AUTODESK UNIVERSITY 2013

Incorporating Engineering into Autodesk® Revit® Structure: Advanced Families

Desirée Mackey, PE – Martin/Martin

SE1591 This class presents several custom families that incorporate structural engineering within the families themselves. By integrating engineering into custom families, a structural engineer can capitalize on more features of Revit Structure to aid in the engineering design process. The families demonstrate several tools within the Family Editor, including nesting, formulas, conditional statements, reporting parameters, and visibility settings. In a companion class, "SE1592: Incorporating Engineering into Autodesk® Revit® Structure: Project Procedures," these families and all of their features are demonstrated within the project environment.

Learning Objectives

At the end of this class, you will be able to:

- Incorporate engineering into Revit Structure
- Use formulas and conditional statements in structural families
- Use advanced family creation techniques to enhance the structural engineering capabilities of 2D and 3D families
- Use reporting parameters and nesting techniques in structural families

About the Speaker

Desirée (Dezi) received her bachelor's degree from University of California, Davis and her master's degree from Massachusetts Institute of Technology and now is a practicing structural engineer at Martin/Martin in Denver, Colorado. In the past several years she has been a regular Autodesk University speaker and has spoken at Revit Technology Conference USA. In addition, she sits on the committee for RoMBIS (Rocky Mountain Building Information Society – a Denver area users group), the Chair of the Structural Engineers Association of Colorado's BIM committee, and she is currently serving as an AUGI board member and Treasurer. Finally, she also acts as a partner in her husband's BIM consulting company, BD Mackey Consulting.

Email: dmackey@martinmartin.com
Blog: bdmackeyconsulting.com/blog
Twitter: @RevitGeeksWife

Introduction

By integrating engineering into custom families and project procedures, a structural engineer can make use of more features of Revit to aid in the engineering design process. This session is a compilation of families that incorporate structural engineering and design. This session will present a wide range of possible opportunities for the incorporation of engineering, separated into three categories: families that perform calculations, families that design themselves and families for project workflows. This class will discuss specifically how to create these families. In the companion class, "SE1592 Incorporating Engineering into Autodesk® Revit® Structure: Project Procedures," these and other custom families are demonstrated within a project environment.

Families that Perform Calculations

A useful application of adding engineering into families is to have families perform calculations. The calculations could be for design or engineering purposes, or simply code checks or other verification-type calculations that are meant to report back information to the user. The following families are some examples of this application.

Reinforced Beam Detail Item

This family was originally conceived based on the idea that drafting 2D concrete details was time consuming. Originally, the item was just intended to combine/nest several other detail items into one. However, once all incorporated, it seemed a logical next step to add some engineering, after all, most of the design information was already present.

This family was created starting from the detail item template.

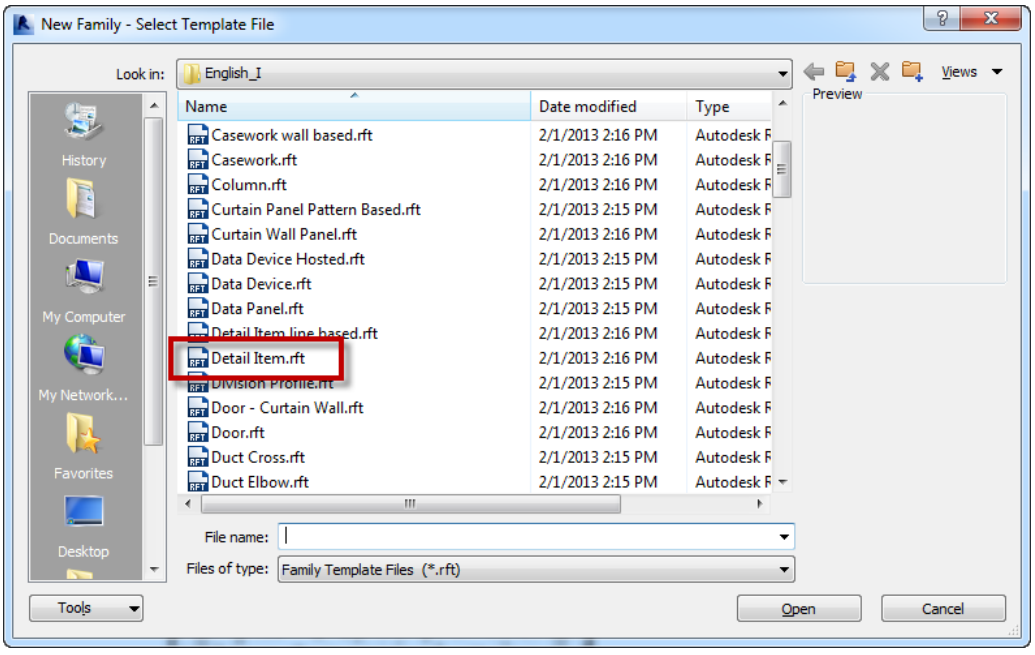

Figure 1: Family template

This family is mostly a compilation of nested details, so the only geometry to be added into the family directly is a filled region to represent the concrete beam. Reference planes for all the geometry were added and constrained first, then the filled region was placed and locked to the proper reference lines.

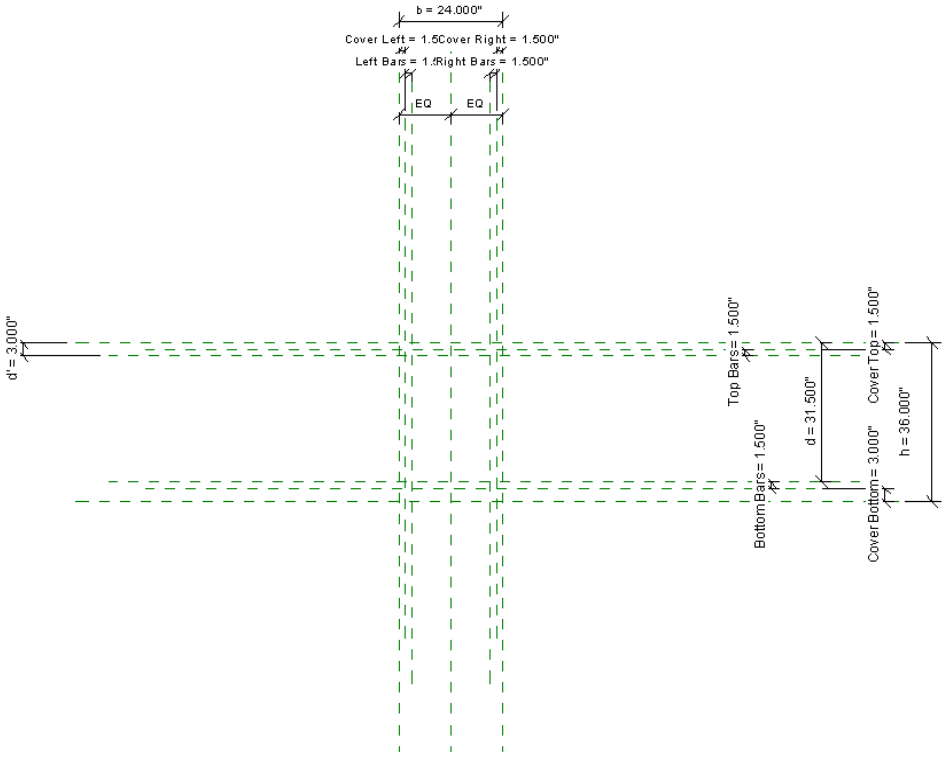

Figure 2: Reference planes

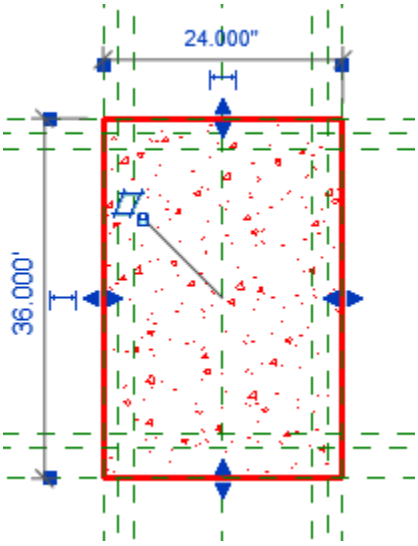

Figure 3: Filled region

Next all the items to be nested were added and also constrained to their proper reference lines. The top and bottom bars are instance-based rebar sections and the tie is an instance-based custom family. While it is possible to control types with the "<Family Type> parameter, in this case it was more appropriate to use instance parameters. With nested families, the most important step is to map the parameters of the nested elements to parameters within the host family. This was carried out for all of the nested items. The top and bottom bars were also arrayed, and the array was constrained to parameters as well.

