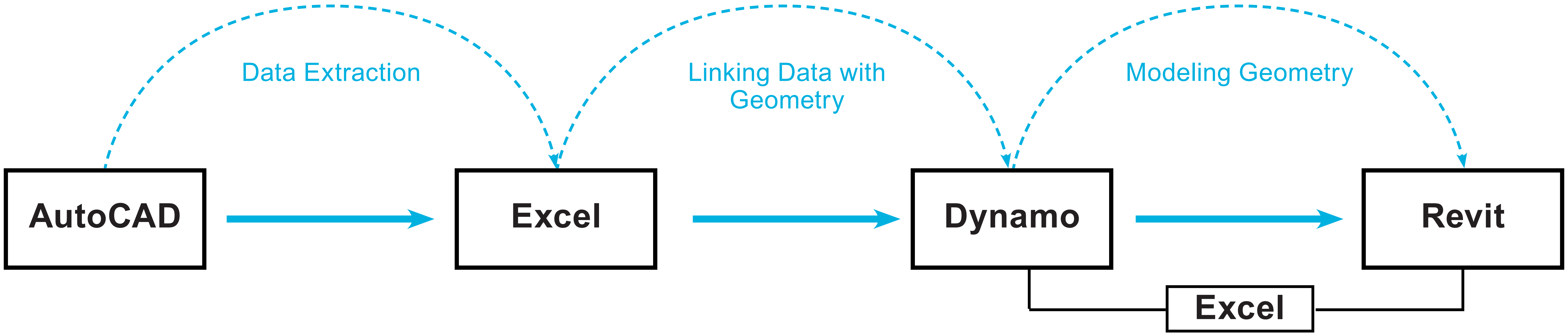
Automated Modeling of Existing CAD Files

Juren Dai
Andy Ilges

Workflow Breakdown

Study Object: Automated Modeling of Rooms (outlines composed of polylines)

