Session 3.1 & 3.2

Everyday Dynamo: Practical uses for BIM managers
Paul Wintour & Scott Crichton, BVN

Class Description
There are many day-to-day laborious tasks that an architect needs to perform. Whether it is sequentially renumbering doors or rooms, cleaning up the Revit model, or a host of other tasks, this class will demonstrate how to automate these processes so that you can spend more time on what you do best, design. By harnessing the power of the Dynamo Package Manager, the class will illustrate how to extend Revit's capabilities to maintain your Revit model. We'll cover how to automate: workset creation, renumbering rooms, renumbering doors, purging unused filters, adjusting a room's upper limit, creating unplaced rooms, and adjusting title block properties to name but a few.

About the Speakers:

Paul Wintour is a UK Registered Architect based in Sydney, Australia. He is the founder of ParametricMonkey.com, a research and learning website dedicated to accelerating the computational literacy within architectural design. Paul holds a post graduate Masters of Architecture and Urbanism from the Architectural Association’s Design Research Laboratory (AADRL). Paul runs an embedded practice design studio at University of Technology Sydney (UTS) as well as advanced BIM electives. He has also taught at the University of New South Wales (UNSW), the University of Hong Kong (HKU), Queensland University of Technology (QUT) and the Architectural Association. Paul was previously the BuildingSMART manager for HOK’s Asia Pacific offices where he specialized in the interface between conceptual parametric modelling and project delivery. Currently, Paul is a Senior Associate at BVN where he provides BIM implementation and design technology services to project teams.
Scott Crichton is an experienced BIM Manager with experience across a range of project sizes and typologies. With over 24 years of experience, Scott has been involved as a professional in construction project design, management and delivery in a range of sectors. Scott draws on his background in a variety of project sizes and complexity of BIM deliverables, managing teams and actively incorporating Computational Design workflows and solutions in small, medium and large scale developments. Including the supervision and coordination, enhancement, review, conditioning and validation of construction building information models on projects, with provision for the linking to project schedules, and independent detailed quantity extraction, for use from inception through to the completion of projects. Scott is the Project BIM Manager at BVN, a lecturer at TAFE Queensland’s Southbank Campus, chair of the Brisbane Computational Design Group, and avid blogger and a contributor on the DynamoBIM forums.

Required software

- Autodesk Revit 2018
- Autodesk Dynamo 1.3
  - BVN, v1.2.5
    - Archi-lab, 2016.13.3
    - Clockwork for Dynamo 1.x, v1.0.3
    - Ladybug, V0.1.5
    - Lunchbox for Dynamo, 2017.2.12
    - Rynamo, v2016.11.2
    - Spring nodes, v100.0.1
    - SteamNodes, v1.2.3
  - Modelical, v1.2.2
- Microsoft Excel
1.0 Project Setup

Setting up a central Revit model ready for your project team can be a major task, especially if the project requires multiple models. You will need to enable worksharing and create worksets, create sheets (possibly from an Excel drawing register), update the sheet properties, and if you are converting a non-workshared model into a workshared model, you'll need to assign elements to the correct worksets. These are just a few of the tasks required. Due to its repetition involved, it is possible to use Dynamo to automate and speed up this process.

1.1 Create worksets

Note that before you create worksets you will need to enable worksharing. To do this in Revit 2016 or earlier, it was possible to click on the 'Worksets' button in the bottom ribbon or alternatively, go to Collaborate > Manage Collaboration > Worksets.

However, for whatever reason, in Revit 2017 it is no longer possible to activate worksharing via the Workset button. It is only possible via the Collaborate > Manage Collaboration > Worksets.

If you are using Revit 2017, you will be asked how you want to collaborate: Collaborate within your network; or, Collaborate using the cloud.
Once worksharing has been enabled, Revit will create default worksets and assigns elements and settings to these worksets. The default worksets are as follows:

- 'Shared Levels and Grids' - Contains all existing levels, grids and reference planes. Note that if you choose to collaborate using the cloud via BIM360, the workset will be named 'Shared Views, Levels and Grids'.
- 'Workset1' - Contains all existing model elements in the project. Once created, this workset can be renamed but it cannot be deleted.
3.1 & 3.2 - Everyday Dynamo: Practical uses for BIM managers
Paul Wintour & Scott Crichton & Scott Crichton, BVN

Worksharing window using Collaborate using the cloud

If you have an office naming standard for worksets, we can automate the creation of them. For example:

- 00_FACADE
- 10_STRUCTURE
- 20_VERTICAL CIRCULATION
- 30_INTERIOR
- 40_FF&E
- 50_SITE
- 60_MASSING
- 99_LEVELS & GRIDS
- Etc.

