate Dynamo software with designing complex parametric geometry; but Dynamo software is not just a tool for creating funky shapes, it's a whiz at processing all kinds of data. This class will demonstrate various examples of how Dynamo software has been applied to common MEP (mechanical, electrical and plumbing) engineering tasks to make them more efficient or more accurate. We will cover linking Revit Space naming utility information to Revit software families, using Dynamo software as an engine for MEP calculations, using Dynamo software to give you visual feedback on how hard equipment is being asked to work in your design, and using Microsoft Excel as a source of data for much of the above. You also need not worry if you are completely new to Dynamo software (most MEP folk will be), as we will cover enough basics to get you going before we dive into the advanced stuff.

Your AU Experts

Andrew Duncan *is MEP BIM Manager for a large Building Engineering team based in London. This role requires that he manages all aspects of the group's building information modelling for building services disciplines, from direct production through to defining processes and standards. He has worked on a wide variety of new build and refurbishment projects whilst with Arup and has helped to deliver designs for buildings located all over the globe including the Middle East, South America, Far East Asia, Europe and the United Kingdom.*
Andrei Capraru *is a Mechanical Engineer working within multi-disciplinary design teams in London. Since joining Arup, Andrei has been involved in projects that had BIM (Building Information Modelling) as their core deliverables. He has design experience in mission critical, educational, healthcare, commercial and custodial sectors. Andrei has been the lead mechanical engineers on the first pilot project for the UK Government BIM Mandate which has been awarded BIM Project of the Year in London and the South Easy by Construction Excellence.*

Contents

Workflow script 1 – Space Parameter Data Population

To get started we will look at the first script in our series, which allows us to populate space data from Excel into Revit Spaces. The spaces in Revit can hold parameters that define their identity, such as Space Name, Number, Area, as well as other engineering data such as Heating Load, Average Illuminance, etc. For this exercise you can use the Revit standard Space Parameters or you can create your own.

Throughout this hand-out and presentation you will see a number of parameters that cannot be found in the Revit standard files. These Parameters form part of the Arup Standard Template, however the Dynamo scripts are not dependent on these names, as it will be explained later on.

1. Script 1 Workflow

Step 1. Populate Excel with engineering data

The Excel spreadsheet will represent the database for the engineering data that hold the Space Type information for the project.

The main driver of the script will be located in the first column of the excel spreadsheet and it needs to have the exact name as the Revit Space Parameter that defines the type of the space. For this example we have used a fictive office building that has cellular offices, open space offices, restaurants, reception, etc.

K6

	A	B	C	D
1	**Space_Type**	**Space_Mechanical_Ventilation_Type**	**Space_Mechanical_Heating_Type**	**Space_Mechanical_Cooling_Type**
2	Atrium	Natural Ventilation	Underfloor Heating	None
3	Cellular Office	Natural Ventilation	Radiant Panel	None
4	Conference Centre	Mixed Mode Ventilation	Fan Coil Units	Fan Coil Units
5	Corridor	Mechanical Ventilation	Radiators	None
6	Gym	Mechanical Ventilation	Fan Coil Units	Fan Coil Units
7	Open Office	Mechanical Ventilation	Fan Coil Units	Fan Coil Units
8	Plantroom	Natural Ventilation	Electric Heater	None
9	Reception	Mixed Mode Ventilation	Underfloor Heating	None
10	Restaurant	Mechanical Ventilation	Fan Coil Units	Fan Coil Units
11	Storage	Mechanical Ventilation	Electric Heater	None
12	Toilets	Mechanical Ventilation	Radiators	None
13				

Figure 1 Excel - Space information database example

The most important thing when setting up the Excel spreadsheet is to ensure that the Parameter Names and the Space_Type values match exactly the name of the Parameters in Revit.

In this example we have defined how the spaces are treated from a mechanical services point of view. For example, an Open office will be served by Mechanical Ventilation (i.e. Air Handling Units) which will provide minimum fresh air to the space and the heating and cooling of the space will be done via Fan Coil Units.

The spaces in Revit will need to be populated with the desired Space types based on the building's configuration. You can do this by either selecting each individual space and populating the Space_Type

parameter, or using schedules or other tools that allows you to mass populate information. This will be the only bit of manual labour that is required for the rest of the class.

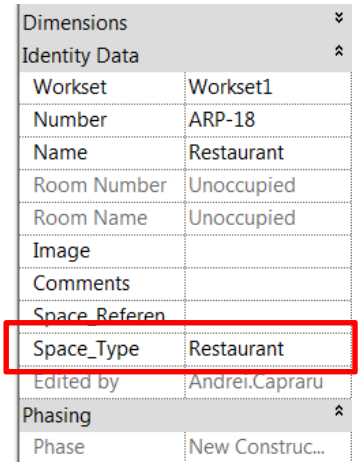


Figure 2 Revit Space Properties - Space_Type data population

Step 2. Run the Dynamo script

Once the Excel data has been setup and populated and the Revit Space_Type Parameter filled in for all spaces, we can run the Dynamo script. Once the run is complete, your Revit spaces will be populated with the information from excel. You can now start producing your Colored Drawings, Generate Engineering Data or Produce Room Data sheets within Revit.

It is that easy! You can either run the script on manual mode and run it after you've done all the changes or have it run automatically in the background capturing the changes live as you modify your excel spreadsheet. Depending on the size of the model and the number of spaces this might slow down your system.

2. The script description

Now that we've covered what the script does and covered the workflow, let's have a look at what is happening in Dynamo and how the script was built. We will run through the script in small chunks and discuss all the more important nodes.

The Space Population Script is split into two main branches:

- The Excel data branch which reads the data from excel and does all the processing required to bring the Lists in the correct format
- The Revit Spaces branch which reads the spaces information and processes the Lists to have a similar List format as the Excel branch

A third branch deals with the error handling of the script, informing on which Space Types have been successfully populate and if there are any inconsistencies between Revit and Excel with regards to the Space Type names.

