BES463803

Dynamo
+ Revit Systems Analysis
True BIM for HVAC

Sean Fruin
Sigma AEC Solutions

Majd Makhlouf
Building Information Researchers and Developers

Learning Objectives

- Discover how to use the Revit's Systems Analysis tools.
- Learn how to streamline the process of creating a Revit analytical model with Dynamo.
- Learn how to run HVAC Sizing and Energy Analysis directly inside Revit.
- Learn how to connect 2D schematics to Revit 3D elements and analytical systems.

Description

A fair amount of work and settings must be addressed to use Revit Systems Analysis effectively and correctly. Customizing the workflows requires an understanding of all the moving parts as well as some coding. This class will help you navigate Revit Systems Analysis features and show the potential for MEP's powerful computational workflows.

Speaker(s)

Sean Fruin is a Mechanical Engineer (EIT), design technologist, and innovator who has an intense fascination with automation and the exploration of computational design solutions for the AEC industry. He has had the opportunity to learn many aspects of the design industry, having worked in manufacturing, MEP designing, and General Contracting. Sean started Sigma AEC Solutions to live his dream, having the opportunity to explore and implement the latest technologies to improve efficiency and increase quality in the AEC industry.

Majd Makhlouf is a Mechanical Engineer and Design Technologist, with a Master of Science in Mechanical Engineering. He is an Autodesk® Revit Certified Professional and a member of the Autodesk® Developer Network. In January 2020, he founded Building Information Researchers and Developers OÜ, a software development company based in Estonia and providing services for the AEC sector worldwide. He specializes in BIM Management, Autodesk® Revit, and AutoCAD Add-in development, both public and custom-developed, Forge web and cloud-based apps, Dynamo Zero Touch Node Packs, and mobile VR/AR applications.

Contents

Building Information Researchers and Developers 1
Learning Objectives 1
Description 1
Speaker(s) 1
Intro 3
Revit MEP Building Block 4
BIM 4
Revit 4
Shared Parameters 5
Revit MEP Families 5
Revit MEP Systems 6
Revit Drafting Views 6
Rooms 6
Space 7
HVAC Zones 7
Revit Energy Analysis 8
Analytical Spaces 8
Analytical Surfaces 9
Analytical Systems 10
System Zones 10
Revit Systems Analysis 11
gbXML 11
Open Studio 11
Energy Plus 12
Revit API 12
Dynamo 12
Generative Design for Revit 13
Data Connections for Customization and Automation 13
Customizing Revit Element Connections 13
Customizing System Analysis 14
Customizing OpenStudio Workflows 14
Customizing OpenStudio Measures 101 15
Overwriting Measure Assumption 16
Measure Unit Conversions 17
Connecting A Revit Parameter To EnergyPlus Simulations 17
Customized Measure Example - Fan Static to EnergyPlus 18
Automation Toolbox 19
Data Exploration 20
Dynamo Geometry 21
Driving Revit with Dynamo 24
Exchanging Data Between Elements 27
Dynamo Custom Nodes 27
Clustering Algorithms 32
Graph Theory 33
Computational Geometry 34
Genetic Algorithms 35
Fuzzy Logic 36
Dynamo UIs 36
Automating MEP Tasks 38
Air Balance Table Coordination 38
System Configurator 39
Systems Builder Analytical System 42
Systems Builder 3D 44
Systems Builder 2D 47
Pressure Drop Calculations 48
Integrated Systems HVAC Systems 49
Project Template 49
Project Setup and Load Calculations 51
Rapid System Configurations 52
Connected 2D, 3D, and Analytical Systems 53
Conclusion 54

Due to the continued exponential improvement in computing power and other advancements, data-driven computer science has reshaped almost every industry. Fueled by frustration with the status quo and enabled by the ease of entry into coding using graphical programming languages, the AEC industry has only just begun to experience a shift.

Autodesk® loves showcasing Design Automation, Generative Design, and Machine Learning. These popular topics and their associated buzz words continuously show up on social media, at conferences, and in graduate papers. As early adopters start to incorporate data-driven workflows into their processes, the proof of concept is affirmed. However, implementing these workflows at scale is complicated, leaving firms confused about how best to automate their teams.

The AEC industry has seen the emergence of companies working hard to monetize design automation with simpleand intuitive tools. Yet, these solutions are doomed to fall short of being a one-stop-shop for fully automating large portions of the design process. It is incredibly challenging to code all the necessary detail to ensure compliance with countless building codes that govern the industry. As algorithms become more specialized due to varied building types, geographical location, and various governing bodies, the tool's market gets smaller, and the program becomes more complicated. For instance, the algorithms used to design an HVAC system for a San Francisco high-rise will fail at meeting the requirements for a school in New York. With an estimated 1,300 different building codes governing the US' design logic alone, firms will be doing themselves a disservice waiting for a third-party solution that is a perfect fit for their design problems.

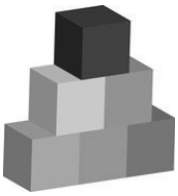
That brings us to the love-hate relationship with Autodesk's suite of AEC software – it is *the* go-to tool for the AEC industry. Its success is due to its ability to provide complete design and documentation workflows across a construction project's lifecycle, interoperability between all required software, and access to cutting edge generative design capabilities. Having a monopoly on the market, however, allows it to continuously increase the costs of its subscription services.

Revit, BIM software from Autodesk®, is a wonderful tool but has limitations, including an exceptionally tedious user interface. For example, new features are still designed and executed with the "point-and-click" mentality of its predecessor, AutoCAD. Virtually all training videos and documentation are demonstrated using manual clicks and dragging workflows. Second, the out-of-the-box program will only do basic workflows. Interoperability between tools falls short, and the supplied example content islacking, having only a few simple generative design workflows and basic families. Third, as more and more features are added, Revit is becoming progressively more fragmentedand scattered, and less user-friendly. All of this is leaving designers confused and frustrated, inevitably defaulting back to old habits resulting in industry-wide standards where data is entered manually. PDF sheets are the project's database, and the starting point is virtually a blank canvas.

The solution to this problem is to get under the hood of the program. With some programming proficiencies, the mysteries of Revit's promising workflows are unlocked and Autodesk's plan is exposed. Their business model is not to provide their customers with an easy button, but rather to provide them with frameworks that allow them to indeed "make anything." This class will use Dynamo and Revit's new Systems Analysis Framework to investigate and integrate various automated workflows. We will learn how to connect data from different frameworks and get under Revit's hood to explore different HVAC System configurations, run energy models, and document the design. To start the journey with some context, we will cover the types of frameworks and how they will be used.

Revit MEP Building Block

Before we can start to build customized workflows, it is critical to understand the program's ins and outs. Revit has an overwhelming number of tools built on a variety of different frameworks. This section outlines all the different Revit frameworks, components, and concepts utilized to build the energy modeling workflows. Note that there have been many changes over the last few releases of Revit around energy analysis workflows and the explanations in this course are representative of Revit 2021.



BIM

The first key concept in the integrated workflow is Building Information Modeling or BIM. The idea behind BIM is to combine a database with a scaled 3D model of building components. The database stores attribute (parameters) about each modeled building component (elements) and populates the documentation (schedules, sheets). The idea behind a true MEP Building Information Modeling workflow is an endlessly evolving process aimed to cut down on the production time of designing and documenting the model. The model's data can drive engineering calculations and turn design logic into algorithms, thereby automating tasks one by one until no tasks are left. This end goal is a stretch but is the northern light guiding principle. To achieve this goal, several different frameworks in Revit, all with unique data structures, exclusive functions, and tools, need to come together. The data required and produced from the array of tools needs to be connected like a gigantic puzzle. First, all the different puzzle pieces need to be identified. Second, the data needs to be stored in a coherent structure this is where Revit shines.

Key Notes

- The only "I" in BIM refers to the information for which we must establish a need.
- A true BIM process is a team effort where access to the team's data is critical.

Revit

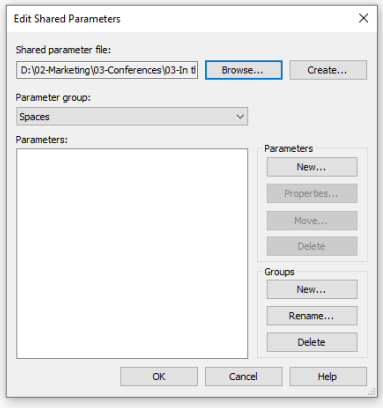
Revit is numerous BIM tools wrapped into a software package, but at its core, Revit is a database. The information in the database populates documentation like views, schedules, and sheets. As 3D Building elements like walls, floors, and HVAC equipment get placed in a model, parameters are applied that incorporate information about the building components. Also, when a new element is placed, a unique ID is applied to a parameter. This data structure is essentially a relational database. A relational database is a set of tables containing data in predefined categories, e.g., walls, doors, pipes, or mechanical equipment. Each table includes one or more data parameters in columns, e.g., fire rating, size, flow rate, or manufacturer. Each row contains a unique instance of data for the categories defined by the columns. Relationships between items of different tables can be

established using Primary and Foreign Keys. A Foreign Key is a field in a relational table that links back to another table's Primary Key column. The key to getting all data from the elements from different Revit MEP frameworks is establishing relationships between elements using the element IDs as Primary and Foreign Keys. Shared Parameters are added to the model to create the data table and establish connections.

Key Notes

- A unique Element ID is automatically assigned to any element placed in the Revit model.
- Some Elements and Revit tools are natively connected, while many are not.

Shared Parameters

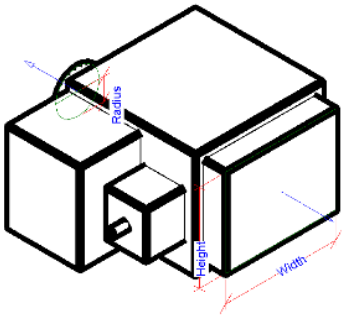


Shared parameters are customized vessels for storing information in the Revit Projects database and differ from other parameters. They can be used across several documents as project and family parameters. They can report their values across these documents; hence the term "shared." A master list of shared parameters is created and stored in a text file separate from a Revit project. These data containers have data types used to constrain their purpose and will be used in unit-specific calculations. Critical components to standardizing shared parameters are utilizing a clear naming structure and assigning the correct data types. Once a Shared Parameter is created, it can be added to most categories in a Revit project or Revit families. Project documentation relies on Shared Parameters since these values can be reported in a Tag and Schedule in the document the family is loaded in.

Key Notes

- There are other types of parameters, but Shared Parameters provide the best flexibility.
- We cheat sometimes and make the parameters unitless, so the input is not constrained.

Revit MEP Families

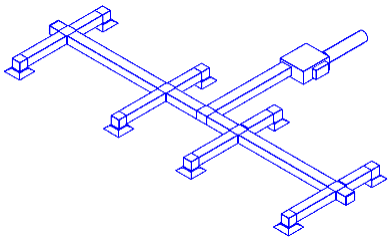


Families are a mixture of geometry, parameters, and formulas used to represent parametric building components. They represent the standard components, both geometrically and parametrically. Moreover, Families are the building block of the database and facilitate data storing, sorting, and informative Schedule and Tag to display information. These graphical representations of building components are organized hierarchically to create the different data tables within the Revit project's database. The Family is made in the Revit Family environment and is saved as a separate file. After a Family is created or edited, it gets loaded into Revit projects. MEP Families provide a critical foundation for creating MEP Systems. MEP connectors are added to the geometry, which develops connections to other elements to form mechanical systems.

Key Notes

- Default Revit Families are not adequate if you want to get the most out of Revit.
- Family location points, connectors, and parameters need to be coordinated with design automation algorithms.

Revit MEP Systems

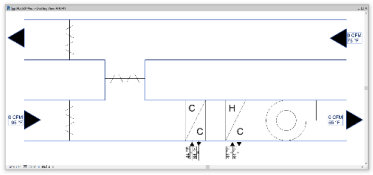


MEP systems define logical relationships between related MEP Families and let engineering data propagate through the elements. This data structure is known as a graph in computer science. A graph is made up of vertices connected by edges and provides an excellent way to model relationships between non-linear data like a Duct or Piping System. When an MEP system is created in Revit, a logical connection between Revit Families is built to conduct efficient analyses using traversal algorithms. Creating MEP systems requires that the families have MEP Connectors that are appropriately assigned. Connectors get assigned both a System Classification and a direction. Families with the same connectors can be paired together. These relations can be viewed with the System Browser display.

Key Notes

- Connectors cannot be hidden or set parametrically on a Family.
- No physical connection between elements needs to be made to form a system.

Revit Drafting Views

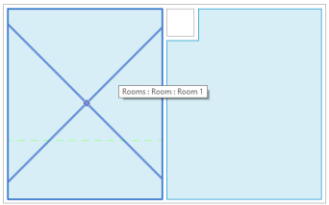


Drafting Views are used to create 2D views that are unconnected to the 3D model. These views are often utilized for details and schematics. Generic Annotations with parameters can be placed in drafting view and combined with Detail Lines to create project-required diagrams, details, and customized schedules. Drafting views are usually a crude collection of geometric elements, containing no information. However, with the correct use of Dynamo/API, even these crude details can report information and relate or control correspondent model elements.

Key Notes

- Parameters can be added to Generic Annotations.
- Generic Annotations can be used like AutoCAD Blocks in Drafting Views.

Rooms

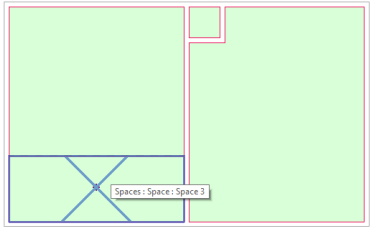


Rooms are used to store information about the subdivision of areas within a building model to identify usage and occupancy. When a Room is placed at a point, the area covered is automatically constrained by room-bounding elements like walls, floors, roofs, and ceilings. Rooms can be automatically created for enclosed regions and be divided by using room separation lines. For Systems Analysis simulation, Rooms are rather useless. However, they add the ability to store data that completely aligns with an Architecture model.

Key Notes

- Room Bounding is a parameter found in the Elements and/or Link model.
- Rooms are not affected by space separation lines.

Space

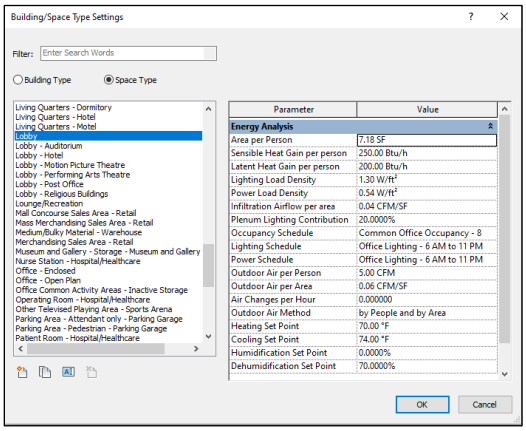


Spaces are like Rooms but are used to store data for HVAC energy analysis. Designers have complete control over the area and volume of the Spaces. Like Rooms, a Space's boundaries stop at room-bounding components such as walls, floors, ceilings, and roofs. Multiple Spaces can be added to one Room by using Space separation lines. Additionally, the "Limit Offset" parameter can be used to control the vertical extent of the volume.

Spaces are intelligent enough to know what Room it resides in. If there is no Room assigned to a Space, the Space is set to "Unoccupied." If the Space is assigned to a room, Spaces automatically records the Room Name and Number and sets it to "Occupied."

As of Revit 2021, it is no longer necessary to place Spaces in all the interior openings of a model like chases. When the energy model is created, an analytical Space will be made in all voids. It is still best practice to place Spaces in all openings of a floor plan view, such as shafts and chases, even if a Room is not present so that the Space Type can be assigned to identify the Space as "Unoccupied."

The Space Type for new Spaces defaults to Building Type. In my opinion, all Spaces should be assigned a Space Type. The Building/Space Type Settings menu significantly impacts the energy model because it controls heating and cooling load parameters like ventilation, occupancy, operation schedules, and internal heat gains. Revit 2021 has added new parameters to define thermal set points for Building Types and Space Types. Building and Space Types are used for energy calculations, providing more flexibility and clarity for end-users. These values get assigned to Space, and Analytical Spaces for energy simulation.



Key Notes

- New Set Point Parameters have been added to the Space Type in Revit 2021.
- New Parameter values cannot be added to the Space Type as of Revit 2021.
- Building Types work the same way as Space Types and is set by default to new Spaces.
- Many built-in parameters do not have any impact on the Energy Model.

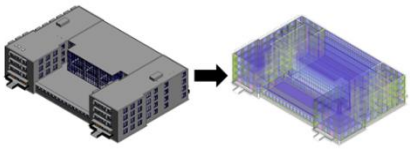
HVAC Zones

In prior versions of the program, Zones were used for heating and cooling load calculations. Spaces needed to be assigned to Zones (spaces designated to the Default zone would not be included in the heating and cooling loads calculations). As of Revit, 2021-this is no longer the case with Revit Systems Analysis; Zones are irrelevant and can be ignored.

Key Notes

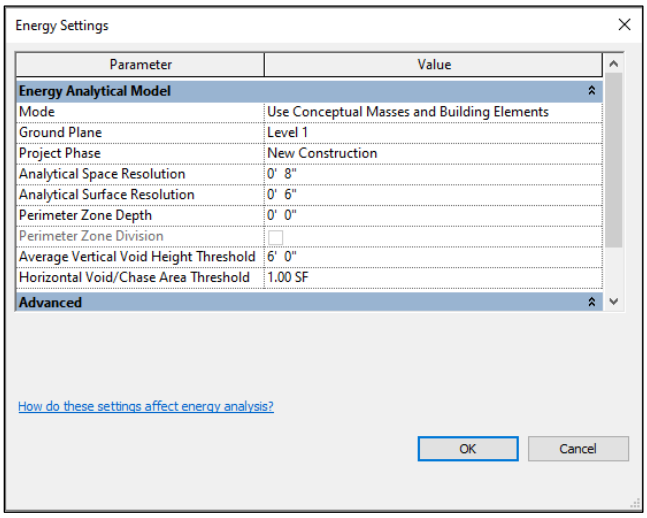
- This is not relevant in Systems Analysis in Revit 2021.

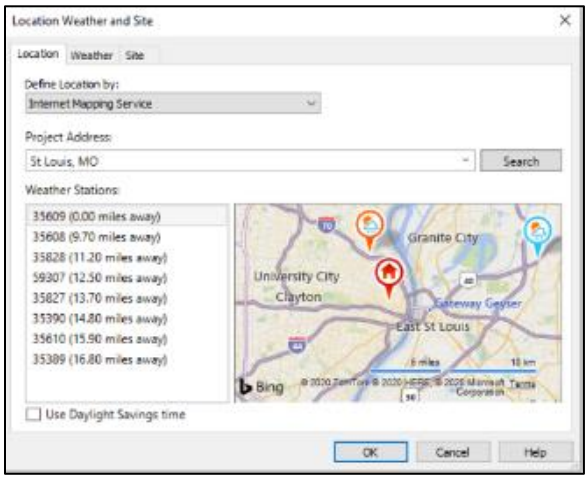
Revit Energy Analysis



Energy Analysis framework creates a gbXML schema for energy modeling directly from a Revit model at any detail level using building components or masses. The analytical energy model created from this framework is comprised of four main parts – climate data, Analytical Spaces, Analytical Surfaces, and Analytical Systems. The analytical model is automatically generated only when a user commands Revit to create or replace one and does not dynamically update with model changes. The geometry of the Analytical Energy Model is produced by intersecting room-bounding building components of the model with a 3D grid called a voxel. The voxel resolution is a critical component for finding an optimum balance between energy model accuracy and processing time. The Analytical Surfaces are created by the intersections of the building element's faces and the voxel. The voids that are bounded by the surfaces create the Analytical Spaces. The energy model can be examined in Revit with a 3D view and schedules.

Energy Settings offer designers various options to automatically create a complete and accurate energy model that meets the design intent and the control to dial in the voxel resolutions for the best results. Additionally, the designer can specify additional information such as thermal properties for the Analytical Surface and Spaces.





Climate data information is collected by assigning a geographic location to the building. The Location tool allows the site to be set using the nearest major city's street address or its latitude and longitude. The Weather tab offers a way to verify the Cooling and Heating Design Temperatures and the Clearness Number for the project location. These settings can also be overwritten in this tab.

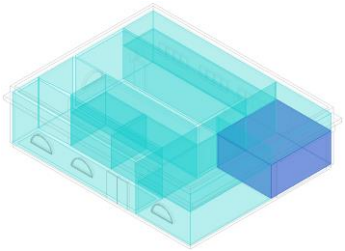
Analytical Spaces

Analytical Spaces are volumes that experience heat transfer through building components and specified internal heat gains. Revit generates Analytical Spaces within enclosed volumes in the model and automatically populates data from Building Type, Rooms, or Spaces. In the Energy settings,

Spaces or Rooms can be specified. Spaces are a better choice since they contain Energy Analysis parameters for better flexibility in calculating energy usage. By default, Analytical Spaces use the Energy Analysis settings from the assigned Building Type. If a Space location point resides in the Analytical Space's enclosed volumes, the Analytical Space parameters inherit the Space Energy Analysis parameter data.

<Analytical Spaces Schedule>

A	B	C	D	E	F	G	H	I	J	K
Room Name	Space Type	Heating Set Point	Area	Volume	Air Changes per Hour	Latent Heat Gain per person	Outdoor Airflow	Peak Latent Cooling Load	Peak Cooling Load	Peak Heating Load
Space 5	<Building>	70 °F	298 SF	2180.7	0	200	21	Not Computed	Not Computed	Not Computed
Space 6	<Building>	70 °F	444 SF	5267.2	0	200	35	Not Computed	Not Computed	Not Computed
Space 8	<Building>	70 °F	88 SF	906.7	0	200	7	Not Computed	Not Computed	Not Computed
Space 7	<Building>	70 °F	200 SF	2556.9	0	200	18	Not Computed	Not Computed	Not Computed
Space 1	<Building>	70 °F	115 SF	799.2	0	200	8	Not Computed	Not Computed	Not Computed
Space 2	<Building>	70 °F	145 SF	1020.5	0	200	10	Not Computed	Not Computed	Not Computed
Space 4	<Building>	70 °F	264 SF	1963.2	0	200	18	Not Computed	Not Computed	Not Computed
Space 3	<Building>	70 °F	357 SF	2648.9	0	200	25	Not Computed	Not Computed	Not Computed
Analytical Space 1	<Building>	70 °F	264 SF	856.5	0	0	0	Not Computed	Not Computed	Not Computed
Analytical Space 3	<Building>	70 °F	357 SF	1156.6	0	0	0	Not Computed	Not Computed	Not Computed
Analytical Space 2	<Building>	70 °F	298 SF	960.9	0	0	0	Not Computed	Not Computed	Not Computed
Analytical Space 4	<Building>	70 °F	260 SF	825.5	0	0	0	Not Computed	Not Computed	Not Computed



Key Notes

- Analytical Spaces are automatically generated when an Energy model is created.
- Analytical Spaces report an intersecting Space in the “Room Name” Parameters.

Analytical Surfaces

Analytical Surfaces are automatically created from walls, floors, roofs, and other building components from either a native or a linked model when the Energy model is created. The surfaces get categorized according to location and function. Thermal properties for these surfaces can be assigned in a variety of ways. Conceptual Types and Schematic Types offer a valid approximation applied quickly to the entire building or surface-bound individual Analytical Spaces. It is possible to model specific surfaces or assemblies' thermal properties and assign materials to the elements. Analysis of Detailed Elements does not require that all elements or materials in the model contain Thermal Properties. Revit gives Conceptual or Schematic Types to elements without detailed thermal information, allowing users to mix and match elements with known Thermal Properties and schematic elements.

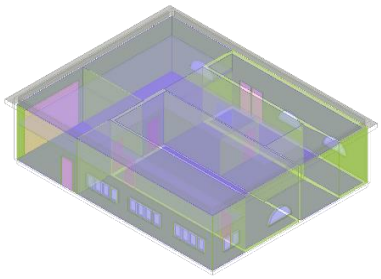
Key Notes

<Analytical Surfaces>

A	B	C	D	E	F
Surface Type	Area	Thermal Resistance (R)	Thermal Mass	Heat Transfer Coefficient (U)	#
Ceiling		2.59	1.82	0.39	10
		1.54	9.45	0.65	5
Exterior Wall		11.66	1.74	0.09	16
Interior Wall		1.75	1.04	0.57	24
Raised Floor		22.16	2.70	0.05	7
Slab on Grade		21.03	34.08	0.05	8

<Analytical Glass>

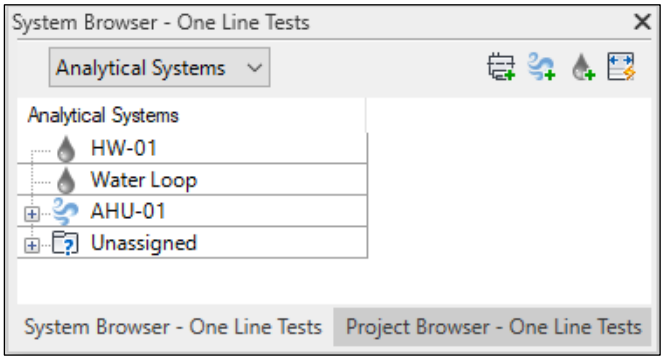
A	B	C	D	E
Opening Type	Area	Solar Heat Gain Coefficient	Visual Light Transmittance	Heat Transfer Coefficient (U)
Operable Window	6	0.76	0.81	0.503
Operable Window	6	0.76	0.81	0.503
Operable Window	18	0.76	0.81	0.503
Operable Window	18	0.76	0.81	0.503
Operable Window	6	0.76	0.81	0.503
Operable Window	18	0.76	0.81	0.503
Operable Window	14	0.76	0.81	0.503



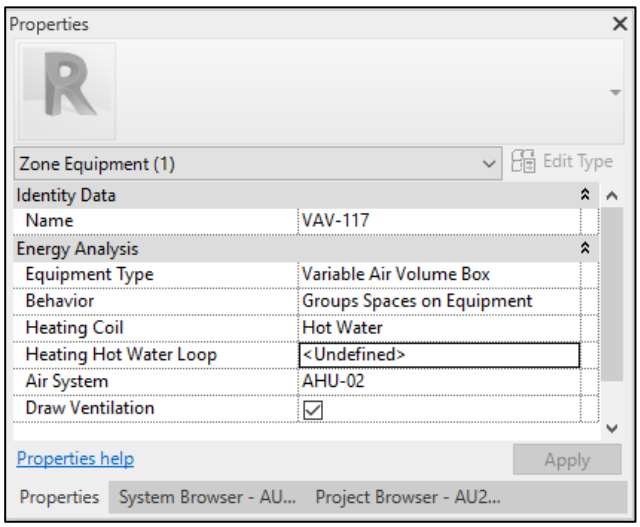
- Analytical Surface automatically generates when an Energy model is created.
- Analytical Surface thermal properties can be specified to come from an element or be assigned by the user with the energy settings.