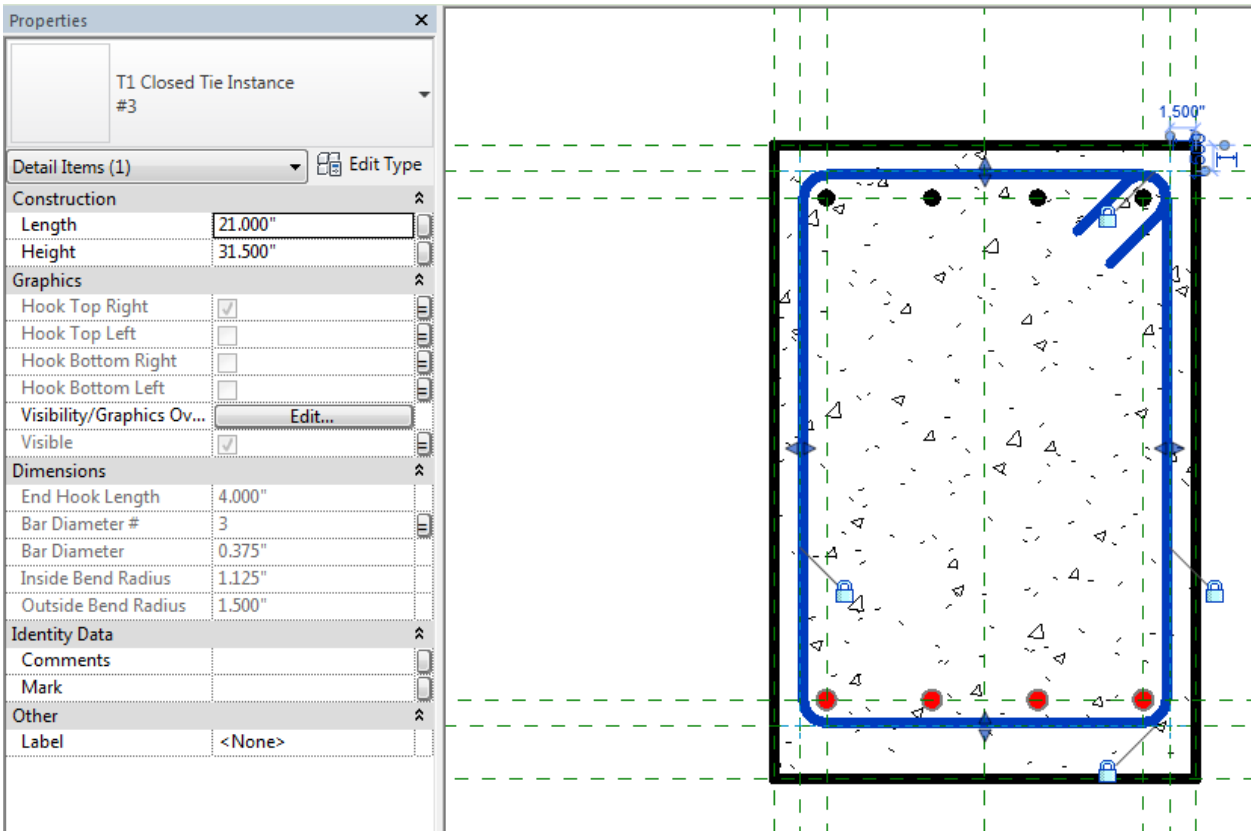

Figure 4: Map parameters

At this point the family is completely functional as a detail item with parametric geometry. However, since all the information is already available, adding some calculations could quickly make this family more useful. In this case, calculations were added to calculate the flexural capacity of the beam, given the configuration of reinforcement that the user chooses. In addition, some calculations for a couple code checks (minimum steel and minimum clear spacing of reinforcement) were also added. The image below shows some of the formulas.

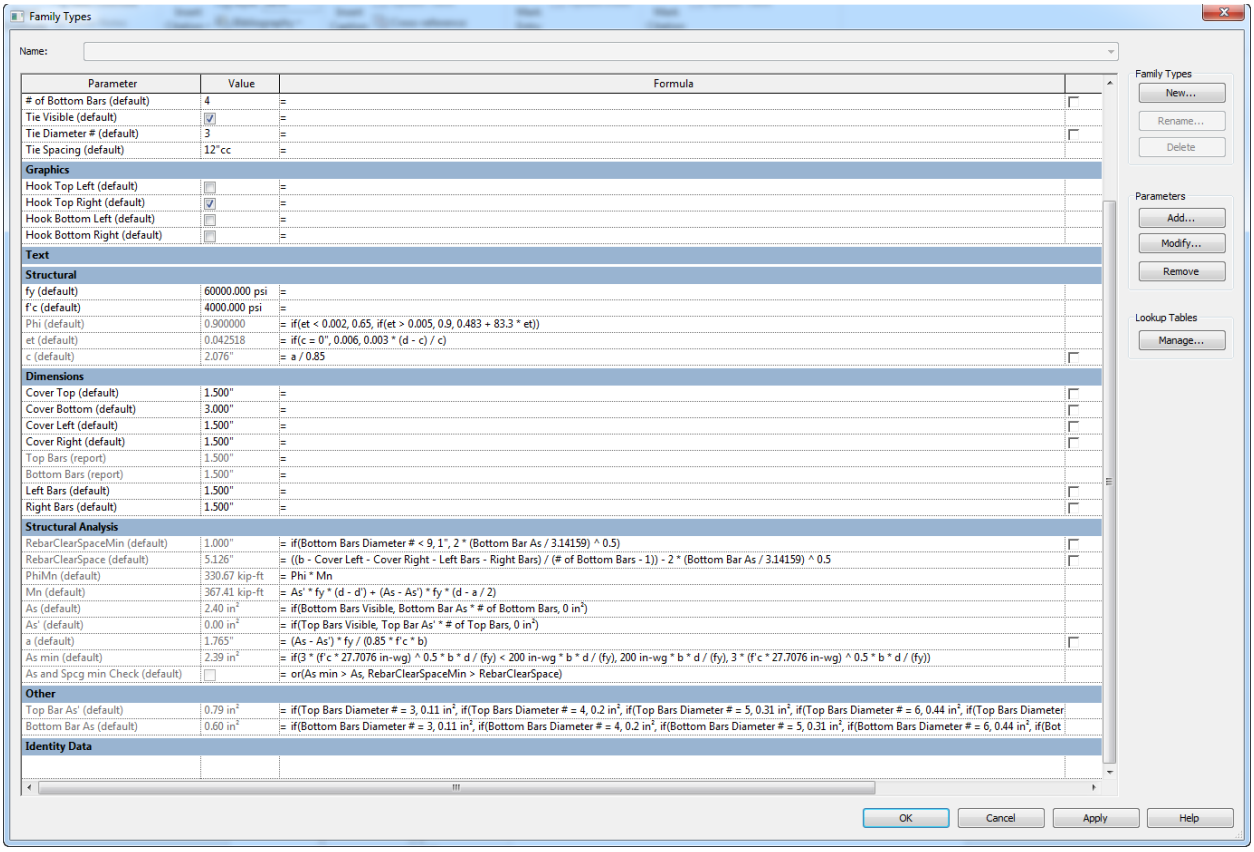

Figure 5: Calculations

The next step was to link the calculations that perform code checks to some way to alert the user if the minimums were violated. This was simply accomplished by adding some red filled regions that match the size and location of the bottom bars. A more simple route could have been to add something like a big square, or text, but this way the detail could still be printed (in black and white) without problems. The red filled regions can be seen in Figure 4.

The final step was to make this family such that it can have a more descriptive tag. Afterall, why build in so much intelligence and not allow it to come through in the project? A series of custom text parameters were created such that they would populate themselves to report the geometry, as chosed by the user.

The final image below is how the component looks inside the project.

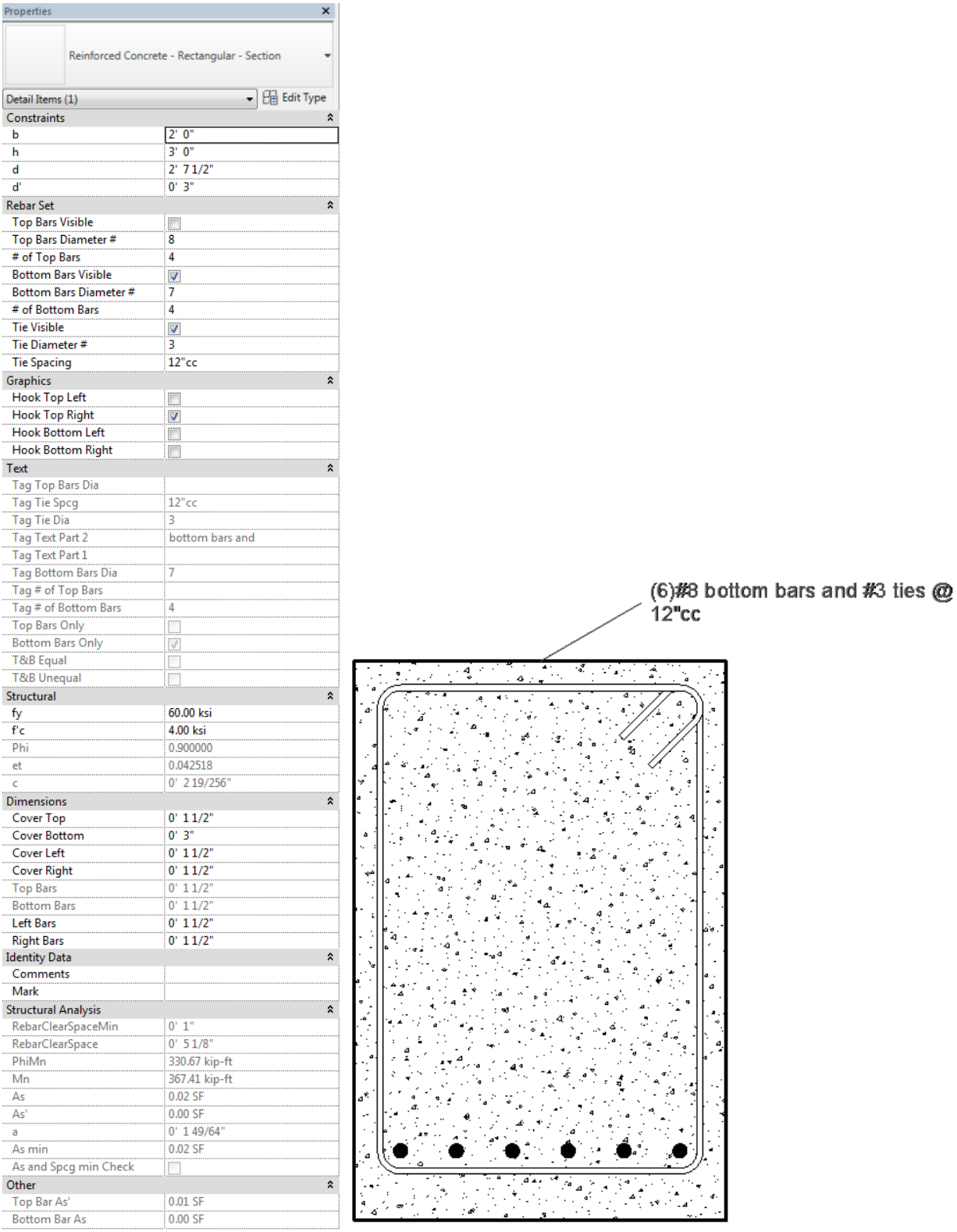

Figure 6: Concrete detail component in project

This idea of adding calculations, code checks, etc., can be expanded to a plethora of other components – a concrete column, for example. This is just one example of how to incorporate some engineering and design information, as well as some basic code checks, into an already useful and robust detail component.

Shear Wall Family

One of the author's favorite families that performs calculations is a component that performs wood shear wall calculations. Over the years, as new features became available within Revit, this component has evolved and has been rebuilt numerous times. The latest version of this family is an adaptive component.

Since this family has been several years in the making, the calculations within it have developed and evolved, and are quite extensive. The calculations use several types of parameters, numerous formulas, logical tests, and even lookup tables, which are newly incorporated into Revit 2014.

The component began with the "Generic Model Adaptive" family template. The first step was to place four points, make them adaptive placement points, and set their orientation to "Orthagonal on Placement".

