Final Documentation and Notes - GalloTools

Adam Ledet, Arturo Altamirano

Synopsis of Work Performed

1. **Streamlining/refining of code** has reduced codebase size significantly. Approx. 5,000 lines of repetitive code have been removed from 'CreateTab' which is acting as 'main'. We have moved most major functions (which replace the repetitive blocks) to their own separate file to make the system more modular, and object oriented. Functionality which benefited most from this was the Filter, LengthCheck and PlaceHangers tools.
2. **Introduction of ToolTips and ToolTip videos** provides the opportunity to better document functionalities to make it easier for users to understand and navigate the toolkit.
3. **Added Families to the Filter tool** along with algorithmic refinement has enabled users to filter basically every object possible within a given schematic view and to filter by multiple criteria at once. Optimizations not only in spatial complexity but also algorithmic efficiency has enabled quick and robust filtering processes.
4. **Introduced the DockablePanel** to enable the user to persistently store and view information from important actions. Many functionalities are written to the panel, which can be removed, cleared or docked anywhere the user likes.
5. **Overhaul of the PlaceHangers and HangerSpecs functions** has introduced new functionalities such as filtering and optimizations to transaction structure have provided a more seamless user experience.
6. **TrapezeHanger Functionality** has been introduced, with a clone function and a manual spec function enabling user convenience and high degree of control in placing trapeze along a run of pipe.
7. **Various micro-functions** improve quality of life/provide utility for users. Such functions include: GetParameters, FindDistance, AlterHangerHost-Remove, AlterHangerHost-Swap, and StringFormat

GalloTools Classes

Below are the classes we primarily edited during our time as well as a description of what the classes contain.

- **auxiliaryFunctions.cs** - functions used by other classes to abridge code. Includes functions such as:
    - InternetConnection - Checks if the user is connected to the internet.
    - SnowflakeDatabase – Passes through Snowflake and records a tool's use.
    - PopulatePanel – Adds information to the Dockable Panel. Has many overrides to display different types of information to the panel.
- **CreateTab.cs** - Main. This creates the ribbon as well as most button functions.
- **DockablePanelHelper.cs** - Class used to create teh Dockable Panel.
- **ElementTracker.cs** - Class with several functions pertaining to the Get Parameters tool.
- **LengthChecker_Helper.cs** - Massive amounts of hard-written code is used for the LengthChecker function. All of it is kept here in a single function that is called whenever any LengthCheck function is called. A switch statement regulates what parameters to check.
- **PipeFilter_Helper.cs** - Contains functions used to actually select Pipes, hangers, or Families in the ModPipeFilter tool.
- **ToolTipVid.cs** - Class containing specifically the SetRibbonItemToolTip function used to set a video for a tooltip.

Refinement

There were many repetitive blocks within the code when we first began working on this project. We figured that by taking these apart and developing optimizations, we would develop a proper understanding of the tools along the way. We do not know exactly, but we feel as though we have cut down at the very least 5,000 lines of code in total across the project.

An example of a repetitive block, repeated 20+ times

```
if (sizestr.Length > 0)
{
    string reflevel = refparam.AsValueString();
    string service = serviceparam.AsString();
    string cleansize = sizestr.Replace("\"", "").Replace("'", "").Replace(" ", "-").Replace("-x-", "x") + "\"";
    if (mats.Contains(mat))
    {
        ElementId elemid = element.Id;
        matelems.Add(elemid);
    }
    if ((mataltnum == "39" || mataltnum == "40") && mats.Contains("Stainless Steel"))
    {
        ElementId elemid = element.Id;
        matelems.Add(elemid);
    }
    if (mataltnum == "56" && mats.Contains("PEX"))
    {
        ElementId elemid = element.Id;
        matelems.Add(elemid);
    }
    if (mat.Length < 1 && purematname.Length > 0 && mats.Contains(purematname))
    {
        ElementId elemid = element.Id;
        matelems.Add(elemid);
    }
    if (mats.Contains("Cast Iron") && mat == "Black Plasticized PVC")
    {
        ElementId elemid = element.Id;
        matelems.Add(elemid);
    }
    if (mats.Contains("PVC") && mat == "Black Plasticized PVC")
    {
        ElementId elemid = element.Id;
        matelems.Add(elemid);
    }
    if (refs.Contains(reflevel))
    {
        ElementId elemid = element.Id;
        refelems.Add(elemid);
    }
    if (services.Contains(service))
    {
        ElementId elemid = element.Id;
        servelems.Add(elemid);
    }
    if (sizes.Contains(cleansize))
    {
        ElementId elemid = element.Id;
        sizeelems.Add(elemid);
    }
    else if (name.Contains("Pulled"))
    {
        string[] sizeparts = cleansize.Replace("\"", "").Split('x');
        string branch = sizeparts[0];
        string main = sizeparts[1];
        string newsize = main + "x" + branch + "\"";
        if (redsizes.Contains(newsize))
```

Below are the optimized objects designed for re-reference (8 total references now):

```
foreach (Element element in collector)
{
    ElementId id = element.Id;
    allelems.Add(id);
    Parameter matparam = element.get_Parameter(BuiltInParameter.FABRICATION_PRODUCT_DATA_MATERIAL_DESCRIPTION);
    Parameter matparamalt = element.get_Parameter(BuiltInParameter.FABRICATION_PART_MATERIAL);
    Parameter sizeparam = element.get_Parameter(BuiltInParameter.FABRICATION_PRODUCT_ENTRY);
    Parameter prisizeparam = element.get_Parameter(BuiltInParameter.FABRICATION_PRI_SIZE);
    Parameter nameparam = element.get_Parameter(BuiltInParameter.ELEM_FAMILY_PARAM);
    string name = nameparam.AsValueString();
    Parameter refparam = element.get_Parameter(BuiltInParameter.FABRICATION_LEVEL_PARAM);
    Parameter serviceparam = element.get_Parameter(BuiltInParameter.FABRICATION_SERVICE_NAME);
    string mat = matparam.AsString();
    string mataltnum = matparamalt.AsInteger().ToString();
    string mataltname = matparamalt.AsValueString();
    string purematname = mataltname.Replace("Pipework: ", "").Replace("vbs - Piping: ", "");
    string sizestr = sizeparam.AsValueString();
    string prisizestr = prisizeparam.AsValueString();

    if (sizestr.Length > 0)
    {

        string reflevel = refparam.AsValueString();
        string service = serviceparam.AsString();
        string cleansize = sizestr.Replace("\"", "").Replace("'", "").Replace(" ", "-").Replace("-x-", "x") + "\"";

        //think reversing the logic is better, prune the niche/unique cases first
        if (name.Contains("Pulled"))
        {
            string[] sizeparts = cleansize.Replace("\"", "").Split('x');
            string branch = sizeparts[0];
            string main = sizeparts[1];
            string newsize = main + "x" + branch + "\"";
            if (redsizes.Contains(newsize))
            {
                matelems = PipeFilterAuxilliaryFunctions.TryAddElement(matelems, element);
            }
        }

        //group these 4 distinct cases
        else if (refs.Contains(reflevel) || services.Contains(service) || sizes.Contains(cleansize) || (redsizes.Contains(cleansize)))
        {
            if (refs.Contains(reflevel))
            {
                refelems = PipeFilterAuxilliaryFunctions.TryAddElement(refelems, element);
            }
            if (services.Contains(service))
            {
                servelems = PipeFilterAuxilliaryFunctions.TryAddElement(servelems, element);
            }
            if (sizes.Contains(cleansize))
            {
                sizeelems = PipeFilterAuxilliaryFunctions.TryAddElement(sizeelems, element);
            }
        }
        matelems = PipeFilterAuxilliaryFunctions.EvaluationLogic(matelems, element,
                                                                 mataltnum, purematname, mat, mats);
    }
```