2.1. Read excel data

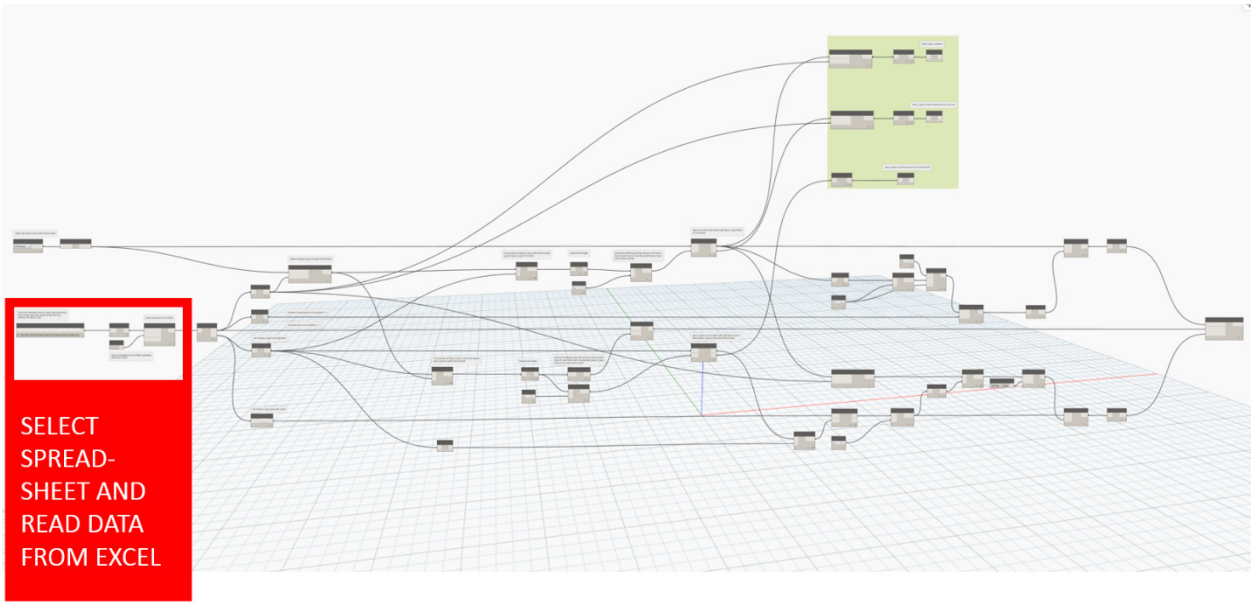


Figure 3 Script overlook Step 1

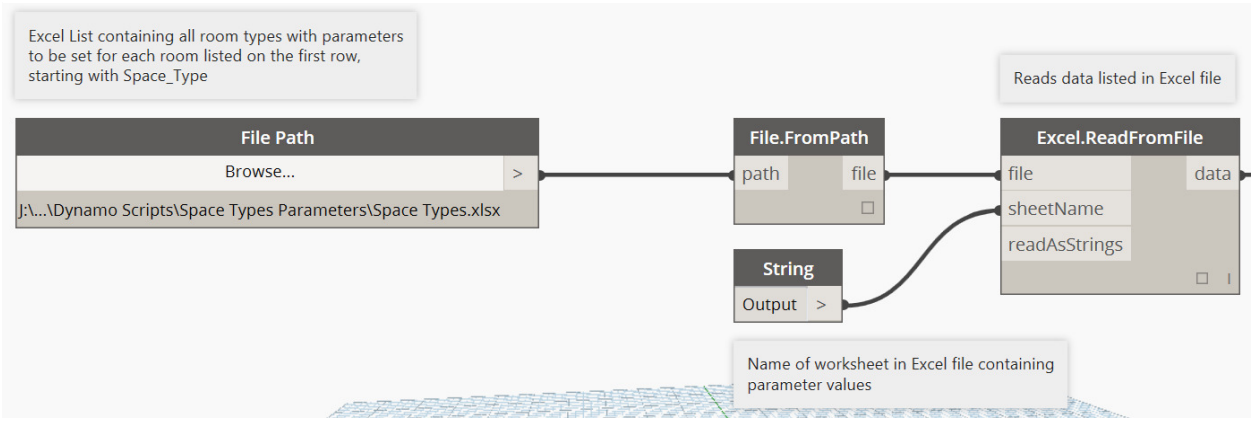


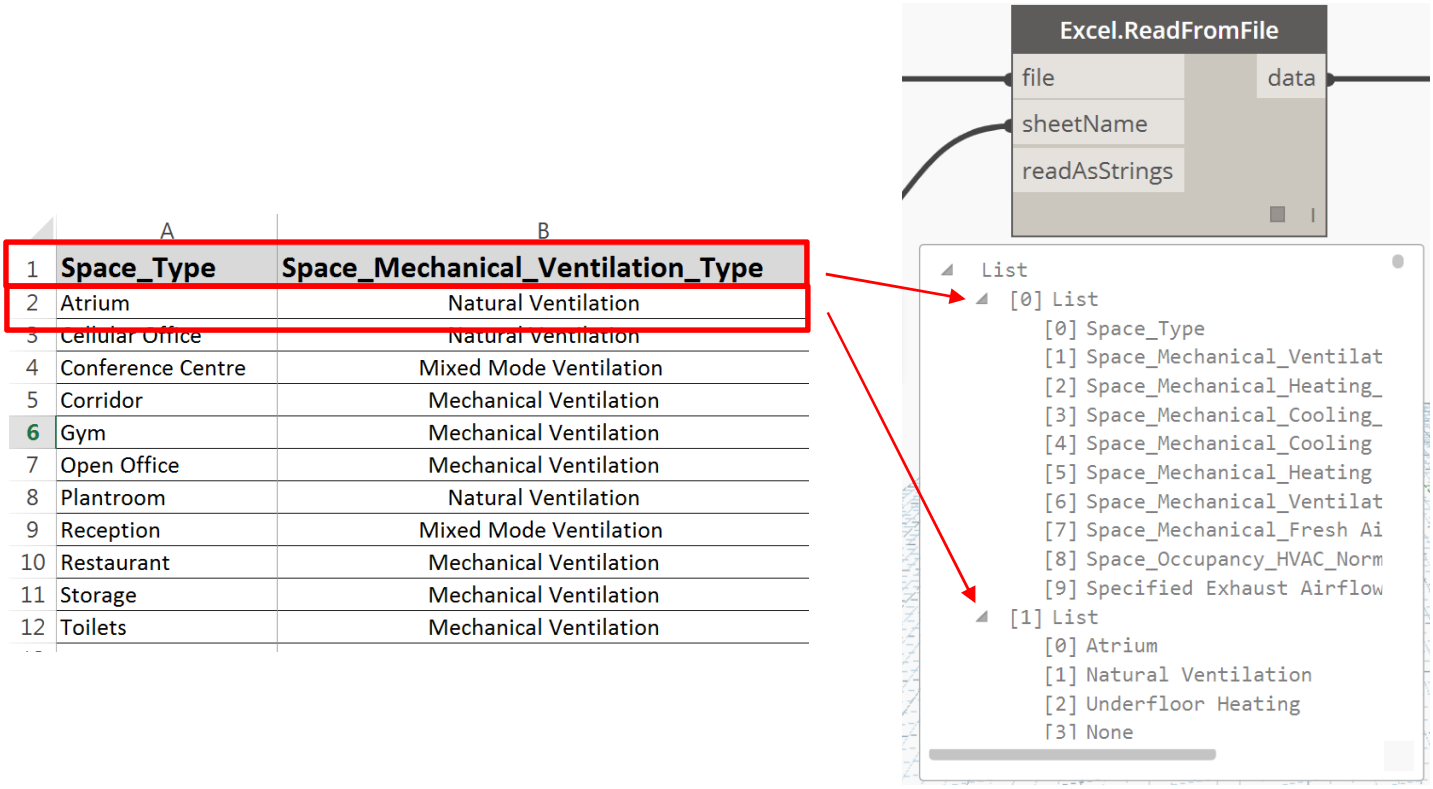
Figure 4 Select spreadsheet and read excel data

The first node in the script defines the Excel file path that contains the Space Type data. The node comes with a Browse function to help the user find the required file easier.

The next node in line is the *File.FromPath* node which takes the file path information from the previous node and generates a file object. The file object is then passed over to the *Excel.ReadFromFile* node which will read the data from the Output Excel spreadsheet and will store the data in the form of a List.

2.2. Process the excel data

By default the *Excel.ReadFromFile* node will create a List with the Excel data contained in the specified spreadsheet. The main List will contain a set of sub-Lists (i.e [0] List, [1] List, etc) that hold the data for each individual row in excel.



	A	B
1	**Space_Type**	**Space_Mechanical_Ventilation_Type**
2	Atrium	Natural Ventilation
3	Cellular Office	Natural Ventilation
4	Conference Centre	Mixed Mode Ventilation
5	Corridor	Mechanical Ventilation
6	Gym	Mechanical Ventilation
7	Open Office	Mechanical Ventilation
8	Plantroom	Natural Ventilation
9	Reception	Mixed Mode Ventilation
10	Restaurant	Mechanical Ventilation
11	Storage	Mechanical Ventilation
12	Toilets	Mechanical Ventilation

Figure 5 Excel.ReadFromFile

If we have a closer look at sub-List [L0] List we can see that it contains all the information contained in the row 1 cells in the exact order they've been entered in the table.

C13

	A	B	C	D	E	F
1	Space_Type	Space_Mechanical_Ventilation_Type	Space_Mechanical_Heating_Type	Space_Mechanical_Cooling_Type	Space_Mechanical_Cooling Load_Per Area	Space_Mechanical_Heating Load_Per Are
2	Atrium	Natural Ventilation	Underfloor Heating	None	80	120
3	Cellular Office	Natural Ventilation	Radiant Panel	None	80	120
4	Conference Centre	Mixed Mode Ventilation	Fan Coil Units	Fan Coil Units	80	120
5	Corridor	Mechanical Ventilation	Radiators	None	80	0
6	Gym	Mechanical Ventilation	Fan Coil Units	Fan Coil Units	80	120
7	Open Office	Mechanical Ventilation	Fan Coil Units	Fan Coil Units	80	120
8	Plantroom	Natural Ventilation	Electric Heater	None	80	0
9	Reception	Mixed Mode Ventilation	Underfloor Heating	None	80	120
10	Restaurant	Mechanical Ventilation	Fan Coil Units	Fan Coil Units	80	120
11	Storage	Mechanical Ventilation	Electric Heater	None	80	0

Figure 6 Excel Space Type database

Now that we understand the structure of the List generated by the *Excel.ReadFromFile* node we can start looking at ways to manipulate the List to get it into a format suitable for combining with the information that we are extracting from Revit.

2.3. List.Deconstruct Node

The deconstruct node will split the initial List generated by the *Excel.ReadFromFile* node into two new separate Lists. The *First* List will contain the elements in the [0] List, and the *Rest* List will contain the remaining elements. In this way we managed to separate the Excel table header and the data within the table into two separate Lists.

The Count node will count the number of elements in [0] List, and the value will be used later on in the script to inform on the number of Parameters to be updated.

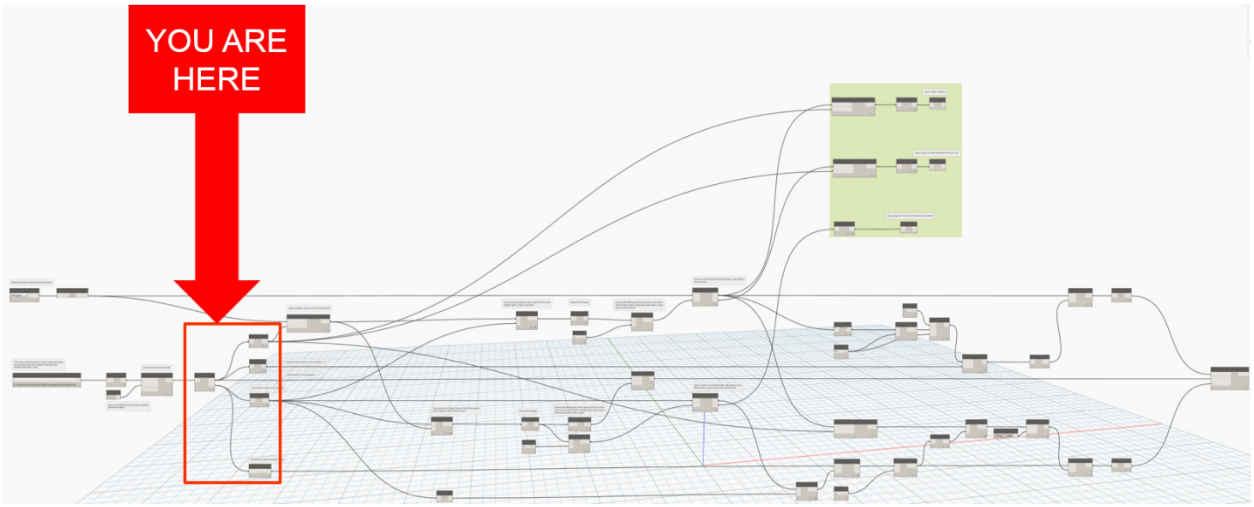


Figure 7 List.Deconstruct location in the script

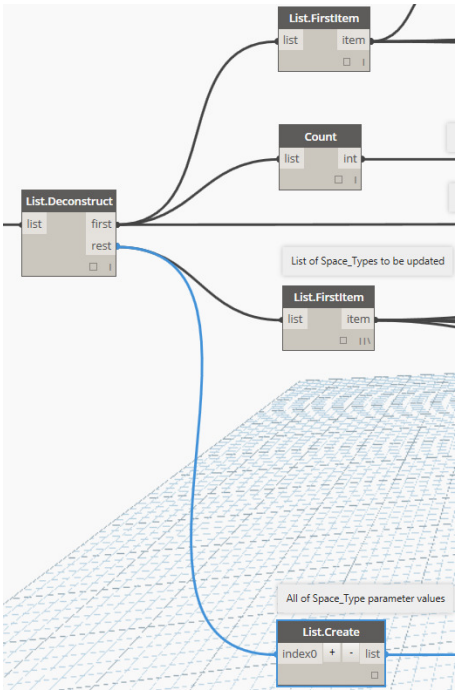


Figure 8 List.Deconstruct outputs

The third node from the top will extract the first sub-List from the *Rest* List. The respond will correspond with thee values contained on the second row of the Excel file. In our case it will hold the parameters assigned to the Atrium space type.