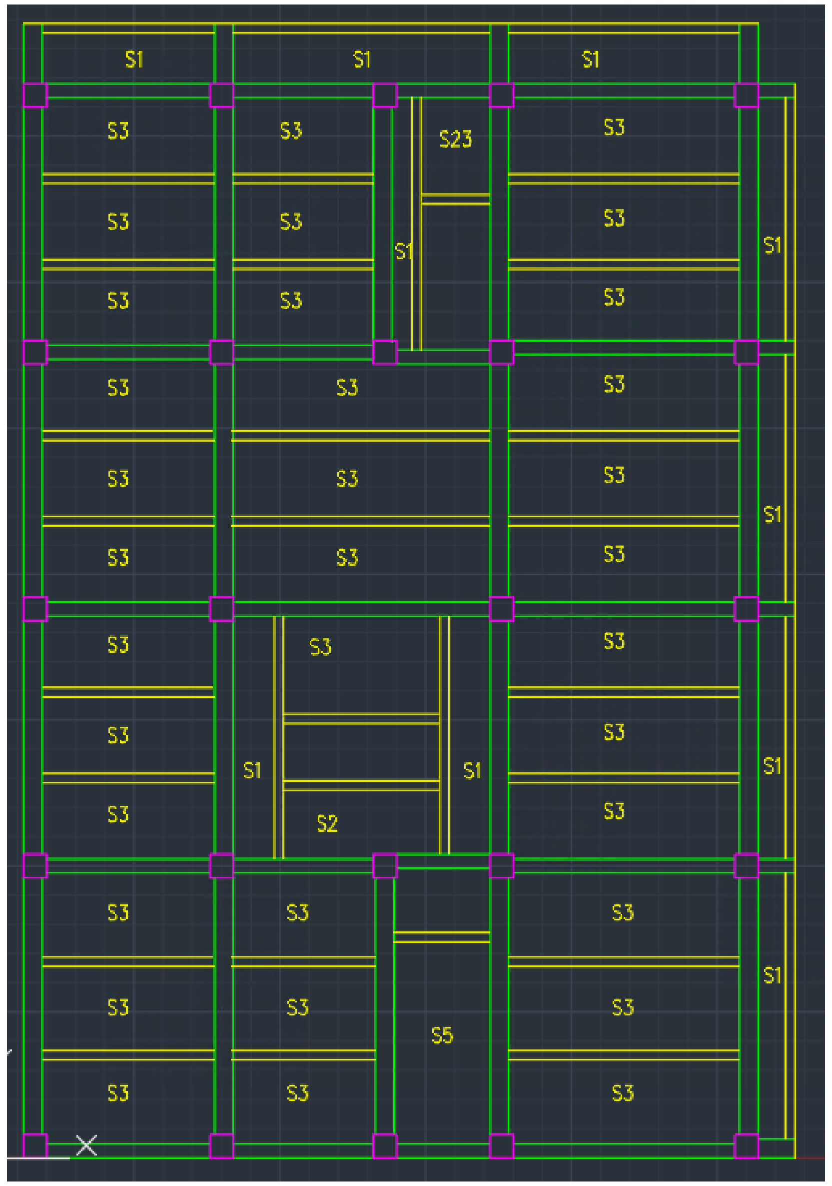


- Purge unecessary information in the CAD
- Add text for each type of geometry to be modeld in Revit and place in the center of geometry
- Export the xyz coordinate of these texts and the text value to Excel
- Add modeling information for each type in Excel

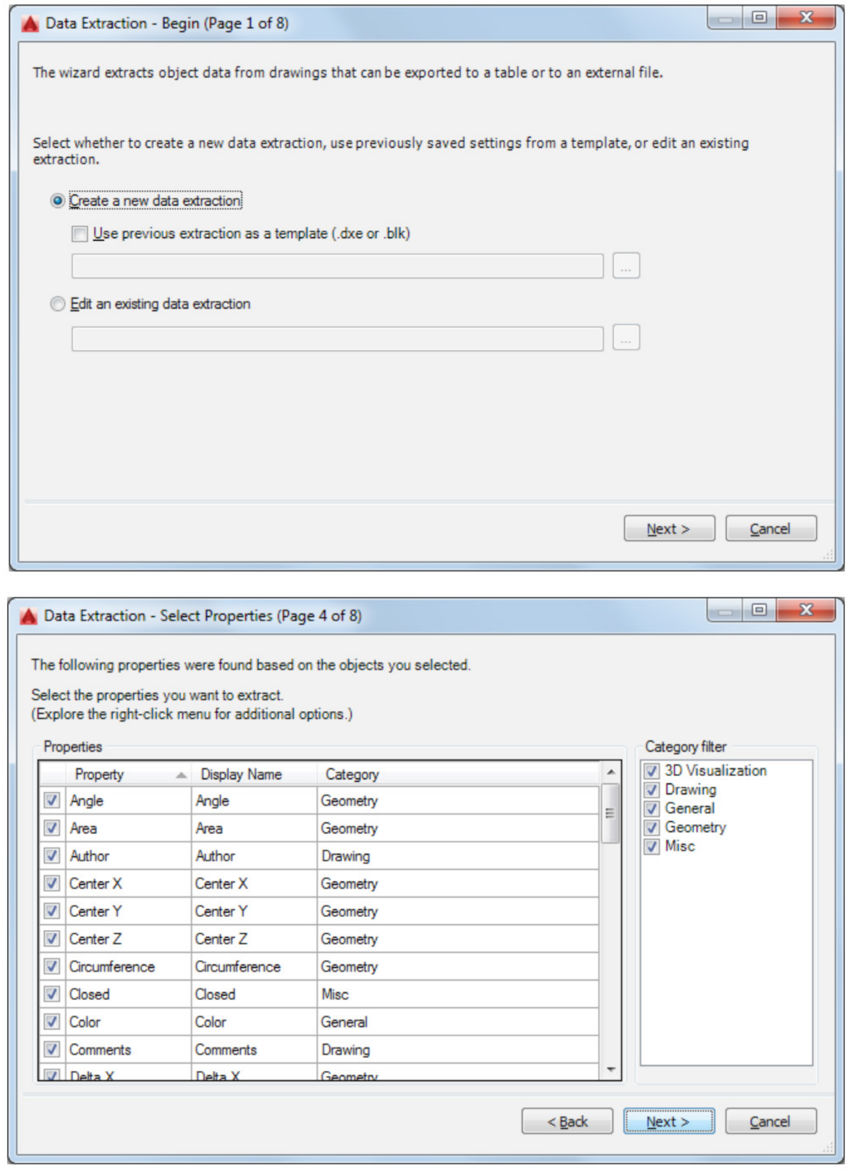
- Import the geometry and get the coordinate of control points of each geometry
- Get the modeling information of each geometry by finding the shortest distance between the centroid of geometry and the xyz coordinate of the text added in the CAD
- Generate the curves by connecting the control points in Revit
- Generate the polygons by these curves in Revit
- Model the elements in Revit from these polygons

