SD226372

Developing Efficient Fabrication Hanger-Placement Solution Based on Dynamo for Revit

John (Zhenjun) Feng
Autodesk

Learning Objectives

- Learn how to generate Revit MEP fabrication hanger design automatically by the rule-based hanger placement tool developed in Dynamo Revit
- Learn how to create C# custom node for Dynamo Revit using Revit MEP Fabrication API

Description

In this class, we will demonstrate a tool to use the Dynamo custom node to automatically place Revit MEP fabrication hangers according to the rules stored in a Microsoft Excel sheet. Dynamo for Revit does not support creation of Revit MEP fabrication parts with standard nodes, so in the sample, we'll develop a C# custom node with Revit API. The topic covers both Dynamo custom node development and Revit MEP Fabrication design automation driven by programming based on Revit MEP Fabrication API.

Speaker

John (Zhenjun) Feng, joined Autodesk (China) Software Research &Development Center (i.e., ACRD) since 2006. He firstly worked as a Senior Software Engineer and Team Lead for ACRD Revit API development team. He ever worked on many Revit API projects including Event API, Dynamic Model Update, External Service Framework etc. Since 2013 he moved to ACRD Revit MEP team as the Development Manager and Scrum Master, worked on developing Machinal, Electrical, Plumbing as well as MEP fabrication hanger features in Revit. At same time, as one member of the Autodesk global Revit API Guild Committee, he provides Revit API technical consultant for some customer issues as well as new API design review for development teams. John got both the Bachelor and Master's degree from Shanghai Jiao Tong University, majored in Computer Simulation & Optimization, CIMS Lab, Automation Department.

zhenjun.feng@autodesk.com

Developing Hanger-Placement Tool with Dynamo Revit

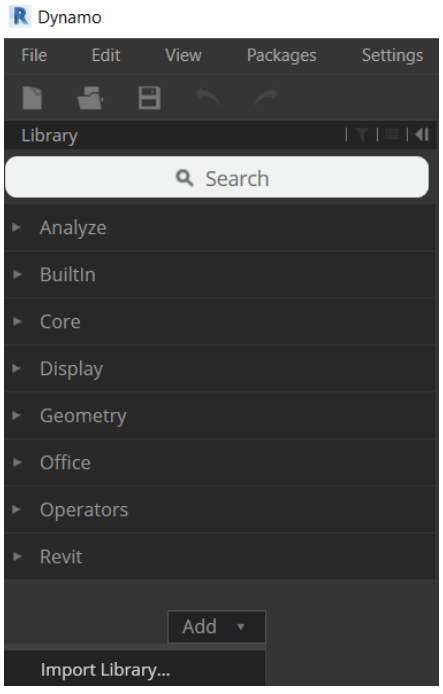
For mechanical detailers who design fabrication detail model with Revit, part by part fabrication hanger placement is time-consuming considering there could be thousands of hangers in the model. It is efficient if there is a tool that can place hangers by one click according to certain rules. The hanger placement tool can greatly improve the design efficiency from hours to several minutes. This class demonstrates a hanger placement tool, and then show step by step about how to make it by developing C# custom node for Dynamo Revit via Revit API.

The key points of rule-based hanger-placement tool

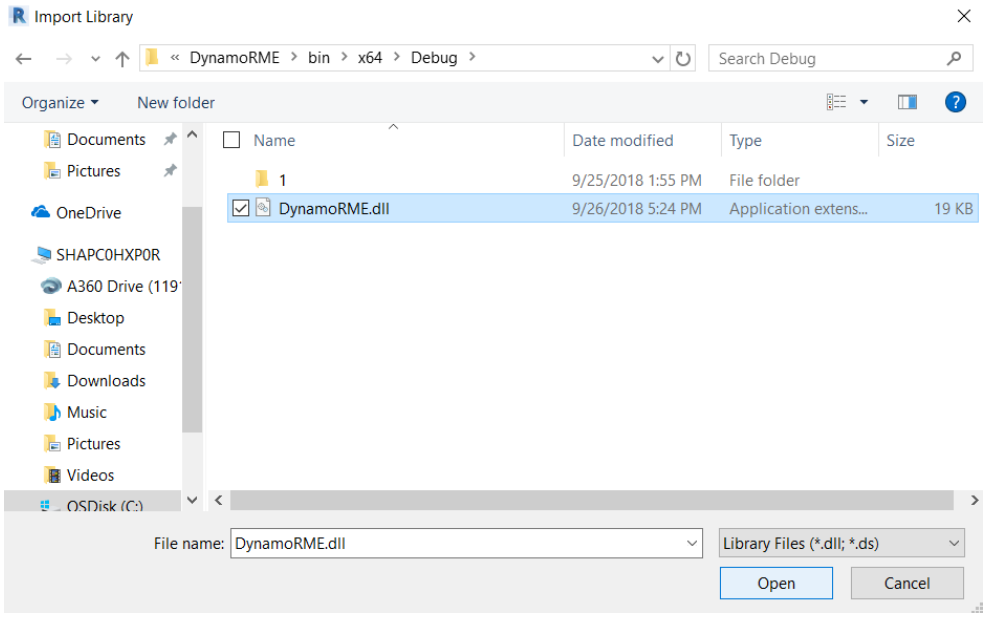
- It can place hangers by one click using certain rules
- The rules are stored in excel sheet as input
- Option to control whether the hangers attach to structure above
- Option to control whether the placement is one-by-one animation