By grouping the main categories together into their own series of evaluations, instead of enumerating all of them repetitively with the block displayed prior, you can save on time and space complexity.

Notice the use of sub-functions to isolate a given portion of the evaluation; by doing this, you limit the verbosity of the code while also making troubleshooting and modification easier.

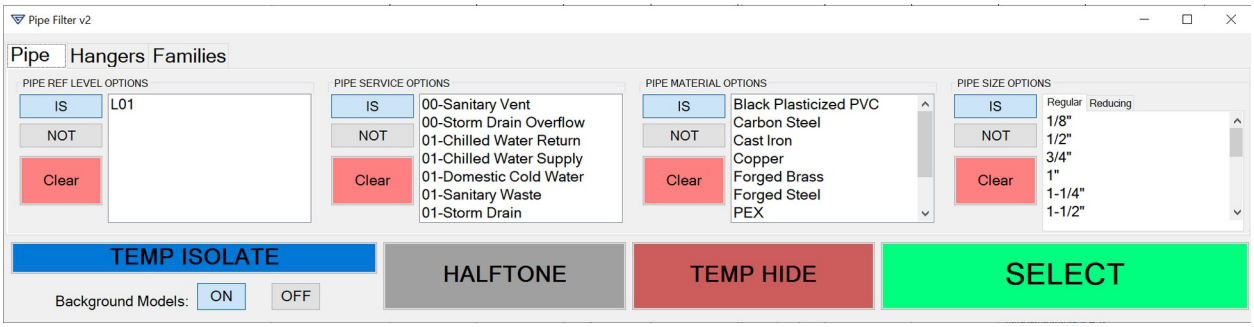
Abstraction/isolating pieces of a process is generally considered good practice, to an extent.

ToolTips

Animated Tooltips can be created with the SetRibbonItemToolTip function. Information regarding the tool can be found in the ToolTipVid class. Animated videos can include sound and will play exactly once while the user hovers over the tooltip.

```
// Update Tooltip given Title, Content, Custom Content, Expanded Content, Image, Expanded Image, and Expanded Video
// Leaving any string parameter blank with "" allows you to add additional parameters in the list.
0 references
public static void SetRibbonItemToolTip(RibbonItem item,
    string title="", string content="", string customContent="", string expandedContent="",
    System.Windows.Media.ImageSource img=null, System.Windows.Media.ImageSource expandedImg=null,
    string videoFilePath= @"Z:\Shared\Public\Gallo BIM\GalloTools\Resources\boogieLiberace.mp4")
{
    //Create Abstract Class
    IUIRevitItemConverter itemConverter =
        new InternalMethodUIRevitItemConverter();

    //Use Abstract Class's GetRibbonItem method to convert item into Autodesk.Windows.RibbonItem
    var ribbonItem = itemConverter.GetRibbonItem(item);
    if (ribbonItem == null)//Leave function early if ribbon item is empty
        return;

    // Populate the new Window Tooltip
    ribbonItem.ToolTip = new RibbonToolTip()
    {
        Title = title,                          // Top of ToolTip in bold
        Content = content,                      // Description
        CustomContent = customContent,          // Bottom of Tooltip
        ExpandedContent = expandedContent,      // Second Description appearing after tooltip expands
        Image = img,                            // Image in Tooltip description [Adds White Background to transparant images]
        ExpandedImage = expandedImg,            // Image appearing after tooltip expands [Image can have transparant background]
        ExpandedVideo = new Uri(videoFilePath)  // Video that plays after tooltip expands [Video can have sound!]
        // Image window size max is ~325x325px. (It should be 330x330px, but that still looks like it clips slightly)
        // The ribbon window appears a a fixed length of up to ~360px based on the description and image lengths.
        // Any further, and the image/video clips.
    };
}
```

Families and the Filter Tool



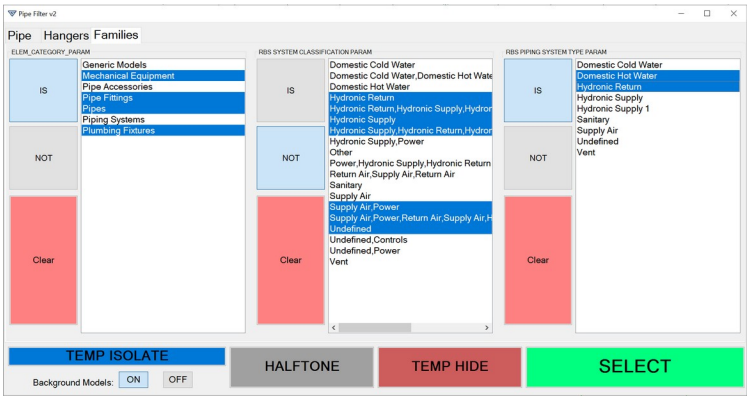
Filter, as well as other forms, use responsive design. Affected windows will automatically appear in the center of the user's screen and size themselves to take up ~50% of the user's screen resolution. They may also be manually resized in all directions.