#1 AutoCAD to Excel
Data Extraction

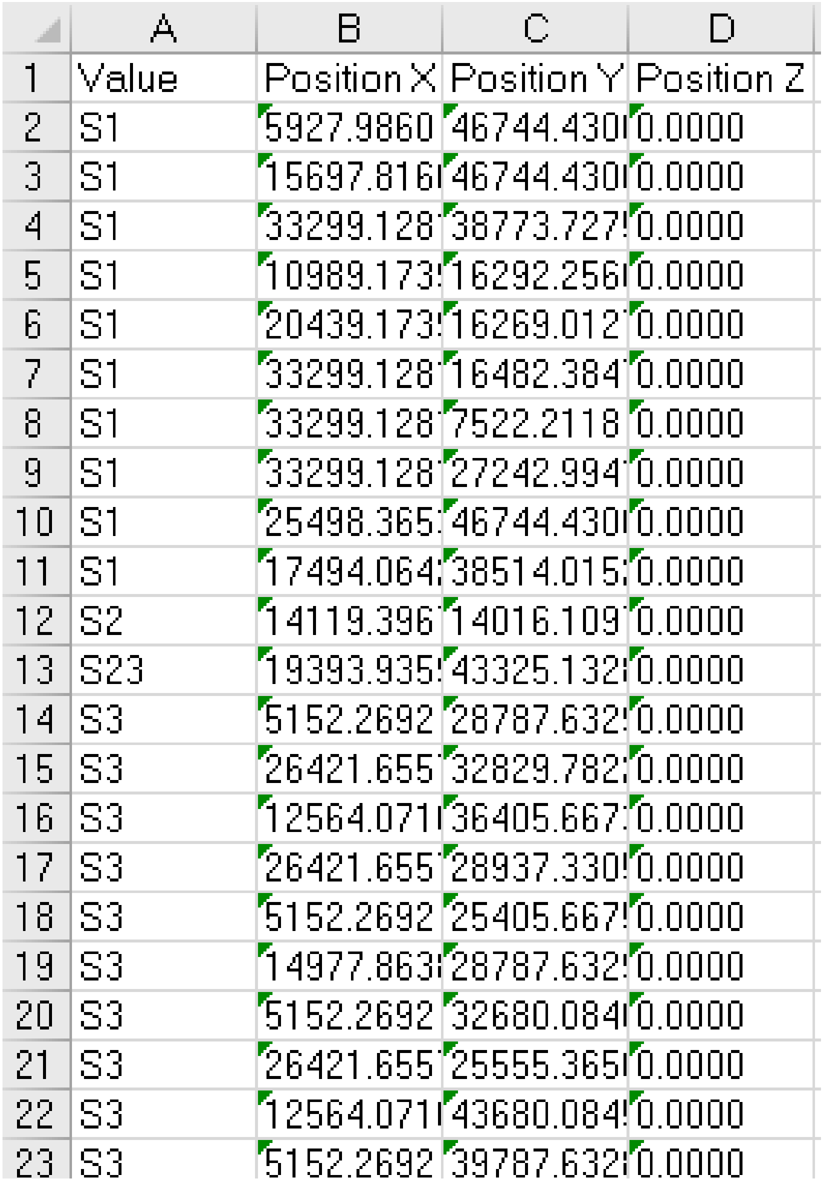
1. Add the slab number to each room that is going to be modeled in Revit. This number represents the type of the assigned room, such as the slab thickness, its level, or its workset, etc.



2. Use the “DataExtraction“ command in AutoCAD to export the xyz position of slab number and the text to an Excel spreadsheet.



3. Double check the generated Excel spreadsheet, and input the information related to each slab number, such as the level or thickness it represents.