The Dynamo script utilises the BVN Package node ‘Workset.CreateAndRenameDefault’ to add the new worksets and rename the existing default worksets.

Dynamo script to create and modify the projects Worksets
The Dynamo script inputs are string lists, but you can also utilise an Excel file as in input by replacing the Code Block with an ‘Excel.ReadFromFile’ node.

Excel list of the projects Worksets

Dynamo script to create and modify the projects worksets from Excel list

The ‘Workset.CreateAndRenameDefault’ node from the BVN Package utilises the ‘Workset.GetAll’ and the ‘WorksetByName’ nodes from Archi-Lab package. The process of renaming the existing worksets is done by a custom Python script.
1.2 Elements on worksets

Once all of the relative worksets have been created for the project model, any existing and new information needs to be added to the relevant workset. A manual process can be very time consuming whereas Dynamo will allow us to automatically assign elements in the project model to the workset based on the elements category.

Worksharing window with office standard worksets created

As the worksets created are to an office standard they can be preprogramed into the Dynamo script for element assignment. The ‘Element.AssignToWorkset’ node utilises the BVN workset naming convention to automatically assign model elements.

Element.AssignToWorkset example
3.1 & 3.2 - Everyday Dynamo: Practical uses for BIM managers
Paul Wintour & Scott Crichton & Scott Crichton, BVN

Expanded version of the Element.AssignToWorkset node

The node on the Archi-Lab’s ‘Workset.GetAll’, ‘Workset.Name’ and ‘Workset.Id’ nodes to find all the worksets in the project, their names and IDs. It is an important point to remember that when you are assigning elements to worksets in Dynamo that you need to use the workset ID as the value in the ‘Element.SetParameterBName’ node.

Detail of the Dynamo script to assign Elements to the 60_MASSING Workset
1.3 Create isolated workset views

To visually verify that elements are on the correct workset, I find it extremely useful to have a 3D view which isolates a single workset. This is a much quicker way to visualise the model breakdown that to use the default Revit worksets display.

To automate this process, use the '3D Views by Workset' node in the Modelical. Views will automatically be named the same as the workset it is based on.

Dynamo script for creating isolated workset views

If new worksets are created, they will by default show in every view thus polluting our isolated views. To fix this, delete the existing views and re-run the script. If you re-running the script without deleting the existing views you will get duplicate views.
1.4 Create sheets from an Excel file

When developing a project document list, the most convenient way is to create the information within Excel. The formulas and formatting in Excel allow for a lot of mundane document management tasks to be expedited. The next step is to get the document list into Revit and create all the sheets from the Excel list. This is where Dynamo allows you to quickly input the Excel spreadsheet to populate sheets and sheet data.

For a simple document spreadsheet containing sheet number and name information, we can use a simple Dynamo script. The title block family and file path of the sheet list needs to be assigned as the inputs for the script which then get sorted and any redundant spreadsheet data removed before the sheets are created.
3.1 & 3.2 - Everyday Dynamo: Practical uses for BIM managers
Paul Wintour & Scott Crichton & Scott Crichton, BVN

Dynamo script to create sheets in Revit from Excel spreadsheet

There are 2 possibilities for sheet creation:

- ‘Sheet.ByNameNumberTitleBlockAndView’ – This is a standard Dynamo node that requires, Sheet Names, Sheet Numbers, Title block Family Type, and View inputs. If the project is in very early stages the views for the sheets are generally not created, so they cannot be used as an input without generating an error.

- ‘Tool.CreateEmptySheet’ – This is a node from the SteamNodes package that requires, Titleblock Family Type, Sheet Numbers, Sheet Names and the refresh inputs. The good thing about this node is that you don’t need to have the views in your model already created for it to work, as it will create empty sheets.