Steps to load and use the Dynamo custom node

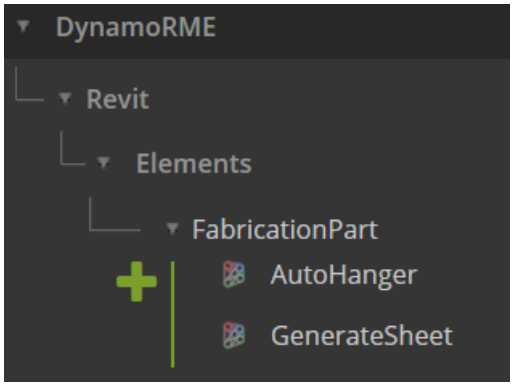
- Open Revit(2019.0.1), Open the model(FabLayout_AU2018_Demo.rvt), open Dynamo from ribbon “Manage” tab, FILES New, click “Add”, “Import Library…”



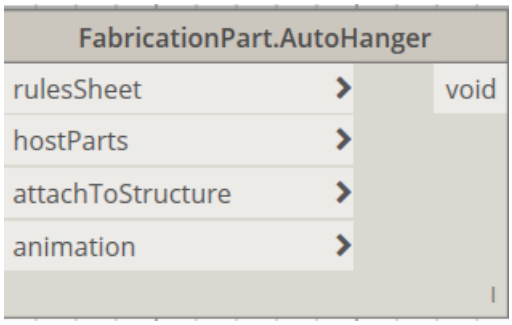
- Browse the assembly file (DynamoRME.dll) and click “Open”



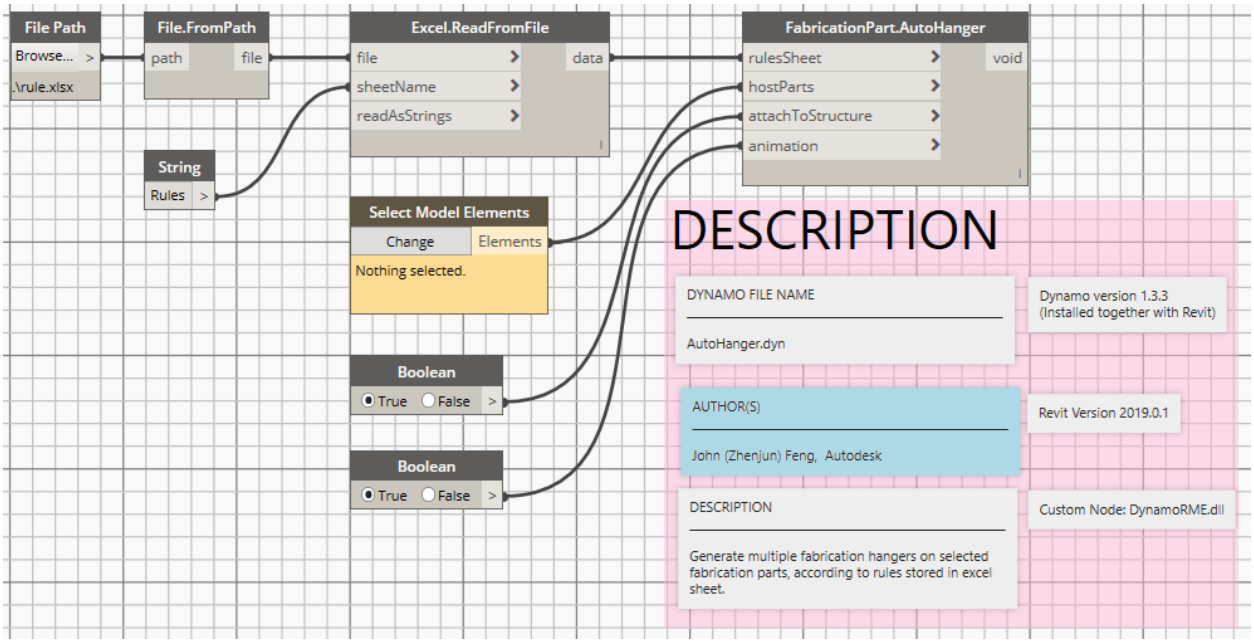
Then you will see the new Dynamo custom node: “DynamoRME”, “Fabrication Part”, it contains two method blocks:



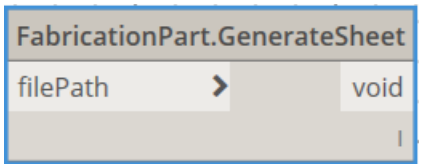
- The “AutoHanger” block below is for hanger placement:



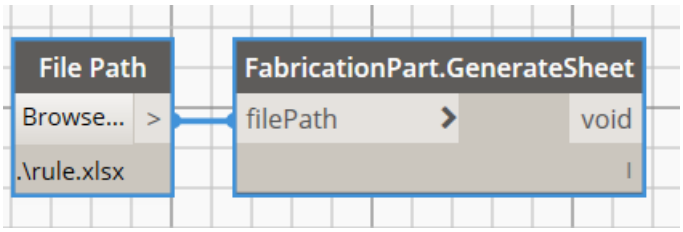
- The hanger placement tool consumes the “FabricationPart.AutoHanger” block, in the graph shown as below:



- The “GenerateSheet” block is for generating excel sheet to store hanger placement rules



- The graph for generating the rules sheet is shown as below:

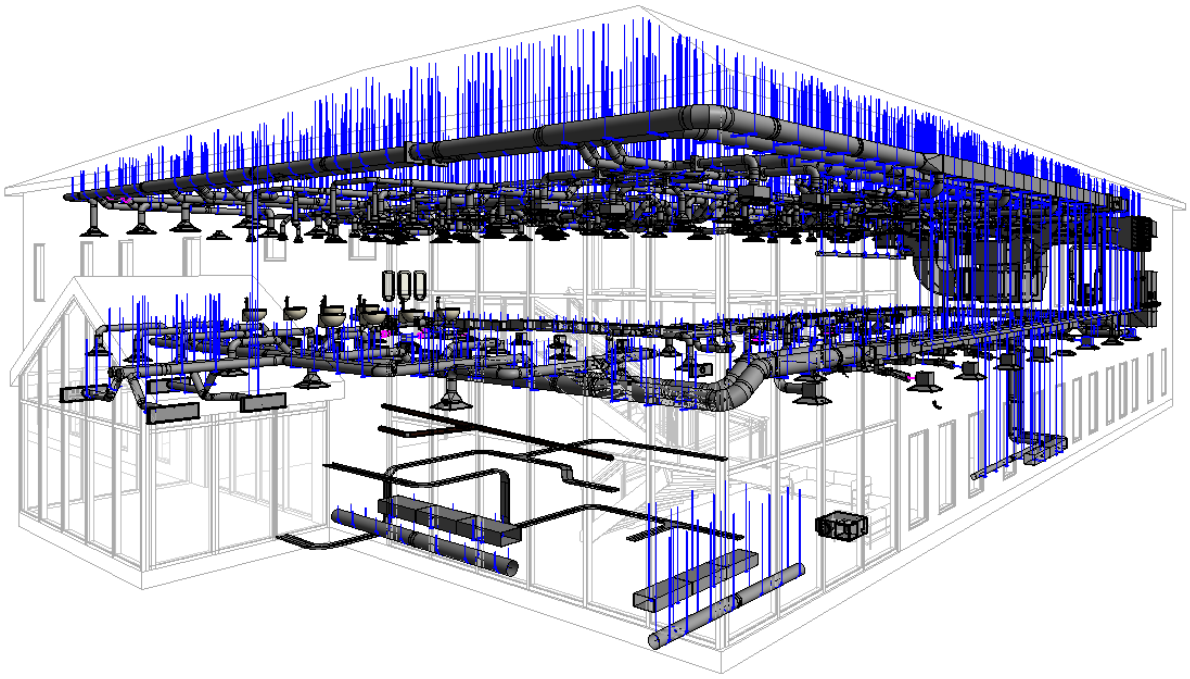


The generated excel file contains 4 sheets:

- Rules: for each fabrication service and each shape that are actually used in the model, there will be one row generated in the excel sheet, so user can fill rule data for that service and shape. If the same service and shape need to have several rules according to size of the host part, then you need to insert new rows grouped together with the same service and shape, and sort them from minimal to max size.
- Services: all the services from the fabrication configuration of the model
- Shapes: there are 4 options(Rectangular/Oval/Round/All). The “All” means for any shape, share a same rule.
- Buttons: all the hanger buttons from the fabrication configuration.

