UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

A SOFTWARE TOOL FOR THE ENERGY SIMULATION OF A SINGLE-FAMILY RESIDENCY WITH OPENSTUDIO AND ENERGYPLUS

Diploma Thesis

Tziamtzis Konstantinos

Supervisor: Fevgas Athanasios

October 2023

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

A SOFTWARE TOOL FOR THE ENERGY SIMULATION OF A SINGLE-FAMILY RESIDENCY WITH OPENSTUDIO AND ENERGYPLUS

Diploma Thesis

Tziamtzis Konstantinos

Supervisor: Fevgas Athanasios

October 2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΝΑ ΕΡΓΑΛΕΙΟ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΤΗΝ ΕΝΕΡΓΕΙΑΚΗ ΠΡΟΣΟΜΟΙΩΣΗ ΚΑΤΟΙΚΙΑΣ ΜΕ ΤΗΝ ΧΡΗΣΗ ΤΩΝ ΛΟΓΙΣΜΙΚΩΝ OPENSTUDIO ΚΑΙ ENERGYPLUS

Διπλωματική Εργασία

Τζιαμτζής Κωνσταντίνος

Επιβλέπων: Φεύγας Αθανάσιος

Οκτώβριος 2023

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων	**Φεύγας Αθανάσιος** Μέλος ΕΔΙΠ, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Μπαργιώτας Δημήτριος** Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Τσομπανοπούλου Παναγιώτα** Αναπληρώτρια Καθηγήτρια, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

The Declarant

Konstantinos Tziamtzis

Ευχαριστίες

Με την ολοκλήρωση της διπλωματικής μου εργασίας, θα ήθελα να ευχαριστήσω την οικογένειά μου για την υποστήριξή που μου προσέφεραν από την αρχή της ακαδημαϊκής πορείας μου μέχρι σήμερα. Θέλω επίσης να πω ένα μεγάλο ευχαριστώ στον επιβλέπων καθηγητή μου, τον Αθανάσιο Φεύγα που με υπομονή και άμεση ανταπόκριση σε κάθε απορία και ερώτημά μου, με καθοδήγησε στην ολοκλήρωση της διπλωματικής εργασίας μου.

Diploma Thesis

A SOFTWARE TOOL FOR THE ENERGY SIMULATION OF A SINGLE-FAMILY RESIDENCY WITH OPENSTUDIO AND ENERGYPLUS

Tziamtzis Konstantinos

Abstract

The purpose of this thesis is the detailed description of the development of a web application, which can provide users with a simple, easy to use energy simulation tool. The tools that are used for the development of the application are OpenStudio, EnergyPlus and the server simulation tool XAMPP. The application is oriented for people with no prior knowledge on energy simulation. The first chapter is an overview of the energy crisis, the energy inefficiency problems the COVID-19 pandemic and the Russo-Ukrainian war caused, and the importance of energy simulation. We also present related works and discuss how our work is going to benefit the users. The second chapter focuses on the introduction of OpenStudio and EnergyPlus and the steps that were followed to build the two building models of the application. The third chapter analyzes the energy conservation “Measures” written in the language Ruby and the features that provide the users for manipulating the simulation of the models. The fourth chapter showcases the structure of the web page the app runs on and how the user inputs are given to the simulation. The final chapter gives a recap of the chapters, explains the differences in behavior of the models during the simulations and proposes ways that the tool can be improved.

Keywords:

OpenStudio, EnergyPlus, energy simulation, energy models, Ruby, web development

Διπλωματική Εργασία

ΕΝΑ ΕΡΓΑΛΕΙΟ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΤΗΝ ΕΝΕΡΓΕΙΑΚΗ ΠΡΟΣΟΜΟΙΩΣΗ ΚΑΤΟΙΚΙΑΣ ΜΕ ΤΗΝ ΧΡΗΣΗ ΤΩΝ ΛΟΓΙΣΜΙΚΩΝ OPENSTUDIO ΚΑΙ ENERGYPLUS

Τζιαμτζής Κωνσταντίνος

Περίληψη

Σκοπός της παρούσας διατριβής είναι η λεπτομερής περιγραφή της ανάπτυξης μιας διαδικτυακής εφαρμογής, η οποία μπορεί να παρέχει στους χρήστες ένα απλό και εύχρηστο εργαλείο ενεργειακής προσομοίωσης. Τα εργαλεία που θα χρησιμοποιηθούν για την υλοποίησή της είναι το OpenStudio, το EnergyPlus και το πακέτο λογισμικού, XAMPP. Η εφαρμογή απευθύνεται σε χρήστες χωρίς προηγούμενες γνώσεις σχετικά με την ενεργειακή προσομοίωση. Στο πρώτο κεφάλαιο γίνεται μια επισκόπηση της ενεργειακής κρίσης, των προβλημάτων ενεργειακής αναποτελεσματικότητας που προκάλεσαν η πανδημία COVID-19 και ο Ρώσο-Ουκρανικός πόλεμος και η σημασία της ενεργειακής προσομοίωσης. Παρουσιάζουμε επίσης σχετικές εργασίες που ενέπνευσαν το θέμα και τον τρόπο με τον οποίο η εργασία μας πρόκειται να ωφελήσει τους χρήστες. Το δεύτερο κεφάλαιο επικεντρώνεται στην γνωριμία με το OpenStudio και το EnergyPlus και στη επεξήγηση των βημάτων που ακολουθήθηκαν για την κατασκευή των δύο κτιριακών μοντέλων της εφαρμογής. Στο τρίτο κεφάλαιο αναλύονται οι συναρτήσεις που αναπτύχθηκαν σε γλώσσα Ruby και οι δυνατότητες που παρέχουν στους χρήστες για τον χειρισμό της προσομοίωσης των μοντέλων. Στο τέταρτο κεφάλαιο παρουσιάζεται η δομή της διεπαφής χρήστη και ο τρόπος με τον οποίο εισάγονται οι είσοδοι της προσομοίωσης. Στο τελευταίο κεφάλαιο γίνεται ανασκόπηση και παρουσιάζεται μια επεξήγηση των διαφορών στη συμπεριφορά των μοντέλων κατά τη διάρκεια των προσομοιώσεων και προτείνονται τρόποι με τους οποίους μπορεί να βελτιωθεί το εργαλείο.

Λέξεις-κλειδιά:

OpenStudio, EnergyPlus, ενεργειακή προσομοίωση, ενεργειακά μοντέλα, Ruby, web development

Table of Contents

List of figures

List of tables

Acronyms

ANN: Artificial Neural Networks

API: Application Programming Interface

APP: Application

ASCII: American Standard Code for Information Interchange

BCL: Building Component Library

BES: Building Energy Simulation

BLAST: Building Loads Analysis and System Thermodynamics

CBES: Commercial Building Energy Saver

CityBES: City Building Energy Simulation

CSS: Cascading Style Sheets

DOD: Department of Defense

DOE: Department of Energy

Dymola: Dynamic Modeling Laboratory

ECM: energy conservation measures

EE: Energy Efficiency

ETS: Emissions Trading System

EU: European Union

EUI: Energy Use Intensity

GUI: Graphical User Interface

HTML: HyperText Markup Language

HVAC: Heating Ventilation Air Conditioning

IAQ: indoor air quality

ICE: Indoor Climate Energy

IDF: Input Data Files

JSON: JavaScript Object Notation

LNG: Liquefied Natural Gas

NBLSD: National Bureau of Standards Load Determination

NREL: National Renewable Energy Laboratory

OA: Outdoor Air

PA: Paris Agreement

RF: Random Forest

SDK: Studio Development Kit

SVR: Support Vector Regression

TRYNS: Transient System Simulation

UBEM: Urban building energy models

UN: United Nations

US: United States

USD: United States Dollar

UTF: Unicode Transformation Format

VRF: Variable Refrigerant Flow

WWR: Window to Wall Ratio

XAMPP: X-operating system Apache Mysql Php Perl

XGBoost: Extreme Gradient Boost

3D: Three Dimensional

CHAPTER 1 INTRODUCTION

The energy market has always been characterized by its cyclical nature. As far back as we can see in history, each market expansion and evolution in the industry has usually been followed by a crisis, economic or political, or a disruption to the flow of everyday life. The last decade, researchers have put a bigger emphasis towards the transition to green energy, due to the increasing demand of energy and the worldwide directive to reduce greenhouse emissions. However, two major crises have impacted the world and in quick succession of one another, with lingering effects that are still being dealt with to this day. Those two crises are the global pandemic of COVID-19 and the war between Russia and Ukraine. The pandemic forced almost everything into a lockdown, everyone stayed at their houses all day, and this resulted in an increase of energy demand and consumption. Unfortunately, 2020 and 2021 had much colder winters and much hotter summers than previous years. Pairing that with the lockdown measures, each government was put on extreme pressure to provide sufficient energy to combat the weather conditions and everyday needs of the citizens, thus resulting in moving away from the green energy plans. In 2022, as the world was recovering and was slowly getting back to its feet, the war between Russia and Ukraine started in February. Because of the sanctions imposed by the EU to Russia, the latter as a response decided to cut the supply of natural gas to Europe. The result of this action was the increase of oil and natural gas prices. Another matter that rose to the attention of almost everyone is that enough buildings and houses weren't built to be as energy efficient as they should be. It is very clear that the combination of all the problems has put the people under extreme economic and social strain. Let's take a more detailed look at the points mentioned.

In 2015, the UN signed the Paris Agreement, a list of goals and measures to completely reduce CO2 emissions by 2030 and with other likewise agreements being signed by nations worldwide. These served to promote economic growth and environmental protection [1][2]. When the COVID-19 pandemic started, the economy took a huge blow with the mismatch in the energy demand and supply created in the early months of 2020 [3]. The winter season and the elevated prices on energy have pushed to the side the

policies on reducing greenhouse emissions and turned the focus to the post-Covid recovery goals and the looming energy poverty [2][3][4]. Among the fossil fuel types, gas was the most controversial.

The gas market is very dependent on the weather and with the 2020 and 2021 winters being colder than previous years, the market has been dealt a serious blow. Asia and Europe were and still are in serious demand for LNG, but the US was also challenged with the severe weather and had to reduce the departing shipments. Furthermore, the extreme summer heat caused a reduction in hydropower generation because of the drought and few transit problems between the EU and the US worsened the situation [3][4][5]. Also, the war has worsened the situation by increasing even further the cost of transportation [5]. The northern parts of the EU rely on wind energy, but they have suffered a huge loss due to poor wind conditions, this amplified the demand for gas and coal [4]. The EU's plans for emissions reduction, imposed by the ETS (Emissions Trading System), led to an obvious increase in oil and coal prices, recording a price peak of 88€/ton on the 8th of December in 2021. Despite the increase in carbon prices, companies have turned to coal because of the scarcity of natural gas and the energy stability it provides [2][4]. The high energy prices serve to amplify the inflation in the energy department with a rise of 17.4% in summer of 2021, marking a peak in the EU's economy [4]. The private sector plays a big role in energy efficiency and stability and nowadays businesses are more aware of the energy they consume and the effects on the environment, although many of them can't balance their finances [6]. Energy efficiency is the reduction of energy consumption, using only the amount needed for each service, limiting the waste produced and using green energy sources to reduce cost and emissions [6][7]. This policy obviously leads to economic growth and stability [5][6][7]. New infrastructures should be designed with the target of energy efficiency in mind because, aside from the benefit of reducing cost, new job positions are created directly and indirectly. Some good examples in this context are Indonesia and China [5][6]. Besides the steps needed to realize an energy transition plan, to achieve energy efficiency we need to consider the design of the building, the materials to use, the incorporation of energy conversion technologies, distribution, storage and waste heat recovery methods. These considerations can accelerate the process of attaining and maintaining energy efficiency. The utilization of renewable energy sources, alongside

the development of nanotechnology can help to achieve energy efficiency through efficient applications on renewable energy sources [2][5].

When the Russo-Ukrainian war started in late 2022, the world suffered a second energy crisis due to the halt of natural gas and oil supply from Russia, since she was the biggest supplier of natural gas and third biggest in oil worldwide [8]. The supply halt of natural gas supply from Russia, drove the governments worldwide to find new production and supply chains of fossil gas, one of the ways being LNG's [8]. The objective was to fill the vacant seat of supplier created by Russia. This rush will probably put the world again in a high carbon period and put more distance to the 1.5°C temperature of the PA [8]. Only a few countries have managed to move away from carbon-based energy and push for electrification in the industry, but the plans for this initiative are still underdeveloped [8]. The war also contributed to the increase of oil prices to 100$/per barrel. This increase in prices had a combined monetary return of 100 billion USD to the 28 largest oil companies [5][8]. Additionally, the price increase of carbon fuels has caused a price increase to other goods and services such as food [5]. Yet despite the price increases, the world still depends on and uses fossil fuels.

Even before the two crises, countries mostly used fossil fuels since the economies were highly dependent on them and this has become more obvious now [9]. However, we have witnessed a source shortage and with experts noting that in about 100 years the sources will be cut in half [9]. Even though fossil fuels cover about 80% of global energy and over 60% of electric generation, unfortunately they contribute greatly to greenhouse gas emissions and are the main cause for carbon dioxide emission [9].

As a logical consequence of the increase in prices, inflation and the lay-offs, poverty percentages skyrocketed, and a new form of poverty was created. This new form is fuel poverty, with the definition as the inability to afford the most necessary amount of energy for home appliances [7][10]. It is a difficult concept of poverty to define precisely. A good rule of thumb that has been established is that a household can have a low income but can secure the energy needed and vice versa a household with high income can't afford the costs or manage them in expense of other necessities [7].

For a change to happen, people need to change their stance and behavior on how they consume energy. People need to conserve energy and use it as efficiently as possible and make changes to their houses to not lose energy [5]. As research has shown, household

energy inefficiency can be countered by addressing the technological backwardness of infrastructure. Even if these objectives are addressed, if the populace does not change their understanding of economical use of energy sources, things won't move forward [11].

Aside from the problems that were caused by Covid and the war, the last decade, the frequency of extreme weather events has seen a rise as well as their severity [12]. For example, in Germany the precipitations have caused flooding at some cities, in Italy and France there was severe cases of drought during summer. Of course, the usual climate problems, like the melting of ice and the rise of the sea level, still exist and are amplified.

In a way, buildings serve as a sort of interface with which humans interact with the climate. More precisely, the way we use energy can have dire consequences, for example changes in the required energy to cover HVAC equipment needs due to extreme weather events. Because of the increased energy consumption, there is an increase of greenhouse gas emissions [12]. Also, there are problems that can be caused to the building that can affect the occupants.

Returning to the point of energy efficiency, the energy conservation policies that were implemented in the last decade had a very interesting effect. The industrial sector had lowered the final electricity consumption from 53.4% to 42,0%, while the residential sector had an increase from 23.1% to 26.9%. In a similar way, final gas consumption decreased from 54.7% to 37.0% and increased from 22.7% to 29.9% for the industrial and residential sectors respectively [13]. As mentioned before, the lack of basic understanding on efficient household energy consumption leads away from financial stability. This becomes very clear if we consider that in a household, multiple energies are used but not all of them are necessary for activities [13]. One good way for households to reduce energy consumption is to replace older electric appliances such as air-cons and refrigerators, instead of trying to expand the life of the device which can result in bigger energy use [10]. This measure is supported and encouraged by many governments [13]. Household electricity usage has fluctuations throughout the day, it increases when the members are in the house and are active and decreases when the members are absent or at bedtime. With this observation in mind, electric power companies are trying to equalize electricity usage by setting nighttime electricity charges [13].

Governments, in their attempts to reach the desired goal of efficiency, are employing energy conservation and green energy policies, these affect all households and

some more severely than others [13]. Usually, small households use more energy per capita, for example, single-parent households have a bigger energy consumption burden. Also, the energy burden increases with age [13]. Energy consumption is also affected by regional position. More specifically, residents at high latitudes have long and cool daylight hours during summer but require more energy during the winter. On the opposite end of the spectrum, residents in lower latitudes enjoy not so cold winters but need more energy during the summer [13].

For households to have a better financial stability over energy consumption, they must optimize the thermal performance of the building, especially in winter. With the rise of the number of households by 46% since 1970, households should get acquainted with energy-efficient technologies [14]. These solutions work on existing buildings and provide a significant financial relief in energy bills and are environmentally friendly [14]. Research has shown that to improve the occupants' thermal comfort, we can implement energy efficiency measures without using more energy [14]. However, the measures vary from building to building, since each one has a different building code, envelope etc. [14]. Usually, with a simple survey we can identify the areas to be improved.

Energy and building retrofits on older buildings offer a good solution for improving energy efficiency. 90% of energy consumption in a building comes from lighting, plug loads and HVAC equipment, with the latter consisting of 55% of the consumption [15]. Despite the pros of energy retrofits, small and medium business owners don't have the money, personnel of larger companies nor the tools to identify the cost-effective and energy efficient retrofits. So, there is a strong need for tools that can be purchased at very low prices or can be offered free of charge [15].

A way to study retrofit scenarios is through Building Energy Simulation (BES) applications. These applications offer a wide array of features such as a detailed view of the building design, and a variety of energy measures that can determine the equipment for financial optimization. Other applications include a simulation of the building performance under the weather conditions of a specified location, as well as the level of comfort and air quality inside the building amongst others [12][16]. When using these applications, the modeler can specify certain uncertainties such as the envelope of the building and other technical parameters. There are also parameters like the weather datasets of the location of the building, that if they are outdated or from a different location

can lead to inaccuracies in the simulation. Other parameters that can insert inaccuracies in the simulations are the presence, the behavior and the level of activity of an occupant. These parameters affect the operation and maintenance of the building equipment, the heat and cooling management alongside the equipment responsible for it [16][17]. To acquire the best result, the building model must resemble as close as possible the actual building [16][17].

There are three categories of simulations, the “white box”, “black box” and “grey box”. The “white box” is the physical energy model approach, it uses physical parameters and thermal equations to conduct simulations. This approach is commonly used by tools such as EnergyPlus, Dymola and TRNSYS. The “black box” approach is based on historical operational big data and machine learning algorithms which refer to artificial neural networks (ANN), extreme gradient boosting (XGBoost), random forest (RF) and support vector regression (SVR) among other techniques [10]. Lastly, the “grey box” approach is the combination of the previous approaches.

Taking a closer look at the first and most used approach, this approach uses the physical energy models of buildings, these are based on heat and mass balance equations. These equations simulate the dynamic thermal behavior with the following heat transfer models: radiation, convection and conduction. These three models are considered the interaction between the building envelope and the surroundings. EnergyPlus, Dymola, TRNSYS, DOE-2 and Matlab are opensource software products that use this kind of method. This method requires a good understanding of the building modeling [10][15]. To create an accurate building model, it is necessary to define the materials of the building’s envelope, the HVAC systems and interior thermal variables. Alongside these variables, the occupancy, thermostat, activity level and equipment function time schedules, the thermal zones and weather file of the desired location must be defined to have a complete model [10][15].

EnergyPlus was developed in 1997 and was released in 2001 and is considered the most widespread energy simulation tool. The specialization of this program is that it can solve the heat function of a large time scale of a building’s thermal performance within a short computation time [10]. It is important to remember that since EnergyPlus was first created for building envelope modeling, installing HVAC systems might be challenging and cause problems during simulations [10][15].

Transient system simulation (TRNSYS) is a tool that is defined as flexible in certain parts and has numerous applications for energy simulation, such as HVAC systems, renewable energy systems, solar systems and the building itself [10]. The TRNSYS program includes a graphical user interface and is incredibly flexible, allowing users to create unique components.

The dynamic modeling laboratory (Dymola) is built on the Modelica language, which is an equation-based, acausal, and object-oriented language for easier modeling of physical systems [10]. Modelica eliminates the need for users to manually transform equations into block diagrams or assignment statements and provides hierarchical model composition, fully reusable libraries, connectors, and causal linkages [10]. A robust visual editor is included in Dymola for creating and executing models [10]. It also offers straightforward interaction with outside data [10]. It may not be appropriate for a huge block of buildings, unlike EnergyPlus, but it can mimic a single building with a reasonable computing cost [10]. Some other notable simulation tools are eQUEST, DOE-2 and IDA ICE.

The fundamental benefit of the “white box” is that the connection between input and output can be observed and explained more accurately [10]. Similarly, the drawback is that entering all the specific building parameters requires a lot of time and work, which might be problematic for many buildings still in the planning stage as well as some existing structures [10].

Even though the modeling process is laborious and repetitious, physical models are preferable because of their dependability, interpretability, and correctness [10]. On the other hand, when users have enough data from existing structures but do not have or have insufficient design building information, data-driven models are preferable [10]. When the necessary data is insufficient for the other two models, hybrid models may be a better choice since they offer a compromise between physical and data-driven models [10]. Simulation tools such as TRNSYS and Dymola are preferable for creating and simulating energy systems, while EnergyPlus is preferable for building envelop simulation [10].

If we try to use pre-simulated approaches, we are going to run to some obstacles such as using pre-established designs that don’t match the geometry or the features of the building that we are modeling [15].

1.1 Subject of the thesis

The goal of the thesis is to describe the construction of a web application that exploits OpenStudio and EnergyPlus, which are two widespread apps for energy simulation. This application was created with the intent to be used by people that aren't well versed in energy simulation products but are conscious of the energy problems that plague the world, and they want to contribute to the solutions. For example, someone who wants to build or already owns a house for his family should be able to check if the house is energy efficient for economic and environmental reasons. Based on these considerations, the solution that is described is the development of a web-based application for energy simulation of a dwelling for a family of four. The models are pre-built, and the user is given the option to edit some parameters of the models, such as the dimensions of the building, and the number and location of windows. The user interaction is done through a web application, and they can view and download the results of the simulations. Also, the user won't need prior knowledge on energy modeling to use the application. This approach is also intended to make the application easier and faster to use. It focuses on the results of the simulations and provides the users with the ability to perform experiments without dealing with the specifics.

1.1.1 Contribution

1. Construction of a detailed model of a house for a family of four.
2. Construction of a simple model of the house of a family of four.
3. Ruby language code that allows changing the window to wall ratio, placing windows on any side of the building, determining the window dimensions and the sill height.
4. Ruby language code that allows to change the R-value of walls and ceiling.
5. Creation of a web application for inserting the simulation parameters via a form and making the simulation results available.

1.2 Organization of the volume

The structure of the chapters that follow is as follows:

Chapter 2 gives an overview of the OpenStudio and EnergyPlus programs, their functions and capabilities, through a step-by-step description of the construction of the two models that the tool uses.

In Chapter 3 we analyze the code of the OpenStudio Measures that control the building's envelope. Also, this chapter serves as a user manual for the parameters that the user can change in the simulation.

Chapter 4 presents the implementation of the website and the user interface. In addition, it is explained how we can import the simulation data without having to use the OpenStudio interface and how to run a simulation via the command line.

Finally, in Chapter 5 we recap what we have done in this thesis and propose some ideas that further improve and develop the tool.

1.3 Related works

The reader is encouraged to read related work that served as inspiration for the subject of the thesis. In "Commercial Building Energy Saver: An Energy Retrofit Analysis Toolkit", Tianzhen Hong et al. created the Commercial Building Energy Saver (CBES), an energy retrofit analysis toolkit, it includes the CBES Application Programming Interface (API) for integrating CBES with other energy software tools and applications, a web app (APP) for end users and other features [15]. The toolkit has a wealth of features, such as (1) Energy Benchmarking that generates an energy analysis evaluation, (2) Load Shape Analysis that pinpoints potential improvements that can realized for a building's operations, (3) Preliminary Retrofit Analysis that makes use of a specially created pre-simulated database, and (4) Detailed Retrofit Analysis that makes use of real-time EnergyPlus simulations [15]. The CBES has 100 reconfigurable energy conservation measures (ECMs) for evaluating 7 distinct building models in 16 temperature zones in California and 6 in other locations [15]. These ECMs include information on indoor air quality (IAQ), technical performance, and cost [15]. In "A Pattern-based Automated Approach to Building Energy Model Calibration" by the same authors, they present an innovative, automated model calibration method that addresses some of the drawbacks of conventional calibration procedures by logically coupling bias pattern identification with parameter adjustment [16]. Also, by Tianzhen Hong, "Automatic Generation and Simulation of Urban Building Energy Models Based on

City Datasets for City-Scale Building Retrofit Analysis" demonstrates the retrofit analysis function of City Building Energy Saver (CityBES), which uses EnergyPlus to produce and simulate Urban building energy models (UBEM) based on user-selected ECMs and building datasets from cities [18]. CityBES is a new tool that can be utilized as a web application, that supports city-scale building energy efficiency plans, predictions and programs [18]. In "Rapid Application Development with OpenStudio", Evan Weaver et al. present several case studies on implementing different energy modeling applications for end users [17]. Tailoring the application for each user and how quickly the users can adapt to them. All apps were created based on the OpenStudio software development kit.

CHAPTER 2 BUILDING ENERGY MODELS

In this chapter we introduce the OpenStudio and EnergyPlus software, their history and features. We present the creation of the building models from the ground up and through the presentation we explain the different inputs to the OpenStudio application.

2.1 Introduction to EnergyPlus

The National Bureau of Standards Load Determination (NBLSD) software was developed in FORTRAN by the US National Bureau of Standards (then NBS, now NIST) in the 1970s to investigate rising energy use [19]. Then, the Building Loads Analysis and System Thermodynamics (BLAST) modeling program was developed by the US Department of Defense (DOD) and incorporated the NBSLD code [19]. At the same time, DOE-2 was developed, and the Postal Service's software was converted to FORTRAN by the US Department of Energy (DOE) [19]. In 1998, DOE-2's funding from the Department of Defense was halted along with BLAST [19]. The Department of Energy also released papers at the same time on a new energy modeling program that goes beyond BLAST and DOE-2 [19]. During that period, the EnergyPlus simulation engine was under a reconstruction process. EnergyPlus was originally written in FORTRAN code and in 2014 it was reprogrammed in C++ [19][20][21]. Engineers, architects, and academics increasingly utilize EnergyPlus as a simulation engine to plan energy flows in buildings. In addition to thermal heat pump systems, at the same time, it has the capacity to address conditions for several thermal zones [19][20][21]. Also, the program evaluates the interaction between thermal zones and the environment and allows the creation of schedules with that can have precision up to five minutes [19][20]. The heat and mass transfers that cause air to flow between thermal zones are taken into consideration by the model [19][20]. Fenestration models also compute the solar energy absorbed by windowpanes, as well as visual comfort and glare, and mimic controlled blinds and electrochromic glass [19][20]. EnergyPlus was initially intended to be only a simulation engine, where text files would be inputted, and results would be retrieved in the same format. The input file (.idf) describes multiple data, including: geometry, structures, thermal loads, thermal comfort systems and others

[19][20]. The Department of Energy encouraged the development of a Graphical user-interface (GUI) that could generate .idf files since producing such files manually was time-consuming and prone to errors [19][20]. The OpenStudio SketchUp Plug-In was created with this thought in mind.

