BUILDING *performance* LAB

Summary Report 1.2.3.c. – Research Viability of Generating Energy Models for the Entire Municipal Building Portfolio

By:
Amanda O'Donnell | Sr. Engineer
Barrett Trenholme | Jr. Engineer
Omar Hamad | Jr. Engineer

Reviewed by:
Duncan Prahl, RA, AIA | Director, Technical Services
Honey Berk, LEED AP O+M, CMVP | Executive Director

Original submission:
August 22, 2023

CIUS
CUNY INSTITUTE FOR URBAN SYSTEMS
THE CITY UNIVERSITY OF NEW YORK

City College of New York
160 Convent Avenue
Marshak Science Building
Room 118
New York, NY 10031

96 Greenwich Street
4th floor
New York, NY 10006
cunybpl.org

Dr. Robert E. Paaswell
Executive Director, CIUS
Michael Bobker
Associate Director, CIUS
Honey Berk
Executive Director, CUNY BPL

Table of Contents

Table of Figures

Introduction

As increasing numbers of cities are implementing limits to carbon emissions for buildings, there is a growing need for 8,760-hour Building Energy Models (BEMs) as an asset management tool, not simply as as a means of achieving code compliance. Presently, preparing BEMs for mainstream modeling engines (e.g., EnergyPlus) requires inputting and connecting vast amounts of data in text-based files, which is inefficient and non-replicable. Graphical User Interfaces (GUIs) such as OpenStudio (OS) are mere "wrappers" of EnergyPlus (E+) objects and features, and they are limited in how much they simplify the BEM setup process. Standalone BEM programs have limited interoperability with geometrical design and building information modeling (BIM) programs, adding redundancy to the design process by requiring the separate creation of BEM-compatible geometries. BEMs are typically generated and retained by consultant teams, and not used after design and construction. Cities, like New York, are making building-related data (e.g., geometry, ASHRAE Level 2 audits, utility data) publicly available. An automated process to translate this data into building-specific energy models can simplify the process of BEM development and utility for building owners, ideally allowing for a scalable, consistent analysis of pre- and post-retrofit measures.

Project Description and Objective

This project, titled AutoGenerateBEM, seeks to generate and calibrate BEMs in an automated or semi-automated fashion, using publicly available and/or private building data (e.g., geometry, systems info, operation schedules, property attributes) collected from energy audits and other sources. By leveraging open-source BEM software tools and public data sources, the intent of the project is to produce a prototype BEM workflow and to evaluate it holistically in the context of running and calibrating BEMs for a large portfolio of municipal buildings.

In FY23, the CUNY Building Performance Lab (CUNY BPL) developed this workflow and is currently testing it on a portfolio of municipal buildings provided by the NYC Department of Citywide Administrative Services Division of Energy Management (DEM). The long-term goal is for the workflow to be applicable to any portfolio of buildings with valid geometry and building and energy attribute data.

Project Deliverable

The work from this project is hosted on CUNY BPL's GitHub repository. The repository is supplemented with documentation and example files to assist users in utilizing and reproducing every aspect of the project's functionality. The repository can be made available upon request to CUNY BPL.

Software

The workflow is achieved via import, export, file translation, and execution capabilities in different software tools, including ArcGIS Pro, Rhino 3D, Grasshopper, OpenStudio (OS), and others.

ArcGIS Pro is a geographic mapping software suite that allows manipulation of 2D and 3D data. ArcGIS has integrated Python scripting support and was used for automating 3D building geometry extraction.

Rhino 3D is a 3D computer graphics and computer-aided design (CAD) application that is used primarily to create and edit 3D geometry. For this project, key functionalities were the ability to read building geometry generated from ArcGIS and to integrate with the Grasshopper plug-in.

Grasshopper is a visual programing design tool that can be used for analyzing and manipulating 3D models in Rhino 3D via scripting. Grasshopper can be extended through many open-source plug-ins including Ladybug, Honeybee, and Dragonfly, which were used in this project. Ladybug and its nested program Honeybee allow for the visualization of weather data and the ability to create energy models and output E+ models from the Grasshopper environment. Dragonfly is an energy modeling visualization tool that creates BEMs of entire districts using structured building schemas such as floor plans. For this project, Dragonfly was used to create floors of buildings and external constructions.

OS is an open-source toolkit used to create BEMs that was developed, and is maintained, by several US Department of Energy National Labs. OS wraps the E+ simulation engine and features a GUI, and a separate Parametric Analysis Tool (PAT) for calibration. To run simulations programmatically, OS offers a software development kit (SDK) and a command line interface (CLI). The SDK allows users to author or utilize existing scripts to automate many steps for BEM generation and manipulation; while the CLI allows for quicker interfacing with OS in applying model changes, running simulations, and calibrating models (replacing the PAT).

In this project, the OS SDK was used in a repository of tools and workflows (written in the Ruby programming language) that merge BuildingSync XML (BSXML) files (a standardized schema for building audit data) and the OS energy models containing building geometry created earlier in the workflow. A repository with the needed capacities existed in some form in the BuildingSync Gem developed by the National Renewable Energy Lab (NREL). This project extended NREL's existing tool into a new, separate tool that better served the project's purposes.