- Run the tool on the opened model(FabLayout_AU2018_Demo.rvt):
    - From Dynamo Revit, open the “AutoHanger.dyn”
    - From “File Path” node, click “Browse” to select the “rule.xlsx”
    - From “Select Model Elements” node, click “Change”, then select fabrication parts in Revit model.
    - Run the script, the hangers will be created in model.

  Please refer to the demo video for more instructions about using the tool and customizing the rules.
- The hanger placement result generated by the tool for an example Revit model



The hanger placement rule

The tool will place different type of hanger based on the host fabrication part’s “Fabrication Service”, “Shape” and “Max Size”. The host may be any fabrication part that we can place hanger on, for example: a straight duct, pipe or cable tray. The rules table contains the following columns:

- Fabrication Service: The fabrication service of the host fabrication part.
- Shape: the shape of the host fabrication part(“Rectangular”, “Oval”, “Round”, or “All”)
- Max Size: The max value of the host fabrication part’s “Width” and “Height”.
- Hanger Button: The Fabrication Service Button of the hanger that will be placed. The “Button” is something like the type of the hanger.
- Hanger Spacing: The distance between two hangers.
- Safe Distance: The minimal distance from a hanger to the fitting, tee, open end, or to the conjunction of two host parts.

Below is one example of the rules sheet:

	Fabrication Service	Shape	Max Size(In	Hanger Button	Hanger Spacing(Feet)	Safe Distance(Feet)
2	Ductwork: +2in WG	Rectangular	999999	Ductwork: +2in WG_Half Strap Hanger_HSH	3	1
3	Ductwork: +2in WG	Round	999999	Ductwork: +2in WG_Round Wire Hanger_RWH	6	2
4	Electrical: Ladder Cable Tray	Rectangular	999999	Electrical: Ladder Cable Tray_Strut Hanger_UH	4	1
5	Ductwork: -4in WG	Round	999999	Ductwork: -4in WG_Oval Strap Hanger_OBH	3	1
6	Piping: DWV PVC 40	Round	10	Piping: DWV PVC 40_Clevis Hanger Blk_CH	5	1
7	Piping: DWV PVC 40	Round	999999	Piping: DWV PVC 40_Riser Clamp_CH	8	2

For example, Row 6 and 7 in the excel sheet, represent two rules:

IF Fabrication Service=Piping: DWV PVC 40

AND Shape=Round

AND 0<Max Size<=10 Inch

THEN

Hanger Button=Piping: DWV PVC 40_Clevis Hanger Blk_CH

Hanger Spacing=5 Feet

Safe Distance=1 Feet

IF Fabrication Service=Piping: DWV PVC 40

AND Shape=Round

AND 10<Max Size <=999999 Inch

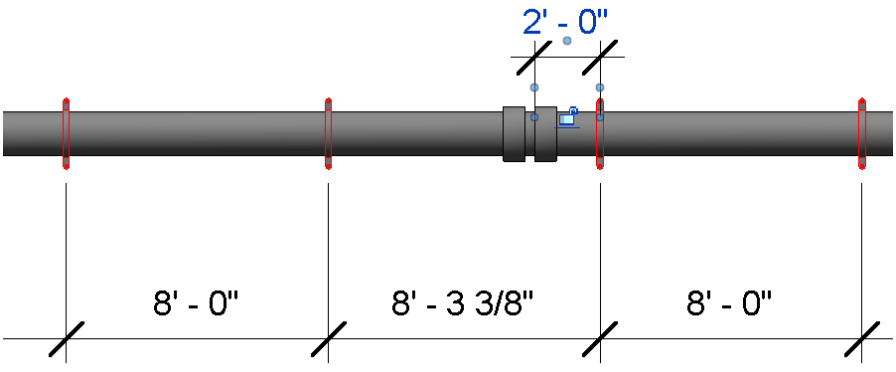
THEN

Hanger Button=Piping: DWV PVC 40_Riser Clamp_CH

Hanger Spacing=8 Feet

Safe Distance=2 Feet

Sometimes it cannot keep both the “Hanger Spacing” and “Safe Distance” rules at same time. In such case, the “Safe Distance” is hard rule, the “Hanger Spacing” can be specially adjusted. For example, when placing hangers on “Piping: DWV PVC 40 Round 16”, the rule of row 7 is applicable, and some detail of the tool’s placement result is as below. There is one hanger that can’t keep the equal distance(8’) due to it must keep safe distance(2’) to the conjunction.