2.2 Introduction to OpenStudio

The US Department of Energy (DOE) and the US National Renewable Energy Laboratory (NREL) created OpenStudio, a cross-platform package of open-source software tools, for the goal of energy modeling of buildings [19][20][21][22][23]. It offers a graphical interface to create building models as well as extensions for other design software like Sketch Up. It is the tool that gives the thermophysical properties and characteristics to the SketchUp-created building's geometry. It integrates them with the internal systems that make up the structure to simulate the energy requirements of the building while accounting for a wide range of parameters [20][21][22][23]. OpenStudio enhances EnergyPlus's functionalities, which also supports the Radiance daylight analysis engine, offers sample applications, and disperse new BEM standards [22][23]. The Application Programming Interface (API) provided by the OpenStudio SDK serves as the building block for the creation of OpenStudio Measures. Additionally, by shielding client applications from changes in EnergyPlus, the OpenStudio API enables them to develop separately [23]. With OpenStudio we can take advantage of the concepts of hierarchy and inheritance. Thanks to these two concepts we can input and assign data to the parts we need without needing to do it manually multiple times, thus improving the time and efficiency of creating a model [23]. Ruby is a scripting language that supports calling the OpenStudio API. By itself, the OpenStudio SDK can run Ruby programs, so it gives the SDK the possibility of making extensions and automations [23]. Measures are specially designed OpenStudio Ruby scripts that are used most frequently to apply an energy efficiency measure (EEM) to a building model in order to enhance simulated energy consumption [23]. Through the Building Component Library (BCL), an online repository for OpenStudio material that includes Measures, OpenStudio incorporates the concept of shared content and crowdsourcing [23]. OpenStudio also supports cloud computing [23].

2.3 Development of the two building models

OpenStudio offers a wide variety of pre-built, fully developed models of various buildings, even standalone rooms such as a small office model. However, we decided to build our models from the ground up. First, because we wanted specific properties for our models which the pre-established models don't provide. Second, because we wanted to experience how a complete model is developed. Lastly, we wanted to experiment and use as many features as possible. We present two models, the first one is a detailed house model with different rooms and thermal zones, while the other is simpler, consisting of only one room. The Detailed model serves as an accurate model of a house for a family of four. The second model is a more adjustable model, where the internal layout isn't so important. Through simulations we aim to determine how close the two models are, result-wise, and if it is necessary to have detailed layouts for accurate results.

OpenStudio Application structures its layout in tabs. To describe OpenStudio Application we are going to follow, as close as possible, the model formulation flow.

2.3.1 Site tab

Each time we open an OpenStudio model, we are greeted with the site tab as is shown in Figure 1. Aside from the tab, we have the File, Preferences, Components & Measures and Help options on the top bar. The File option is the common part to save or open a file. In the Preferences option we can change the language and the units used among other options (Figure 2). From this moment onward we are going to use the Metric (SI) units. In Components & Measures we can apply Measures and components that we have created or downloaded from the Building Components Library (BCL). Finally, in Help we can visit the OpenStudio manuals.

In the site tab we can select a weather file of a location, so when a simulation is run EnergyPlus can draw weather information from it. Also, we need to specify a Design Days file. This helps with HVAC sizing. Weather files can be downloaded from the official EnergyPlus website. For our two models, we are going to choose Athens as our location (Figure 3, Figure 4).

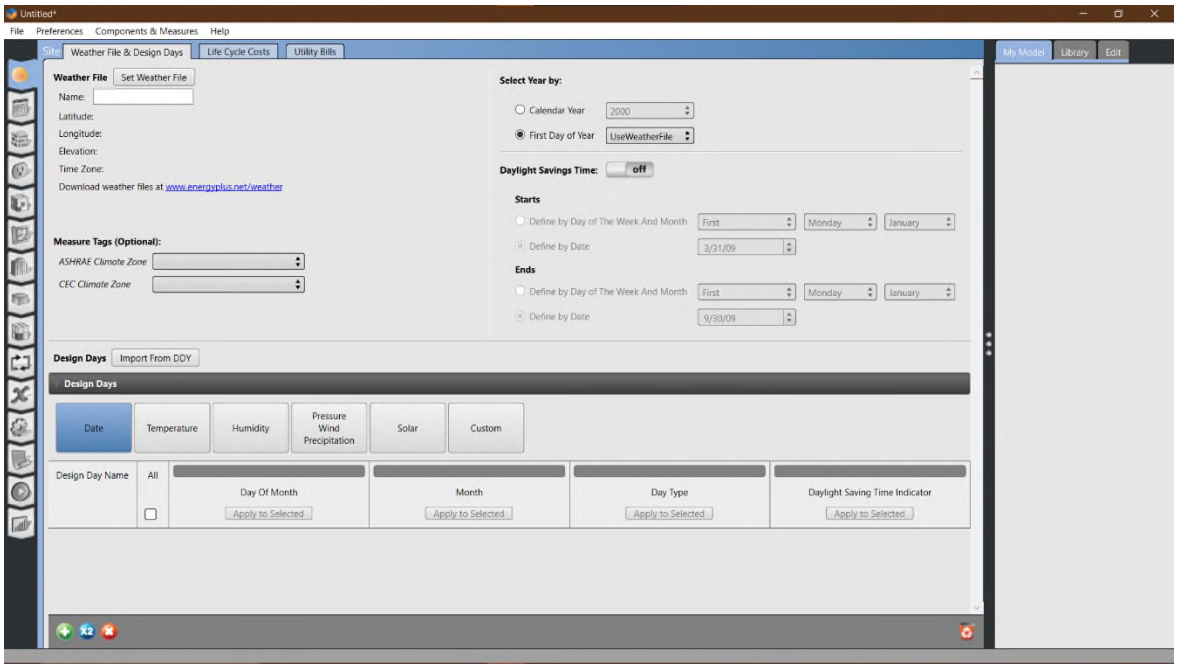

Figure 1: Main page when opening OpenStusio.

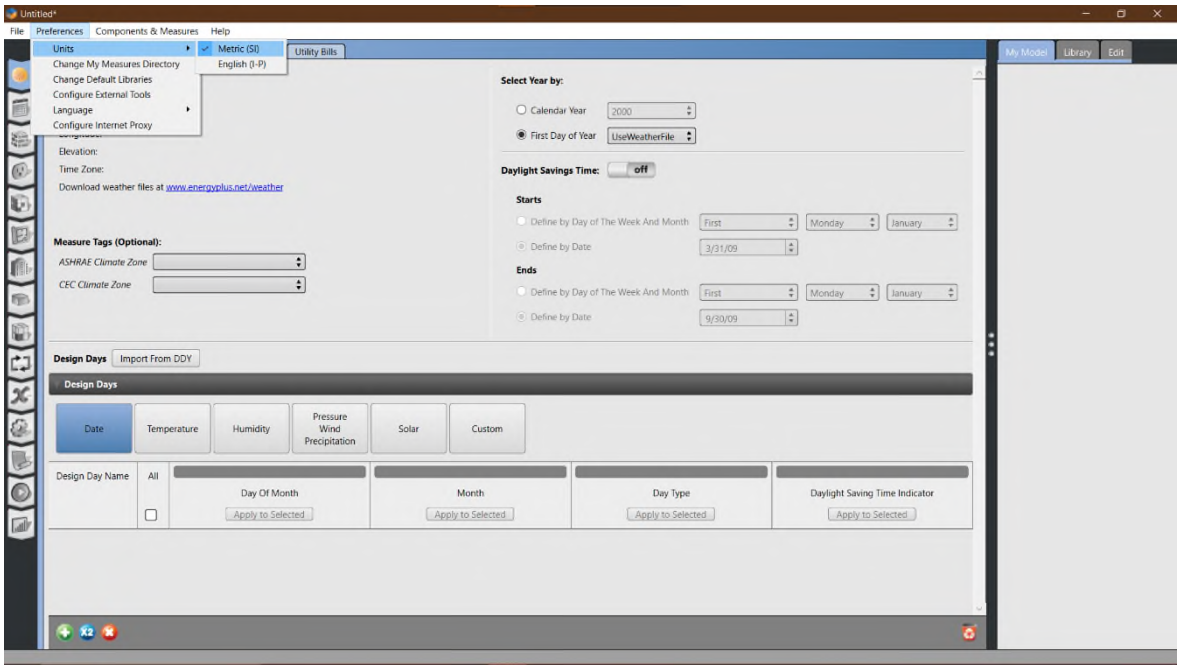

Figure 2: In Preferences option we change the units.

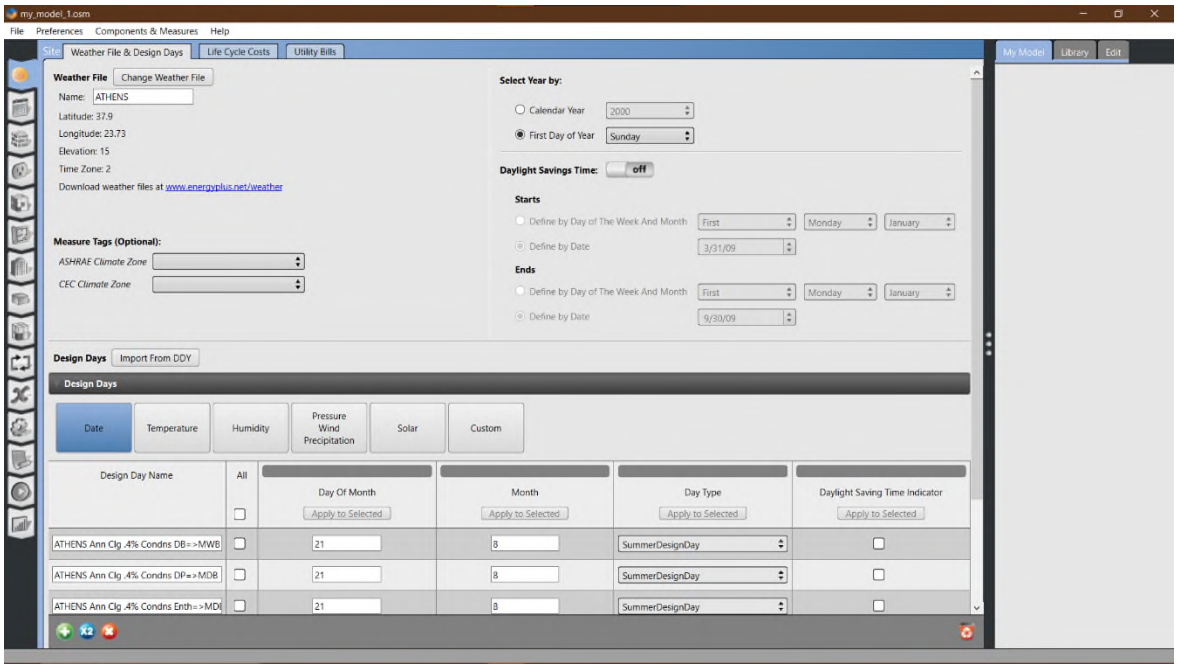

Figure 3: Weather files specified for the Detailed model.

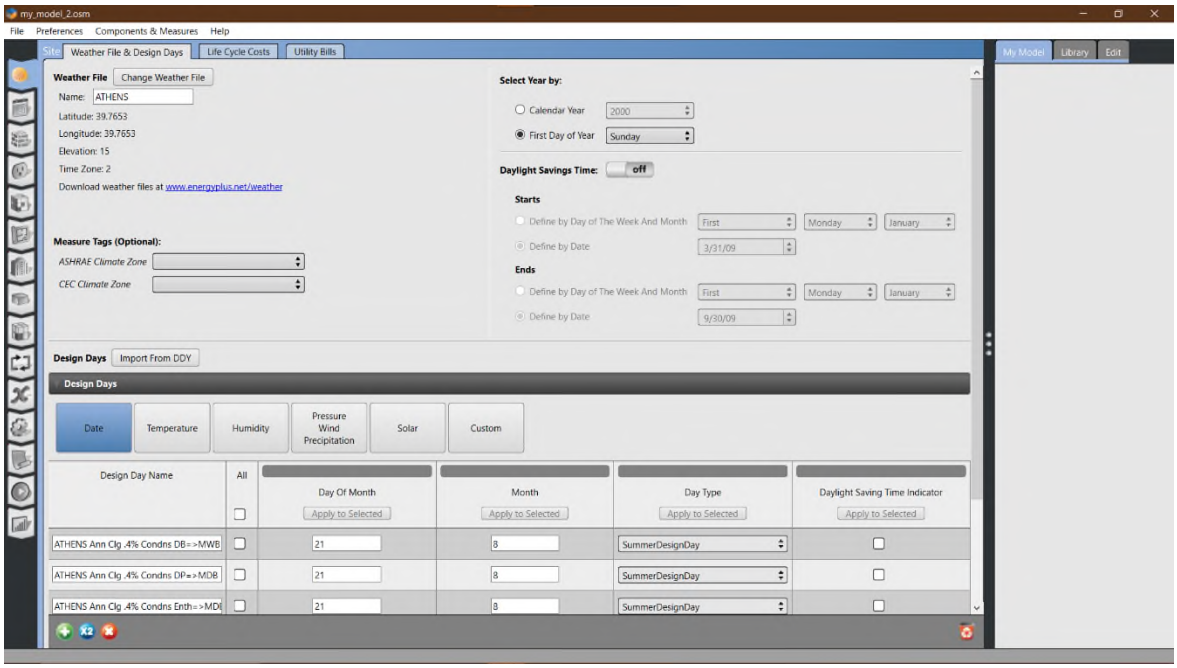

Figure 4: Weather files specified for the Simple model.

2.3.2 Constructions tab

In this tab, we specify the materials that are used for our Surfaces and Sub-Surfaces like walls and windows, and we create a Construction set with our Surfaces which are

utilized throughout the model. The Construction sets that we use are the same for both models.

The Constructions tab is divided into three subtabs, Materials, Construction and Constructions Sets subtabs. Starting with the Materials subtab, here we gather the materials for our Surfaces and Sub-Surfaces (Figure 5). The materials used are dragged and dropped from the libraries of OpenStudio and their properties are shown in Tables 1, 2 and 3.

Table 1: Construction materials.

Materials	Roughness	Thickness (m)	Conductivity (W/m*K)	Density (kg/m³)	Specific heat (J/kg*K)	Thermal Absorptance	Solar Absorptance	Visible Absorptance
1/2IN Gypsum	Smooth	0.0127	0.16	784.9	830	0.9	0.4	0.4
1IN Stucco	Smooth	0.0253	0.6918	1858	837	0.9	0.92	0.92
8IN Concrete HW	Medium Rough	0.2033	1.7296	2243	837	0.9	0.65	0.65
F08 Metal surface	Smooth	0.0008	45.28	7824	500	0.9	0.7	0.7
G05 25mm wood	Medium Smooth	0.0254	0.15	608	1630	0.9	0.5	0.5
I01 25mm insulation board	Medium Rough	0.0254	0.03	43	1210	0.9	0.6	0.6
MAT-CC05 4 HW CONCRETE	Rough	0.1016	1.311	2240	836.8	0.9	0.85	0.85
Metal Decking	Medium Smooth	0.0015	45.006	7680	418.4	0.9	0.6	0.6
Roof Insulation [21]	Medium Rough	0.2105	0.049	265	836.8	0.9	0.7	0.7
Roof Membrane	Very Rough	0.0095	0.16	1121.29	1460	0.9	0.7	0.7
Wall Insulation [40]	Medium Rough	0.0794	0.0432	91	837	0.9	0.5	0.5

Table 2: No mass construction materials.

No mass materials	Roughness	Thermal Resistance (m^2*K/W)	Thermal Absorptance	Solar Absorptance	Visible Absorptance
CP02 CARPET PAD	Smooth	0.1	0.9	0.8	0.8

Table 3: Glazing window construction materials.

Glazing Window Materials	Clear 3 mm
Optical Data Type	Spectral Average
Thickness (m)	0.003
Solar Transmittance At Normal Incidence	0.837
Front Side Solar Reflectance At Normal Incidence	0.075
Back Side Solar Reflectance At Normal Incidence	0
Visible Transmittance At Normal Incidence	0.898
Front Side Visible Reflectance At Normal Incidence	0.081
Back Side Visible Reflectance At Normal Incidence	0
Infrared Transmittance at Normal Incidence	0
Front Side Infrared Hemispherical Emissivity	0.84
Back Side Infrared Hemispherical Emissivity	0.84
Conductivity (W/m*K)	0.9
Dirt Correction Factor For Solar And Visible Transmittance	1
Solar Diffusing	Off

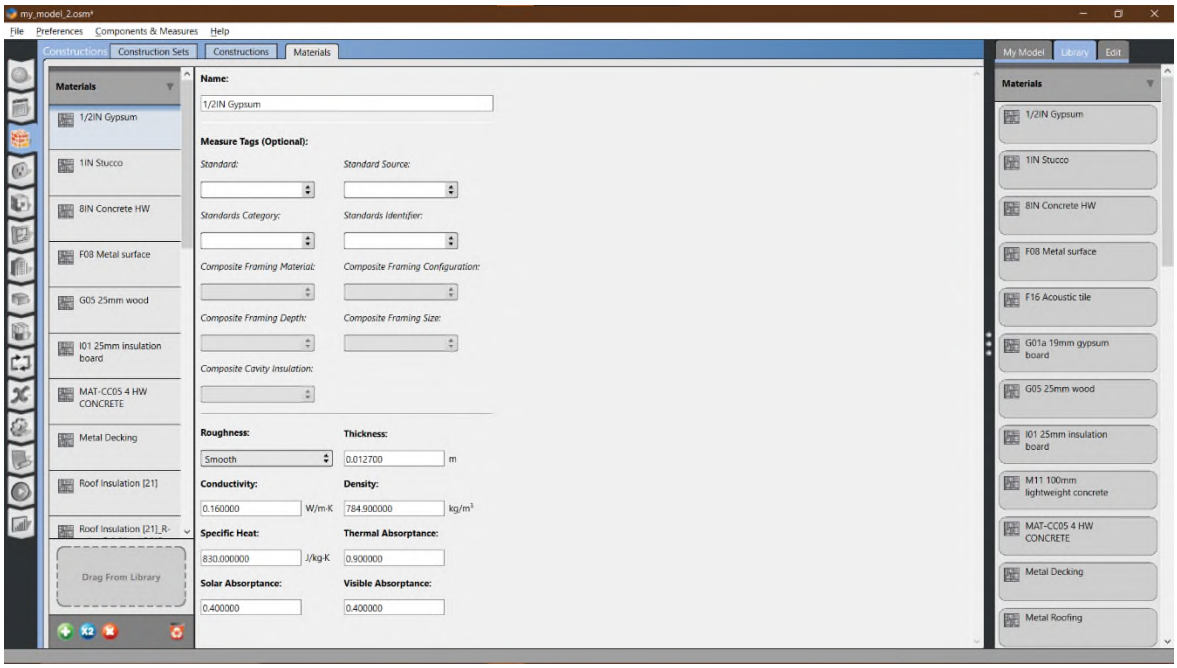

Figure 5: Materials subtab.

As soon as materials are imported to the model, we move on to the Constructions subtab. Here we build all Surfaces and Sub-Surfaces that are used in the models. Each Surface is coupled with a Construction. A Construction is made by layers of materials [23]. Surface materials are arranged from the outside in [23]. With this logic in mind, we drag and drop the materials we need for each surface. Table 4 shows the different Surfaces we created and the materials in each layer and Figure 6 shows how it looks like for our wall Construction in OpenStudio.

Table 4: Constructions and their materials in each layer.

	floor	inner door	outer door	roof	walls	window
Layer 1	MAT-CC05 4 HW CONCRETE	G05 25mm wood	F08 Metal surface	Roof Membrane	1IN Stucco	Clear 3 mm
Layer 2	CP02 CARPET PAD		I01 25mm insulation board	Roof Insulation [21]	8IN Concrete HW	
Layer 3				Metal Decking	Wall Insulation [40]	
Layer 4					1/2IN Gypsum	

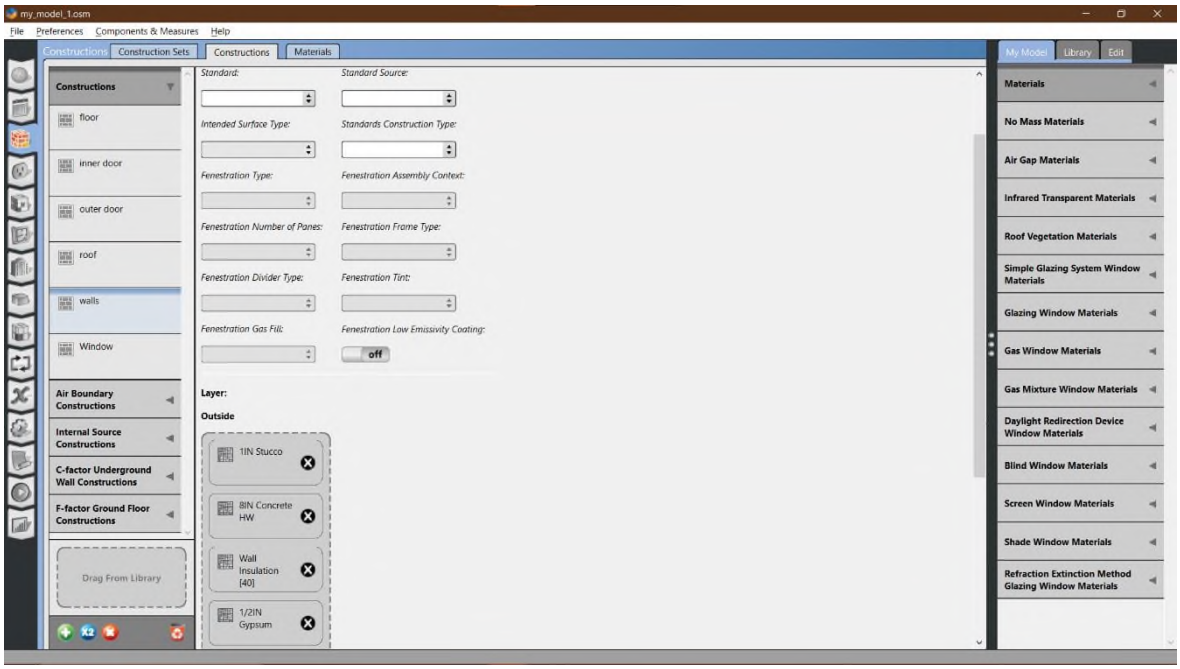

Figure 6: Constructions and how it looks like in OpenStudio, specifically for walls.

Again, a big emphasis that the lower the layer the closer it is to the interior of the building and vice versa. For example, in the walls Construction, Layer 1 is completely exposed to the outside of the building, while Layer 4 is inside the building.

The last subtab is the Constructions Sets. We create a set so when it is needed to specify a Construction for a surface. Then we input our set, and the Construction is assigned automatically to their corresponding fields. All we need to do is, again, to drag and drop the Constructions we created to the right field of the set as shown in Figure 7(a) and Figure 7(b).

This tab is a very early example of the concept of Hierarchy and Inheritance that OpenStudio incorporates. We can select a Construction Set that can be applied to a complete building with OpenStudio [23]. In each Construction Set we can specify a Construction for each Surface or Sub-Surface type (for example, walls, roofs, windows, and doors) [23]. When we assign a Construction set, OpenStudio first checks if we have assigned a Construction specifically for a Surface or Sub-Surface. If this is not the case, OpenStudio checks whether the Space the Surface or Sub-Surface belongs to, has been assigned a Construction set [23]. If not, OpenStudio checks if the Story the Space belongs to has been assigned a Construction set. Lastly, if the previous check does not yield a positive result, the Construction set of the structure is checked [23].

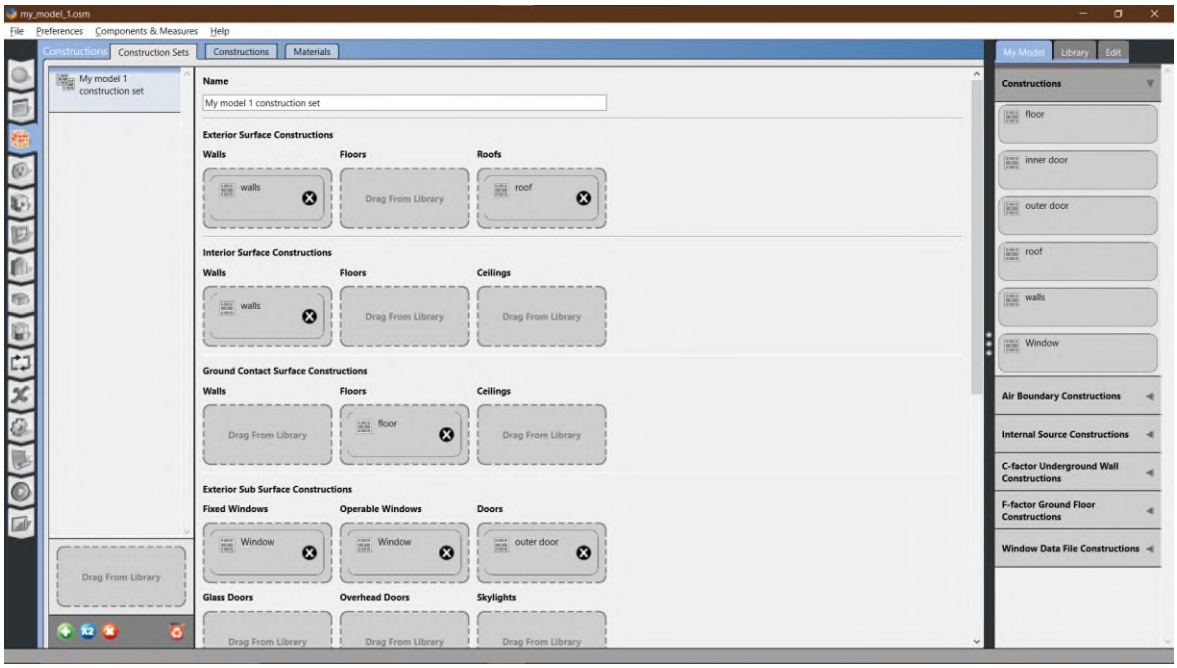

Figure 7(a): Complete Construction Set for the models.