Data Sources

Geometry Data Sources

The dataset used in this project was the ESRI 3D Multipatch NYC Layer, downloaded from the NYC Office of Technology and Innovation (OTI) website (NYC Office of Technology & Innovation, 2023). The dataset contains over one million models of building exteriors in 3D format with a model level of definition (LOD) of 1.5. LOD1 is a basic form of 3D modeling that mainly depicts exterior walls and depicts max height as a flat roof. LOD1.5 sits between LOD1 and LOD2, and typically includes sloped roofs, spires, and any curvature in the building (Abbaspour et al., 2017) (see

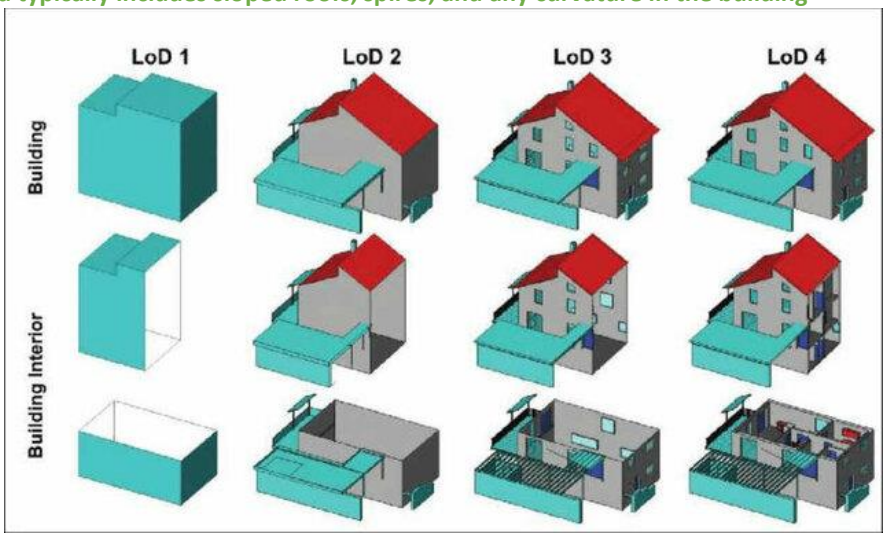

Figure 1).

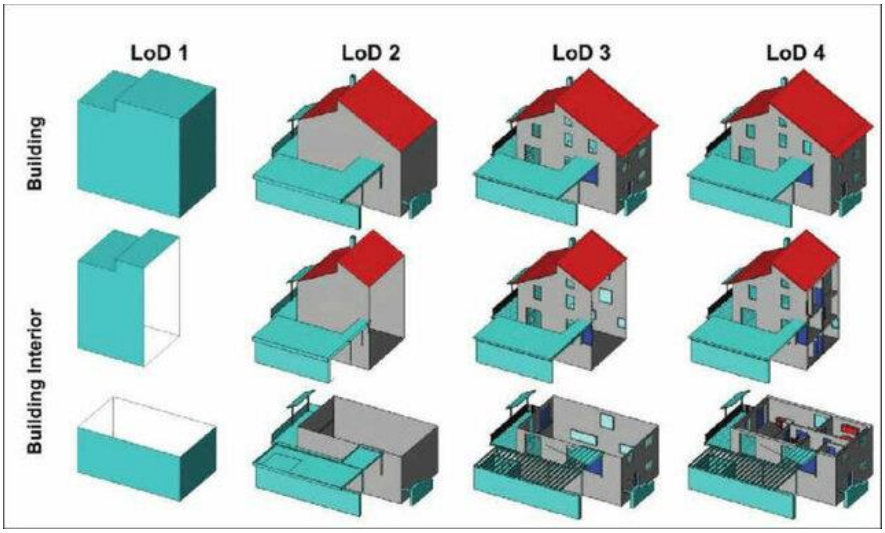

Figure 1. Increasing complexity of LOD in 3D modeling (Source: Karlsruhe Institute of Technology)

Energy Data Sources

Audit Template (AT) is a web-based tool developed by the Pacific Northwest National Lab (PNNL) for the US Department of Energy that follows ASHRAE Standard 211 ("Standard for Commercial Building Energy Audits") for building energy audits. It standardizes energy audit data entry by allowing the auditor to input data in pre-defined fields via a web interface. AT was adopted by NYC several years ago as a data entry interface for Local Law 87 (LL87) compliance audits (i.e., this project's source for building energy data). LL87 audits comprise ASHRAE Level 2 audits and retro-commissioning studies.

As noted in the last section, the BSXML format is a schema for describing building audit data (Maile et al., 2023). BSXML allows for easy data exchange with other applications using the XML format, which interfaces with many existing libraries in common programming languages (e.g., REXML in Ruby).