Responsive design is achieved in both X and Y directions by placing columns and rows into TableLayoutPanels.

```
// Get Families
4 references
public static IList<ElementId> GetFamilies(IEnumerable<Element> allfamelements,
                                           List<String> famcats, List<String> famservices, List<String> famsystemtype,
                                           System.Windows.Forms.RadioButton IsCatButton, System.Windows.Forms.RadioButton NotCatButton,
                                           System.Windows.Forms.RadioButton IsServFamButton, System.Windows.Forms.RadioButton NotServFamButton,
                                           System.Windows.Forms.RadioButton IsSystemType, System.Windows.Forms.RadioButton NotSystemType)
{
```

```
// Get Families
if (famselected.Count > 0)
{
    famelements = PipeFilter_Helper.GetFamilies(allfamelements,
                                                famcats, famservices, famsystemtypes,
                                                IsCatButton, NotCatButton,
                                                IsServFamButton, NotServFamButton,
                                                IsSystemType, NotSystemType);
```

Example of function call used in 'Select Pipes' rather than re-writing code to find families in each button of the Filter tool. In total, ~10,000 lines of redundant code removed using this optimization. This also means that functionality changes need only be done once in the appropriate location.

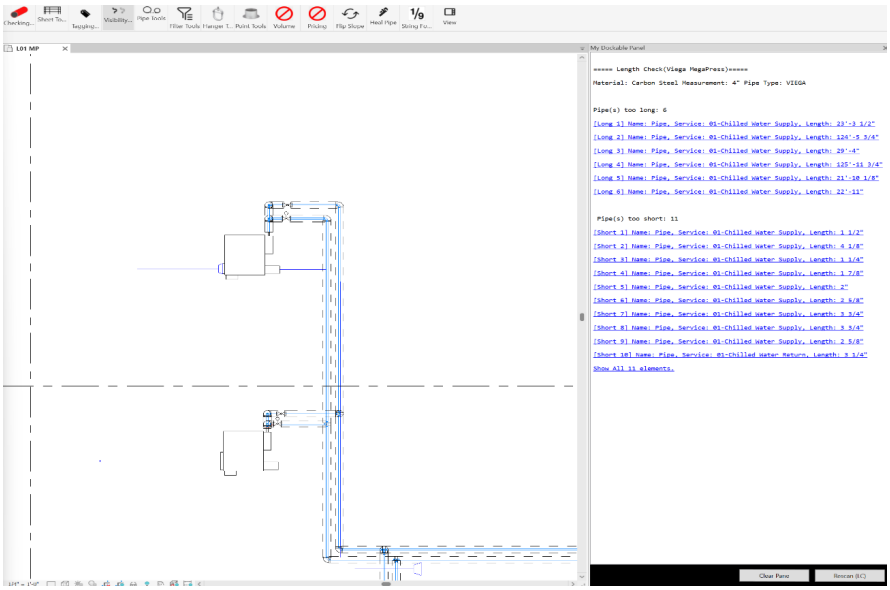


Made selection for IS and NOT MultiExtended, meaning they can be used with click-and-drag or by using CTR to select multiple different entries. The CLEAR button unselects all

entries from the list; selecting one item without pressing CTR will unselect everything except the chosen item for a given section.

The Dockable Panel

The Dockable Panel is a visual element that can be controlled by the user, it can be moved wherever they like and has a docking property which will auto fill to a specific portion of the Revit application view.



The panel has a GUID and its own visual designer file, it is essentially a persistent window with built in functionality to make it more flexible than standard WinForms. To add text, you simply need to create an event handling function or directly append text to the pane through a reference to its respective instance.

```
Autodesk.Revit.UI.TaskDialog.Show("Debug", "Select Button Click");

CreateTab.MyPanelInstance.AppendText("=====" + "Pipe Filter(Select) - " + services[0].ToString() + "=====");

// Check if sorting pipes; get pipes
if ((pipeselected.Count) > 0)
{
    pipeelements = PipeFilter_Helper.GetPipes(collector, mats, sizes, redsizes, refs, services,
                                        IsRefButton, NotRefButton,
                                        IsServButton, NotServButton,
                                        IsMatButton, NotMatButton,
                                        IsSizeButton, NotSizeButton);

    CreateTab.MyPanelInstance.AppendText("Pipe count: " + pipeelements.Count);

    auxilliaryFunctions.PopulatePanel(pipeelements, Doc,
                                    uiapp, dockablePaneId,
                                    "Pipe",
                                    "Name", BuiltInParameter.FABRICATION_PRODUCT_DATA_MATERIAL_DESCRIPTION,
                                    "Service", BuiltInParameter.FABRICATION_SERVICE_NAME,
                                    "Size", BuiltInParameter.FABRICATION_PRI_SIZE);
```

The pane is initialized by default on startup of CreateTab, so it can be written to at any point in the whole VS Solution.

There is a clear pane button connected to a backend function that reinitializes the pane to wipe away its current state, and a listener function will activate the Rescan button when the user writes LengthCheck data to the pane. Visibility for these buttons is the same for a WinForm, you can set the .Visible attribute to True or False to show or hide at any time.