Analytical Systems

Analytical Systems were added to Revit 2020.1 and are used to add HVAC Systems to the Energy Models gbXML in order to inform load calculations, equipment sizing, and system simulation. Analytical Systems get applied to different building regions and are configured from Zone Equipment, Air Systems, and Water Loops that get added to the model using the tools in the System Browser. Zone Equipment, Air Systems, and Water Loops are added to the project by clicking their respective icons in the Systems Browser. The System Browser also displays the *hierarchical relationships of the analytical system components and identifies if an element is not ad*equately defined with a warning icon.



All the different system components first get assigned a Type and Name in the Properties palette. Depending on the type selected, additional options appear in the Properties palette to specify more equipment specific characteristics. This Analytical Systems workflow provides the ability to define countless different system configurations.

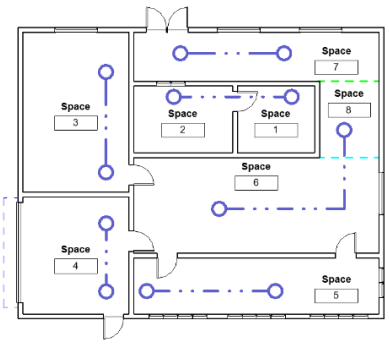


Key Notes

- Equipment is assigned default performance parameters.

System Zones

System Zones are graphic elements (line, enclosed shapes, and/or 3D forms) used to assign Analytical Spaces to the deified Zone Equipment. The tool allows designers to define a thermal zone in a model quickly.

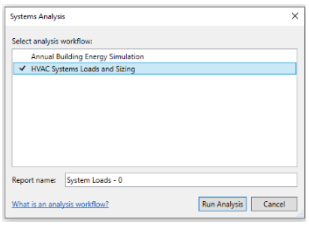


Analytical Space that intersects the System Zone geometry is automatically assigned to that System Zone when an Energy Model is created. The System Zones then are manually set to Zone Equipment in the Properties Palette.

Key Notes

- Analytical Space gets assigned Systems Zones when an energy model is created.
- Unassigned Analytical Space can manually be moved to a Systems zone in the System Browser.

Revit Systems Analysis



System Analysis connects Revit to EnergyPlus™ for building energy modeling. Once any energy model is created, a Systems Analysis simulation can be run using a Systems Analysis Workflow. A Systems Analysis Workflow is essentially an OpenStudio Workflow that runs within Revit. The analysis runs in a background process and takes a considerable amount of time for complex models. The output is a time-stamped HTML report that can be viewed in Revit and populates peak heating and cooling for Analytical Spaces.

Revit 2021 provides two Systems Analysis Workflows out of the box: Annual Building Energy Simulation and HVAC Systems Loads and Sizing. These workflows provide a default list of assumed plant equipment configurations with fixed associated performance values. The framework allows designers to create customized workflows that can be added to the Revit project. Customization options are endless: default values can be overwritten, simulation variables can be connected to Revit parameters, and reports can be set to pull more data from the simulation. Customizing the workflows involve editing OpenStudio® measures.

Key Notes

- The out of the box workflows have built-in assumptions.
- System Analysis bridges the gap between Revit and EnergyPlus™

gbXML

gbXML is an industry-standard XML schema language format for building energy modeling interoperability. It allows disparate 3D building information models (BIM) and architectural/engineering analysis software to share information.

Key Notes

- OpenStudio is the only credited gbxml.
- The gbXML is simply a long-structured text file.

Open Studio

OpenStudio® is an open-source platform used to create and manage EnergyPlus™ models and serves as the bridge between Revit and EnergyPlus™. The conversion process includes two main components – Measures and Workflows. OpenStudio® Workflows are configuration files that describe what to run, run it, and find the dependent files. Measures are sets of commands that efficiently renovate the energy model. There are three types of measures – model measures, energy plus measures, and reporting measures.

When we run a Revit's Systems Analysis workflow, OpenStudio® will automatically run the background's Energy Model simulation. First, the Revit project's assigned weather file, OpenStudio® workflow file, OpenStudio® Measures, and the project's gbXML are automatically gathered and bundled into an EnergyPlus™ file. Next, EnergyPlus™ runs the simulation, followed by OpenStudio® running the reporting Measures. The output is an HTML report which is populated Analytical Space Parameters.

Key Notes

- The OpenStudio® files are Ruby programming language files and are saved outside of Revit.

Energy Plus

EnergyPlus™ is a powerful, open-source, and cross-platform whole building energy simulation program with a broad array of features and capabilities for building performance and energy analysis. The program is Ashrea 140 compliant and works under the hood of other trusted Building Energy Modeling software like Trane® Trace 3D Plus. The program reads an input text file known as an IDF and writes outputs to text files. Designers do not typically interact with these files. Instead, OpenStudio® is used to manipulate the inputs and outputs to create different workflows.

Key Notes

- Noah Pflaum once declard, "EnergyPlus is a beast of a program."
- Designers typically do not interface with EnergyPlus.

Revit API

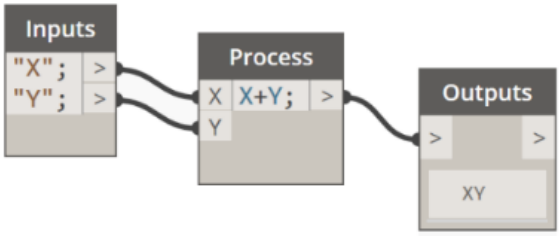
Revit API (Revit's application programming interface) is a collection of operations with a simple description of what users can use to communicate with the software. This allows software developers a way to interact with Revit in order to build external applications and customize workflows. Most of the Revit API is universal and applies to all Revit products. However, Revit MEP has some specific features and is continuously being updated with new functionality being added. Learning the Revit API is exceedingly challenging if you do not have experience in writing code.

Key Notes

- A good starting point to learn how to develop tools with Revit's API is to build macros.

Dynamo

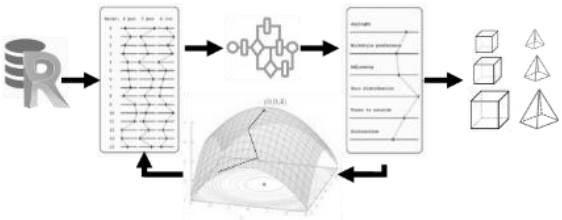
Dynamo is a Visual Programming language that connects to Autodesk® Revit. Dynamo is built on top of the Revit API to make Revit API development easier and accessible to the non-developer end user. It has the power to access internal/external data and model geometry. In some cases, there is not a native Dynamo node for a particular function. Python or C# can be used to build customized nodes. All this data can be connected to define relationships, analyze geometrical relationships, and execute a sequence of actions that create algorithms. Algorithms can be used for a wide array of applications, from processing data, setting Revit Parameters, and placing Revit Families throughout the model toexecuting Revit commands.



Key Notes

- Dynamo is relatively easy to learn compared to other programming languages.
- Dynamo is open source with all the code exposed for further customization.

Generative Design for Revit



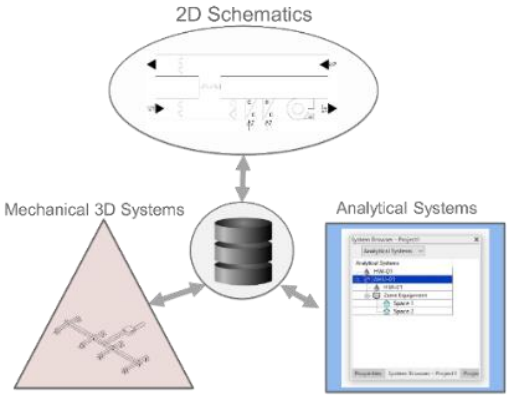
Generative Design for Revit enables designers to create, test, and evaluate design options rapidly. Generative Design studies require Dynamo scripts that define inputs andoutputs to execute algorithms multiple times with unique input combinations. Designs create parametric mathematical systems that encompass their design logic with Dynamo. The computer then rapidly generates design options and can search for optimization using genetic algorithms. Inputs are either constants or variables which can be various data types like cost, flow rates, angles, aspect ratio, and area, to name a few. The feature can be accessed directly in Revit 2021 via the AEC Collection, or a version can also be accessed directly from Dynamo.

Key Notes

- Project Fractal > Project Refinery > Generative Design for Revit
- Generative Design has some tools to help run Dynamo scripts faster.

One of Sigma AEC's goals is to automate the HVAC system design processes. It is our view that HVAC design automation software should take a building model, along with a few user inputs, to produce a range of HVAC systems configurations for energy modeling. We believe this can now be achieved by using all the different frameworks and tools available in the Autodesk® AEC collection. The examples outlined in the next two sections outline concepts needed to start automating energy modeling. The integrated workflow section outlines a highly automated and integrated workflow that brings these concepts together to streamline the process of designing an HVAC system, proving Revit can be much more than a documentation tool. The underlying idea is to establish a smooth and efficient workflow where the designer is guided by the program to quickly get a project off the ground and eliminate wasteful tasks. The first step is to build a project template and algorithms that work together.

Data Connections for Customization and Automation



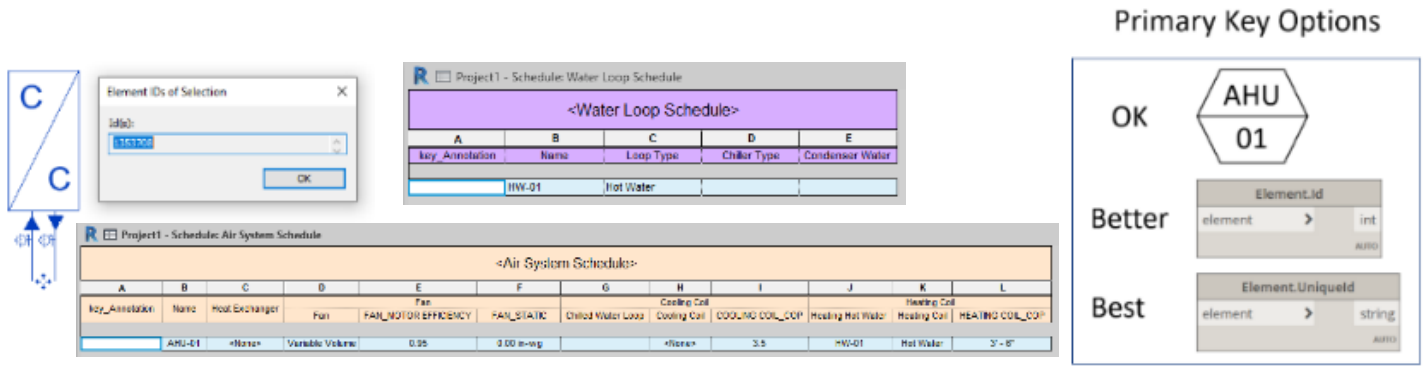
A critical objective is to establish a single source of truth by creating a connection between the 2D schematics, the 3D model, and the inputs and outputs from the EnergyPlus simulations. With these connections being established, data is entered in one location that can propagate to the other required destinations, eliminating potential data drops and data redundancy issues. This chapter will cover the techniques used to connect the different MEP data and customize the Revit System Analysis Workflows.

Customizing Revit Element Connections

Out of the box, there is often no way to connect and exchange data between different elements in the BIM model. For example, Generic Annotations used to create diagrams in Revit drafting views cannot be connected to a corresponding element of the 3D model. Automation can bridge this gap if the

relational database structure is established between the different elements. Once connections are established with parameters, data can be exchanged with Dynamo.

To set this up, a Shared Parameter is added to Categories to store the Unique i.d. of an Element in a different category. The example below shows this set up where the parameter called *KEY_Annotation* has been added to the Air Systems and Mechanical Equipment categories to connect to a Generic Annotation.



Key Notes

- Information exchange needs to be automated using a Dynamo script, add-in, or Macro.
- The Id does not need to be an element Id; any unique number or string will work as a key.

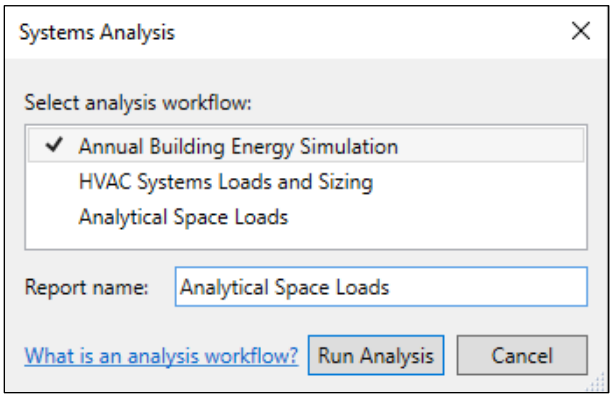
Customizing System Analysis

Customizing a Revit Systems Analysis Workflow involves adding parameters to analytical elements and manipulating OpenStudio files outside of the familiar Revit environment. The OpenStudio files are comprised of two main types of files Workflows and Measures. Workflows are executed through Revit bycollecting the gbXML and location data from the project and configuring that data with the predefined OpenStudio Measure in order to set up and run an energy simulation with EnergyPlus. This data exchange between Revit and the EnergyPlus simulations is dependent on files that can be found at the following file path.
C:\Program Files\NREL\OpenStudio CLI For Revit 2021.

Customizing OpenStudio Workflows

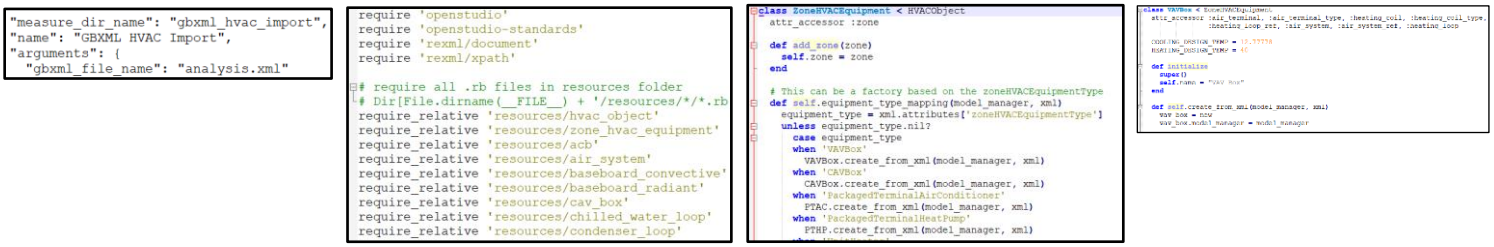
The OpenStudio Workflow files are executed through the Systems Analysis dialog box. When Revit 2021 is installed, the Annual Building Energy Simulation and HVAC Systems Loads and Sizing Systems Analysis Workflows are included by default and located at the file path below.

C:\Program Files\NREL\OpenStudio CLI For Revit 2021\workflows

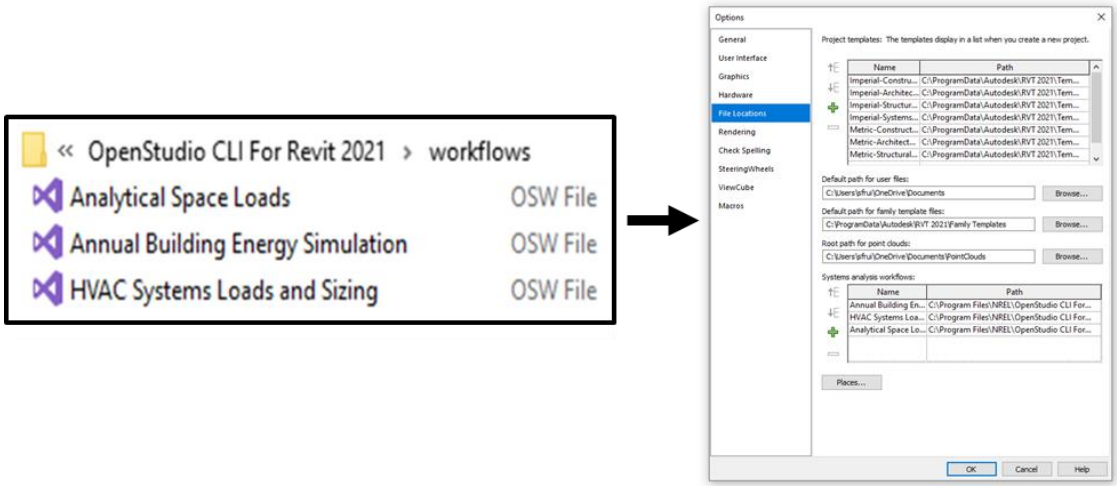


In simple terms, an OpenStudio workflow sets up what measures are executed and in what order. Customizing workflows involves adding and removing Measures from the file with a text editor. The list of Measures' for the Workflow can be found in the “steps” section of the code. The diagram below shows how measures are packed into workflow like Russian dolls.

Open Studio Workflow ← Open Studio Measure ← Open Studio Measure ← Open Studio Measure



Once a customized Workflow is ready, it needs to be added to Revit by loading the new file through the Options, which ca be found at the bottom of the file applications menu.



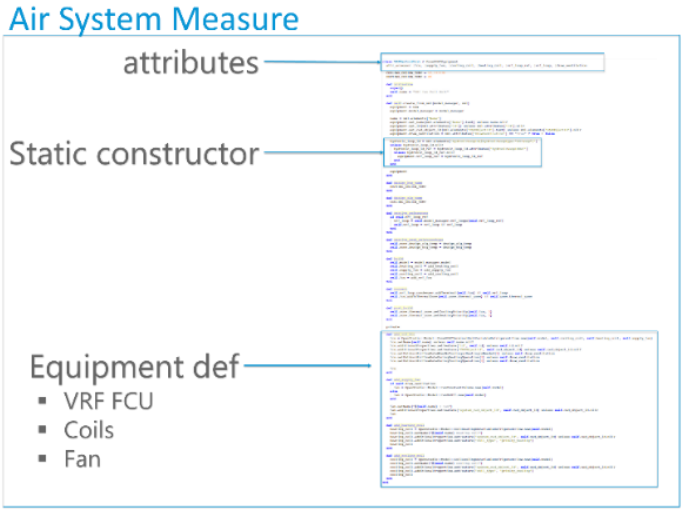
Key Notes

- OpenStudio Workflows are the JSON files that describe a simulation workflow.
- Execution order: 1st Model Measures > 2nd EnergyPlus Measures > 3rd Reporting Measures.

Customizing OpenStudio Measures 101

There are three types of OpenStudio Measures. Model Measure uses OpenStudio’s API to manipulate the OpenStudio model. EnergyPlus Measures manage the EnergyPlus IDF files. Reporting Measures run after the simulation and are used to export data from the simulation in different forms. Most editing for setting up customized System Analysis workflows is done by manipulating the OpenStudio Measures that define the Zone Equipment, Air Systems, and Water Loops.

The measures for Water Loops, Air Systems, and ZoneEquipment are remarkably similar and define equipment definitions for energy simulations. Out of the box, these measures are designed to have a minimal amount of user inputs. Therefore, assumptions have been applied for variables like leaving air temperatures, COP, and head pressure. These assumptions can be changed by editing the measure. All the HVAC equipment measures are installed in the file path below when Revit 2021 is installed.
C:\Program Files\NREL\OpenStudio CLI For Revit 2021\measures\gbxml_hvac_import\resources



Overwriting Measure Assumption

A quick and straightforward way to change most of the default assumptions is to open the measure and directly change the values. The fixed assumptions can be found right after the attributes section or in the Equipment definition. The orange text represents the default values or constant values.

```
PREHEAT_DESIGN_TEMP = 4
COOLING_DESIGN_TEMP = 12.777778
HEATING_DESIGN_TEMP = 12.777778
DOAS_COOLING_DESIGN_TEMP = 12.777778
DOAS_HEATING_DESIGN_TEMP = 15.555556
COOLING_DESIGN_HUMIDITY_RATIO = 0.0085
HEATING_DESIGN_HUMIDITY_RATIO = 0.005
DEFUALT_FAN_STATIC = 995
```

```
def add_supply_fan
  fan = nil

  if self.supply_fan_type == "VariableVolume"
    fan = OpenStudio::Model::FanVariableVolume.new(self.model)
    fan.setFanPowerMinimumFlowFraction(0.3)
  elsif self.supply_fan_type == "ConstantVolume"
    fan = OpenStudio::Model::FanConstantVolume.new(self.model)
  end

  if fan
    fan.setPressureRise(995)
    fan.setFanTotalEfficiency(0.60)
    fan.setMotorEfficiency(0.94)
    fan.setName(self.name + " Supply Fan") unless self.name.nil?
    fan.additionalProperties.setFeature('system_cad_object_id',
  end

  fan
end
```

In some cases, default values may not be exposed in the measure. In these situations, the method needs to be added to the equipment definition. For instance, a cooling coil for a PTAC unit has a default COP (Nominal Efficiency) of 3, but there is no line in the *add_cooling_coil* definition to set this value. Therefore, a new line needs to be added to override the default value.

The example below demonstrates this by running two simulations, with and without the COP being set. Both ways are given a result in the results table (Nominal Efficiency), with the not Specified COP being set to a default value of 3.00.

No Specified COP

```
def add_cooling_coil
  cooling_coil = OpenStudio::Model::CoilCoolingDXSingleSpeed.new(self.model)
  cooling_coil.setName("#{self.name} + Cooling Coil")
  cooling_coil.additionalProperties.setFeature('system_cad_object_id', self.cad_object_id) unless self.cad_object_id.nil?
  cooling_coil.additionalProperties.setFeature('coil_type', 'primary_cooling')
  cooling_coil
end
```

	Type	Design Coil Load [Btu/h]	Nominal Total Capacity [Btu/h]	Nominal Sensible Capacity [Btu/h]	Nominal Latent Capacity [Btu/h]	Nominal Sensible Heat Ratio	Nominal Efficiency [Btuh/Btuh]	Nominal Coil UA Value [Btu/h-F]	Nominal Coil Surface Area [ft2]
ZONE EQUIPMENT-1 + COOLING COIL	Coil:Cooling:DX:SingleSpeed		6400.20	5059.40	1340.80	0.79	3.00		

Specified COP

```
def add_cooling_coil
  cooling_coil = OpenStudio::Model::CoilCoolingDXSingleSpeed.new(self.model)
  cooling_coil.setName("#{self.name} + Cooling Coil")
  cooling_coil.additionalProperties.setFeature('system_cad_object_id', self.cad_object_id) unless self.cad_object_id.nil?
  cooling_coil.additionalProperties.setFeature('coil_type', 'primary_cooling')
  cooling_coil.setRatedCOP(3.6)
  cooling_coil
end
```

	Type	Design Coil Load [Btu/h]	Nominal Total Capacity [Btu/h]	Nominal Sensible Capacity [Btu/h]	Nominal Latent Capacity [Btu/h]	Nominal Sensible Heat Ratio	Nominal Efficiency [Btuh/Btuh]	Nominal Coil UA Value [Btu/h-F]	Nominal Coil Surface Area [ft2]
ZONE EQUIPMENT-1 + COOLING COIL	Coil:Cooling:DX:SingleSpeed		6400.20	5059.40	1340.80	0.79	3.60		

Measure Unit Conversions

OpenStudio uses metric units, so converting is necessary for us stubborn Americans. The method below converts values from the first unit to the second unit, as specified by two strings. The table shows the syntax for other standard unit conversions.

```
temp_c = OpenStudio.convert(temp_f,"F","C")
if temp_c.is_initialized
  temp_c = temp_c.get
end
```

Unit Type	IP	SI
Temperature	F	C
Delta Temperature	R	K
Flow	ft^3/min	m^3/hr
Pressure	inH_{2}O	Pa

Connecting A Revit Parameter To EnergyPlus Simulations

With a few more steps, OpenStudio Measures can be connected to Revit. First, shared Parameters need to be added to Zone Equipment, Air Systems, and Water Loops categories. The gbXML automatically detects and records the Parameters' names and values for these three categories when an Energy Model is created. Second, the OpenStudio Measures need to be edited to acknowledge and transfer the value to an OpenStudio object. The example below shows this process for setting the coiling coil COP for the PTAC measure.

The first step is to add a new attribute for the parameter in the text file's attributes section.

```
class PTAC < ZoneHVACEquipment
  attr_accessor :ptac, :supply_fan, :cooling_coil, :heating_coil, :heating_coil_type, :heating_loop_ref, :heating_loop,
                :heating_inlet_water_temp, :heating_outlet_water_temp, :draw_ventilation, :cooling_coil_cop
```

Second, the static constructor needs to have a few lines added to read the added data present in the gbXML.

```
first_elem = REXML::XPath.first(xml, path="AnalysisParameter[Name[text()='Cooling Coil COP']]")
if first_elem
  equipment.cooling_coil_cop = first_elem.elements['ParameterValue'].text.to_f
end
```

Third, the Equipment definitions need to be adjusted to read and set the new parameter.

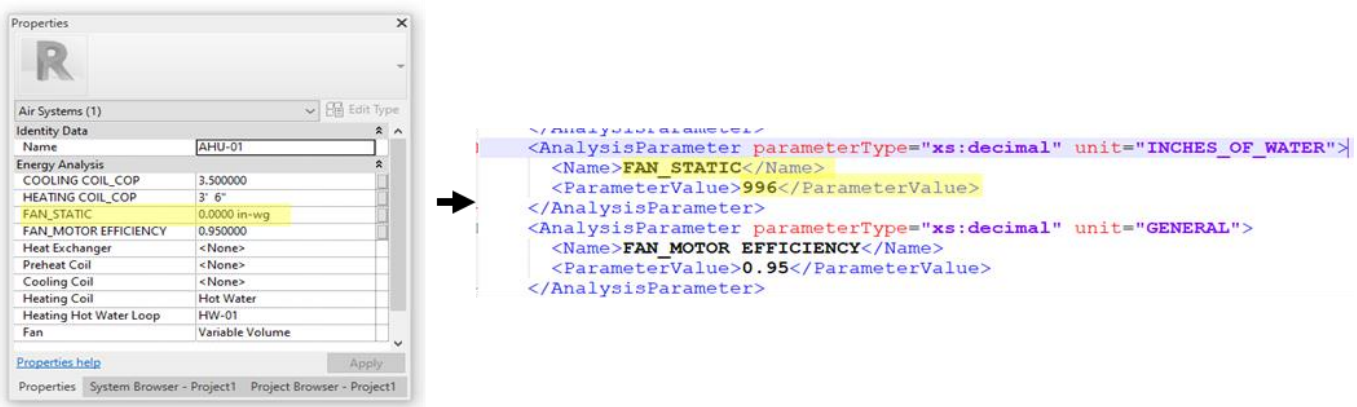
```
cooling_coil.setRatedCOP(self.cooling_coil_cop) if self.cooling_coil_cop
```

Key Notes

- OpenStudio Measures use metric units.
- OpenStudio API has many methods for creating, altering, and managing EnergyPlus models using Measures.