For a more complicated document spreadsheet containing sheet number and name data as well as additional parameter information such as ‘drawn by’ or ‘sheet issue date’ we need a more detailed Dynamo script.
BVN utilises an organisation method for assigning different trade packages in a document set to be assign a unique letter, the ‘BVN Alphabet’. This allows us to group and easily identify views and sheets in the Project Browser and group trade package documentation together. The more complex Excel document spreadsheet is identifying Sheet Number, Sheet Name, BVN Alphabet series, Sheet Issue Date, Design Stage, as well as having individual tabs for the different series letters. Although these additional fields allow for greater functionality in Excel they add complexity in how Dynamo extracts the data. The enhancements to this script allows for the additional sheet tabs in Excel to be read.

The script will also allow for information to be missing in the Excel spreadsheet, hence ‘List.Clean’ nodes are added to ensure nulls and empty lists are removed.
Additional sheet/view parameters can be input into the Revit model from the Excel spreadsheet.

1.5 Modify title block instance properties

When working in Revit it is important to understand the difference between sheets and title blocks. Sheets are the actual views that exist in the Project Browser. Whereas title blocks are the families that is placed on the sheet. These two elements behave differently and have different parameters associated to them.

Once a sheet list has been created, it is possible to see all the parameters associated with the sheets. This includes various instance parameters such as 'Drawn By' or 'Approved By'. These parameters can be individually updated within the Sheet List or exported to BIMlink if you need to make batch changes.

Generate a Sheet List
Often however, the title block family that resides on the sheet will have its own instance parameters which control the visibility of key plans, north point, scale bars, etc. The problem therefore arises that none of the title block’s parameters are visible within the Sheet List. Since title blocks can’t be scheduled, this means that to adjust any instance parameters on a title block, you need to navigate to each individual sheet, select the title block, and then modify its attributes in the Properties pallet. If you have a large amount of title blocks to update, this can become very time consuming. Of course, it is always possible to convert these instance parameters to type parameters within the title block family itself but this isn’t always desirable.

Sample sheet instance parameters

Sample title block instance parameters
To solve this problem, one can generate a simple Dynamo script as illustrated below. The script first collects all the title block families loaded into the project and returns its 'Sheet Number'. The title blocks are then alphabetically sorted and filtered based on the 'Sheet Number' parameter. In the example below, the script is collecting just the title blocks where the 'Sheet Number' parameter beginning with 'A'. This corresponds to BVN’s 'Introductory' drawing series. The script then collects certain instance parameters associated with those title blocks, namely, ‘Graphic Scale’, 'Project North', and 'True North'. These visibility parameters can then be updated by toggling between 'True' or 'False' in the Boolean node. This will update the instance parameters across multiple title blocks instantaneously saving much manual labour.

Dynamo script to modify a title block’s instance parameters

1.6 Create unplaced rooms

If you are lucky enough to have received a detailed design brief which list all the required rooms in your project, you may want to load these into your project to save time entering them manually. Within Revit these become ‘unplaced rooms’. When you go to place a room, you will have the option of placing a new room or an existing unplaced room if they have already been created.

Unplaced rooms

To automate the creation of these rooms en-mass, you can use the 'Room.CreateUnplaced' node as part of the BVN Dynamo package. Simply create a list of room names to be created and wire it up as shown.
Dynamo script to create unplaced rooms from a list

If you don’t want to do a bunch of syntax modifications in a code block to create your list, you can also use the 'Excel.ReadFromFile' node to import the list from an Excel spreadsheet.

Dynamo script to create unplaced rooms from an Excel file

Before running the script, ensure you are in a standard view (floor plan, 3d, etc.) and not a schedule or sheet view, otherwise Dynamo will be unable to create the rooms. Also, you may want to set Dynamo to run under 'Manual' as opposed to 'Automatic' so that you don’t get duplicate rooms if you accidently re-run the script. If you need the room number as well, you may want to check out the 'Room.UnplacedByNameAndNumber' node from Clockwork package which works in a very similar way.
2.0 Documentation

The following examples demonstrates how to quickly clean up your model for documentation purposes.