2.4. Read the Revit space data

At this point we have extracted the data out of excel and split it in two Lists, one that contains the header of the excel table and the second List that contains remainder of the table. In Revit terminology, we now have the first List that contains the Space Parameter Names and the second List that contains the Space Parameter Values. In order to proceed further with our Excel manipulation we need to look at the data extracted from Revit and compare the Space Parameter Names between Revit and Excel to check they match.

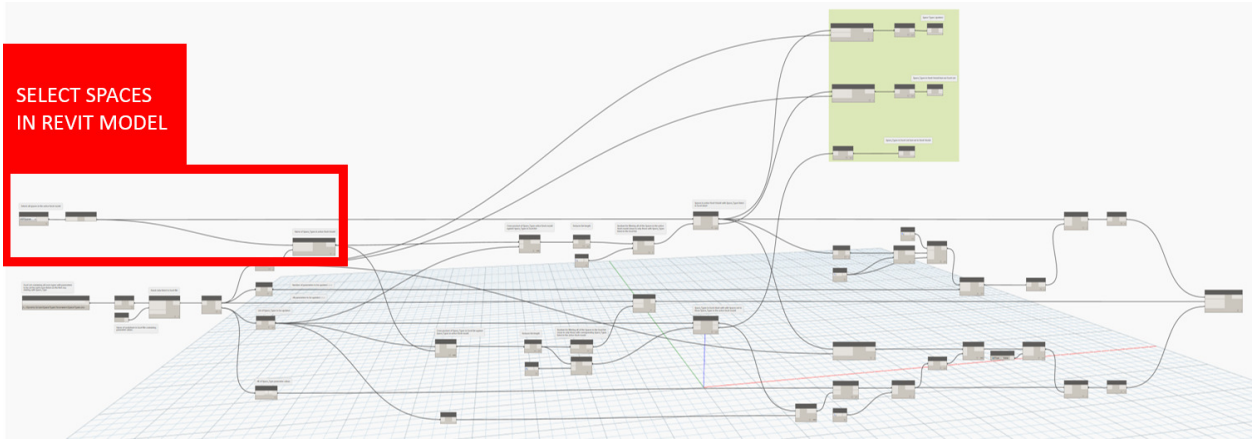


Figure 9 Revit data interrogation

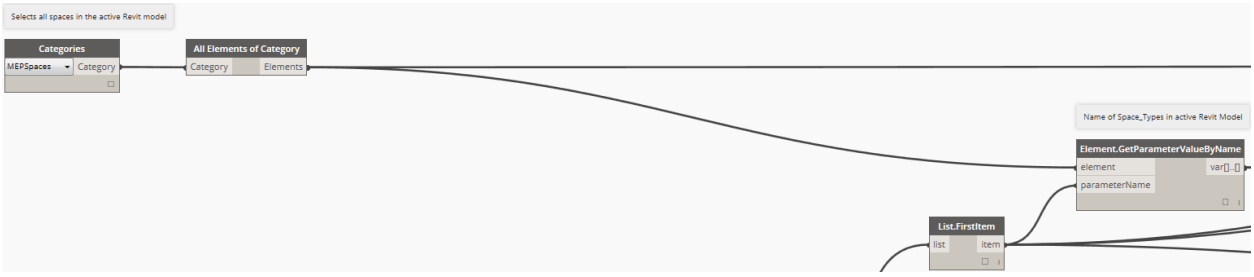


Figure 10 Access Revit MEPSpaces and read their Parameters

To access the Space data from Revit we used the Categories node. This node allows you to select any of the Revit build-in Categories. For our script we are interested in the MEPSpaces category, which will be passed into the *All Elements of Category* node. This node will selects all the MEP Spaces from the model.

We now have access to all the spaces in Revit. In order to interrogate their parameters we need to use the *Element.GetParameterValueByName*. The node syntax requires a Parameter Name or a List of Parameter Names as its secondary input. For this input we can now use the List.FirstItem node described in section 2.2 which contains the Excel header names. The output of this node is a List of all

the spaces in the Revit model and their corresponding Space Type. For example the [0] Reception List element matches the Revit Space no 1 which has the Space_Type Parameter = Reception.

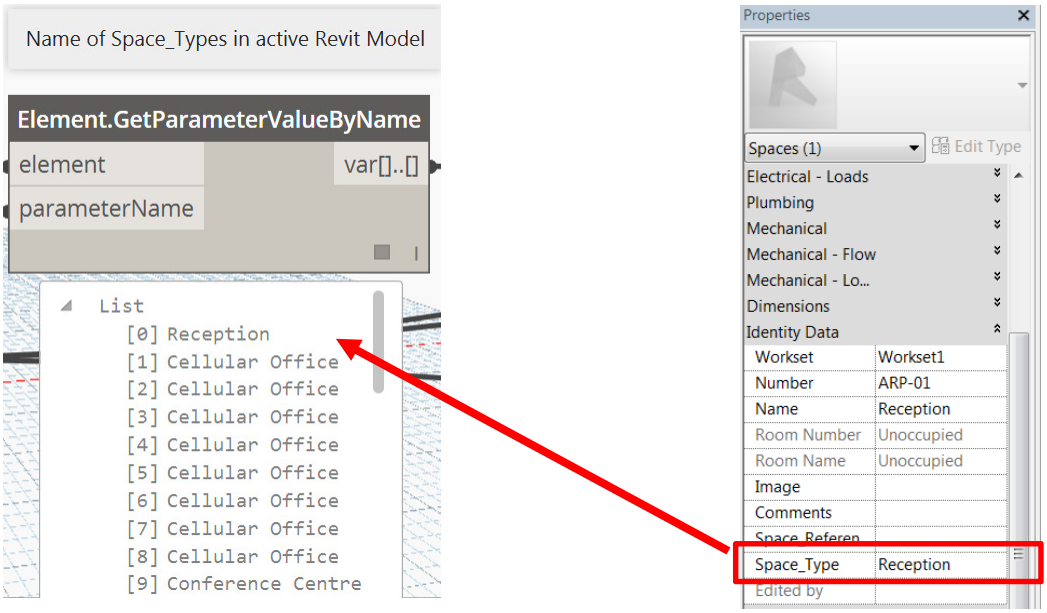


Figure 11 Element.GetParameterValueByName output example

2.5. Compare Excel Space Types with Revit Space Types

2.5.1. Excel Data Filtering

Now that we have a List of Space Type values from Excel and a List with all the Revit Spaces' Space_Type Parameter values we can use a "==" node to compare the two Lists. The node will generate a List of true and false values. For this node Ensure the Lacing Type is set to Cross Product as we want to check how