Customized Measure Example - Fan Static to EnergyPlus

This example demonstrates setting the Fan Static of an Air System for the EnergyPlus simulation from within Revit. First, add a Shared Parameter to the Air Systems category called *FAN_STATIC*. This data then gets added to the gbXML when an Energy Model is created.



The rest of the steps are all done by editing the OpenStuideo Air System Measure using Notepad++. At the top of the file, an attribute called fan_static is added to the end of the list.

```
class AirSystem < HVACObject
  attr_accessor :air_loop_hvac, :supply_fan, :heating_coil, :cooling_coil, :preheat_coil, :oa_system, :heat_exchanger,
                :spm, :supply_fan_type, :heating_coil_type, :heating_loop_ref, :heating_loop, :cooling_coil_type,
                :cooling_loop_ref, :cooling_loop, :preheat_coil_type, :preheat_loop_ref, :preheat_loop,
                :heat_exchanger_type, :is_doas, :zone_hvac_equipment, :preheat_spm, :fan_static
```

Second, the static constructor needs to be edited to read the Fan Static (yellow) parameter from the gbXML. A conversion is added to change the units from in water to pascals (blue). An if statement is added to set a default value if the FAN_STATIC is null or empty (green). This last step stops the simulation from failing by assigning the default value if the Revit's Parameter is empty or does not exist.

```
first_elem = REXML::XPath.first(xml, path="AnalysisParameter[Name[text()='FAN_STATIC']]")
  if first_elem
    air_loop.fan_static = OpenStudio.convert(first_elem.elements['ParameterValue'].text.to_f, "inH_{2}O","Pa").get
  end
      if air_loop.fan_static.nil?
      air_loop.fan_static = 995
  end
  air_loop
end
```

Third, the OpenStudio fan object needs to the setPressureRise to fan_static (blue).

```
def add_supply_fan
  fan = nil

  if self.supply_fan_type == "VariableVolume"
    fan = OpenStudio::Model::FanVariableVolume.new(self.model)
    fan.setFanPowerMinimumFlowFraction(0.3)
  elsif self.supply_fan_type == "ConstantVolume"
    fan = OpenStudio::Model::FanConstantVolume.new(self.model)
  end

  if fan
    fan.setPressureRise(self.fan_static)
    fan.setFanTotalEfficiency(0.60)
    fan.setMotorEfficiency(0.94)
    fan.setName(self.name + " Supply Fan") unless self.name.nil?
    fan.additionalProperties.setFeature('system_cad_object_id', self.cad_object_id) unless self.name.nil?
  end

  fan
```

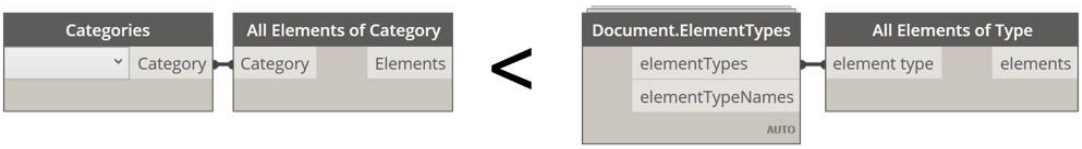
By establishing connections between EnergyPlus simulation parameters and elements from different Revit categories, data can be freely exchanged, allowing systematic and integrated workflows between different frameworks.

Automation Toolbox

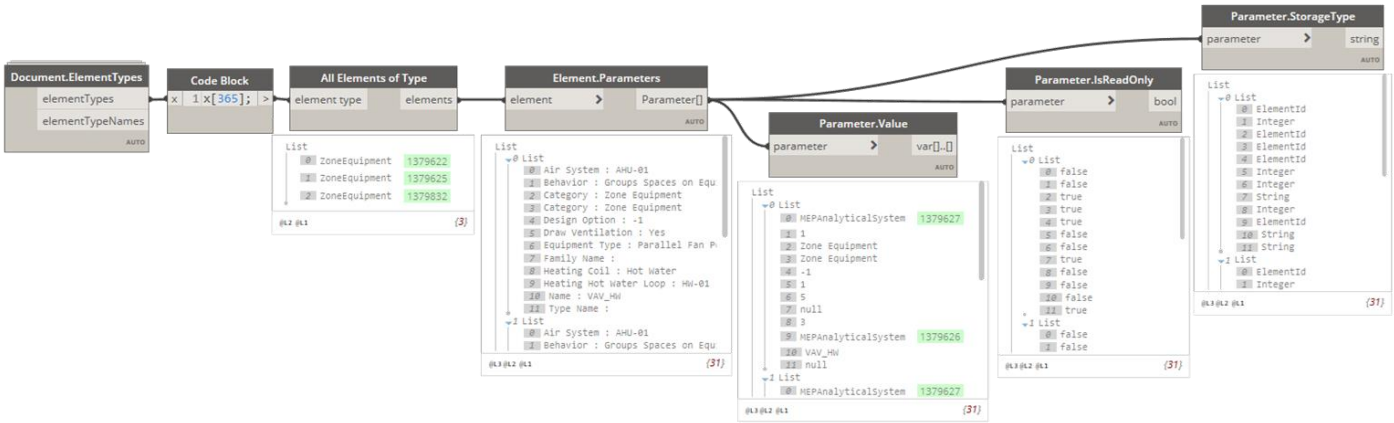
Complex automated workflows are merely a combination of API functions, established algorithms, and data. Collecting a set of these algorithmic building blocks is vital for efficiently piecing together advanced automated workflows. This chapter introduces essential Dynamo functionality, vital customized Nodes, and a few elegant general-purpose algorithmic methods while highlighting when and where these different techniques can be used. The first step is to get familiar with what data is available and access it to fuel algorithms.

Data Exploration

The data within Revit models is what sets Revit apart from its predecessors. A simple way to start exploring the Revit API and the data stored in a project's database is with Dynamo. All the different API classes within the project can be obtained using the *Document.ElementTypes* node from the Clockwork package. This is a powerful approach that offers a way to access more elements compared to the *All Elements of Categories node*. For example, Projects Information cannot be accessed with the *Categories* node but can with the *Document.ElementTypes* node.



Once elements are retrieved from the Element Type, their parameters can be accessed with the *ElementParameters* node. The parameters can then be investigated using various nodes to determine their storage type (if it is read-only) and the parameter values.



In the example above, the ZoneEquipment elements are retrieved, and the parameters are inspected. Two interesting findings can be seen in this example. First, the Air System's storage type is an ElementId, meaning an element is used to set this parameter. Second, the *EquipmentType* Parameter has a storage type as an Integer, yet reports a string in the *parameter.Value* node. Quite a few of the built-in parameters are stored as Integers yet displayed in the Revit UI as strings. This is known as an Enum. In computer programming, an Enum type is a particular data type that enables a variable to be a set of predefined constants. The variable must be equal to one of the values that has been predefined for it. We have found that the Revit API is fraught with inconsistencies in naming things such as this. To set Enum parameters accurately, the names need to be mapped out by poking around and creating a table like the one below for the Air System *Equipment Type* parameter.

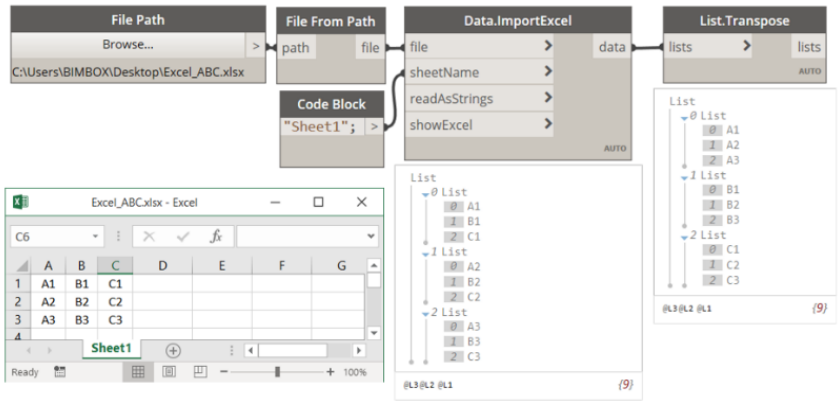
Revit GUI Value	Database Values
Parallel Fan Powered Box	5
Variable Air Volume Box	6
Unit Heater	12

Another useful investigation is to check the geometry of the different elements. Geometry is a crucial ingredient for engineering analysis and forming algorithms. Once an element is retrieved, there are four nodes we can use to get the element's geometry. These nodes will produce varying outputs based on the type of Revit element being accessed. The table and image below illustrate this. Note the error when using the *Element.Geometry* node on the EnergyAnalysisSurfaces.

Geometry Outputs

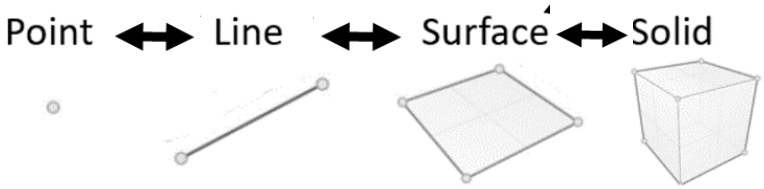
Node	Output Types
Element.Geometry (element → var[])	Solids, Lines
Element.GetLocation (element → Geometry)	Lines, Points
Element.Faces (element → Surface[])	Surfaces
Element.Solids (element → Solid[])	Solids

Data from other sources can be brought into the Dynamo environment as well. Excel is the most used tool for manipulating and managing data in the MEP engineering community. Below is an example of import data from Excel to be used in Dynamo. I like to say, "If you can get it in Excel, you can get it into Revit."



Dynamo Geometry

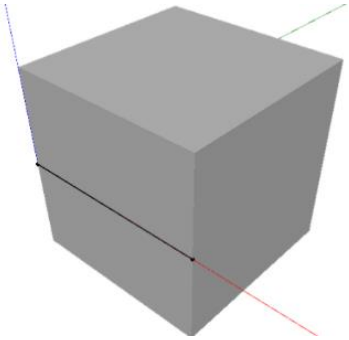
Geometry plays a crucial role in engineering design, and Dynamo is fantastic at creating and analyzing geometry. Dynamo also makes the process of extracting geometry from a Revit model effortless. This breakthrough has bridged the gap between the geometry in the engineering books and the geometry in Revit models, thereby making computation design a much easier process. Geometry can be broken down into a hierarchy. Understanding these fundamentals allows the transformation between different geometries in logical ways. The most straightforward and simple component of any geometry is a point. Points come together to make lines, lines come together to create surfaces, and surfaces form solids.



In the example below, we start with points and construct a solid. We begin by making two points and connecting them with a line using the *line.ByStartPointEndPoint* node. A vector is then created to make a new line by moving the original line with the *Geometry.Translation* node. These two curves are then used to make a surface using the *surface.ByLoft* node. This surface is then thickened with the *surface.Thicken* node. Note this is just one of many ways to go from points to solids.

Code Block

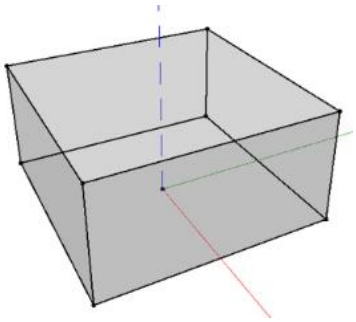
```
pnt_A = Point.ByCoordinates(0,0,0);
vector_A = Vector.ByCoordinates(1,0,0);
pnt_B = Point.Add(pnt_A, vector_A);
Line_A = Line.ByStartPointEndPoint(pnt_A,pnt_B);
vector_B = Vector.ByCoordinates(0,1,0);
Line_B = Geometry.Translate(Line_A, vector_B);
surface = Surface.ByLoft([Line_A,Line_B]);
solid = Surface.Thicken(surface, 1);
```



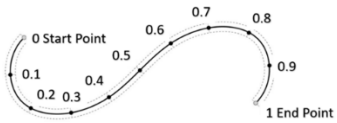
Likewise, we can go in the reverse order- from solids to points. In this next exercise, we will deconstruct a solid cube. To start, all the surfaces that make up the solid are obtained using the *PolySurface.BySolid* node. Then, the curves that make up each surface are derived using the *surface.PerimterCurves* node. Finally, we identify the starting points of each curve using the *Autodesk.Curve.StartPoint* node.

Code Block

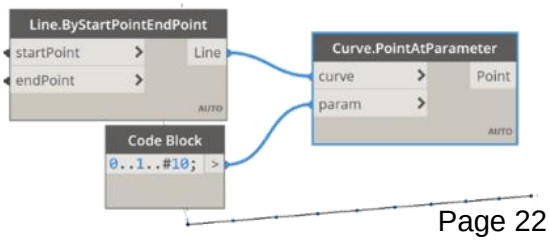
```
solid polySurface = PolySurface.BySolid(solid);
surfaces = PolySurface.Surfaces(polySurface);
curves = Surface.PerimeterCurves(surfaces);
points = Autodesk.Curve.StartPoint(curves);
```



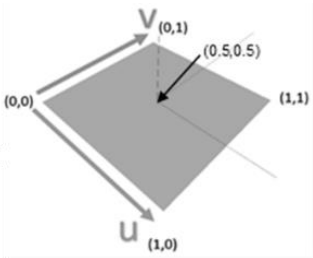
Curves and surfaces are frequently used as the base for algorithms and the subsequent geometry from the 3D model elements. Numerous geometric properties get combined to create geometric algorithms and turn Revit into a powerful design tool. Dynamo has nodes for extracting properties like position, orientation, and measurements.

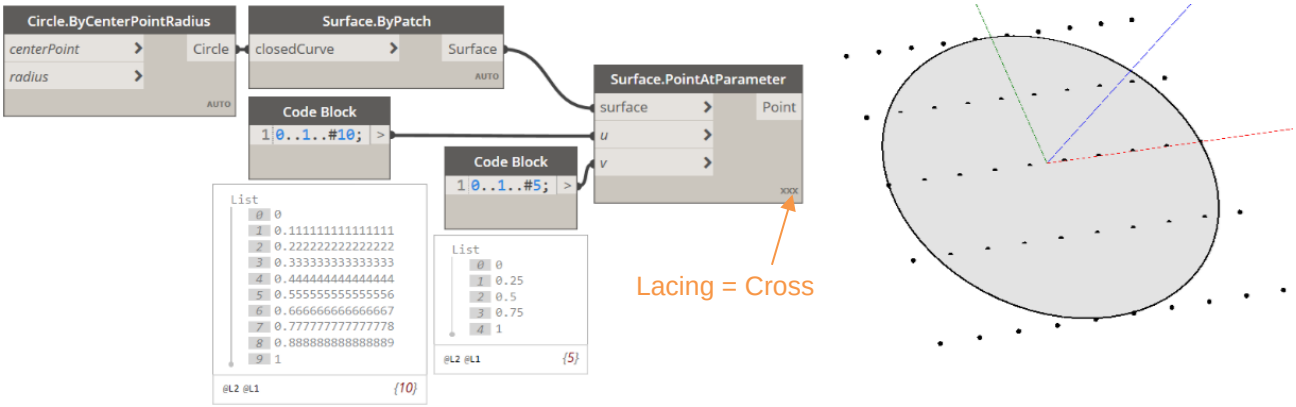


The first critical property to understand about curves and surfaces is the "parameter." The parameter describes a point on a line or surface. Lines can be deconstructed to a list of points using this concept. Points are retrieved at specified locations using parameter values. In general terms, the parameter represents a percent of the total length. For instance, the starting point of the line is at a parameter value of zero, and the endpoint is at a parameter value of one. A value between zero and one can be used to get points at any location along a line. Multiple points can be retrieved using a series of numbers between 0 and 1. In the example to the right, 10 points are identified along a line.

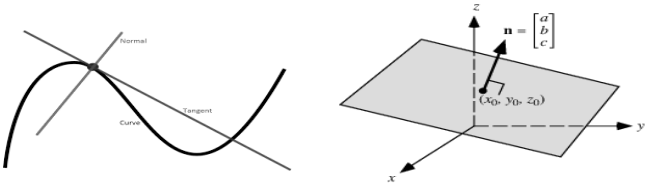


A point can be obtained on a surface at a specified set of UV parameters. Two variables are required since a surface is a function of two variables. For instance, take the *surface.PointAtParameter* node. The inputs' (*u,v)* axes are aligned to the X and Y axes of the global coordinate system and extend the axis's length aligned bounding rectangle. If the input to both *u* and *v* is equal to zero, then the point would be in the left corner. If they are both equal to one, then the point would be in the right corner. In the example below, 50 points form a grid covering the bounding rectangle of the circular surface.



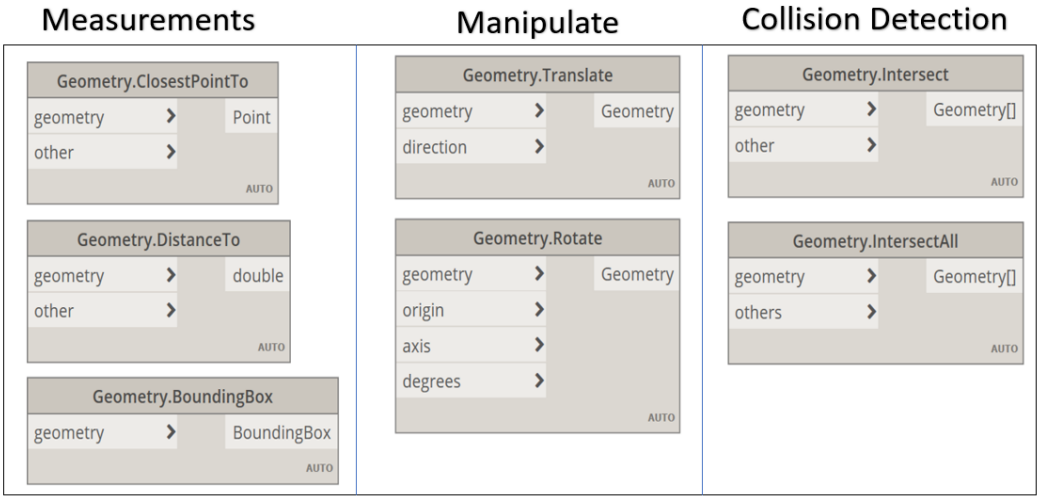


Once a location is established on a line or surface, the orientation can be evaluated at that point. Here, we will look at tangents and normal properties. Curves have tangent vectors and normal planes, whereas surfaces have normal vectors and tangent planes. The table below shows some useful nodes for properties for both curves and surfaces.

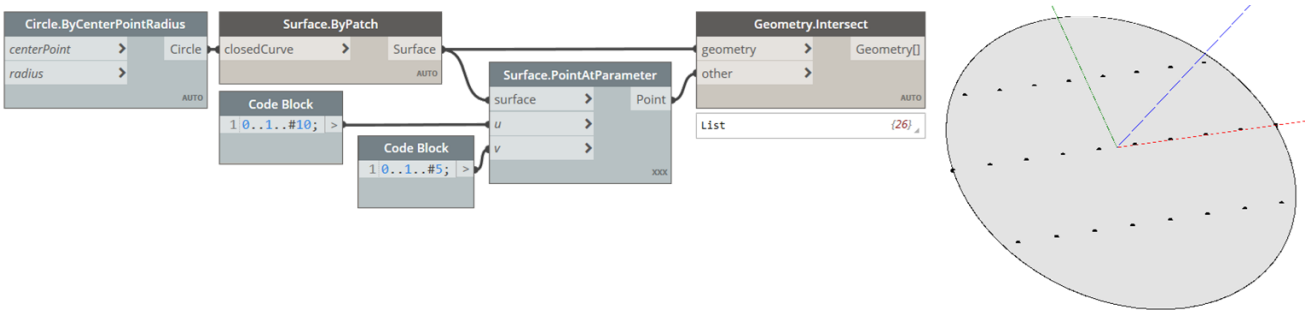


		Orientations	
Measure	Location	Normal	Tangent
Curves			
Curve.Normal curve → Vector AUTO	Curve.EndParameter curve → double AUTO Curve.PointAtParameter curve → Point param AUTO	Curve.NormalAtParameter curve → Vector param AUTO	Curve.TangentAtParameter curve → Vector param AUTO
Surfaces			
Surface.Perimeter surface → double AUTO Surface.Area surface → double AUTO	Surface.PointAtParameter surface → Point u v AUTO	Surface.NormalAtParameter surface → Vector u v AUTO	Surface.TangentAtUParameter surface → Vector u v AUTO Surface.TangentAtVParameter surface → Vector u v AUTO

There are also nodes called *geometry modifiers* that can be used to drive our algorithms. These nodes can be broken into three categories- measurements, manipulation and collision detection.

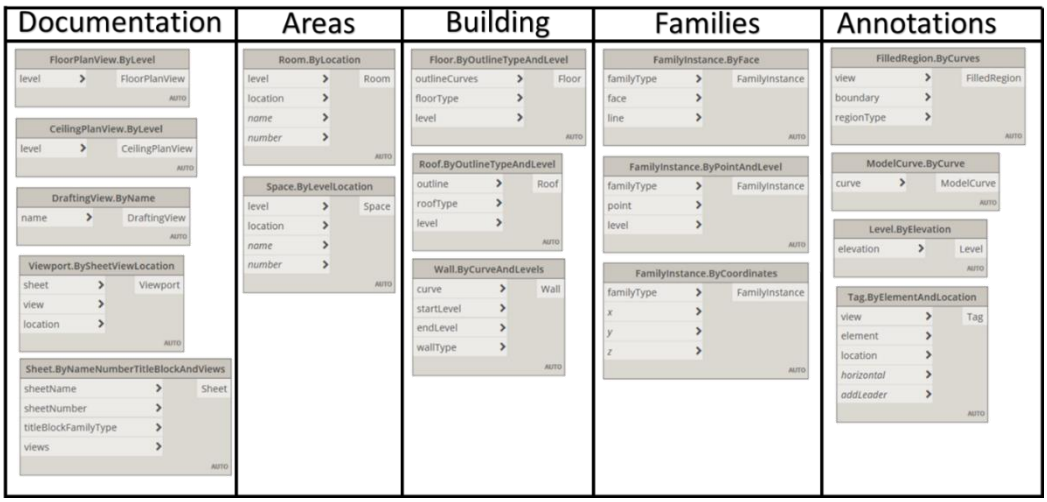


In the example below, points that do not intersect the circular surface get removed from the grid points list. The *Geometry.intersects* node only returns the intersecting points.



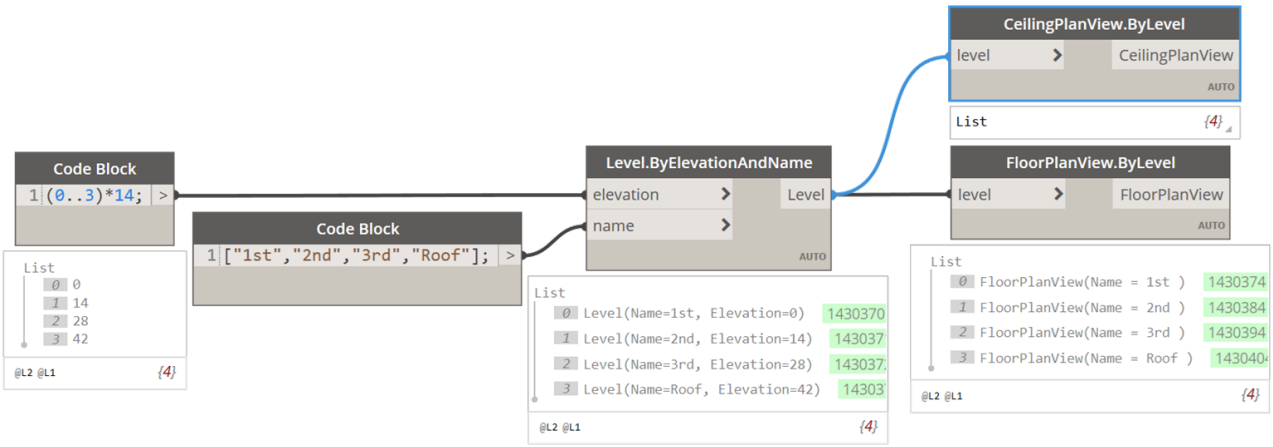
Driving Revit with Dynamo

Dynamos offers several ways to create all the different Revit elements needed in a project with complete parametric control. The Revit nodes in Dynamo offer the ability to generate documentation and place 3D elements. The table and examples below highlight the variety of objects that can be created.

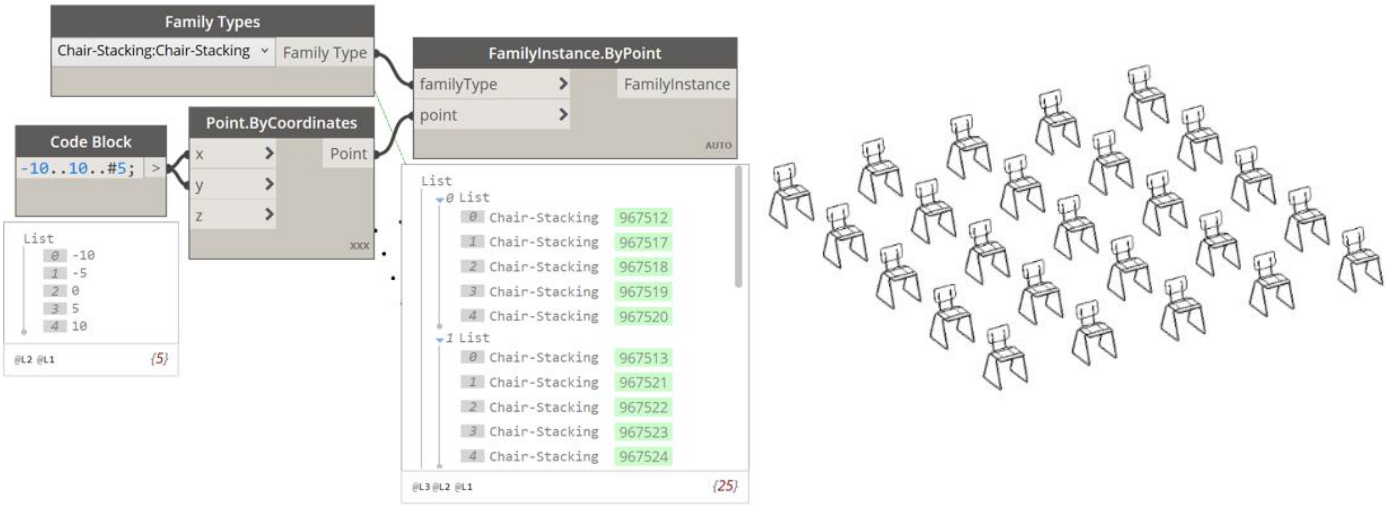


By linking up the nodes, a series of steps can be performed to achieve the end goal. Each node is one step in a series of instructions building off previous steps. The examples below demonstrate how to start parametrically driving Revit with Dynamo.