2.1 Adjust room’s upper limit

A common, time-consuming Revit task that is often complained about is having to manually adjust the upper limit of rooms. This is because unless you are very diligent as you place rooms by setting the upper limits correctly, rooms will default to an unbounded height of 2438mm (8 feet). This can prove frustrating when you want to apply a colour scheme in section.

Room schedule showing a room’s default unbounded height

However, with a simple Dynamo script this can be automated via a single click. To do this you’ll need the ‘Room.AdjustUpperLimit’ node as part of the BVN package. Simply connect a Boolean node to the input and run.
All your rooms will now be updated to have an upper limit based on the level above and with a 0 offset.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number</td>
<td>Name</td>
<td>Area</td>
<td>Upper Limit</td>
</tr>
<tr>
<td>1</td>
<td>Room</td>
<td>101.32 m²</td>
<td>LEVEL 01</td>
<td>4000</td>
</tr>
<tr>
<td>2</td>
<td>Room</td>
<td>131.12 m²</td>
<td>LEVEL 01</td>
<td>4000</td>
</tr>
<tr>
<td>3</td>
<td>Room</td>
<td>147.51 m²</td>
<td>LEVEL 01</td>
<td>4000</td>
</tr>
<tr>
<td>4</td>
<td>Room</td>
<td>169.86 m²</td>
<td>LEVEL 01</td>
<td>4000</td>
</tr>
<tr>
<td>5</td>
<td>Room</td>
<td>132.61 m²</td>
<td>LEVEL 01</td>
<td>4000</td>
</tr>
</tbody>
</table>

Room schedule showing updated unbounded height parameter

2.2 Adjust room’s centre location

Sample floor plan showing irregular placed rooms locations

One of the difficulties of documenting in Revit is the inability to place rooms exactly in the centre of the room. Ideally Revit would automatically snap to the centroid when in the process of placing a room, instead of just aligning to adjacent elements. Since the reference cross hairs of the room are turned off by default, initially the position of the
Everyday Dynamo: Practical uses for BIM managers
Paul Wintour & Scott Crichton, BVN

Room may seem irrelevant. However, when attempting to 'tag all not tagged', Revit will place tags centred to the room's location. This means that drawings can appear messy and illegible. To solve this problem, BVN has developed two custom Dynamo nodes, both of which are available via the BVN Dynamo Package.

Dynamo script to reposition rooms and room tags

The 'Room.CentreLocation' node moves all rooms within the project to the centroid of the room. If you have an odd shaped room and the centroid is going to fall outside of the room boundary, the node will keep the room in its current position. Under the hood, the script uses the LunchBox Room Element Collector to collect all rooms within the project. The outlines of the rooms are then generated using polylines. If there are internal room bounding elements which create multiple outlines, the script will take the one with the longest length. Typically, this will be the outermost outline but this is not fail safe. The centroid of the polylines is then calculated and a containment test performed to test whether the new location (centroid) is contained within the room. If true, the rooms are moved to their new location with the rest remaining as-is. The script is based on the node developed by Modelical.

Inside the 'Room.CentreLocation' custom node
Repositioned rooms and room tags

Once all the room have been repositioned it is possible to move any existing room tags to the new room location point. While it is possible to use the ‘tag all not tagged’ command within Revit, you may have some rooms that either don’t need tags or which require a different tag family. By using Dynamo and referencing the tags already placed in the project, users have greater control over the outcome.

The 'RoomTag.MoveToRoomLocation' node is based on the script by Modelical. However, the benefit of using the BVN package of Modelical's is that the BVN node will automatically select all the room tags in the project. The problem with having to manually select which tags to update is as follows. Firstly, using the 'Select model elements' node in Dynamo you need to draw a window to reference Revit elements. It is not possible to have a pre-selection or use Ctrl to select multiple elements. The alternative therefore is to visually isolate the room tags for easy selection. However, it is not possible to isolate solely the room tag category. For room tags to be visible, rooms must also be visible. Therefore, having the script automatically collect the room tags in preferable. Note that the script will search for any tags that contain the term ‘Room Tag’ in the family name. If you name your families something different, you'll need to modify the node as necessary.
2.3 Renumbering rooms

Often, we find ourselves faced with many rooms on a floor plan that must be numbered sequentially based on their layout. This type of data is not easily manipulated within a tabular format of an Excel spreadsheet nor a Revit schedule. We need to see the spatial configuration of rooms in a plan to understand how the renumbering needs to be applied.