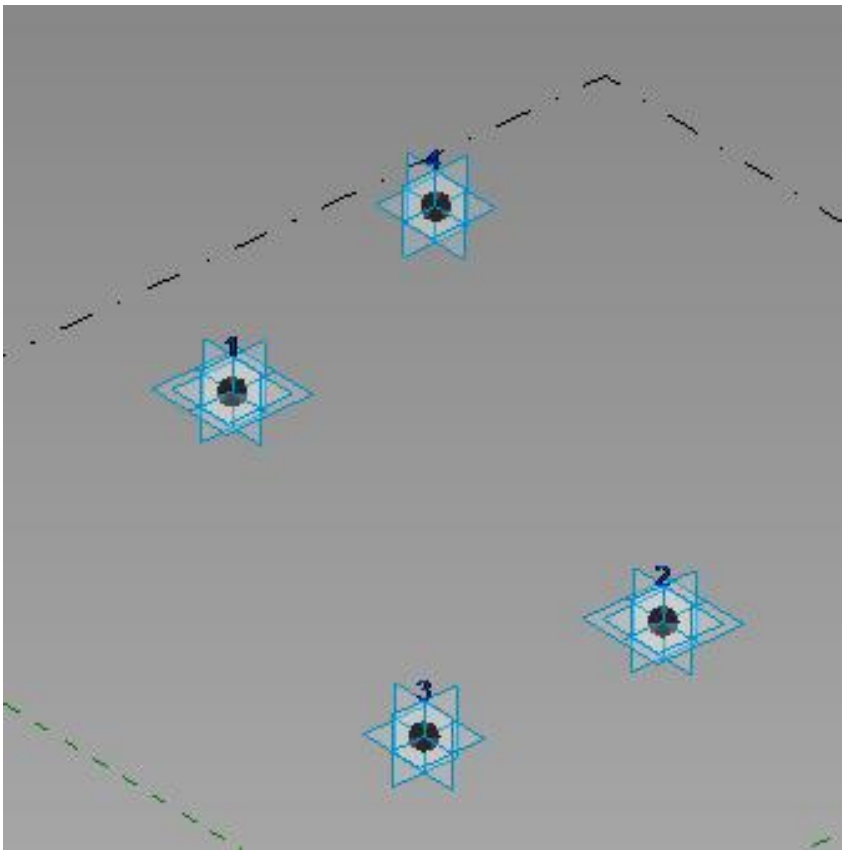

Figure 7: Four placement points

Dimensions constrined to the horizontal and vertical planes were added and labeled as reporting paramters – once in the project these parameters will repor the wall length and height.

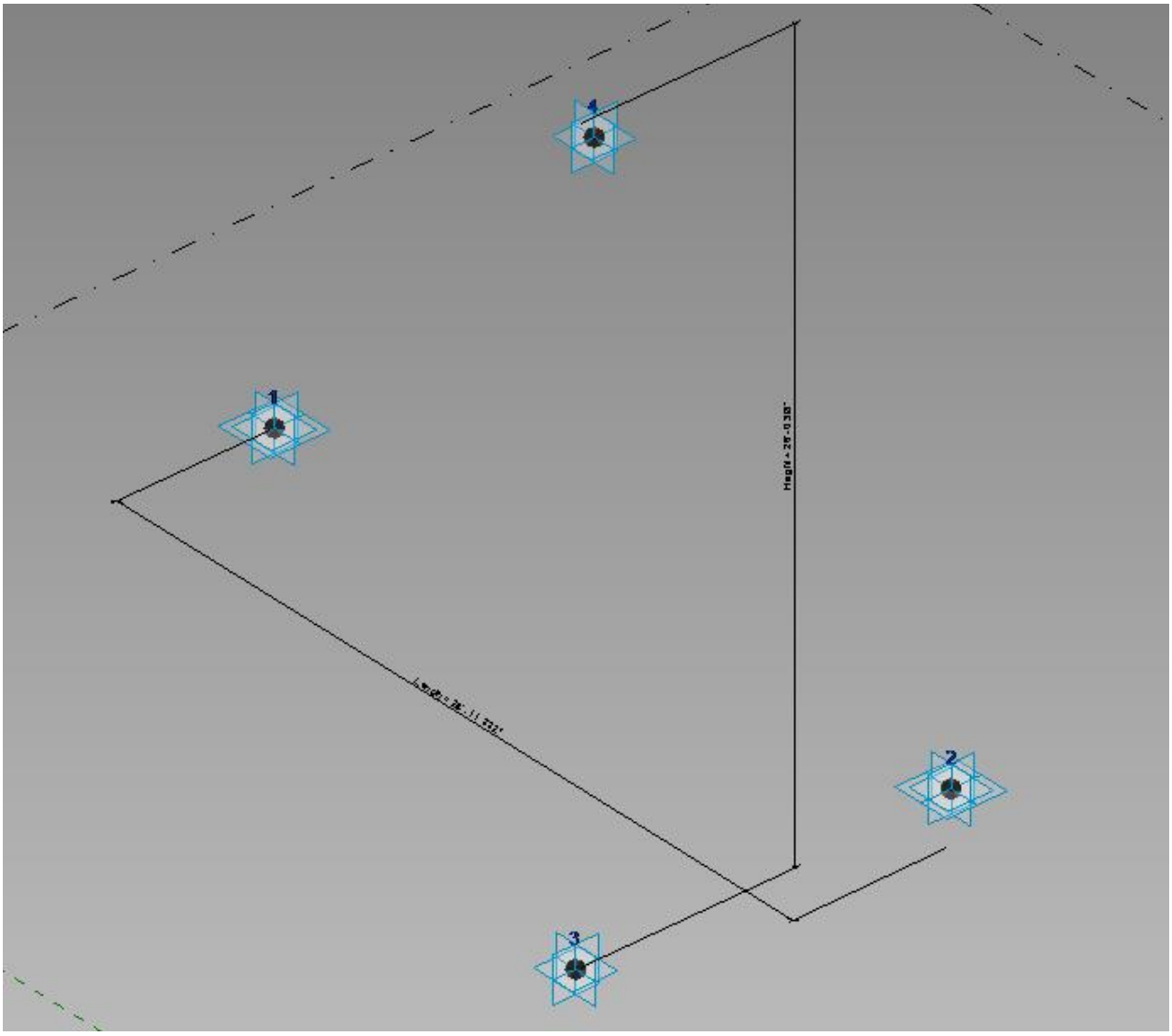

Figure 8: Dimensions

Next points were added by offsetting the left- and right-most points, and those points were connected with a reference line. The goal here is to create lines that represent the start and end of the wall.

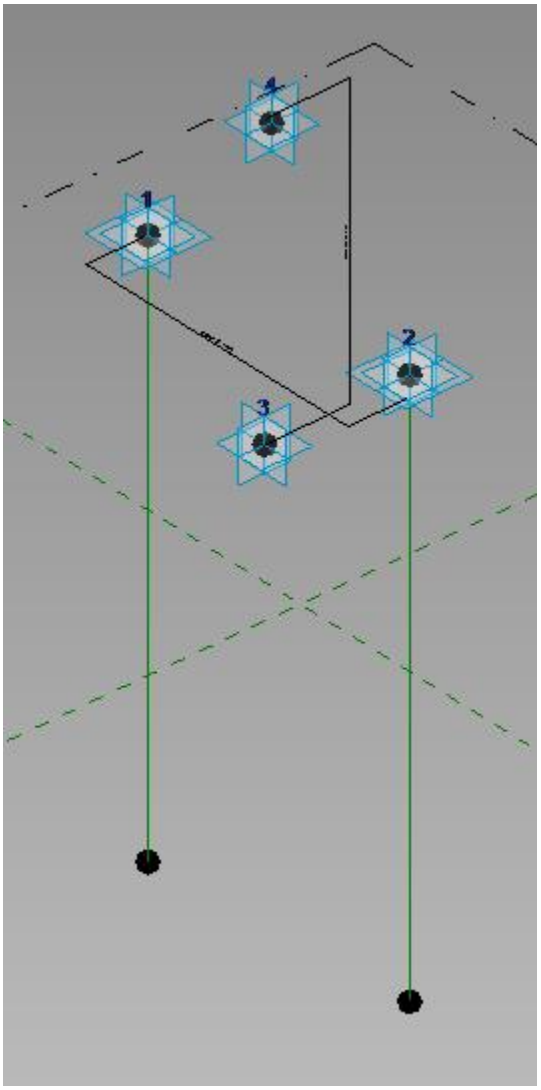

Figure 9: Offset points with reference lines

Points were then hosted on each of the reference lines, and then hosted by intersection to the horizontal plane of the lowest placement point. The intent is to place points that represent the bottom corners of the wall segment. A line was then drawn, connecting these points. This line is the base of the wall. An important thing to note when drawing lines that are intended to connect, and stay connected, to points, is to draw the line with the "3D Snapping" option toggled on. Finally, another pair of lines were offset at the base of the wall, and a line was drawn connecting them. This line will be the symbolic line once in the project.

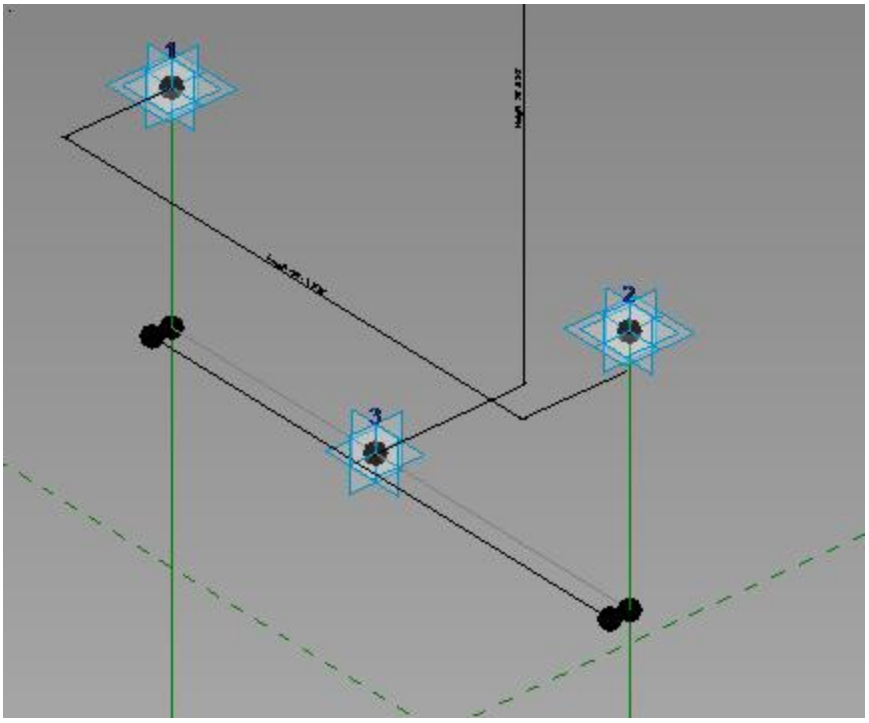

Figure 10: Symbolic lines

At this point the geometry creation of the component is complete. Next, the engineering needs to be added, which is the more difficult task.