BSXML is particularly suitable for LL87 audits due to the way in which the major elements map to the components of the audit. The general audit component is described by "Sites" and "Systems" elements: the former maps to the building occupancy and usage information, and the latter maps to constructions, HVAC equipment, lighting, loads, power generation, etc. Similarly, the "Measures" element in BSXML can be used to describe the possible energy conservation measures (ECMs) resulting from the audit.

Workflow

The AutoGenerateBEM workflow is comprised of four main work steps, illustrated in Figure 2.

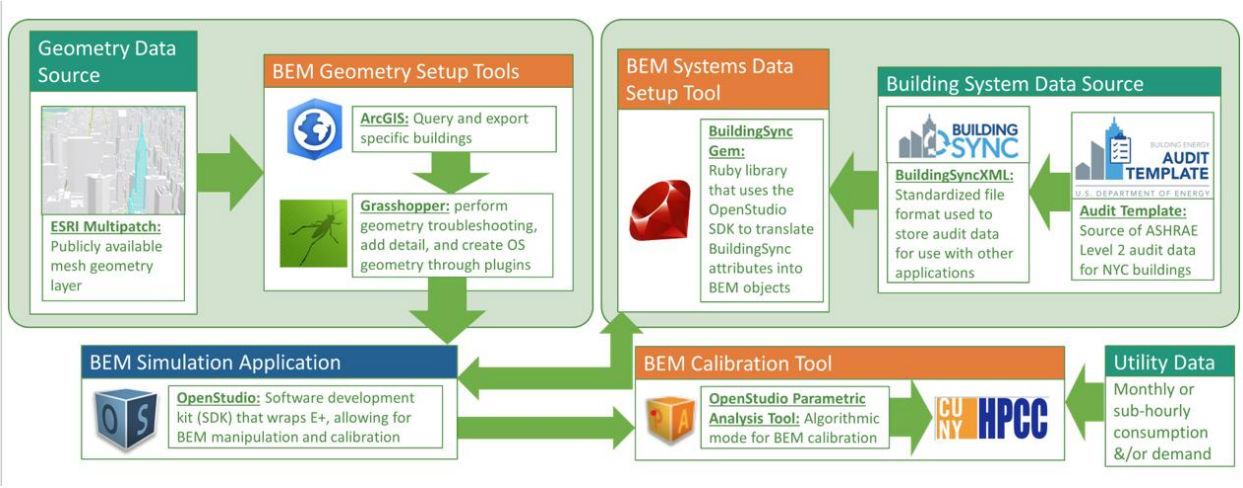

Figure 2. AutoGenerateBEM workflow overview

Steps 1 through 4 were developed separately, so they are automated but they function as individual modules. Work is ongoing to advance the interoperability of the steps and to automate the entire workflow. More information on the execution of each step is provided below.

Step 1 – Extract Building Models from NYC Building Portfolio

The geometry data for the NYC building portfolio, the ESRI 3D Multipatch NYC Layer, was downloaded from the NYC OTI website[1]. ESRI Multipatch[2] features were used as a geometry data source because they can be queried and can store attribute data in conjunction with the individual building's 3D model. Multipatch features are 3D objects that represent a collection of patches that present the boundary (or outer surface) of a 3D feature. Multipatches can be 3D surfaces or 3D solids (volumes) (Penn State, 2021). The 3D models represent New York City in 2014, when the aerial survey was conducted, and they provide models with roof structures and building exteriors.

NYC OTI 3D datasets are separated into 21 different geographical areas. The first step involved combining all 21 geographical areas into one file in ArcGIS Pro. The Multipatch data included the NYC Department of Buildings Building Identification Number (BIN) for every building. Using the BINs allowed for the joining of the geometry data with an external data source in ArcGIS. The base data, contained in an MS Excel spreadsheet, comprised municipal buildings with BSXML data from energy audits conducted for NYC LL87 compliance. This dataset was joined to the 3D buildings data, and the buildings with BSXML data were queried and isolated. Figure 3 illustrates the results of this process.

The final step was to export the isolated Multipatch files from ArcGIS as a CAD-compatible file format (DWG), for use in Rhino 3D in Step 2.

[1] https://www.nyc.gov/content/oti/pages/data-analytics/citywide-data-sharing
[2] https://desktop.arcgis.com/en/arcmap/latest/extensions/3d-analyst/multipatches.htm

Figure 3. 3D rendering of buildings in the dataset with BSXML files, showing Lower Manhattan, NYC

Using Python for ArcGIS (ArcPy), Python scripting was used to automate geometry extraction from the Multipatch files. An iterative workflow, which included matching building geometries to building information databases and subsequent export to CAD, was created and tested. The script used the geometry dataset and a list of BINs as inputs, then outputted the matching buildings as individual DWG export files.