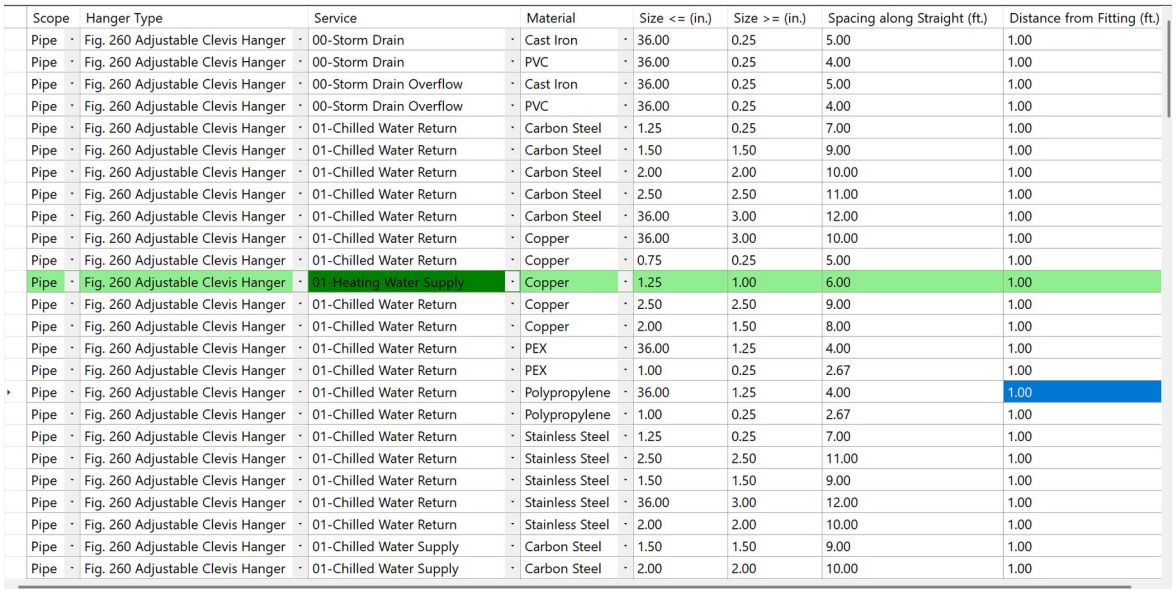
PlaceHangers and HangerSpecs

By moving most operations inside of transactions and cutting down on some inefficient code, the reloading/flashing effect that made the program feel ‘clunky’ can be mitigated.

On form close, a singular sliding window style comparison logic loop is used to detect multiple different logical conflicts within the table.

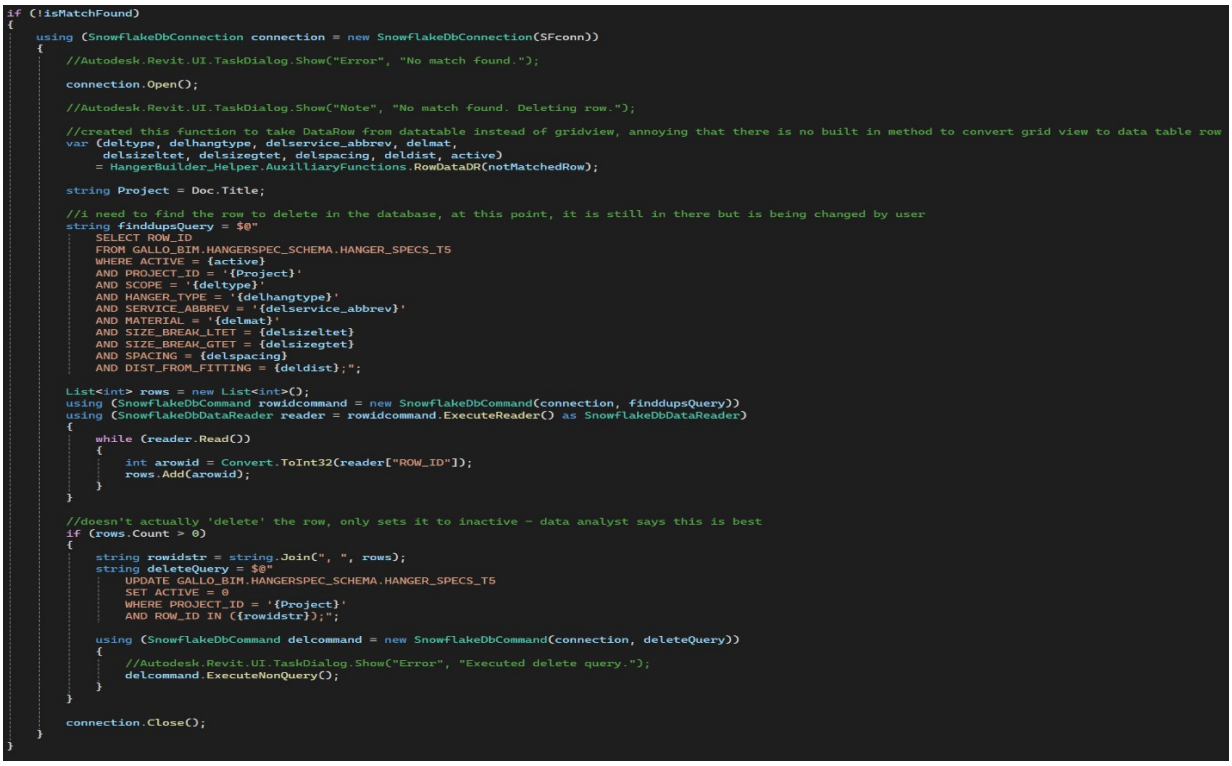
```
private void HangerBuilder_FormClosing(object sender, FormClosingEventArgs e)
{
    GridView1.EndEdit();
    GridView1.CommitEdit(DataGridViewDataErrorContexts.Commit);
    GridView1.CurrentCell = null;

    bool edit = false;    // TODO: implement actual edit detection
    bool formatError = false;
    bool duplicateError = false;
    bool rangeOverlap = false;

    //iterate over the final state of the grid view to determine illegal conditions
    for (int i = 0; i < GridView1.Rows.Count; i++)
    {
        var row1 = GridView1.Rows[i];
        if (row1.IsNewRow) continue;

        string serviceKey1 = $"{row1.Cells[2].Value}{row1.Cells[3].Value}";

        //parse ltet1 / gtet1 and see if they are double format
        if (!double.TryParse(Convert.ToString(row1.Cells["SIZE_BREAK_LTET"].Value), out double ltet1) ||
            !double.TryParse(Convert.ToString(row1.Cells["SIZE_BREAK_GTET"].Value), out double gtet1))
        {
            row1.DefaultCellStyle.BackColor = System.Drawing.Color.LightGoldenrodYellow;
            formatError = true;
            continue;
        }

        //if the size range is inverted or slots are excessively small value
        if (ltet1 < gtet1 || Math.Abs(ltet1 + gtet1) < 0.01)
        {
            row1.DefaultCellStyle.BackColor = System.Drawing.Color.LightGoldenrodYellow;
            formatError = true;
        }

        //compare to every other value to find duplicates or overlapping ranges
        for (int j = i + 1; j < GridView1.Rows.Count; j++)
        {
            var row2 = GridView1.Rows[j];
            if (row2.IsNewRow) continue;

            if (!double.TryParse(Convert.ToString(row2.Cells["SIZE_BREAK_LTET"].Value), out double ltet2) ||
                !double.TryParse(Convert.ToString(row2.Cells["SIZE_BREAK_GTET"].Value), out double gtet2))
            {
                continue; // skip row with invalid numbers
            }

            string serviceKey2 = $"{row2.Cells[2].Value}{row2.Cells[3].Value}";

            //check if rows are identical or logical overlaps in ltet/gtet for same service rows
            if (AreRowsIdentical(row1, row2))
            {
                row1.DefaultCellStyle.BackColor = System.Drawing.Color.LightGoldenrodYellow;
                row2.DefaultCellStyle.BackColor = System.Drawing.Color.LightGoldenrodYellow;
                duplicateError = true;
            }
            else if (serviceKey1 == serviceKey2 && IsRangeInside(gtet1, ltet1, gtet2, ltet2))
            {
                row1.DefaultCellStyle.BackColor = System.Drawing.Color.LightGoldenrodYellow;
                row2.DefaultCellStyle.BackColor = System.Drawing.Color.LightGoldenrodYellow;
                rangeOverlap = true;
            }

            //if green, raise edit flag
            else if (row2.DefaultCellStyle.BackColor == System.Drawing.Color.LightGreen)
            {
                edit = true;
            }
        }
    }
```

