pCOLAD-for-Dynamo development

part 3

Contents

[Introduction 3](#_Toc448486781)

[CSV control always on top of Dynamo 3](#_Toc448486782)

[Cancel button and top right X button 4](#_Toc448486783)

[Automatic disappearing message 4](#_Toc448486784)

[Different functions don’t work anymore 5](#_Toc448486785)

[Create a local repository for version control 6](#_Toc448486786)

[Time to make a flowchart (should have done that at the start!). 7](#_Toc448486787)

[Reminder for working with the code after some time. 19](#_Toc448486788)

Figures

[Figure 1 Different routing for Cancel and Close button 4](#_Toc448486789)

[Figure 2 Global flow chart of pCOLAD-for-Dynamo 7](#_Toc448486790)

[Figure 3 Scenario changing displays 8](#_Toc448486791)

[Figure 4 Version x of flow chart 10](#_Toc448486792)

[Figure 5 Dutch Dynamo User Group meeting 5-4-2016 12](#_Toc448486793)

[Figure 6 Interesting subjects for Dutch Dynamo User Group 12](#_Toc448486794)

[Figure 7 Selected subjects 13](#_Toc448486795)

[Figure 8 Flow chart y 14](#_Toc448486796)

[Figure 9 Code Map of pSHARE 18](#_Toc448486797)

[Figure 10 Code Map related to update of Obstruction field background. 18](#_Toc448486798)

# Introduction

In 2013 Hans Hubers initiated a research called pCOLAD together with Michela Turrin, Irem Erbas and Ioannis Chatzikonstantinou. It was aimed at developing a method, prototype and case study for parametric COLaborative Architectural Design. pCOLAD was well received and presented at the eCAADe2014 conference in NewCastle. It used Grasshopper and VB.net as software. In 2013 this software was the best choice for a parametric solution. However in the course of 2014 a generative parametric solution became useful for Revit, called Dynamo. Since Revit is the most used BIM software, it seems adequate to investigate if the results of the pCOLAD project can be converted to Revit/Dynamo and if a way can be found to use the VB.net prototype in that environment or if adaption is needed or even a new prototype. A report was written about the first approach.

This third report is about the progress in 2016.

# CSV control always on top of Dynamo

Looking for a solution of the fileSystemWatcher I tried solving also the issue of having the CSV control always on top of Dynamo. I posed the question on [GitHub](https://github.com/DynamoDS/Dynamo/issues/6231), but solved it myself:

The answer to this question might be interesting for future amateur developers like me. The trick is that you make the DynamoView the owner of the xaml window control. You can do this in theNodeViewCustomization class of your custom UI node like in the Dynamo Samples. E.g. imagine that you make a custom UI node called YourNode. And with that node you create an instance of a xaml window called myXamlWindow. Then give 'YourNode' a property: public static DynamoView dv; Then in theCustomizeView method put next code:

public void CustomizeView(YourNode model, NodeView nodeView)

{

var YourNodeControl = new YourNodecontrol();

NodeView nv = nodeView;

YourNode.dv = FindUpVisualTree<DynamoView>(nv);

nodeView.inputGrid.Children.Add(YourNodeControl);

YourNodeControl.DataContext = model;

}

private static T FindUpVisualTree<T>(DependencyObject initial) where T : DependencyObject

{

DependencyObject current = initial;

while (current != null && current.GetType() != typeof(T))

{

current = VisualTreeHelper.GetParent(current);

}

return current as T;

}

And in your myXamlWindow.cs file:

public partial class myXamlWindow: Window

{

public myXamlWindow()

{

InitializeComponent();

Owner = YourNode.dv;

}

}

The problem of leaving controls behind when closing Dynamo, is now also automatically solved.

# Cancel button and top right X button

When closing the history display or csv display with the red cross top-right the pSHARE button stays green. First approach was to catch the closing event and then simply bind to the CancelCommand. But since within that routine also the \_CSVcontrol.Close() method was used, you get an error, because with the red cross button top right you already started the closing. The trick was to understand that although closing ends at the same closing method, the routing is different (Figure 1). When you hit the red cross button top right you first trigger the closing event (and a method in the code behind (CSVControl.xaml.cs) if you attached it). When you hit the Cancel button you first go to the CancelCommand method in pSHARE.cs and then to the method in the code behind. So if you set a property there (pSHARE.Canceling) you can detect that in this Window\_Closing method, and do nothing. The other way around you set a property of the CSVControl (ClosingStarted) to detect in the ShowParams method of pSHARE.cs and skop the closing in that case.

**X**

Cancel

Figure 1 Different routing for Cancel and Close button

So the solution was: adding next line to the definition of the window (or click this event in the properties window under eventhandlers tab):

Closing="Window\_Closing"

And in the pSHARE.cs in the ShowParams method:

if (!\_CSVControl.ClosingStarted)

{

\_CSVControl.Canceling = true;

closeCSVControl();

}

And in the CSVControl.xaml.cs:

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

if (!Canceling)

{

ClosingStarted = true;

// run the CancelCommand by simulating click on Cancel button

ButtonAutomationPeer peer = new ButtonAutomationPeer(Cancel);

IInvokeProvider invokeProv = (IInvokeProvider)peer.GetPattern(PatternInterface.Invoke);

invokeProv.Invoke();

}

}

# Automatic disappearing message

When saving in Dynamo you get two messages yourself. Not anymore. And now also you get an automatic disappearing message that save was successful. For this in the MyDataCollector namespace the TempMessage class was made and a TempMessageXaml window. With Blend for Visual Studio 2015 two storyboards were made (need to hit the story board tab for that) and a trigger. This resulted in next code:

<Window

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:MyDataCollector"

xmlns:i="http://schemas.microsoft.com/expression/2010/interactivity"

xmlns:ei="http://schemas.microsoft.com/expression/2010/interactions" x:Name="window"

x:Class="MyDataCollector.TempMessageXAML"

mc:Ignorable="d"

xmlns:diag="clr-namespace:System.Diagnostics;assembly=WindowsBase"

d:DesignHeight="300" d:DesignWidth="300" SizeToContent="WidthAndHeight" ShowInTaskbar="False"

WindowStyle="None" Visibility="Visible" AllowsTransparency="True" Background="#FF8BCCF9"

Foreground="Black" d:DataContext="{d:DesignInstance {x:Type local:TempMessage}}" WindowStartupLocation="CenterScreen"

>

<Window.Resources>

<Storyboard x:Key="TempShow">

<ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="(UIElement.Visibility)" Storyboard.TargetName="window">

<DiscreteObjectKeyFrame KeyTime="0:0:3" Value="{x:Static Visibility.Hidden}"/>

</ObjectAnimationUsingKeyFrames>

</Storyboard>

<Storyboard x:Key="Transparency">

<DoubleAnimationUsingKeyFrames Storyboard.TargetProperty="(UIElement.Opacity)" Storyboard.TargetName="window">

<EasingDoubleKeyFrame KeyTime="0" Value="0"/>

<EasingDoubleKeyFrame KeyTime="0:0:2" Value="1"/>

<EasingDoubleKeyFrame KeyTime="0:0:3" Value="0"/>

</DoubleAnimationUsingKeyFrames>

</Storyboard>

</Window.Resources>

<Window.DataContext>

<local:TempMessage/>

</Window.DataContext>

<Window.Triggers>

<EventTrigger RoutedEvent="FrameworkElement.Loaded">

<BeginStoryboard Storyboard="{StaticResource TempShow}"/>

<BeginStoryboard Storyboard="{StaticResource Transparency}"/>

</EventTrigger>

</Window.Triggers>

<!--<Window.Effect>

<BlurEffect/>

</Window.Effect>-->

<TextBox Text="{Binding MessageString}" SelectionBrush="Black" HorizontalAlignment="Center"

VerticalAlignment="Center" Background="{x:Null}" BorderBrush="{x:Null}"

diag:PresentationTraceSources.TraceLevel="High" Margin="50" />

<!--<TextBox Text="Effe checke..." SelectionBrush="Black" HorizontalAlignment="Center" VerticalAlignment="Center" Background="{x:Null}" BorderBrush="{x:Null}" />-->

</Window>

# Different functions don’t work anymore

Old Value doesn’t work anymore and still several messages when you save the csv.file.

Also if you don’t change something and hit Share button you don’t get the message that nothing changed. Earlier check showed that the date of saving is different… Shouldn’t be in csv file? Only in History file?

Or maybe Old Value never was updated? Getting confused… When you share your parameters in Grasshopper, you write the data of the form to the inputFile and copy the inputFile to the inputFileCopy. So at that time the new and old csv file are the same. When you load the csv file and somebody changed it, comparing with the old csv file shows the differences in red, but does not change the Old Values.

The way it should be: When you load the shared csv file (inputFile), you compare the values with the local copy (inputFileCopy). If a New Value is different, then you set the Old Value to the New Value of the local copy. Any difference shows in red.

Now if you hit Run the changes by somebody else lose their red colour. And in pCOLAD-for-Grasshopper changes are not in red anymore.

When you share your data the shared csv file is written with the new data of course, and so is the local copy. If somebody else changes the shared csv file the local copy is not affected, unless you work on the same machine of course (like me when testing!). For this it would be wise to make different local copies for pCOLAD for Dynamo and Grasshopper.

Now when you share from Grasshopper, you get @#$%! in Old Value of external parameters. And images of Test1 don’t show-up in Grasshopper. Well they do if you run the third time; the first time you get error about index not valid value or something; the second time you get everything but the images.

See what happens when you start with empty csv files. In Grasshopper you get your own parameters. In Dynamo also. Then share in Grasshopper (btw an extra file is made next to the Grasshopper local file called: old\_pCOLAD\_for\_Grasshopper\_Local1.csv). You get a message in Dynamo and are forced to start over. In the OnChanged method that is called when file change is detected there is openCSV(), and addNewParameters(), which together reload myDataTable and copyDataTable. So myDataTable now contains the external parameters and the Dynamo parameters. And the copyDataTable is filled with the local csv file which is still empty. But because comparing with an empty copyDataTable is not possible, it gets the standard column names in openCSV(). But also an UpdateCSVControl event is raised and in the handler we find the Compare() call which compares the MyCollectorClass.myDataTable with the copyDataTable that causes the @#$%! Values.