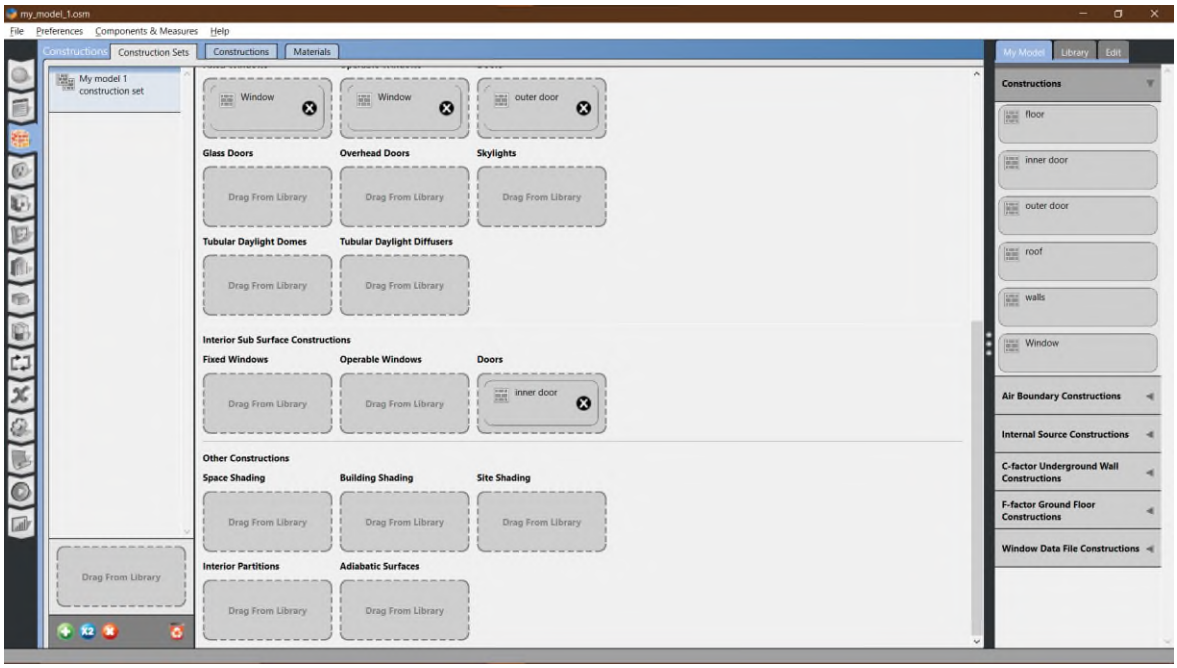

Figure 7(b): Complete Construction Set for the models.

2.3.3 Geometry tab

Now that we have our Construction set, we move to the geometry tab to establish the layouts of our models. In this tab, we can create the model of a building as well as to observe any changes to it when we apply a Measure that changes its structure, e.g., manipulating windows and geometry. Here, we also assign Space types and thermal zones.

A Space is a collection of Surfaces and Sub-Surfaces that enclose a volume of air. When we define a Space Type, we define the Construction that the Surfaces and Sub-Surfaces have, the different Schedule sets and the types of loads that exist in this Space. After the definition, we can apply this Space type to any Space that has the same purposes, in the model without needing to create the definition again [23]. A Space Type can be applied to one or more Spaces that share the same utility, whether they are in the same building or in different ones [23]. A Thermal Zone encompasses all Surfaces, Sub-Surfaces, loads and HVAC equipment of one or more Spaces that are served by the same thermostat [23].

Next, we present our Simple model since it is simpler and helps us ease into the features of this tab. The geometry tab is divided in two subtabs, 3D view and Editor. In the first one we see the building model and how it is oriented (Figure 8), in the second tab we create the floor plans for the model, assign zones and Spaces and place doors and windows.

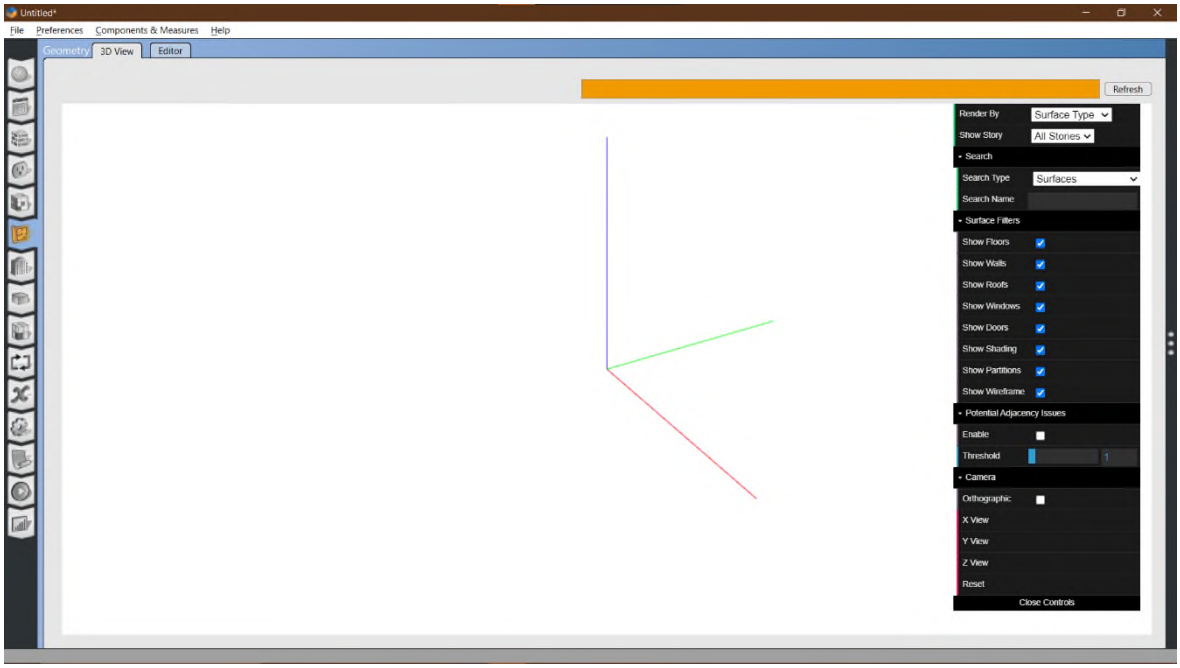

Figure 8: 3D view subtab.

The first time we use the editor subtab, we must choose the file format or the software module we would like to use to create the floorplan (Figure 9). Here, we can also use a plugin extension of Sketch Up CAD software to design our model. For the purpose of the thesis, we choose the FloorspaceJS option for both models. Now we are going to draw a 14x10 rectangle (Figure 10) which is the perimeter of the building. Then we assign the Space type and thermal zone to this rectangle (Figure 11) (Figure 12).

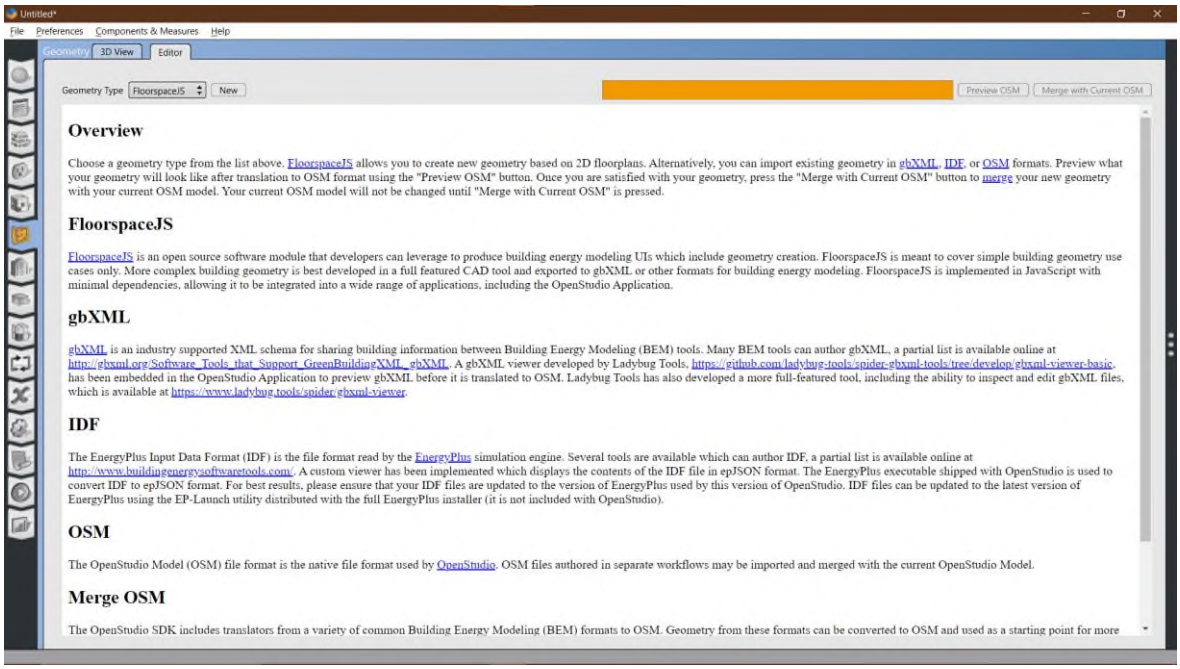

Figure 9: Editor subtab and geometry type options.

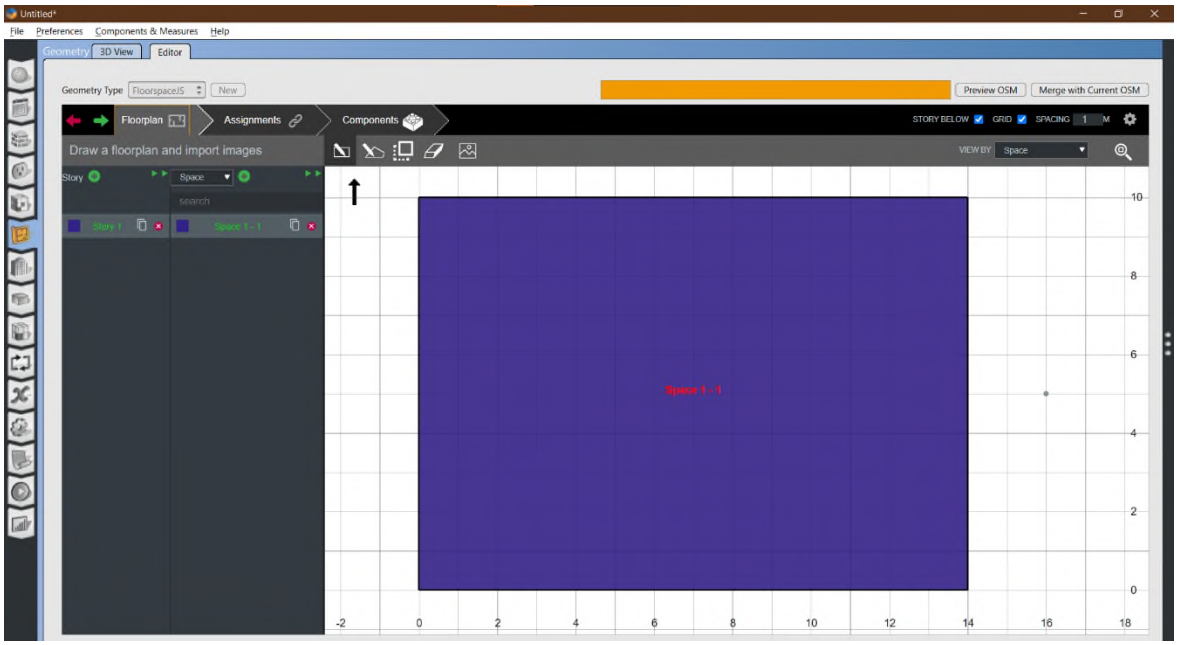

Figure 10: Floorplan of the Simple model.

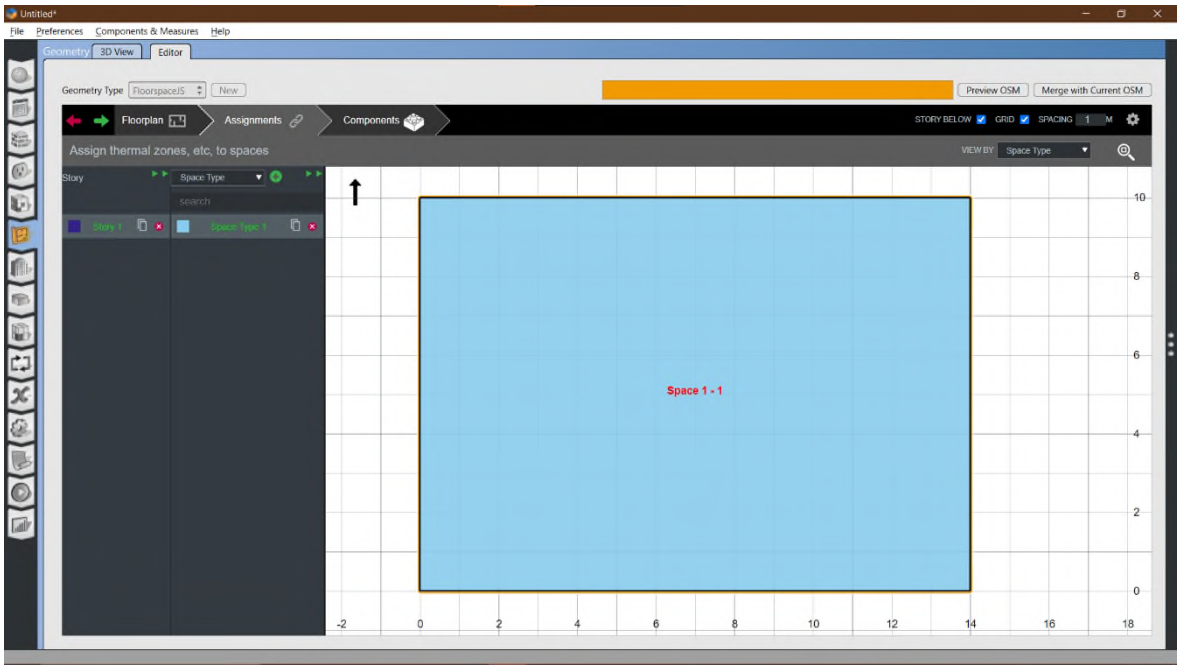

Figure 11: Assigning a Space type to the Space of the Simple model.

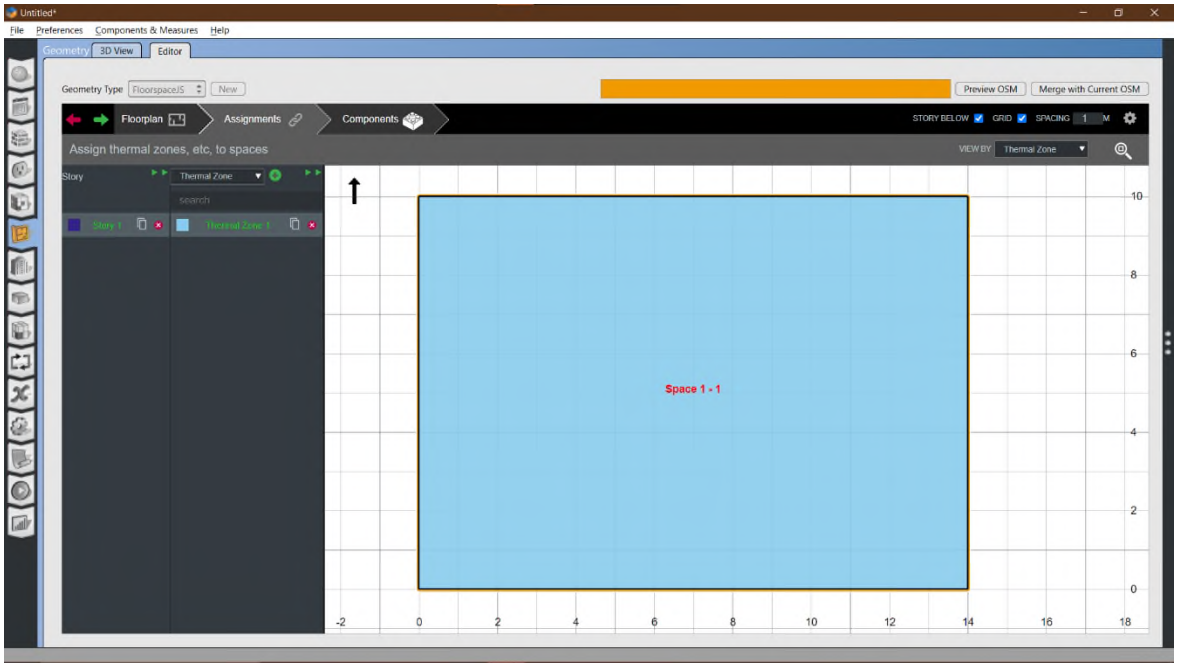

Figure 12: Assigning a thermal zone to the Space of the Simple model.

In the Components section, we place four windows and one door at the model as shown in Figure 13. We expand the Story and set the properties as shown in Figure 14. Next in the Floorplan section we expand the Space and specify the Construction set to be used (Figure 15).

In the same manner, in the components section, we expand the Window and Door options and we set their respective values (Figure 16) (Figure 17).

As soon as we have completed the floorplans of the Simple model, we can switch to the 3D view subtab of the Geometry tab, to access the 3D model (The roof has been disabled to see the interior) (Figure 18).

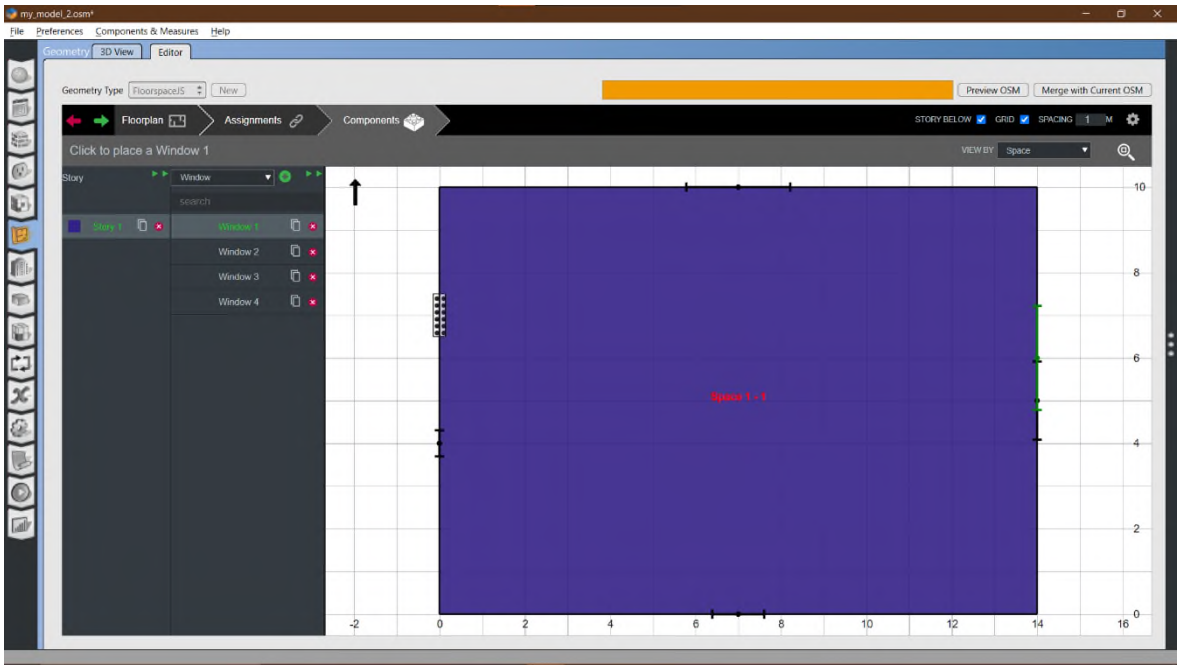

Figure 13: Set windows and door for the Simple model.

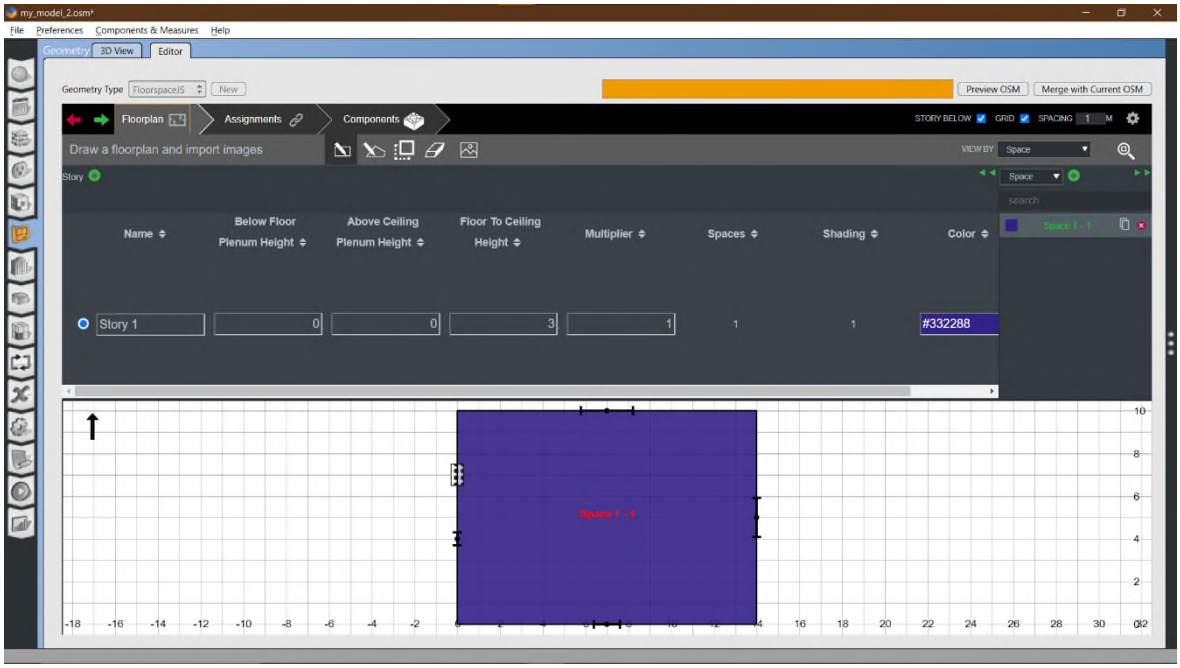

Figure 14: Expanding Story and setting values.

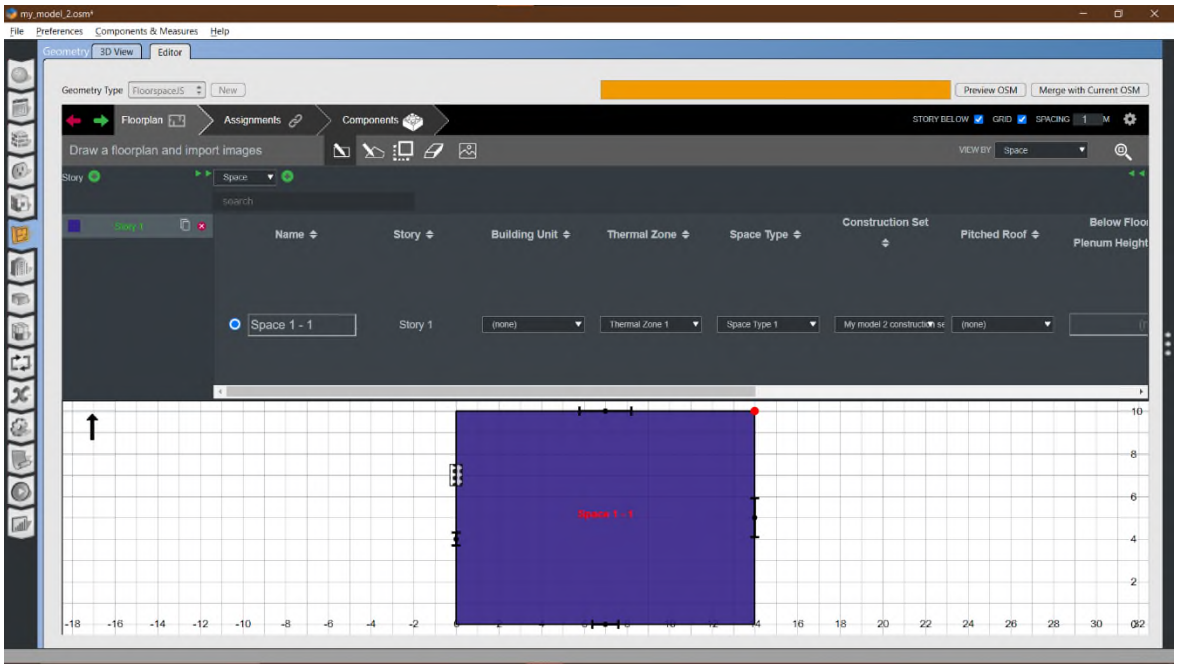

Figure 15: Expanding Space and setting values.

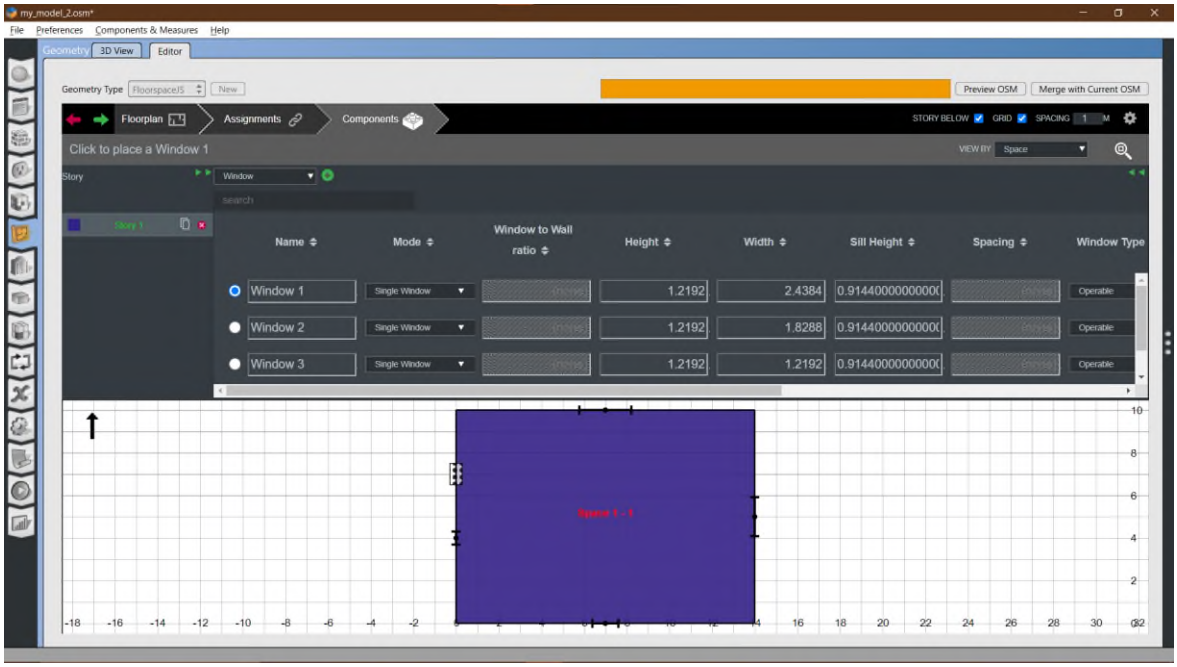

Figure 16: Expanding Window in Components section and setting values. Width of the fourth window is 0.6096.