Please notice the sample rule is just for an example. We do not guarantee the rule is best for production use. You should design the rule by self that fit your real need. But the rule-based design tool's workflow and the custom node software architecture can be reused.

Developing C# custom node for Dynamo Revit via Revit API

How to develop the hanger placement tool step by step is shown as below, the core part is to develop a custom node for Dynamo Revit:

- Using Visual Studio, create a C# "Class Library" project
- Add assembly references from Dynamo Core, Dynamo Revit, and Revit API
  This custom node project references five assemblies, the followings:
  - C:\Program Files\Dynamo\Dynamo Core\1.3\DSOffice.dll
  - C:\Program Files\Dynamo\Dynamo Revit\1.3\Revit_2019\RevitNodes.dll
  - C:\Program Files\Dynamo\Dynamo Revit\1.3\Revit_2019\RevitServices.dll
  - C:\Program Files\Autodesk\Revit 2019\RevitAPI.dll
  - C:\Program Files\Autodesk\Revit 2019\RevitAPIUI.dll

  Set all the assembly references' properties "Copy Local" to false.
- Using namespaces in the custom node assembly's C# source file

```
//using namespaces from RevitAPI
using DB = Autodesk.Revit.DB;
using UI = Autodesk.Revit.UI;
using Fabrication = Autodesk.Revit.DB.Fabrication;
using Connector = Autodesk.Revit.DB.Connector;

//using namespaces from DynamoRevit
using RevitServices.Persistence; //to access DocumentManager
```

- Implement a DynamoRevit wrapper class for the `FabricationPart` class in Revit API. In DynamoRevit, there is a base class `Revit.Elements.Element` defined in RevitNodes.dll assembly, for our custom node, we need to add a sub-class derives from `Revit.Elements.Element`, and implement some basic functions to make it an adapter for Revit API's FabricationPart element class:

```
namespace Revit.Elements
{
   public class FabricationPart : Revit.Elements.Element
   {
      internal DB.FabricationPart InternalFabricationPart
      {
         get;
         private set;
      }
      public override DB.Element InternalElement
      {
         get { return InternalFabricationPart; }
      }
      private FabricationPart(DB.FabricationPart part)
```

```
      {
         SafeInit(() => InitFabricationPart(part));
      }
      private void InitFabricationPart(DB.FabricationPart part)
      {
         InternalFabricationPart = part;
         InternalElementId = part.Id;
         InternalUniqueId = part.UniqueId;
      }
```

- For `Revit.Elements.FabricationPart` class, we define two public static methods that will be visible in the custom nodes, so we will have two building blocks that can be used in Dynamo graph. One method is for the hanger placement:

```
public class FabricationPart : Revit.Elements.Element
{
    public static void AutoHanger(
          object[][] rulesSheet,
          IEnumerable<Element> hostParts,
          bool attachToStructure,
          bool animation)
    {
          //access the Revit model document
          DB.Document revitDoc = DocumentManager.Instance.CurrentDBDocument;
          //then may call any RevitAPI to do what you want
          //skip implementation detail below
          ... ...
    }
```

Another method is to generate the hanger placement rules sheet:

```
public class FabricationPart : Revit.Elements.Element
{
      public static void GenerateSheet(string filePath)
      {
         DB.Document revitDoc = DocumentManager.Instance.CurrentDBDocument;
         //skip implementation detail below
         ... ...
       }
```

- Build solution, the custom node assembly(.dll file) will be produced. If you want to run and debug the C# code in the custom node, from Visual Studio, set break point in the code, in project properties debug tab, set Revit as the start project then start debugging(or from debug menu in VS, attach to running Revit process), and then run the Dynamo hanger placement tool that consumes the custom node to hit the break point.
- You may refer to the demo video for more details about how to develop the tool.

List of additional class materials

All necessary files are provided for you to try the hanger placement tool, the followings:

- Assembly .dll file(DynamoRME.dll) of the custom node
- Dynamo script files for both hanger placement and the excel rules sheet generation: (AutoHanger.dyn and GenSheet.dyn)
- A sample Revit fabrication model to run the tool on.
- One sample excel file that stores hanger placement rules. You may modify the rule data as needed. (The sample excel file and the sample Revit model are matched. If you want to place hanger on your own Revit fabrication model, you must generate the rules excel file via the GenSheet.dyn and fill the rule data by self.)
- Demo videos: 1. How to use the tool 2. How to develop the tool

References

Dynamo Revit 2019 help

Open Source Dynamo Source Code

How to create Dynamo custom node