Step 2 – Create BEM, Including Geometry Data Sources

Upon importing the geometry data into Rhino, many inconsistencies were identified, inherently due to the way the buildings were originally modeled. Note that there was no way to tell in advance what the "quality" or complexity of the geometry would look like and, consequently, how much pre-processing would be necessary before moving into BEM generation with Grasshopper. Geometric issues identified by CUNY BPL through this process, to date, were:

- Coordinate systems are not maintained in the translation of geometry file format from ESRI Multipatch to DWG, so geometry was relocated in the process.
- Basement geometry is not modeled in the original geometry data.
- Ground-adjacent floor geometry often has errors in the original geometry data.
- Buildings are modeled with single and multiple volumes per building in the original geometry data, introducing the need for additional steps in BEM creation workflow.
- Buildings with negative spaces, such as courtyards, would likely be misrepresented and modeled as closed volumes in the original geometry data.

CUNY BPL used Grasshopper to read and manipulate geometry to solve the issues detailed above. BSXML files were used to determine the number of below- and above-ground floors for the buildings, and Ladybug, Honeybee, and Dragonfly were used to generate an OS model. This workflow included a Grasshopper visual script, which embeds several python code snippets to interface with necessary BSXML files and other folder structures.

The workflow developed for the Rhino/Grasshopper and Ladybug tools was automated through a Windows batch script, which included toggling initial settings in Rhino and loading the exported DWG

file from ArcGIS for a given number of buildings. A similar shell script was developed for MacOS devices, although it is only capable of handling a single DWG file as an input. The command line scripts are functional initial steps, however a more robust workflow for this automation could be developed and would ideally provide further error-checking and geometry validation.

Step 3 – Add Inputs from Energy and Building Data Sources to BEM

The BuildingSync project sought to develop tools that leverage BSXML files. Of particular importance was the BuildingSync Gem, which is a Ruby library that translates BSXML files to OS models, runs them for simulation, and writes simulation results to output BSXML files. In terms of model generation and manipulation, the Gem uses pre-existing Ruby OS scripting and automation tools developed for other purposes, such as "openstudio-extension" and "openstudio-standards". The latter manipulates BEMs for compliance with building codes such as ASHRAE Standard 90.1, which means the BuildingSync Gem is equipped to populate model input data of an OS model (after translation from BSXML) with default values compatible with its building type, construction year, and other parameters.

However, the existing Gem only supports BSXML files up to MLOD 100[3], with very simplistic geometries and HVAC system mappings that abstract much of the details in the building data. This did not align with CUNY BPL's goal of working with BSXML files from LL87 audits, which are rated at MLOD 200, so it was determined that the Gem would need further development to align with the project goals, described further in the next section, "Geometry Input".

The BuildingSync schema is well-documented for the purpose of creating and understanding single-building energy audits. However, the current project was designed to work on a large portfolio of municipal buildings, so the first obstacle was the lack a tool that could be used to study the contents of large sets of BSXML files. A summary of elements and element values would be required to understand the general nature of the different aspects of the files (e.g., HVAC systems and attributes). Also, the ability to query files with unique identifiers (e.g., BIN, gross floor area) and store them in separate directories would be needed. Accordingly, several querying and file management tools were developed for the purpose of counting the occurrences of a specific element across the set of BSXML files and writing the summary to a CSV file. The insight gained from such summaries was used to inform the modifications required for the Gem to support MLOD 200 BSXML files (i.e., those exported from AT for LL87 audits); and the ability to query and move sets of BSXML files will be useful for any future Gem users or developers working with portfolios of buildings.

Modifying the Gem addressed the key aspects of using the BSXML data for: 1) creating representative space types for the BEM; 2) applying the thermal properties and window-to-wall-ratio on the envelope; and 3) creating detailed HVAC systems, described further in the next sections. These enhancements greatly expanded on the original BuildingSync Gem; for example, instead of creating an abstracted system based on a "primary system type" attribute for a whole building, the Gem modifications allow for the reading of subcomponents of HVAC system elements in the BSXML file (see Figure 13 in the Appendix for a comprehensive view of HVAC systems breakdown in Audit Template BSXML files). This enables the construction of specific systems that can then be tuned to best reflect the actual systems in a building; for example, the script chooses the appropriate heating source (e.g., district heating) based on actual data points in the input BSXML file.

[3] CityGML refers to level of detail as LOD, with a range of LOD1 to LOD4; BuildingSync (BSXML)uses the BIM IFC standard, which uses model level of detail, or MLOD, with a range of LOD 000 to LOD 500.

Finally, the “openstudio-standards” tool incorporates logic into the Gem that sets appropriate default values for BEM attributes when no corresponding data is found in the BSXML (e.g., using the default constructions for ASHRAE 90.1-2004 for a building constructed in 2005, but its BSXML file contains no information on building envelope).

The model generation workflow associated with Step 3 is summarized in Figure 4.