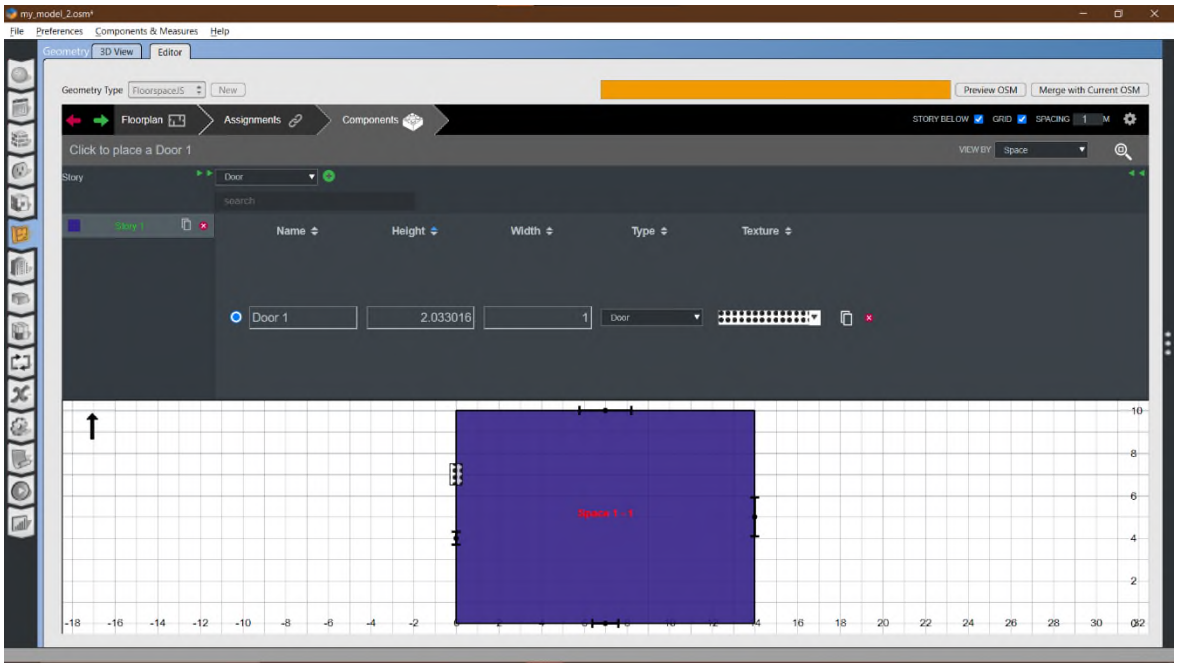

Figure 17: Expanding Door option in Components section and setting values.

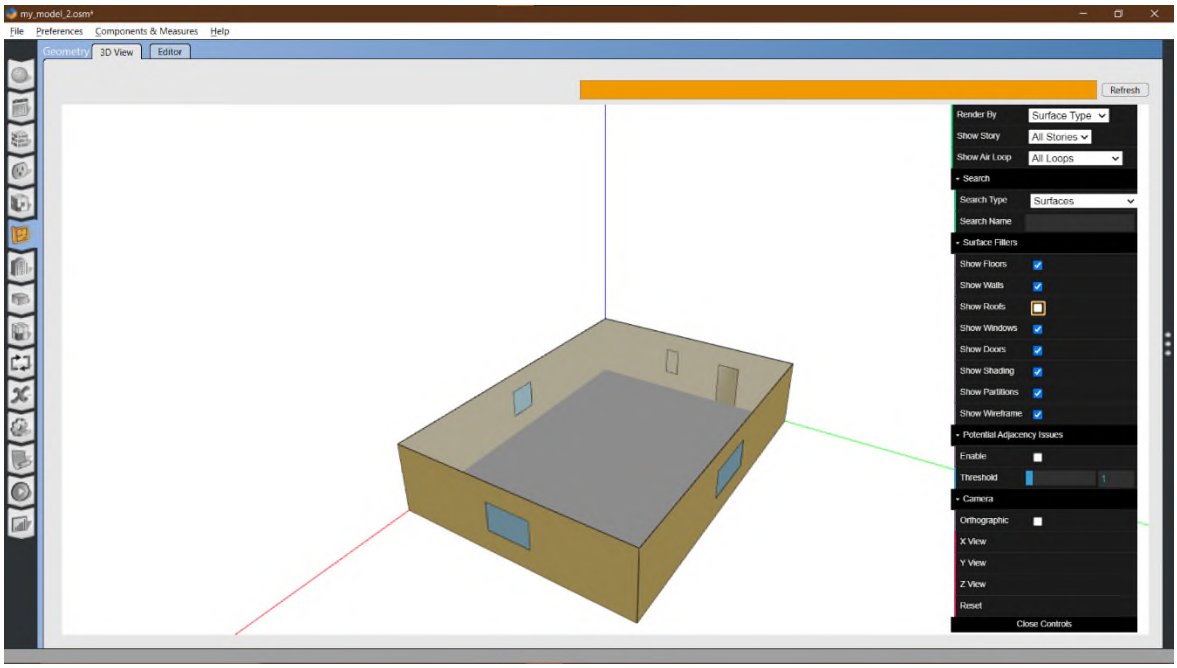

Figure 18: 3D view of the completed Simple model.

Continuing with the development of the Detailed model we demonstrate the steps to create the respective floorplan. Table 5 showcases the name of each room and its dimensions, and Table 6 showcases the properties of doors and windows. Figure 19 shows the complete floorplan with the floors and doors.

Table 5: x and y axis dimension for rooms.

	x-axis dimension (m)	y-axis dimension (m)
Master bedroom	7	4
Children bedroom	7	4
Hall 1	8	2
Hall 2	6	2
Living room	4	4
Family living room	4	4
Bathroom	2	2

Table 6: Properties of windows and doors.

	Mode	Height (m)	Width (m)	Sill height (m)	Type
Window	Single window	1.2192	0.6096	0.9144	Operable
Door		2.033016	1		Door
Interior door		2.033016	1		Door

An interesting thing to note is that OpenStudio understands that two Spaces as part of the same building through adjacent walls. So, for two Spaces to connect through doors, we need to place one door to each Space, and they must face each other. This is not possible using the embedded FloorspaceJS editor. Therefore, it is necessary to use the "Surface Matching" Measure. This Measure automatically places and matches any adjacent Surfaces and Sub-Surfaces.

Next, we assign the Space types (Figure 20).

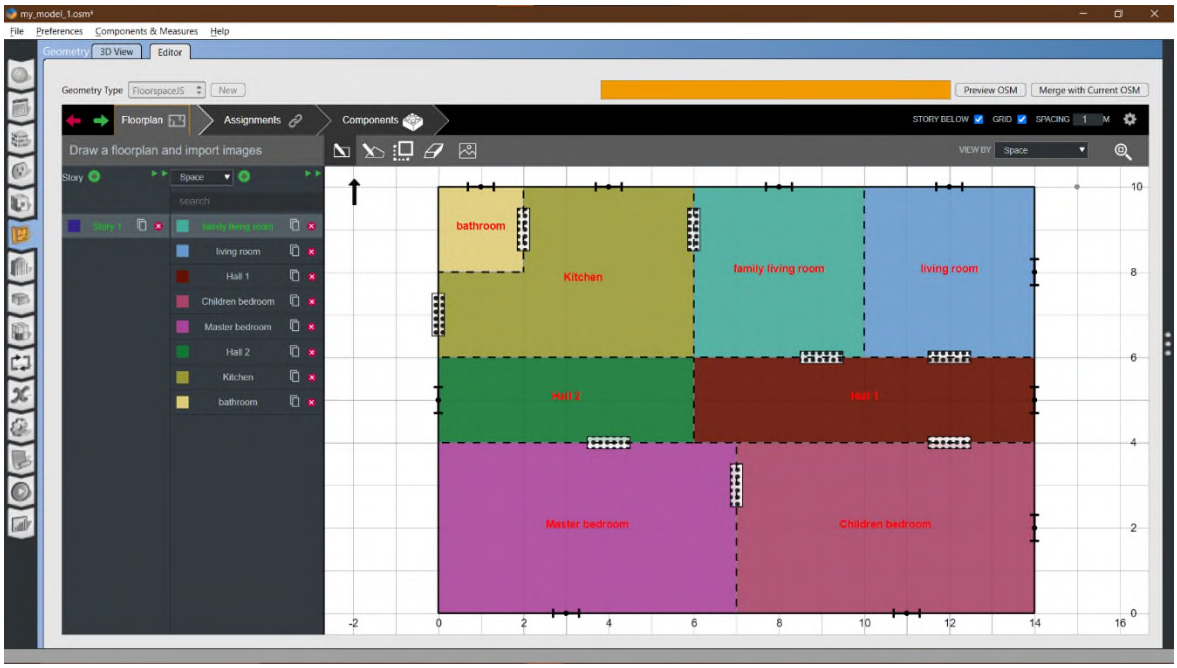

Figure 19: Complete floorplan of Detailed model.

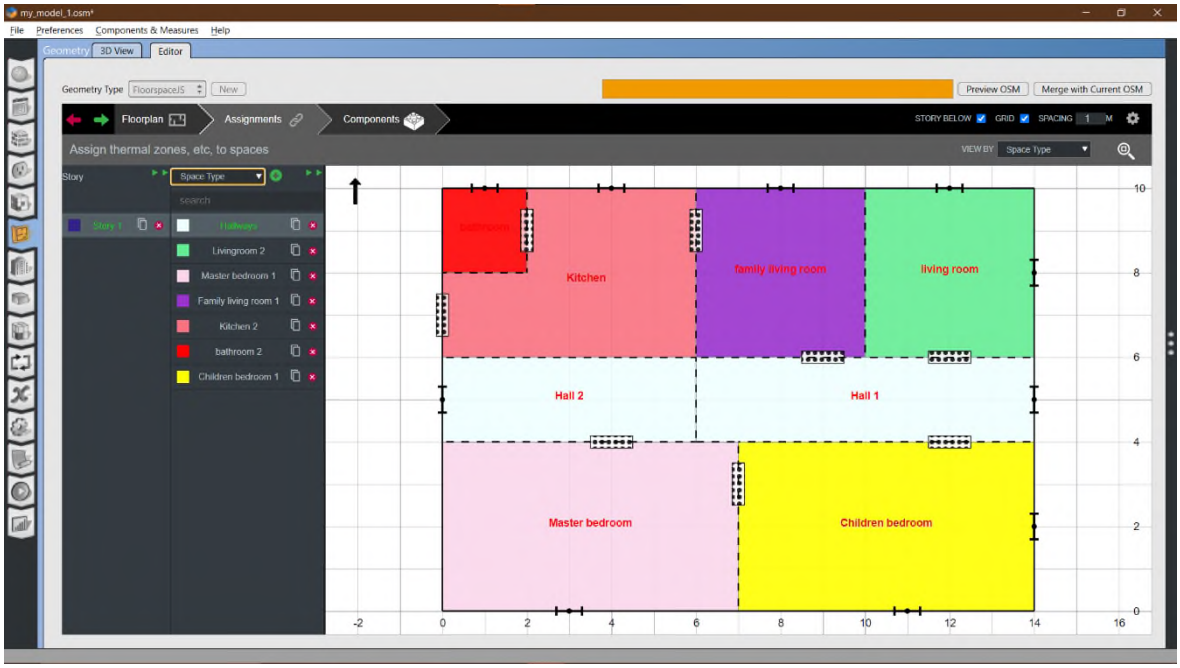

Figure 20: Assigning Space types to the Spaces of the Detailed model.

Now we assign our thermal zones. When assigning thermal zones, there are some rules of thumb that we can follow. If the surface areas of the Spaces are similar, the well-mixed assumption is most likely correct [23]. If one Space has more external windows and walls than another Space, it is more likely that the Space with less exterior Surfaces is warmer or colder than the Space with exterior Surfaces, especially on extremely hot or cold

days [23]. Such Spaces should not be combined into a single Thermal Zone. While grouping Spaces, we should always consider the orientation of the façades, for example, on the East and West façades into a single Thermal Zone may result in less desired outcomes since they may have significantly varied outside boundary conditions at the same time of day [23]. Although physical proximity is not required for simulation, it is doubtful that air in two Spaces would be adequately mixed if the Spaces were not contiguous or connected by a doorway or hallway that allows air to flow freely [23]. If the well-mixed assumption is not acceptable for the entire Space, huge Spaces may need to be divided into smaller Spaces [23]. Some Spaces may necessitate specific setpoints and schedules [23]. With these rules in mind, the Simple model has only one thermal zone, while the Detailed model has five. For the purposes of our thesis and the real life building our models are based on we assign the following thermal zones for the Detailed model, the two bedrooms make up the first, the two hallways the second, the family living room and the kitchen the third, the bathroom and living room make up the final two thermal zones (Figure 21).

Finally, we set the values in the expanded Story and Spaces, as well as in the Window and Door expansion with the values from Table 6, like in the Simple model. With these last steps, we have completed the geometry for the Detailed model. Figure 22 depicts the 3D view of the model (The roof has been disabled to be able to see the interior).

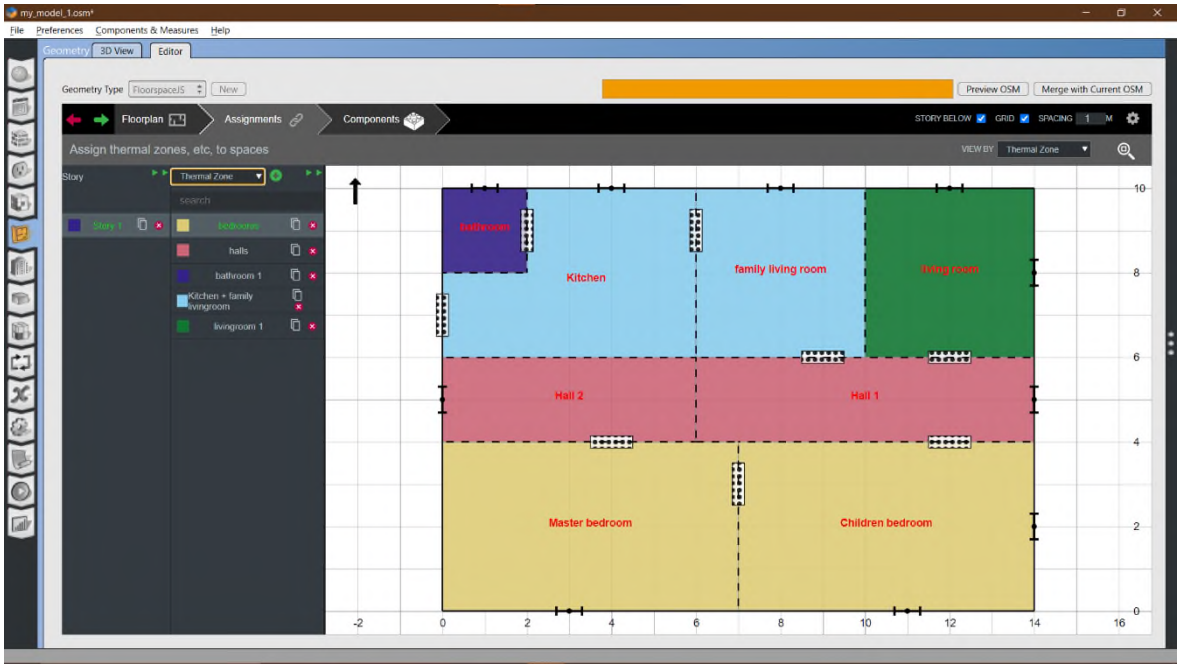

Figure 21: Assigning thermal zones to the Detailed model.

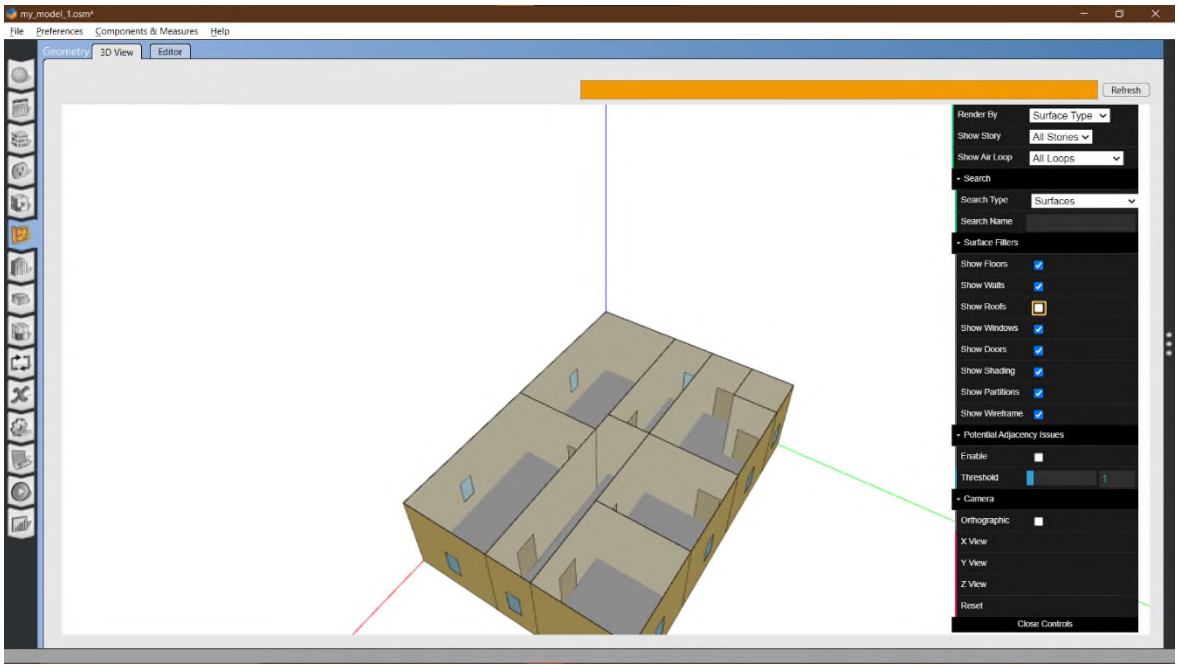

Figure 22: 3D view of the completed Detailed model.

2.3.4 Schedules tab

In this tab, we set the routines the residents are following, the occupancy of the different rooms throughout the day, as well as their activities. Here, we also set the schedules of the equipment, the lights time schedules and the threshold of cooling and heating thermostats in daily basis.

There are two subtabs, the Schedule Sets and Schedules. In the latter subtab, we create standalone schedules which can even be used independently, without needing to be part of a set. Grouping schedules in a set for a Space assigns them automatically to their corresponding load types. For example, when a Schedule Set is applied to a Space the respective electrical equipment Schedule is applied to the electric equipment loads in this Space. Moreover, we can define hourly based rulesets that correspond to weekends and holidays.

Lighting and Equipment schedules have usually values of 1 or 0, and are used mostly to state if the lights are or the electrical equipment is used or not. On the other hand, Occupancy schedules can have decimal values from 0 and 1. For example, if a Space is occupied by up to 4 people, then the schedule can have any of values: 0, 0.25, 0.5, 0.75 and 1, depending on the number of occupants. The level of an occupant activity is defined through an Activity Level schedule. It states the intensity of activity performed by a person

in Watts per person. Lastly, the Cooling and Heating schedules are set. They are responsible for the regulation of the temperature in a Space using Celsius degrees as units.

We assume that the family of four that is going to “live” in the models is as generic as can be, which means that both parents work an 8.00-16.00 shift, the kids go to school and do not have any extracurricular activities. When all the members are at home, except for breakfast and lunch when everyone is in the kitchen, the rest of the time the parents reside in the rooms of the kitchen and the family living room and the kids in their room, the children’s bedroom. If the members are at home, the equipment in each room that they reside in is considered in use, etc.

For the Simple model we created 8 schedules. An Activity, Equipment, Lighting and Occupancy schedule is shown in Figure 23 (a), (b), (c), (d), and (e) respectively.

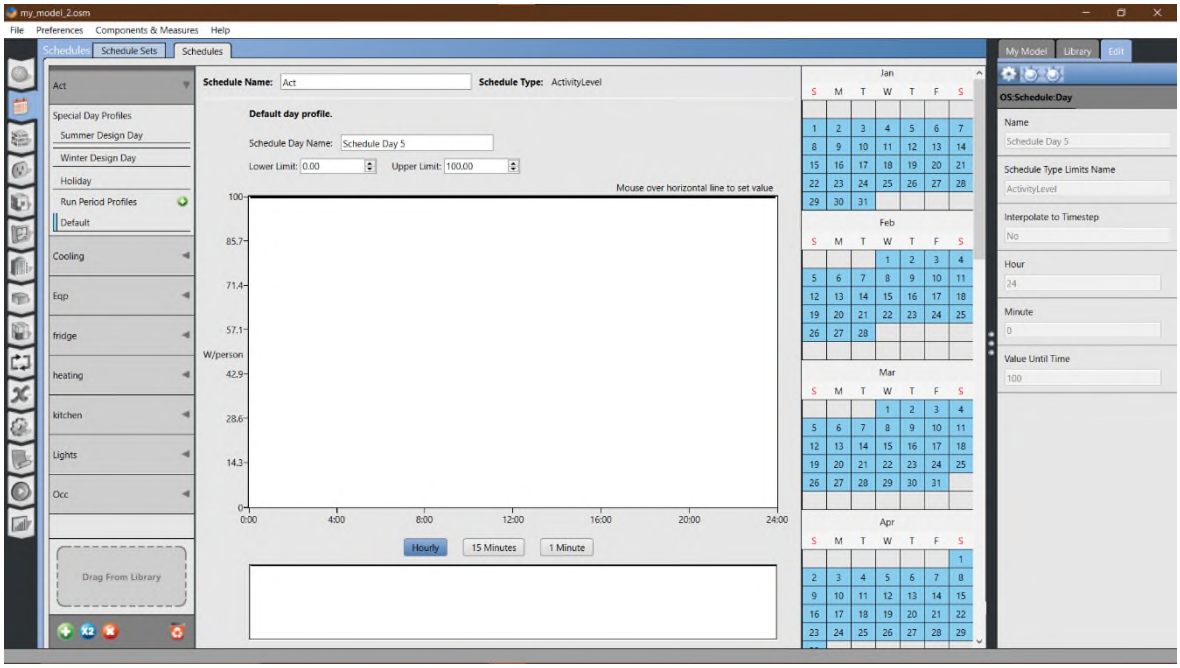

Figure 23(a): Activity schedule for the Simple model.

As we can see in the last two figures, there are two schedules corresponding to the Occupancy schedule. Because at the weekends, the children don’t have school, they stay at home. So, we created a sub schedule, alongside the default one, and put a priority tag to it, so when the schedule is checked during a weekend, it outputs the prioritized schedule.

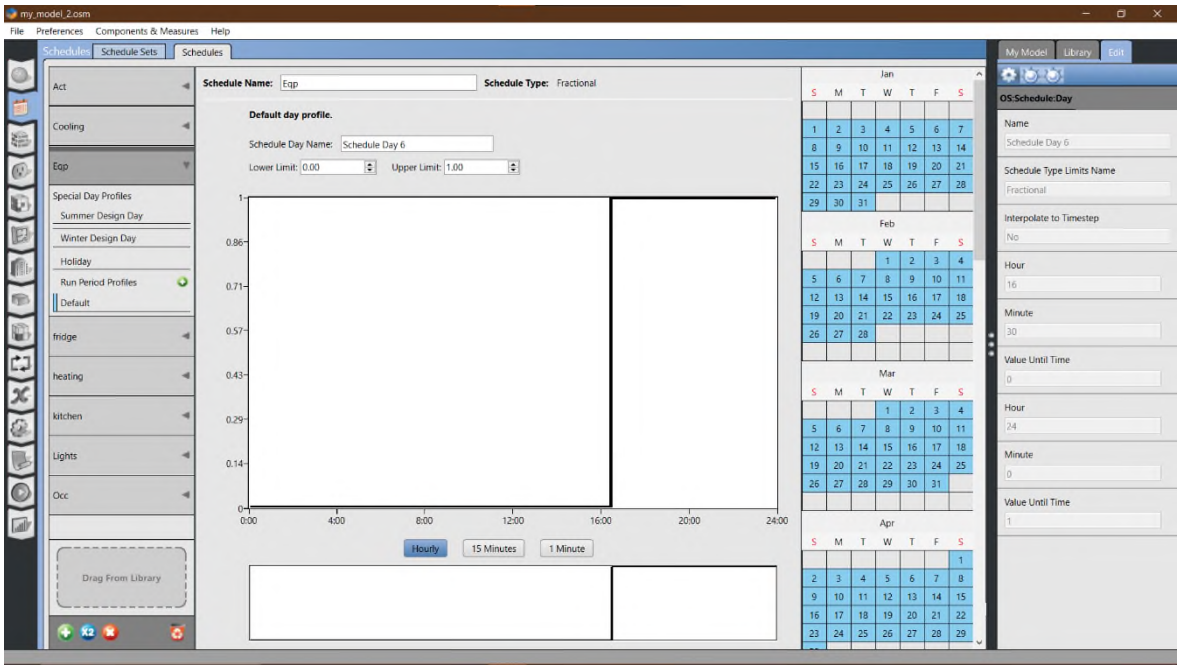

Figure 23(b): Equipment schedule for the Simple model.