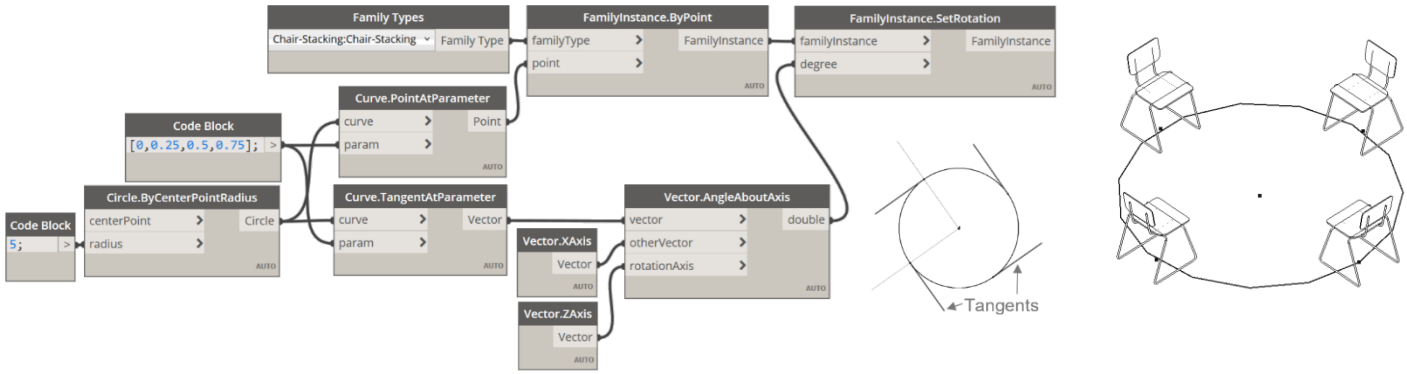
There is often a need to follow some order of operations —for example, a series of elevations and a list of corresponding level names can be used to create levels and create views after the levels have been created.



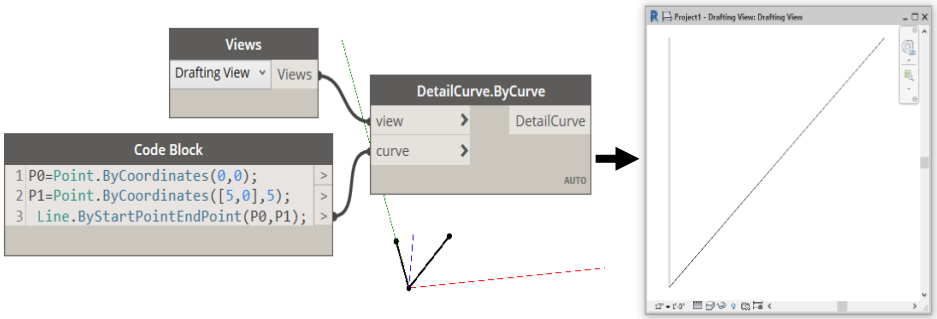
By combining Dynamo geometry techniques, several families can be parametrically placed in patterns. In this example, we construct a grid of points by feeding a sequence into the point's x and y components.*ByCoordinates* node. The series is formed using design script inside the Code Block. Note that the *Point.ByCoordinates* node is set to “cross lacing.” The selected family is then placed at these points using the *FamilyIntance.ByPoint* node.



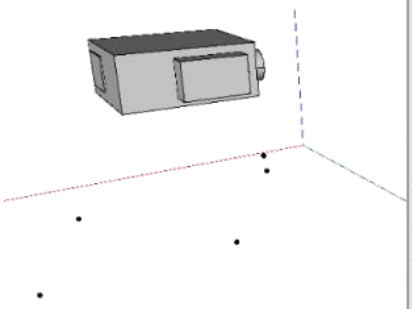
In the example belowRevit elements are placed along a circle and then rotated to face the circle's center. To do this, we first form a circle using the circle.ByCenterPointRadius node. Next, we use the curve.PointAtParameter node to retrieve points along the circle from the list of parameters from the Code Block. We then compute the tangent at the same parameters using the *curve.TangentAtParameter* node. After the families are placed at the parameter points, we can rotate them using the *FamilyInstance.SetRotation* node. The tangent vector is used with the *Vector.Angle.AboutAxis* node to find the needed rotation.



This example demonstrates using geometry created with Dynamo to draw a line in a drafting view. First, a line is created using two points, followed by that line being drawn in a Revit Drafting view using the *DetailCurve.ByCurve* node.



The example above examines placing a Revit Family at the average location of a set of random points. First, Dynamo creates a set of random points. Next, the average X Y components of the points are found and combined with a specified Z component. Then, a Revit family is obtained and placed in the model at the calculated point.



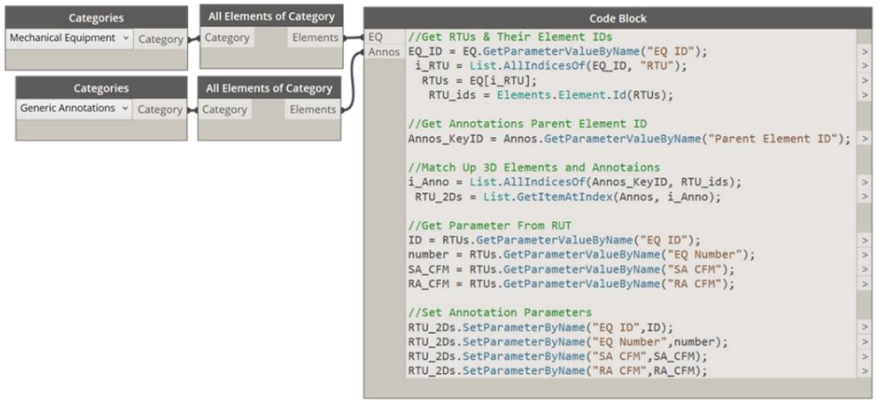
Code Block

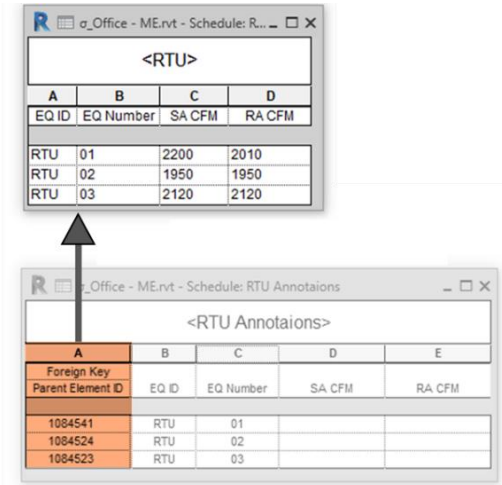
```
//Inputs
n=5;
z=4;

//Random Points
Xs = 10*Math.RandomList(n);
Ys = 10*Math.RandomList(n);

//Average Coordinates
pnts = Point.ByCoordinates(Xs,Ys);
Xaves = Math.Average(Xs);
Yaves = Math.Average(Ys);

//Get Revit Family
f="VAV Unit - Parallel Fan Powered";
t="10 inch Inlet";
FT = FamilyType.ByFamilyNameAndTypeName(f,t);

//Place Family
VAV = Revit.FamilyInstance.ByCoordinates(FT,Xaves,Yaves,z);
 Elements.Element.Geometry(VAV);
```

Exchanging Data Between Elements

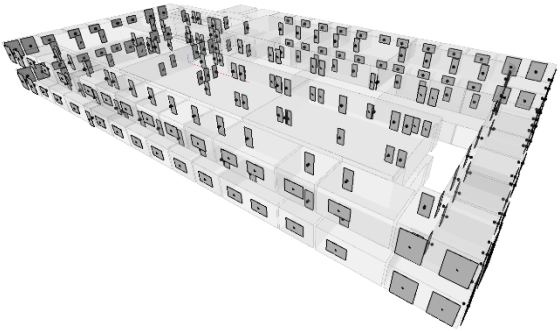
Often, data from different categories needs to be combined or exchanged in some manner. The illustrations below show how Dynamo can exchange data between elements.

The first example uses established keys mentioned in the previous chapter. The Dynamo script exchanges parameter values from Mechanical Equipment to set parameters in the corresponding Generic Annotation. The Element Id from the Mechanical Equipment is used as the Foreign Key in the Generic Annotation.





Another method for establishing connections between elements of different categories is to use geometric relationships. For instance, there is no way with Revit to match Analytical Surfaces to Spaces. This example translates the center point of the Analytical Opening into the model to find the corresponding Space.

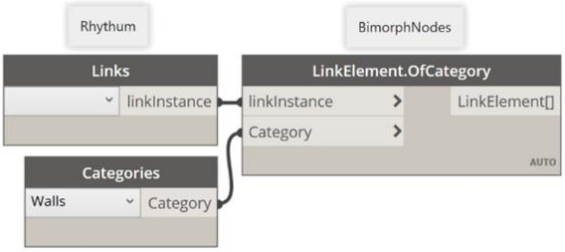


Dynamo Custom Nodes

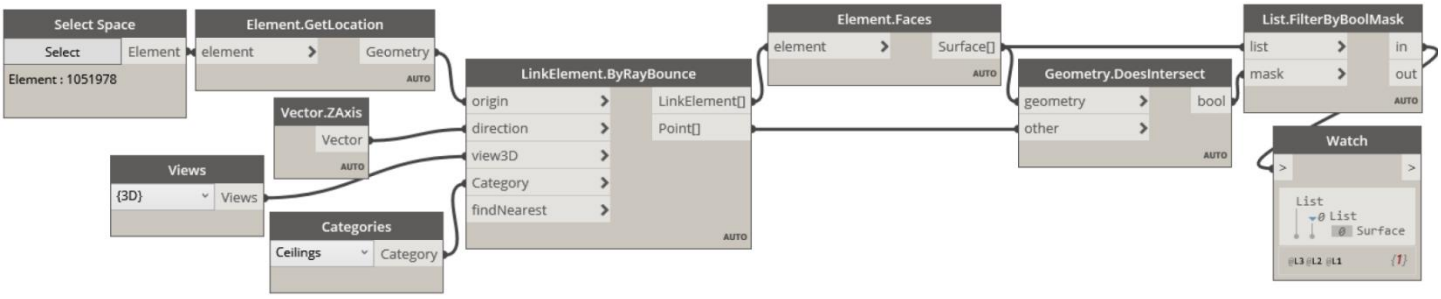
Dynamo native library does not have all the required functionality available in Revit's API but Dynamo custom node significantly expands the power of Dynamo. Custom nodes can be used to package groups of nodes into a single node that can be saved and deployed to others. Custom nodes can also be created with Python and C#. This allows access to the Revit API and functions from other libraries. Dynamo custom node handles many situations (iterative algorithms, large data sets) better than Dynamo.

The Dynamo Package Manager is a repository of customized nodes that have been contributed by power users and distributed to the Dynamo community. Solutions to common problems can often be found in the Package Manager and the Dynamo Forum. Dynamo would not be what it is today if this community and its contributing members did not exist. Below are examples of custom nodes that are required for later tasks.

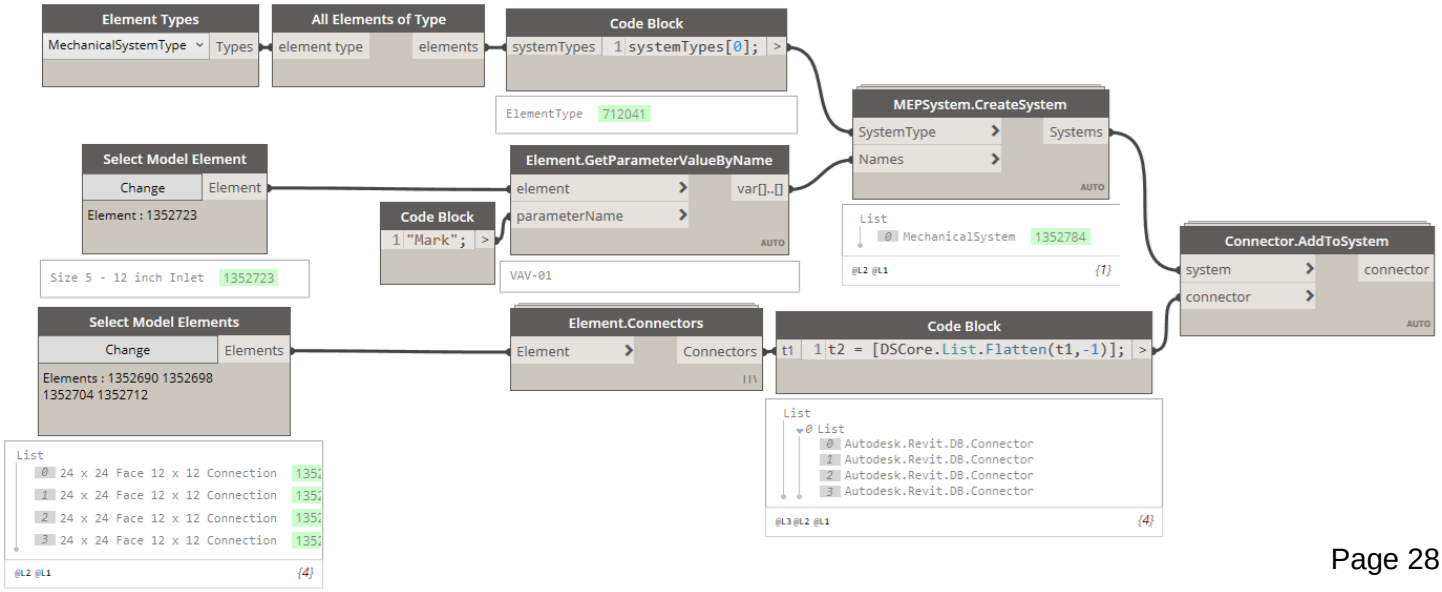
Custom nodes are needed to retrieve elements from a linked model. The examples to the right show how to collect linked elements using two C# nodes from Rhythm and Bimorphs packages.



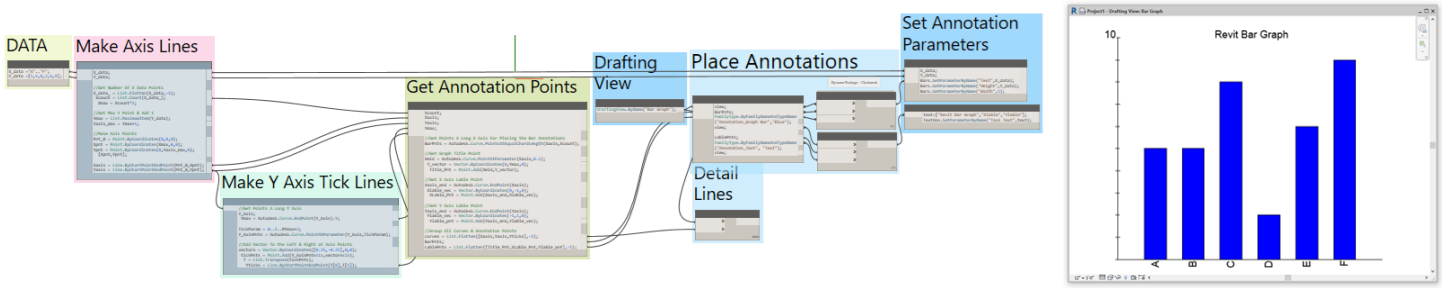
Dynamo has a fantastic node called *RayBounce.ByOriginDirection* that returns an Elements and position hit by ray bounce from the specified origin point and direction. However, to find elements using this method from a linked model requires a custom node. Bimorphs packages have this covered with the *LinkedElement.ByRayBounce* node. This node can be utilized for a whole range of possibilities. The example below uses the node to get the surface of a linked ceiling above a Revit Space.



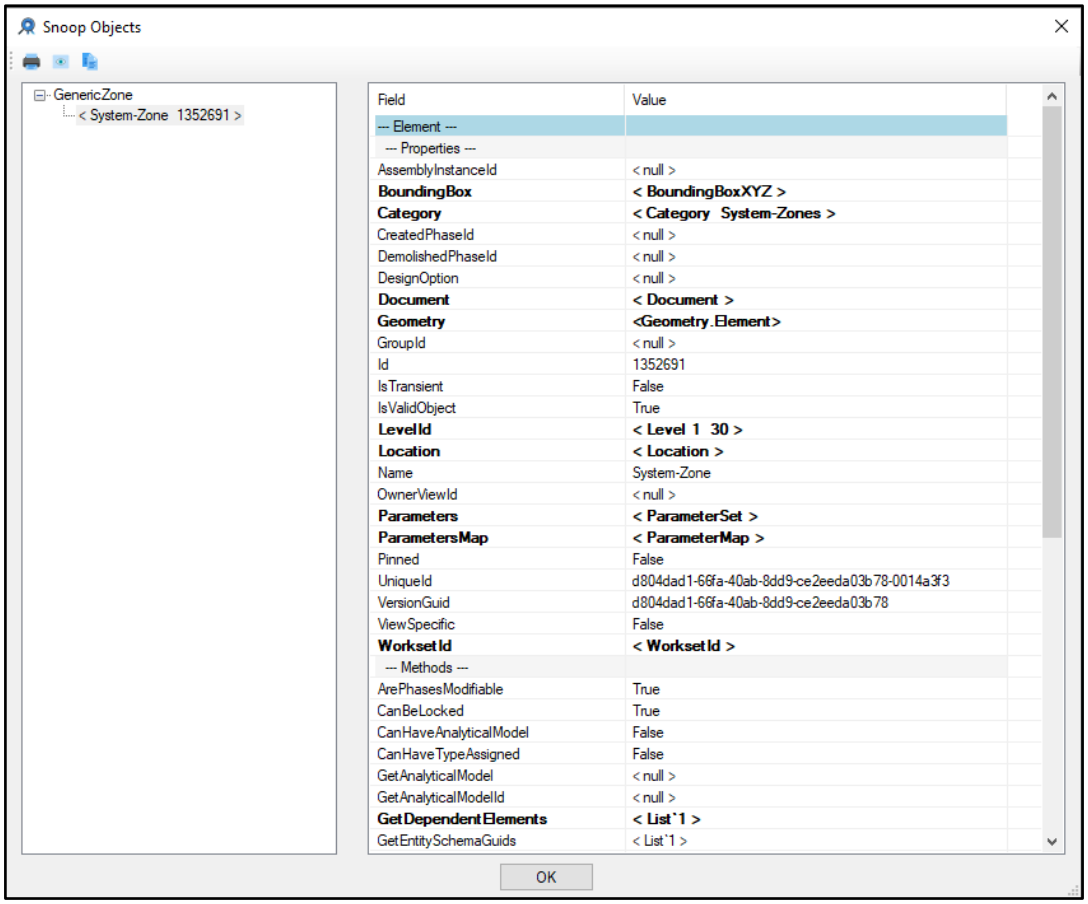
Gaining access to MEP Systems and Mechanical Connectors requires access to the Revit API. This example shows how to create a System and add elements with nodes from the MEPover package.

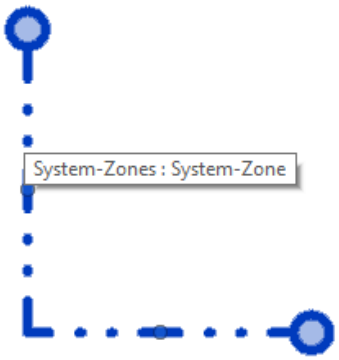


Placing Generic Annotations into Drafting Views requires a custom node called *FamilyInstance.ByPointinView* from the Clockwork package. The exercise below creates a parametric bar graph inside a Revit drafting view, demonstrating the robust potential of placing Generic Annotation and Detail lines with Dynamo algorithms.

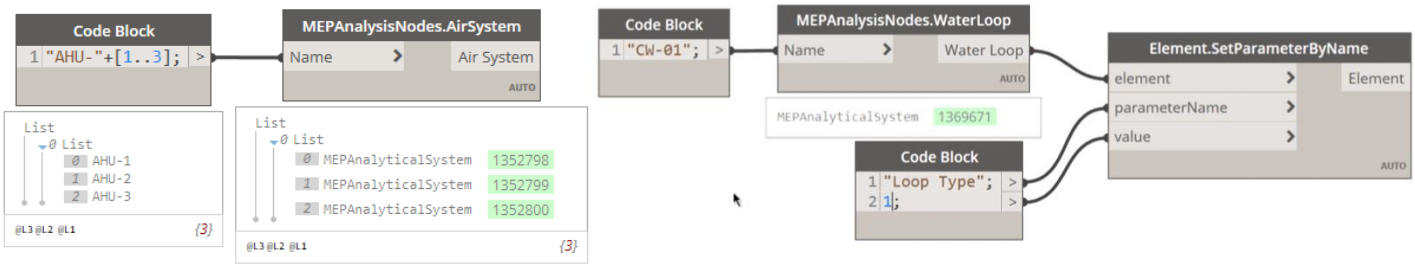


Sometimes all the API information needed to create a custom node can be challenging to find. For instance, when setting out to automate the creation of Analytical Systems, the API documentation had not been completed yet. A Revit add-in called Revit Lookup was vital in exposing the raw data for the elements that make up the Analytical Systems. This free add-in identifies read-only parameters, parameter types, relationships, and helps developers become familiar with the API's structure. These functions make it an essential tool for creating custom nodes.

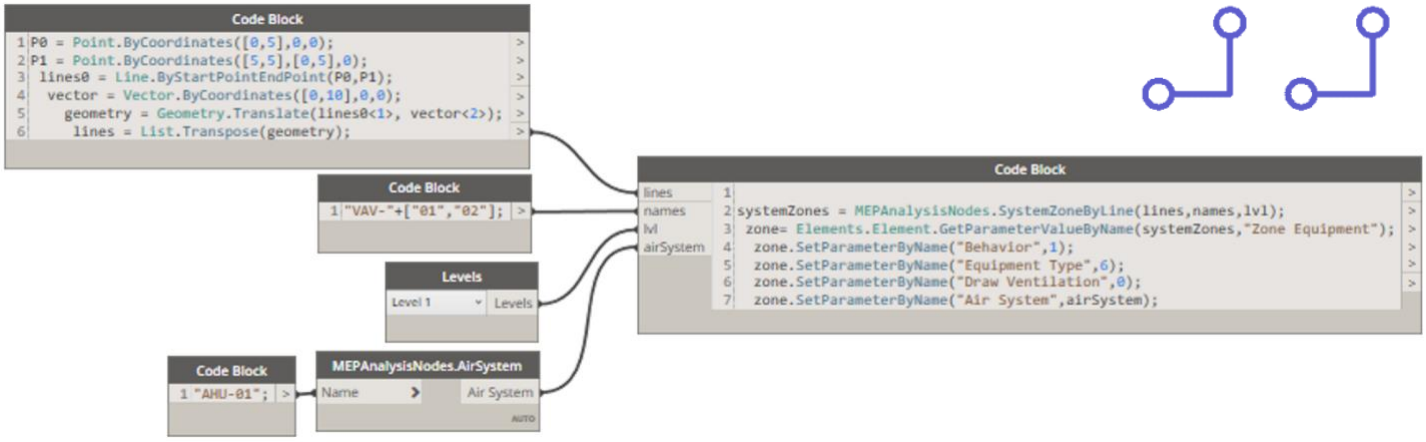




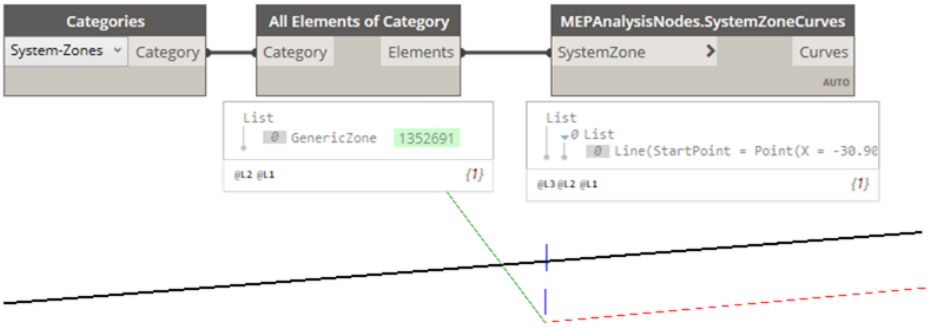
Revit Lookup gives us the capability to create the required nodes for automating the creation of Analytical Systems. The *MEPAnalysisNodes.Air System* and *MEPAnalysisNodes.WaterLoop* node creates Air Systems and Water Loops from a list of names. Once the elements are created, the parameters can be set to assign the equipment properties.



The *MEPAnalysisNodes.SystemZoneByLine* creates a System Zone while also building and assigning a new Zone Equipment from Dynamo geometry. The input to the node is a list of Dynamo curves, the name, and the level. In this example, two Analytical VAV Boxes are created and assigned to a previously created Air System.