In this case, most of the parameters added to the family were shared parameters so they can be scheduled (and tagged) in the project. A series of equations were added to calculate the shear and overturning demand on the wall. Since this family is intended to then design the shear nailing, anchor bolt spacing and hold downs required for the wall, further calculations, logical tests and lookup tables were utilized to facilitate these calculations.

Since calculations, basic formulas and logical tests are relatively common, this example will discuss the lookup table calculations. Lookup tables have been available in Revit MEP for some time, but only now available to everyone in Revit 2014. (Work-arounds and external database links were also available.)

Adding the capability of lookup tables within Revit is wonderful, however, it is not quite everything it seems. The functionality is limited to the basic lookup function, meaning the command takes a value, finds it in one column of the table and reports the corresponding value in another column of the table. While this is useful, it leave a little to be desired when dealing with capacities since the function required is a more robust lookup function that finds the first value in a column that is larger than a given column, and then reports the value in a corresponding column.

For example, this family calculates the shear demand on the wall, and then the desired step would be for the family to look in the table for the first shear wall nailing that gives a capacity greater than the demand. This is possible, however it would require a complicated combination of if-statements and lookup functions. In fact, achieving this would not be any quicker, or any less tedious, than doing the same process with if-statements only (in fact it would be longer), but the tables offer flexibility and edit ability that would be more readily available.

To overcome this challenge, the approach taken by this family was to limit (slightly) the available shear wall types and configurations, and then employ a multi-step process to find capacities. First, given the user-inputted values for desired sheathing type, thickness and nail size, a code was assigned to the desired configuration. Error messages were also embedded such that the user would be notified if the desired configuration is invalid.

The following is the formula used to assign the “SW Type Code”:

=if(and(Struct I, Seismic, SP Thickness = 0' 0 5/16", Nails xd = 6), 1,
if(and(Struct I, Seismic, SP Thickness = 0' 0 3/8", Nails xd = 8), 2,
if(and(Struct I, Seismic, SP Thickness = 0' 0 7/16", Nails xd = 8), 3,
if(and(Struct I, Seismic, SP Thickness = 0' 0 15/32", Nails xd = 8), 4,
if(and(Struct I, Seismic, SP Thickness = 0' 0 15/32", Nails xd = 10), 5,
if(and(Struct I, Wind, SP Thickness = 0' 0 5/16", Nails xd = 6), 6, if(and(Struct I, Wind, SP Thickness = 0' 0 3/8", Nails xd = 8), 7, if(and(Struct I, Wind, SP Thickness = 0' 0 7/16", Nails xd = 8), 8, if(and(Struct I, Wind, SP Thickness = 0' 0 15/32", Nails xd = 8), 9, if(and(Struct I, Wind, SP Thickness = 0' 0 15/32", Nails xd = 10), 10, if(and(Sheathing, Seismic, SP Thickness = 0' 0 5/16", Nails xd = 6), 11, if(and(Sheathing, Seismic, SP Thickness = 0' 0 3/8", Nails xd = 6), 12, if(and(Sheathing, Seismic, SP Thickness = 0' 0 3/8", Nails xd = 8), 13, if(and(Sheathing, Seismic, SP Thickness = 0' 0 7/16", Nails xd = 8), 14, if(and(Sheathing, Seismic, SP Thickness = 0' 0 15/32", Nails xd = 8), 15, if(and(Sheathing, Seismic, SP Thickness = 0' 0 15/32", Nails xd = 10), 16,
if(and(Sheathing, Seismic, SP Thickness = 0' 0 19/32", Nails xd = 10), 17,
if(and(Sheathing, Wind, SP Thickness = 0' 0 5/16", Nails xd = 6), 18,
if(and(Sheathing, Wind, SP Thickness = 0' 0 3/8", Nails xd = 6), 19,
if(and(Sheathing, Wind, SP Thickness = 0' 0 3/8", Nails xd = 8), 20,
if(and(Sheathing, Wind, SP Thickness = 0' 0 7/16", Nails xd = 8), 21,
if(and(Sheathing, Wind, SP Thickness = 0' 0 15/32", Nails xd = 8), 22,
if(and(Sheathing, Wind, SP Thickness = 0' 0 15/32", Nails xd = 10), 23,
if(and(Sheathing, Wind, SP Thickness = 0' 0 19/32", Nails xd = 10), 24,
0))))))))))))))))))))))))

Once the code is determined, the only remaining piece of information to decide is the nail spacing. Since there are a discrete number of nail spacing available, the lookup table was

utilized to find the capacity of each option, then logical statements were again utilized within the family to choose the appropriate spacing and capacity.

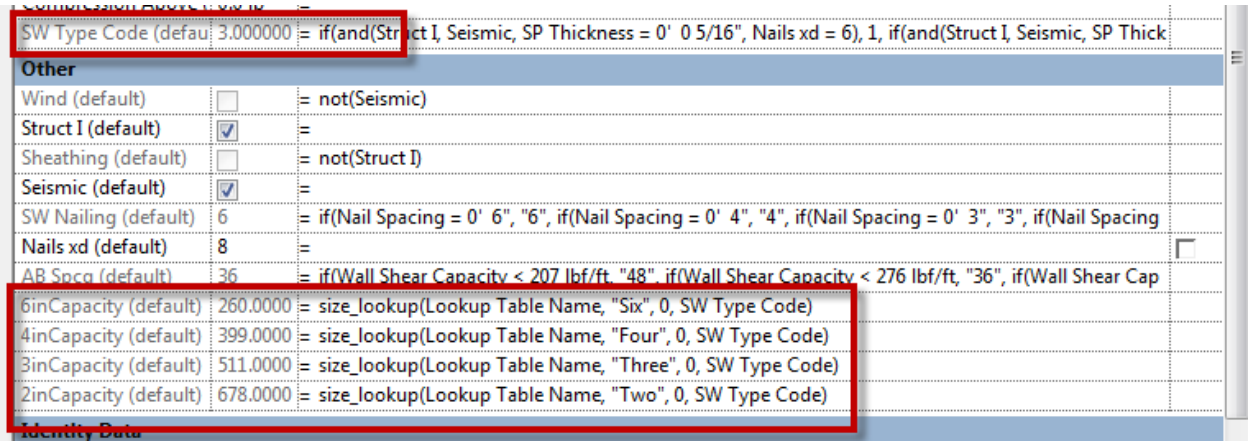

Figure 11: Lookup functions

A few items of note with lookup tables are as follows. First, the tables have to be in a format similar to a type catalog, and have to be loaded into the family (and reloaded if the table is updated). Loading and reloading tables is achieved within the Family Types dialog, using the "Manage" button shown in the image below.

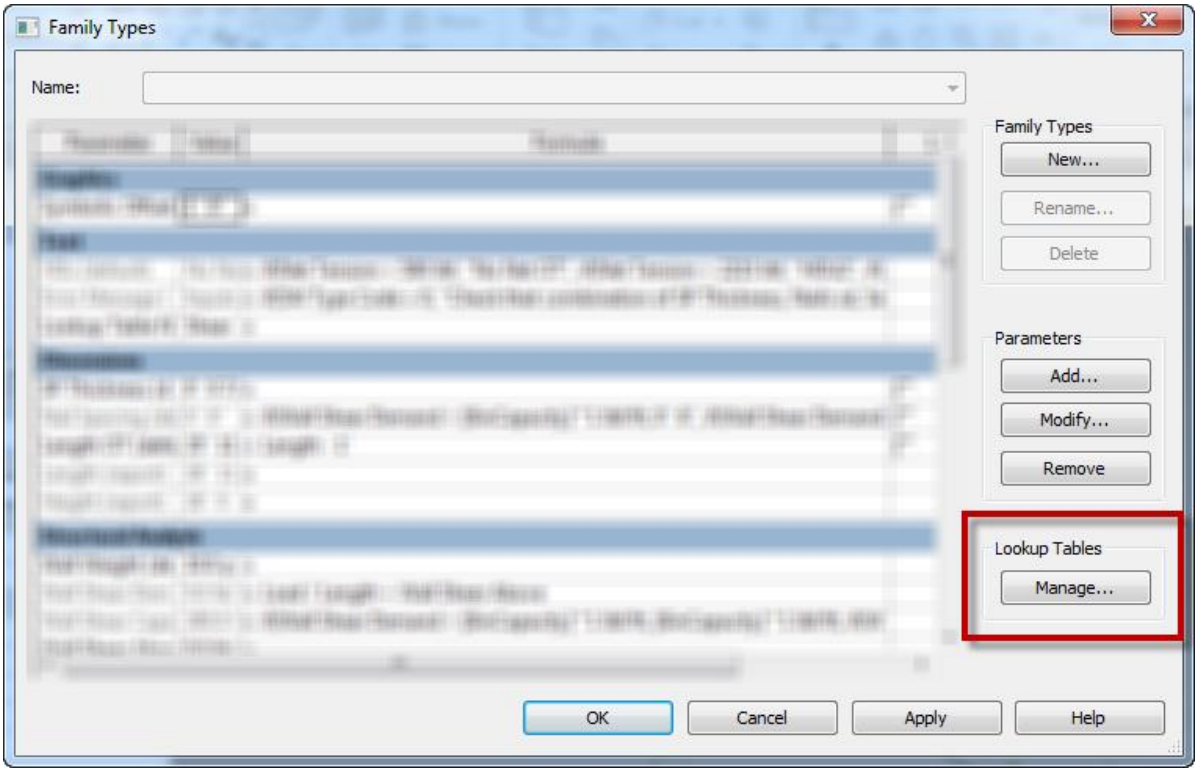

Figure 12: Manage lookup tables

Next, the name of the lookup table has to be entered as a parameter value within the family. Finally, the format of the lookup function is: “size_lookup(parameter where table name is entered, column from which to report, default value if a value is not found, parameter name of value to be looked up). Also note that the first column is skipped when looking for the value.

Once all calculations were added to the family, the only remaining task is to make tags for the family. A few custom text parameters were added, and then a couple basic custom tags were created to report the calculated shear wall nailing, anchor bolt spacing and hold downs.