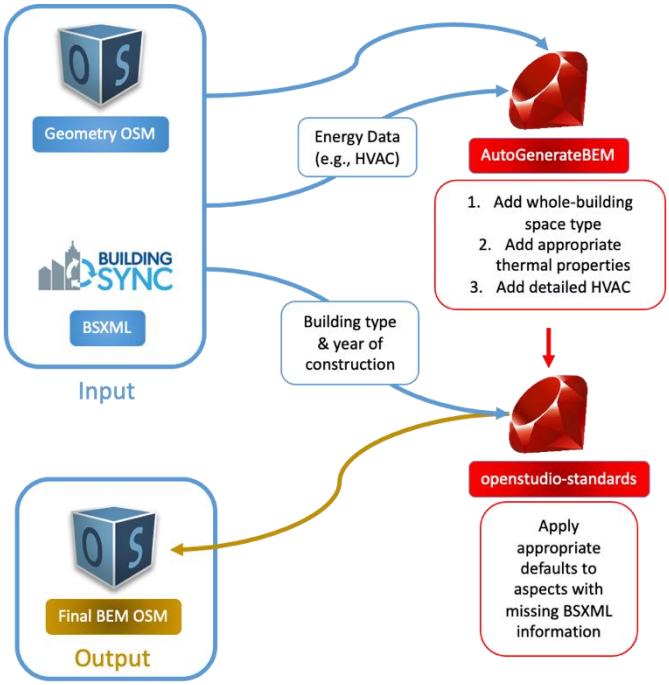

Figure 4. Summary of model generation workflow in the AutoGenerateBEM tool

Geometry Input

As described previously, a workflow to create BEM geometry was developed and the BEM geometry was meant to be the input of a BSXML workflow, along with the BSXML file. This differs from the original BuildingSync Gem in how it receives a single input (the BSXML file) and constructs an assumed geometry. See Figure 5 for the code snippet for this step of the workflow.

```ruby
# @param osmfolderorfile [String] : path to folder containing OSMs generated from geometry workflow
## filenames must contain BIN
# @param bin [String] : (Optional) Building Identification Number used to open the associated OSM
# @return model [OpenStudio::Model]
def open_OSM(osmfolderorfile,bin="")
  path = ""
  if osmfolderorfile.include? ".osm"
    path = OpenStudio::Path.new(osmfolderorfile)
  else
    osm = Dir.glob(["#{osmfolderorfile}/*#{bin}*.osm"]).select {|path| !path.include? "AutoBEM_modified"}
    puts "Warning: Multiple OSMs found in #{osmfolderorfile} with BIN #{bin}." if osm.length > 1
    path = OpenStudio::Path.new(osm.first)
  end

  translator = OpenStudio::OSVersion::VersionTranslator.new
  return model = translator.loadModel(path).get
end
```

Figure 5. Code for the function to import an existing OSM containing geometry

Currently, the geometry generation workflow can create models with one thermal zone per floor. Future enhancements to the workflow might introduce appropriate thermal zoning and other mechanisms to incorporate proper space division on different floors. More information about the geometry workflow is documented in the wiki located in the GitHub repository.

Building Space Types

Modifying the Gem introduced the ability to define and assign one representative space type for the whole building in the model. The tool reads the building types from `auc:Section` elements in the BSXML files and blends all the types present according to the ratio of each building type's floor area to total building area (see Figure 6).

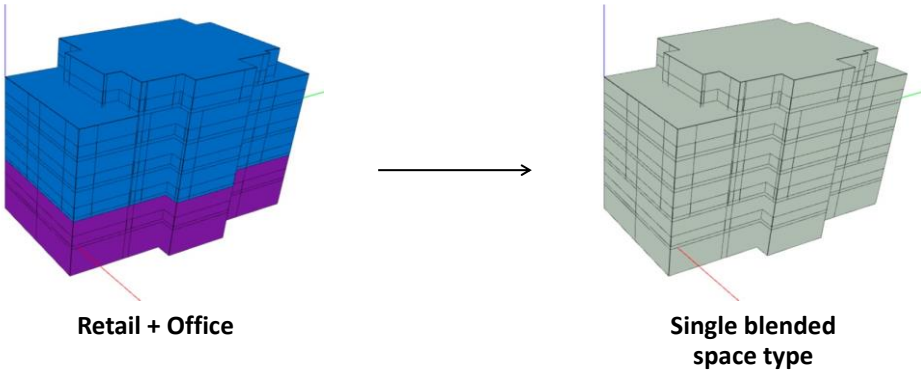

Figure 6. Visualization for the process of blending space types

Thermal Properties for Constructions

The Gem was improved with the capability to read constructions elements from the BSXML files (i.e., `auc:WallSystem, auc:RoofSystem, auc:FenestrationSystem, auc:FoundationSystem`) (see Figure 7). If a U-Value or R-Value is defined for a construction, the tool will add it to the model and assign the corresponding values (see Figure 8 and Figure 9).

```ruby
# read
# @param wall_system [REXML:Element]
# @param ns [String]
def read(wall_system, ns)

  @id = wall_system.attributes['ID'] if wall_system.attributes['ID']

  xmlwallInsulationRValue = XPath.first(wall_system,".//#{ns}:WallInsulationRValue")
  xmlexteriorWallConstruction = XPath.first(wall_system,".//#{ns}
  :ExteriorWallConstruction")
  xmlexteriorWallFinish = XPath.first(wall_system,".//#{ns}:ExteriorWallFinish")

  @wallInsulationRValue = help_get_text_value_as_float(xmlwallInsulationRValue) unless
  xmlwallInsulationRValue.nil?
  @exteriorWallConstruction = help_get_text_value(xmlexteriorWallConstruction) unless
  xmlexteriorWallConstruction.nil?
  @exteriorWallFinish = help_get_text_value(xmlexteriorWallFinish) unless
  xmlexteriorWallFinish.nil?
end
```