Additionally, there is no longer an edit button, the user simply changes whatever they want, and the cell/row will become green and force them to confirm their edit. The color nature of the row is evaluated on close, preventing duplicates within the database.



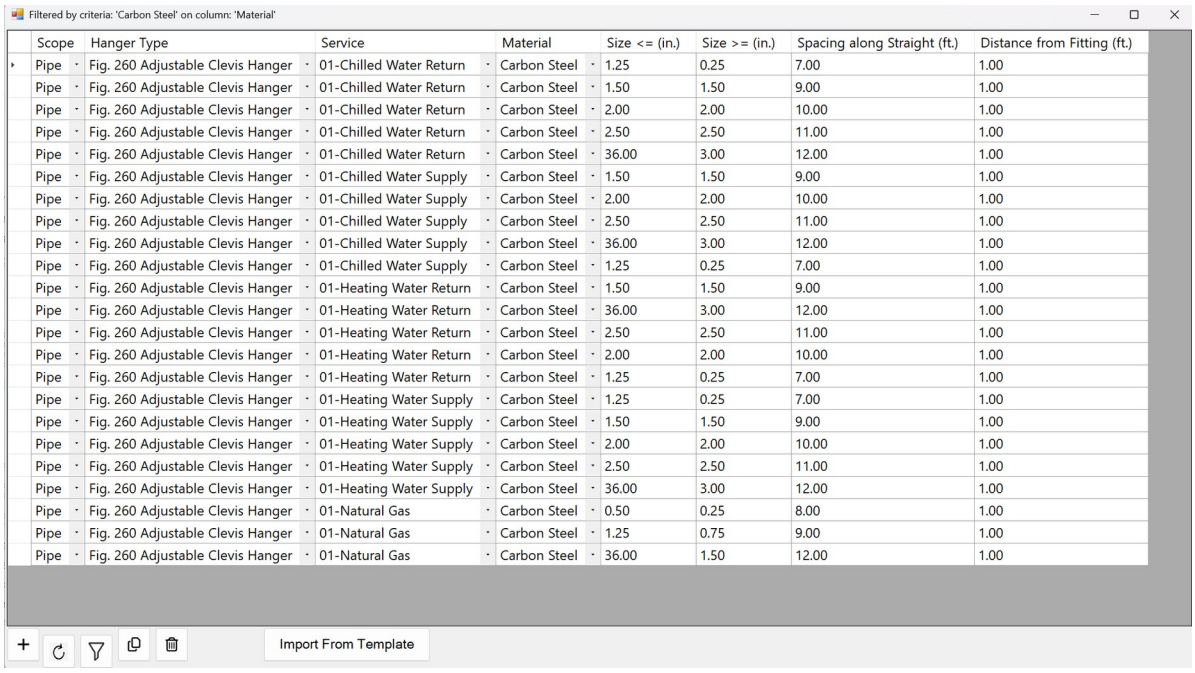
Scope	Hanger Type	Service	Material	Size <= (in.)	Size >= (in.)	Spacing along Straight (ft.)	Distance from Fitting (ft.)
Pipe	Fig. 260 Adjustable Clevis Hanger	00-Storm Drain	Cast Iron	36.00	0.25	5.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	00-Storm Drain	PVC	36.00	0.25	4.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	00-Storm Drain Overflow	Cast Iron	36.00	0.25	5.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	00-Storm Drain Overflow	PVC	36.00	0.25	4.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	Carbon Steel	1.25	0.25	7.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	Carbon Steel	1.50	1.50	9.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	Carbon Steel	2.00	2.00	10.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	Carbon Steel	2.50	2.50	11.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	Carbon Steel	36.00	3.00	12.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	Copper	36.00	3.00	10.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	Copper	0.75	0.25	5.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Heating Water Supply	Copper	1.25	1.00	6.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	Copper	2.50	2.50	9.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	Copper	2.00	1.50	8.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	PEX	36.00	1.25	4.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	PEX	1.00	0.25	2.67	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	Polypropylene	36.00	1.25	4.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	Polypropylene	1.00	0.25	2.67	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	Stainless Steel	1.25	0.25	7.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	Stainless Steel	2.50	2.50	11.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	Stainless Steel	1.50	1.50	9.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	Stainless Steel	36.00	3.00	12.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Return	Stainless Steel	2.00	2.00	10.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Supply	Carbon Steel	1.50	1.50	9.00	1.00
Pipe	Fig. 260 Adjustable Clevis Hanger	01-Chilled Water Supply	Carbon Steel	2.00	2.00	10.00	1.00

This required some specific logic to effectively handle the new dynamic changing nature that comes with no longer having a discreet edit button. Essentially, we need to find the newly changed row in the database and nullify it, this is done on confirmation to enable the user to still ‘cancel’ any green row changes.