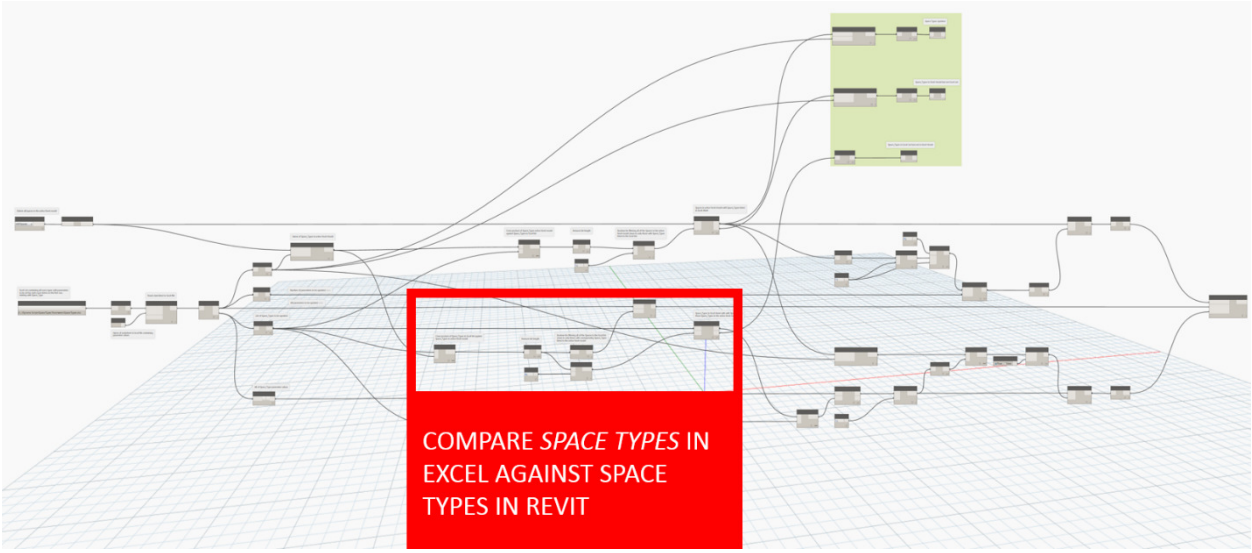


Figure 12 Compare Revit data with Excel Data using Space Type the criteria

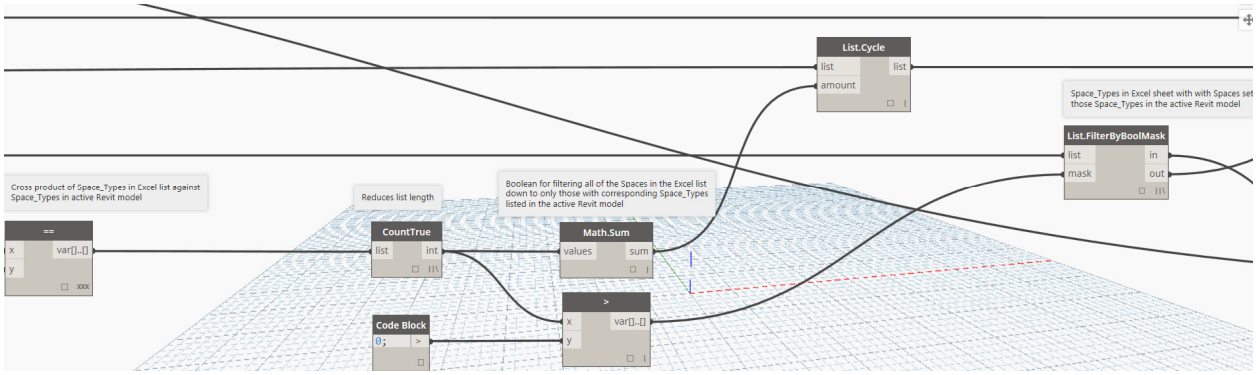


Figure 13 Revit vs Excel Space Type Comparison script

many Revit Spaces have the Restaurant Space Type value, how many are Open Offices and so on. For more details on Lacing please refer to the Class Slide Presentation.

The *CountTrue* node will quantify the number of spaces that have been assigned against a certain Space Type value. For example we can determine the number of Restaurant spaces or the number of Toilet spaces in the building.

The *Math.Sum* node sums up all the values in the List generated by the previous node. The result represents the number of spaces in Revit that have one of the Excel Space_Type values. For this script, the result from this note tells us that 85 Revit spaces will be populated at the end of the script.

The greater than 0 node will helps identify the positions in the List that don't match a Space_Type. In our case we have quite a number of false positions due to the empty cells in Excel. An empty but formatted cell (i.e. has a border applied to it) in excel is recorded as a null value by Dynamo. Empty and unformatted cells will be ignored. So if you end up with too many null values in the *Read.ValuesFromExcel* node, you might want to consider removing the formatting on some of the empty rows.

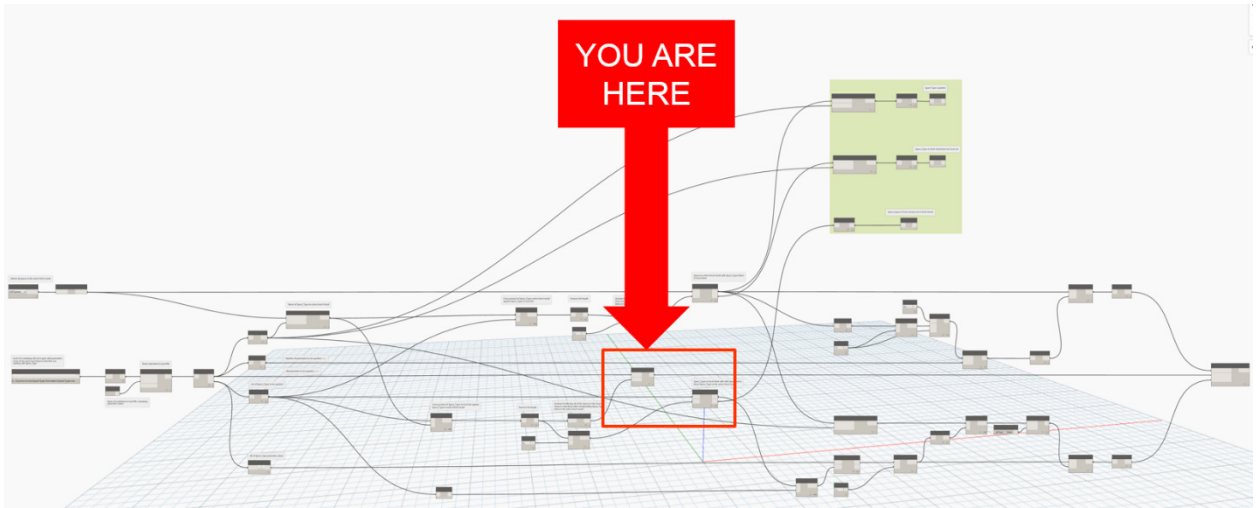


Figure 14 List.Cycle node position in the script

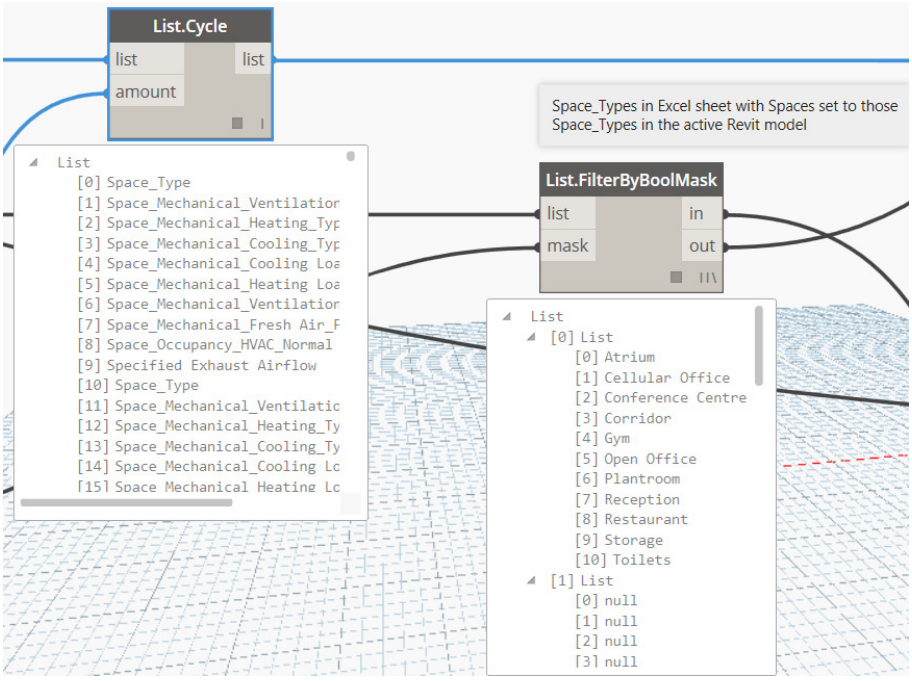


Figure 15 List.Cycle and List.FilterByBoolMask Example

The *List.Cycle* node takes as input the First List generated at the beginning of the script that contains the first row of the Excel file or the Parameter Names that are to be populated in Revit. The second input is the *Math.Sum* node described earlier. The result of the node will be a List that has as values the Excel Row 1 data repeated 85 times, which is the number of spaces in Revit that will be populated at the end of the script. The resulting List will feed into the final Revit data writing node, which will be discussed in more details further down.

List.FilterByBoolMask filters the original List to only those Space Types present in the Revit model. For this example the second List is full of null items as all the Revit Space_Types are found in the Excel table. The In node output stores the first List or the items that made the criteria, the Out node output stores the element that didn't meet the criteria. The Out List is pushed to the error handling part of the script to inform the user on the discrepancies between Excel and Revit Space Type data.

2.5.2. Revit Data Filtering

The last action we have done on the Revit data was to extract all the Space Parameter values. We now need to process them in a similar manner to the ones in excel, however the amount of formatting required is minimal as we have focused on getting the Excel List in a format similar to the Revit output. The only task required for the Revit data is to eliminate the Lists that contain Space_Type information that don't match the excel Space Types.

The process is similar to the one discussed earlier, and consists broadly of comparing the Space_Type Parameter data with the Excel Space Type information which generates a true/false set of Lists used then as masking to eliminate the data that doesn't meet the criteria.