There is also a Compare() call in ShowCSV() method. And in the History() call when you return to csv view.

Strange thing. If you start with empty csv files in Grasshopper and Dynamo; you activate pSHARE in Grasshopper. You get your parameters in display. You open your file in Dynamo. You get your parameters displayed in Dynamo. Hit the share button in Grasshopper. Get your message in Dynamo and are forced to start over. You get the parameters of Grasshopper and Dynamo displayed. Nothing red. Change a value in Grasshopper and share. You get message in Dynamo and are forced to start over. Hit the Run button and the Red button. You get again all correct parameters but nothing in red.

Change a value in Dynamo and hit Run, New Value is updated but nothing in red. Hit the share button. Get a message in Grasshopper. Start over again and you get the warning:

2.We have a problem. Original error: InvalidArgument=Value of '1' is not valid for 'index'. Parameter name: index Please start again... You do that and you get the right parameters, but no Old Value and nothing in red. The error is related to oldLines being empty. That list should be filled with lines from oldFile. And oldFile = localFile.Remove(localFile.LastIndexOf("\") + 1) & "old\_" & localFileName

I don’t understand why this file was created. Answer: because you want to access the DropBox file as little as possible!

# Create a local repository for version control

Now it becomes important to have version control on pCOLAD10.sln visual studio solution (Grasshopper). I would prefer to do that on the local machine. [Here](https://msdn.microsoft.com/en-us/library/hh850445.aspx#local_sln_add) it shows how to do that:

Of course we don’t move the solution to the recommended C:\Users\etc. Right click your solution in Solution Explorer and choose Add Solution to Source Control and choose Git. Now you can commit changes and create branches in Team Explorer. Of course you can not Sync or Publisch etc. because you don’t have a server. Now in the Team double click the branch you want to work with. You can create new branches by right clicking a branche and choose new Local Branch From … For adding a branch to the published pCOLAD\_for\_Dynamo don’t forget to right click and choose Publish.

In fact it would be better if you copy the input file to a local file!!!

And copy that file to an old\_local file, like in pCOLAD for Grasshopper

Let's do that in a new branch 0.2

What about the History.csv file? Will that be in the dropBox? Yes automatically created in the directory of the inputFile.

So, ShareInputFile should become the local file and inpuFileCopy a copy of that.

At what moment do you copy the ShareInputFile to the local file? Not every time you run OpenCSV()

because that is in the Share method, in the History method, in pSHAREinputs method, and in the OnChanged method.

pSHAREinputs method comes first, but runs every time you hit the run button.

Logically you should copy to the local file the first time you run pSHAREinputs method,

when somebody else changes the shared file ~~and when you change it yourself~~ (not in pDynamo because you keep the datatable - in Grasshopper you only have the form that holds the data (true?).

It is confusing because Grasshopper runs automatically and Dynamo doesn’t. Maybe have to change that? Anyway, the confusion now in pDynamo is when to write to localFile and when to oldLocalFile? You only load and copy to localFile when you start a session in Dynamo with a pSHARE (firstTime condition), and when somebody else changes that file.

# Time to make a flowchart (should have done that at the start!).

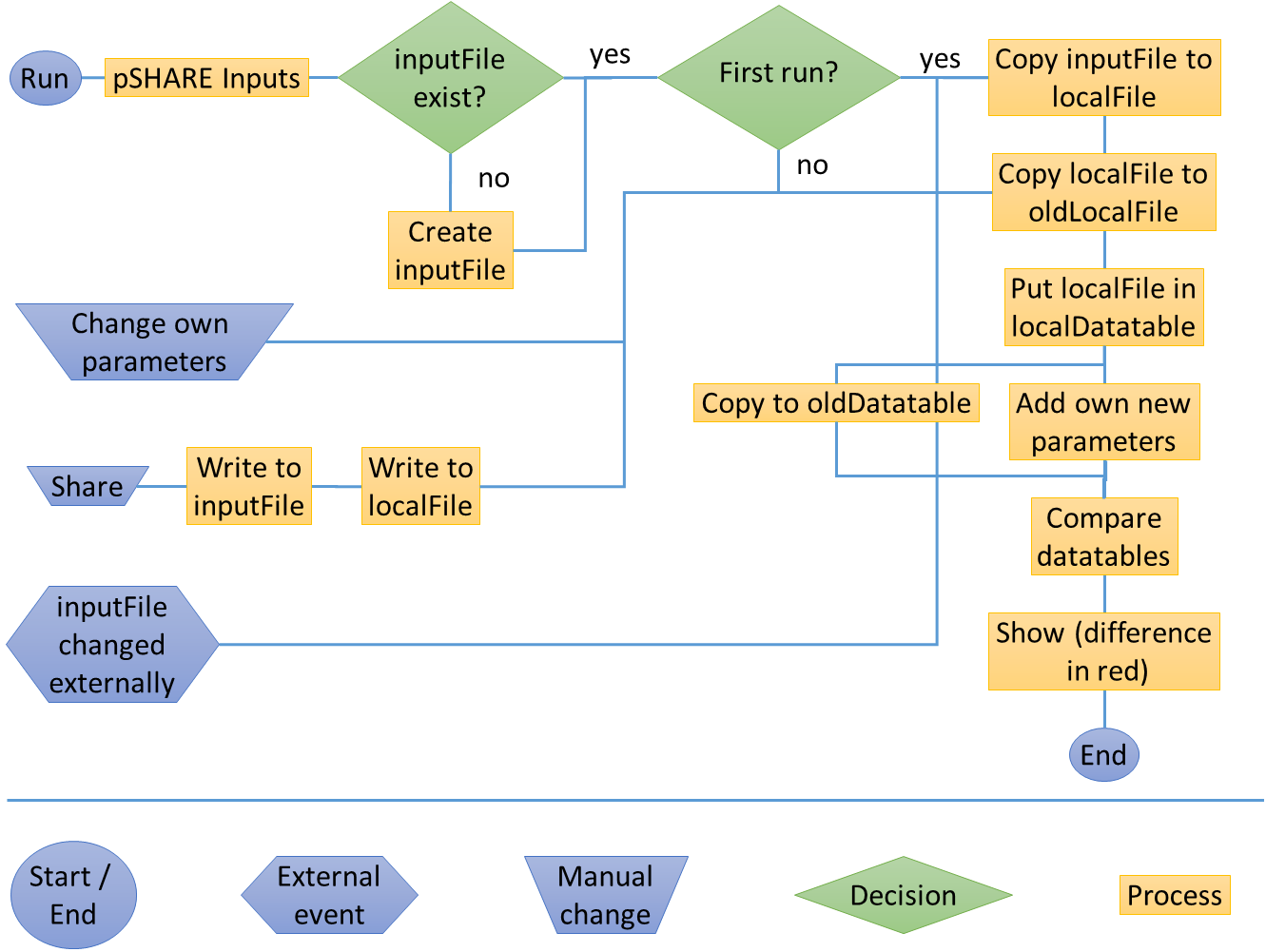


Figure 2 Global flow chart of pCOLAD-for-Dynamo

But how can you get a message that shared file changed since you last worked in Dynamo? So oldLocalFile should only be copied from localFile if it doesn’t exist. And otherwise it should be written too when you share. But when do you write the New Value to the Old Value? Well before you write to oldLocalFile of course. But when do you write to share and local file then?

Turns out that checking if files exist, was put in a function: CSVtoList(string filename) in the Catch error part. That seems practical but is confusing. Better make it a separate function.

Several times you get an error about accessing the Dynamo view from a wrong thread. The solution is to always use a dispatcher:

dv.Dispatcher.Invoke(DispatcherPriority.Normal, new Action(() =>

{

//put your code using dv here

}));

"@#$%!" in Old Value now. Still being confused made a scenario (Figure 3)



Figure 3 Scenario changing displays

But how can UserX in case 3 now that Y proposes deletions? Csv files don’t support strike through. So have to put a sign in front of value too. E.g. ? This can wait, so no deletion at this moment.

Important to realize that in future the columns that are in oldDataTable will not always be in localDataTable and vice versa. So use foreach loops and column names and not indexes in the code. Same goes for primary keys (the parameters, so the rows). So use the primary key values to identify the rows.

First process the cells that are both in local and old. Important to notice that in Figure 3 local is the result of the processing. The way it should be displayed. So during processing initially local contains copy of shared csv file + own parameters. If you compare a cell that doesn’t exist in one or the other data table you get an error. For rows you can find out if the Rows.Find(“key value”) returns null. For columns if the oldDataTable.Column[localDataTable.Column.ColumnName] == null. So with a foreach loop through all rows of localDataTable and If Rows.Find (“key value”) != null and If oldDataTable.Column[localDataTable.Column.ColumnName] != null a foreach sub loop through the columns ensures that you can compare the Items in the cells. Else and Else run SetChanged() on the Item in the localDataTable.

In code:

foreach (DataRow localDataRow in MyDataCollectorClass.localDataTable.Rows)

{

DataRow oldDataRow = MyDataCollectorClass.oldDataTable.Rows.Find(localDataRow["Parameter"]);

if (oldDataRow != null)

{

foreach (DataColumn localDataColumn in MyDataCollectorClass.localDataTable.Columns)

{

//Item in localDataColumn can be null WHY? if there is no oldLocalFile yet??

//then you can not compare it, but won't you get an error when sharing?