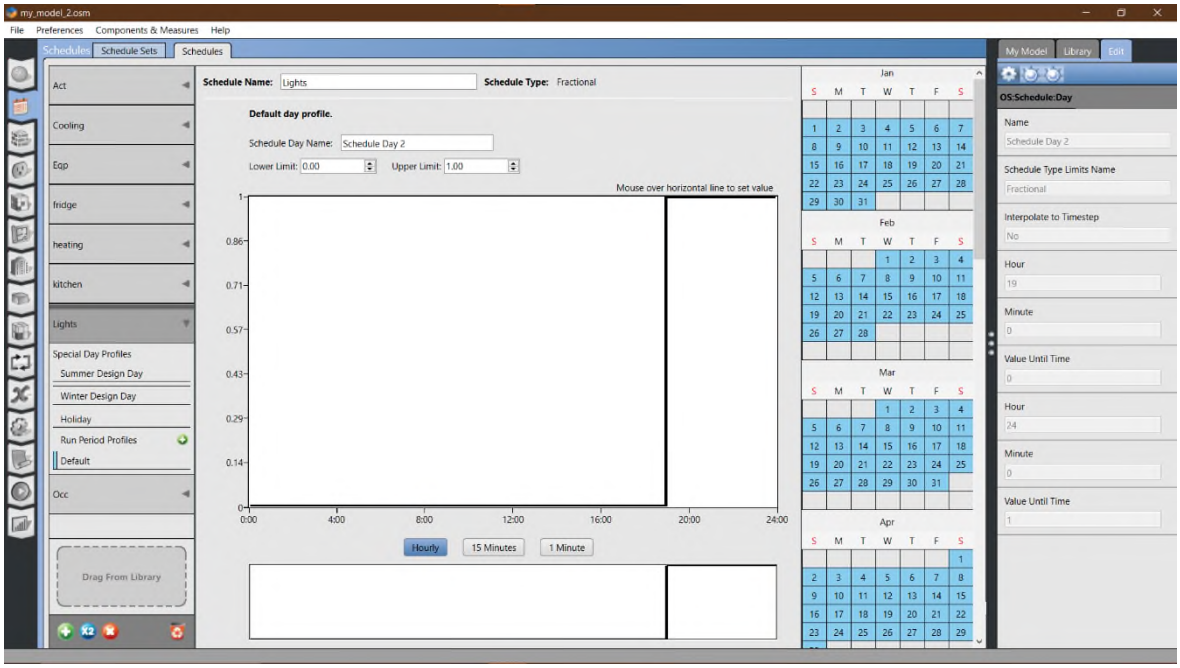

Figure 23(c): Lighting schedule for the Simple model.

These four schedules are collected in a set at the Schedule Sets subtab (Figure 24). The set is created by dragging and dropping our schedules in the corresponding fields. Also, we create four stand-alone schedules for the cooling, heating (Figure 25 (a) and (b)), fridge and the kitchen (Figure 26 (a) and (b)), since these can't be put in a set and don't conform with the schedule of regular equipment.

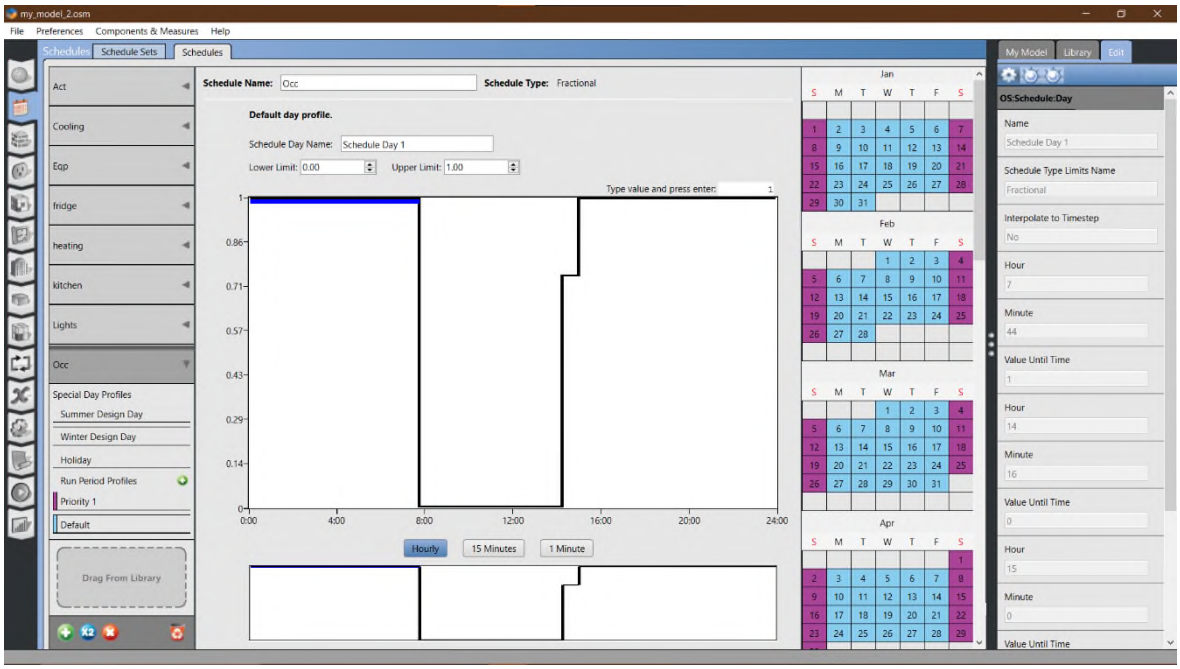

Figure 23(d): Occupancy schedule for the Simple model (Default schedule for the everyday).

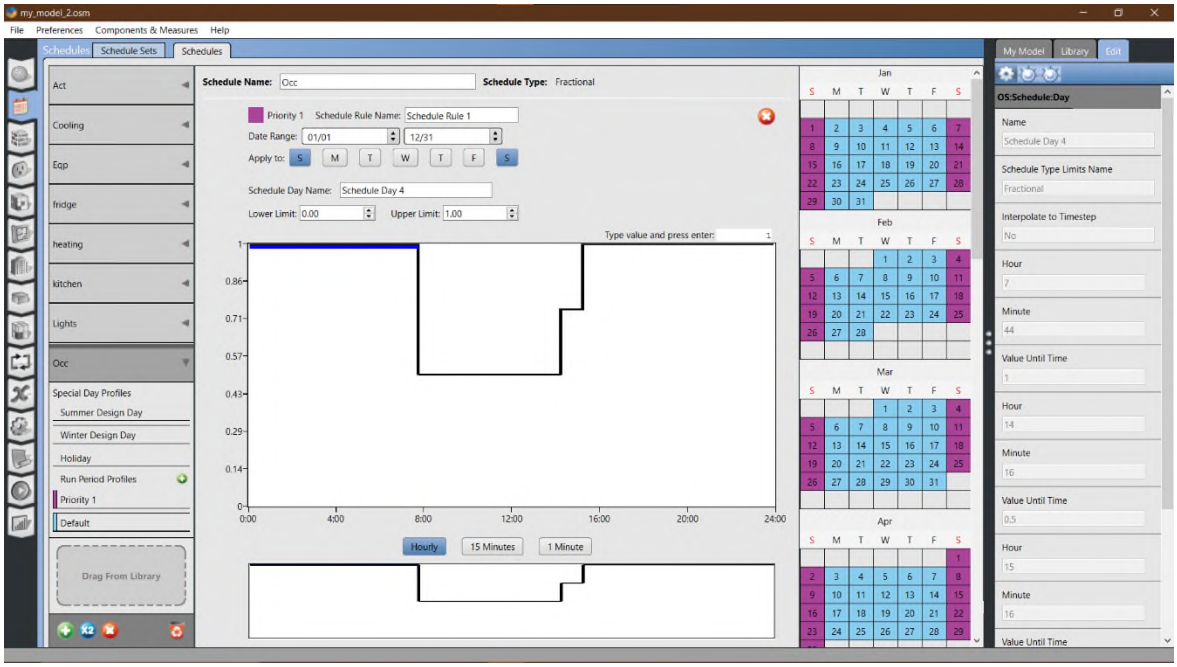

Figure 23(e): Occupancy schedule for the Simple model (Schedule for the weekends).

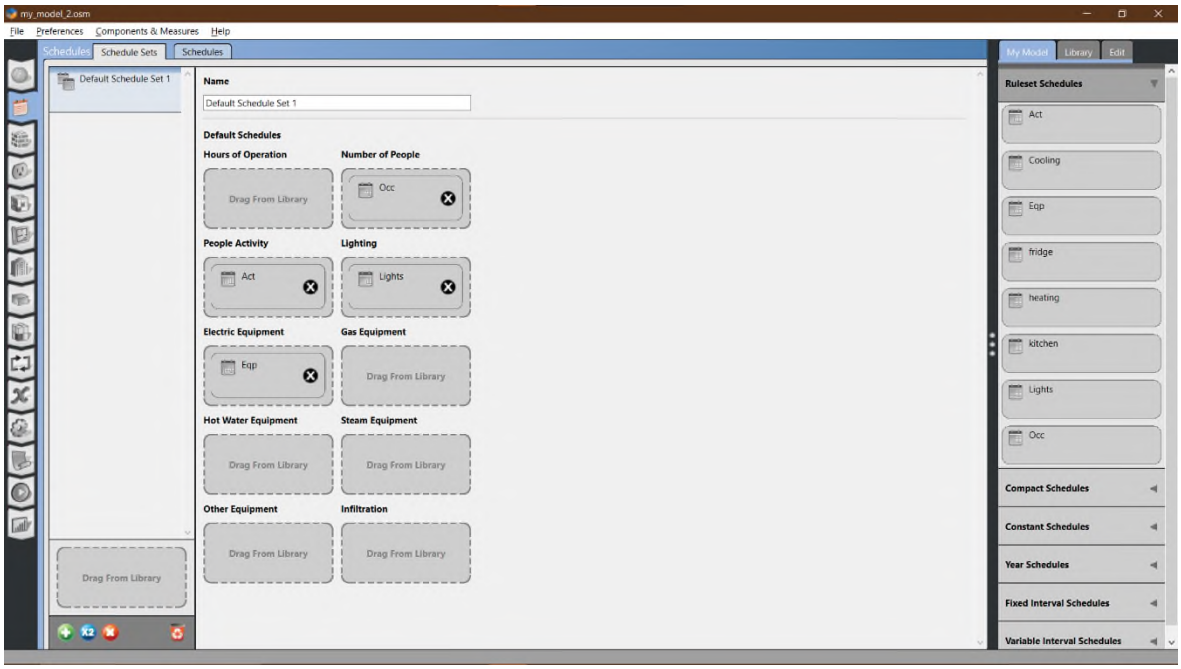

Figure 24: Schedule set for the Simple model.

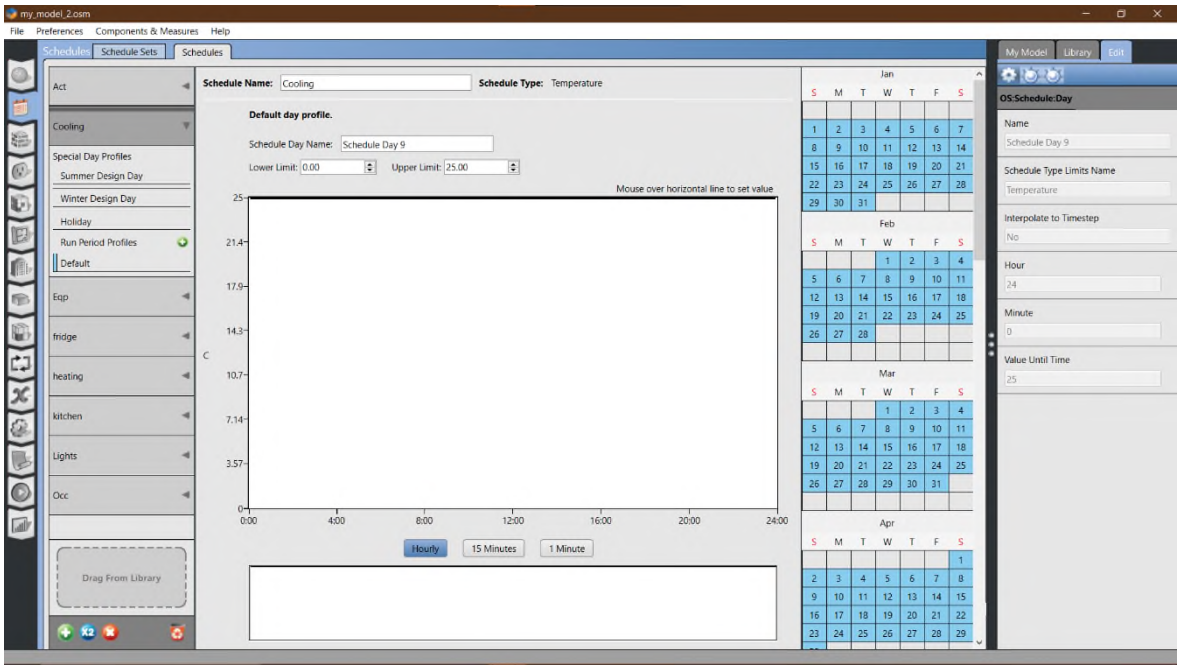

Figure 25(a): Cooling schedule for the Simple model.

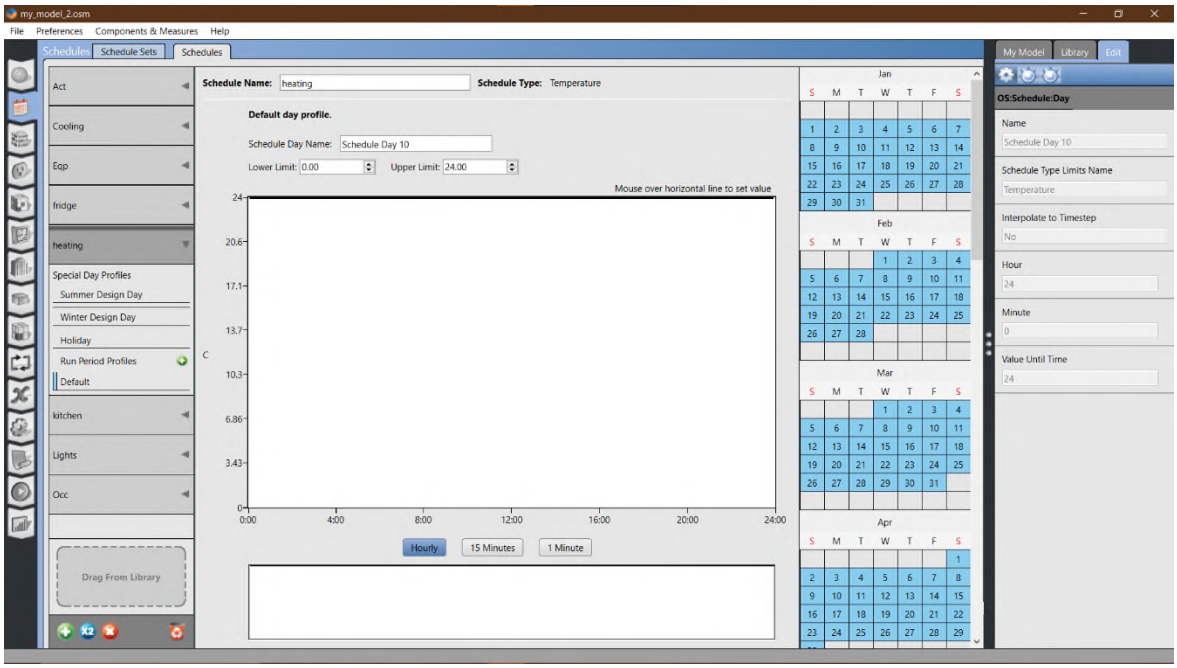

Figure 25(b): Heating schedule for the Simple model.

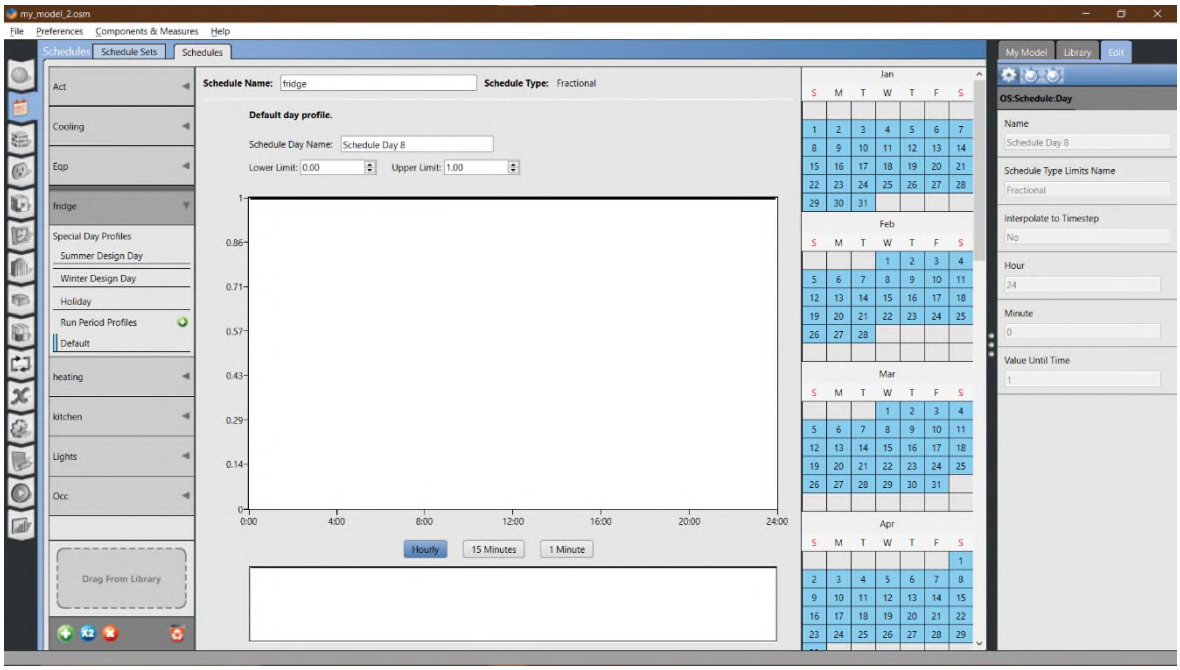

Figure 26(a): Fridge schedule for the Simple model.

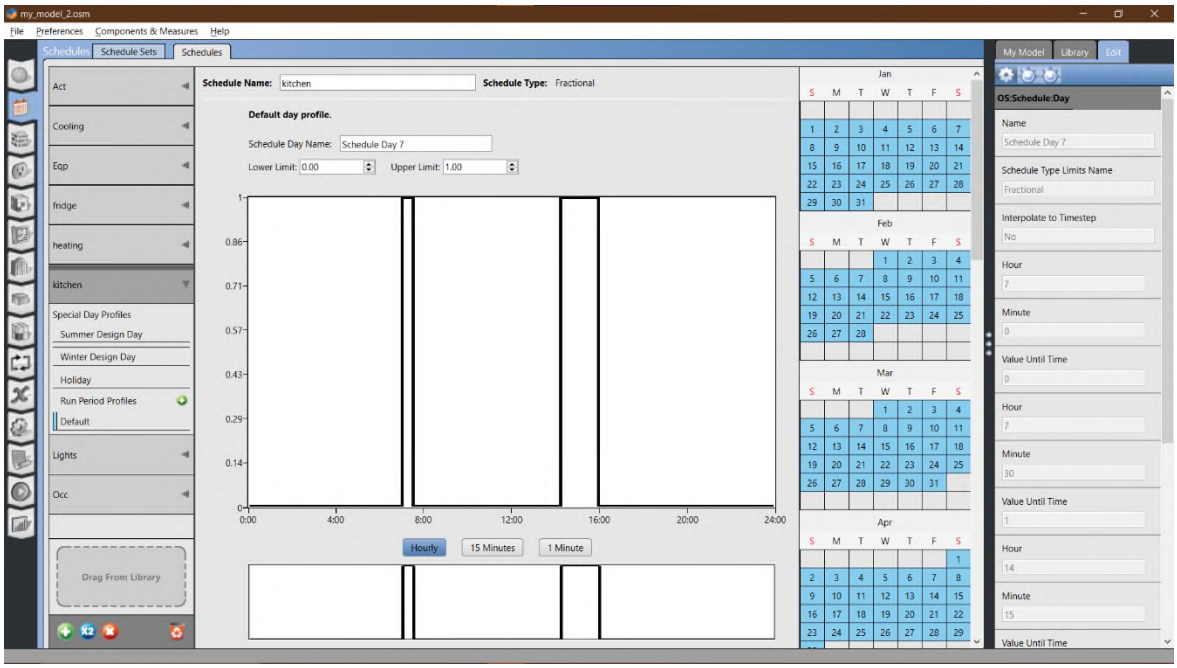

Figure 26(b): Kitchen schedule for the Simple model.

We have finished setting up the schedules and schedule sets for the Simple model. Continuing with our schedule planning for our Detailed model, since here we have a lot more rooms, we need to observe where most activity, occupancy and equipment are located throughout the days. Table 7 showcases the schedules and sets that we need to create for each Space type.

Table 7: Schedule sets for each Space type.

Schedule sets	Number of People	People Activity	Lighting	Electric Equipment
Bathroom set	Bathroom Occupancy	Bathroom Activity	Bathroom Lights	
Children Bedroom set	Children Bedroom Occupancy	Children Bedroom Activity	Children Bedroom Lights	Children Bedroom Equipment
Family living room set	Family living room Occupancy	Family living room Activity	Family living room Lights	Family living room Equipment
Hallways set			Hallways Lights	
Kitchen set	Kitchen Occupancy	Kitchen Activity	Kitchen Lights	
Livingroom set			Livingroom Lights	
Master bedroom set	Master bedroom Occupancy	Master bedroom Activity	Master bedroom Lights	

We also create some stand-alone schedules for the cooling, heating, fridge and kitchen, for the same reasons we had with the Simple model. We make some of these schedules more detailed by adding one or two priority schedules which correspond for the fifteen days of winter holidays and the fifteen days of summer break. For the sake of not wanting to tire, even more, the reader, we showcase only one detailed schedule (Figure 27 (a), (b), (c)), since most of them follow similar logic. If we were to showcase them all, it would total to 51 figures.

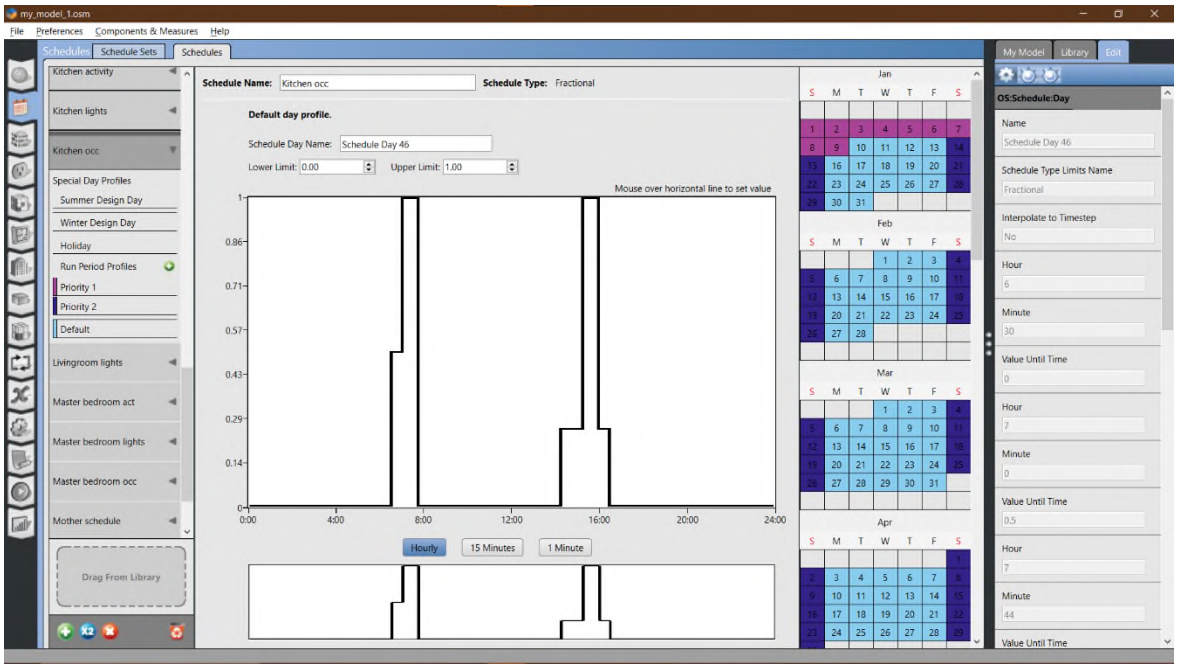

Figure 27(a): Default, everyday schedule for the occupancy of the kitchen Space.

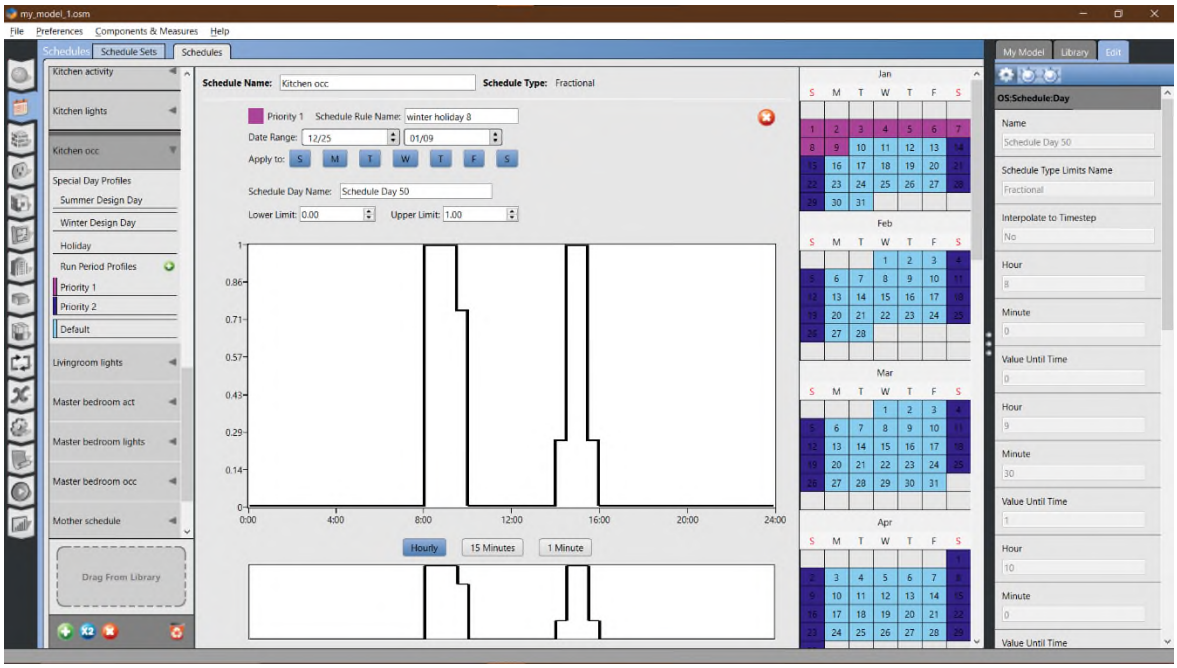

Figure 27(b): Schedule for the occupancy of the kitchen during winter holidays.