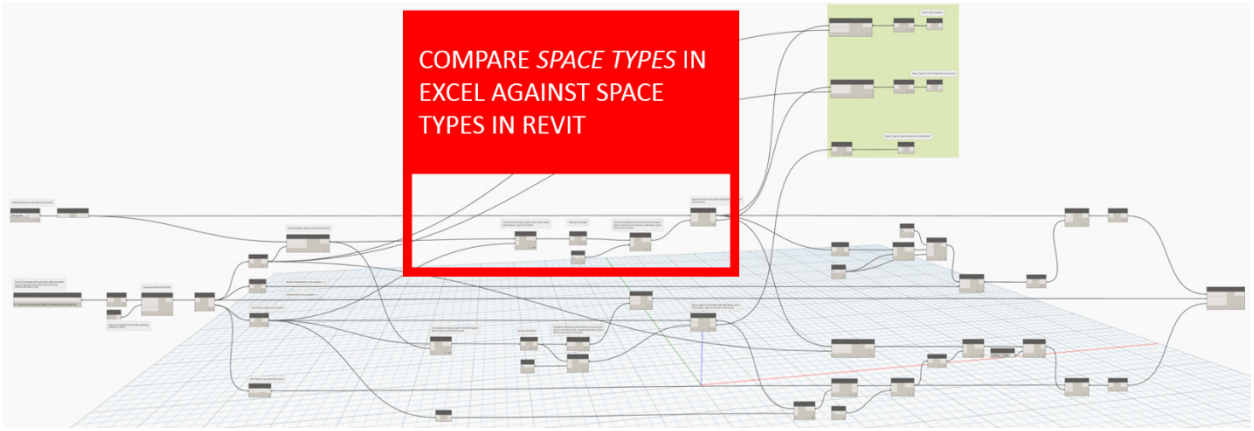


Figure 16 Revit Data Filtering

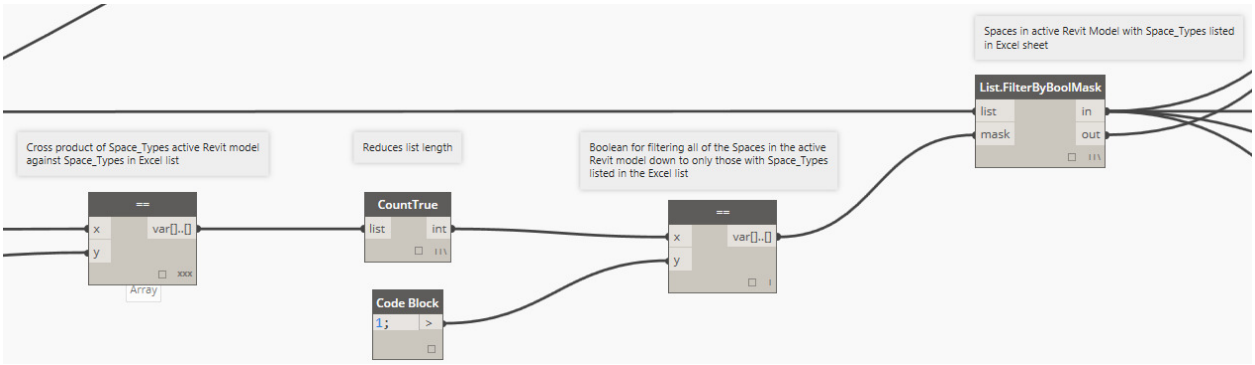


Figure 17 Dynamo nodes used for Revit data filtering

2.6. Prepare Lists for Revit Parameter writing

The next step in our script is to arrange the Revit Spaces List, the Excel Parameter List and the Excel Values List in the same format such that a short Lacing option can be used the Element.SetParameterByName which will be discussed in section 2.7.

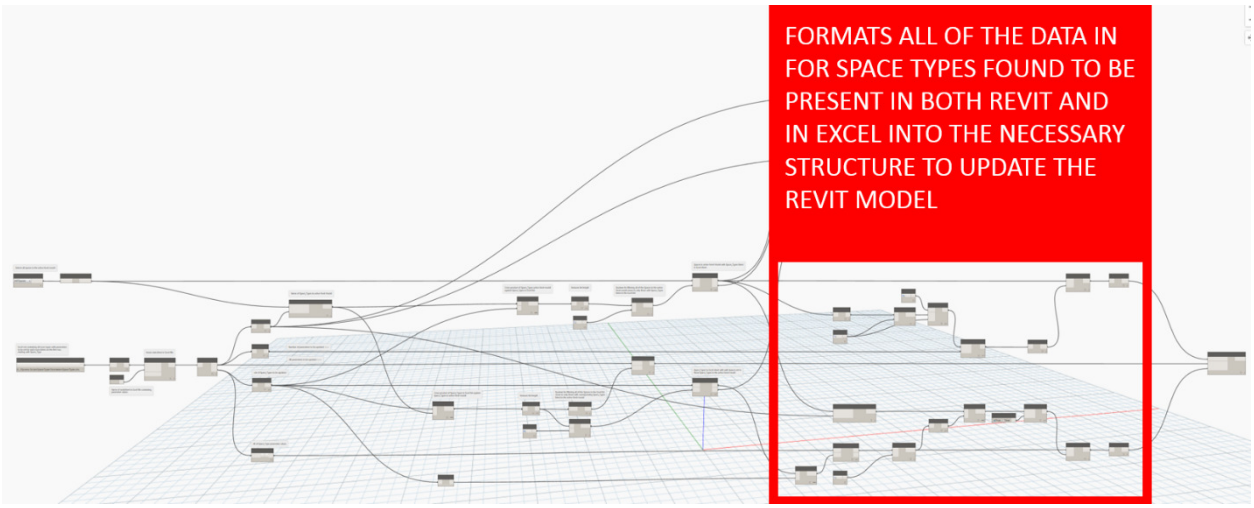


Figure 18 Revit and Excel List formatting

2.6.1. Revit side List manipulation

In section 2.4 we've described the List.Cycle node where we repeated the Parameters which we wanted populated into Revit by the number of spaces in Revit. To do this we need to create a List that repeats the space element to match the number of parameters we want to populate. For example if we had only 3 parameters to populate then the space elements would've been repeated 3 times.