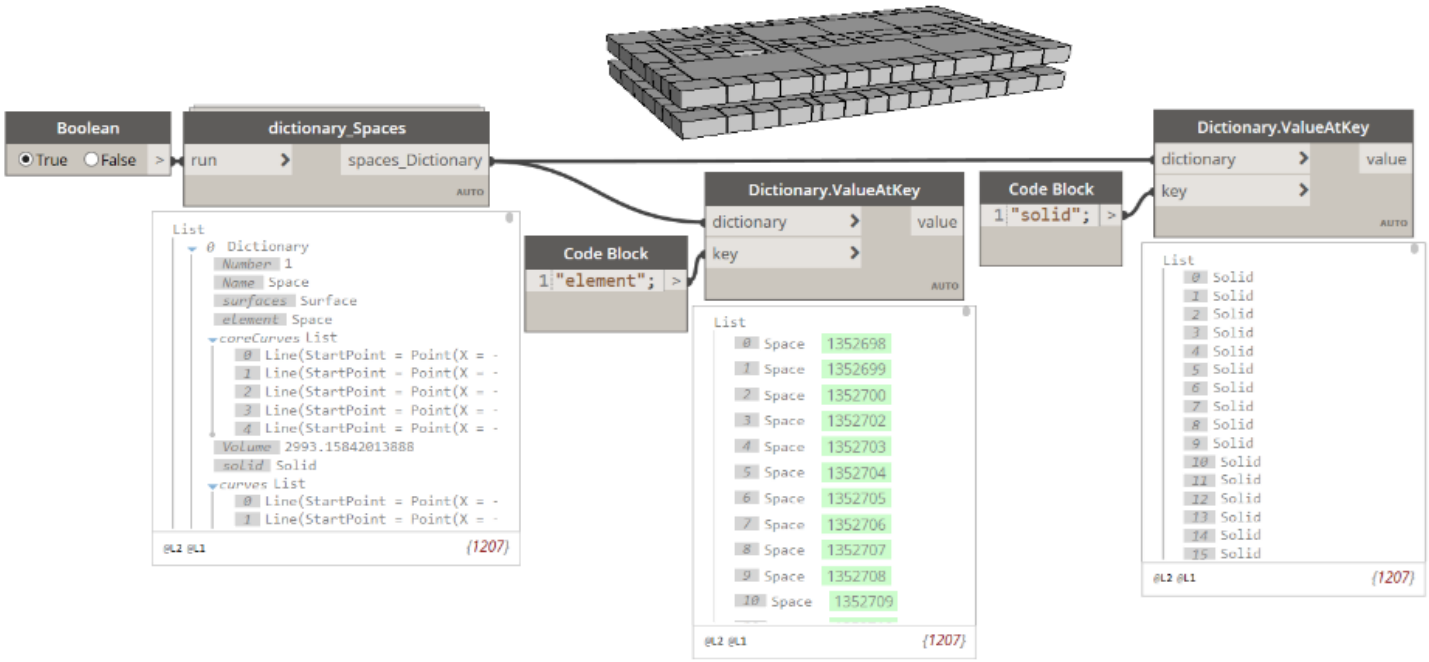
Surprisingly, the trickiest System Analysis node we created was the MEPAnalysisNode.SystemZoneCurves, which retrieves the mode line that defines the System Zone. The out-of-the-box method to get the curve in the generic zone class was faulty. It appears to work once per session, then for some reason stops working later. To make sure it was consistent, we used Revit Lookup to go back in the class hierarchy to the introductory Revit element class in order to get the dependent elements to the generic zone in question. This included the sketches where the curves are initially drafted, which gave us access to the curves we needed. So always



remember, think outside the BOUNDING box!

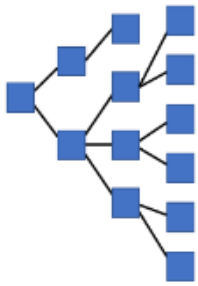
By simply consolidating nodes into one customized node, graphs become cleaner, clearer, and faster. For instance, inevitable data errors can be masked and calculation time is reduced for geometric functions since the geometry is not rendered and messy wires are hidden. We have created custom nodes for retrieving frequently used elements like Spaces. The nodes are utilized to clean and output the data in organized dictionaries. A dictionary is a collection of data that is linked to a key value. Dictionaries make retrieving data in workflows more straightforward. Data is stored and retrieved with keys, thus removing the need to maintain list structure during a workflow.

In the example below, the custom node called “dictionary_Spaces” collects all the Spaces from the model, cleans essential parameter/geometric attributes and combines the data into a dictionary. The Space elements and solid geometry is then retrieved using the Dictionary and the corresponding key.

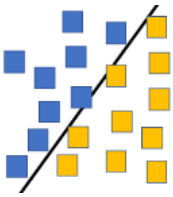


Parameter data and geometry from Revit elements is the oil for the algorithmic gears that drive automation. When combined with general-purpose algorithms, engineering design problems can be computationally solved. These general algorithms are often independent of design problems but provide approaches to solve pieces of the whole puzzle. The next sections serve as an introduction to some popular algorithm techniques and all the different flavors of computational geometry.

Clustering Algorithms



A relatively common task requires grouping a set of objects into subsets so that the elements get grouped with similar elements. For instance, thermal zones within a building are comprised of groups of Spaces. These groups are based on locations, thermal set points, and similar load profiles. A class of algorithms called “clustering” has many useful tools for exploring data and has solved these types of problems. The most straightforward clustering algorithm is a single if statement that splits a set of data into two subsets with one conditional statement. A decision tree is merely multiple if statements linked together to conditionally control data. This algorithm can group data into an endless subset using classification rules, forming a hierarchical clustering.

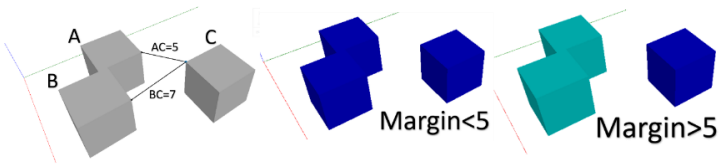


Well-known and understood algorithms from many different fields often get used in clustering techniques. For example, linear regression developed in statistics provides a straightforward approach to clustering data and geometry based on mathematical averages.

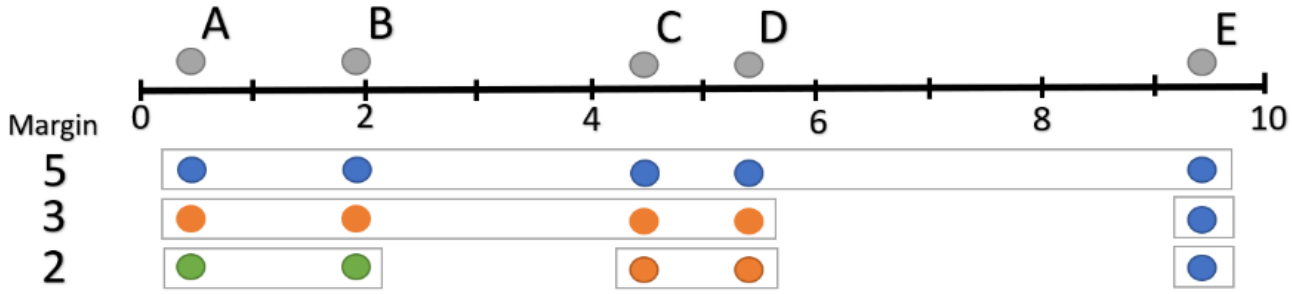
Another way to cluster data is in Non-Hierarchical groups. Non-hierarchical cluster analysis aims to find a grouping of objects which maximizes or minimizes some evaluating criterion. One such measure is the Euclidean distance between objects. The Python code below splits a list of geometry elements into groups if the distance between them is larger than the provided margin.

Python Script

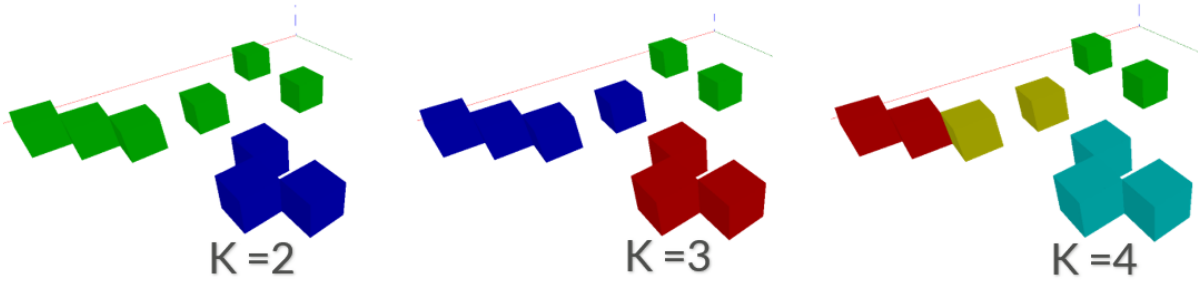
```python
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import Geometry

pnts = IN[0]
Δmax = IN[1]
Δ = Geometry.DistanceTo

clusters, Queue = [], []
while pnts:
    cluster = []
    Queue.append(pnts.pop() )
    while Queue:
        P0 = Queue.pop()
        cluster.append(P0)
        for i in xrange(len(pnts)-1,-1,-1):
            if Δ(P0, pnts[i]) <= Δmax:
                Queue.append(pnts.pop(i) )
    clusters.append(cluster)

OUT = clusters
```



This algorithm can be easily manipulated to use data rather than geometry to group items based on parameter values rather than distance. This can be pictured like points on a number line.

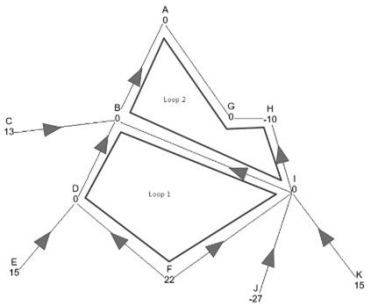


A popular, non-hierarchical method is k-means. k-means is an efficient, effective, and simple centroid-based clustering algorithm. This algorithm tries to partition n objects into k groups with the highest achievable distinction. The algorithm is often run multiple times with different starting conditions to produce an array of arrangements.



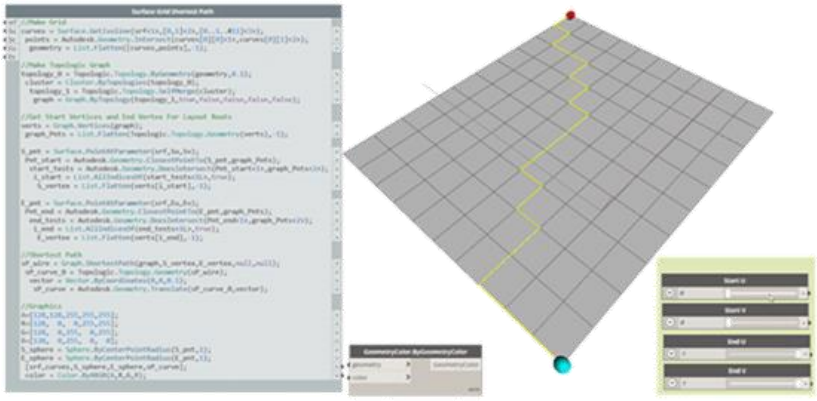
Graph Theory

Graph Theory is a branch of mathematics used to structure non-linear relations between objects. A graph is made up of vertices connected by edges that represent a network. For instance, a piping system's link structure can be represented by a graph in which the vertices represent fittings and edges represent the pipes. Overall, there are many different forms in which graphs cancome . A graph might be directed where an edge pointing from vertex x to vertex y is said to be incident from x and incident to y. A graph may be non-directed where edges are not directed ones. A simple graph is defined by having no loops, parallel edges, orconnected graphs if there exists a path between every pair of vertices. The list goes on! The base of most algorithms are Breadth First Search and Depth First Search traversal algorithms. The important thing to understand is that these graph characteristic and traversal methods create especially useful algorithms for MEP design.



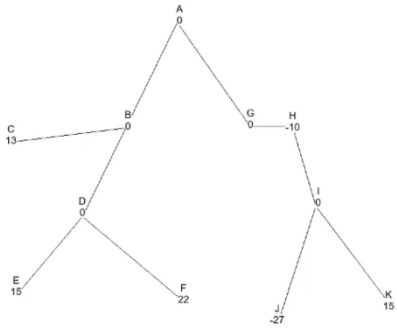
A Dynamo package called [Topologic]() offers powerful nodes for hierarchical and topological representations of architectural spaces, buildings, and artifacts through non-manifold topology. The package contains a collection of nodes for creating and analyzing network graphs.

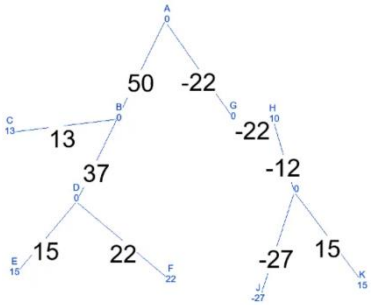
Shortest path algorithms find the shortest path from a source node to the destination node in the most optimal way. The image on the right shows nodes from the Topological Dynamo package used to find the shortest path through a grid.

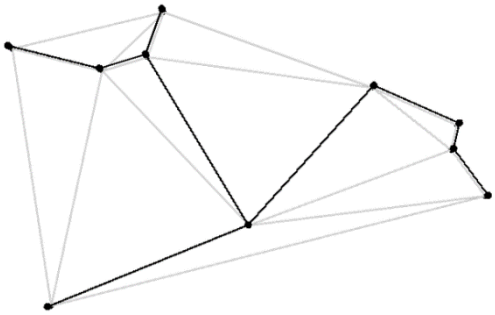


These types of problems fall under graph theory and can be solved with countless methods. The method selected depends on the structure of the graph.

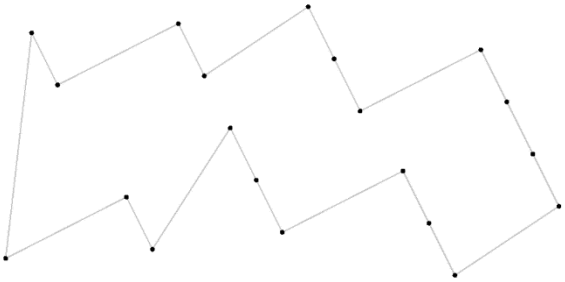
A tree is a simple structure. A particular set of rules defines trees: one root node may or may not connect to others, but ultimately, all connections stem from one specific place. Calculating flow back to a root node is a relatively easy process.







A graph with loops can be simplified. One such algorithm is the minimum spanning tree. Where a tree can be calculated from a more complex graph with cycles. This kind of tree reaches out to (spans) all the nodes while minimizing the length of the edges of the tree

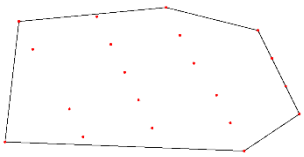


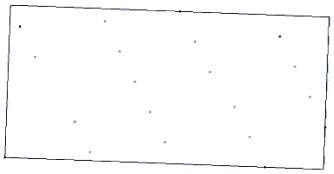
Traversal algorithms are also used to find approximate solutions for complex mathematical problems like the traveling Sales Man. This problem aims to find the shortest yet most efficient route through a given set of points.

Computational Geometry

Computational geometry is simply the design and analysis of efficient algorithms used to solve geometry problems on a computer. MEP design often boils down to problems dealing with polygons and points.

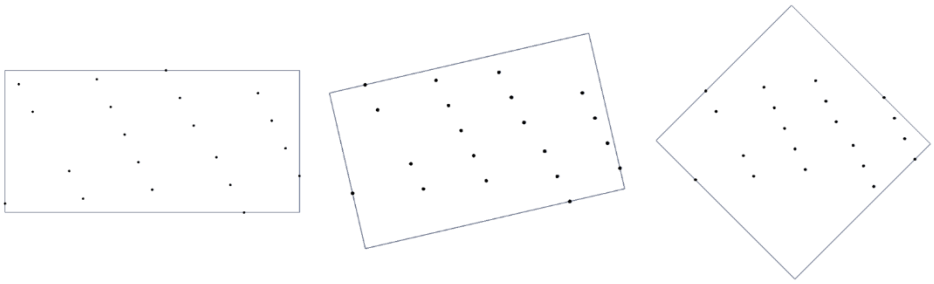
The convex hull of a set of points can be described as the closed polygonal chain of all outer points of the set, which entirely encloses all set elements. Computing a convex hull is one of the first advanced geometry algorithms. There are many variations on how to solve it.





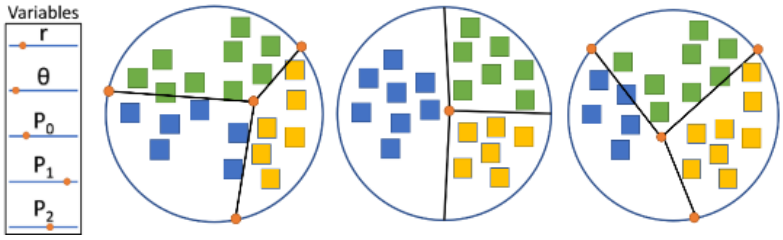
Finding an oriented bounding box is an essential tool for computational design. Built on top of the convex hull is an algorithm to compute the non-axis aligned minimum-area rectangle with a set of points.

It is also useful to have an algorithm that finds a bounding box at any specified angle.



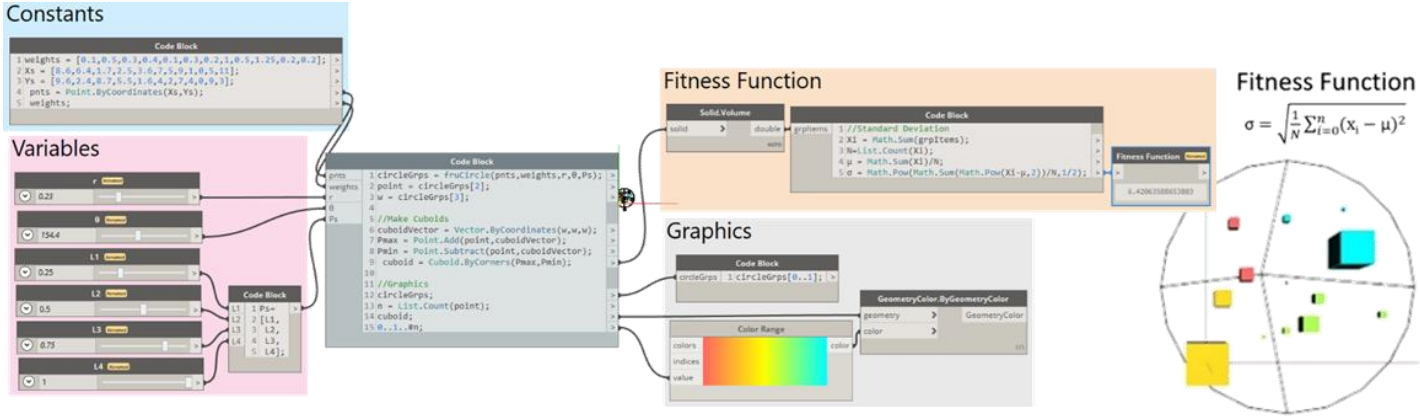
By bounding a set of points, they can be grouped by divisions. Bounding rectangles or circles constrain the algorithm so that the number of divisions can be specified.

Geometry systems are produced by adding variables to manipulate the geometric constraints to form a parametric model. A general characteristic feature of these systems is storing the construction steps and executing those steps after a change is made to the input data. For example, the circle in the example below has five variables to control the pie pieces used to group the points. The center points have two variables, the polar coordinates (r, θ) inside the circle. The line starts the parametric focal point and ends at a point on the circle at a specified angle forming the other three variables. Different variable combinations can then be cycled through to develop grouping configurations.



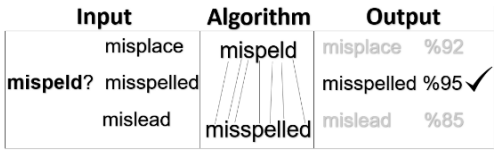
Genetic Algorithms

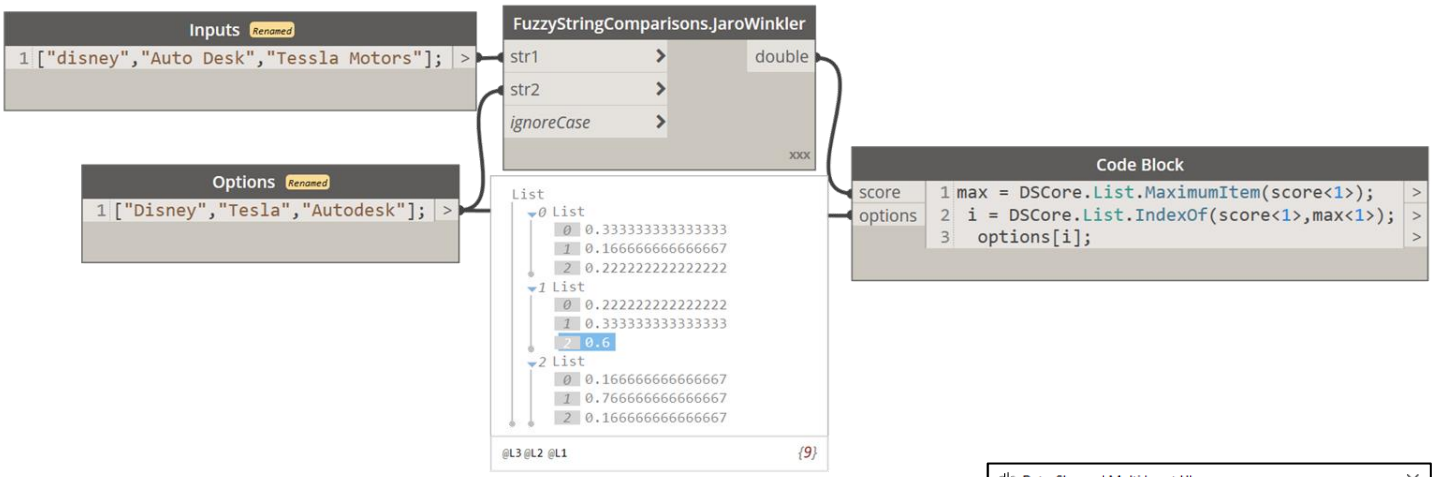
The foundation of a genetic algorithm is a geometry system that combines variable and constant inputs, executes several operations, and outputs several solutions. A genetic algorithm is a calculated trial and error search inspired by Charles Darwin's natural selection theory. The fittest individuals (inputs variables) are selected for reproduction to produce offspring of the next generation of inputs. The process optimizes designs quickly by adding a feedback loop. At the end of each generation, the design is scored or given a "fit score." Characteristics or attributes from designs with high fitness are then used to create a new set of inputs for the design problem. This process is called "cross over." Like biology, the method also introduces mutation or random modifications to the new set of inputs after each generation provides diversity to the following generation. This guarantees that the algorithms will not converge on false optimal solutions. The process repeats, creating a new and improved generation until it finds the best combination of inputs.



Fuzzy Logic

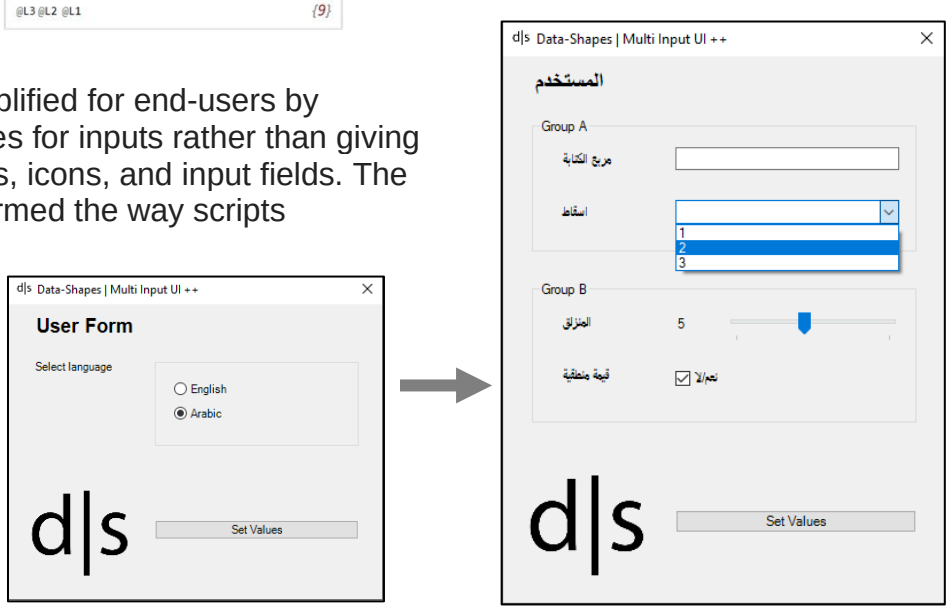
Fuzzy logic helps deal with uncertainty by using approximations. Fuzzy logic values range between 1 and 0. One being entirely accurate and zero being false. The scores are then used to find the closest match between inputs in a data source. Fuzzy string search is used in various applications like misspell words and Dynamo search for nodes. The example below performs this method with a node from the Fuzzy Dynamo package.

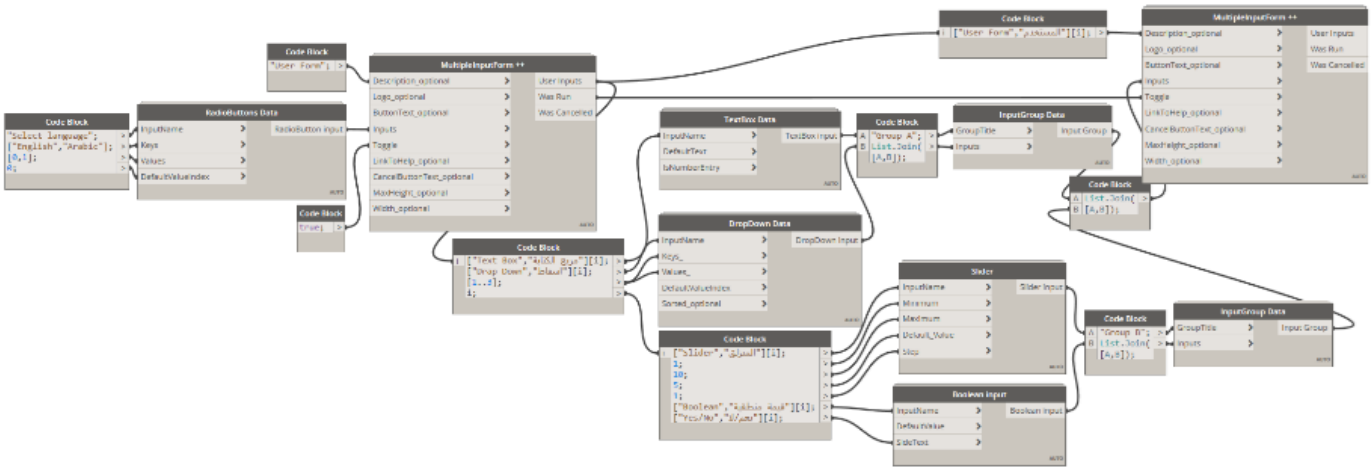




Dynamo UIs

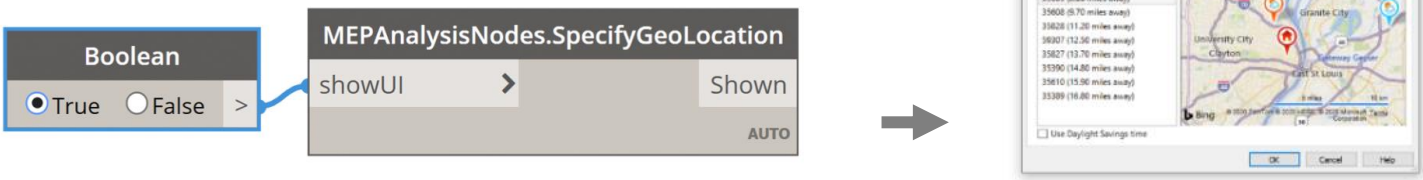
Workflows can be significantly simplified for end-users by presenting them with user interfaces for inputs rather than giving written instructions to navigate tabs, icons, and input fields. The Data-Shapes package has transformed the way scripts can interact with end-users. User forms can be created with numerous input types and use any project data to drive the inputs. The central aspect that differentiates Data-Shapes UI from Dynamo Player is that multiple UIs can be used in a single script. With this ability, information from one UI can



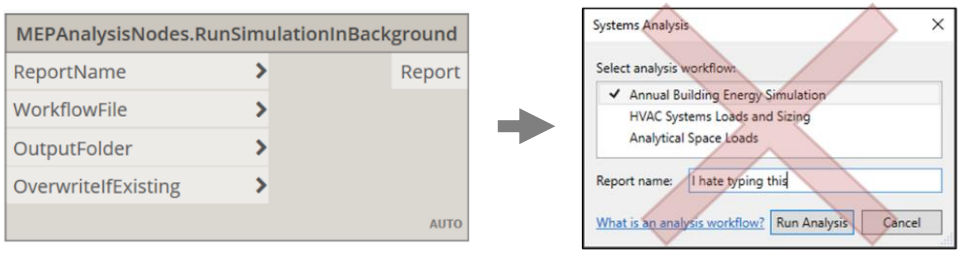


determine what inputs pop up on a subsequent UI.