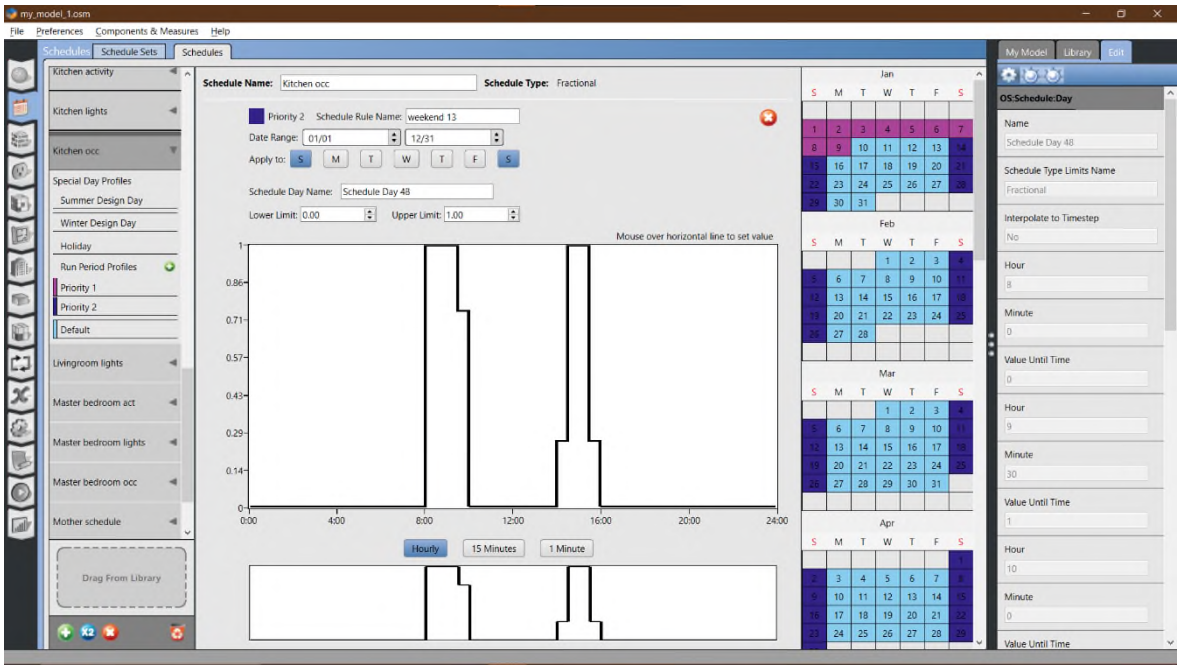

Figure 27(c): Schedule for the occupancy of the kitchen during weekends.

2.3.5 Loads tab

The next tab we are going to work on is the Loads tab. Like in the previous steps, we start with the Simple model to demonstrate the basics and then move on to the Detailed one. There are several categories to define loads, but we are going to focus on three of

them: The People Definitions, Lights Definitions and Electric Equipment Definitions. People are substantial thermal loads in Spaces, but they are addressed differently than lighting or equipment. Occupancy, like other Loads, can be defined by the number of persons present in a Space or by occupant density [23]. People are also defined by the amount of radiant energy they emit into an area [23]. The remainder of the heat rejected is divided into sensible[1] and latent heat[2] addition [23]. Based on normal metabolic rates and heat rejection, EnergyPlus can automatically determine the proportion between sensible and latent heat gain from occupants [23]. We define the family of four as one People Definition, as shown in Figure 28 for the Simple model.

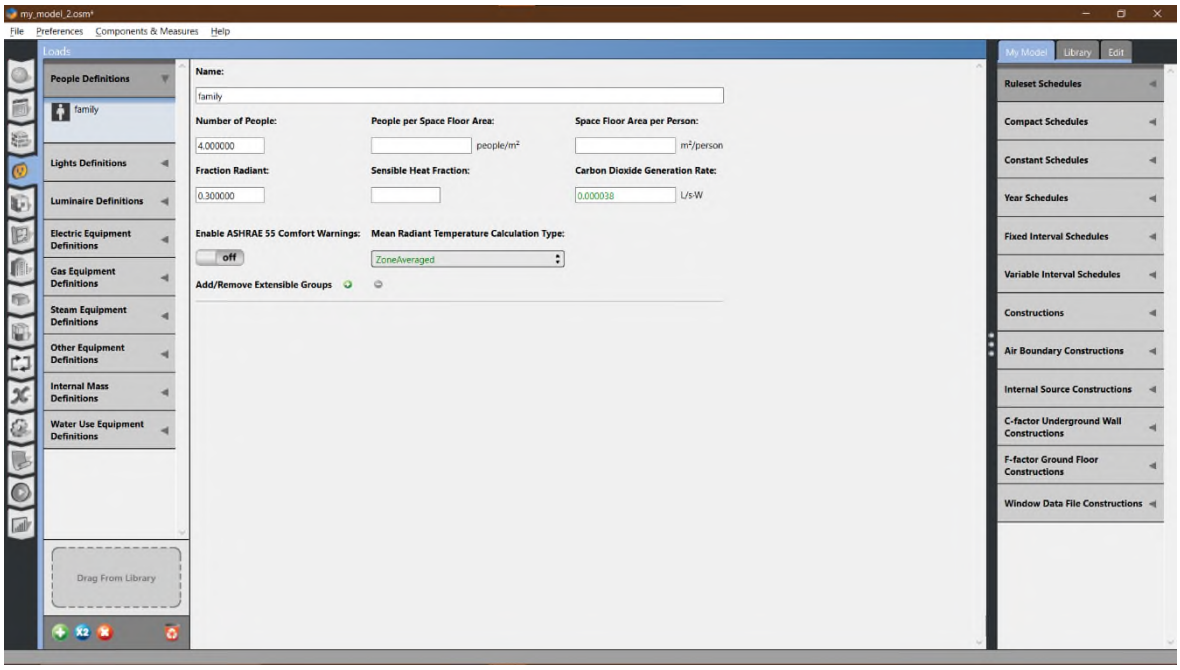

Figure 28: People Definitions for the Simple model.

EnergyPlus cannot be used to simulate the distribution of electric lights or to verify that a particular lighting plan produces adequate illumination [23]. The electric energy needed by lights and luminaires is accounted for during an EnergyPlus simulation, as is the thermal impact of this energy usage on the surrounding Space [23]. The typical lighting load is provided in watts per square foot [23]. This is also known as Lighting Power Density (LPD) and is a typical technique of describing loads such as linear fluorescent lighting that covers

[1] Sensible heat: "Sensible heat is the energy needed to raise the temperature of a substance" [24].

[2] Latent heat: "Latent heat is the energy needed to overcome the intermolecular forces to trigger a phase change" [24].

a vast area, especially when comprehensive information about the lighting design is unavailable [23]. The heat transfer mechanisms from Lights and Luminaires are the Fraction Radiant, which is a portion of energy radiated to the Space's Surfaces as longwave radiation, the Fraction Visible, which is a portion of energy rejected to air leaving the space to an air handler. And the Fraction Convected, which is a portion of energy rejected to the Space's air volume [23]. The fraction convected is computed rather than inputted explicitly, using the equation: $f_{convected} = 1 - f_{radiant} - f_{visible} – f_{return\ air}$ [23]. The Lights Definitions we set up is shown in Figure 29 for the Simple model.

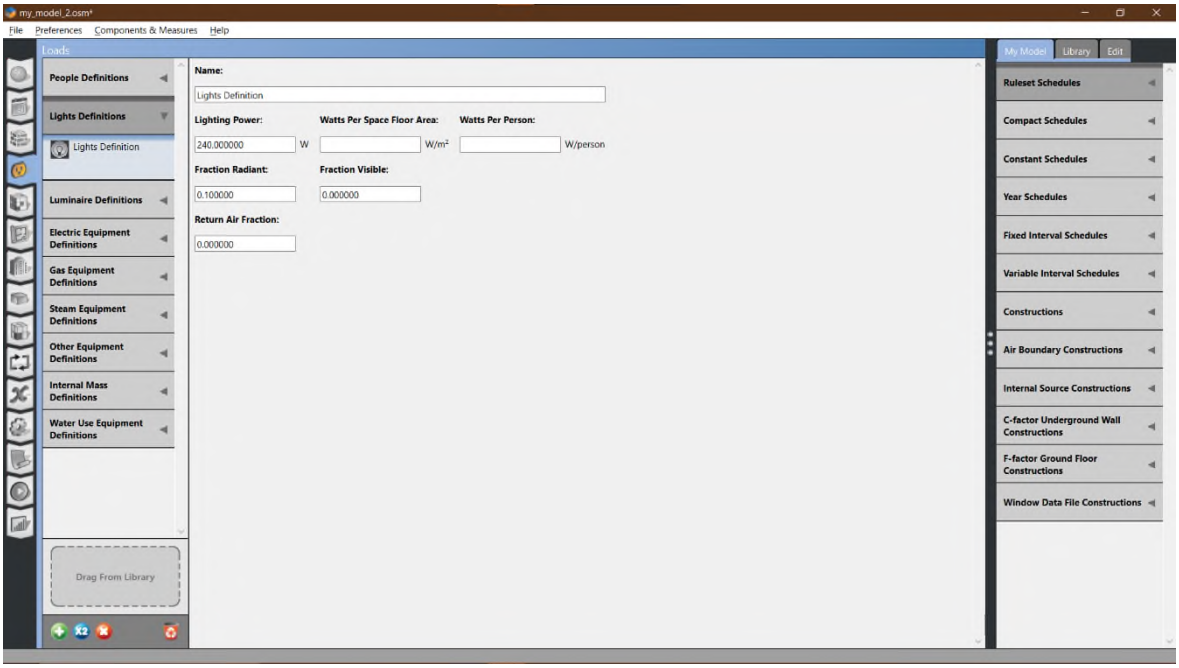

Figure 29: Lights Definitions of the Simple model.

EnergyPlus, like lighting, does not reflect specific electric, gas, steam, or other Equipment within Spaces. To EnergyPlus, however, they are merely machines that use power and emit heat into space. The only way EnergyPlus differs is in its maximum power draw, operating schedule, and heat transfer mechanisms into space. Equipment, like lights, can be measured in terms of power per unit, floor area, or person. However, unlike Lights, Equipment may use a variety of fuels (for example, electricity, gas, steam, or another). The simulation findings account for each fuel type independently. The heat transfer mechanisms permitted for Equipment objects are the Fraction Latent - a portion of energy added to the Space's air volume as moisture, the Fraction Radiant - a portion of energy radiated to the Space's Surfaces as longwave radiation, the Fraction Lost - a portion of energy that does not impact the Space's heat balance, either by performing useful

mechanical work or by rejection outside the Space, and the Fraction Convected - a portion of energy rejected to the Space. Fraction Convected is computed using the following equation rather than just entering it: $f_{convected} = 1 - f_{latent} - f_{radiant} - f_{lost}$ [23]. We create four definitions for each laptop with the same properties (Figure 30), one definition for the kitchen (Figure 31) and one for the fridge (Figure 32).

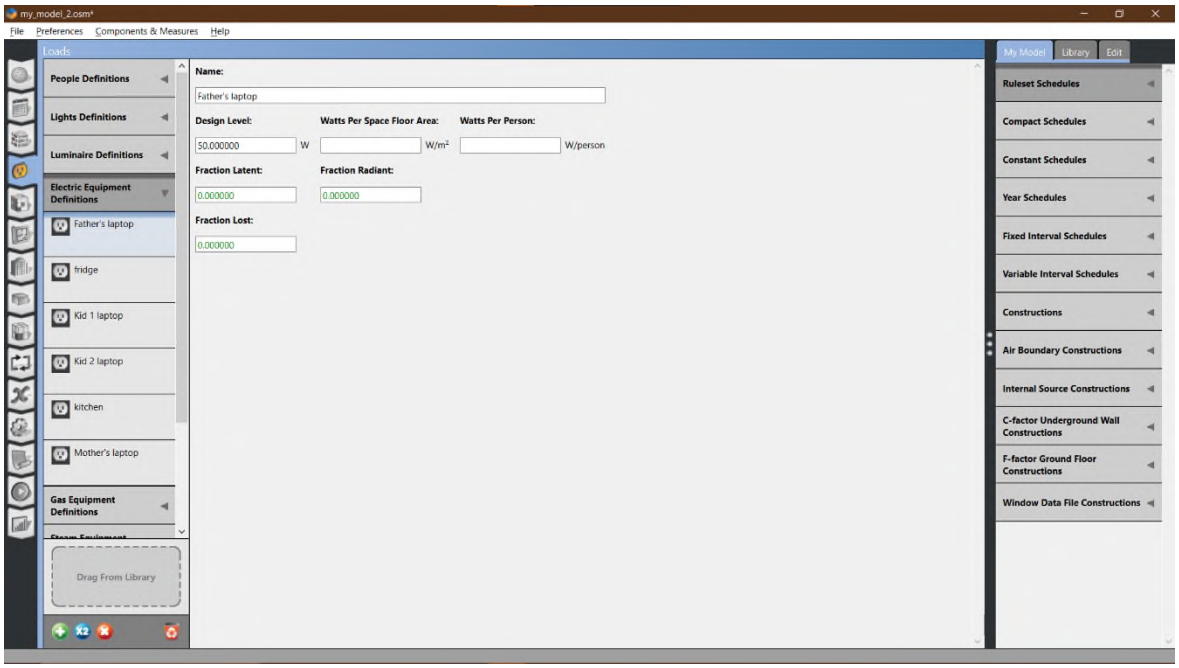

Figure 30: An equipment definition for a laptop, the other laptop definitions have the exact same properties.

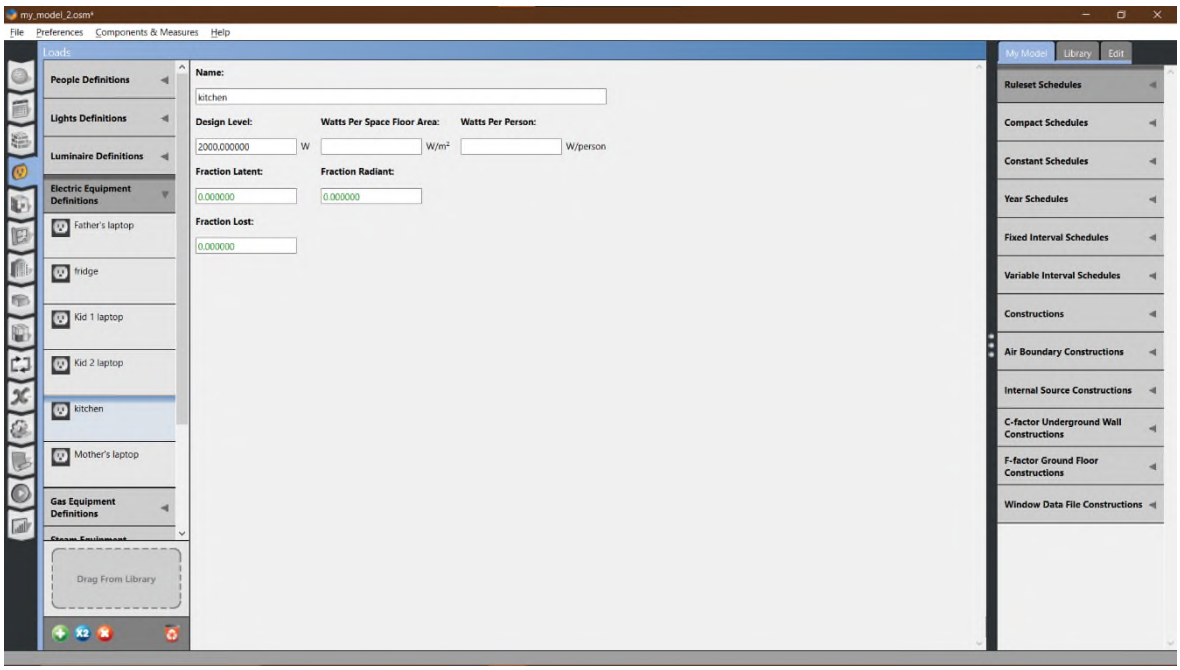

Figure 31: Equipment definition for the kitchen.

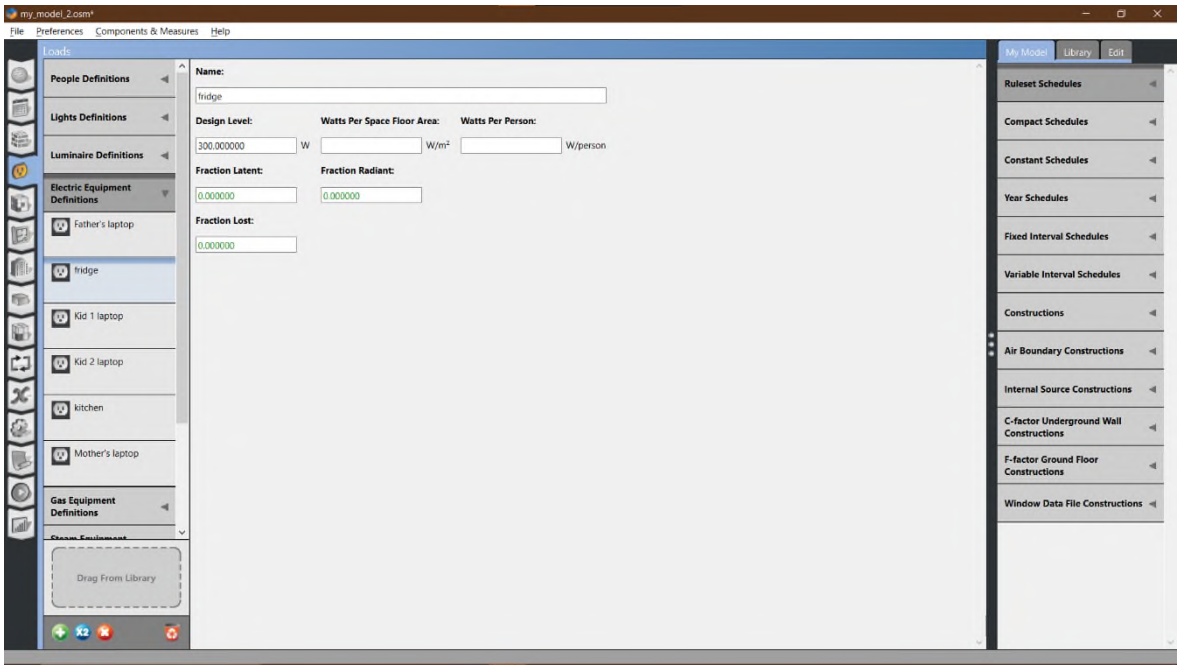

Figure 32: Equipment definition for the fridge.

With the definition of the Simple model finished, we move to the Detailed model. We use the same definitions for the lights and electric equipment, but we add people definitions for each main room. The main rooms are the bathroom, family living room, kitchen, and two bedrooms. The kitchen can end up having four members at once, while the rest of the main rooms have up to two members at once. So, the first two main rooms' people's definitions are the same as people definition of the Simple model and the other three have a value of 2 in the "Number of People" field (Figure 33).

We have now finished with the Loads tab for both models, and we move to the Space Types and Spaces tabs.

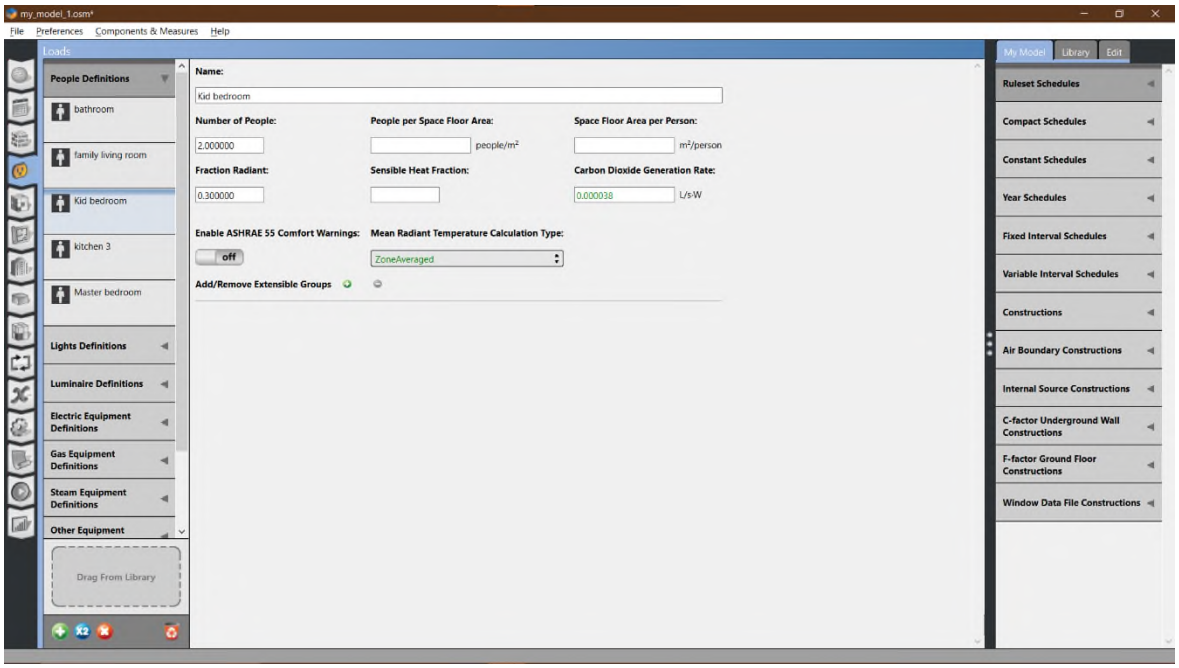

Figure 33: People definition for a Space that can end up with 2 people at once, specifically the children bedroom.

2.3.6 Space Types and Spaces tabs

We start with the Space Types tab, in this tab we assign Construction sets, schedules sets and definitions to different Space types. Again, we start with the Simple model. The Space Types tab is divided into four subtabs, we are going to focus on the first two, the General and Loads subtabs. In the General subtab, we assign the Default Construction Set and Default Schedule Set that is followed by the Loads, as well as other things like the Design Specification Outdoor Air, by dragging and dropping our created Construction and schedule sets from the right column under the My Model option (Figure 34). In the Loads subtab, we assign the loads that can be observed in the Space type, so we drag and drop the people, lights and electric equipment definitions we created previously to the Definitions column. Once in place, all definitions are assigned automatically their corresponding schedule from the set that we specified for this Space type, except for the kitchen and the fridge. For the latter two, we must manually drag and drop their specific Ruleset schedules we created since they don't fall under the regular electric equipment

schedules (Figure 35). An important thing to note is that only the people definition can be assigned an activity schedule.

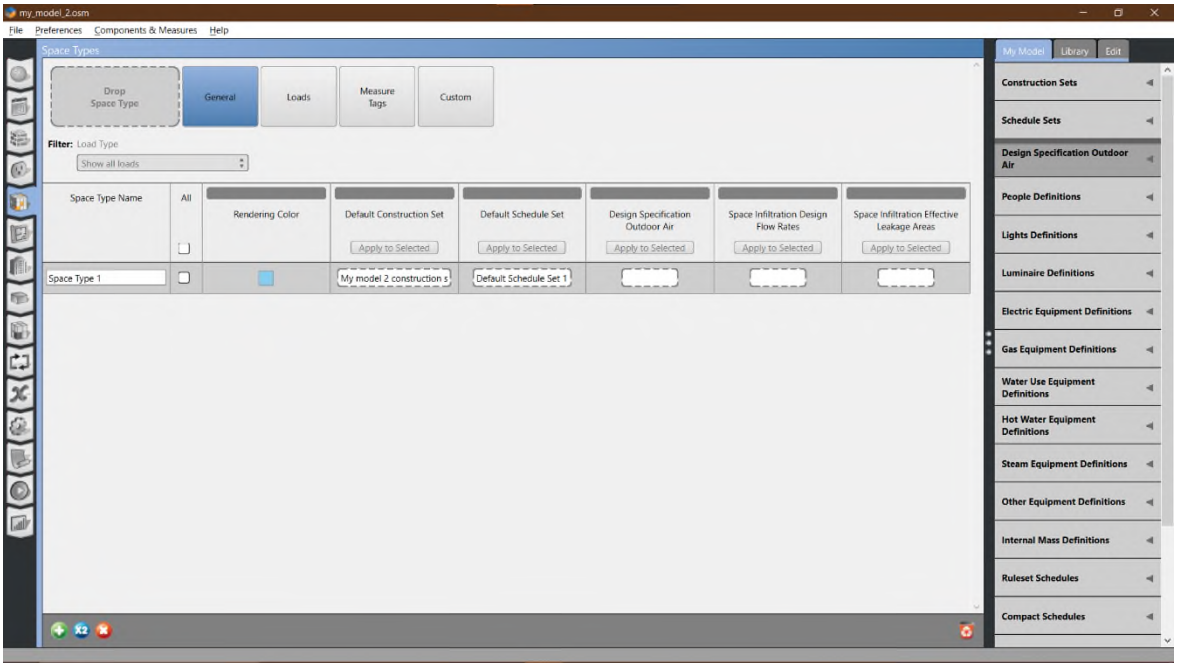

Figure 34: Assigning a construction and a schedule set to the Space type of the abstrcat model.

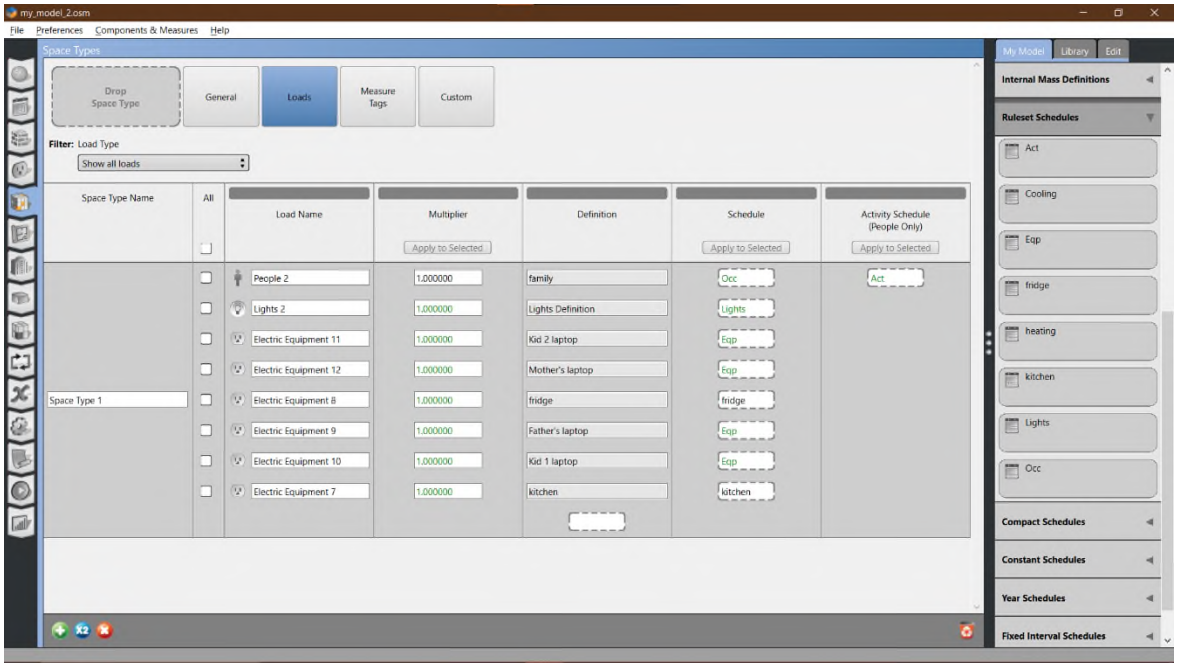

Figure 35: Loads definitions assigned to the Space type of the Simple model, with their corresponding schedules.