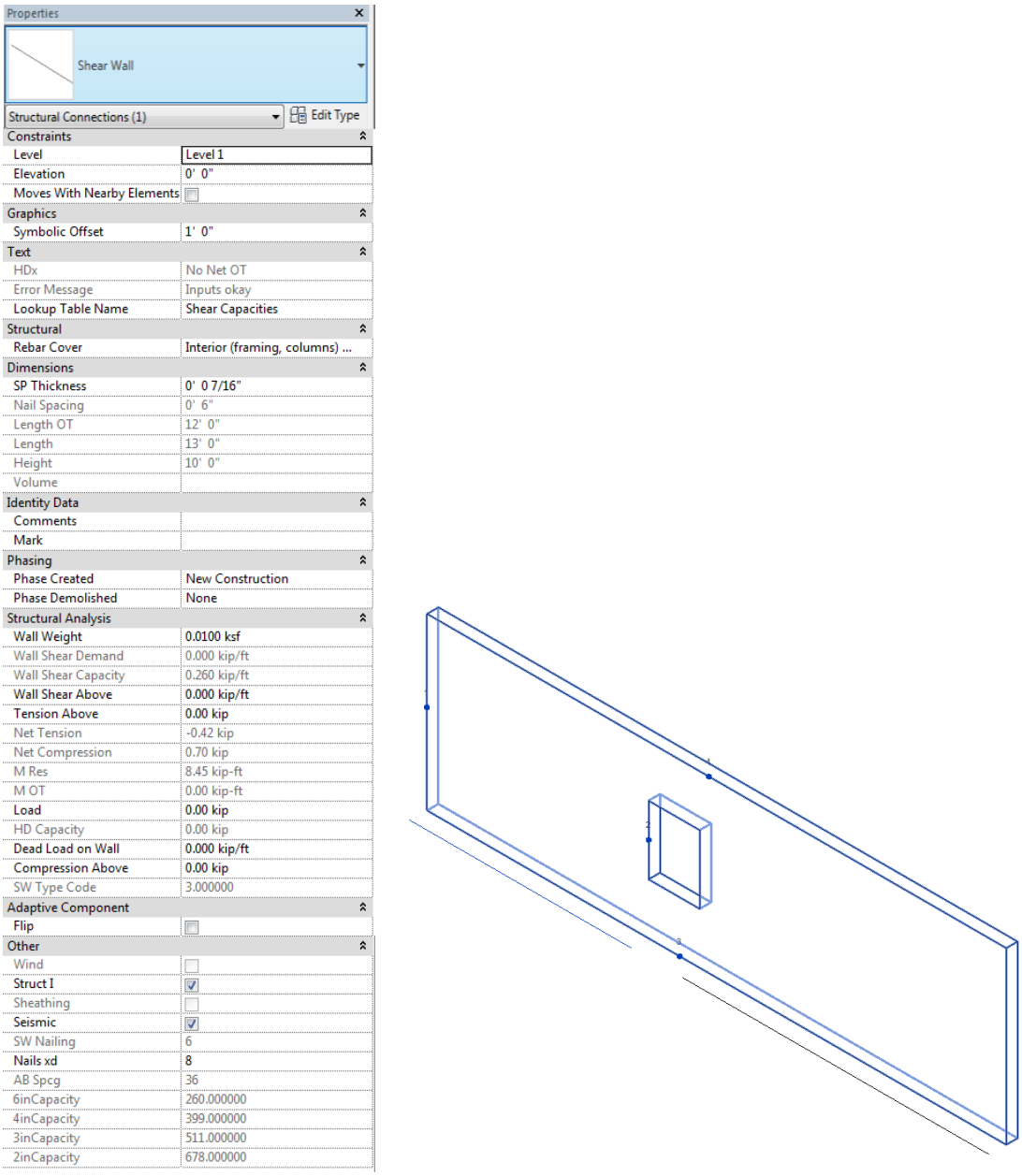

Figure 13: Component placed in project

The above image shows the component as it would appear in the project environment. This component is an excellent example of maximizing what Revit can do – not only does the component provide useful graphics, but it also provides design and calculations. As it has been updated over the years, this component has evolved to incorporate many things, and will likely further evolve and grow in the future, especially if additional functionality is added to the lookup tables and lookup functions.

Slender Column (Bonus Material)

This family is discussed in the companion class "Incorporating Engineering into Autodesk Revit Structure: Project Procedures" as an example of a family that facilitates the use of working schedules, conditional formatting, view filters and custom formatting, but it also is an example of a family that performs calculations. The family utilizes custom parameters and calculations such that, once placed in a project, the column calculates its slenderness ratio and its axial compressive capacity.

This family is simply the out-of-the-box structural steel column family, with only one addition. A reporting parameter was added such that the height of the column reports once placed in a project. The reporting parameter dimensions the levels within the family since it will be used in formulas. (Reporting parameters need to reference host elements in order to be used in formulas.)

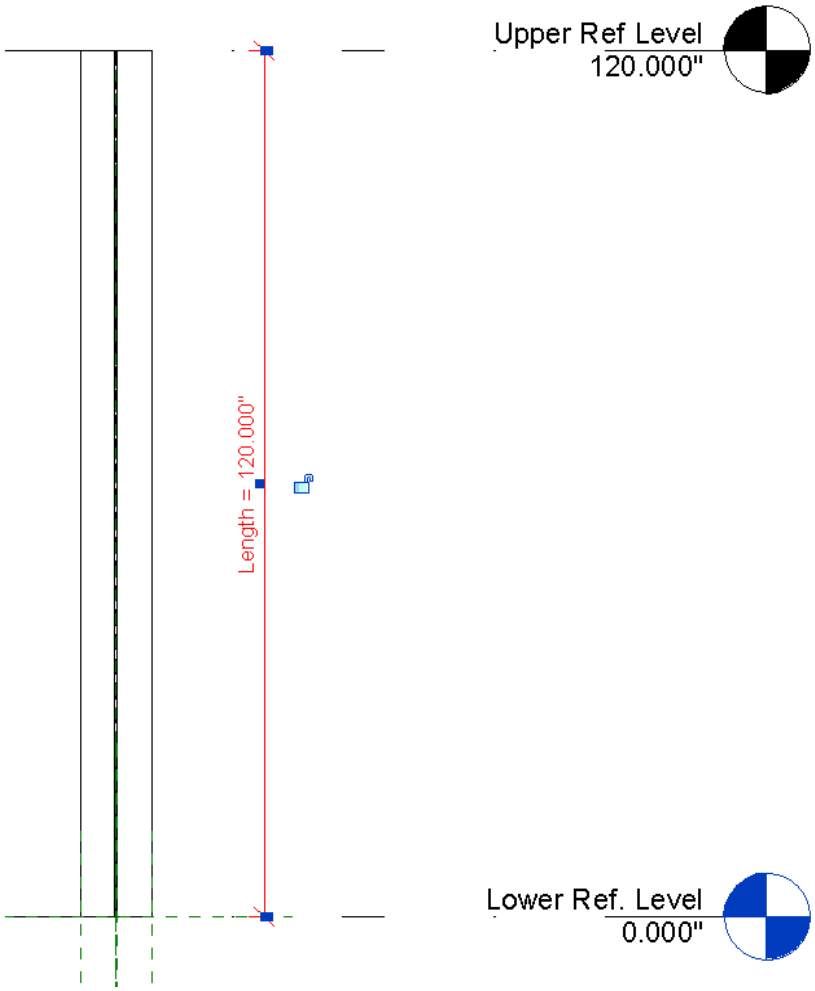

Figure 14: Reporting parameter

Parameters and calculations were added to the family.

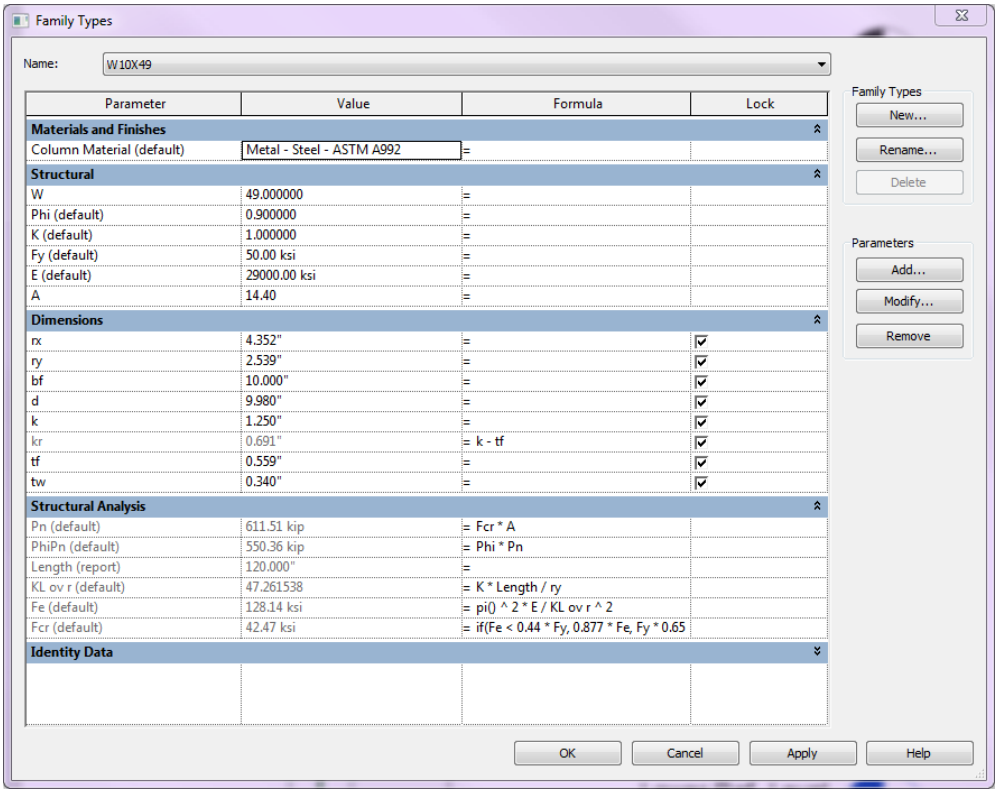

Figure 15: Column calculations

Also, since the out-of-the-box column's type catalog does not include some of the geometric parameters required for the calculations, the type catalog also was edited.

With these small, relatively simple, edits to the out-of-the-box column, some engineering capabilities were added to the element, and the same process could be employed to display other sorts of engineering information in any number of other families.