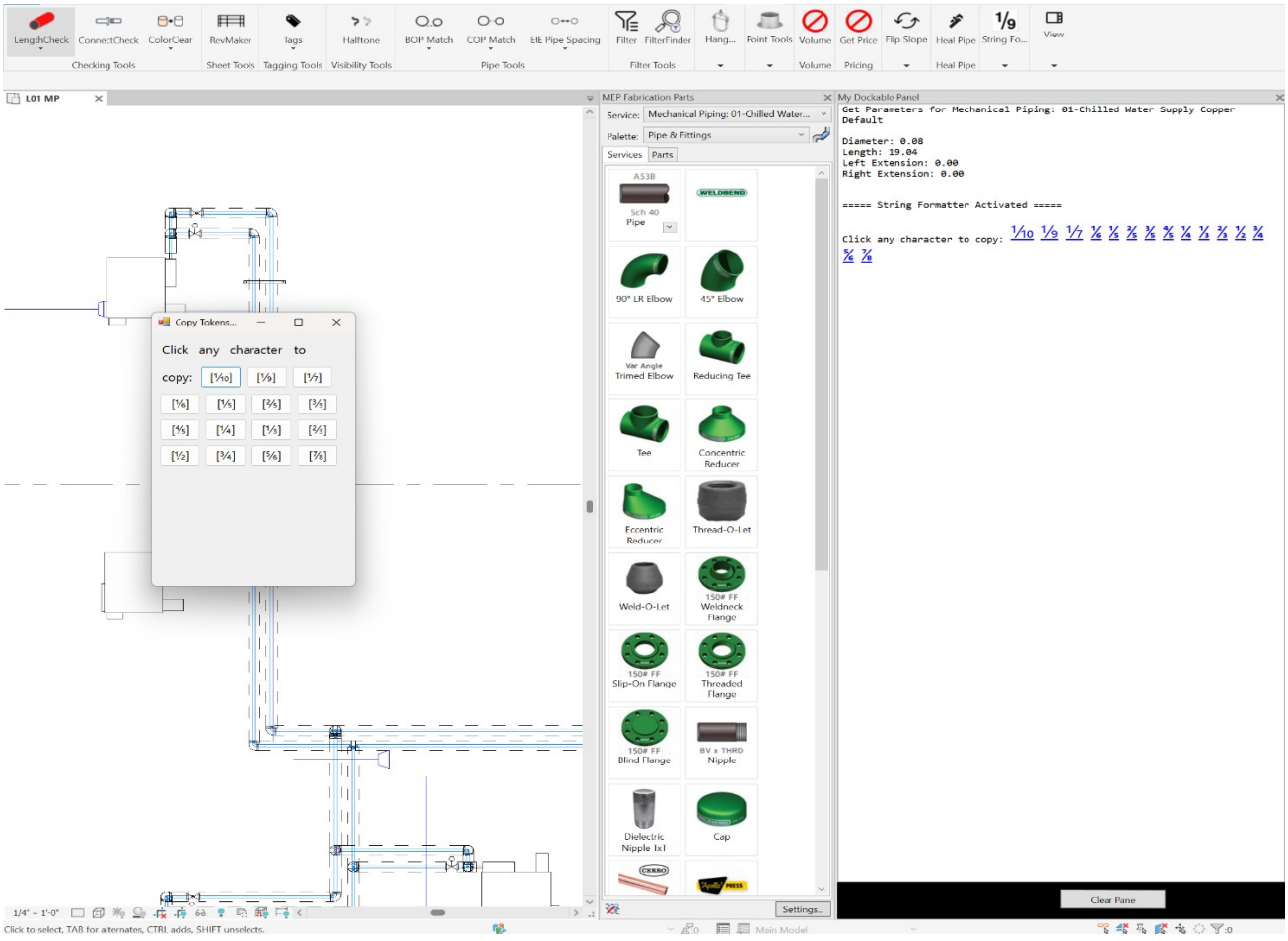
```
if (!isMatchFound)
{
    using (SnowflakeDbConnection connection = new SnowflakeDbConnection(SFconn))
    {
        //Autodesk.Revit.UI.TaskDialog.Show("Error", "No match found.");

        connection.Open();

        //Autodesk.Revit.UI.TaskDialog.Show("Note", "No match found. Deleting row.");

        //created this function to take DataRow from datatable instead of gridview, annoying that there is no built in method to convert grid view to data table row
        var (deltype, delhangtype, delservice_abbrev, delmat,
             delsizeltet, delsizegtet, delspacing, deldist, active)
            = HangerBuilder_Helper.AuxilliaryFunctions.RowDataDR(notMatchedRow);

        string Project = Doc.Title;

        //i need to find the row to delete in the database, at this point, it is still in there but is being changed by user
        string finddupsQuery = $@"
            SELECT ROW_ID
            FROM GALLO_BIM.HANGERSPEC_SCHEMA.HANGER_SPECS_T5
            WHERE ACTIVE = {active}
            AND PROJECT_ID = '{Project}'
            AND SCOPE = '{deltype}'
            AND HANGER_TYPE = '{delhangtype}'
            AND SERVICE_ABBREV = '{delservice_abbrev}'
            AND MATERIAL = '{delmat}'
            AND SIZE_BREAK_LTET = {delsizeltet}
            AND SIZE_BREAK_GTET = {delsizegtet}
            AND SPACING = {delspacing}
            AND DIST_FROM_FITTING = {deldist};";

        List<int> rows = new List<int>();
        using (SnowflakeDbCommand rowidcommand = new SnowflakeDbCommand(connection, finddupsQuery))
        using (SnowflakeDbDataReader reader = rowidcommand.ExecuteReader() as SnowflakeDbDataReader)
        {
            while (reader.Read())
            {
                int arowid = Convert.ToInt32(reader["ROW_ID"]);
                rows.Add(arowid);
            }
        }

        //doesn't actually 'delete' the row, only sets it to inactive - data analyst says this is best
        if (rows.Count > 0)
        {
            string rowidstr = string.Join(", ", rows);
            string deleteQuery = $@"
                UPDATE GALLO_BIM.HANGERSPEC_SCHEMA.HANGER_SPECS_T5
                SET ACTIVE = 0
                WHERE PROJECT_ID = '{Project}'
                AND ROW_ID IN ({rowidstr});";

            using (SnowflakeDbCommand delcommand = new SnowflakeDbCommand(connection, deleteQuery))
            {
                //Autodesk.Revit.UI.TaskDialog.Show("Error", "Executed delete query.");
                delcommand.ExecuteNonQuery();
            }
        }

        connection.Close();
    }
}
```

The Filter function simply prompts the user to select a value from any given cell in the table, it will then get all other rows that have that same value in that same column. It is worth noting that this can be done ‘cumulatively’ in a way that continuously narrows down by criteria. Essentially, a user can further filter the result of a filter operation across several columns of selected criteria by using the tool repetitively. The most recent filter operation will show in the top header of the resulting table.