As soon as we define a Space type, we can assign it to a specific Space. Thus, the Space inherits all the information, be it Construction sets, schedule sets and loads. Moving

on to the Detailed model, here we have more Space types, and we follow the same steps for each one. Starting with the General subtab, we assign to each Space type the same Construction set and their specific schedule set we created previously (Figure 36).

On the Loads subtab, we assign the people definitions we created to the corresponding Space type we created them for. Also, we add the Lights and Electric Equipment loads, as shown in Table 8.

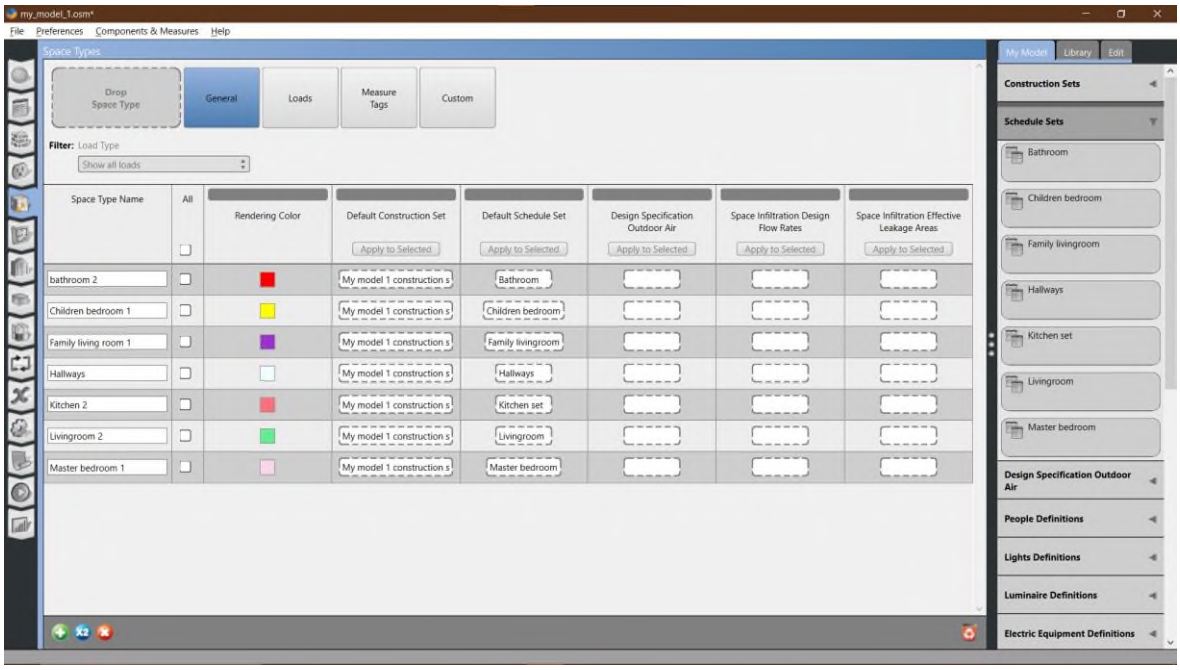

Figure 36: Assigning the construction set and the schedule sets to the Space types of the Detailed model.

Table 8: Space types with their people, lights and electric equipment definitions.

Space Types	**Definitions**
Bathroom	Bathroom (People definition) Lights (Lights definition)
Children Bedroom	Kids bedroom (People definition) Lights Kid 1 laptop (Electric equipment definition) Kid 2 laptop (Electric equipment definition)
Family living room	Family living room (People definition) Lights Father's laptop (Electric equipment definition) Mother's laptop (Electric equipment definition)
Hallways	Lights
Kitchen	Kitchen (People definition) Lights Kitchen (Electric equipment definition) Fridge (Electric equipment definition)
Living room	Lights
Master bedroom	Master bedroom (People definition) Lights

As we mentioned previously, once we assign the definitions, each one automatically inherits the corresponding schedule that exists in the schedule set of the Space type. Regarding the electric stove and fridge, we assign their standalone schedules.

Now that we have completed our Space types for both models, we can continue to the Spaces tab. Here we assign Space types to our Spaces. Fortunately, we have already done this when designing the floorplan of the models and assigned the Space types and thermal zones. This tab is divided into six subtabs, but we are only interested in the first four: Properties, Loads, Surfaces, and Subsurfaces. The properties subtab shows the different Spaces of the model, the assigned Space types and thermal zones, as well as the Construction and schedule sets for each one (Figure 37, Figure 38). Loads provide an overview of the different loads that exist in each Space (Figure 39, Figure 40 (a), (b)). Surfaces and Subsurfaces show information of the Surfaces and Sub-Surfaces of each Space like their type (for example, if they are a regular wall, roof, ceiling, door, or what type of window) and what Construction they use (Figure 41, Figure 42, Figure 43, Figure 44). A very important piece of information is the Outside Boundary Condition and Outside Boundary Condition Object. The first one shows if the surface or subsurface is connected to the exterior of the building, is connected to the ground or is a ceiling. Outside Boundary Condition Object shows the adjacent surface or subsurface that connects two Spaces. For example, in Figure 44 we have the subsurface “Face 73” which is a door, specifically the bathroom door that connects to the kitchen. If we hadn’t applied the “SurfaceMatching” Measure, the Outside Boundary Condition Object field would be empty, which is a mistake, and it would cause problems during the simulation. We could of course have corrected the mistake manually by searching in the 3D view subtab of the geometry tab, which Face is adjacent to this door, but this method is clearly inferior to just applying the Measure.

This tab serves as a good way to see if we have made the correct assignments to the Spaces of the model.

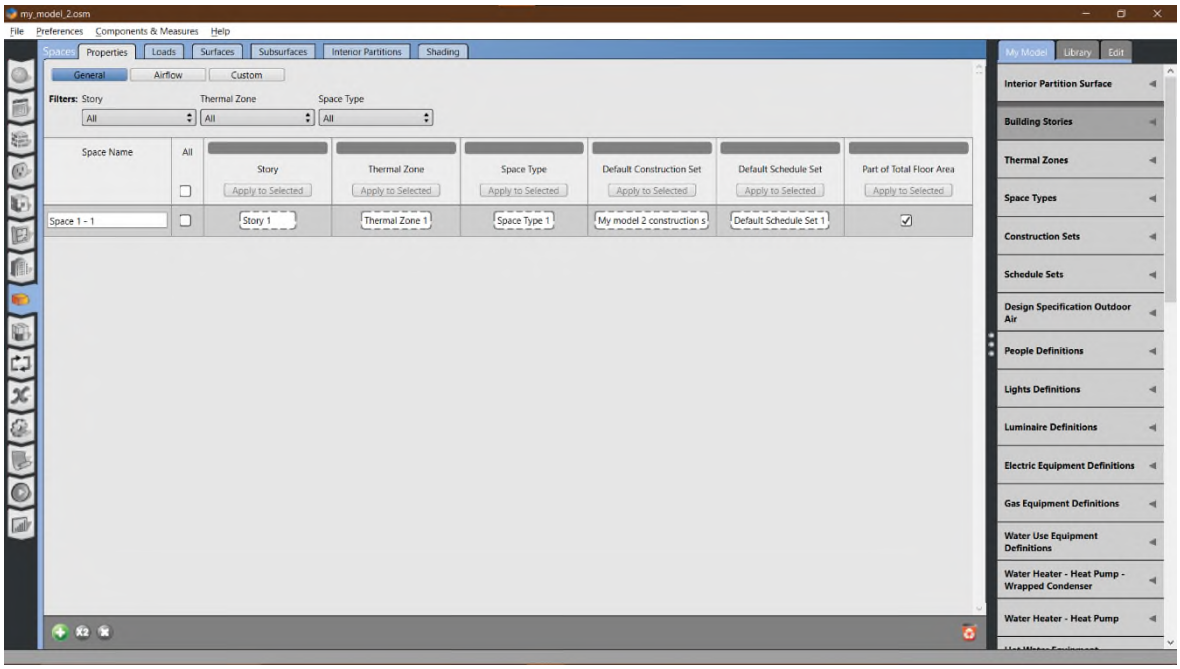

Figure 37: Properties subtab of the Spaces tab of the Simple model.

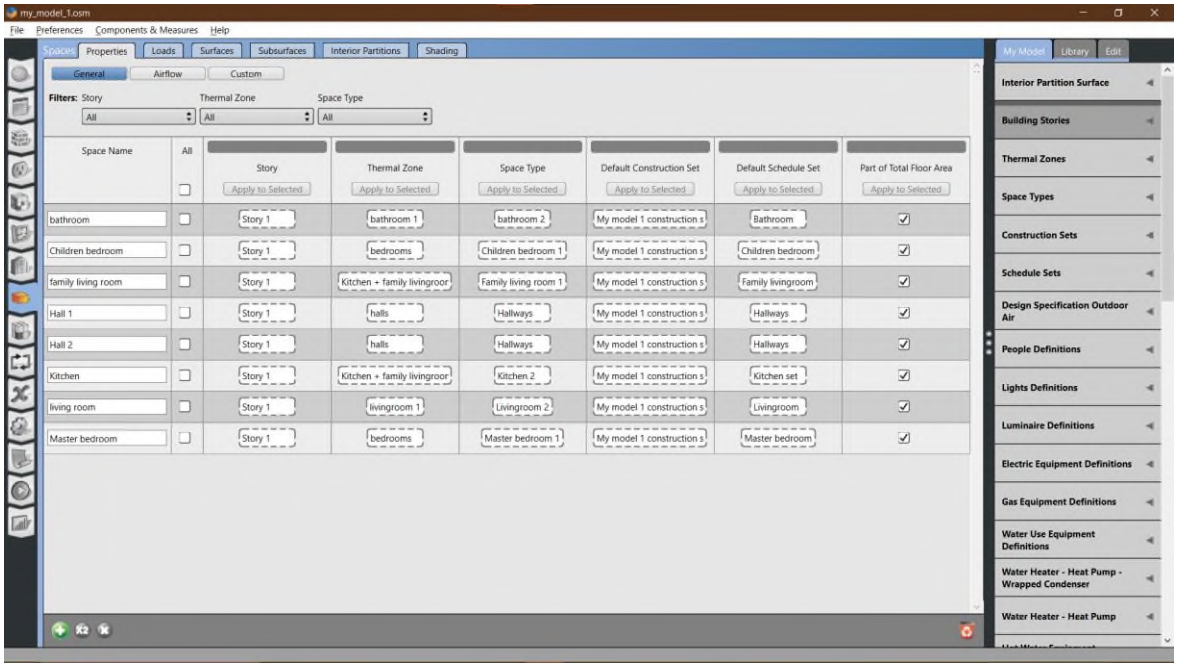

Figure 38: Properties subtab of the Spaces tab of the Detailed model.

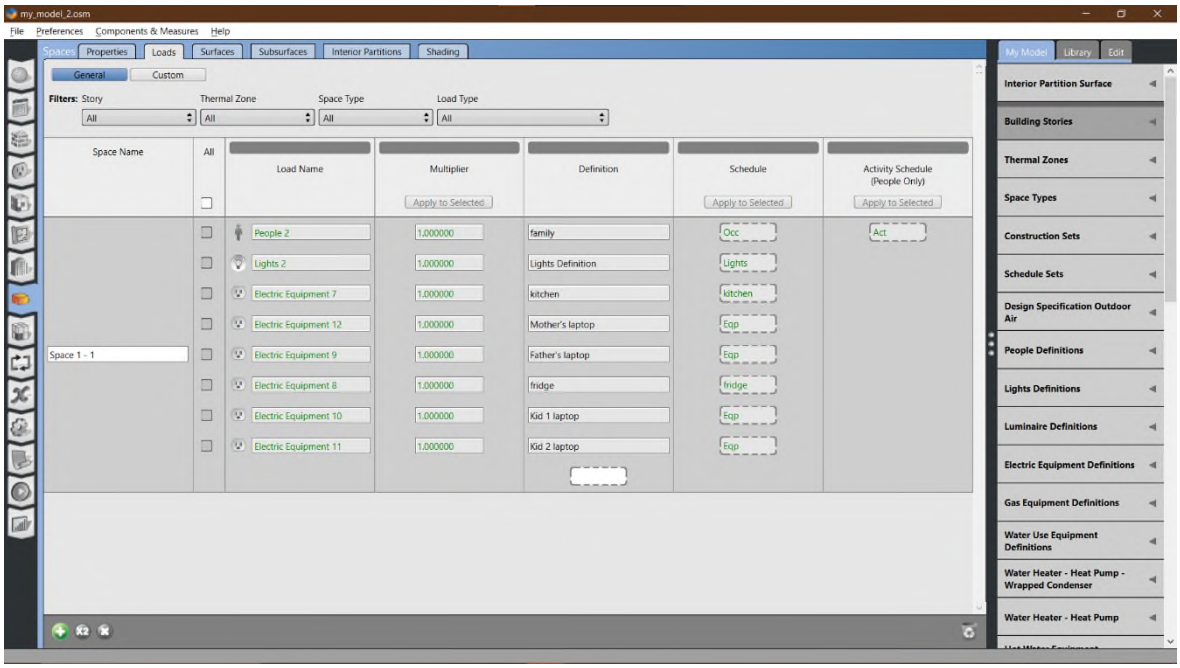

Figure 39: Loads subtab of the Spaces tab of the Simple model.

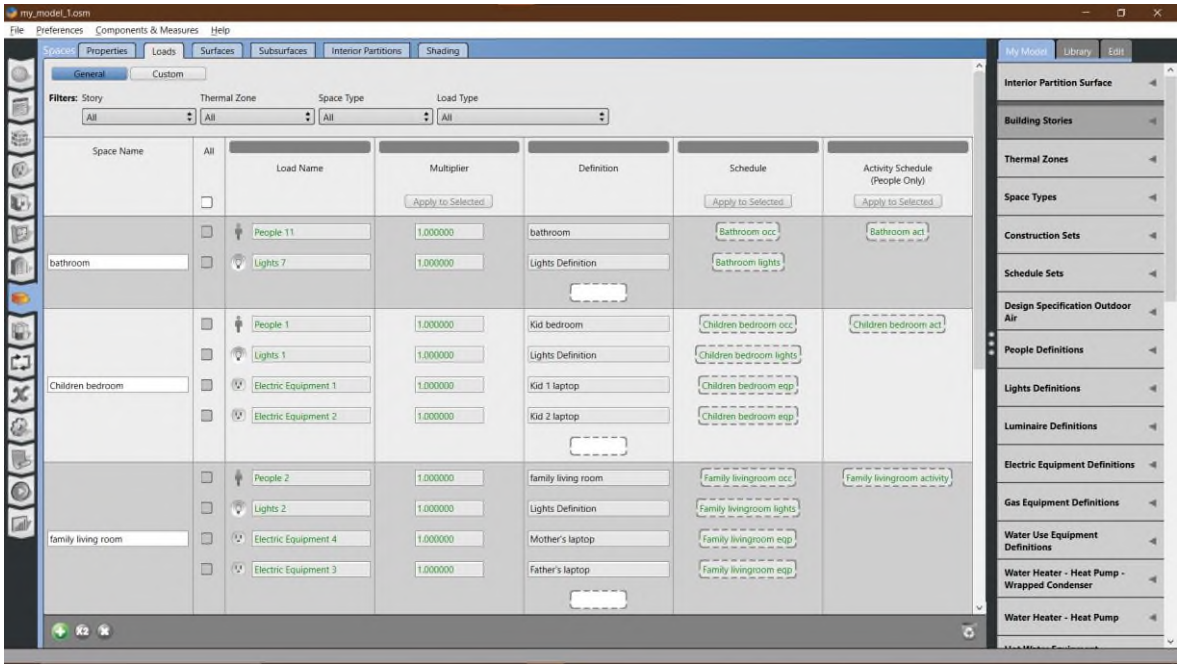

Figure 40(a): Loads subtab of the Spaces tab of the Detailed model.

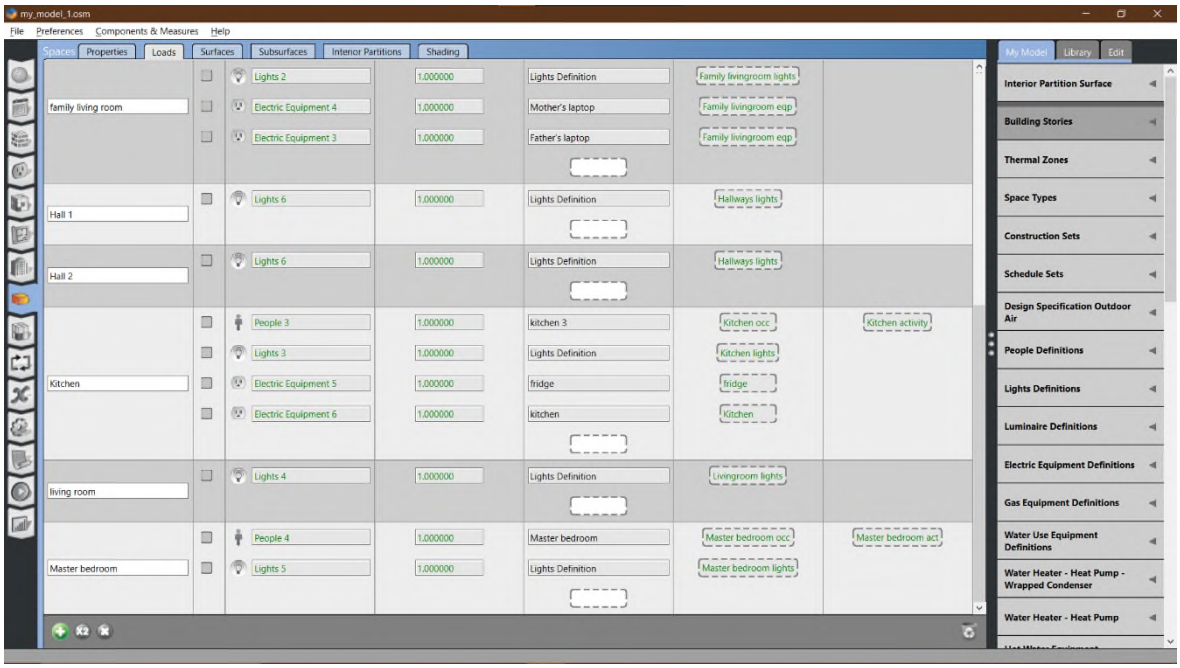

Figure 40(b): Loads subtab of the Spaces tab of the Detailed model.

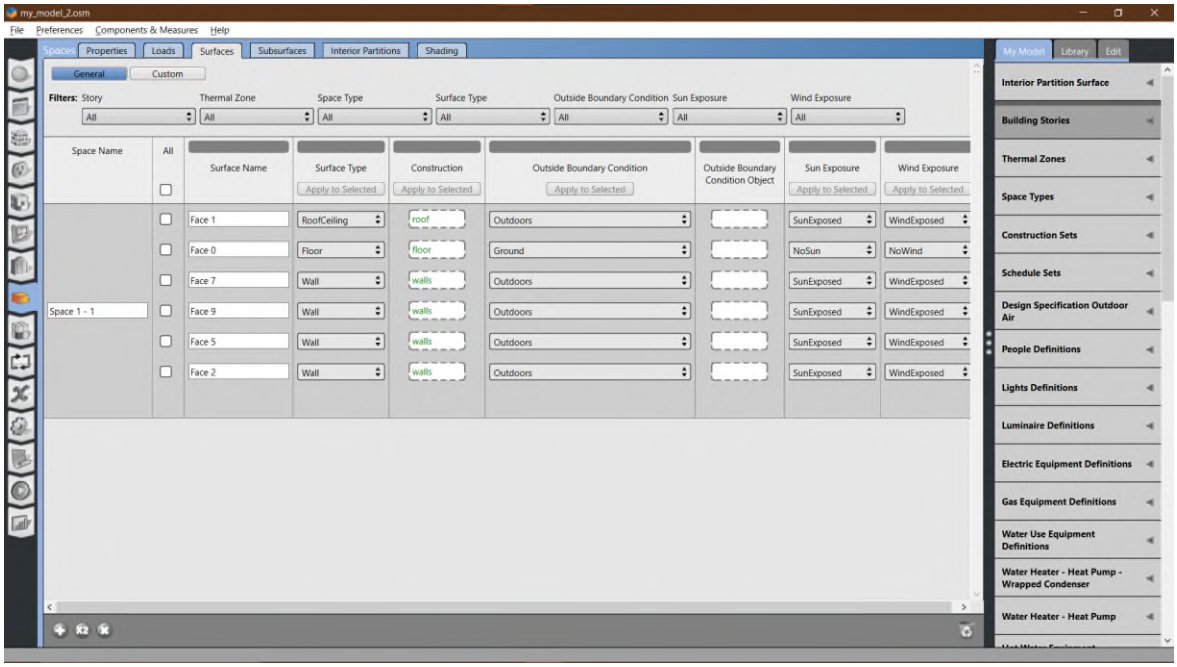

Figure 41: Surfaces subtab of the Spaces tab of the Simple model.

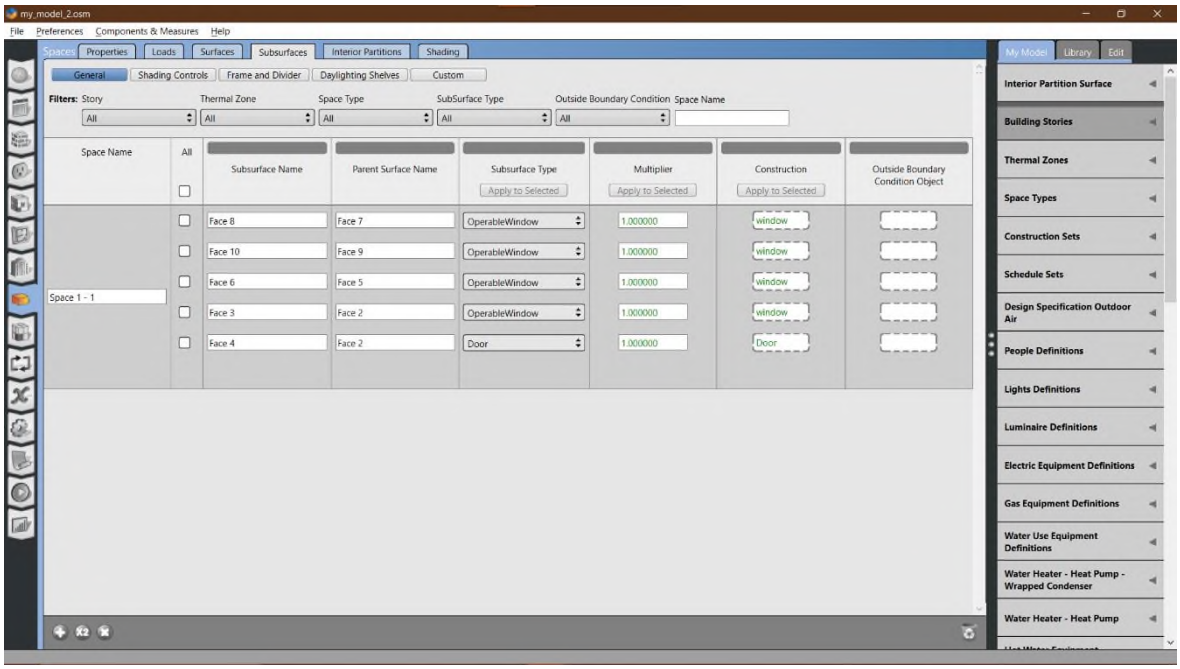

Figure 42: Subsurfaces subtab of the Spaces tab of the Simple model.

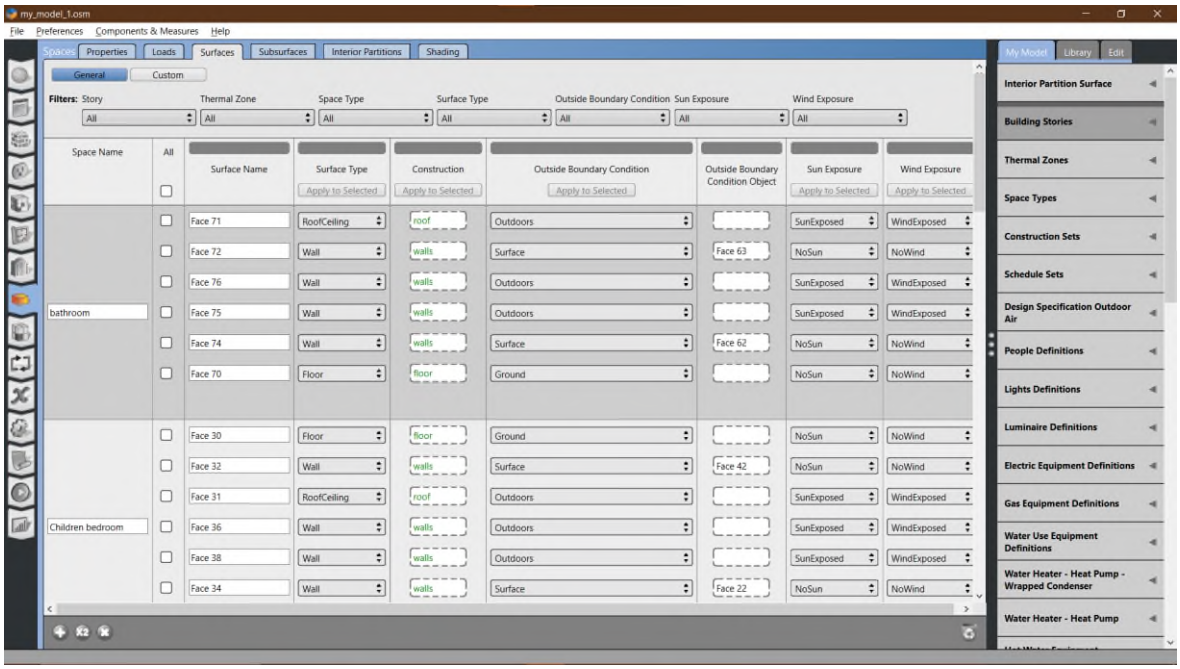

Figure 43: Surfaces subtab of the Spaces tab of the Detailed model.