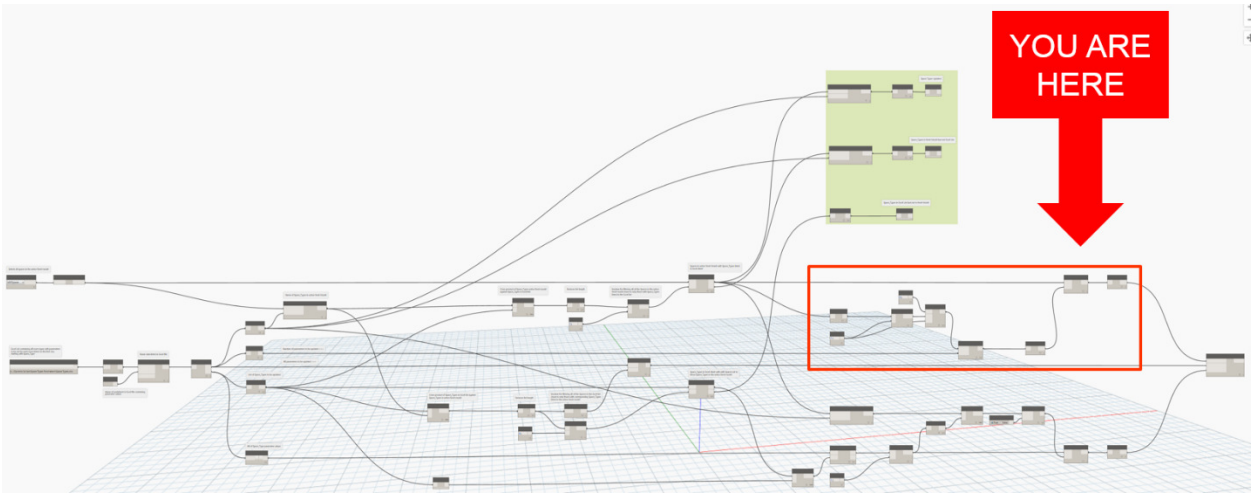


Figure 19 Revit List side data manipulation

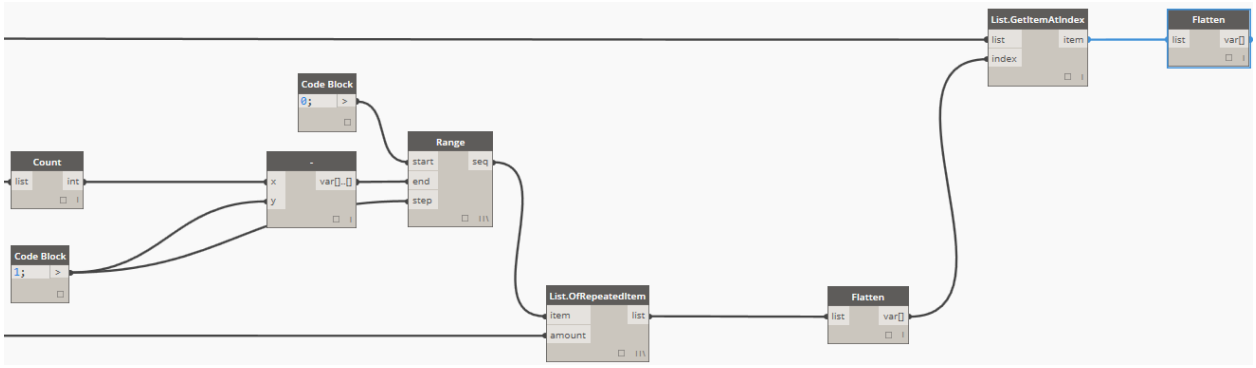


Figure 20 Dynamo nodes

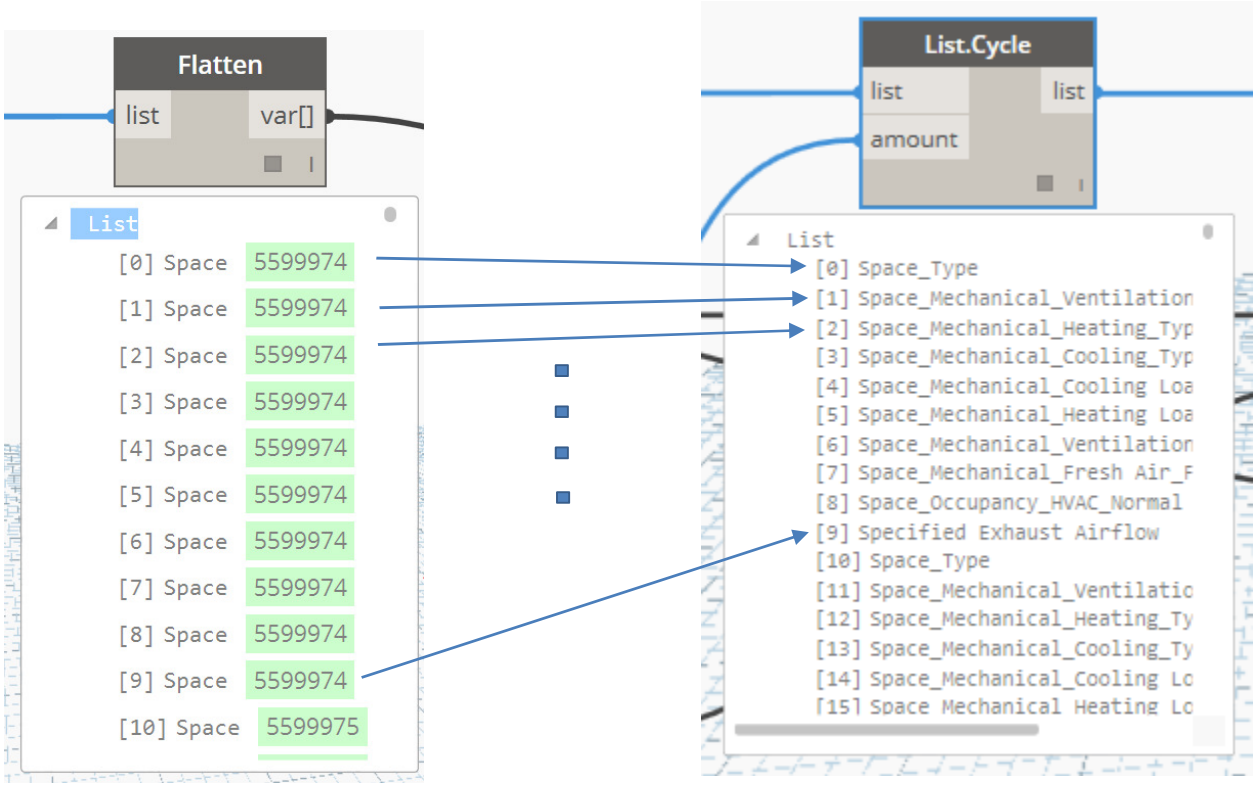


Figure 21 Repeat the Space data List to match number of parameter to be populated

2.6.2. Excel values List manipulation

We spoke extensively about the Space Type values and about the Excel Table Header which defines the Revit Space Parameters that are going to be populated. We also described at length how the data contained by these two Lists has been manipulated and formatted, however we haven't yet spoken about the actual values that will given to the Revit Space Parameter. The List generated at the beginning of this script and which resulted from the Rest List out of the List.Deconstruct node will need to be

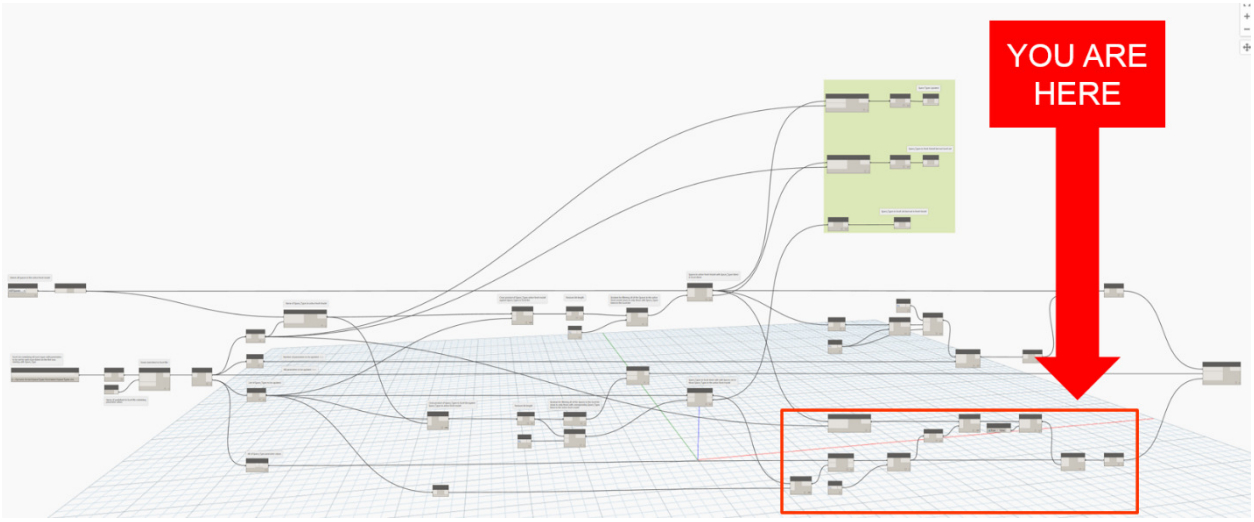


Figure 22 Revit List side data manipulation

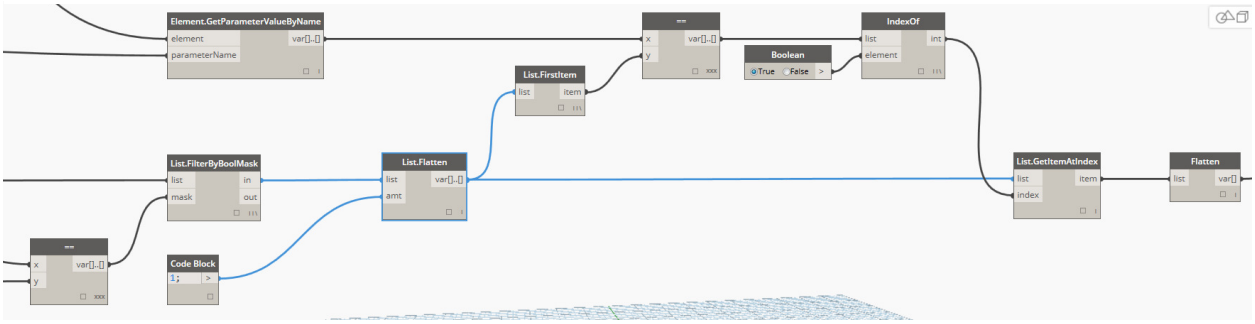
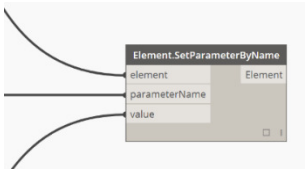


Figure 23 Dynamo script for re-arranging the Excel value List

repeated and formatted such that it meets the Revit Space List, i.e. each row will need to be repeated 85 times to match the Revit Space List formatting. The process is similar to the one described in the previous section.

2.7. Write Excel Data into Revit



The last node in our script is the *Element.SetParameterByName* node which takes as input the space List, the *ParameterName* List and the Excel values List which we've prepared in the previous sections. Now that we have all three elements formatted in the same manner we can go run the script. As the 3 Lists are identical in format, the short lacing setting will be required for the node.

Workflow script 2 – MEP Design Data Calculations

The MEP Design Data Calculations script helps to generate Heating, Cooling and Ventilation loads data for the spaces based on early stage design information. The Heating and Cooling loads will be determined by the area of the space and the building's estimated load in W/m². The occupancy and ventilation rates will be determined by defining the occupancy density and the ventilation criteria for the room in either ACH or l/s/person. The design parameters used to determine the engineering data can be populated in the Revit Spaces using the Space Parameter Data Population Dynamo Script.

	A	E	F	G	H	I
1	Space_Type	Space_Mechanical_ Cooling Load_Per	Space_Mechanical_ Heating Load_Per	Space_Mechanical_Ventilation _Air Change_Rate	Space_Mechanical_Fresh Air_Flow Rate_Per Person	Space_Occupancy _HVAC_Normal
2	Atrium	80	120	1	10	3
3	Cellular Office	80	120	1	10	10
4	Conference Centre	80	120	1	10	5
5	Corridor	80	0	1	10	1
6	Gym	80	120	6	10	12
7	Open Office	80	120	1	10	10
8	Plantroom	80	0	4	10	1
9	Reception	80	120	1	10	1
10	Restaurant	80	120	3	10	3
11	Storage	80	0	6	10	1
12	Toilets	80	0	10	10	1

Figure 24 Example of design Parameters used to determine engineering data

Most of the script uses the same principles as the script described in Workflow 1. We start by accessing the space information, re-order the data contained in the Lists, process the data and push it back in Revit.