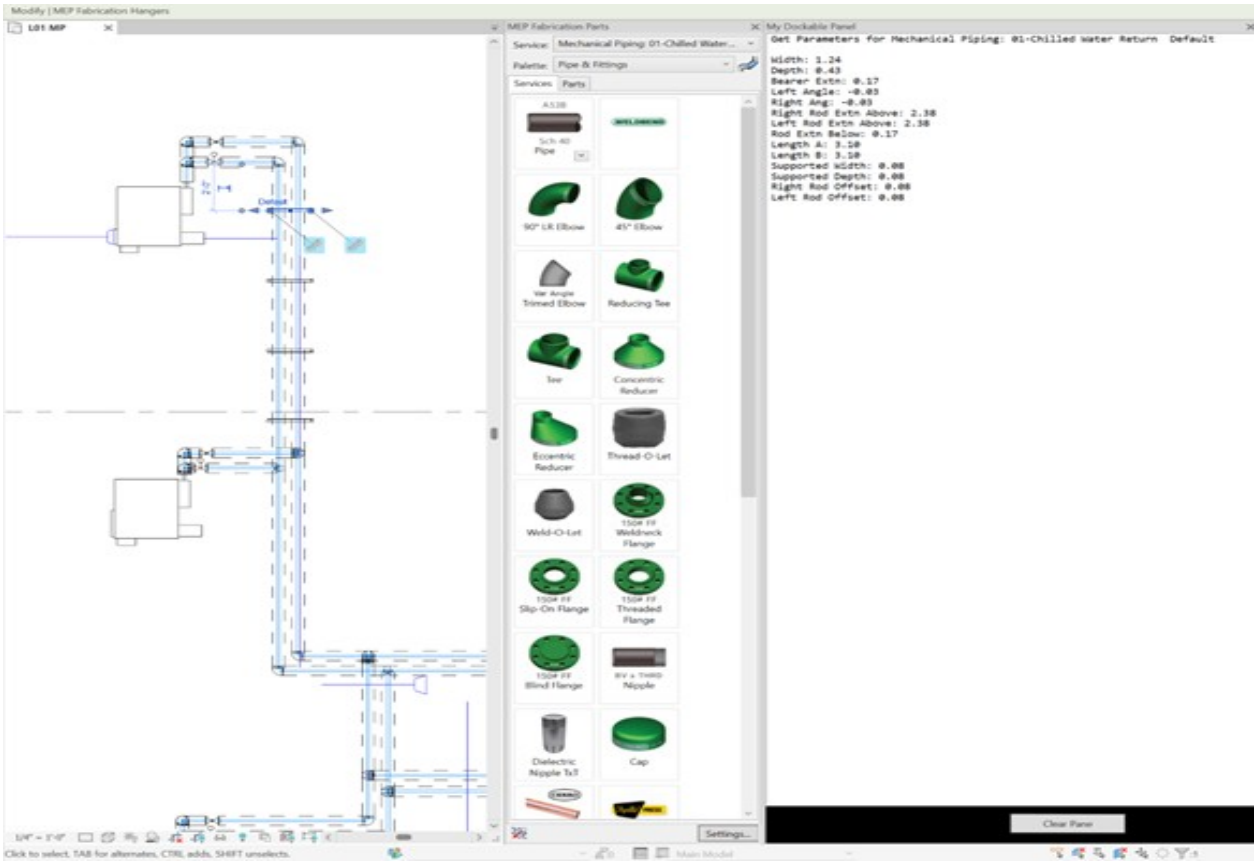

Filtered by criteria: 'Carbon Steel' on column: 'Material'
Scope
Hanger Type
Service
Material
Size <= (in.)
Size >= (in.)
Spacing along Straight (ft.)
Distance from Fitting (ft.)
Pipe Fig. 260 Adjustable Clevis Hanger 01-Chilled Water Return Carbon Steel 1.25 0.25 7.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Chilled Water Return Carbon Steel 1.50 1.50 9.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Chilled Water Return Carbon Steel 2.00 2.00 10.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Chilled Water Return Carbon Steel 2.50 2.50 11.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Chilled Water Return Carbon Steel 36.00 3.00 12.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Chilled Water Supply Carbon Steel 1.50 1.50 9.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Chilled Water Supply Carbon Steel 2.00 2.00 10.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Chilled Water Supply Carbon Steel 2.50 2.50 11.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Chilled Water Supply Carbon Steel 36.00 3.00 12.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Chilled Water Supply Carbon Steel 1.25 0.25 7.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Heating Water Return Carbon Steel 1.50 1.50 9.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Heating Water Return Carbon Steel 36.00 3.00 12.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Heating Water Return Carbon Steel 2.50 2.50 11.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Heating Water Return Carbon Steel 2.00 2.00 10.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Heating Water Return Carbon Steel 1.25 0.25 7.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Heating Water Supply Carbon Steel 1.25 0.25 7.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Heating Water Supply Carbon Steel 1.50 1.50 9.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Heating Water Supply Carbon Steel 2.00 2.00 10.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Heating Water Supply Carbon Steel 2.50 2.50 11.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Heating Water Supply Carbon Steel 36.00 3.00 12.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Natural Gas Carbon Steel 0.50 0.25 8.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Natural Gas Carbon Steel 1.25 0.75 9.00 1.00
Pipe Fig. 260 Adjustable Clevis Hanger 01-Natural Gas Carbon Steel 36.00 1.50 12.00 1.00
Import From Template

Additional Functions

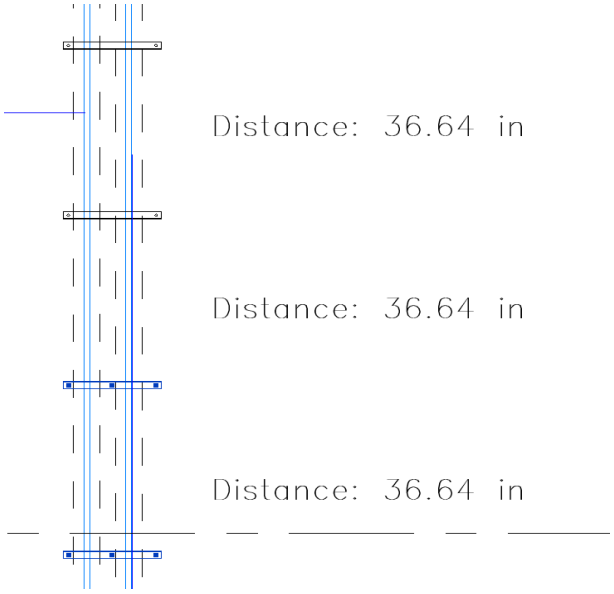
String Format – This tool displays a series of special characters which represent fractions. They are unique characters that cannot be typed. Before this, users would use third party tools on the internet to get more compact fractions for their text boxes. Now these special characters will show in a pop-up menu and populate to the panel for re-reference.