Families that Design Themselves

It is possible to not only perform calculations within families, it is possible to have those calculations drive sizes, configurations, etc. of families such that they could essentially design themselves.

Pad Footing

The first example of this idea is a pad footing that adjusts its size based on its load demand. To calculate the required footing size, the values required are area load demands (Dead, Live, Snow, etc.), the area that is tributary to the footing, and the allowable soil bearing capacity.

Starting with the out-of-the box family, a few shared parameters were added such that the engineer can enter area loads values and soil bearing pressures. In this cases, these values were set within the family, but would be easily modified later. The only real challenge lies in determining the area tributary to the footing, since that information is only available once the footing is placed into a project. Since reporting parameters can only be used in formulas if they dimension reference values, extracting the tributary area was achieved by placing reference planes that would be aligned and locked to grids once placed in the project.

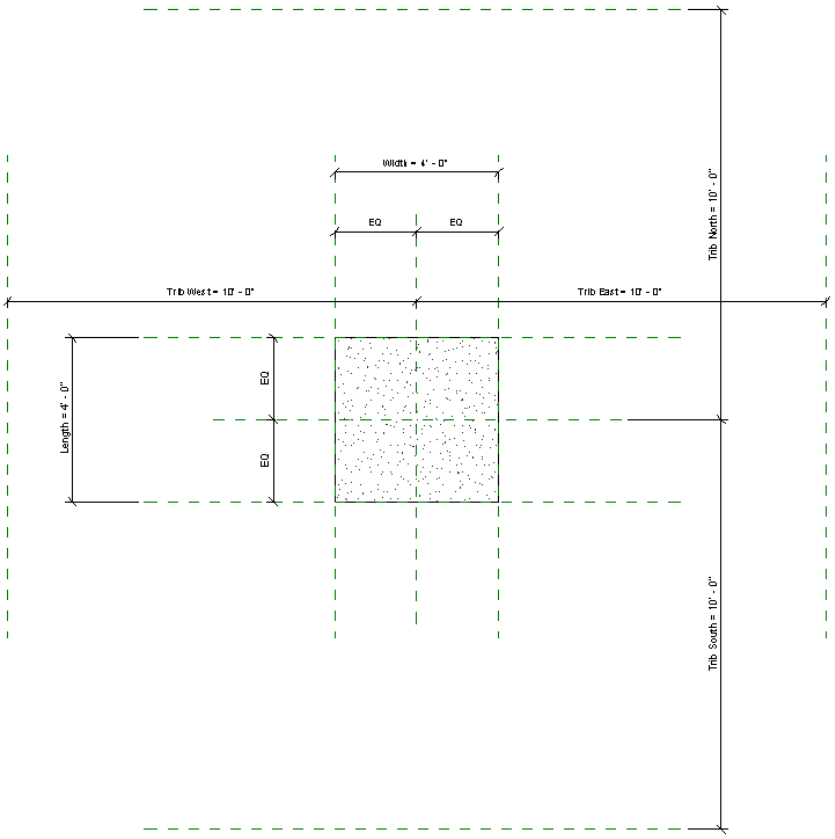

Figure 16: Footing in family editor showing tributary area dimensions

Once the tributary area was calculated, the total load demand was easily calculated, and from the demand, the required footing area was calculated. Then, using instance-based geometry parameters, the footing size automatically updates. In this case, the footing thickness was set to a constant value, but it could easily be calculated as well.

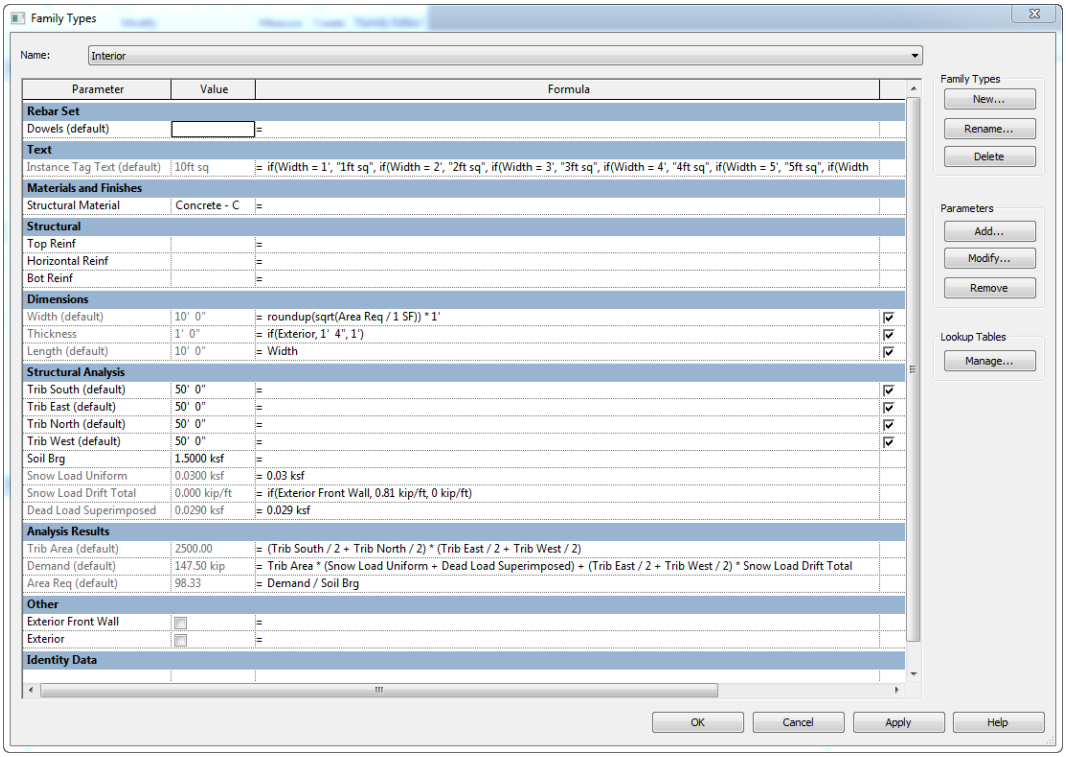

Figure 17: Engineering incorporated into footing family

Once created, this family could be placed in a project and the engineer would know that the basic design of the footings would update as the model evolves.

Finally, since the geometry is instance-based, instead of type-based, a custom instance-based tag was created for the footing. This was achieved by including a simple text parameter and a custom tag that look to the value in that parameter.

Structural Column

To design the columns, the height and load demand are required. Similarly to the footings, the tributary area was determined within the column using reference lines that are aligned and locked to grids once placed in the project. Given the area loads and the tributary area, the load on the column can be calculated.

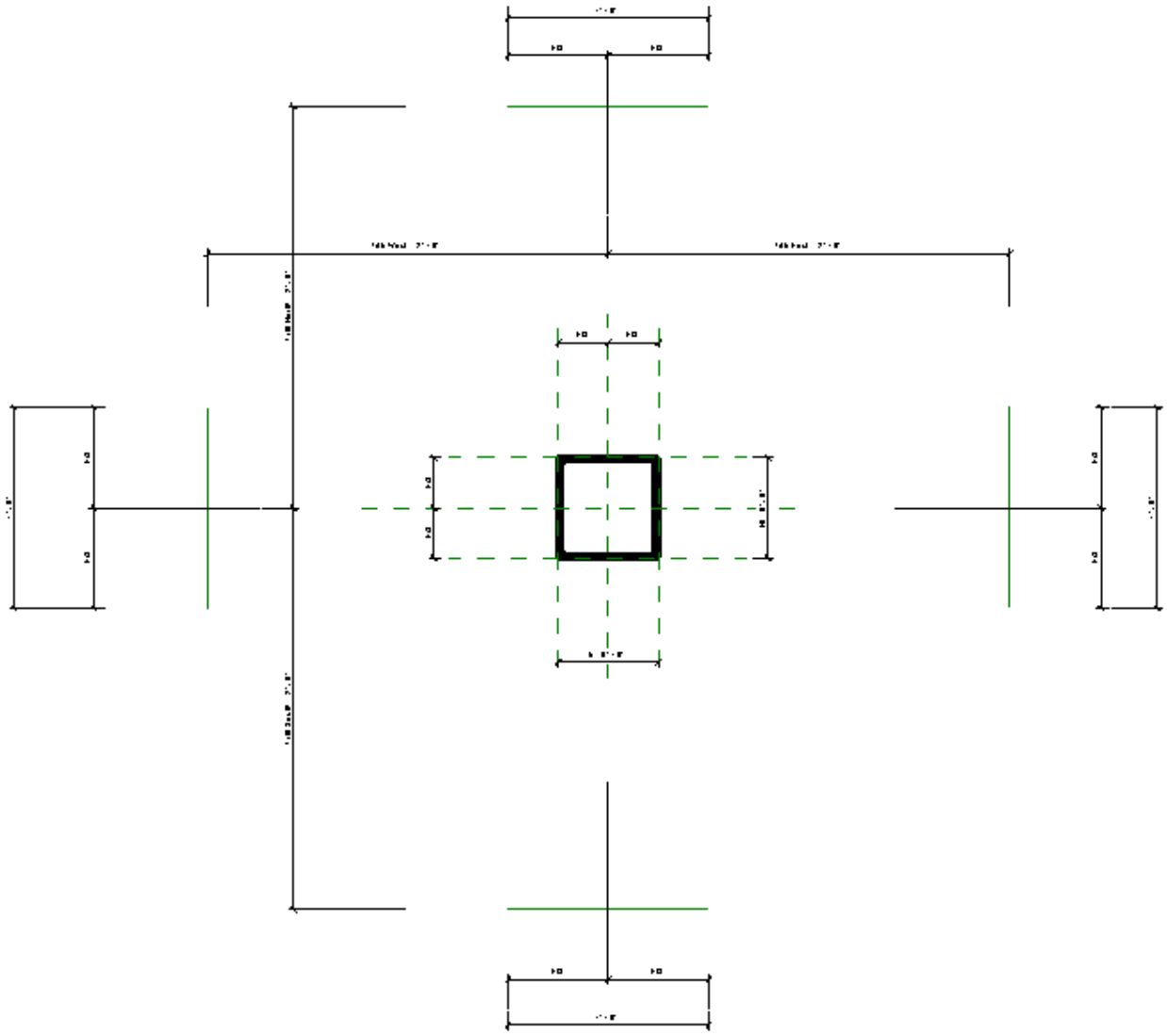

Figure 18: Column tributary area

Then, using the same process as in the slender column example, a reporting parameter was added to determine the height of the column, and since it dimensions host elements, the length can be used in formulas.