//So add an empty Item

MyDataCollector.Item test1 = (localDataRow[localDataColumn.ColumnName] as MyDataCollector.Item);

if (test1 == null)

{

localDataRow[localDataColumn.ColumnName] = new MyDataCollector.Item("");

}

//MyDataCollector.Item test2 = (oldDataRow[localDataColumn.ColumnName] as MyDataCollector.Item);

if (MyDataCollectorClass.oldDataTable.Columns[localDataColumn.ColumnName] != null)

{

if ((localDataRow[localDataColumn.ColumnName] as MyDataCollector.Item).textValue !=

(oldDataRow[localDataColumn.ColumnName] as MyDataCollector.Item).textValue)

{

if ((localDataRow[localDataColumn.ColumnName] as MyDataCollector.Item) != null)

{

(localDataRow[localDataColumn.ColumnName] as MyDataCollector.Item).SetChanged();

}

}

}

else

{

//a column is missing in oldDataTable, but that doesn't matter, you don't have to add it, just

//run SetChanged() on Item in this cell in localDataTable

if ((localDataRow[localDataColumn.ColumnName] as MyDataCollector.Item) == null)

{

localDataRow[localDataColumn.ColumnName] = new MyDataCollector.Item("");

}

(localDataRow[localDataColumn.ColumnName] as MyDataCollector.Item).SetChanged();

}

}

}

else

{

// a row is missing

foreach (DataColumn localDataColumn in MyDataCollectorClass.localDataTable.Columns)

{

if ((localDataRow[localDataColumn.ColumnName] as MyDataCollector.Item) == null)

{

localDataRow[localDataColumn.ColumnName] = new MyDataCollector.Item("");

}

(localDataRow[localDataColumn.ColumnName] as MyDataCollector.Item).SetChanged();

}

}

}

Testing this went wrong all the time. Took me days to find out that I made an error in copying the localCSVfile to oldLocalCSVfile. I assumed that path.GetDirectory(localCSVfile) + “old” + path.GetFileName(localCSVfile) would give the path for the copy. But the directory doesn’t end with \ so “old” had to be “\\old”. I was comparing with the wrong oldDataTable!

Now the thing with writing the Old Value. When and where should that be done? I would assume best when sharing. When you share the data you write the localDataTable to the share file and local file. You should then immediately have no red in the display. But if you continue working and change something that should become red. When you hit Run. Hitting Run restarts everything. So the shared file will be copied to the local file? No you have a condition (firstRun). But no problem, you write to both when sharing. But the old local file will not be touched, because it exists already. So during sharing you can put New Value into Old Value if you don’t write to the local file at that moment. So write to local file before that. And write to old local file after that. Then the old local file is the same as the shared file and contains the Old Value, while the local file does not. WRONG

What if you always copy local file to old local file when you hit Run? Before you add the new parameters. So you write to local file when Sharing. No, there is no way to see changes then when you start.

After trying several versions of the flow chart, it turns out that the code is very differently organized. E.g. the OpenCSV() method is used for both showing the shared file and the (changed) own parameters and the history file and for making the oldDataTable (only if there was no content). This is maybe smart for reusing code, but not smart at all for picking things up later or by somebody else. Reorganising the code, however is very risky, because many things are not in the flow chart. E.g. hitting the history button, or selecting the checkbox in front of a parameter in order to put the user name in the Obstruction field.

But it has to be done… Also names were not clear. E.g. inputFile instead of sharedFile. Another example of too confusing coding is that the display of the parameters is bound to myPropDataTable, which already is not a very meaningful name, but it is filled with: localDataTable (this was necessary because the inputs of a Dynamo node must be collected with a call to a static method of a different static class) , but localDataTable is sometimes filled with data from the local file and the own parameters, but if the History button is hit with the data from the History file (should use a stored dataTable if no changes on the shared file in the meanwhile - in fact it is already because you have condition formPopulated and if false the stored csvDataTable is used - however not clear if that could be the history data?). Changed OpenCSV() to OpenCSV(file) so you can keep track on which file is being used.

Finally after many trials and errors the flow chart in Figure 4 was adopted.

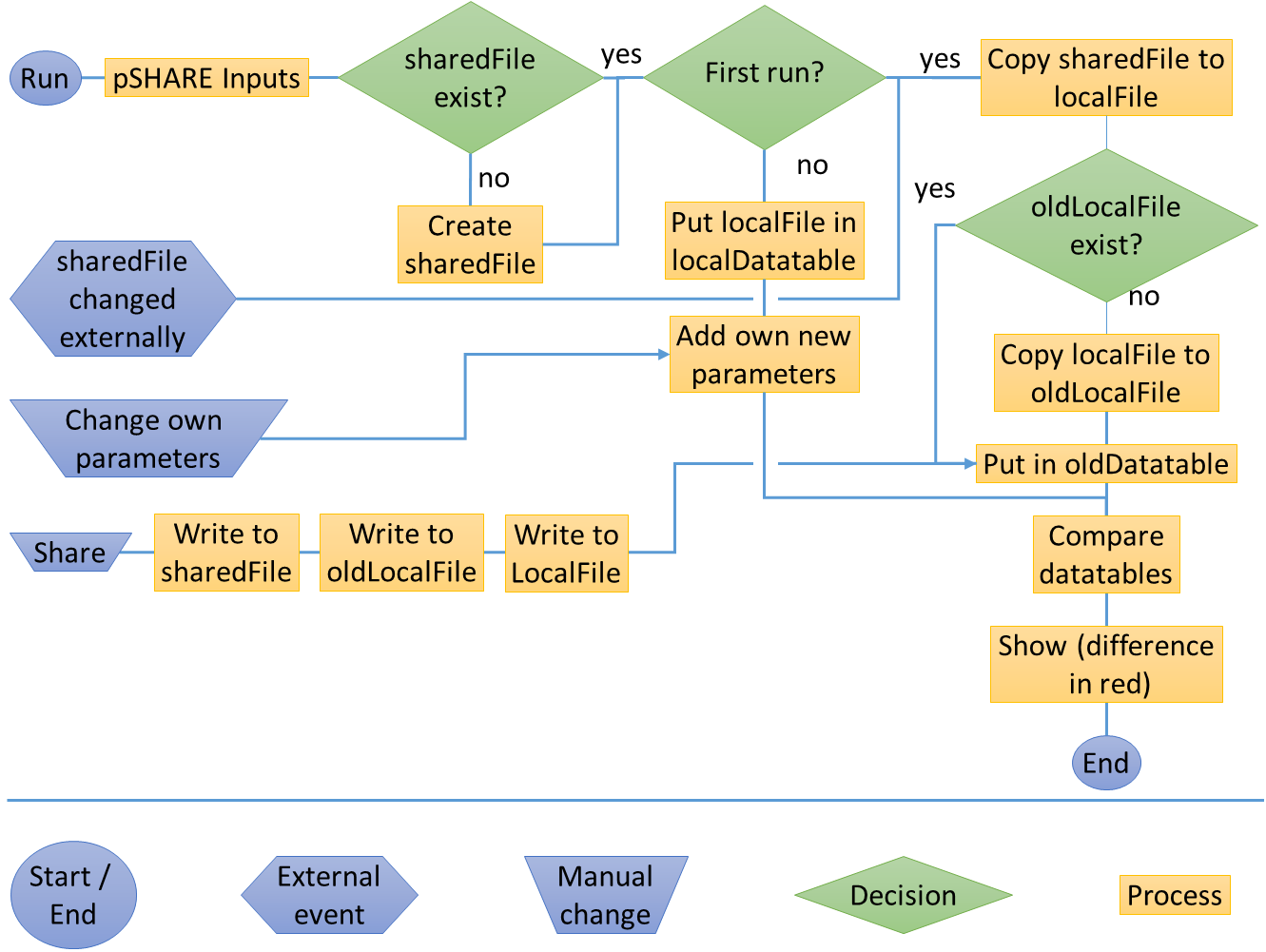


Figure 4 Version x of flow chart

Only it is impossible to get immediate update of the display when you change your own parameters. You have to hit Run (now by code: DynamoModel.ForceRun()). Which causes the next run on the top line in the flow chart. This should not have an effect on the result, but it does: New Value is updated and red, but Old Value is empty and red, also for another own parameter. The date and author are also red. Even if you shared the data yourself before. So somewhere in the code when you hit Run you do something else then in the flow chart. Sigh… check again step by step. Indeed, the “Put in oldDatatable” runs. But that should not be a problem either. Maybe the “Add own new parameters” overwrites the Old Values too? In fact it is not only adding. It is merging of two dataTables. This means that the newParamTable rows are compared to the rows of localDataTable. If some fields are different, the fields of localDataTable are overwriten. If a primary key value is different a new row is added. But debugging shows that at that moment there are no Old Values changed. In fact in newParamTable, which is made with the outputs of pSHARE, there is only: Parameter; New Value; Importance; Comments and Extra Attributes. Of course: Old Value like Obstruction are added through the display and code. So the problem is elsewhere. Since you write Old Value only when you hit the Share button, they don’t change when you only change your parameters and run the solution. So it must be in the Compare() method that something goes wrong. There you first set all items to isChanged = false. But doesn’t that change the items? No probably the thing is that you check if in every field an item exist, and if not you make one with “” as text property, otherwise you get an error when you try to compare their values. So when there is no Old Value item, you put in such an empty item and then comparing of course gives a difference with … No that can’t be it, because you compare then with also an empty item. I don’t know what did it, but the problem is gone. For a prototype that is all that matters. Continue.

When you write in Dynamo New Value to Old Value in shared file, why in GH you get no difference too? GH should compare shared file with GH’s oldLocalFile… Any way a problem of pCOLAD-for-GH. Not urgent now.

Discovered a problem with Obstruction. When you uncheck a row of yourself, you should get a warning to just change the value if you don’t agree with it. When you uncheck a row of somebody else and hit Share button, the first gets also the user name in field Obstruction. It is horrible! When you change one thing, other things don’t work like they should anymore. What happens when you hit a checkbox in front of a row?