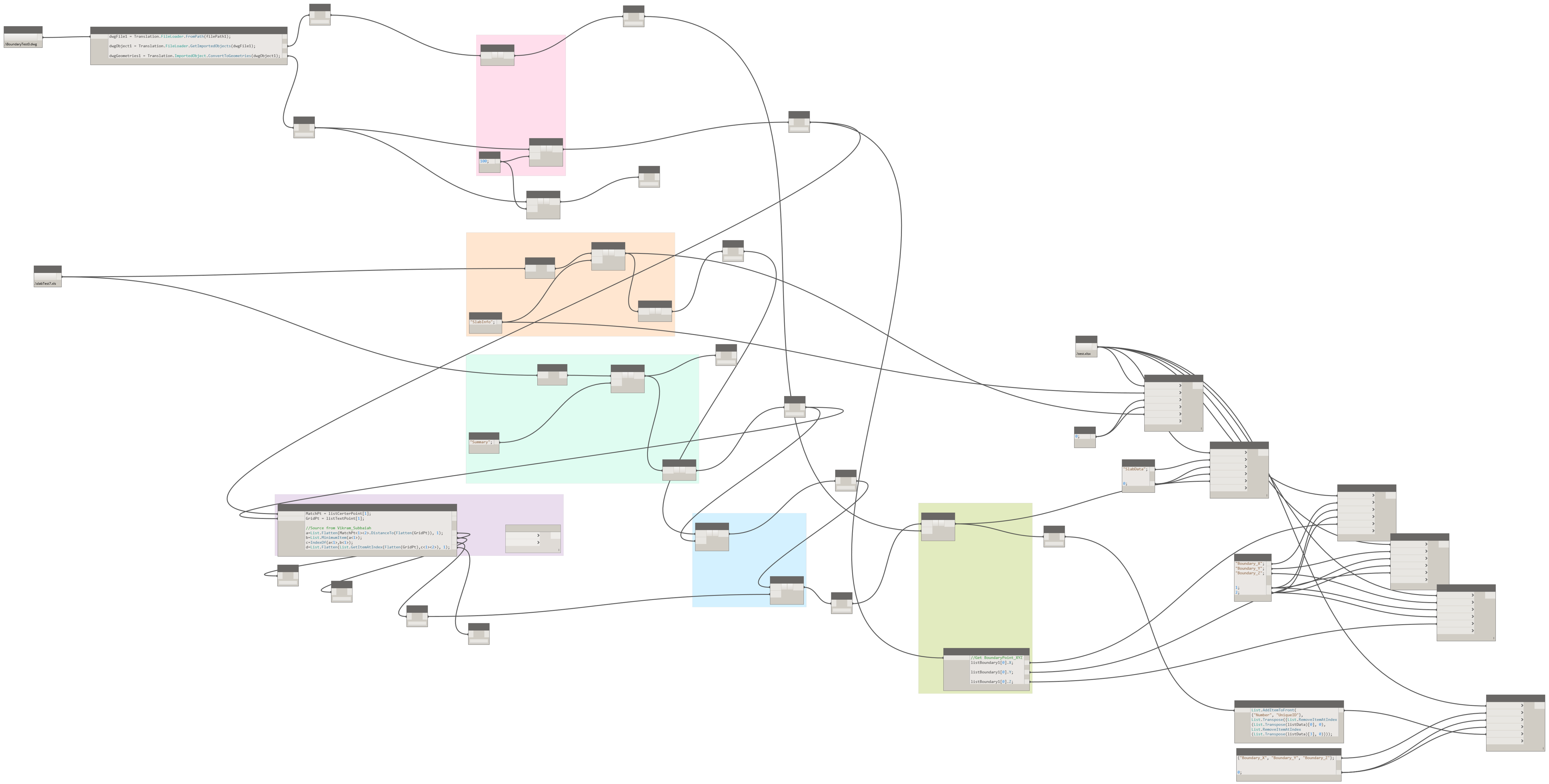


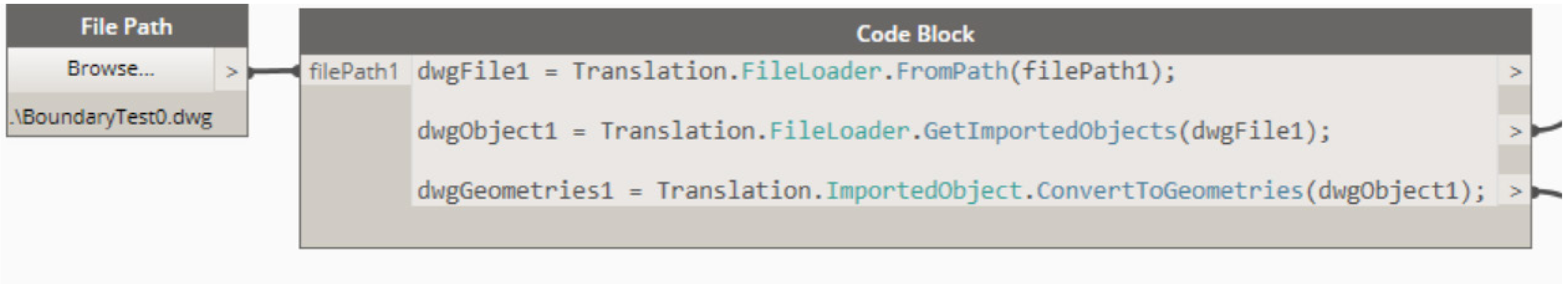
	A	B	C	D
1	Value	Position X	Position Y	Position Z
2	S1	5927.9860	46744.430	0.0000
3	S1	15697.816	46744.430	0.0000
4	S1	33299.128	38773.727	0.0000
5	S1	10989.173	16292.256	0.0000
6	S1	20439.173	16269.012	0.0000
7	S1	33299.128	16482.384	0.0000
8	S1	33299.128	7522.2118	0.0000
9	S1	33299.128	27242.994	0.0000
10	S1	25498.365	46744.430	0.0000
11	S1	17494.064	38514.015	0.0000
12	S2	14119.396	14016.109	0.0000
13	S23	19393.935	43325.132	0.0000
14	S3	5152.2692	28787.632	0.0000
15	S3	26421.655	32829.782	0.0000
16	S3	12564.071	36405.667	0.0000
17	S3	26421.655	28937.330	0.0000
18	S3	5152.2692	25405.667	0.0000
19	S3	14977.863	28787.632	0.0000
20	S3	5152.2692	32680.084	0.0000
21	S3	26421.655	25555.365	0.0000
22	S3	12564.071	43680.084	0.0000
23	S3	5152.2692	39787.632	0.0000

#2 Excel to Dynamo

Linking Data with Geometry

(run in Dynamo Studio Inteface only, as it supports the specific nodes in this script)






```python
#Assign the intput
importedObejcts1 = IN[0]

#Convert Objects to String
stringObjects1 = []
for items in importedObejcts1:
    stringObjects1.append(String.FromObject(items))

#Filter the Curve Objects
boolMask1 = []
for items in stringObjects1:
    boolMask1.append(String.StartsWith(items, "Curve", False))
filteredObject1 = []
for itemsString, itemsMask in zip(stringObjects1, boolMask1):
    if itemsMask == True:
        filteredObject1.append(itemsString)
uniqueID = []
for items in filteredObject1:
    uniqueID.append(String.Split(items, "ID = ", ", Sub Type"))
uniqueID1 = zip(*uniqueID)[1]


#Assign the output
OUT = uniqueID1
```

#2-01

Import the geometry from AutoCAD

Import the xyz coordinate of the slab number from Excel and its text

By using Python, we retrieve the unique ID of each geometry imported from AutoCAD This unique ID will be used to manage these geometry.

Number	Thickness	Offset	UniqueID
S1	150	0	10
S3	120	0	100
S3	120	0	101
S3	120	0	102
S3	120	0	103
S3	120	0	104
S3	120	0	105
S1	150	0	106
S1	150	0	107
S1	150	0	108
S1	150	0	109
S1	150	0	11
S1	150	0	110
S1	150	0	111
S3	120	0	112
S3	120	0	113
S2	90	-20	114
S2	90	-20	115
S23	100	-300	116

```python
#Assign the intput
convertedCurve1 = IN[0]
toleranceSimplify = IN[1]

#Simplify Converted curves
simplifiedCurve1 = []
for items in convertedCurve1:
    try:
        simplifiedCurve1.append(Curve.Simplify(items, toleranceSimplify))
    except:
        simplifiedCurve1.append(items)

#Convert Curve to NurbsCurve to get ControlPoints
nurbsCurve1 = []
for items in simplifiedCurve1:
    nurbsCurve1.append(Curve.ToNurbsCurve(items))
curvePoints1 = []
for items in nurbsCurve1:
    curvePoints1.append(NurbsCurve.ControlPoints(items))

#Create Polygon to get the Centroid
polygon1 = []
for items in curvePoints1:
    polygon1.append(Polygon.ByPoints(items))
center1 = []
for items in polygon1:
    center1.append(Polygon.Center(items))

#Assign the output
OUT = [curvePoints1, center1]
```

#2-02

Convert curves which were imported from the CAD drawing to NurbsCurve to get control points.