Sequential room numbering
To solve this problem, we can use the 'Room. RenumberByModelCurve' custom node as part of the BVN package. Users are required to draw a (model) spline through the rooms to be numbered. Only rooms that intersect the curve will be processed. This allows for batch processing which is often desirable. The number sequence precision can then be set. For example, a precision on 1 will result in 0, 1, 2 etc., whereas a precision of 2 will result in 00, 01, 02, etc. An optional prefix, such as the level number, can also be set. There is also the option to exclude certain room names which cannot avoid being intersected by the model curve, such as circulation.

Dynamo renumbering room script

To ensure accuracy, ensure the spline passes approximately through the centre of the room and that room elements are centred within the room boundaries. This is because behind the scenes, the script is collecting all rooms using the 'LunchBox Room Collector' node and extracting their location point. Using this point, the script evaluates the closest point to the curve. These points, and the associated rooms, are then sorted based on their curve parameter. A sequence of numbers is then created and converted to a string to avoid any precision issues. This string is then concatenated with the desired prefix. The model curve can then be deleted or retained for future modification.
2.4 Renumbering doors

One task which consumes much more time than it should, is the renumbering of doors based on the room in which it is located. Depending on your office’s naming conventions, a door generally is named something along the lines of, Element prefix - Room # - Sequence #, for example, D-101-01. Added complexity also comes into play when trying to define which room the door is located. Generally, a door will be numbered based on the ‘to’ room, unless that room is a corridor or an external area, in which case it reverts to the ‘from’ room.

Sample door numbering example

Since Revit 2014, when Autodesk introduced the room calculation point, it has been possible to schedule the ‘to’ and ‘from’ parameters in door families. This feature has its limitations and quirks as documented here. However, assuming the family has been built correctly, how can we automate the naming convention?

One possible answer is to use BIMlink to export the schedule into Excel. Using a series of conditional statements, we could semi-automate the process. But for this to work you need to be a bit of an Excel wizard and it’s not that user friendly. Conversely, we can create a simple Dynamo script which fully automates the process, is easy to use, and eliminates the need for either Excel or BIMlink.
To do this you can use the 'Door.RenumberByRoom' node found in the BVN package. Users have the option to enter in a prefix, such as 'D', and a separator, such as '-'. Depending on your offices' naming conventions, the custom node can be edited and re-ordered as required. The script reads all doors in the project and returns the ‘to/from’ room number per door. By default, doors are renumbered based on the ‘to’ room. If however there is no ‘to’ room, such as an external door, it will revert to the ‘from’ room. If there is neither a ‘to’ or ‘from’ room, that is, there is a missing room, the doors will be excluded from the renumbering process. Once this is known, all doors within the same room are sequentially numbered in an anti-clockwise direction.
2.5 Purge unused filters and view templates

Within Revit, there are certain elements which cannot be easily purged. For example, the out-of-the-box 'Purge Unused' command (Manage > Settings > Purged Unused) is unable to delete filters, view templates or line styles. Even with the aid of Ideate Explorer we are unable to select them and therefore, unable to delete them.

Revit’s limitation of purging certain elements

While it's possible to delete these items manually, this isn't always a straightforward process. When deleting filters, Revit will indiscriminately delete the filter without forewarning you that the filter is currently in use. This means that there is no way to differentiate between those filters are in use and those which aren’t.
When deleting view templates, Revit will at least warn you that the view template it is in use and give you the option to choose a replacement view template. But this is a very labour intensive task to go through each view template, one-by-one, see if it is in use, and if not, delete.

To help solve this problem, BVN has published a custom node as part of the BVN Dynamo Packages. The 'Filter.GetUnused' custom node collects all the unused filters in the project so that it can be used in conjunction with either the 'Tool.Eraser' node from the SteamNodes Dynamo package or the 'Delete Elements' node from the Archi-lab Dynamo package.
**Dynamo purge unused filters script**

To purge unused view templates, we can use Modelica's 'Delete Unused ViewTemplates' custom node.

**Dynamo purge unused view templates script**