If you do a find “Check” in the CSVControl.xaml you get next result:

Find all "check", Find Results 1, Current Document

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(62): <VisualStateGroup x:Name="CheckStates">

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(63): <VisualState x:Name="Checked">

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(74): <VisualState x:Name="Unchecked"/>

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(84): <Trigger Property="IsChecked" Value="true">

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(207): <CheckBox x:Name="myCheckBox" Loaded="myCheckBox\_Loaded"

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(208): IsChecked="{Binding DataContext.isChecked, Mode=TwoWay, RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type DataGrid}}}"/>

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(366): Visibility ="{Binding ElementName=History, Path=IsChecked, Converter={StaticResource BoolToVisibleConverter}}"/>

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(369): Visibility="{Binding ElementName=History, Path=IsChecked, Converter={StaticResource BoolToVisibleConverter}}" HorizontalAlignment="Right"/>

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(370): <Button Style="{StaticResource newTemplate}" x:Name="UnCheckAll" Grid.Column ="0" Content="UnCheckAll" VerticalAlignment="Bottom"

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(371): Margin="55,0,5,0" Command="{Binding UnCheckAllCommand}"

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(372): Visibility="{Binding ElementName=History, Path=IsChecked, Converter={StaticResource BoolToVisibleConverter}}"/>

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(373): <Button Style="{StaticResource newTemplate}" x:Name="CheckAll" Grid.Column ="0" Content="CheckAll" VerticalAlignment="Bottom"

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(374): HorizontalAlignment="Left" Command="{Binding CheckAllCommand}"

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(375): Visibility="{Binding ElementName=History, Path=IsChecked, Converter={StaticResource BoolToVisibleConverter}}"/>

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(380): HorizontalAlignment="Right" IsChecked="{Binding HistoryOn, Mode=TwoWay}">

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(430): <VisualStateGroup x:Name="CheckStates">

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(431): <VisualState x:Name="Checked" >

D:\Data\Research\Dynamo\pCOLAD for Dynamo\CSVControl.xaml(448): <VisualState x:Name="Unchecked"/>

As can be seen in row 208 the IsChecked state is bound two way to DataContext.isChecked. The DataContext is set in the ViewModel to pSHARE. All that must be OK, because the display updates correctly. It is when you hit the Share button that things go wrong. To be sure, let’s check the state of localDataTable and myPropDataTable just after hitting a checkbox. Indeed just before executing the Share(object obj) method both are correct. Stepping through the code by setting breakpoints at crucial lines, everything (data tables and files) is ok until you force Dynamo to Run (DynamoModel.ForceRun()). After OpenCSV(sharedFile) the locaDataTable is still ok. After returning the pSHAREoutputList the code jumps to the get { return \_isChecked; } Why is that? Just before that line you have UpdateCSVControl(null, EventArgs.Empty); So check there. It sets MyPropDataTable to localDataTable and still ok. Then it goes through ObjectToForeGroundConverter.cs (change that into ObjectToBackgroundConvert) and sets all the properties of the items. Then it goes through DataRowToFullPathListConverter.cs and sets the item properties again. Or maybe the other way around: it sets the item properties and goes through converters if necessary. This suggests that the CSVControl.xaml is updating? Makes sense since it’s Grid is bound to MyPropDataTable. Oops, then suddenly from ObjectToBackGroundConverter it jumps to dr["Obstruction"] = new Item(cellContent.Trim()); in the isChecked property setter. And cellContent is “”. Very strange. And MyPropDataTable lost the username in obstruction field. If you hit Run again, things are ok. But you can not DynamoModel.ForceRun() put in twice. In fact it should not be needed to Run after Share… only after you change your own Parameters through pCOLLECTs. Try this in new branche 0.3. That solves the Obstruction problem, but brings back the changes in red problem if you change a value and Share, both New Value and Old Value are the same and red. Try to understand again. If you don’t Run after Share, why New Value and Old Value are same and red? They should be same in Dynamo files, but not in GH old file - OK was problem of GH. In Dynamo maybe Old Value should be empty? Or, well …

When a name is in field Obstruction the checkbox is empty. But should only be empty if that name is username.

Found on the Dynamo forum that a Dutch Dynamo User Group was formed. The first meeting was 19-1-2016. Participating were about 16 companies. Amongst them: Groosman, Broekbakema, Ector Hoogstad, Roosros, Multical, Deerns, IMd, Aronsohn, B+M. Ekko Nap of Groosman and Arnout van IJsseldijk of IMD structural engineers are the motors behind this group. Dieter Vermeulen from Autodesk is also a name to remember. Asked on the forum if I could attend next meetings. Which was granted. Check #DuchtDynamoUserGroup on Twitter.

In the meanwhile the Dutch Dynamo User Group had its second meeting at IMD in Rotterdam on 5-4-2016. It became a part of the Revit Gebruikers Groep (RGG). 45 people attended (Figure 5). The group was split in about half beginners and half advanced Dynamo users. 



Figure 5 Dutch Dynamo User Group meeting 5-4-2016

An inventory was made of the subjects that people were interested in. I added Collaborative Design of course (Figure 6). Unfortunately the 5 selected subjects most advanced Dynamo users were interested in didn’t comprise Collaborative Design (Figure 7). I participated in the sustainability group (duurzaamheid). We started with the problem of getting the windows surfaces in a list per room. This seems necessary for BREAM. Standard Revit doesn’t give this possibility in the schedules. Dynamo also cannot find the room a window belongs to. You can find the wall it is hosted in. But a wall can be used as boundary for different rooms… Using the [Revit Lookup plug-in](https://github.com/jeremytammik/RevitLookup) I noticed that a window has a hidden property that gives the room parameters. So it should be possible to code a Dynamo node that can access this. I will propose the group to dive more into this. But there must be a budget for it of course… BTW, the Revit Lookup plug-in is open source and very interesting to analyse how it accesses the Revit API and Revit APIUI. You have to compile the source after locating the referenced dll’s. In the Post Build Events we find:

copy "$(ProjectDir)RevitLookup.addin" "$(AppData)\Autodesk\REVIT\Addins\2016"

copy "$(ProjectDir)bin\Debug\RevitLookup.dll" "$(AppData)\Autodesk\REVIT\Addins\2016"

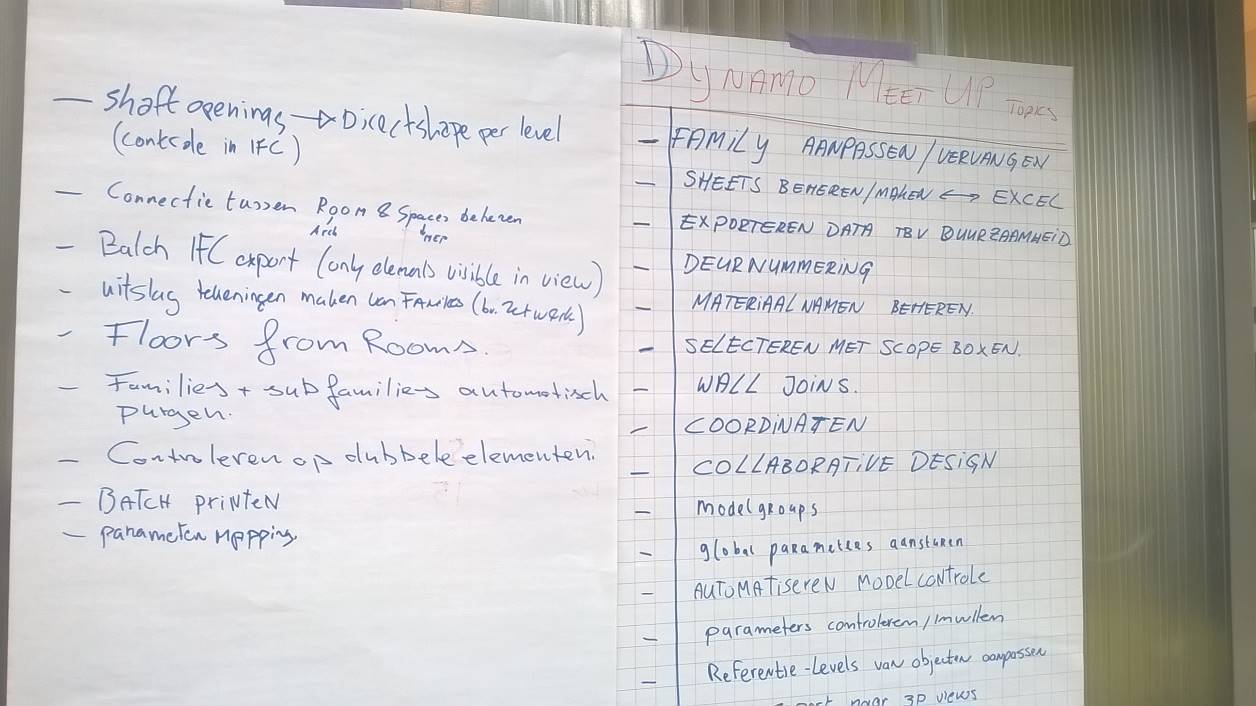


Figure 6 Interesting subjects for Dutch Dynamo User Group

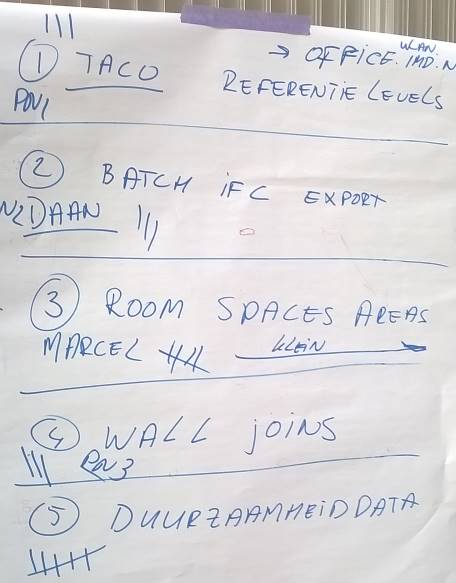


Figure 7 Selected subjects

Also interesting is that RGG uses [Trello](http://help.trello.com/article/708-what-is-trello) and DDUG will start using it too. You can find it [here](https://trello.com/b/A5QYT5V1/dutch-dynamo-user-group). In order to raise my goodwill I solved the Windows in Rooms problem and made a card with attached files.