Figure 19: Column height

Although all the required information is known, there are too many adequate HSS members to end here, and as described in the shear wall family example, lookup tables don't quite work for this application. So, six preferred sizes were pre-selected, and the family selects from only these columns. This is a shortcoming of the family, but a necessary simplification. Once the size is determined, the family updates the instance-based geometry of the column.

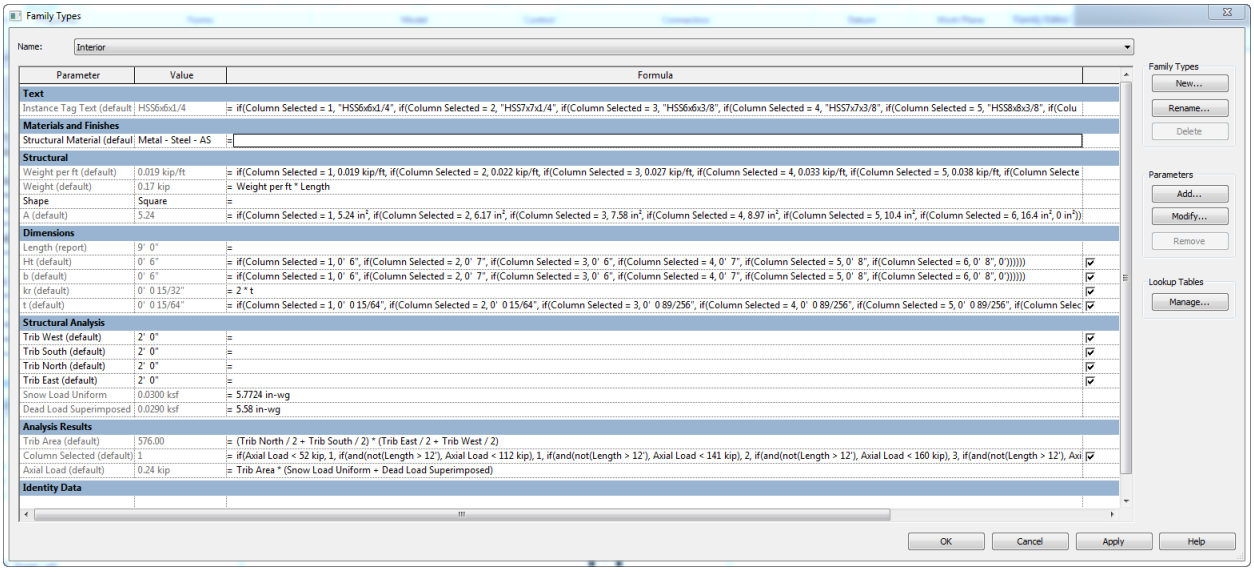

Figure 20: Column calculations

As was true with the footing, an instance-based tag is used to tag the column.

This idea of inputting minimal information and extracting dimensions and other information out of the project once the family is placed can be applied to any number of other structural elements. Furthermore, these elements could be combined into a project such that nearly every element in an entire building could “design itself”.

Deflecting Double Tee (Bonus Material)

This family is covered in another class, “SE1595: Using Autodesk® Revit® Structure in Investigative Engineering”, but it is another great example of a family that incorporates engineering. This family was created for a specific client and project, so it has a very specific purpose. Given several user-inputted values, including a value for a “rain event”, the family calculates the expected mid-span deflection and updates the geometry of the beam to show the deflection.

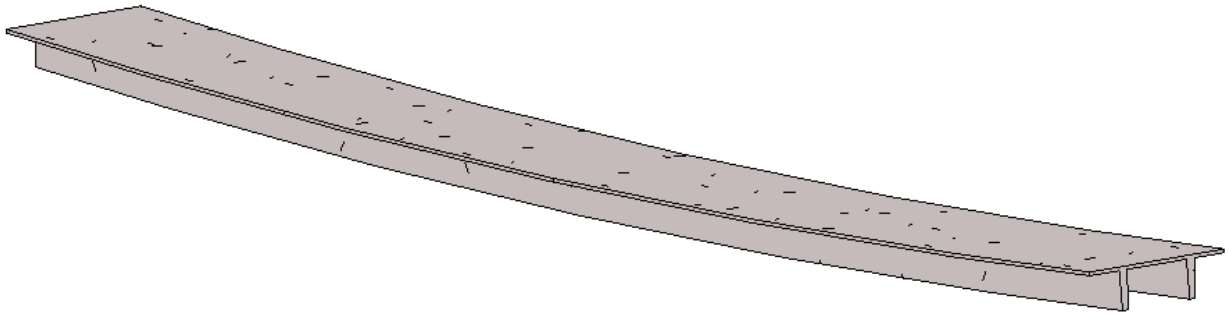

Figure 21: Double tee deflected shape

This, combined with an adaptive component that is intended to show the water on the roof, creates a visual aid for exactly what occurs on the roof when it rains.

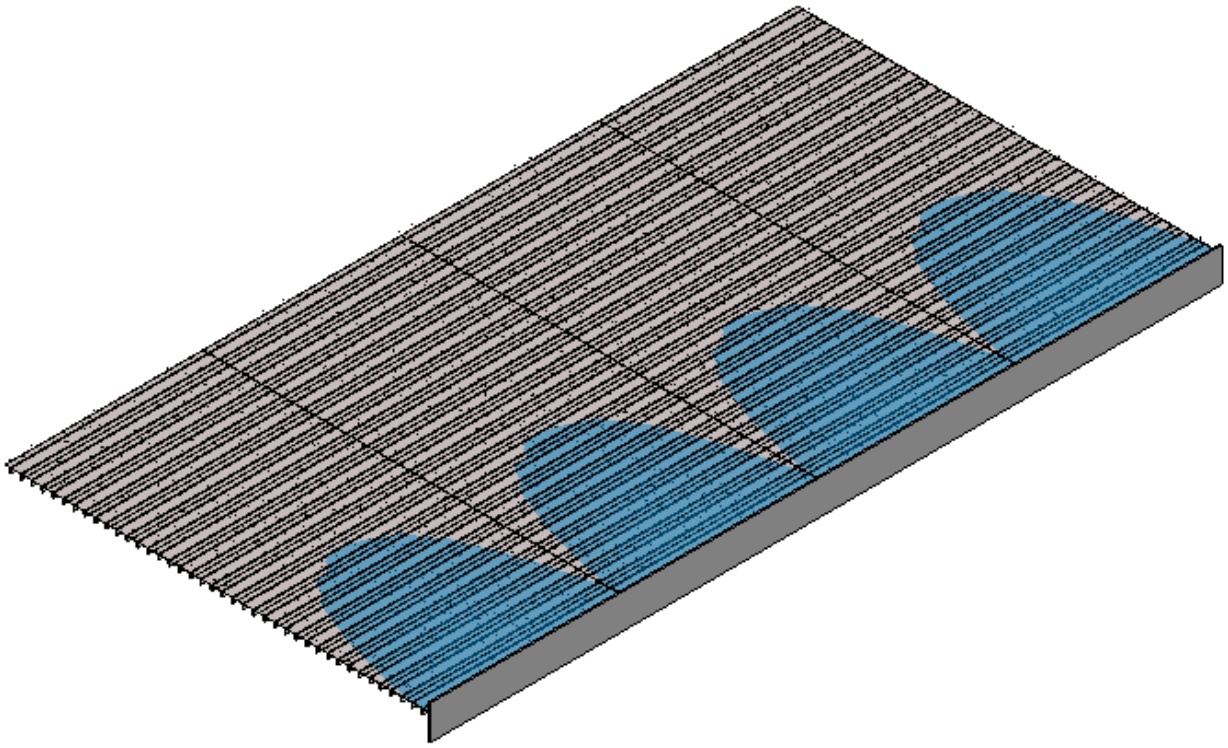

Figure 22: Rain on roof

The input given was an initial height of water at a parapet wall, which was called a “rain event”. The roof sloped up away from the parapet wall, so the amount of water observed at the wall would not be the same amount of water at each double tee as we moved away from the wall. In addition, the double tees were installed with an initial midspan deflection of ½”.

So, the first task was to create a double tee family that would show the deflected shape. This was achieved by using the out-of-the-box family and modifying the beam’s sweep to be an arc. The arch was then constrained by using the radius (since directly constraining the deflection proved to be problematic). Using basic geometry, the radius of the arc can be calculated from the other dimensions.

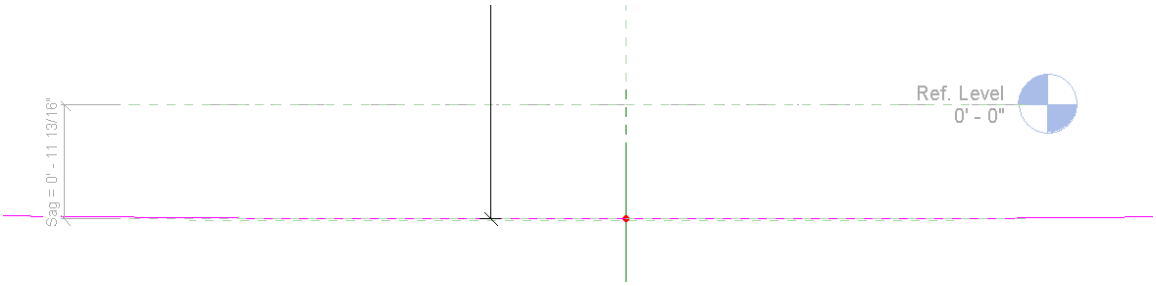

Figure 23: Double tee sweep

The next step was to incorporate all of the math and engineering. Using an assumption that water would puddle at the wall, the low point, and the depth over any given beam would be linearly related to both the height of water at the wall and the distance of the beam from the wall, the family calculates the final height of water at each beam. It then converts this height of water to a volume, and then to a weight. Given the weight, or load, on any given beam, the deflection at midspan was determined based on some test data that was provided. Then, given this final deflection amount, the radius of the arc-shaped sweep was determined, and the deflected shape of the beam is shown.