Figure 7. Code for an example function to read properties of a wall from the BSXML file

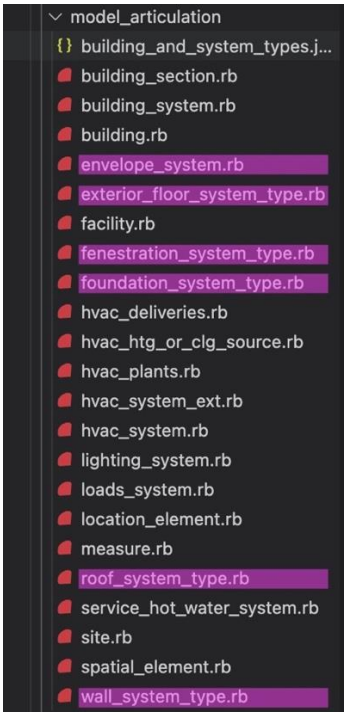

Figure 8. Classes in the tool pertaining to the envelope components

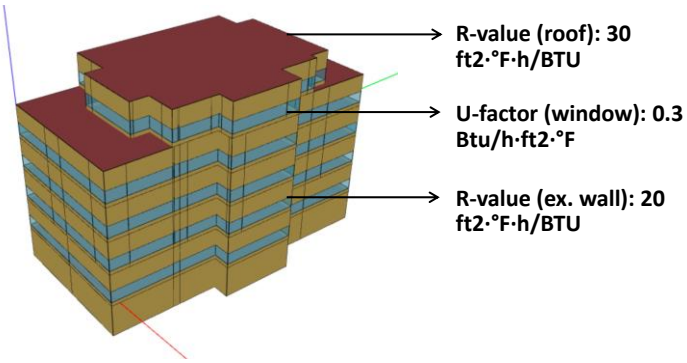

Figure 9. Representation of the thermal properties assigned in the BEM

Detailed HVAC Creation

In what was perhaps the biggest undertaking of this project, the Gem was modified to read almost all the components and attributes described in a BSXML file to constitute an HVAC system. This differs from the original functionality, which was limited to a description of the building's primary HVAC system type without any further details. Implementation of system creation based on cooling and heating sources and plant components has been completed (see Figure 10, Figure 11, and Figure 12), while the full implementation of system creation based on delivery components and pump and fan controls is currently in progress.

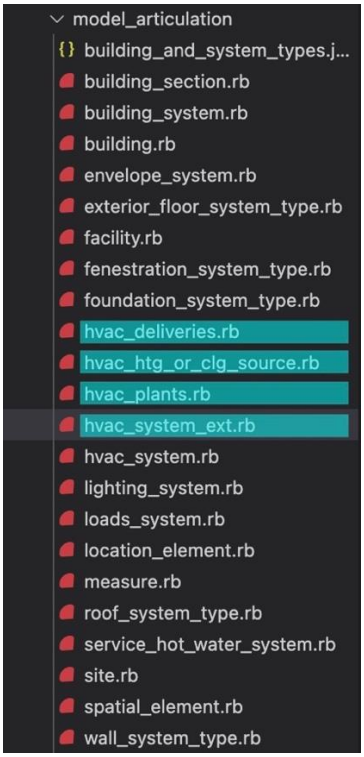

Figure 10. Classes in the tool pertaining to HVAC creation and assignment

```ruby
# read xml & create associated objects
def read_xml(base_xml)
  @id = base_xml.attributes['ID']
  @zoning_system_type = help_get_text_value(base_xml['.//auc:ZoningSystemType'])

  @plants_heating = []
  XPath.match(base_xml,'.//auc:HeatingPlant').each do |htg_plant|
    @plants_heating << BuildingSync::Plant.new(htg_plant)
  end

  @plants_cooling = []
  XPath.match(base_xml,'.//auc:CoolingPlant').each do |clg_plant|
    @plants_cooling << BuildingSync::Plant.new(clg_plant)
  end

  @plants_condenser = []
  XPath.match(base_xml,'.//auc:CondenserPlant').each do |cndns_plant|
    @plants_condenser << BuildingSync::Plant.new(cndns_plant)
  end

  @deliveries = []
  XPath.match(base_xml,'.//auc:Delivery').each do |dlvry|
    @deliveries << BuildingSync::Delivery.new(dlvry)
  end

  @heating_sources = []
  XPath.match(base_xml,'.//auc:HeatingSource').each do |htg_src|
    @heating_sources << BuildingSync::HeatingOrCoolingSource.new(htg_src)
  end

  @cooling_sources = []
  XPath.match(base_xml,'.//auc:CoolingSource').each do |clg_src|
    @cooling_sources << BuildingSync::HeatingOrCoolingSource.new(clg_src)
  end

end
```