To seamlessly integrate Revit UIs into Dynamo workflows, custom nodes were developed to open Revit windows with Dynamo. This has a variety of applications, such as setting the projects location for weather data.



Other custom nodes were designed to remove interfaces that could be filled in with Dynamo, such as the UI for activating a Systems Analysis simulation. While building this node, we discovered that more flexibility could be offered through functions like specifying an output folder or offering an option to overwrite the existing report.



This chapter laid out most of the computational design tools and techniques that are used in the upcoming chapters. Thanks to the Dynamo community, tons of general-purpose algorithms and custom nodes for needed API functions have already been developed for Dynamo. Therefore, it is more important to get familiar with the Revit API and many algorithmic methods instead of writing them from scratch. Understanding the reasoning behind different methods gives you more tools to pick from and opens your mind to many creative ways to efficiently solve complex design problems. These techniques are the building blocks for more complex algorithms and can be reused for various problems, adding powerful tools to your algorithmic toolbox.

Automating MEP Tasks

Nearly all aspects of setting up and running an energy simulation can be automated once standardized Revit content is established and data connection between frameworks is created. This chapter details how to automate fundamental design tasks needed to set up and run a Revit Systems Analysis Workflow. These processes include setting up and exchanging information between area elements (Room, Space, and Analytical Space), configuring different HVAC Systems, and integrating systems into Revit projects.

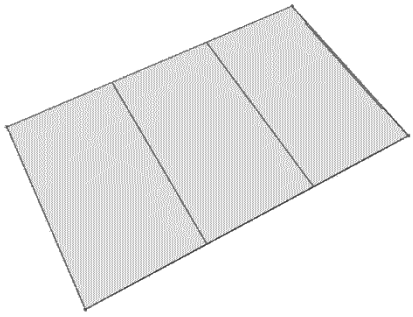
Air Balance Table Coordination

Rooms, Space, and Analytical Spaces all define areas within a building and have a critical function for distinct Revit workflows. One essential requirement that is often overlooked in regard to how these elements are set up is the need for an air balance table. The air balance table tabulates design criteria to ensure ventilation requirements are met for all occupied areas. The table includes temperature, airflow rates, ventilation requirements, and thermostat setpoints, among other things. Data is needed from Rooms, Spaces, and Analytical Spaces to create the report; therefore, connections between these elements need to be established.

When a MEP model is created from a linked Architecture model, Rooms can be created from Rooms in the linked model. The reason for adding Rooms to the MEP model is so an Air Balance Table can be created that matches the Architectural Rooms exactly. This frees up Spaces to be used to subdivide large or odd shaped Rooms. The example to the right collects data from a list of linked rooms, finds the new host level, creates, and then pins the new Rooms.

Project Set Up Room By Linked Rooms Renamed

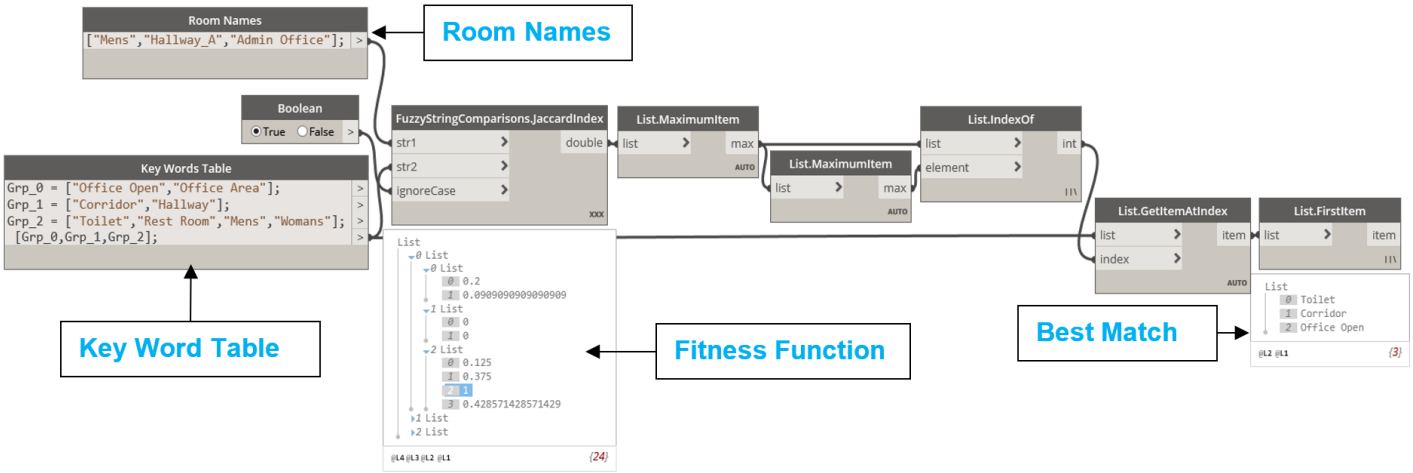
```
def projectSetUp_roomByLinkedRooms(linkedRooms:var[],levels:var[])
{
//Get Linked Room Data
roomPt = Revit.Room.Location(linkedRooms);
roomName = Revit.Room.Name(linkedRooms);
roomNumber = Revit.Room.Number(linkedRooms);

//Get New Rooms Level
lvls = Level.Elevation(levels);
 lvl_test = roomPt.Z >= lvls;
  newLvls = levels[DSCore.List.FirstIndexOf(lvl_test<1>,true)];

//Make New Rooms
newRooms = Revit.Room.ByLocation(newLvls,roomPt,roomName,roomNumber);
 return = newRooms.SetPinnedStatus(true);
};
```



Spaces are created in a similar fashion but are more involved. First, an area limit is established for the maximum area of a Space. The area of the linked Rooms is divided by this limit to find Rooms that need to be split into multiple Spaces. The Center Boundary curves of the Rooms that need to be split are used to form a surface. The minimum rectangle is found and used to split the surface. The center points of the surfaces are then used to find the new Space points. The curve used to split the surface is used to create Space Separation lines in the model to bound the Spaces.

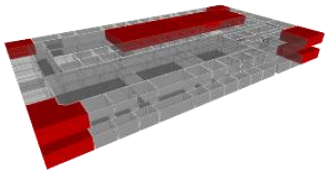
Since Revit Spaces are used to control the ventilation CFM in the energy simulations, it is important to ensure that the volumes are set correctly. There are a few parameters that control a Spaces volume. One that will need to be edited is the “Limit Offset” parameter.

Next, Space Types need to be assigned to the new Spaces. The Space Types can be automatically matched to the room names using fuzzy logic and a table that contains common alternative names for each Space Type. For instance, restrooms may also be called, Men's, Women's, bathroom, toilet and so on. All these names get mapped to a Space Type and fuzzy logic is used to find the best name.

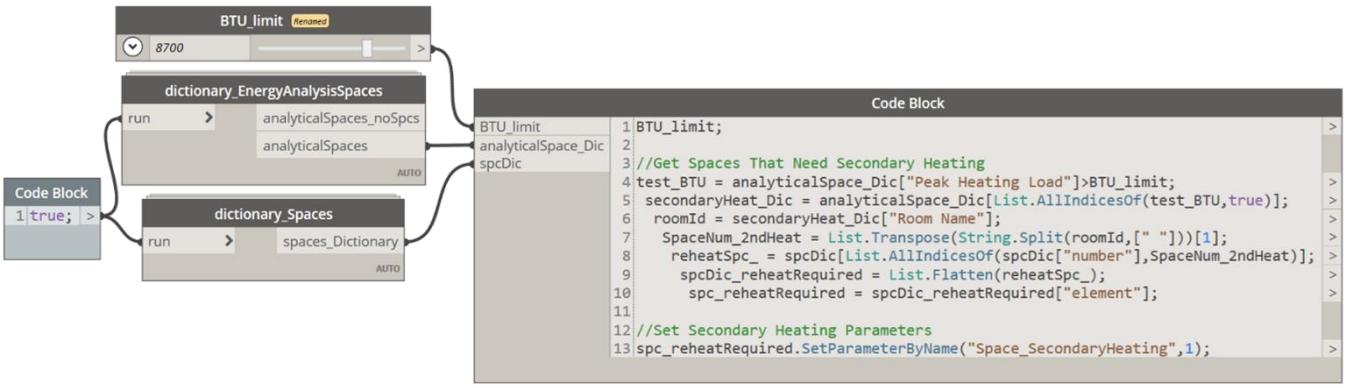


System Configurator

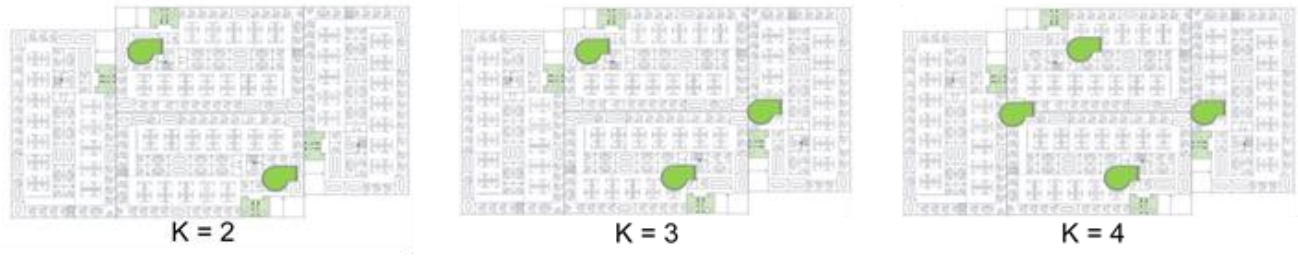
Geometry systems can be created to assigned Spaces to Mechanical Equipment. As mentioned earlier, geometry systems are puzzles consisting of available data, algorithmic approaches, and the designer's knowledge. Setting up systems that produce good results relies on the designer's lateral thinking ability and ability to deconstruct the design problem. All ambiguous steps of the design problem must be paired to an appropriate algorithmic strategy. Then, all the different operations need to be connected to solve the puzzle. Required data can be gathered directly from the model, variables can be utilized to create many design options, and user interfaces allow the end-user to enter any remaining inputs. Below are a few strategies used to assign different equipment types to Spaces using geometry systems.



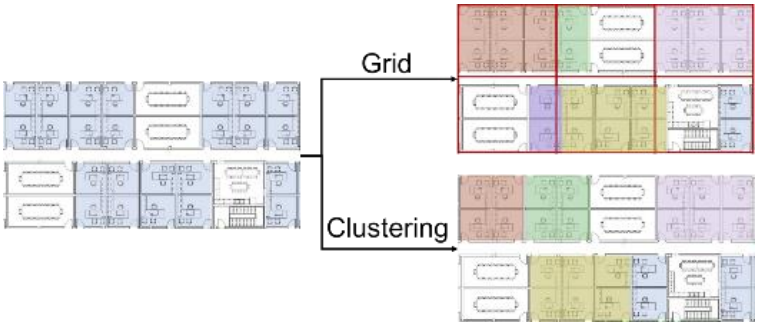
This first example uses an algorithm to determine which spaces require secondary cooling by utilizing the Analytical Space load data. For instance, suppose a Space requires 3500 BUT/hr of heating and includes a VAV Box that only supplies 1000 BUT/hr of heating. In this case, the Space parameter "Space_SecondryHeating" is set to true. The Dynamo scripts first check to see if the "Peak Heating Load" is above the specified limit. If so, the corresponding Space is found using the "Room Name" parameter, and Space's "Space_SecondaryHeating" parameter is set to true.



This second example uses the K-means algorithm to group Spaces that require exhaust fans. This process is a visual perception problem, meaning that a straightforward grid might blindly divide a cluster of Spaces. The desired number of exhaust fans designates the number of centroids inputted into the K-means algorithm.

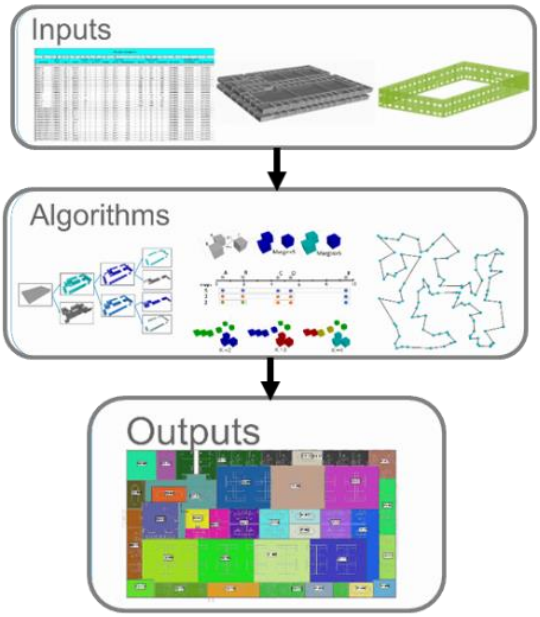


The third example discusses a combination of general-purpose algorithms used to group Spaces into thermal zones. The rules are straight forward - group spaces by those sharing attributes, such as similar heating and cooling loads, temperature setpoints, pressure settings, etc. Do not group Spaces with different exterior wall orientation or large separations. The visual perception problem shows up again, along with distance constraints being added to the mix.

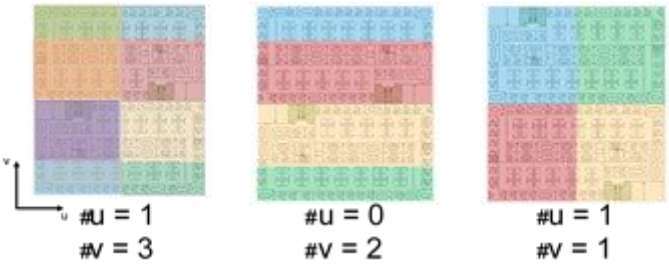


Space, Analytical Spaces, and Analytical Surfaces all serve as fixed inputs to the algorithm. A user interface is added, allowing the designer flexibility and control over the constraints. These constraints set the maximum number of Spaces, CFM, area, distance, and loads for the thermal zones.

First, the Exterior Analytical Surfaces and the Analytical Spaces are grouped in order to exchange data. The second step uses a combination of data, the *Geometry.GroupByDistance* custom nodes, and a series of logic if statements to create a decision tree to group the Spaces. The third step uses clustering algorithms to divide the previous groups into thermal zones. Then, the traveling salesman algorithm is used to assign unique ids to each thermal zone.



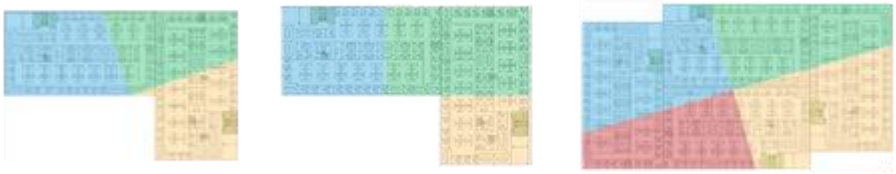
This last system configuration example uses a geometry system to assign thermal zones to Air Systems and Water Loops. A simple and straightforward approach splits the Spaces up using a grid where the number of U and V divisions is specified.



With the grid as the starting point, larger design spaces can be produced by adding more variables and data into the geometry system. Variables to rotate and shift the grid lead to a more dynamic way of grouping spaces. Tables of equipment performance data can be cycled through forming different combinations. Then, quantifiable measures are calculated for each configuration. For example, adding up the cooling loads for all the spaces in each group to find the total cooling load on each coil.

Quantifiable Measure

- Total Area
- Total Cooling Load
- Maximum Distance
- Excess Compacity



Using the generative design framework for Revit or Dynamo, the quantifiable measures are transformed into measurable goals set by the designer. The design space can then be explored, evaluated, and optimized. One optimization goal might be to evenly distribute the cooling loads among all the cooling coils. The standard deviation between all the coil total loads can be calculated and used as the fitness function for the genetic algorithm. This generative design study's single objective would be to minimize the standard deviation of the cooling loads on each coil.

Frequently, the goal is to optimize several objectives where the goal is to discover the best compromise. With the cooling coil example, the goal might be to minimize the distance between Spaces assigned to a coil while evenly distributing the cooling loads among the cooling coils. Another study may look at the tradeoff between the system's pressure drops and to the number of fans.

The methods outlined in this section can be combined to set up a geometry system to configure all the required systems for the entire building in many different arrangements. One way to record the results is with parameters that have been added to the Spaces. The results can then be reviewed with schedule and floorplans view or in 3D with Dynamo. This data can then be used to integrate the systems into the Revit model.

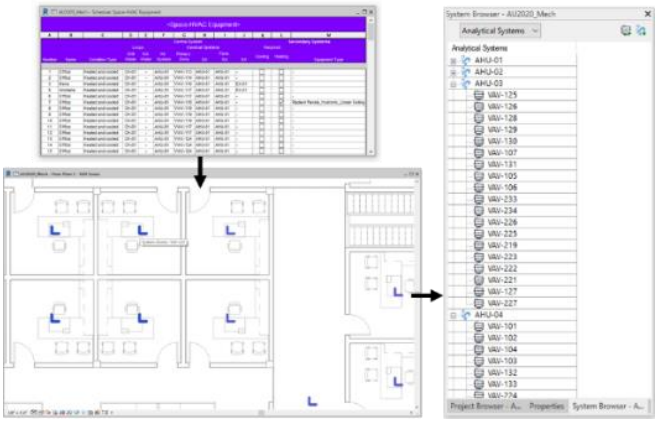
AU2020_Mech - Schedule: Space-HVAC Equipment

<Space-HVAC Equipment>

A	B	C	D	E	F	G	H	I	J	K	L	M
			Central System									Secondary Systems
			Loops		Central System					Required		
							Fans					
Number	Name	Condition Type	Chill Water	Hot Water	Air System	Primary Zone	SA	RA	EA	Cooling	Heating	Equipment Type
1	Office	Heated and cooled	CH-01	-	AHU-01	VAV-113	AHU-01	AHU-01	-	☐	☐	-
2	Office	Heated and cooled	CH-01	-	AHU-01	VAV-114	AHU-01	AHU-01	-	☐	☐	-
3	Mens	Heated and cooled	CH-01	-	AHU-01	VAV-110	AHU-01	AHU-01	EX-01	☐	☐	-
5	Womans	Heated and cooled	CH-01	-	AHU-01	VAV-117	AHU-01	AHU-01	EX-01	☐	☐	-
6	Office	Heated and cooled	CH-01	-	AHU-01	VAV-117	AHU-01	AHU-01	-	☐	☐	-
7	Office	Heated and cooled	CH-01	-	AHU-01	VAV-118	AHU-01	AHU-01	-	☐	☑	Radiant Panels_Hydronic_Linear Ceiling
8	Office	Heated and cooled	CH-01	-	AHU-01	VAV-119	AHU-01	AHU-01	-	☐	☐	-
9	Office	Heated and cooled	CH-01	-	AHU-01	VAV-119	AHU-01	AHU-01	-	☐	☐	-
10	Office	Heated and cooled	CH-01	-	AHU-01	VAV-119	AHU-01	AHU-01	-	☐	☐	-
11	Office	Heated and cooled	CH-01	-	AHU-01	VAV-117	AHU-01	AHU-01	-	☐	☐	-
12	Office	Heated and cooled	CH-01	-	AHU-01	VAV-117	AHU-01	AHU-01	-	☐	☐	-
13	Office	Heated and cooled	CH-01	-	AHU-01	VAV-124	AHU-01	AHU-01	-	☐	☐	-
14	Office	Heated and cooled	CH-01	-	AHU-01	VAV-124	AHU-01	AHU-01	-	☐	☐	-
15	Office	Heated and cooled	CH-01	-	AHU-01	VAV-124	AHU-01	AHU-01	-	☐	☐	-

Systems Builder Analytical System

This section demonstrates translating the System Configurator's outputs,which are stored in the Spaces, into Analytical Systems for energy simulations. Analytical Systems can automatically be set up using our custom Dynamo nodes, so that the whole process is wrapped into one clean definition. The only input required for the definitions is the output from the *Dictionary_Spaces* node. The result is a VAV Zone Equipment, Air Systems, and water Loops for a specified system configuration



Multiple definitions are set up with different system types and assumptions. The system assumptions are listed in the first few lines of the definitions..

First, Air Systems are created by getting the unique AHU from all the spaces. Then, applies the fan, coils, and heat exchanger assumptions. For example, setting the cooling Coil parameter to 2 means Direct Expansion. Remember the Enum table. The second step makes the lines for the System Zones. Perpendicular lines are created at the Space location point and grouped by the Zone assigned to each Space.

In the third step, System Zones and matching Zone Equipment are created from the lines, associated levels, and thermal zone names

The last step sets the parameters for the Zone Equipment (Behavior, Equipment Type, and Air System).

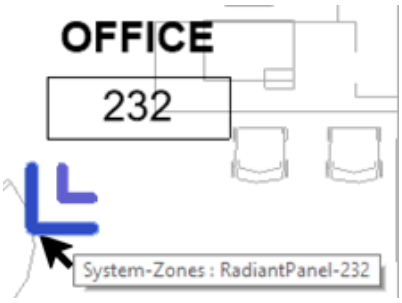
Analitcal Air Systems and VAVs From Spaces Dictionaries *Renamed*

```
def analitcalSystemBuilder_VAVnoReheat_AHUdxGas(spcDic:Dictionary[])
{
/*Analytical Systems Assumtions
VAV boxes No Coils @ Space Zones
AHU-Variable Volume Fan, Direct Expansion Cooling Coil, Funace Heating Coil*/

//Group Space Dictionaries By VAV Zones
grpSpcDic_VAV = List.GroupByKey(spcDic,spcDic["zone"]);

//Make Air Systems & Set Parameters
VAV_AHU = List.FirstItem(grpSpcDic_VAV["groups"]["AHU"]<1>);
 airSystems = MEPAnalysisNodes.AirSystem(List.UniqueItems(VAV_AHU));
  A=airSystems.SetParameterByName("Fan",1);
  B=airSystems.SetParameterByName("Cooling Coil",2);
  C=airSystems.SetParameterByName("Heating Coil",2);
  D=airSystems.SetParameterByName("Preheat Coil",0);
  E=airSystems.SetParameterByName("Heat Exchanger",0);

//Make VAV Lines
spcPts = grpSpcDic_VAV["groups"]["pont"];
ptX = Point.Add(spcPts,Vector.XAxis());
ptY = Point.Add(spcPts,Vector.YAxis());
 Xline = Line.ByStartPointEndPoint(spcPts,ptX);
 Yline = Line.ByStartPointEndPoint(spcPts,ptY);
  VAV_lines = List.Flatten(List.AddItemToFront(Xline<1>,Yline<1>)<1>,-1);

//Make VAV System Zones (lines,Name,Levels)
VAV_name = grpSpcDic_VAV["unique keys"];
VAV_lvl = List.FirstItem(grpSpcDic_VAV["groups"]["level"]<1>);
 genericSystem =MEPAnalysisNodes.SystemZoneByLine(VAV_lines,VAV_name,VAV_lvl);

//Get Zone Equipment & Parameters
VAV_zoneEquip = genericSystem.GetParameterValueByName("Zone Equipment");
 VAV_airSystem=airSystems[List.IndexOf(List.UniqueItems(VAV_AHU),VAV_AHU)];
  F=VAV_zoneEquip.SetParameterByName("Behavior",1);
  G=VAV_zoneEquip.SetParameterByName("Equipment Type",6);
  H=VAV_zoneEquip<1>.SetParameterByName("Air System",VAV_airSystem<1>);
  };
```

A different but similar definition is used to place Zone Equipment for Spaces that require a secondary system. A shared parameter is used to store the family name of the equipment that will be used to cover the remaining loads. The Dynamo script then reads that data and creates the proper Analytical Zone. In this example, Radiator Panels with Electrical Resistance heating get placed.