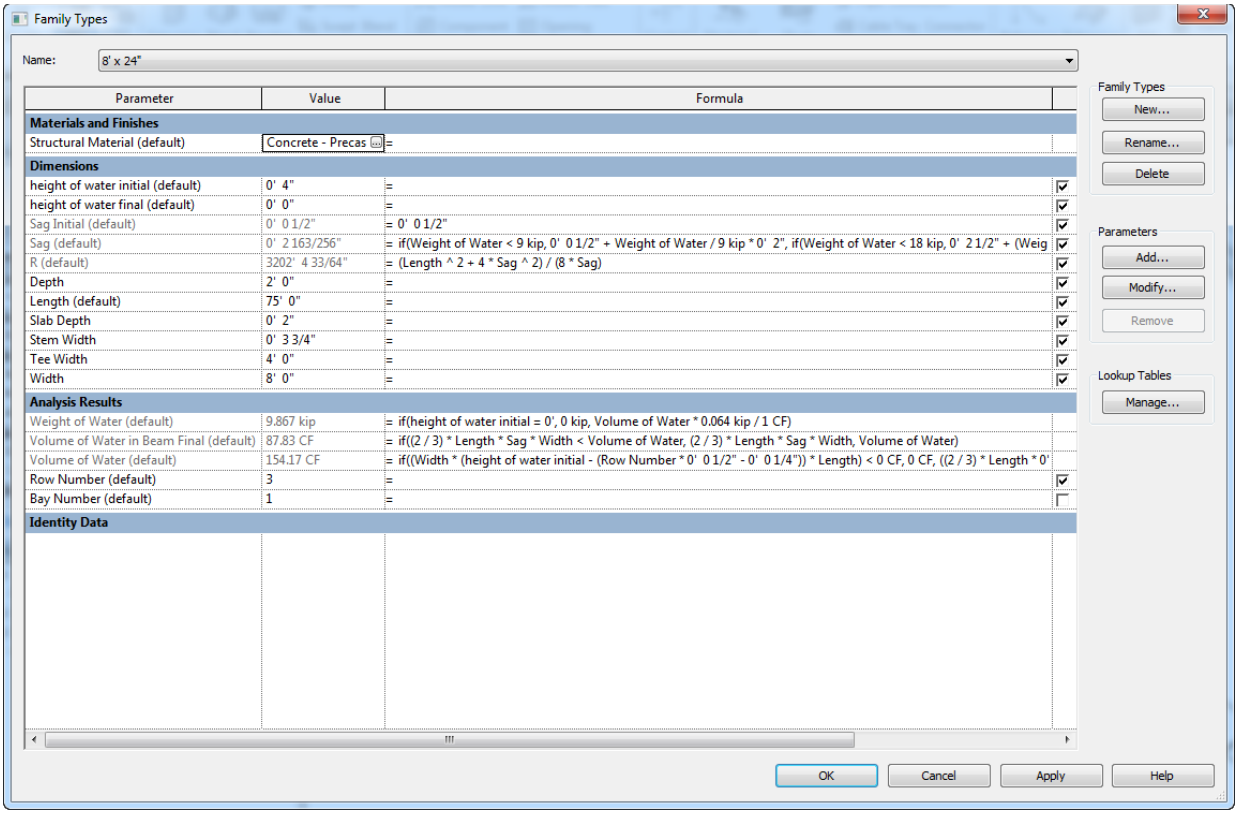

Figure 24: Double tee calculations

Below shows two conditions: an initial condition with no water, and then the approximate deflected shape given a 10” initial height of water.

Figure 25: Double tee initial shape

Figure 26: Double tee deflected shape

While the application of this family is rather specific, the idea behind what was completed could apply to any number of situations. Combining engineering information with Revit with this approach provides a way to achieve not only a visual representation, but also a way to simplify and compile engineering and calculations that may otherwise be rather repetitive and iterative.

Families for Project Workflows

Markup Symbol

This final example is simply a custom annotation family that can be utilized for an internal (or possibly even external) workflow process. This custom family is a symbol that can be placed in various views throughout a project, and then can be scheduled to aid in communication among a project team.

The family began with the "Generic Annotation" family template. The geometry in the family is simple; it is just a few different colored filled region and some text. The different colors are used for different types, so their visibility parameters are linked to some yes/no type parameters.

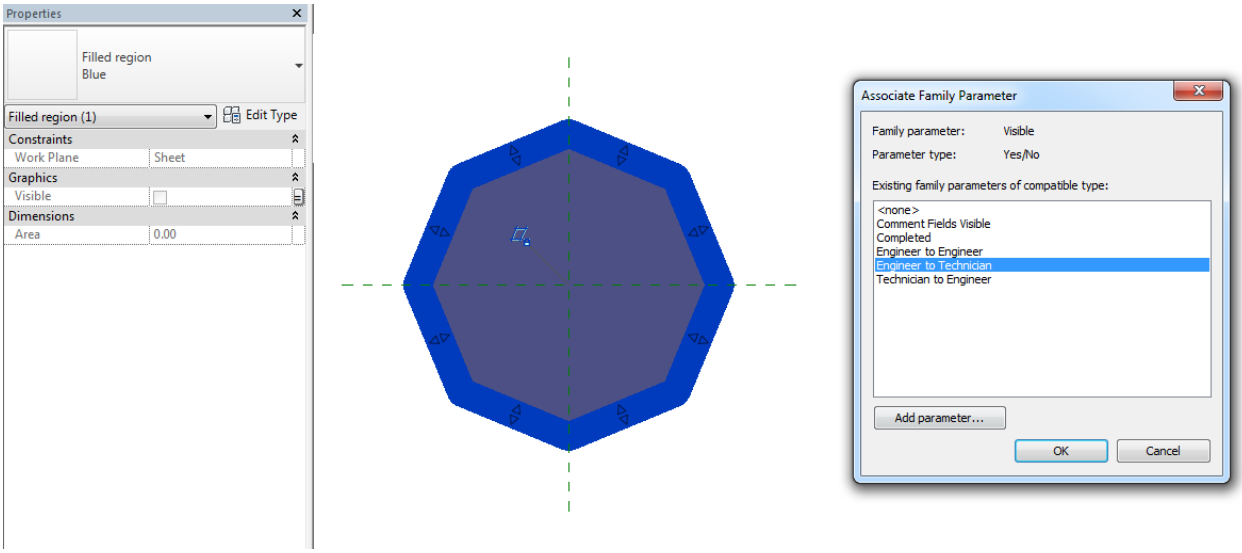

Figure 27: Symbol types

Several text parameters were then added to allow the user to input various types of information.

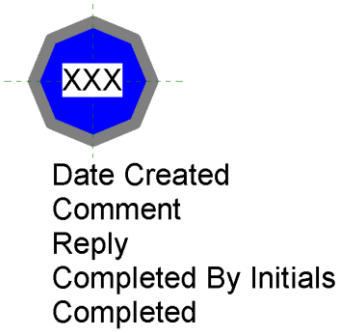

Figure 28: Workflow symbol text fields

One item to note is that none of these parameters are shared because generic annotations can be scheduled in a note block, so none of the parameters need to be shared in order to appear in

the schedule. The following image is how the symbol would look once placed in the project environment.

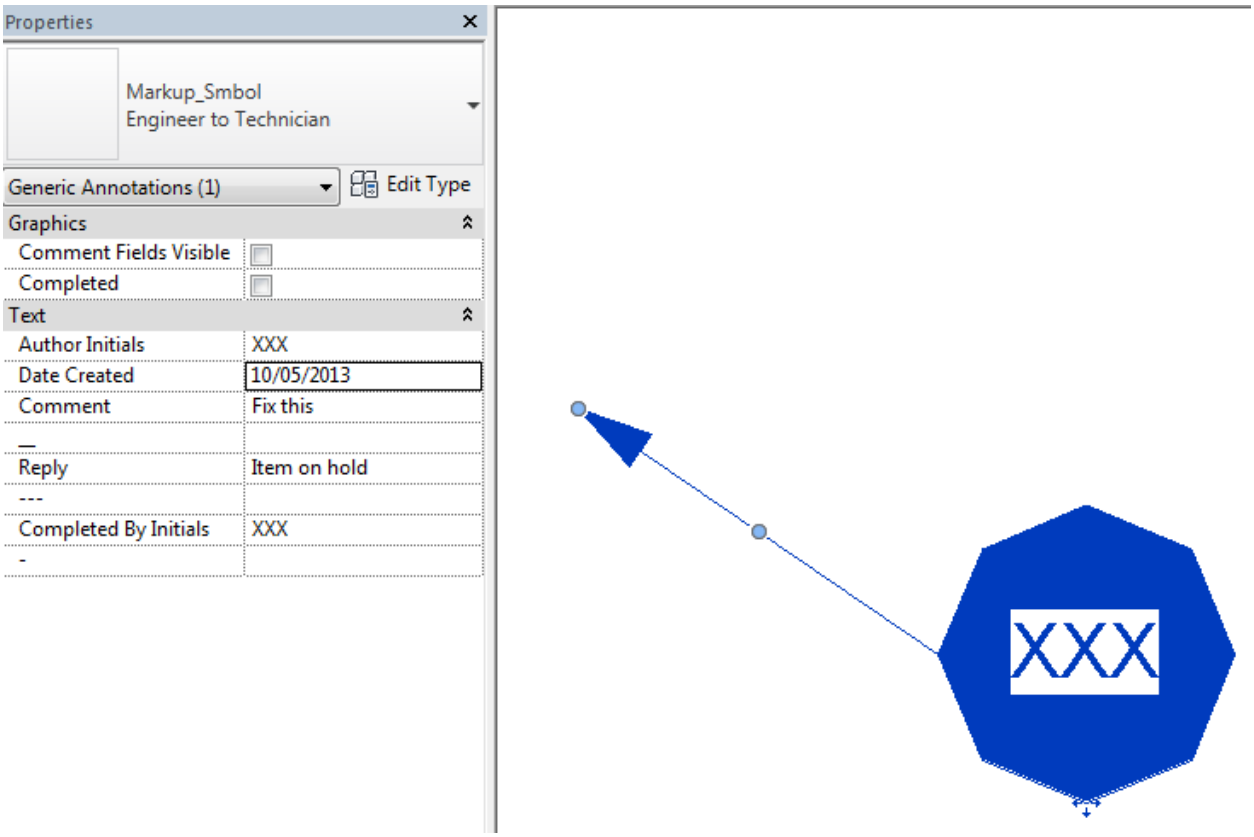

Figure 29: Mark-up symbol

As is discussed in the companion class “Incorporating Engineering into Autodesk Revit Structure: Project Procedures," the user would then utilize a schedule of these elements to complete the workflow.

This particular workflow family is something the author created many years ago, but it seems that these simple items are the ones that have the largest effect on efficiency, so it seemed relevant to share in this section.

Conclusion

This session shared a few possible applications, but many various custom families, to incorporate engineering and design into Revit families. It is this author’s opinion that the potential for such integration is vast and ever-growing. In the companion class, "SE1592 Incorporating Engineering into Autodesk® Revit® Structure: Project Procedures," the uses of the custom families are discussed and demonstrated within the project environment.