Create polygons from these imported curves, in order to get the centroid.

By measuring and finding the shortest distance between the centroid and the position of text, we match the geometry with its modeling information.

Code Block

Inputs: listCerterPoint, listTextPoint

```
MatchPt = listCerterPoint[1];
GridPt = listTextPoint[1];

//Source from Vikram_Subbaiah
a=List.Flatten(MatchPt<1><2>.DistanceTo(Flatten(GridPt)), 1);
b=List.MinimumItem(a<1>);
c=IndexOf(a<1>,b<1>);
d=List.Flatten(List.GetItemAtIndex(Flatten(GridPt),c<1><2>), 1);
```

	A	B	C	D	E	F	G
1	Number	UniqueID					
2	S1	10	22511.79	25678.46	28845.13	32011.79	
3	S3	100	2311.793	4778.46	7245.126	9711.793	
4	S3	101	10511.79	12511.79	14511.79	16511.79	
5	S3	102	22311.79	25611.79	28911.79	32211.79	
6	S3	103	2311.793	4778.46	7245.126	9711.793	
7	S3	104	10511.79	12511.79	14511.79	16511.79	
8	S3	105	22311.79	25611.79	28911.79	32211.79	
9	S1	106	18161.79	18161.79	18161.79	18161.79	
10	S1	107	18561.79	18561.79	18561.79	18561.79	
11	S1	108	12661.79	12661.79	12661.79	12661.79	
12	S1	109	12261.79	12261.79	12261.79	12261.79	
13	S1	11	33011.79	33536.79	34061.79	34586.79	
14	S1	110	19361.79	19361.79	19361.79	19361.79	
15	S1	111	19761.79	19761.79	19761.79	19761.79	
16	S3	112	12661.79	14895.13	17128.46	19361.79	
17	S3	113	12661.79	14895.13	17128.46	19361.79	
18	S2	114	12661.79	14895.13	17128.46	19361.79	
19	S2	115	12661.79	14895.13	17128.46	19361.79	
20	S23	116	21511.79	20528.46	19545.13	18561.79	
21	S23	117	21511.79	20528.46	19545.13	18561.79	
22	S5	118	17411.79	18778.46	20145.13	21511.79	
23	S5	119	17411.79	18778.46	20145.13	21511.79	
24	S3	12	2511.793	4845.126	7178.46	9511.793	
25	S1	120	34586.79	34586.79	34586.79	34586.79	
26	S1	121	34186.79	34186.79	34186.79	34186.79	
27	S1	122	34186.79	34186.79	34186.79	34186.79	
28	S1	123	34186.79	34186.79	34186.79	34186.79	
29	S1	124	34186.79	34186.79	34186.79	34186.79	
30	S3	125	2311.793	4778.46	7245.126	9711.793	
31	S3	126	10411.79	14111.79	17811.79	21511.79	
32	S3	127	22311.79	25611.79	28911.79	32211.79	
33	S3	128	2311.793	4778.46	7245.126	9711.793	
34	S3	129	10411.79	14111.79	17811.79	21511.79	
35	S3	13	10511.79	12511.79	14511.79	16511.79	
36	S3	130	22311.79	25611.79	28911.79	32211.79	
37	S3	131	2311.793	4778.46	7245.126	9711.793	
38	S3	132	10411.79	14111.79	17811.79	21511.79	
39	S3	133	22311.79	25611.79	28911.79	32211.79	
40	S3	134	2311.793	4745.126	7178.46	9611.793	
41	S3	135	10411.79	14111.79	17811.79	21511.79	
42	S3	136	22311.79	25611.79	28911.79	32211.79	
43	S3	137	2311.793	4778.46	7245.126	9711.793	
44	S3	138	2311.793	4778.46	7245.126	9711.793	
45	S3	139	2311.793	4778.46	7245.126	9711.793	

SlabData | **Boundary_X** | Boundary_Y | Boundary_Z

#2-03

Export the xyz coordinate of control points of each geometry, as well as its corresponding uniqueID. Also, the slab information is remained for next step.