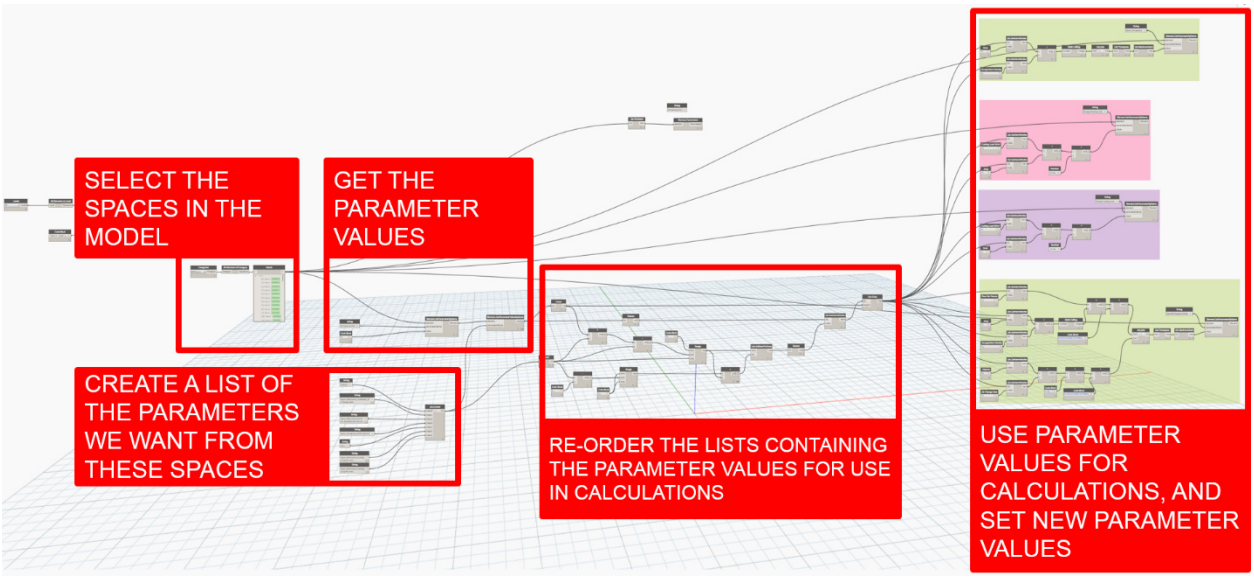


Figure 25 General description of the MEP Design Data Calculation tool

In this script we are accessing the space information slightly differently than in the previous script. Since we know exactly which parameters we want to access, we have used the Get.ParameterValueByName node and we have defined exactly which Parameters we want to read from directly.

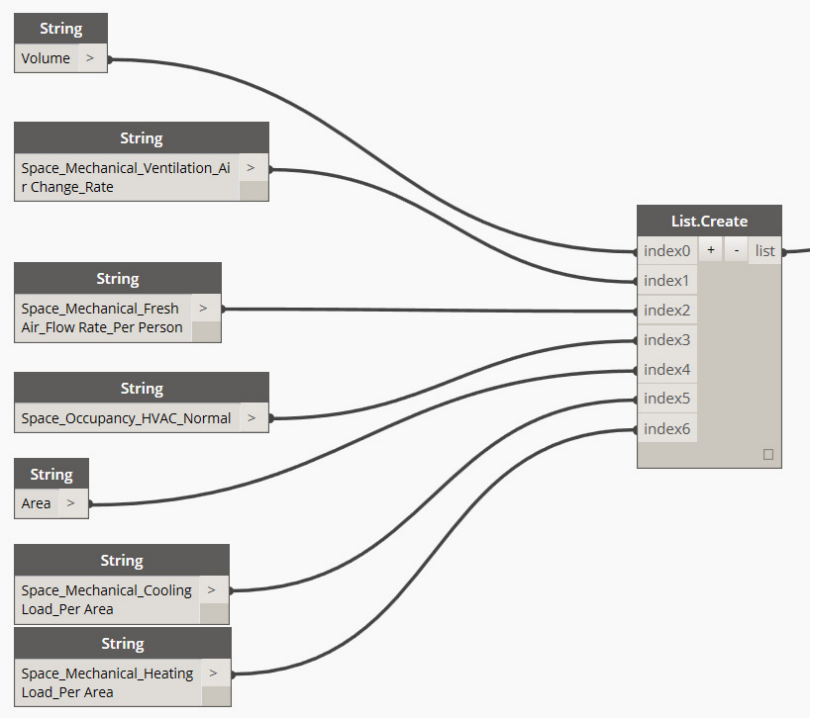


Figure 26 Read Space parameters from Revit

The relatively large number of nodes in the middle of the graph which are described as *RE-ORDER THE LISTS CONTAINING THE PARAMETER VALUES* are actually just transposing the List resulted from the *Get.ParameterByName* node. As discussed during the class, all those nodes can be replaces by *List.Transpose* which achieves the exact same result.

Now that we have accessed all the design information out of the Revit spaces we can look generating Engineering Data like Heating and Cooling Loads, Ventilation Rates and Occupancy values.

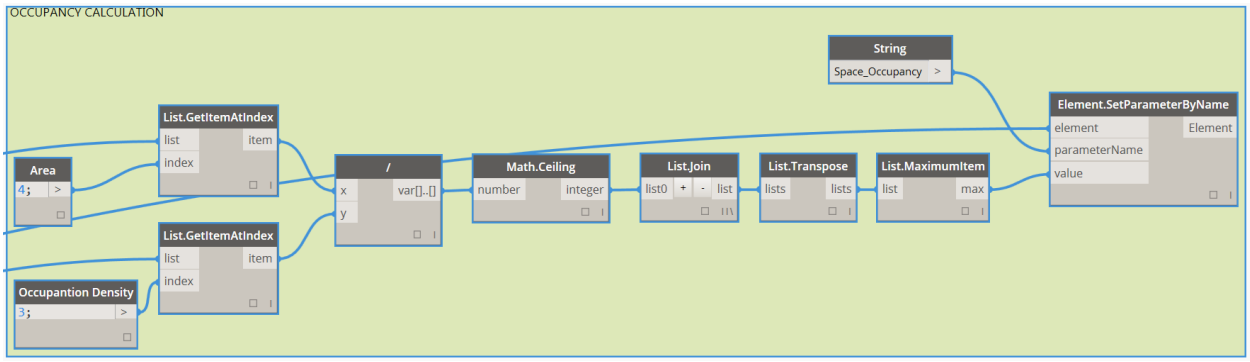


Figure 27 Occupancy calculations nodes

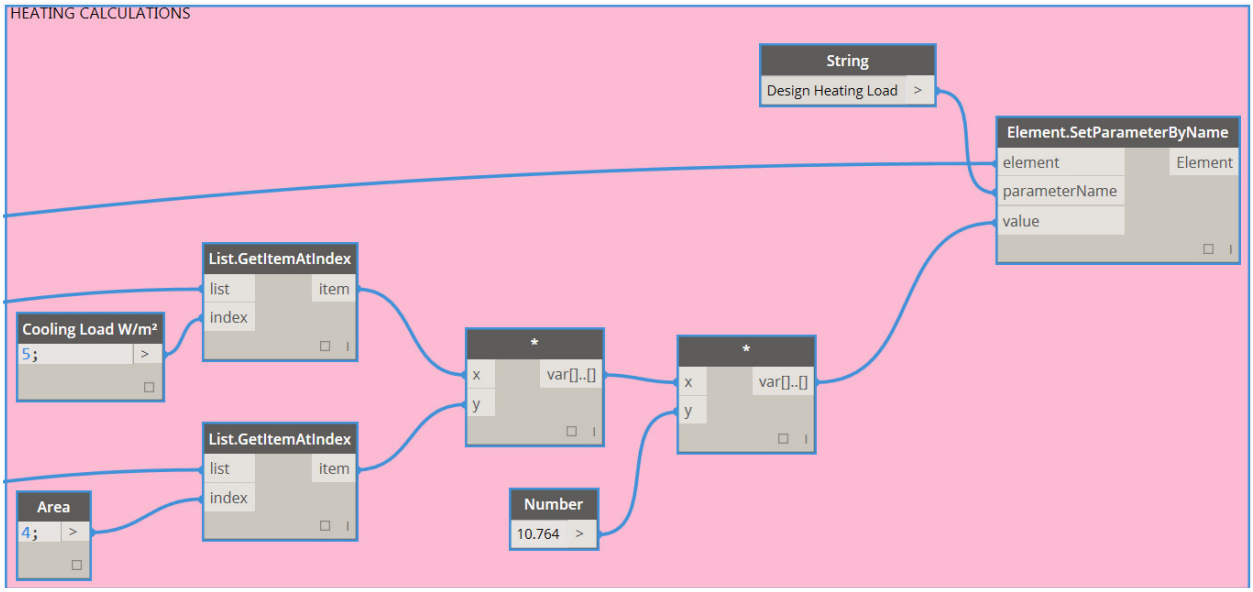


Figure 28 Heating load calculation nodes

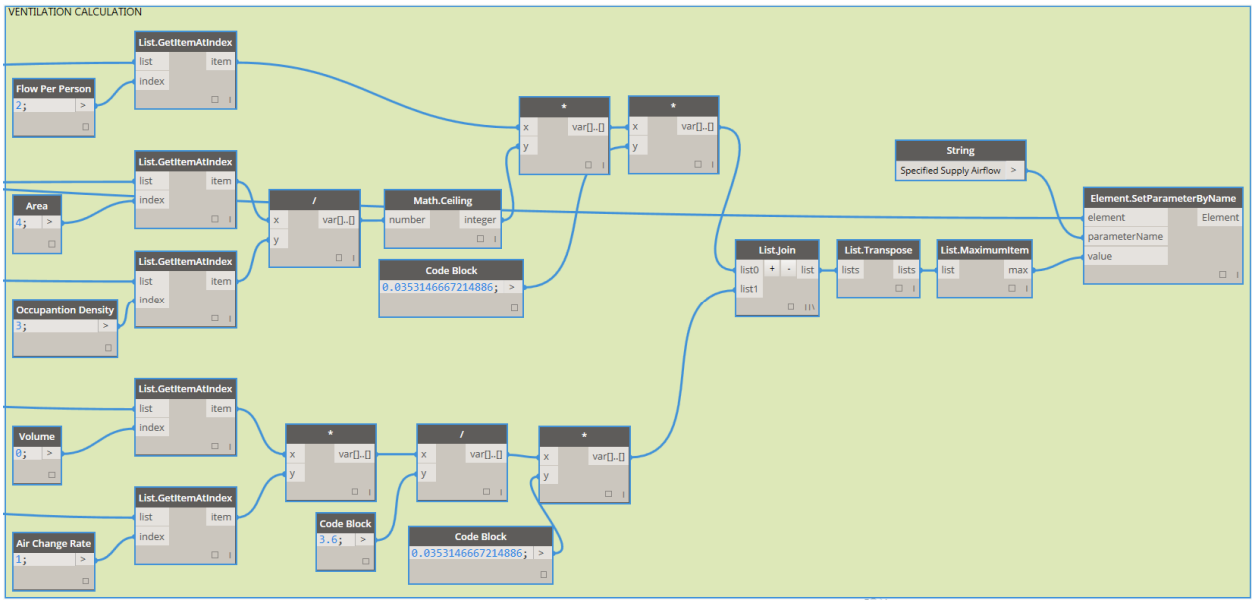


Figure 29 Ventilation calculation nodes

Most of the nodes used for the calculations have already been discussed in the first script. The only new introduction is the *List.Maximum* node which generates a list that stores the maxximum value of multiple lists at the same index number. In our case we are using this to determine the maximum between the l/s/person requirement and the air changes per hour of a space.