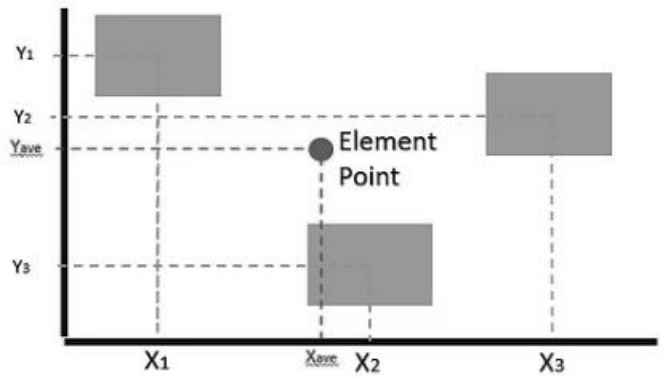
Analitcal Zone Equipment Secondry System From Space Dictionaries Renamed

```
def analitcalSystemBuilder_SecondarySystem(spcDic:Dictionary[])
{
//Get Space Dictionaries That Requier A Secondry System
scondSystem_test = spcDic["secondary system"]==="";
 spcDic_filter = DSCore.List.FilterByBoolMask(spcDic,scondSystem_test);
  spcDic_ = spcDic_filter["out"];

secondaryEquipment = String.Split(spcDic_["secondary system"],["_"]);
 family = DSCore.List.GetItemAtIndex(secondaryEquipment<1L>,0);
 type = DSCore.List.GetItemAtIndex(secondaryEquipment<1L>,1);

//Make Secondary Heating Lines
shift_0 = Autodesk.Vector.ByCoordinates(-1,-1,0);
shift_X = Autodesk.Vector.ByCoordinates(2,0,0);
shift_Y = Autodesk.Vector.ByCoordinates(0,2,0);
 pnt_0 = Point.Add(spcDic_["point"],shift_0);
 pnt_X = Point.Add(pnt_0,shift_X);
 pnt_Y = Point.Add(pnt_0,shift_Y);
  line_X = Line.ByStartPointEndPoint(pnt_0,pnt_X);
  line_Y = Line.ByStartPointEndPoint(pnt_0,pnt_Y);
   lines = DSCore.List.Transpose(DSCore.List.AddItemToFront(line_X,[line_Y]));

//Make System Zone
name = family+"-"+spcDic_["number"];
lvl = spcDic_["level"];
 systemZone = MEPAnalysisNodes.SystemZoneByLine(lines,name,lvl);

//Get Zone Equipment And Set Parameters
zoneEquipment = systemZone.GetParameterValueByName("Zone Equipment");
 A = zoneEquipment.SetParameterByName("Behavior", 1);
 B = zoneEquipment.SetParameterByName("Equipment Type", 10);
 C = zoneEquipment.SetParameterByName("Cooling Coil", 0);
 D = zoneEquipment.SetParameterByName("Heating Coil", 1);
 };
```



Systems Builder 3D

This section demonstrates placing the Revit families and creating MEP systems based on the System Configurator's outputs. Each type of equipment has a set of Algorithmic rules utilizing the model's geometry and data to find the general location point in the 3D model. A simple example of these rules is to calculate the average point of all the Spaces that serve a piece of equipment. For instance, finding the average point of the Space on a thermal zone and placing a VAV box there.



In the example to the right, VAV boxes are placed in the model using the thermal zones' average point and shifting that point to a specified elevation. First, Spaces are grouped by Zones, and their average point of each group is found. Then the z component is shifted upwards to a 10' elevation from the host level. Next, the supply CFM is summed together for all the grouped Spaces to find the VAV box's CFM. This value is used to select the family type that will be placed. Finally, the level is obtained from the first Space of the groups. The last step puts the VAV Box family and sets the parameters.

Place VAV Boxes No Rehest

```
spcDic
//Zone Average Point
zoneGrp = DSCore.List.GroupByKey(spcDic,spcDic["zone"]);
 zonePts = zoneGrp["groups"]["point"];
  zoneAves = Math.Average([zonePts.X,zonePts.Y,zonePts.Z]);
   VAVpnts = Point.ByCoordinates(zoneAves[0],zoneAves[1],zoneAves[2]);

//Get VAV Box Families Based on CFM
zoneSA = Math.Sum(zoneGrp["groups"]["CFM SA"]);
 VAVfamily = zoneSA>=1730?"VAV_RectInlet":"VAV_RndInlet";
 VAVtype =                zoneSA<45   ?"04in":
          zoneSA>45   && zoneSA<=60  ?"05in":
          zoneSA>60   && zoneSA<=75  ?"06in":
          zoneSA>75   && zoneSA<=145 ?"07in":
          zoneSA>145  && zoneSA<=235 ?"08in":
          zoneSA>235  && zoneSA<=340 ?"09in":
          zoneSA>340  && zoneSA<=475 ?"10in":
          zoneSA>475  && zoneSA<=825 ?"12in":
          zoneSA>825  && zoneSA<=1180?"14in":
          zoneSA>1180 && zoneSA<=1730?"16in":
                                      "24x16in";
VAVfamilyType = FamilyType.ByFamilyNameAndTypeName(VAVfamily,VAVtype);

//Get VAV Levels
VAVlvls = DSCore.List.FirstItem(zoneGrp["groups"]<1>)["level"];

//Place VAV Boxes
VAVs = Revit.FamilyInstance.ByPointAndLevel(VAVfamilyType,VAVpnts,VAVlvls);
 VAVs.SetParameterByName("Offset from Host",10);
 VAVs.SetParameterByName("Mark",zoneGrp["unique keys"]);
```

Elements that belong on the roof, like exhaust fan and rooftop units, use a similar method, but add the Ray Bounce technique to find a point that sits on the roof. In the example to the right, Exhaust Fans are placed in this manner. First, Spaces that do not require an exhaust fan are removed, and the remaining Spaces are grouped by their assigned Exhaust Fans. Second, the average point of the grouped is found. Third, the average point is shifted past the building's bounding box. Fourth, the ray bounce node is used to find the first element below the point above the building.

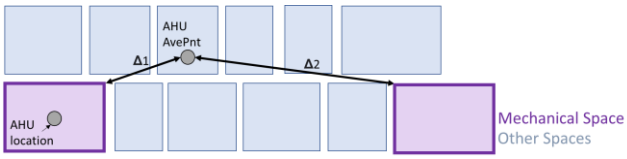
Place Exhaust Fans On Roof (Renamed)

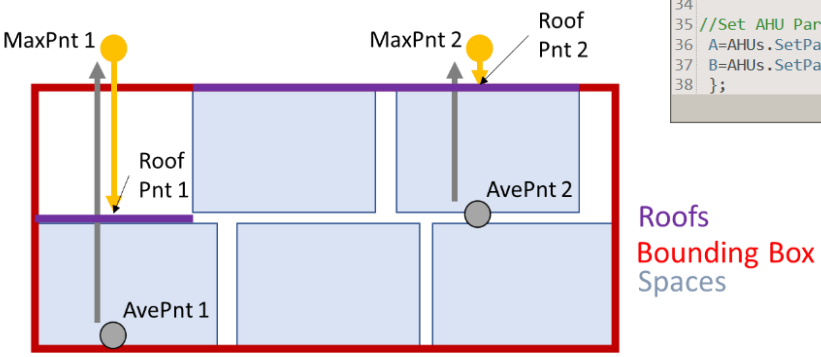
```
spcDic
spcDic;

//ZGroup Space Dictionaries By Exhaust Fans
exhuastFanGrp = DSCore.List.GroupByKey(spcDic,spcDic["fan EA"]);
 i_none = DSCore.List.AllIndicesOf(exhuastFanGrp["unique keys"],"");
  exFanIds=DSCore.List.RemoveItemAtIndex(exhuastFanGrp["unique keys"],i_none);
  exFanGrps = DSCore.List.RemoveItemAtIndex(exhuastFanGrp["groups"],i_none);

//Exhuast Fan Average Point
exFanPts = exFanGrps["point"];
 exFanAves = Math.Average([exFanPts.X,exFanPts.Y,exFanPts.Z]);
  exFanAve = Point.ByCoordinates(exFanAves[0],exFanAves[1],exFanAves[2]);

//Get Roof Location Point
spcSolids = DSCore.List.Flatten(spcDic["solid"]);
 spcsBBox = Autodesk.BoundingBox.ByGeometry(spcSolids);
  Zmax = Autodesk.BoundingBox.MaxPoint(spcsBBox).Z;
   rayOrigin = Point.ByCoordinates(exFanAve.X,exFanAve.Y,Zmax+10);
   rayDirection = Vector.ByCoordinates(0,0,-1);
   rayView = Document.ActiveView(Document.Current);

//Ray Bounce On Linked Elements (BimorphNodes)
rayExhuast=LinkElement.ByRayBounce(rayOrigin,rayDirection,rayView,null,true);
 exFanPnts = rayExhuast["Point[]"];

//Get Family
family = "σ_Exhuast Fan_Upblast";
type = "Direct Drive";
exFanfamilyType = FamilyType.ByFamilyNameAndTypeName(family,type);

//Place Exhuast Fans
exFan = Revit.FamilyInstance.ByPoint(exFanfamilyType,exFanPnts);
 exFan.SetParameterByName("Offset from Host",exFanPnts.Z);
 exFan.SetParameterByName("Mark",exFanIds["unique keys"]);
```

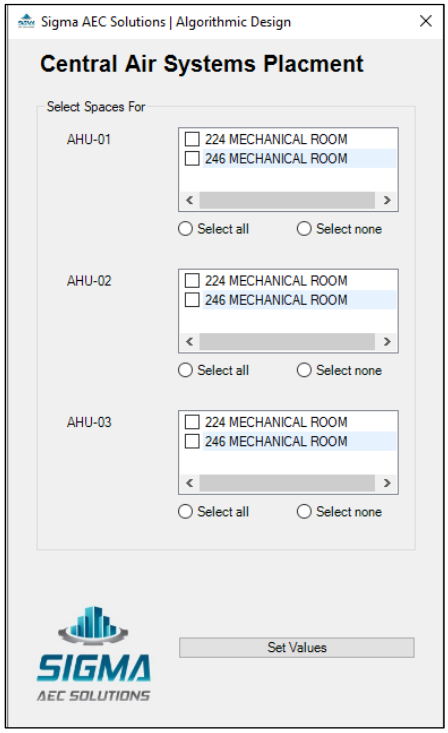
In the example to the right, AHUs are placed by finding the nearest Space with a Space Type set to Mechanical Room. First, the Spaces' are grouped by assigned AHU. Second, the average point of the grouped Spaces is found. Third, the distance between each AHU average point and all the Mechanical Spaces is calculated. The AHU family is then placed in the closest Mechanical Space.

Place AHU From Space Dictionaries (Renamed)

```
def mechanicalSystemBuilder_PlaceAHU(spcDic:Dictionary[])
{
//Get Mechanical Spaces
objType = DSCore.Object.Type(spcDic["space type"]);
 i_bldType = DSCore.List.AllIndicesOf(objType,"System.String");
  spcTypeDic = DSCore.List.RemoveItemAtIndex(spcDic,i_bldType);
  i_mech=DSCore.List.AllIndicesOf(spcTypeDic["space type"].Name,"Mechanical");
   spcDic_Mech = spcTypeDic[i_mech];

//Get Space Dictionaries That Are Assigned An AHU
AHU_test = spcDic["AHU"]=="";
 spcDic_filter = DSCore.List.FilterByBoolMask(spcDic,AHU_test);
  spcDic_ = spcDic_filter["out"];

//Group Spaces By AHU
AHUgrpPts = DSCore.List.GroupByKey(spcDic_["point"],spcDic_["AHU"]);
 AHUpnts = AHUgrpPts["groups"];
  aves = Math.Average([AHUpnts.X,AHUpnts.Y,AHUpnts.Z]);
   AHUave = Point.ByCoordinates(aves[0],aves[1],aves[2]);

//Measure Distance To Mechanical Spaces From AHU Averages and Get Minimum
Δs = Geometry.DistanceTo(AHUave<1>,spcDic_Mech["point"]<2>);
 min = DSCore.List.MinimumItem(Δs<1>);
  i_min = DSCore.List.IndexOf(Δs<1>, min<1>);
   AHU_spcDic = spcDic_Mech[i_min];

//Get AHU FAmily Type
ft = FamilyType.ByFamilyNameAndTypeName("Air Handling Unit","Gas Fired_DX");

//Place AHU
pt = AHU_spcDic["point"];
lvl = AHU_spcDic["level"];
 AHUs= Revit.FamilyInstance.ByPointAndLevel(ft,pt,lvl);

//Set AHU Parameters
 A=AHUs.SetParameterByName("Location",AHU_spcDic["Number"]);
 B=AHUs.SetParameterByName("Marck",AHUgrpPts["unique keys"]);
};
```





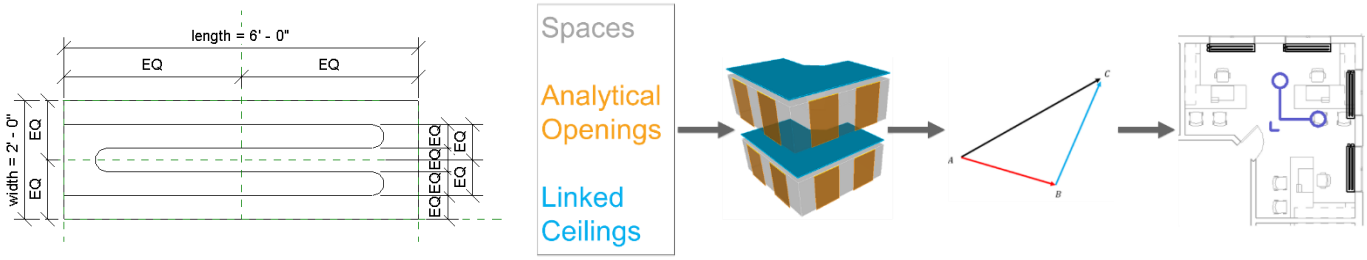


Another method for specifying the location of central plant equipment is to use a user interface that allows the designer to select a particular Space. Data Shapes live list node has a lot of flexibility. In this example the Mechanical Space and AHUs are the inputs for the node.

If you set the “HighlightInView” input of the node to *true*, the listview will function similarly to a Dynamo watch node. Clicking items in the list multiple times will browse Revit views containing the element and center these views on the element. This provides the end-user with a perfect way to browse through the mechanical Spaces.

With all the central plant equipment placed in the mechanical rooms, another script can distribute the equipment efficiently using a grid of points. The bounding surface of the Space can be obtained, and a grid of points can be placed on the surface.

Points can also be moved around by adding/subtracting vectors, identifying the location with the Ray Bounce node, or identifying starting points like doors, windows, Space location points, etc. For example, an algorithm to place ceiling Radiator panels. The algorithm needs to be made with the Revit family in mind. This Radiator family was created to make the algorithms simple by having the placement point be the bottom and center of the object. First, Analytical window openings get paired with Spaces that need the radiator panels. The second step shifts the center point of the analytical surface into the Space by one foot. The ray bounce node is used to find the ceiling above the point. The third step finds the ceiling's edge and gets the closest point to the Room's point. The Radiator family is then placed at this location.

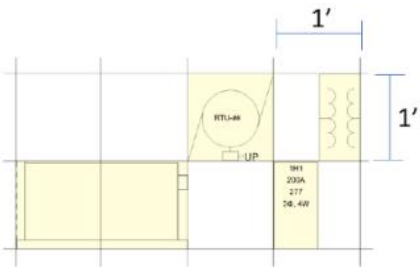


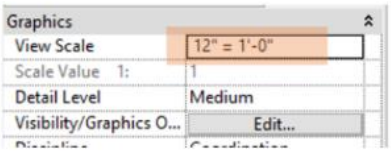
Once the equipment is in the model, MEP systems can be created, and pressure drops can be evaluated. To make the MEP systems from the elements of the *MEP systems.CreateSystem* and the *Connector.AddToSystem,* nodes are used from the MEPover Dynamo package.

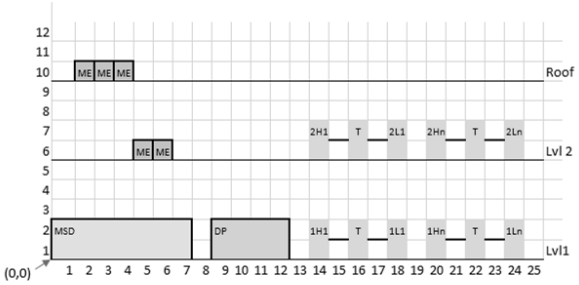
Systems Builder 2D

Two-dimensional schematics can automatically be set up in Revit drafting views by logically placing Generic Annotations and drawing detail lines. This section outlines this process for AHU schematics, hot water riser diagram, and psychometric charts.

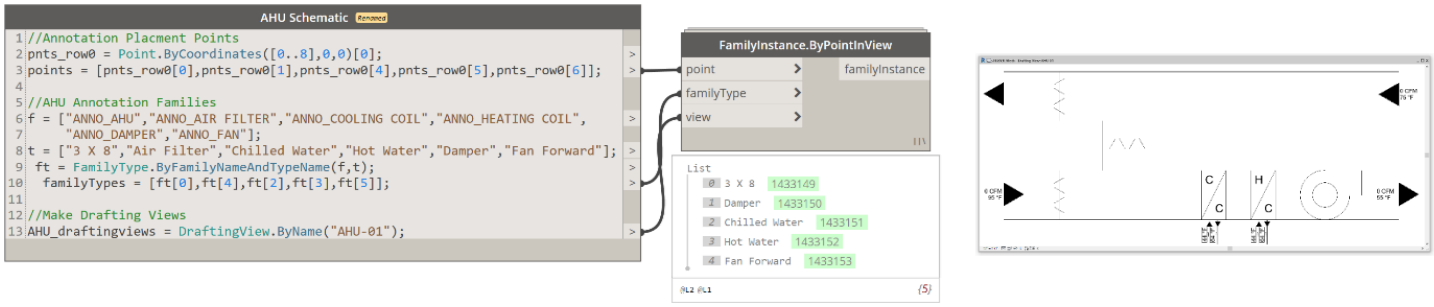
It helps to think about the drafting views as a sheet of graph paper. This simplifies laying out the annotations and the lines that will connect the annotations. All the annotation sizes are set to fit in 1”x1” squares for this to work. The view scale is set to 12” = 1’-0”.



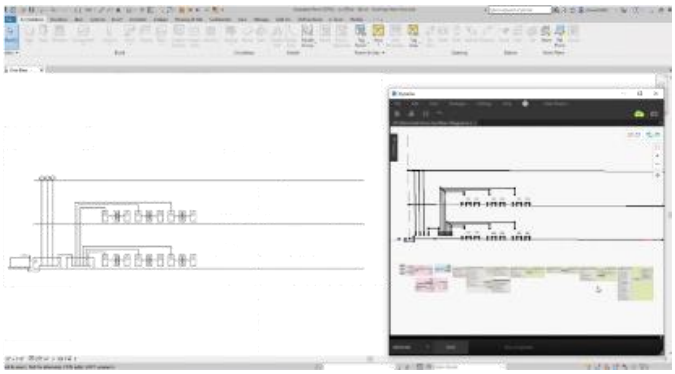




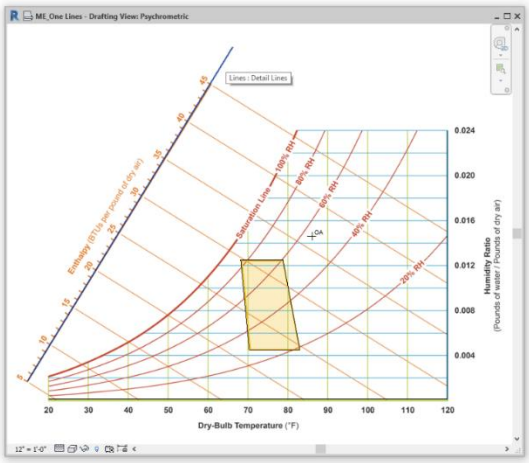
A diagram can be set up for all the AHU used in the project. First, a new drafting view is created for each AHU. Then, Generic Annotations that make up the equipment are logically placed. All the text on the diagram is done with Labels tied to parameters.



With mechanical equipment placed and assigned to systems, riser diagrams can be created with Dynamo. This process is quite a bit more complex to automate compared to the AHU diagram but can be broken down into straightforward subproblems. The first step collects all the mechanical equipment families of a specified chilled water system and groups them by their host level. The second step sorts them by the X component of their location within the 3D model. The third step makes a series of points for placing the Generic Annotation that will represent the family and shift the points up to their level elevation. The fourth step places the detail line



for the levels, adds text notes for the level names, and places the equipment annotation. The last step connects the equipment with detailed lines.



This next example explains how the creation of a Psychometric chart can be automated for each system. The result is a Revit embed Psychometric chat where points can be accurately plotted using Generic Annotations and be connected to coils in the model. The script takes advantage of Dynamo's ability to handle geometry. A picture of the Psychometric chart is placed inside a drafting view, and detailed lines are drawn over all the charts axes. Dynamo can then find the closest point on each detail line from any location point on the graph. The parameter can then be mapped to match the scale of that axis. Generic annotations can then be placed at a specified location on the chart.

Pressure Drop Calculations

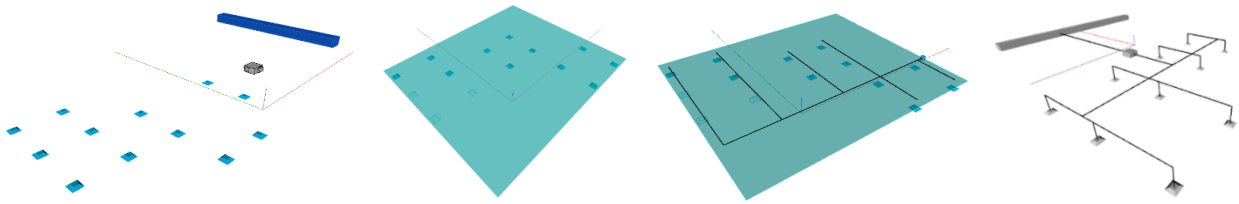
With mechanical equipment placed and systems assigned, routing can be approximated to get an estimated fan static pressure and pump head. System pressure losses are critical for good energy simulations since the pumps and fan energy consumption depend on the pressure of the system. Accurately routing and calculating the pressure drop is an overly complex problem and depends on the length, size, velocity, duct fitting lookup tables, and roughness of materials. Revit can perform these calculations, but the system needs to be fully routed and connected. Revit has built-in tools for finding auto-routing but is rather elementary at best.

Graph theory provides a handy set of tools for leveraging the 3D model for estimating system pressure drop for the initial energy simulations. Implementing all the calculations would lead to a slow algorithm. Therefore, the right balance between simplicity and accuracy is needed. The idea here is that a reasonable estimate is better than using the Revit System Analysis default values.

The first step is to create a maze from the model data. In this example, Mechanical, Electrical, corridors, and open offices are Spaces where ducts and pipes can be routed. Spaces with these Space Types that are assigned to a specified System are collected from the model and their bounding curves are converted to surfaces.

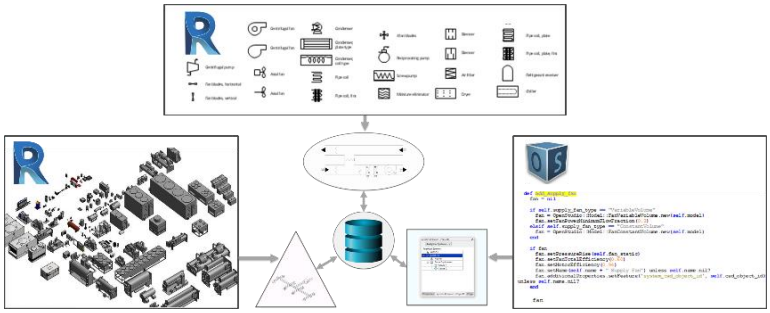
The second step joins the surfaces together. A specified elevation and the intersection with a grid is calculated. Lines that do not intersect the grid are removed. The grid is centered at the Air Handling Unit location point.The third step finds the closest point from the VAV Boxes to the maze of lines. The fourth step utilizes the minimum spanning tree algorithm to route the main ducts from AHU to VAV Boxes. This custom node, based on a found python code, outputs a wire representing the Minimum Spanning Tree of the input graph. The sixth step finds the VAV Box with the largest path back to the AHU. The diffuser is used to create the branch system layout.

The CFM can then be calculated for each edge of the tree graph. By assigning a velocity to the edges, the estimated frictional pressure drop for each edge can be calculated. The pressure loss can be calculated for the vertices using lookup tables. Pressure drops from equipment can be collected from the elements.



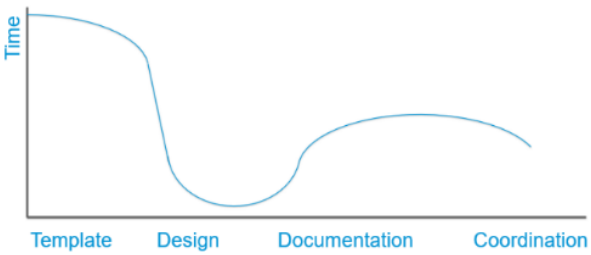
This next chapter shows how Dynamo can turn Revit into a powerful design tool for engineering. With computational logic and access to the Revit API, the user can rapidly configure energy simulation, and results can quickly be integrated into the model in a standardized way. Using these methods requires new skills and new ways to think about design, but the payoff is huge. The next chapter demonstrates how to integrate all these automated techniques together in clear and streamlined workflows.

Integrated Systems HVAC Systems



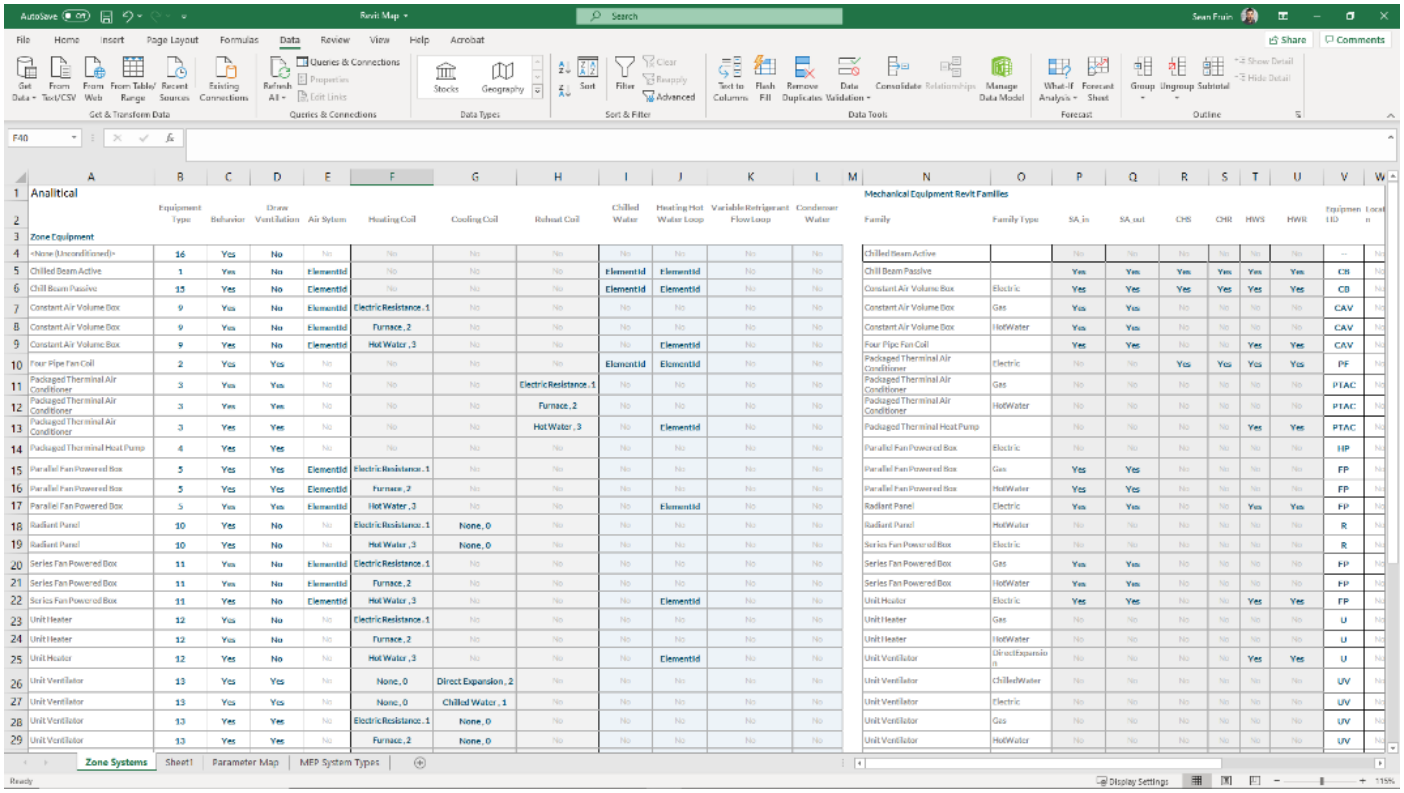
By encoding design logic into Dynamo graphs, processes can be linked together to create a true BIM experience. All the developments in the previous chapters open the door for creating customized, streamlined, and integrated workflows. Different combinations of designer inputs, generative design workflows, and documentation techniques blend in user-friendly workflows. These automation systems can achieve better results with less human intervention than the standard point and click workflows we have grown accustomed to. This final chapter looks at a few of these integrated systems for simultaneously setting up a Revit project and conducting thermal loads, configuring analytical systems for EnergyPlus simulations, and integrating the design into a Revit project in a consistent and connected fashion.