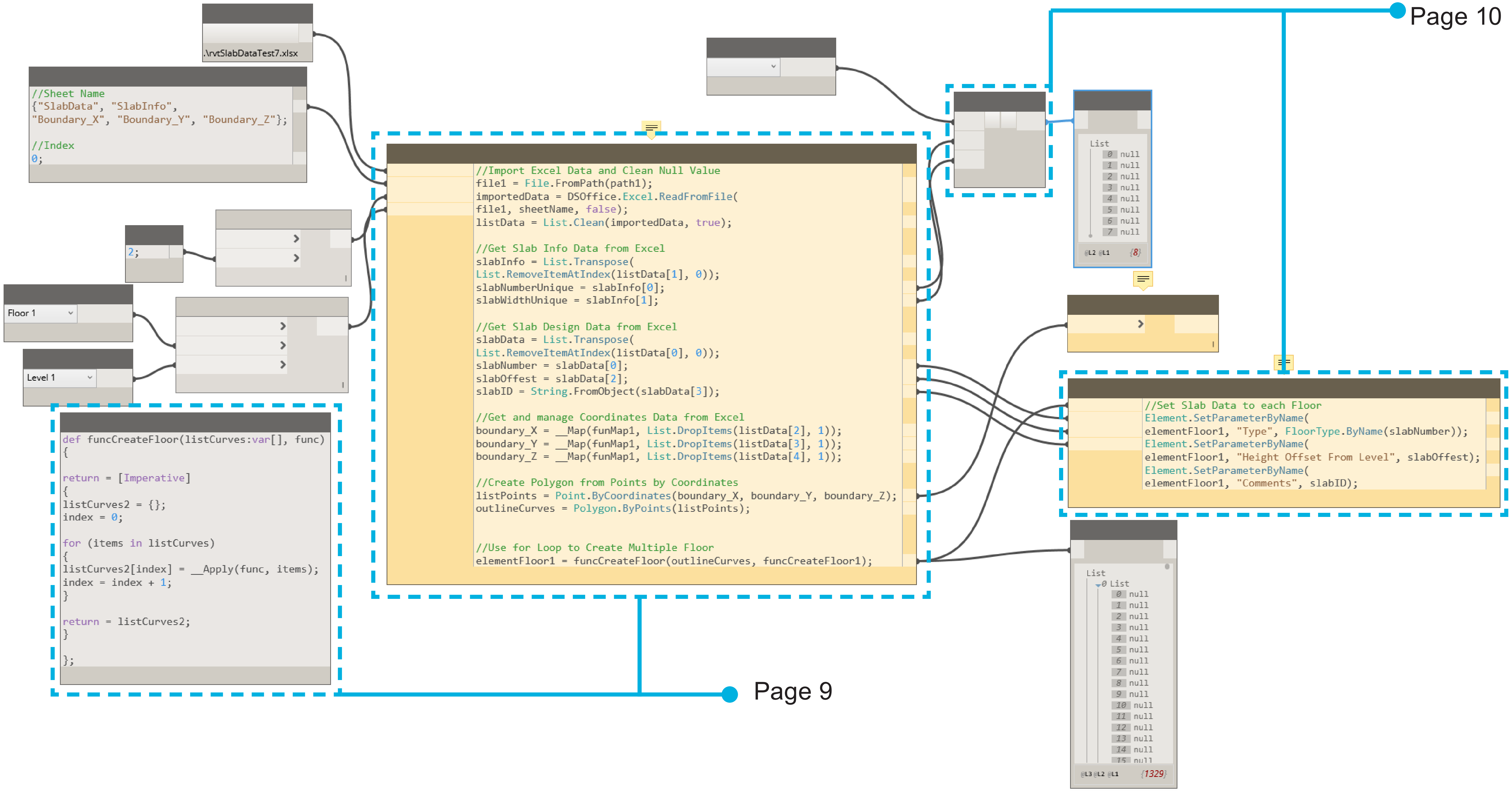
Left: X-axis Coordiante

Below: Slab Information

Number	Thickness	Offset	UniqueID
S1	150	0	10
S3	120	0	100
S3	120	0	101
S3	120	0	102
S3	120	0	103
S3	120	0	104
S3	120	0	105
S1	150	0	106
S1	150	0	107
S1	150	0	108
S1	150	0	109