Get Parameters – This will enter the user to a persistent background mode that will display a selected elements FabricationDimension definitions. This applies to anything that is a FabricationPart (pipes, hangers, most fittings, etc.) This runs persistently in the background, it does not interfere with any other processes.



Find Distance – Allows the user to select a pair of elements and displays a text box with the distance between them. Updates dynamically with movement of either object to reflect current distance.



Alter Hanger Host (Remove/Change) – Attaches/removes a hanger to/from a selected host, worth noting this may alter alignment of the hanger upon re-attachment.

```
XYZ translationVector = hostRef.GlobalPoint.Subtract(hangerRef.GlobalPoint);
trans.Start();
try
{
    // For whatever reason, the tool only functions proerly on hangers that have undergone some form of edit.
    // This is an 'automatic jiggling' of the hanger to ensure that even hangers unedited function properly with the tool.
    // This does have the unintended effect of making the tool ONLY work for Bearer Hangers.
    double originalBearerExt = selectedHangerFab.GetRodInfo().GetBearerExtension(0);
    selectedHangerFab.GetRodInfo().SetBearerExtension(0, originalBearerExt + 0.1);
    selectedHangerFab.GetRodInfo().SetBearerExtension(0, originalBearerExt);

    selectedHangerHost.DisconnectFromHost();
    ElementTransformUtils.MoveElement(doc, selectedHangerId, translationVector);

    foreach (Connector c in hostPipeFab.ConnectorManager.Connectors)
    {
        selectedHangerHost.PlaceOnHost(selectedhostPipeId, c, 0);
    }
    trans.Commit();
}
```

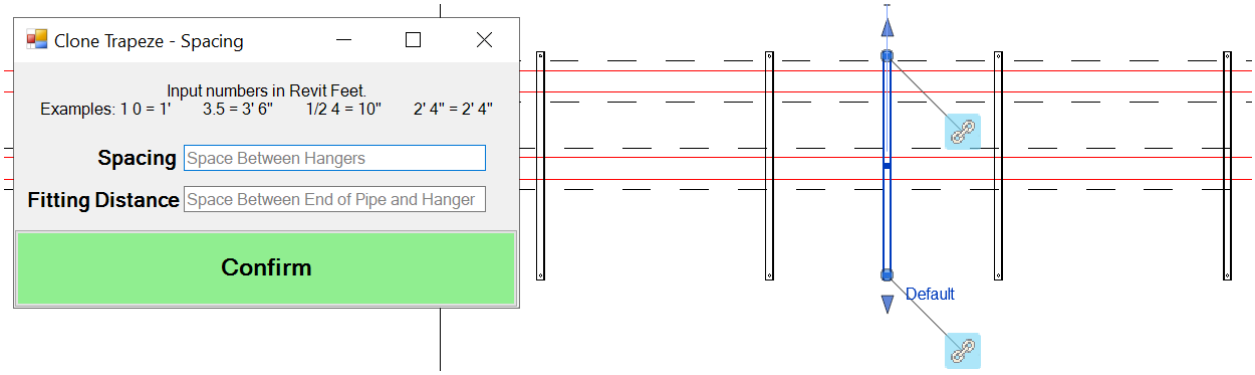
Code snippet from ChangeHost function. When a trapeze hanger is first created, it throws errors when attempting to attach / remove to host unless it has first been

altered. To address this, a ‘jiggle’ is added to the bearer prior to host disconnection attempt. This also means that these functions only work on bearer hangers.

Trapeze Hangers

Span Trapeze - Automatically spaces a selected hanger between two pipes. The hanger will not rotate and will be automatically placed between the two pipes. The automatic placement is done in order to evenly space the hanger between the two pipes in one direction.

Clone Trapeze – Allows a trapeze to be copied across the entire length of its host pipe. Spacing and Fitting can be specified in a relatively robust manner including ‘Revit Feet.’ Examples are included in the tool. The original cloned hanger will **not** be deleted. Notice, in the below picture, how the selected hanger is improperly spaced.



Code below shows a snippet of Clone Trapeze that causes the tool to check if a user already has a hanger selected when they select the tool. If so, the tool immediately activates. Otherwise, it prompts the user to select a hanger.

```
// If the user has a hanegr selected, use it for the tool. Otherwise, prompt user to select a hanger
ICollection<ElementId> selectedIds = uidoc.Selection.GetElementIds();
if(selectedIds.Count == 1 && doc.GetElement(selectedIds.First()).Category.Id.IntegerValue == (int)BuiltInCategory.OST_FabricationHangers)
{
    selectedHangerId = selectedIds.First();
}
else
{
    Autodesk.Revit.DB.Reference hangerRef = uidoc.Selection.PickObject(ObjectType.Element, new hangfilter(), "Select hanger to duplicate.");
    selectedHangerId = hangerRef.ElementId;
}
selectedHangerFab = doc.GetElement(selectedHangerId) as FabricationPart;

// Get associated pipe from hanger
ElementId selectedhostHangerId = selectedHangerFab.GetHostedInfo().HostId;
if (selectedhostHangerId.IntegerValue == -1)// Check if hanger's parent is the view, in which case it isn't attached to a Pipe.
{
    TaskDialog.Show("Error", "Trapeze Hanger must be attached to a pipe to duplicate with this tool.");
    return Result.Failed;
}
```

One may wish to apply this code snippet to other tools to prevent users from needing to remember if they need to select their element before or after the tool. It, however, could cause unintended selections if the user selects the tool and didn’t realize they had an applicable item selected.

Future Ideas

Visuals within the Panel - I would like to create a mode where a given part can be 3D viewed in an isolated mode within the panel. The user can alter this element and see meta-data about it like what is produced in GetParameters. I introduced the panel to act as a repository for data and a view that we can explicitly control from the API. Any sort of visual or metric you want to visualize can be displayed.

Parallel Programing – The use of parallel programming could enable more effective processing of large-scale operations like the filter tool or length checking. This would introduce a lot of overhead, having to handle a threading library, but would potentially boost performance noticeably.