Pick-up pCOLAD-for-Dynamo again. When you start empty in GH and in Dynamo, share in GH, get message and start over in Dynamo, share in Dynamo everything is red except date and author. Everything should be white! According to Figure 4 share is not forcing a Run. And it shouldn’t. Just before it shows the success message it says: myPropDataTable = MyDataCollectorClass.localDataTable.Copy(); Maybe should run Compare() then? Or should the change of myPropDataTable invoke Compare() automatically? Check with Breakpoint and then F11… Well indeed it shows that all item properties are going to be set. But this doesn’t go to the ObjectToBackGroundConverter. Also not after Compare(). Only if you hit Run… Reason is of course that ObjectToBackGroundConverter only reacts on SetChanged notification. No it doesn’t. There is a SetSame property of Item… add it to Compare(). But debugging showed that oldDataTable is empty (only headers). How is that possible? Anyway, just copy myPropDataTable. Then Compare. And still everything red! Check when ObjectToBackGroundConverter is called. That is in CSVControl.xaml in the <DataTemplate x:Key="changedCells"> and the <DataTemplate x:Key="commentCells">. Of course these are only used when you build the xaml. Most easy would be to force a Run, but when you change a value, it would be good to have that red too… Well, it is of course, because when you change a value (except Comments) you have to Run the solution any way. So change the flow chart and go to Run after Share and after change

Do not overwrite Old Value, comments and **obstruction**

Here you write New Value

To Old Value.

pSHARE Inputs

Copy sharedFile to

localFile

Compare

datatables

Copy localFile to

oldLocalFile

oldLocalFile

exist?

no

yes

Put in oldDatatable

sharedFile

exist?

no

yes

Create

sharedFile

Show (difference

in red)

First run?

yes

no

sharedFile

changed

externally

Change own

parameters

Write to

sharedFile

Write to

oldLocalFile

Merge own

parameters

Put localFile in

localDatatable

External

event

Manual

change

Decision

Process

Share

Write to

LocalFile

All three the same

Figure 8 Flow chart y

That would result in Figure 8. However we should be aware that in this case, when you Share, all csv files are the same. But when you only change parameters without sharing, you won’t see any changes, because the localFile is put into localDatatable and was not changed. However you merge with your own parameters and if you don’t overwrite Old Value, comments and obstruction, as is commented in grey callout? Or when you Run, write to LocalFile? Then LocalFile is not same as sharedFile… Is that a problem? Yes, according to the chart, when external share, you don’t change LocalFile! You could of course write at that moment. In fact in code you already do. So no problem.

An other solution could be to use global variables to distinguish if you hit the Share button or if there was an external share… Maybe that was the formPopulate property (which was used in GH)? It would avoid too much writing and reading files. The formPopulate property is checked when you run OpenCSV(file) with the idea that the CSVControl would only be hided if you set the pSHARE button to Off and showed again if set to On (then of course no need to load from csv file - localFile or History file). But for some reason this was changed to closing the CSVControl. And if you set pSHARE button to On, you call ShowCSV() where you check the property isCSVControlOpen! In GH the On/Off is used to turn pSHARE completely off. It will not have any output. Probably has to do with the automatic running in GH. What happens if you turn off in Dynamo and external share occurs? That is ok. Gives message and forces Run.

OpenCSV(sharedFile) is called when you start and when you return from History display. And OpenCSV(HistoryFile) is called when you hit the History button. The formPopulate is set to true when you merge own parameters. It is set to false when external change occurs, when you hit Share, and when you show the History file.

So we have to make a difference between showing the CSV file and opening CSV files.

The formPopulate is also checked when a CSVUpdateHandler event is received (which is raised as soon as output of pSHARE changes, so when Run is hit). Eh, is that not what I am looking for to have red fields corrected when sharing? CSVUpdateHandler calls Compare(), sets MyPropDataTable to localDataTable, updates the solution (this.OnNodeModified(forceExecute: true); What is difference with forceRun? Maybe only the pSHARE node? Any way same as Start in flow chart. It also makes sure that runtype is manual (runtype(dm); and then if formPolulate is false: it closes the CSVControl and sets the On property to false (ShowParams(OnOff);) and sets the OnOff button to red (RaisePropertyChanged("OnOff"). All that is old stuff. Was maybe necessary in old version of Dynamo when Run didn’t work when you didn’t change a node? Is Compare() used when you Run? Compare() is called: in CSVUpdateHandler, in ShowCSV(), in Share(), in History when back to csv. So when is ShowCSV() called? When you hit the On/Off button in pSHARE node. So I guess it is ok to do without CSVUpdateHandler.

Recap:

1. The formPopulate is set to true when you merge own parameters.
2. It is set to false when external change occurs, when you hit Share, and when you show the History file.
3. If it is false when you Run (and UpdateCSVControl event is raised) Compare() is called, MyPropDataTable is set to localDataTable, pSHARE forceExecute is set to true. But this seams all not necessary anymore. Test commenting out. Gives warning: Please hit the run button first… That is because MyPropDataTable is null. When should it be set to localDataTable? Well after Compare(), but that is in ShowCSV() which is where you get this message instead. Remember that you start in MyDataCollectorClass and that you can not reference pSHARE there and that DataGrid in CSVControl is bound to MyPropDataTable in pSHARE. To go from MyDataCollectorClass to pSHARE is what you use CSVUpdateHandler for. But ShowCSV is in pSHARE too. You got there because you hit the On/Off button. So just set MyPropDataTable everytime you hit On of On/Off button? Try that. Could work, but now when external share the display doesn’t update… Why is that? Because of the (this.OnNodeModified(forceExecute: true); missing? Could well be. So instead force rerun when external share? Well it is already there… But you don’t pass by the On/Off button script. And so MyPropDataTable is not updated. Any chance of updating MyPropDataTable another way? Well you forceRun in OnChanged method of MyCollectorClass. So you get to pSHARE class in public override IEnumerable<AssociativeNode> BuildOutputAst(List<AssociativeNode> inputAstNodes). So copy the MyDataCollectorClass.locaDataTable to MyPropDataTable there. Ok that works.

But what was I looking for? Eh, how to get correct red fields when you hit Share. We got rid of CSVUpdateHandler, so there is not the solution. Somehow we have to force the update of the CSVcontrol. Most simple is to close it and then create it again. However, we are using binding and if you change an Item, it should update automatically. Why is that not working? Because you use a template? No, because e.g. the TextValue of a TextBox in a DataGridCell updates correctly. That is TwoWay of course. If you try to set the BackGround to a converter TwoWay you get an error and it turns out that you need a path to a property for TwoWay. Tried to use a TextChanged trigger on the textbox, but no success. The command doesn’t fire. In fact it does, but you have to put an eventhandler in code-behind. Found solution with Blend: go to the CSVControl.xaml tab, find myXamlTable in the Objects and Timeline window, right click, choose Edit Additional Templates > Edit Generated Items (Item Template) > Edit Current, click in the Text tag, select in the Properties window the Event sign (top right) and type a name (CommentChanged in my case) in the LostFocus field. That will create the handler in code-behind. LostFocus is better than TextChanged, because the latter fires every time you type a character. The handler has the TextBox as sender. Using the FindUpVisualTree function you can find the DataGridRow and its index and compare the values in the “Comments” column. The code:

private void CommentChanged(object sender, RoutedEventArgs e)

{

//this is triggerd by the LostFocus event in the datatemplate "commentCells"

//try to set the background of the TextBox to pink

DataTable dt = MyDataCollector.MyDataCollectorClass.oldDataTable;

TextBox tb= (TextBox)sender;

DataGridRow dgr = FindUpVisualTree<DataGridRow>(tb);

var i = dgr.GetIndex();

DataRow dr = dt.Rows[i];

MyDataCollector.Item it = dr["Comments"] as MyDataCollector.Item;

String s = it.textValue;

if (!tb.Text.Equals(s))

{

tb.Background = Brushes.Pink;

}

}

private static T FindUpVisualTree<T>(DependencyObject initial) where T : DependencyObject

{

DependencyObject current = initial;

while (current != null && current.GetType() != typeof(T))

{

current = VisualTreeHelper.GetParent(current);

}

return current as T;

}

Now how can I force the update of the whole CSVControl when I hit Share? You run through the ObjectToBackGroundConverter when you load the CSVControl. So just close and recreate it? Ai, now when I change a parameter value and hit Run, nothing happens. Should show the new Value in red. However twice ShowParams(OnOff); in the Share command effectively updates the control. But strange enough the changed value is red, so is the date and there is no OldValue.

Ok, why is changing a parameter value not showing after Run. Check if it has to do with the LostFocus event in the TextBox. No. So probably with the removal of the CSVUpdateHandler. Yes! So now cells update red correctly. External Share also OK. Only it turns out that Share in Dynamo replaces the OldValue in the shared file. That should not be the case of course. Then others can not see what the old value was compared to the new. So you have to write to the shared File before you set the Old Value to the New Value. Eh, how do you get the new Old Value then ever in the shared file? So you have to set the Old Value in the localDataTable to the New Value of the oldDataTable. Yes. Almost done!

## Getting Obstruction work correctly

Obstruction is not yet correctly working. Should turn red when you uncheck, and checkbox should only check for your own name. When a name is in field Obstruction the checkbox is empty. But should only be empty if that name is username. The last issue is easily solved by checking the username property. But turning another field red then the one you clicked is different cook.