Project Template



Integrated workflows start before a project gets off the ground. The idea here is to put significant time and care upfront to save considerable time on all future projects. This involves clearly mapping out the process from start to finish! Project requirements (sheets, diagrams, and schedule), needed inputs, and steps required to get to the deliverables need to be identified. The next step is establishing a well-organized set of standardized Revit content. The content includes Revit families, Revit MEP settings, Revit View Templates, OpenStudio Measures, OpenStudio Workflows, and, most importantly, Shared Parameters. In short, the idea is to ensure everything is in its place. Then, and only then, can the Dynamo scripts be created that eliminate manual tasks for the end-users without errors. Making all the content was an overwhelming and challenging task at first and is still a work in progress. Smoothly integrating Revit Systems Analysis into our workflows required us to step back and think about our Revit content.

An Excel spreadsheet was set up to help map out all the requirements and connections between all the Revit content for Air Systems, Water Loops, Zone Equipment, Annotations, and MEP families. After a lot of consideration, the equipment from Air Systems, Water Loops, and Zone Equipment became the starting point for making the other content. First, every possible variation of the Zone Equipment and central plant equipment that make up the Air Systems and Water Loops was recorded in column A of the Excel sheet below. The second step recorded all the parameter values that needed to be set to produce each equipment variation. The third step documented each Revit family that would correspond to the Analytical Systems. That information included Family Names, Type Names, required MEP connectors, Equipment designations, and required parameters. The fourth step involved the same process for the corresponding Generic Annotations for the 2D diagrams.



	A	B	C	D	E	F	G	H	I	J	K	L
1	Analitical											
2		Equipment Type	Behavior	Draw Ventilation	Air System	Heating Coil	Cooling Coil	Reheat Coil	Chilled Water	Heating Hot Water Loop	Variable Refrigerant Flow Loop	Condenser Water
3	Zone Equipment											
4	<None (Unconditioned)>	16	Yes	No	No	No	No	No	No	No	No	No
5	Chilled Beam Active	1	Yes	No	ElementId	No	No	No	ElementId	ElementId	No	No
6	Chill Beam Passive	15	Yes	No	ElementId	No	No	No	ElementId	ElementId	No	No
7	Constant Air Volume Box	9	Yes	No	ElementId	Electric Resistance .1	No	No	No	No	No	No
8	Constant Air Volume Box	9	Yes	No	ElementId	Furnace , 2	No	No	No	No	No	No
9	Constant Air Volume Box	9	Yes	No	ElementId	Hot Water , 3	No	No	No	ElementId	No	No
10	Four Pipe Fan Coil	2	Yes	Yes	No	No	No	No	ElementId	ElementId	No	No
11	Packaged Therminal Air Conditioner	3	Yes	Yes	No	No	No	Electric Resistance .1	No	No	No	No
12	Packaged Therminal Air Conditioner	3	Yes	Yes	No	No	No	Furnace , 2	No	No	No	No
13	Packaged Therminal Air Conditioner	3	Yes	Yes	No	No	No	Hot Water , 3	No	ElementId	No	No
14	Packaged Therminal Heat Pump	4	Yes	Yes	No	No	No	No	No	No	No	No
15	Parallel Fan Powered Box	5	Yes	Yes	ElementId	Electric Resistance .1	No	No	No	No	No	No
16	Parallel Fan Powered Box	5	Yes	Yes	ElementId	Furnace , 2	No	No	No	No	No	No
17	Parallel Fan Powered Box	5	Yes	Yes	ElementId	Hot Water , 3	No	No	No	ElementId	No	No
18	Radiant Panel	10	Yes	No	No	Electric Resistance .1	None, 0	No	No	No	No	No
19	Radiant Panel	10	Yes	No	No	Hot Water , 3	None, 0	No	No	No	No	No
20	Series Fan Powered Box	11	Yes	No	ElementId	Electric Resistance .1	No	No	No	No	No	No
21	Series Fan Powered Box	11	Yes	No	ElementId	Furnace , 2	No	No	No	No	No	No
22	Series Fan Powered Box	11	Yes	No	ElementId	Hot Water , 3	No	No	No	ElementId	No	No
23	Unit Heater	12	Yes	No	No	Electric Resistance .1	No	No	No	No	No	No
24	Unit Heater	12	Yes	No	No	Furnace , 2	No	No	No	No	No	No
25	Unit Heater	12	Yes	No	No	Hot Water , 3	No	No	No	ElementId	No	No
26	Unit Ventilator	13	Yes	Yes	No	None, 0	Direct Expansion , 2	No	No	No	No	No
27	Unit Ventilator	13	Yes	Yes	No	None, 0	Chilled Water , 1	No	No	No	No	No
28	Unit Ventilator	13	Yes	Yes	No	Electric Resistance .1	None, 0	No	No	No	No	No
29	Unit Ventilator	13	Yes	Yes	No	Furnace , 2	None, 0	No	No	No	No	No

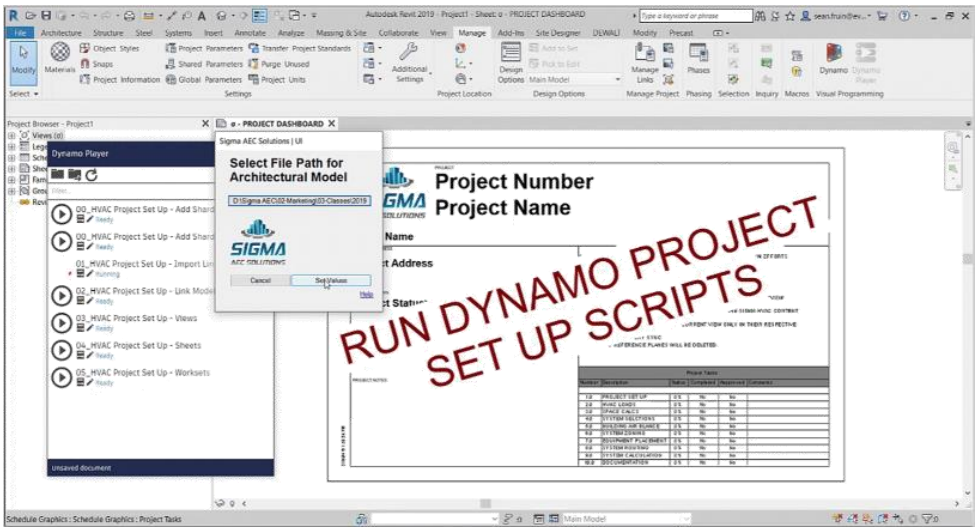
	M	N	O	P	Q	R	S	T	U	V	W
1		Mechanical Equipment Revit Families									
2		Family	Family Type	SA_in	SA_out	CHS	CHR	HWS	HWR	Equipment ID	Location
3											
4		Chilled Beam Active		No	No	No	No	No	No	--	No
5		Chill Beam Passive		Yes	Yes	Yes	Yes	Yes	Yes	CB	No
6		Constant Air Volume Box	Electric	Yes	Yes	Yes	Yes	Yes	Yes	CB	No
7		Constant Air Volume Box	Gas	Yes	Yes	No	No	No	No	CAV	No
8		Constant Air Volume Box	HotWater	Yes	Yes	No	No	No	No	CAV	No
9		Four Pipe Fan Coil		Yes	Yes	No	No	Yes	Yes	CAV	No
10		Packaged Therminal Air Conditioner	Electric	No	No	Yes	Yes	Yes	Yes	PF	No
11		Packaged Therminal Air Conditioner	Gas	No	No	No	No	No	No	PTAC	No
12		Packaged Therminal Air Conditioner	HotWater	No	No	No	No	No	No	PTAC	No
13		Packaged Therminal Heat Pump		No	No	No	No	Yes	Yes	PTAC	No
14		Parallel Fan Powered Box	Electric	No	No	No	No	No	No	HP	No
15		Parallel Fan Powered Box	Gas	Yes	Yes	No	No	No	No	FP	No
16		Parallel Fan Powered Box	HotWater	Yes	Yes	No	No	No	No	FP	No
17		Radiant Panel	Electric	Yes	Yes	No	No	Yes	Yes	FP	No
18		Radiant Panel	HotWater	No	No	No	No	No	No	R	No
19		Series Fan Powered Box	Electric	No	No	No	No	No	No	R	No
20		Series Fan Powered Box	Gas	Yes	Yes	No	No	No	No	FP	No
21		Series Fan Powered Box	HotWater	Yes	Yes	No	No	No	No	FP	No
22		Unit Heater	Electric	Yes	Yes	No	No	Yes	Yes	FP	No
23		Unit Heater	Gas	No	No	No	No	No	No	U	No
24		Unit Heater	HotWater	No	No	No	No	No	No	U	No
25		Unit Ventilator	DirectExpansion	No	No	No	No	Yes	Yes	U	No
26		Unit Ventilator	ChilledWater	No	No	No	No	No	No	UV	No
27		Unit Ventilator	Electric	No	No	No	No	No	No	UV	No
28		Unit Ventilator	Gas	No	No	No	No	No	No	UV	No
29		Unit Ventilator	HotWater	No	No	No	No	No	No	UV	No

Another critical component of the template is to set up the Space Types. Required code compliance information (air changes per hour, heating setpoints, outdoor air per person etc.) and internal load information (infiltration, Occupancy Schedule, lighting load density, etc.) are stored. In the example below, Occupancy Categories from ASHRAE 62.1 were converted into Revit Space Types.

The customization of the OpenStudio Workflows and Measures also need to be established in the Revit template. Customized workflow to perform ASHRAE 90.1 baseline model and load calculations can be created and stored in the template. The shared parameters that allow the connections between Revit and OpenStudio need to be added to Air Systems, Zone Equipment, and Water Loops. Then, the OpenStudio Equipment measures require editing to pick up these variables. Laying down this groundwork is vital for successful automated and integrated workflows.

Project Setup and Load Calculations

With a well-crafted Architectural model, a solid template, and a few inputs from a designer, almost the entire process of setting up a Revit model and running thermal loads can be automated. Aside from saving a lot of time, the results ensure a consistent starting point, standardized process, and a slim how-to manual. Once a new project is created, the first step is to Run a Dynamo script that has the end-user fill in the required information to set up the model. A series of UIs is presented to the end-user.



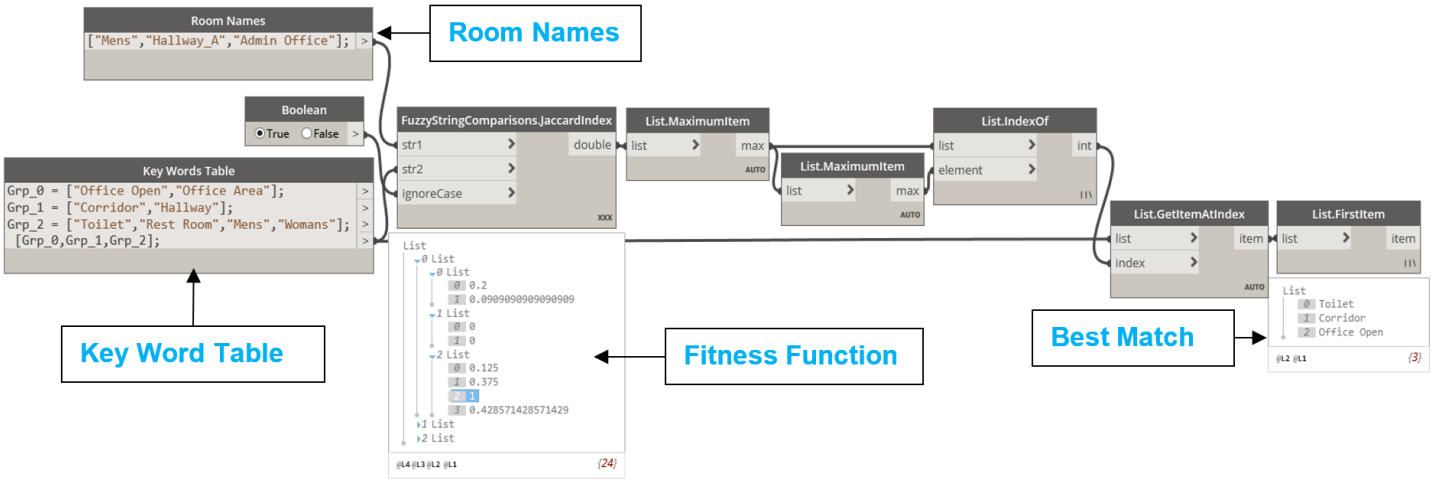
The first UI has the user select the architecture model. The second UI collects information from the linked model to choose the levels that should be copied and what Phases should be used for creating views and sheets. A third UI pops up, asking the designer to set the thermal properties of the building material. The last UI is the Revit Project Location. The script ends with a reminder to create the needed Phases for the project. (There is no way to automate Phase creation with the Revit API).

Once the phases have been manually created, the next script can be run to set up Rooms and Spaces according to the linked model's Rooms. The script makes one Room for every Room in the linked model and makes one Space or many Spaces based on the Room’s area. Spaces are then added to any leftover areas like chases.

Since Revit Spaces will be used for engineering calculations like ventilation, it is important to make sure volumes are correct. There are a few parameters that control this value. One that will need to be edited is the “Limit Offset” parameter. This parameter is set to 8ft as the default. For example, this parameter can be set to 10ft to cover code compliance for ventilation calculation for UFC.

To analyze Revit Spaces for heating and cooling loads for a Systems Analysis workflow, the Space Type property must be assigned to each appropriate Space. This process is a bit time-consuming, as it involves selecting the space, then selecting the Space Type button, and then picking from a long list of Space Types. The fuzzy logic tool is used to find the best matching Space Type. If the best score is below 80%, Space is kept at the assigned building Space Type. This script ends by reporting the

number of Rooms and Spaces created and the number of Spaces that need review due to low fuzzy logic score or geometry issues like self-intersecting bounding curves.



In the background, these two scripts perform a variety of tasks: setting up levels, views, sheets, and keynote schedules, placing views and schedules on the sheets, creating Roomsand Spaces, setting weather and thermal data for load calculations, and creating an energy model and running the customized Systems Analysis workflow for cooling and heating load calculations. Manually doing this process typically takes a couple of days to a week, depending on the project's size. Building an integrated and automated workflow cuts down the process to a day, no matter the project's size. Note that the Systems Analysis for the larger project may take hours but can be run in the background allowing work to still beperformed.

Mini - Schedule: Analytical Spaces

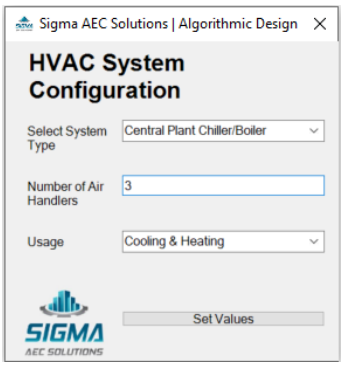
<Analytical Spaces>

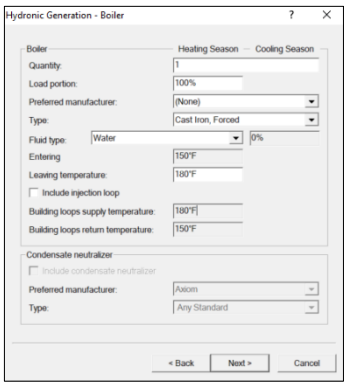
A	B	C	D
Room Name	Peak Latent Cooling Load (Btu/hr)	Peak Cooling Load (Btu/hr)	Peak Heating Load (Btu/hr)
Space 5	471.3	5091.1	3566.1
Space 6	702.9	7593.3	5318.8
Space 8	139.1	1503.0	1052.8
Space 7	731.8	4923.5	5887.5
Space 1	182.2	1968.0	1378.5
Space 2	229.7	2481.4	1738.1
Space 4	417.4	4509.3	3158.5
Space 3	564.8	6100.8	4273.3
Analytical Space 1	0.0	0.0	0.0
Analytical Space 3	0.0	0.0	0.0
Analytical Space 2	0.0	0.0	0.0
Analytical Space 4	0.0	0.0	0.0

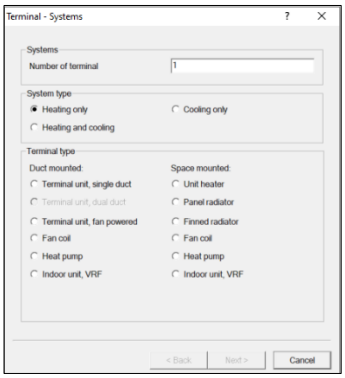
Rapid System Configurations

Once the Revit model is set up and thermal loads have been done, Space airflow calculations, thermal zoning, and HVAC system selection can be performed. The appropriate HVAC system selection for a given building is affected by climate, available utilities, geometric constraints, cost vs. efficiency consideration, and designer preferences. Finding the best solution requires evaluating many alternatives based on system energy consumption and lifecycle cost. With an algorithmic approach to the selection process, several design options can be rapidly created and sent to EnergyPlus for comprehensive energy simulations. This section outlines algorithmic strategies to streamline this process. The combination of the generative design frameworks, Dynamo geometry Systems, and the custom Systems Analysis nodes make this possible.

One way to approach the configuration of different systems is to present designers with a series of UIs so the algorithms can capture their preferences. The UIs below guide a designer in setting up the Zone Equipment, Air Systems, and Water Loops. After the designer has filled in the UI, the script sets up the Analytical Systems for an Energy Analysis annual Building simulation.







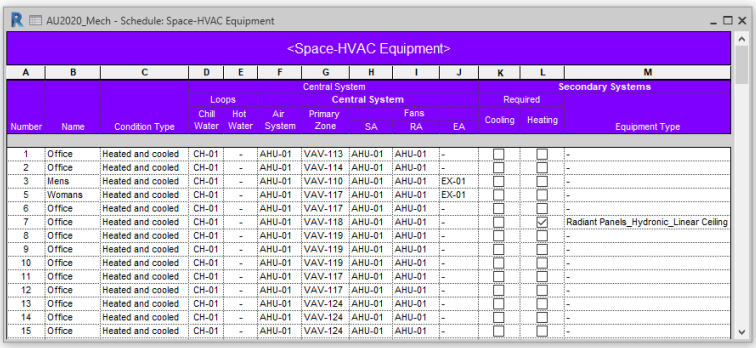
Another approach is creating a Revit 2021 Generative Design Study, where all the needed inputs are filled via the Generative design UI. This method has two advantages. First, the designer can specify goals, optimize for the goal, and evaluate designs through geometric and numerical results. Second, Dynamo scripts run much faster using the generative design framework. This leads to more complexity being added, such as pressure drop estimation. The designer specified system can then be integrated into analytical systems for energy analysis.

Dynamo can rapidly configure different HVAC system configurations, run energy simulations with EnergyPlus, and integrate the best design into the project. Owners can quickly be presented with systems alternative back up with life cycle cost analysis.

Connected 2D, 3D, and Analytical Systems

A geometry system is used to develop a design option and 2D annotations in newly created drafting Views. Revit creates 2D views and 3D views using Dynamo. This step provides the design with a starting put and can be overwritten at any time.

The result is recorded in Shared Parameters added to the Space elements to have a single source of truth and break up the scripts to limit computational overload issues. An example of a Space Schedule showing this information is on the right.



AU2020_Mech - Schedule: Space-HVAC Equipment

<Space-HVAC Equipment>

A	B	C	D	E	F	G	H	I	J	K	L	M
			Central System									Secondary Systems
			Loops		Central System					Required		
Number	Name	Condition Type	Chill Water	Hot Water	Air System	Primary Zone	SA	Fans RA	EA	Cooling	Heating	Equipment Type
1	Office	Heated and cooled	CH-01	-	AHU-01	VAV-113	AHU-01	AHU-01	-	☐	☐	-
2	Office	Heated and cooled	CH-01	-	AHU-01	VAV-114	AHU-01	AHU-01	-	☐	☐	-
3	Mens	Heated and cooled	CH-01	-	AHU-01	VAV-110	AHU-01	AHU-01	EX-01	☐	☐	-
5	Womans	Heated and cooled	CH-01	-	AHU-01	VAV-117	AHU-01	AHU-01	EX-01	☐	☐	-
6	Office	Heated and cooled	CH-01	-	AHU-01	VAV-117	AHU-01	AHU-01	-	☐	☐	-
7	Office	Heated and cooled	CH-01	-	AHU-01	VAV-118	AHU-01	AHU-01	-	☐	☑	Radiant Panels_Hydronic_Linear Ceiling
8	Office	Heated and cooled	CH-01	-	AHU-01	VAV-119	AHU-01	AHU-01	-	☐	☐	-
9	Office	Heated and cooled	CH-01	-	AHU-01	VAV-119	AHU-01	AHU-01	-	☐	☐	-
10	Office	Heated and cooled	CH-01	-	AHU-01	VAV-119	AHU-01	AHU-01	-	☐	☐	-
11	Office	Heated and cooled	CH-01	-	AHU-01	VAV-117	AHU-01	AHU-01	-	☐	☐	-
12	Office	Heated and cooled	CH-01	-	AHU-01	VAV-117	AHU-01	AHU-01	-	☐	☐	-
13	Office	Heated and cooled	CH-01	-	AHU-01	VAV-124	AHU-01	AHU-01	-	☐	☐	-
14	Office	Heated and cooled	CH-01	-	AHU-01	VAV-124	AHU-01	AHU-01	-	☐	☐	-
15	Office	Heated and cooled	CH-01	-	AHU-01	VAV-124	AHU-01	AHU-01	-	☐	☐	-

Once an option is selected, the MEP Systems and 2D schematics quickly get incorporated into the project. An often overlooked benefit of automating setting up the Analytical System and placing the 2D and 3D elements into the model is that the connections between the three frameworks are established. This creates a fully connected and parametric model.

An example of utilizing this is to create the 2D schematics into Intelligence Dashboard. The 2D schematics provide the best holistic view of the system in order to find the best location to establish a single truth source. The Generic Annotations that make up the schematics are used to store and display data. Required inputs can with entered with labels. The data can then be transmitted between other elements with a run of a Dynamo script. Once the right system is found, it gets integrated into the project for further refinement and documentation. As seen in the previous chapter, several of these steps can be automated.

There are endless possibilities for configuring different workflows. Ask five other engineers what their approach to designing a system is and you will get five different answers. Nevertheless, there are plenty of automation opportunities, which streamline the process and create customized, user-friendly workflows. Revit is the way it is for this exact reason.

While adding the 2D annotation, 3D models, and Analytical Systems into the model, the groundwork is laid down for connecting. Usually, these elements do not have any connection, so when changes are made, we must make updates in multiple places.

Conclusion

Like several workflows in the AEC Collection, the Revit Systems Analysis framework requires considerable setup and takes a fair amount of knowledge of the program to master. Manually navigating all the different Revit tabs to set the variety of inputs necessary is daunting, convoluted, and time-consuming. Moreover, the data generated is detached from other aspects of the model. These frustrations can be eliminated for end-users by setting up a standardized process, eliminating tasks using computational design, and guiding the process by automatically displaying UIs to the designer. Of course, setting up these automated workflows is tricky. It takes multiple skills, good team dynamics, and above all, enthusiasm to break from the status quo.

For individuals, I highly recommend investing time in learning these new and inevitably essential skills. Wading through document pages and watching hours of YouTube videos is good, but the only way to learn is to practice. Whether using nodes or writing code, one must start with the easy tasks like creating views, exchanging Revit and Excel data, and placing elements in a project. This essential first step will familiarize you with the data, giving you insight into how you might manipulate it. Next, focus on connecting different data sources. Interoperability is vital in constructing complex algorithms. Finally, start identifying time-consuming tasks and tackle the low-hanging fruit like project setup. This 3-step process will help you learn to map out a strategy, break the problem into small puzzle pieces, and build multi-step algorithms. Before you know it, your brain will be flooded with new ideas, and you will never look back.

Companies need to boost computer programming skills, embrace lateral thinking, and cut stubborn dead weight within their organizations to get the most out of their costly software. Winning companies will start treating their project data and processes as an asset and begin to eliminate rogue and manual processes. With standardized methods in place, models become large, valuable datasets that reflect different markets' unique project requirements worldwide. The dataset could contain information such as the market's unique manufacturing costs, insights into production times, utility costs and availability, code and other legal requirements, and more. These multiple, robust, and comprehensive datasets will inevitably fuel machine learning which will lead to lucrative improvements and market advantage in perpetuity.

The future of the AEC is not in traditional off-the-shelf software but instead lies in customized workflows that use robust APIs. Autodesk has invested in and continues to improve its APIs to fulfill its promise of building more with less. Automation will ultimately replace most design functions now performed by people. Now it is up to the industry to embrace new approaches to solving design challenges. So, get to programming, embrace change in your organizations, and ultimately set yourself up for future success.