#3 Dynamo to Revit
Modeling Geometry

(run in Revit Application Inteface)



#3-01

Code Block

Inputs: path1, sheetName, funMap1, funcCreateFloor1

```
//Import Excel Data and Clean Null Value
file1 = File.FromPath(path1);
importedData = DSOffice.Excel.ReadFromFile(
file1, sheetName, false);
listData = List.Clean(importedData, true);

//Get Slab Info Data from Excel
slabInfo = List.Transpose(
List.RemoveItemAtIndex(listData[1], 0));
slabNumberUnique = slabInfo[0];
slabWidthUnique = slabInfo[1];

//Get Slab Design Data from Excel
slabData = List.Transpose(
List.RemoveItemAtIndex(listData[0], 0));
slabNumber = slabData[0];
slabOffest = slabData[2];
slabID = String.FromObject(slabData[3]);

//Get and manage Coordinates Data from Excel
boundary_X = __Map(funMap1, List.DropItems(listData[2], 1));
boundary_Y = __Map(funMap1, List.DropItems(listData[3], 1));
boundary_Z = __Map(funMap1, List.DropItems(listData[4], 1));

//Create Polygon from Points by Coordinates
listPoints = Point.ByCoordinates(boundary_X, boundary_Y, boundary_Z);
outlineCurves = Polygon.ByPoints(listPoints);


//Use for Loop to Create Multiple Floor
elementFloor1 = funcCreateFloor(outlineCurves, funcCreateFloor1);
```

Import the xyz coordinate of control points of each nurbs curve from Excel, and organize the list.

Generate polygons based on these control points.

Generate Revit floors from these polygons.

funcCreateFloor

```
def funcCreateFloor(listCurves:var[], func)
{

return = [Imperative]
{
listCurves2 = {};
index = 0;

for (items in listCurves)
{
listCurves2[index] = __Apply(func, items);
index = index + 1;
}

return = listCurves2;
}

};
```

```python
def tolist(obj1):
    if hasattr(obj1,"__iter__"): return obj1
    else: return [obj1]

def output1(l1):
    if len(l1) == 1: return l1[0]
    else: return l1

fs = UnwrapElement(IN[0])
names = tolist(IN[1])
width = tolist(IN[2])
#Convert width to Metric Unit
widthMetric = []
for items in width:
    widthMetric.append(items / 304.8)

nfs = []

TransactionManager.Instance.EnsureInTransaction(doc)
for i in xrange(len(names)):
    try:
        try:
            x = fs.Duplicate(str(names[i]))
            cs = x.GetCompoundStructure()
            ind = cs.StructuralMaterialIndex
            cs.SetLayerWidth(ind,widthMetric[i])
            x.SetCompoundStructure(cs)
            nfs.append(x.ToDSType(False))
        except:
            nfs.append(Revit.Elements.FloorType.ByName(str(names[i])))
    except:
        nfs.append(None)
TransactionManager.Instance.TransactionTaskDone()
OUT = output1(nfs)
```

#3-02

Set slab data to each floor generated before.

Return all the slab type that’s been applied to the model. This is to check if any elements get missing during the process.

Code Block

Inputs: elementFloor1, slabNumber, slabOffest, slabID

```
//Set Slab Data to each Floor
Element.SetParameterByName(
elementFloor1, "Type", FloorType.ByName(slabNumber));
Element.SetParameterByName(
elementFloor1, "Height Offset From Level", slabOffest);
Element.SetParameterByName(
elementFloor1, "Comments", slabID);
```

Conclusion:

Extracting data out of AutoCAD through Excel, recreating polygons based on imported curves, and then modeling from these polygons is an efficient and easy-to-understand method to convert AutoCAD elements to Revit. Due to the different element-specific conditions, eg. modeling columns and rooms from imported curves are two differnt mechanisms, each scenario shall be categorized and studied separately, but they can all be solved by this same workflow and the thought.

Automated modeling is a prospective topic, and we'll continue our research in this field. It takes efforts to develop the automation workflow that works for complex practical project environment, but once it matures, we'll greatly benefit from its outcomes.

Reference:

Autodesk University

dynamobim.org/forum

Special thanks to:

Viakram Subbalah

Jia Lin Chou