How does it work? The checkbox is in a seperate column and bound to the IsChecked property of pSHARE. A Loaded event with the name "myCheckBox\_Loaded" in CSVControl.xaml is handled in code-behind. There it only checks if the field “Obstruction”is empty. Check if the value contains the userName instead. How to turn it red when you change? Same trick as with Comments? What event to use then? The Click event seems best. But wait a minute. Everytime you run Compare() and the value is different from oldDataTable the Item.IsChanged is set to true. Could you not just use the RaisePropertyChanged("MyPropDataTable");Probably not, because it is already in the IsChecked set method. Maybe because in the XAML the DataGrid’s ItemsSource="{Binding MyPropDataTable, Mode=OneWay}" is set to OneWay. Can we change that to TwoWay? Would that also solve the Comments issue? Try. No doesn’t work. Compare() doesn’t run when you hit the check box. Also a bit overkill to check all cells, if you know which cell was changed. So solve it in code-behind. It became next code:

private void myCheckBox\_Click(object sender, RoutedEventArgs e)

{

CheckBox cb = (CheckBox)sender;

DataGrid dg = FindUpVisualTree<DataGrid>(cb);

DataGridRow dgr = FindUpVisualTree<DataGridRow>(cb);

//get the value in the Obstruction column of DataGrid. Find the indexes.

var dgri = dgr.GetIndex();

//throws null exception

//var dgci = dg.Columns.Single(c => c.Header.ToString() == "Obstruction").DisplayIndex;

int dgci=0;

foreach (DataGridColumn dgco in dg.Columns)

{

if (dgco.Header!=null && dgco.Header.Equals("Obstruction"))

{

dgci = dg.Columns.IndexOf(dgco);

}

}

//get the cell

DataGridCell dgc = ExtensionHelpers.GetCell(dg, dgr, dgci);

//var actualItem = dgc.GetValue(MyDataCollector.MyDataCollectorClass.userName);

DataTable odt = MyDataCollector.MyDataCollectorClass.oldDataTable;

DataTable ldt = MyDataCollector.MyDataCollectorClass.localDataTable;

DataRow odr = odt.Rows[dgri];

DataRow ndr = ldt.Rows[dgri];

MyDataCollector.Item oldItem = odr["Obstruction"] as MyDataCollector.Item;

MyDataCollector.Item newItem = ndr["Obstruction"] as MyDataCollector.Item;

if (!newItem.textValue.Equals(oldItem.textValue))

{

dgc.Background = Brushes.Pink;

}

else

{

dgc.Background = Brushes.Transparent;

}

}

But of course it is violation the MVVM pattern. The CSVControl should not know anything about the DataTables. Only binding to dependency properties. Better would be to solve it in the ViewModel (pSHARE). Since there are already bindings to IsChecked property, you should use that and set the Item.IsChanged property if Obstruction field in oldDataTable and locaDataTable are different. Would also solve issue if you hit the CheckAll or UnCheckAll buttons. So again: how can you force running through ObjectToBackGroundConverter. Check again: first run CSVUpdateHandler(null, EventArgs.Empty). Well, it does force to go through ObjectToBackgroundConverter, but the Item.IsChanged is false. [Vaguely I remember](../../../../Office/R&O/2016/Op%20te%20sturen/Onderzoekverslag%20Hubers%202015.docx#New_Item) that when getting the advice to use an Item object in stead of working with strings in the datatable, it was also adviced to make new Items in certain cases.

Or maybe I used the IsChanged property, while only SetChanged has a notifier. Nah, in Compare() you use SetChanged. That’s not it. So try with New Item. Nah, you also do that in Compare(). Searching for a solution I thought a Code Map could maybe help (Figure 9). The red dots are breackpoints. The figure shows the complexity or the code. And this is only the pSHARE code.

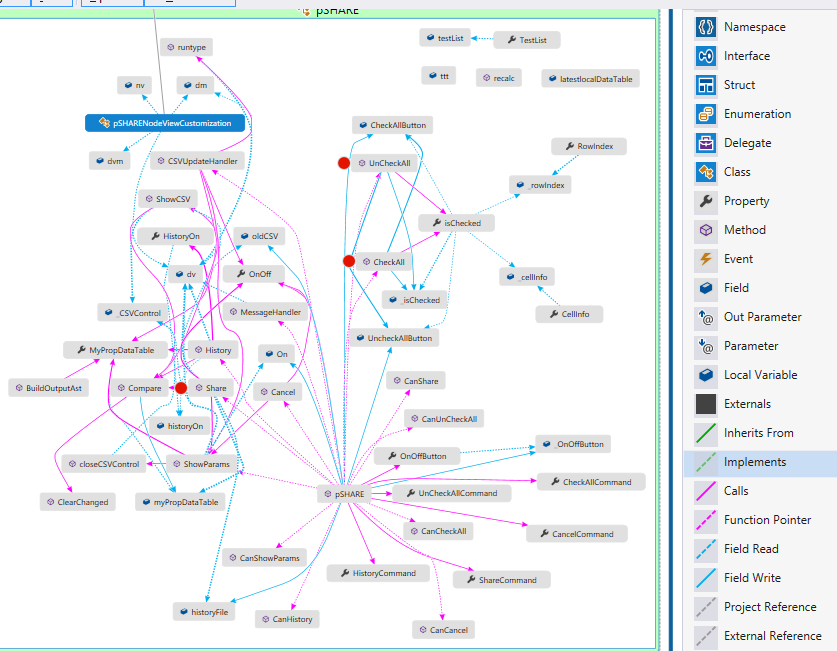


Figure 9 Code Map of pSHARE

However if we remove all the information that has nothing to do with the problem of not updating the colour of changed Obstruction fields we get Figure 10.

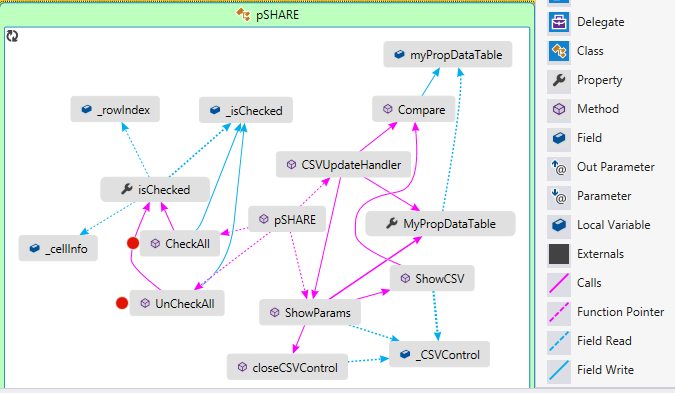


Figure 10 Code Map related to update of Obstruction field background.

It makes wonder what is the difference between a Property, a Field and a Local Variable. When changing the icon of Local Variable it turns out that in the map we don’t have them. [Here](http://stackoverflow.com/questions/295104/what-is-the-difference-between-a-field-and-a-property-in-c) we can read what the difference is between a Property and a Field. Most important to remember is that a Property exposes a Field, but that a Property has a get and set accessor. You can bind a control to a Property, not to a Field. So in Figure 10 it shows that Compare method writes to a Field and that’s why it doesn’t update the background colour of the control. It should write to the MyPropDataTable property. Try it.

Now maybe found the solution. The PropertyChanged is null when you change the IsChanged property of an Item. Somehow I got the feeling it had something to do with the way it was setup in the beginning. As mentioned before: [Vaguely I remember](file:///D:\Data\Office\R&O\2016\Op%20te%20sturen\Onderzoekverslag%20Hubers%202015.docx#New_Item) that when getting the advice to use an Item object in stead of working with strings in the datatable, it was also adviced to make new Items in certain cases. Checked it. And was done, because the INotifyPropertyChange interface was not used there. Later I implemented it for the IsChanged property. However, because this property is not bound to a control, it will always be null . The property is used in the ObjectToBackGroundConverter which is used in the binding for the Tekst in DataTemplate “changedCells”, which is the default template for the columns. So when the tekst changes this is triggered, but not when you change the IsChanged property of an Item. Even not if you do it through the SetChanged method, where you have the NotifyPropertyChanged(“IsChanged”). That has no function then. So make a binding to IsChanged for the Background. Hmm, debugging shows that now when you check a checkbox you go through the converter, but the value is not Item, but true. While it will only set background to red if value is item and item.IsChecked is true. So don’t bind to IsChanged, but to Item. That of course then doesn’t work with the IsChanged property.

Get this right: the DataContext of the CSVcontrol is pSHARE. The ItemSource of the DataGrid is MyPropDataTable, which is filled with Items. The DataTemplate for the default columns of DataGrid has a binding to these Items in the TextBox. It is kind of strange that just Binding works. Maybe because it automatically takes the only String property of Item? Probably because of the public override string ToString() method of Item. Indeed if you explicitly bind to TextValue everything works ok for the text. So why if I bind IsChanged to BackGround this does not work? Mind this: it should go through the ObjectToBackgroundConverter where you then should check if the item.IsChanged is true. Debug step by step. Encounter in ObjectToBackgroundConverter a value that is not of type Item. So that is the problem. If you bind the IsChanged property, you send a Boolean to the converter, while if you bind the TextValue, the XAML binds to the Item. Just Binding without a property should send the Item though. But I tried that before… Try again. But now in the NotifyPropertyChanged method PropertyChanged is null, and so you don’t go through ObjectToBackgroundConverter. So you have to bind to the Boolean IsChanged, and then in the ObjectToBackgroundConverter return in the case of a Boolean the Item. But how do you know which Item belongs to the Boolean? Can I use the object parameter for this? No, can only be a system type. But maybe we can use a [multi-binding](http://www.codeproject.com/Articles/328978/Introduction-to-multi-binding-and-multi-value-conv). However, then we can find the Item which IsChanged is true, but now way to find the corresponding Item in the oldDataTable. Since we always go through the checkbox and we already have a command bound to it, we should solve it there. Didn’t we try already? O, well that is not MVVM either. So just leave it in code behind. In fact what shows here is that the approach of making a Parameter class with properties that are shown in a DataGridRow would be better. That could be a mission after testing this prototype. Or, why would that solve this problem? In fact you would still have to compare the Obstruction property of the Parameter class with an old instance and react on the change of a checkbox. In fact now the situation is better, because you have instances of the Item class in a datatable. So you can set the IsChanged property of the Item. It must be possible to react on that.