Figure 11. Snippet of the HVAC creation code, which creates the components and assigns them to one overarching system

```ruby
# @param osmfolderorfile [String] : path to folder containing OSMs generated from geometry workflow
## filenames must contain BIN
# @param bin [String] : (Optional) Building Identification Number used to open the associated OSM
# @return model [OpenStudio::Model]
def open_OSM(osmfolderorfile,bin="")
  path = ""
  if osmfolderorfile.include? ".osm"
    path = OpenStudio::Path.new(osmfolderorfile)
  else
    osm = Dir.glob(["#{osmfolderorfile}/*#{bin}*.osm"]).select {|path| !path.include? "AutoBEM_modified"}
    puts "Warning: Multiple OSMs found in #{osmfolderorfile} with BIN #{bin}." if osm.length > 1
    path = OpenStudio::Path.new(osm.first)
  end

  translator = OpenStudio::OSVersion::VersionTranslator.new
  return model = translator.loadModel(path).get
end
```

Figure 12. Code for the function to import an existing OSM containing geometry

Step 4 – Calibrate BEM to Available Calibration Criteria

Currently, BEM calibration is largely a manual process performed using utility bills from the same year as the weather file used for the energy model. Building energy usage can be calibrated to a specific year by inputting the exact energy consumption acquired from historic building energy consumption data. This manual method of calibration is time-consuming and takes significant expertise to validate.

In contrast, auto-calibration can take much less time and oversight to produce a calibrated model. Auto-calibration methods primarily involve optimization-based approaches that iteratively run the same BEM hundreds, or even thousands, of times with algorithmically varied inputs until a model calibration is achieved. Due to the potential time difference, auto-calibration is better suited to large-scale BEM efforts than manual calibration, and it has assumed a significantly larger role in research in recent years (Chong et al., 2021).

Although the topic abounds in current research, best practices in auto-calibration (e.g., variables to calibrate, algorithm parameters) are not well defined and more testing will need to be done on the specific data available for NYC buildings. However, there is reason for optimism considering the significant amount of energy audits available.

Initial testing on auto-calibration has involved installing OpenStudio Server (Long et al., 2023) and the OpenStudio Parametric Analysis Tool (PAT). Two BEMs have been successfully calibrated using a basic calibration method as a proof-of-concept for the auto-calibration workflow. However, while the basic calibration met ASHRAE Guideline 14-2014 statistical metrics for calibration, the parameters given in the calibrated models were not realistic representations of the actual building. This is expected with the naïve calibration method; as the workflow is improved, results from auto-calibration should become more realistic and, thus, usable.

Conclusion and Future Work

This project successfully created a prototype workflow for the largely automatic creation of BEM geometry and the fully automated application of audit data from the BuildingSync BSXML format to BEM files. The work is hosted on CUNY BPL's GitHub repository and can be made available to other users or developers.

It is yet to be decided whether this workflow would be best as a "plug and play" approach, where the individual steps can be used separately, versus a wholistic approach, where all the steps are integrated into one single process. Yet, it stands true that this workflow shows great potential for full automation of BEM generation and calibration. This project has focused mainly on the workflow's functionality and less on the analysis of the output; for this, a significant number of case study buildings will need to be run to get a better understanding of how the results of automated BEM generation compare to utility data and other types of measured data.

Additional development will be required to improve the geometry-to-BEM workflow, especially around flexibility needed to handle various databases and building data formats and troubleshooting geometry errors in 3D models. A current limitation is the lack of detailed building spaces and thermal zones implemented in the BEM geometry (due to this data not being available), which greatly limits the usefulness of HVAC data. The current workflow assumes a single thermal zone on each floor, regardless of floor size; this dictates that every added HVAC system serves the whole building, which is not representative of HVAC configurations in a typical building. Several pathways will be explored to address

this issue, including tools to automatically generate spaces based on actual floor plans, generate assumed spaces based on building type and defined HVAC systems and their attributes, or generate assumed spaces based on other building data sources.

Improvement to the auto-calibration workflow will involve the definition of custom OS measures that account for available audit data, as well as replacing the use of the OpenStudio PAT with the OpenStudio SDK for sending calibration jobs to the server. In addition, the OpenStudio Server should be moved to a cloud-based server instead of being hosted locally, to allow for a larger number of calibration tests to be completed.

References

Abbaspour, Rahim & Mirvahabi, Simin S.. (2017). Generation of a Data Model for Indoor Navigation Based on Volunteered Geospatial Information (VGI). 10.4018/978-1-5225-2446-5.ch013.