Workflow script 3 – MEP Equipment Data Calculations

Now that we have Engineering Data populated in the spaces we can use it to help us select the mechanical equipment. For this example we are only looking at the heating load of a space, however the script can be easily modified to perform the same task for the cooling load just by modifying the parameter name for reading the data from Space_Mechanical_Heating to Space_Mechanical_Cooling.

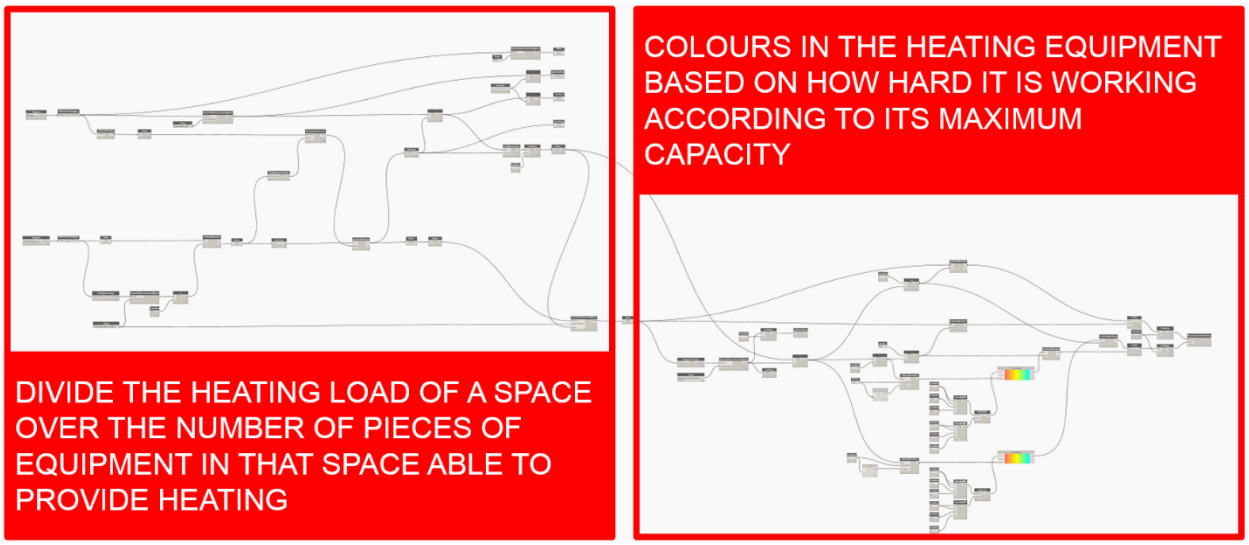


Figure 30 Equipment load calculation

For our example we want to populate the fan coil units in the space with the heating load required to cover space's demand. For this we need to identify the number of FCUs in the space and divide the space's total heat demand over the number of FCUs serving that space.

We already discussed in the previous workflows how to access space information, we now need to access the fan coil unit information. We achieve this by using the Categories node again as in the previous scripts, but we now set it to select all the Mechanical Equipment in the Revit model. Now that we have all the mechanical equipment instances selected we need to filter it down to the units that can hold a heating load value. To do this we filter off the elements in the list that don't have a Performance_Capacity_Heating parameter. The nodes to achieve this are shown in Figure 31.

The next step is to determine which fan coil units correspond to which space. This is being achieved by using the Geometry.DoesIntersect node which will give a true or false value depending of the space geometry intersects with the fan coil unit's geometry.

The last part of the graph is used to color the fan coil units based on how hard they are working. If the heating load set to the fan coil unit exceeds its maximum capacity, the unit will be colored red to flag up that the equipment needs to be reselected.

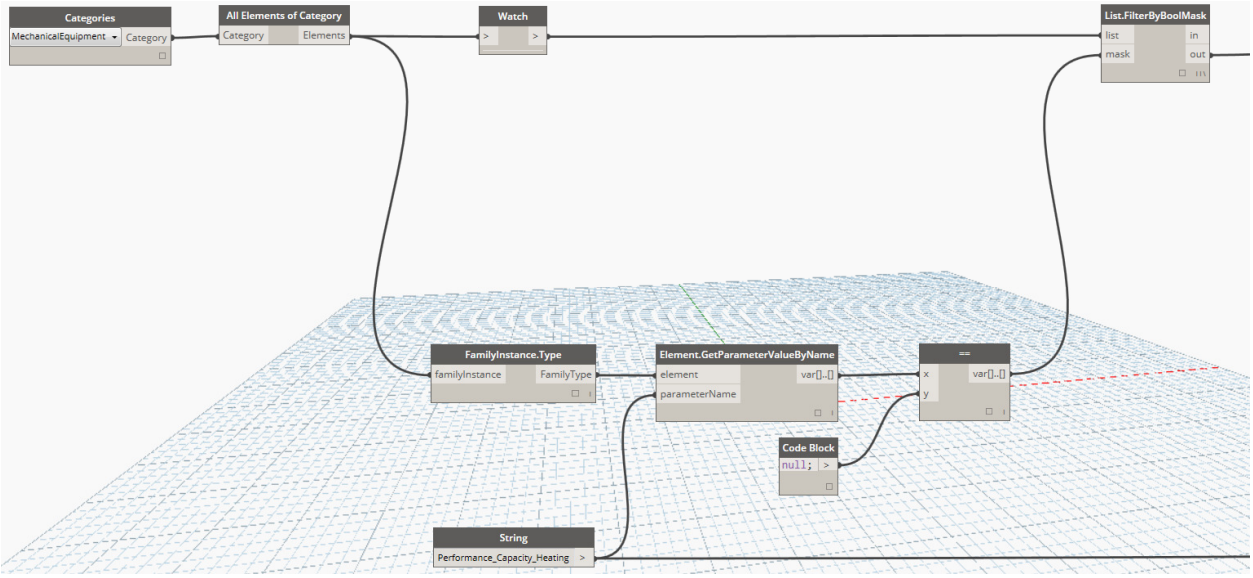


Figure 31 Select all fan coil units in the Revit model

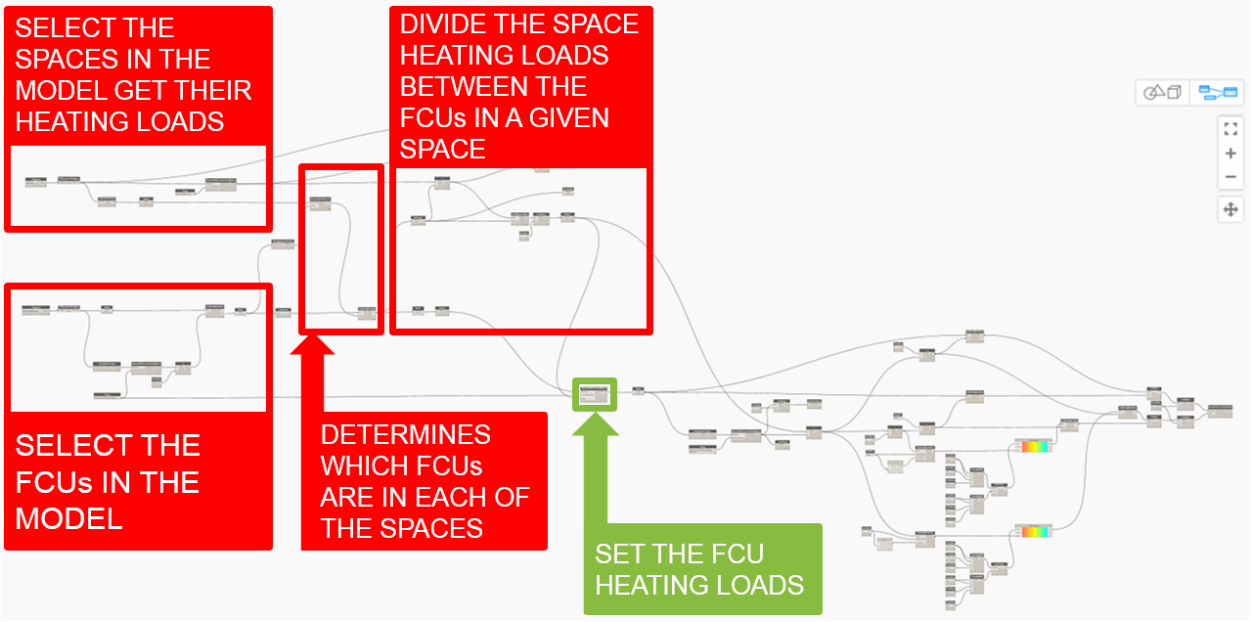


Figure 32 Setting fan coin unit loads workflow