Again: what is going on? The user clicks on a checkbox. Since the checkbox is bound to the IsChecked property of the pSHARE class, you can react on the change there. What you want is that the code finds out if the text in the Obstruction column of the MyPropDataTable property of pSHARE differs from the one in oldDataTable. If that is the case, then the IsChanged property of the corresponding Item in MyPropDataTable should be set to true. For this you use the SetChanged property that has the NotifyPropertyChanged method in its setter. When this fires, the CSVcontrol should update. This goes for the whole MyPropDataTable. If there is a change compared to oldDataTable the corresponding cell in the CSVcontrol should turn red. Therefore a DataTemplate is used with a converter. This works fine whenever MyPropDataTable is updated, because that property is bound to the ItemSource of CSVcontrol. So why doesn’t it work in the case of the Item.SetChanged in the Obstruction column? Check the code in IsChecked property of pSHARE. In fact the dependency property is called isChecked and the field \_isChecked. Two methods for distinguishing between properties and fields are mixed throughout the whole code: the underscore (\_) in front of the field and the Capital first character of a property. Anyway, in the setter of isChecked we find the Compare() command. In there however we don’t find changes to MyPropDataTable, only to localDataTable. So simply set MyPropDataTable to a copy of localDataTable after a Compare() command. Comment out all special code for Obstruction column. And try again... Or, why not set MyPropDataTable to localDataTable as a reference. Shouldn’t that update MyPropDataTable everytime you update localDataTable, or would that maybe give too much events… First try with copy. Oops, an infinite loop appears to have resulted from cross-dependent views. Apparently the updating of MyPropDataTable causes the code to go through the setting of isChecked. Of course, since the ItemSource of the DataGrid is bound to it. And also IsChecked="{Binding DataContext.isChecked, Mode=TwoWay, RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type DataGrid}}}"/> I don’t recall what that RelativeSource is about? [Here](https://msdn.microsoft.com/query/dev14.query?appId=Dev14IDEF1&l=EN-US&k=k(System.Windows.Data.Binding.RelativeSource);k(VS.XamlEditor);k(TargetFrameworkMoniker-.NETFramework,Version%3Dv4.5)&rd=true) it says that it is a way to change the DataContext. All that seems overdone. Why mention the DataContext.isChecked and then also use the DataGrid as relative source? After some further work it becomes clear. You want to bind to the isChecked property of pSHARE not to a property of Item, which is the content of the ItemSource the DataGrid is bound to. Also because of the TwoWay binding, the relative source must be the DataGrid, not MyPropDataTable.

Debugging the change of a checkbox shows that when you put a new Item in a data row cell (dr["Obstruction"] = new Item(cellContent.Trim());) you repeatedly set the Item’s overide ToString() method and not only for the Item in Obstruction but also for the Item in Parameter, and not only for the actual row but also for row 0. Why? Must be something in Code Behind or in the CSVcontrol, because no step in between is noticeable. However we have a string cellContent = dr["Obstruction"].ToString(); So if you use cellContent further on in the code it is logical that it goes to Item.ToString to get the value. Anyway now you don’t update the CSVcontrol at all. So call a PropertyChanged method that updates either the whole table, or only the Obstruction cell. RaisePropertyChanged("isChecked"); of course only updates the checkbox. So try setting MyPropDataTable. Tried Item.SetChanged, but PropertyChanged in NotifyPropertyChange is null. Of course, because you don’t bind to IsChanged to go through converter but to Item. Well if I make a new Item, shouldn’t we go through the converter then? Apparently not… Only when you create the CSVcontrol. So use the Item.SetChanged and bind to IsChanged? Doesn’t work because DataContext of CSVcontrol is not Item but pSHARE? And the DataGrid ItemSource is MyPropDataTable which is filled with Items. Why if I bind to the IsChanged property of Item and use ObjectToBackgroundConverter and I hit the checkbox I don’t go through the converter and PropertyChanged is null? Is that because trigger it through a column of DataGrid that is outside the autogenerated part? In fact you set an Item.IsChanged of localDataTable, not of MyPropDataTable, maybe try that. Yes that is it. Now you go through the converter, but with value = true. So set binding to the Item (so just Binding). YES! Now we are getting somewhere. Now build check in if oldDataTable is different then localDataTable for the Obstruction cell. Finally the code became:

private bool \_isChecked;

/// <summary>

/// property of pSHARE telling if a row is checked and so a value obstructed

/// </summary>

public bool isChecked

{

get { return \_isChecked; }

set

{

if (CheckAllButton)

{

foreach (DataRow dr in MyDataCollectorClass.localDataTable.Rows)

{

//DataRow MyPropDr = MyPropDataTable.Rows.Find(dr["Parameter"]);

string cellContent = dr["Obstruction"].ToString();

if (cellContent.Contains(MyDataCollectorClass.userName))

{

// remove username from the cell

cellContent = cellContent.Replace(MyDataCollectorClass.userName, "");

//remove double and end commas

cellContent = Regex.Replace(cellContent, "/{2/}", "/").Trim('/');

checkOldDataTable(cellContent, dr);

}

}

RaisePropertyChanged("isChecked");

}

else

{

if (UncheckAllButton)

{

foreach (DataRow dr in MyDataCollectorClass.localDataTable.Rows)

{

string cellContent = dr["Obstruction"].ToString();

if (cellContent == "")

{

cellContent = MyDataCollectorClass.userName;

checkOldDataTable(cellContent, dr);

}

else

{

if (!cellContent.Contains(MyDataCollectorClass.userName))

{

cellContent += "/" + MyDataCollectorClass.userName;

checkOldDataTable(cellContent, dr);

}

}

}

RaisePropertyChanged("isChecked");

}

else

{

\_isChecked = value;

DataRow dr = MyDataCollectorClass.localDataTable.Rows[\_rowIndex];

string cellContent = dr["Obstruction"].ToString();

if (\_cellInfo != null && !\_isChecked) //add the userName

{

if (cellContent == "")

{

cellContent = MyDataCollectorClass.userName;

checkOldDataTable(cellContent, dr);

}

else

{

cellContent += "/" + MyDataCollectorClass.userName;

checkOldDataTable(cellContent, dr);

}

}

else

{

// remove username from the cell

cellContent = cellContent.Replace(MyDataCollectorClass.userName, "");

//remove double and end commas

cellContent = Regex.Replace(cellContent, "/{2/}", "/").Trim('/');

checkOldDataTable(cellContent, dr);

}

}

}

}

}

private void checkOldDataTable(string \_cellContent, DataRow \_dr)

{

Item it = new Item(\_cellContent.Trim());

DataRow oldDr = MyDataCollectorClass.oldDataTable.Rows.Find(\_dr["Parameter"]);

if (!oldDr["Obstruction"].Equals(it))

{

it.SetChanged();

}

\_dr["Obstruction"] = it;

DataRow MyPropDr = MyPropDataTable.Rows.Find(\_dr["Parameter"]);

MyPropDr["Obstruction"] = it;

}

One little issue remains. The forward slash as separation between userNames in the Obstruction column works different in GH. It doesn’t delete the “/” at the end. Also when you first remove your userName with the checkbox and then later put it back again, it could change place. So in fact you should not compare the Items in oldDataTable and localDataTable, but check if they contain the same userNames. So parse the TextValue of the cellContent and build a list or something and compare the content. [Here](http://stackoverflow.com/questions/13630545/check-whether-two-comma-separated-strings-are-equal-for-content-set) we find the best method for this. It uses the HashSet, which is a set of unique values without order. You can consider it a to be a Dictionary<TKey, TValue> collection without values. Now the checkOldDataTable function becomes:

private void checkOldDataTable(string \_cellContent, DataRow \_dr)

{

Item it = new Item(\_cellContent.Trim());

DataRow oldDr = MyDataCollectorClass.oldDataTable.Rows.Find(\_dr["Parameter"]);

var set1 = new HashSet<string>(\_cellContent.Split('/').Select(t => t.Trim()));

var set2 = new HashSet<string>(oldDr["Obstruction"].ToString().Split('/').Select(t => t.Trim()));

set1.Remove("");

set2.Remove("");

bool setsEqual = set1.SetEquals(set2);

if (!setsEqual)

{

it.SetChanged();

}

\_dr["Obstruction"] = it;

DataRow MyPropDr = MyPropDataTable.Rows.Find(\_dr["Parameter"]);

MyPropDr["Obstruction"] = it;

}

There are new versions of Dynamo (1.0.0 ) and Visual Studio 2015 (14.0.25123.00 Update 2). Check if everything still works. Interesting readme file, stating that now the x.y.z versioning system will be used. Changes in x show breaks to the API causing developers to refactor their code. The y position changes are used for backward compatibility. The z position only for minor bugs. Changes can be found here: https://github.com/DynamoDS/Dynamo/wiki/Dynamo-Node-Changes and here: <https://github.com/DynamoDS/Dynamo/wiki/API-Changes>. And of course nothing works anymore. All references to be reset and new package location etc. etc.