Chong, A., Gu, Y., Jia, H. (2021). Calibrating building energy simulation models: A review of the basics to guide future work. Energy and Buildings, 253(9):111533. https://doi.org/10.1016/j.enbuild.2021.111533

Long, N., Ball, B., Petersen, A., Horsey, R., Coleman, T., Fleming, K., Swindler, A., Macumber, D., Hale, E., Weaver, E., Sparke, A., Benne, K., Frank, S., Marrec, J. (2023, June 13). NREL OpenStudio® Server. https://github.com/NREL/OpenStudio-server

Maile, T., Mosiman, C., Maile, T., Gupta, M., Long, N., Shapinsky, T., Marzullo, T., Macumber, D., Fleming, K., Goldwasser, D., Moore, N., Coleman, T. (2023). BuildingSync-gem. https://github.com/BuildingSync/BuildingSync-gem

NYC Office of Technology & Innovation. (June 27, 2023). Spotlight: The new era of tech in NYC. https://www.nyc.gov/content/oti/pages/

Penn State. (2021, January 28). GEOG 497: 3D Modeling and Virtual Reality. Retrieved from Penn State College of Earth and Mineral Sciences 28 June 2023: https://www.e-education.psu.edu/geogvr/node/887

Appendix

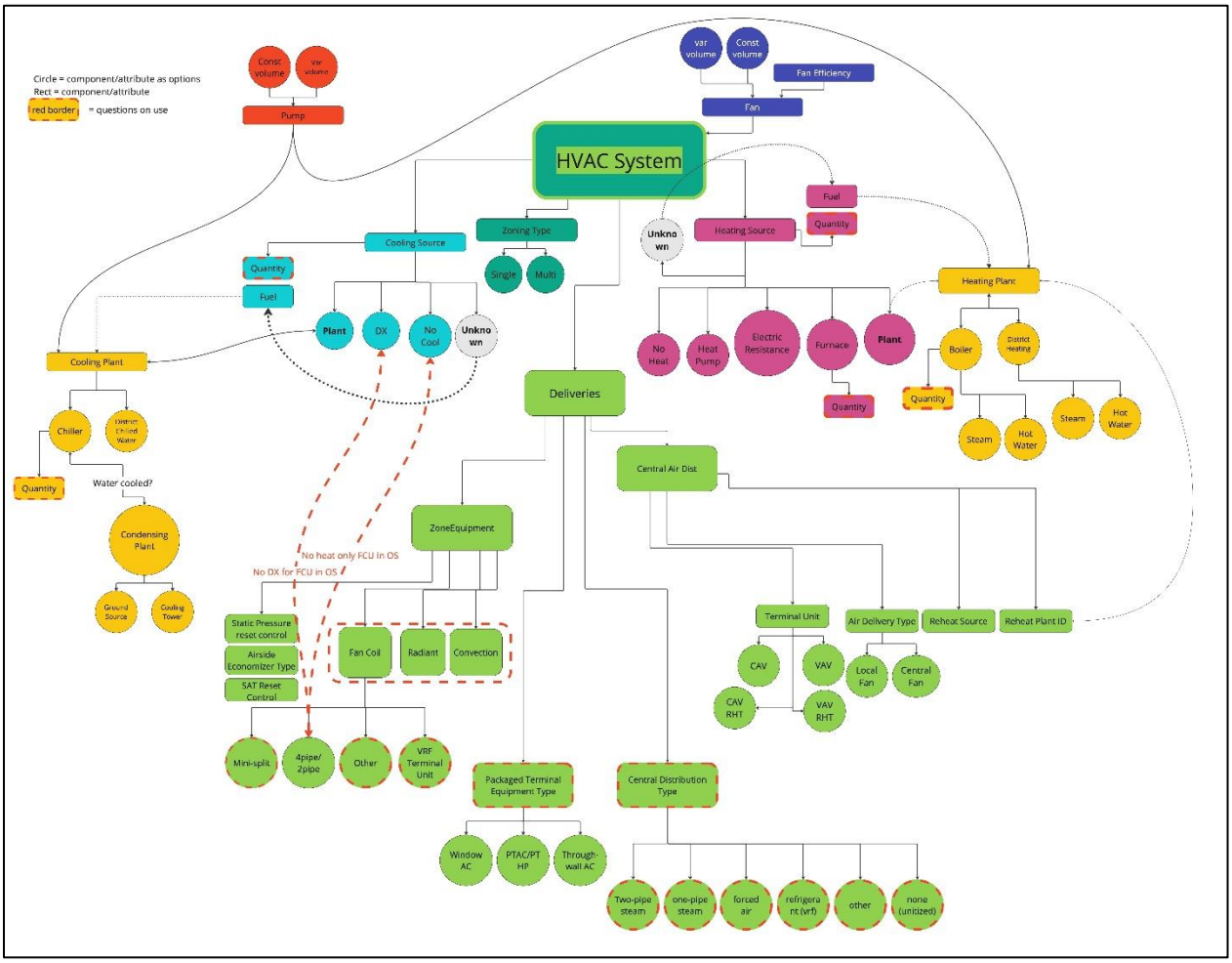

Figure 13. HVAC systems breakdown in Audit Template BSXML files