These are the references in pCOLAD02.csproj:

C:\ProgramFiles\Dynamo0.9\DSCoreNodes.dll

C:\ProgramFiles\Dynamo0.9\DynamoCore.dll

C:\ProgramFiles\Dynamo0.9\DynamoCoreWpf.dll

C:\ProgramFiles\Dynamo0.9\DynamoUtilities.dll

C:\ProgramFiles\Dynamo0.9\Microsoft.Practices.Prism.dll

packages\CommonServiceLocator.1.3\lib\portable-net4+sl5+netcore45+wpa81+wp8\Microsoft.Practices.ServiceLocation.dll

packages\Prism.Core.6.1.0\lib\net45\Prism.dll

packages\Prism.Mef.6.1.0\lib\net45\Prism.Mef.Wpf.dll

packages\Prism.Wpf.6.1.0\lib\net45\Prism.Wpf.dll

C:\ProgramFiles\Dynamo0.9\ProtoCore.dll

C:\ProgramFiles\Dynamo0.9\ProtoInterface.dll

And these are the references in MyDataCollector.csproj:

"DSCoreNodes", C:\ProgramFiles\Dynamo0.9\DSCoreNodes.dll

"DynamoCore", C:\ProgramFiles\Dynamo0.9\DynamoCore.dll

"DynamoCoreWpf", C:\ProgramFiles\Dynamo0.9\DynamoCoreWpf.dll

"Microsoft.Expression.Interactions,Version=4.5.0.0,Culture=neutral,PublicKeyToken=31bf3856ad364e35,processorArchitecture=MSIL"/, "Microsoft.Practices.ServiceLocation,Version=1.3.0.0,Culture=neutral,PublicKeyToken=31bf3856ad364e35,processorArchitecture=MSIL", ..\packages\CommonServiceLocator.1.3\lib\portable-net4+sl5+netcore45+wpa81+wp8\Microsoft.Practices.ServiceLocation.dll

"PresentationCore"/, "PresentationFramework"/, "Prism,Version=6.1.0.0,Culture=neutral,PublicKeyToken=91a96d2a154366d8,processorArchitecture=MSIL", ..\packages\Prism.Core.6.1.0\lib\net45\Prism.dll

"Prism.Mef.Wpf,Version=6.1.0.0,Culture=neutral,PublicKeyToken=91a96d2a154366d8,processorArchitecture=MSIL", ..\packages\Prism.Mef.6.1.0\lib\net45\Prism.Mef.Wpf.dll

"Prism.Wpf,Version=6.1.0.0,Culture=neutral,PublicKeyToken=91a96d2a154366d8,processorArchitecture=MSIL", ..\packages\Prism.Wpf.6.1.0\lib\net45\Prism.Wpf.dll

"ProtoInterface", C:\ProgramFiles\Dynamo0.9\ProtoInterface.dll

"System"/

"System.Core"/

"System.Windows.Interactivity,Version=4.5.0.0,Culture=neutral,PublicKeyToken=31bf3856ad364e35,processorArchitecture=MSIL",

"System.Xaml"/

"System.Xml.Linq"/

"System.Data.DataSetExtensions"/

"Microsoft.CSharp"/

"System.Data"/

"System.Xml"/

"WindowsBase"/

Of course the ones in C:\ProgramFiles\Dynamo0.9\ don’t work anymore. N.B. [here](https://github.com/DynamoDS/Dynamo/wiki/API-Changes) it says.

ProtoInterface.dll has been removed from Dynamo project and some of the common interfaces and Attributes defined inside ProtoInterface.dll are now moved to DynamoServices.dll. [If your library/package was referencing ProtoInterface.dll then it is recommended to update the reference and re-compile your projects with DynamoServices](https://github.com/DynamoDS/Dynamo/wiki/Migration-of-ProtoInterface-To-DynamoServices).

So in pCOLAD02 replace:

C:\ProgramFiles\Dynamo0.9\DSCoreNodes.dll

C:\ProgramFiles\Dynamo0.9\DynamoCore.dll

C:\ProgramFiles\Dynamo0.9\DynamoCoreWpf.dll

C:\ProgramFiles\Dynamo0.9\DynamoUtilities.dll

C:\ProgramFiles\Dynamo0.9\Microsoft.Practices.Prism.dll

C:\ProgramFiles\Dynamo0.9\ProtoCore.dll

C:\ProgramFiles\Dynamo0.9\ProtoInterface.dll

And in MyDataCollector:

"DSCoreNodes", C:\ProgramFiles\Dynamo0.9\DSCoreNodes.dll

"DynamoCore", C:\ProgramFiles\Dynamo0.9\DynamoCore.dll

"DynamoCoreWpf", C:\ProgramFiles\Dynamo0.9\DynamoCoreWpf.dll

"ProtoInterface", C:\ProgramFiles\Dynamo0.9\ProtoInterface.dll

Microsoft.Practices.Prism C:\Program Files\Dynamo 0.9\Microsoft.Practices.Prism.dll

And it is a miracle, it compiles ok. However the pCOLAD package might be in the wrong place. This should go to:

So you have to change the line in pCOLAD02.csproj that says:

<Copy SourceFiles="@(Pack)" DestinationFolder="C:\Users\jhubers\AppData\Roaming\Dynamo\0.8\packages\pCOLADpackage\%(RecursiveDir)" />

To

<Copy SourceFiles="@(Pack)" DestinationFolder="C:\Users\jhubers\AppData\Roaming\Dynamo\Dynamo Core\1.0\packages\pCOLADpackage\%(RecursiveDir)" />

And now everything works fine again.

When you don’t put in any packages the interface looks like Figure 11.

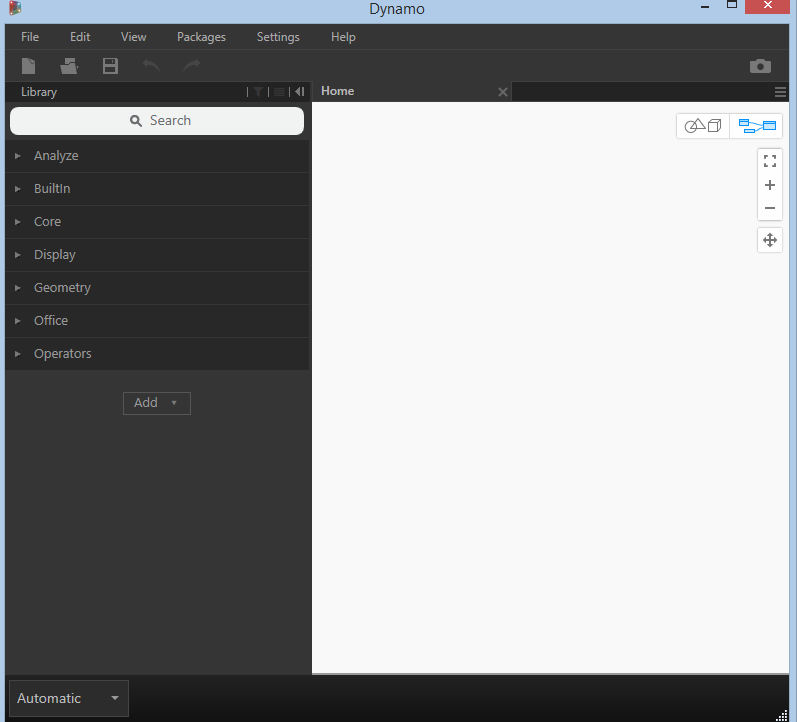


Figure 11 Interface 1.0 without packages

With pCOLAD package it looks like Figure 12.

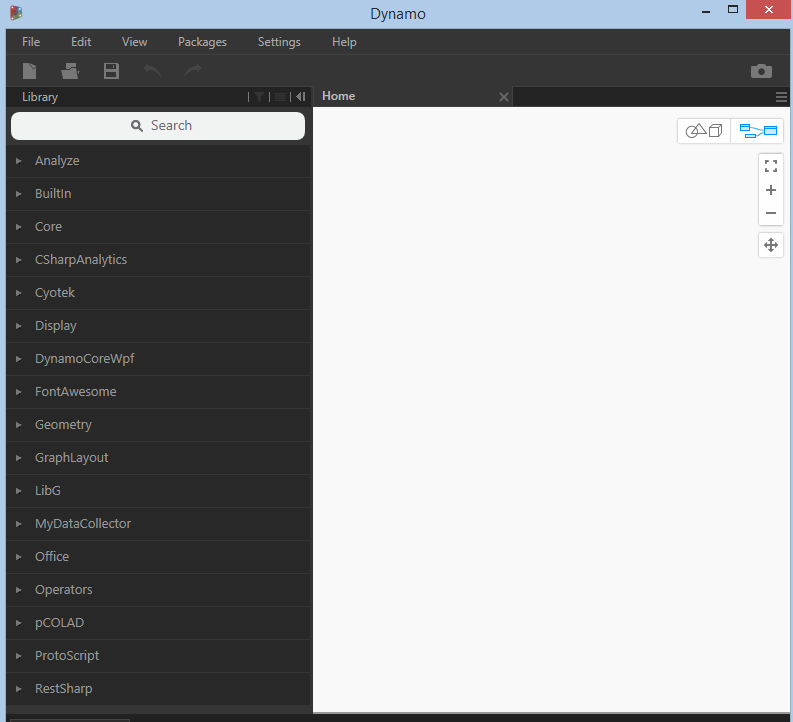


Figure 12 Interface 1.0 with pCOLAD package

This is probably due to some missing [IsVisibleInDynamoLibrary(false)] attributes.

Where should they be put? CSharpAnalytics, Cyotek, DynamoCoreWpf, FontAwesome, GraphLayout, LibG, MyDataCollector, ProtoScript, RestSharp should not be there.

## Reminder for working with the code after some time.

When you start again developing Dynamo/pCOLAD:

1. Open the VS pCOLAD.sln file.
2. Set in Tools/Options/Debugging/Enable Just My Code to true. To speed up. Maybe safer not to change it, because Grasshopper/pCOLAD needs it to be false. Only the first time debugging and loading the symbols of the used dlls takes much time. Next time (because caching is turned on in options) is much faster. With Enable Just My Code to true you’ll miss some intellisense feedback during debugging and some impossible locations to break.
3. Start Dynamo.
4. Go back to VS and hit Shift + Ctrl + Alt + F5 to start debugging. Or hit Ctrl + Alt + P and attach the DynamoSandbox process by hand. Wait until everything is loaded.
5. Go to Dynamo and start the pCOLAD test file.
6. Hit RUN and hit the Red start button in pSHARE.

When you start with Grasshopper/pCOLAD:

1. Open the pCOLAD10.sln file.
2. Set in Tools/Options/Debugging/Enable Just My Code to false.
3. Hit F5.

And then something went wrong and files created with Dynamo using pCOLAD nodes would not run anymore. Had to go back to version on GitHub. But it turned out that some of the files were not backed-up there: Fullscreen and Zoomboarder files. Also I got errors like: The name 'InitializeComponent' does not exist in the current context. After copying the missing files saving and closing the solution and reopening things were ok again.