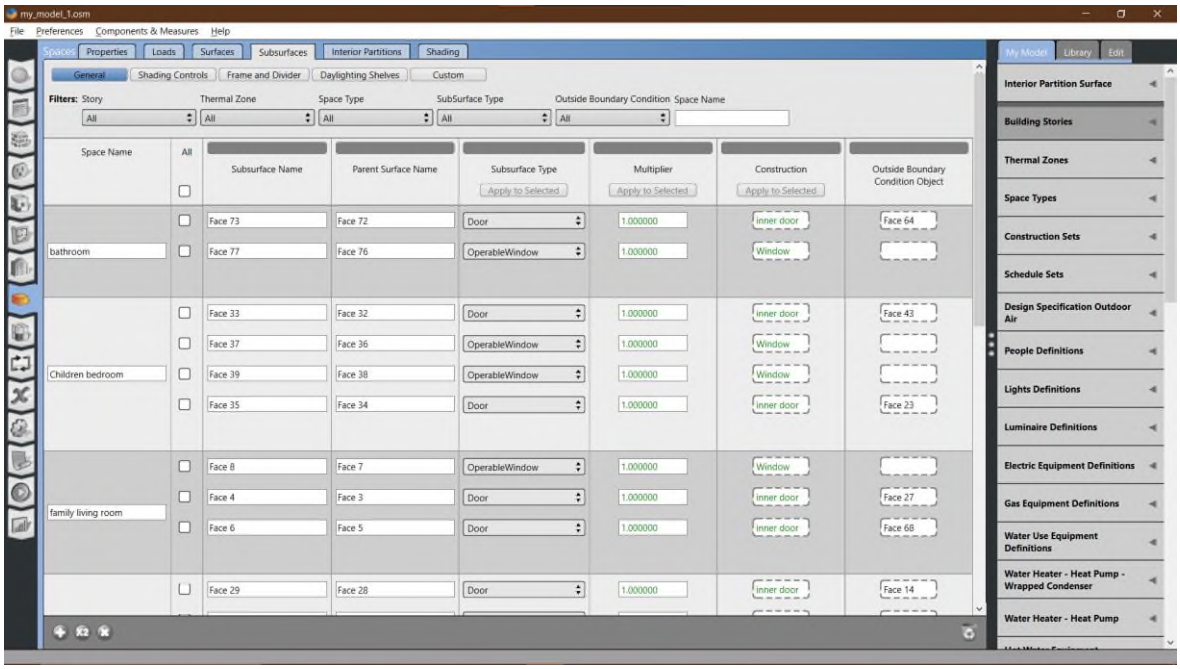

Figure 44: Subsurfaces subtab of the Spaces tab of the Detailed model.

2.3.7 Thermal Zones and HVAC Systems tabs

We explain the HVAC Systems tab next. Here we create, inspect and edit plant loops, service water, refrigeration, and Variable Refrigerant Flow (VRF) systems. For our Simple model, we create one split Air Conditioner and for the Detailed two. We start by selecting to add an empty air loop (Figure 45). Then we drag and drop an Outdoor Air (OA) System, to the supply side of the loop, from the AirLoopHVAC Outdoor Air System library provided by OpenStudio (Figure 46). In the same manner, we drag and drop the Unitary - Single Speed DX cooling - Cycling - Electric reheat component to the supply side, from the AirLoopHVAC Unitary System library (Figure 47). We click on the Air Loop Zone Splitter and select the thermal zone we want our AC to be in, for the Simple model there is only one zone, for the Detailed model, one AC is in the bedrooms thermal zone and one in the kitchen and family living room thermal zone (Figure 48). Then we drag and drop a Diffuser, from the AirTerminal Single Duct Constant Volume No Reheat library, in the loop (Figure 49). Lastly, we drag and drop a Setpoint Manager Single Zone Cooling, from the library to the supply side to complete our air conditioner model (Figure 50).

Continuing with the Thermal Zones tab, it is divided into four subtabs HVAC Systems, Cooling Sizing Parameters, Heating Sizing Parameters, and Custom. We only engage with the HVAC Systems subtab, which is the most important for us. We see the name of our thermal zone and in the Air Loop Name the name of the air loop we created for our AC and the diffuser in the zone equipment field. Here, we add Natural Ventilation to the Zone Equipment from the Zone Ventilation Design Flow Rate library. Finally, we insert the cooling and heating schedules we created to the Cooling Thermostat Schedule and Heating Thermostat Schedule fields respectively (Figure 51, Figure 52).

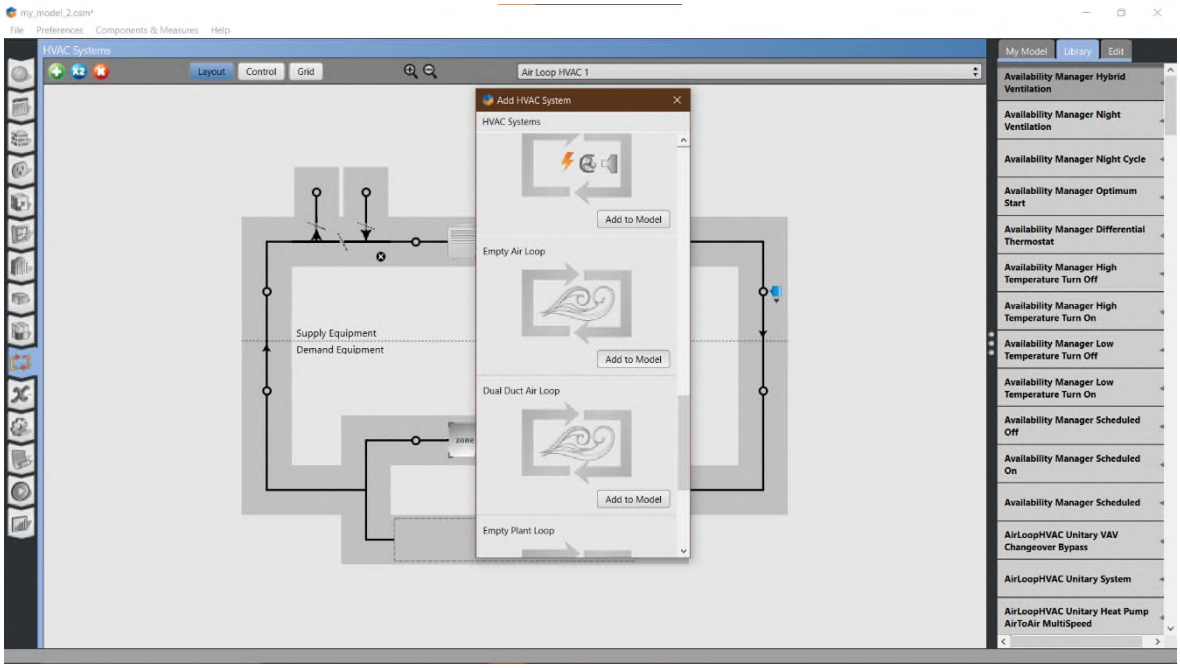

Figure 45: Selecting an Empty Air Loop as the template for the AC.

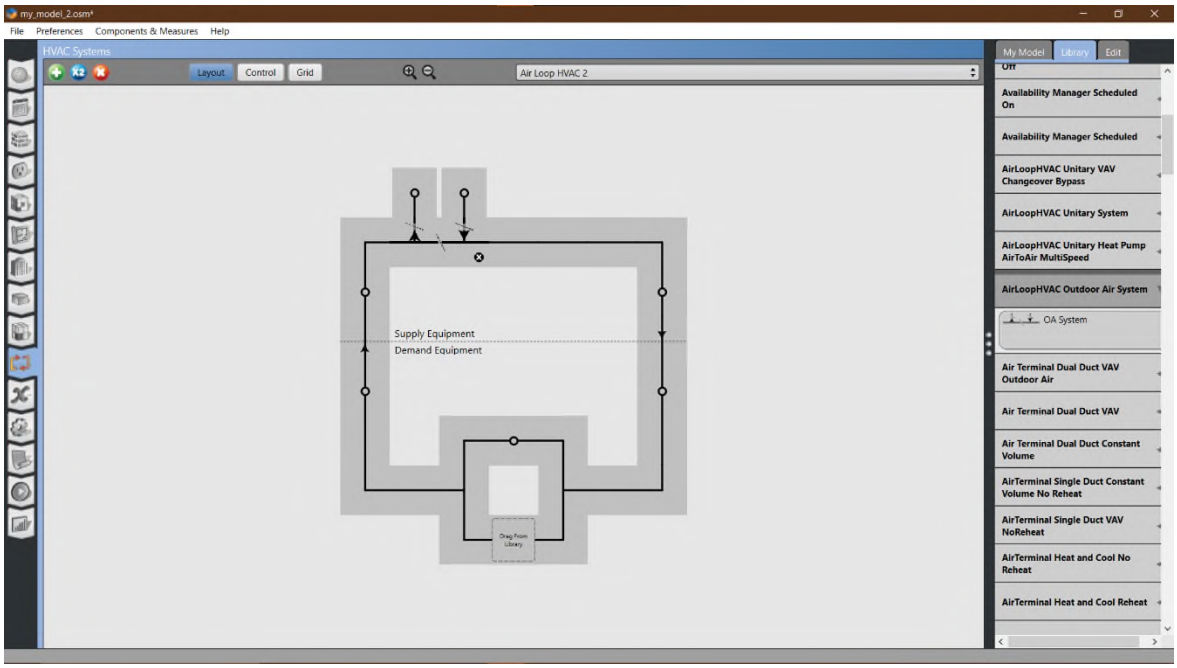

Figure 46: Setting an Outdoor Air (OA) System to the supply side.

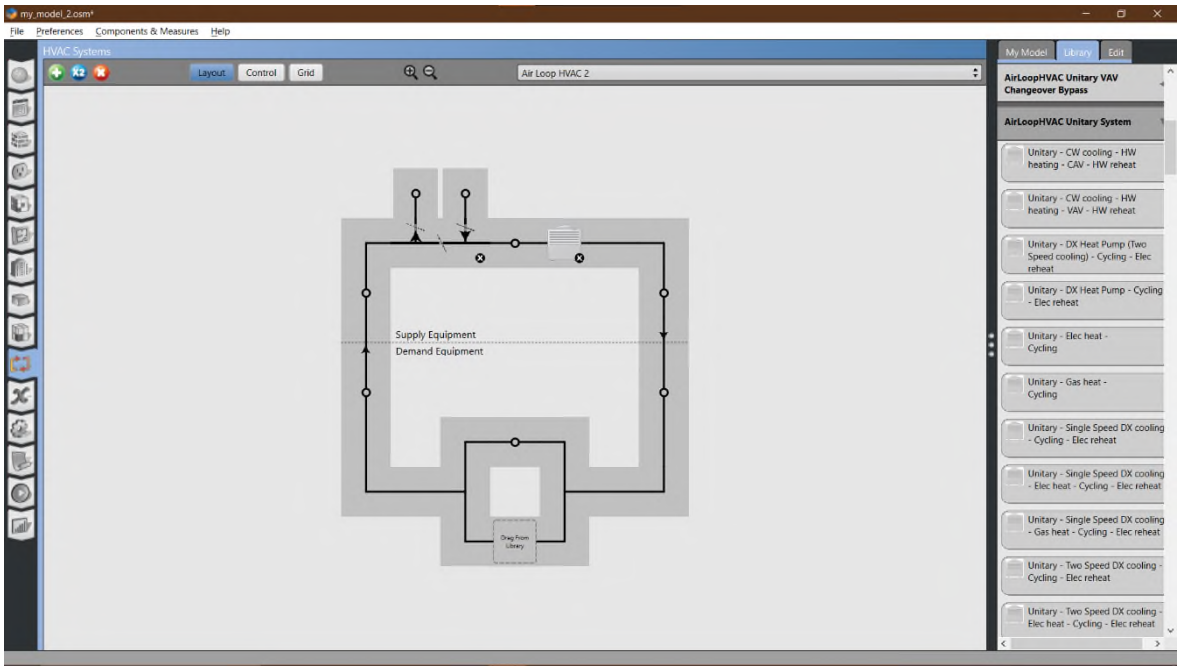

Figure 47: Setting an Unitary - Single Speed DX cooling - Cycling - Electric reheat component to the supply side.

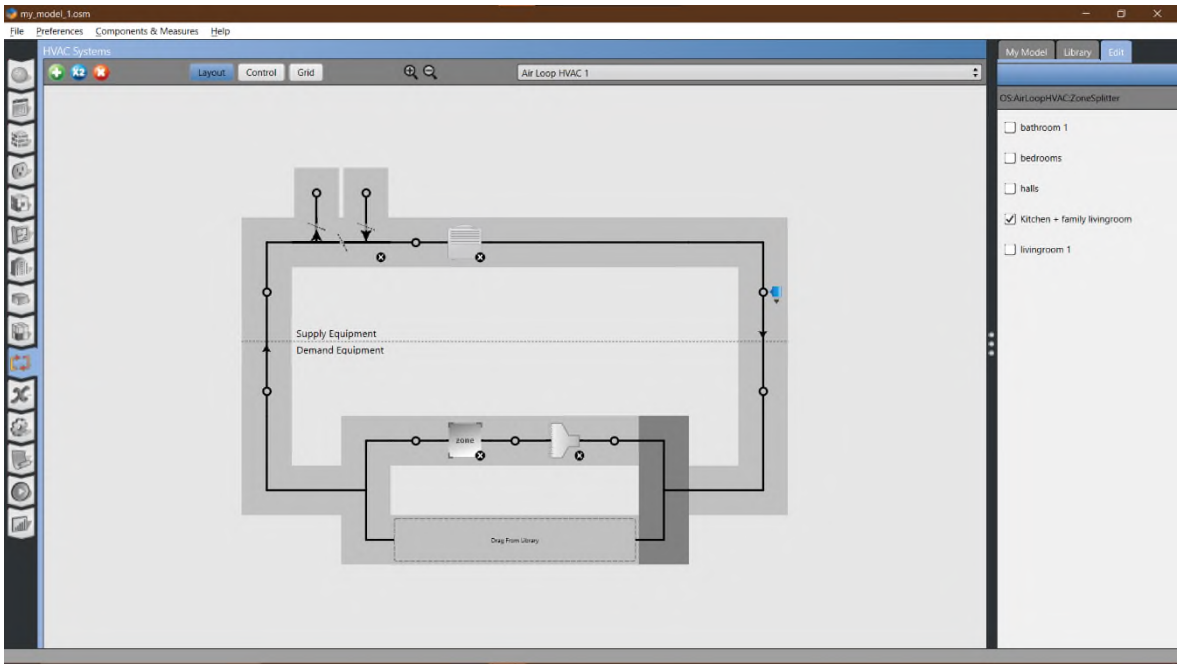

Figure 48: Selecting the thermal zone that the AC unit is going to be in.

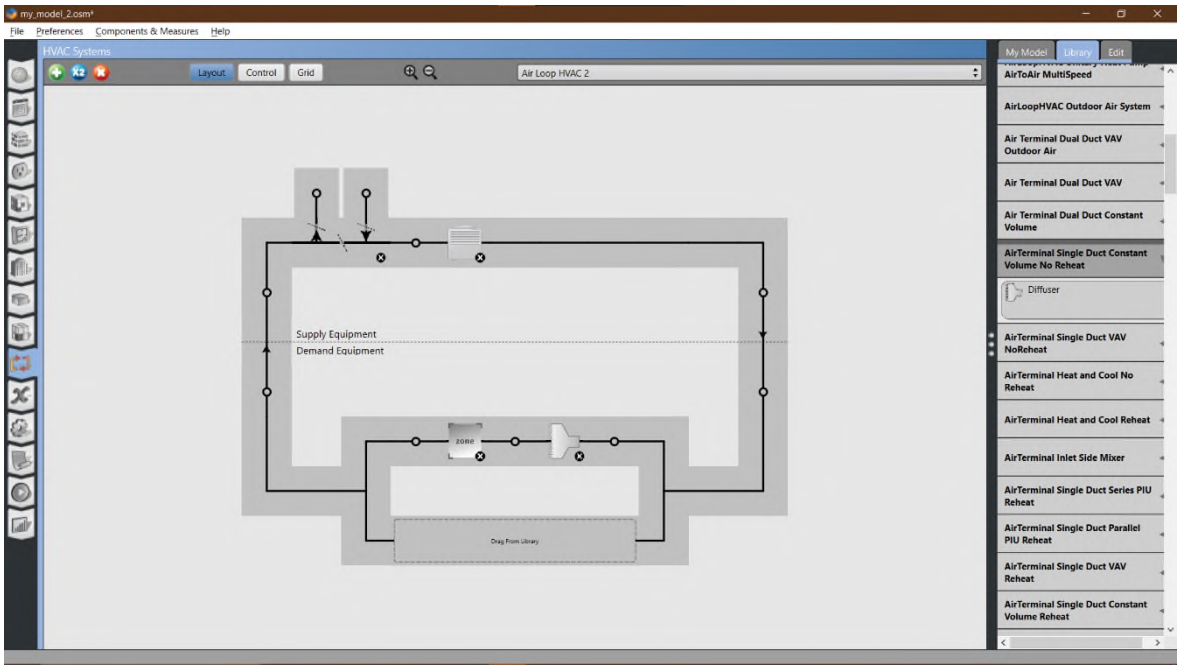

Figure 49: Placing a Diffuser in the air loop.

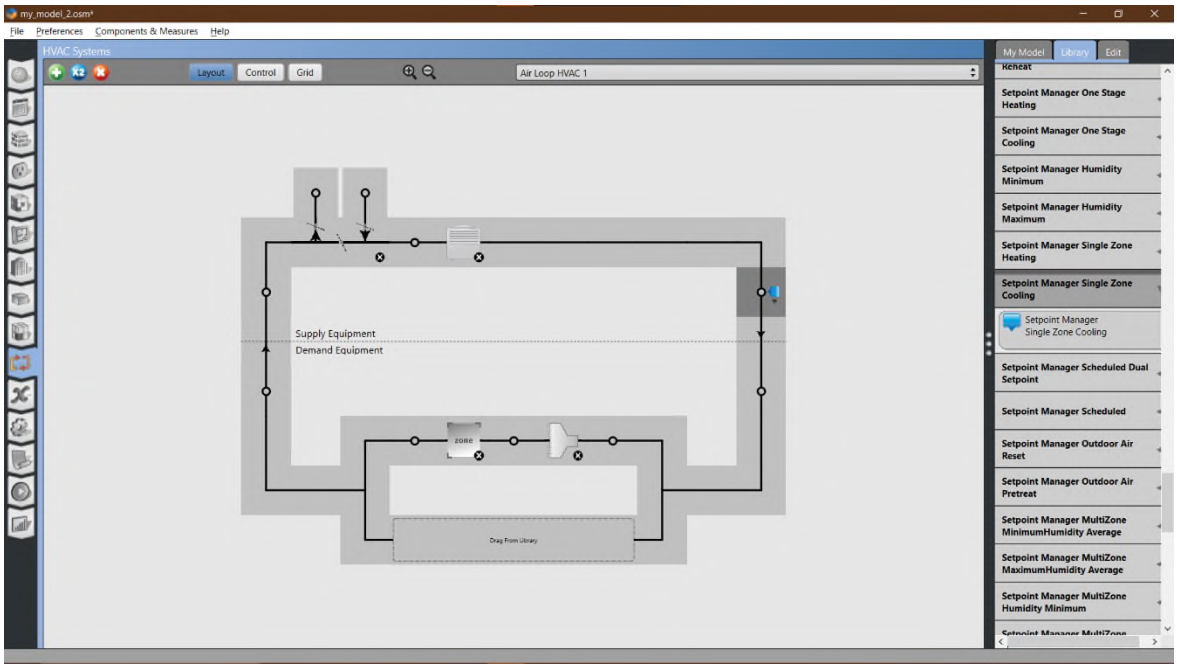

Figure 50: Placing a Setpoint Manager Single Zone Cooling in the air loop.

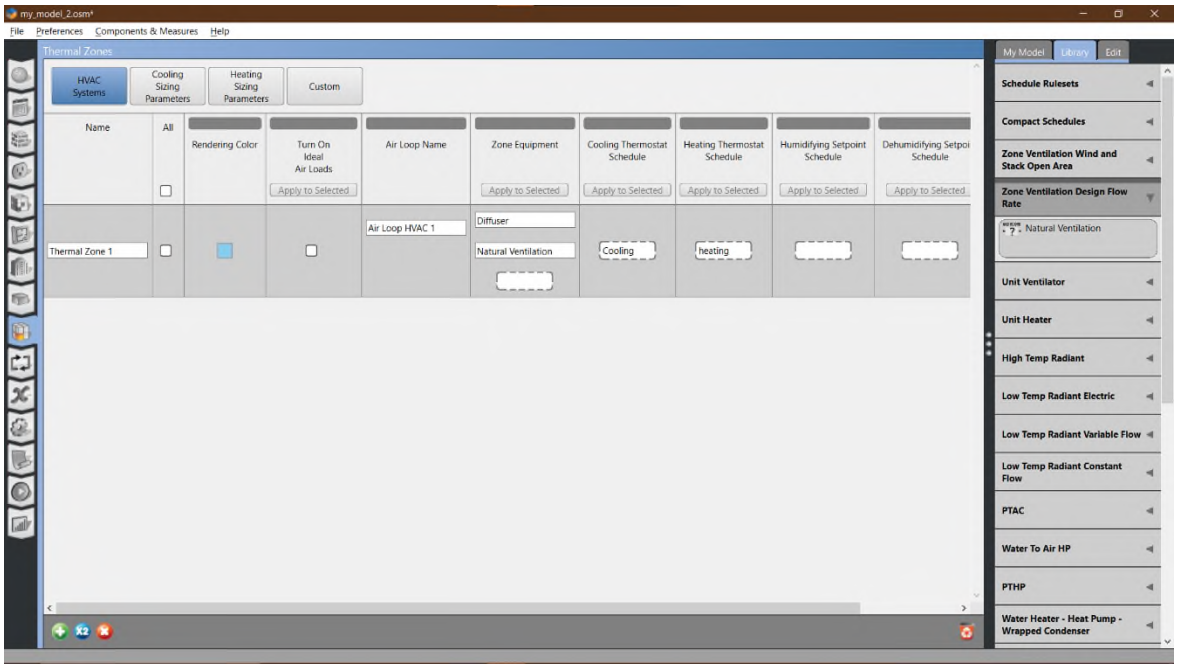

Figure 51: Complete thermal zone for the Simple model.

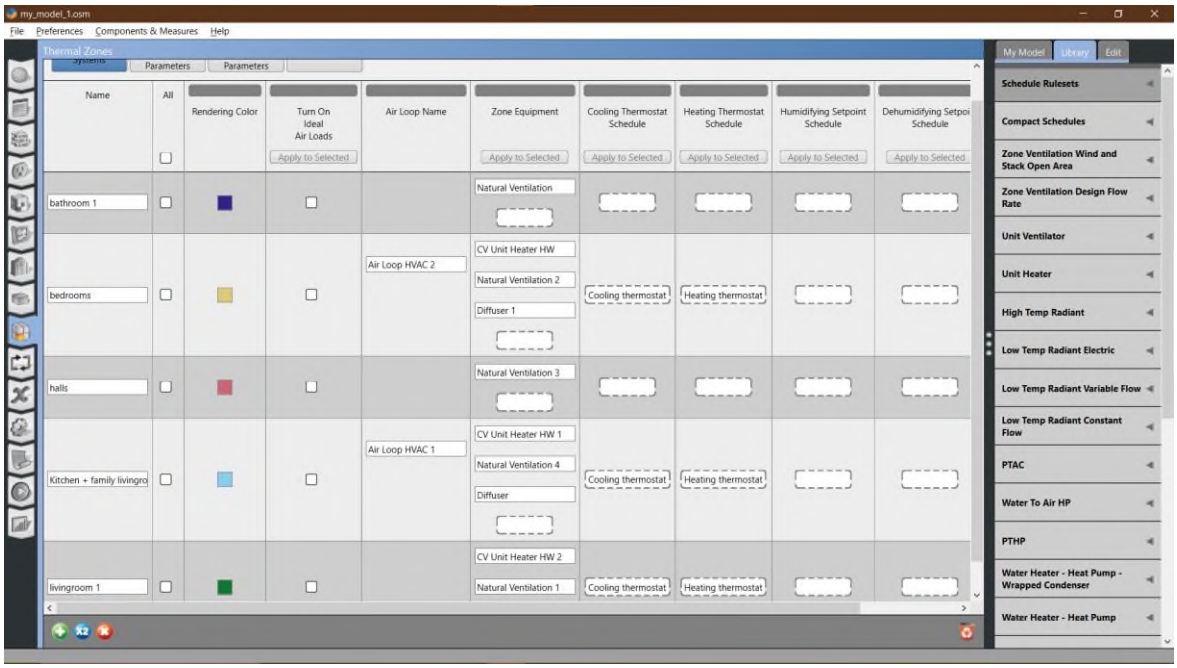

Figure 52: Complete thermal zones for the Detailed model.

2.3.8 Output Variables, Measures, Run Simulation and Results Summary tabs

Here we discuss the last four tabs in the same section since they are very small, starting with the Output Variables tab. In this tab we can turn on variables that can give us more detailed results when we run a simulation. For our models we choose the variables: People Occupant Count, Surface Inside Face Temperature, Surface Outside Face Temperature, Zone Air Temperature and Zone Outdoor Air Drybulb Temperature. We set the detail level of the variables from their dropdown menus to Timestep.

In the Measures tab, we can specify OpenStudio, EnergyPlus and Reporting Measures that are applied to the model only during the simulation. This way we can test different situations with parameters on our model without making fundamental changes to the model. It is important to note that the Measures are always applied each time we run a simulation. A very helpful Measure that we should always run during a simulation is the “OpenStudio Results”, this Measure compiles the results of the simulation in a more comprehensive report than the standard EnergyPlus report. To do this, we must drag and drop the Measure from the Reporting->QAQC library to the Reporting Measures field (Figure 53).

To run a simulation, we simply go to the Run Simulation tab and click on the play button at the top left of the window (Figure 54). If everything goes well and we haven’t

made a mistake along the way, the simulation runs and concludes with messages shown in Figure 55.

To study the results of the simulation, we simply go to the Results Summary tab and thanks to the “OpenStudio Results” Measure, we are greeted with a comprehensive report with pie charts and diagrams for our simulation (Figure 56 (a), (b), (c), (d), (e)).

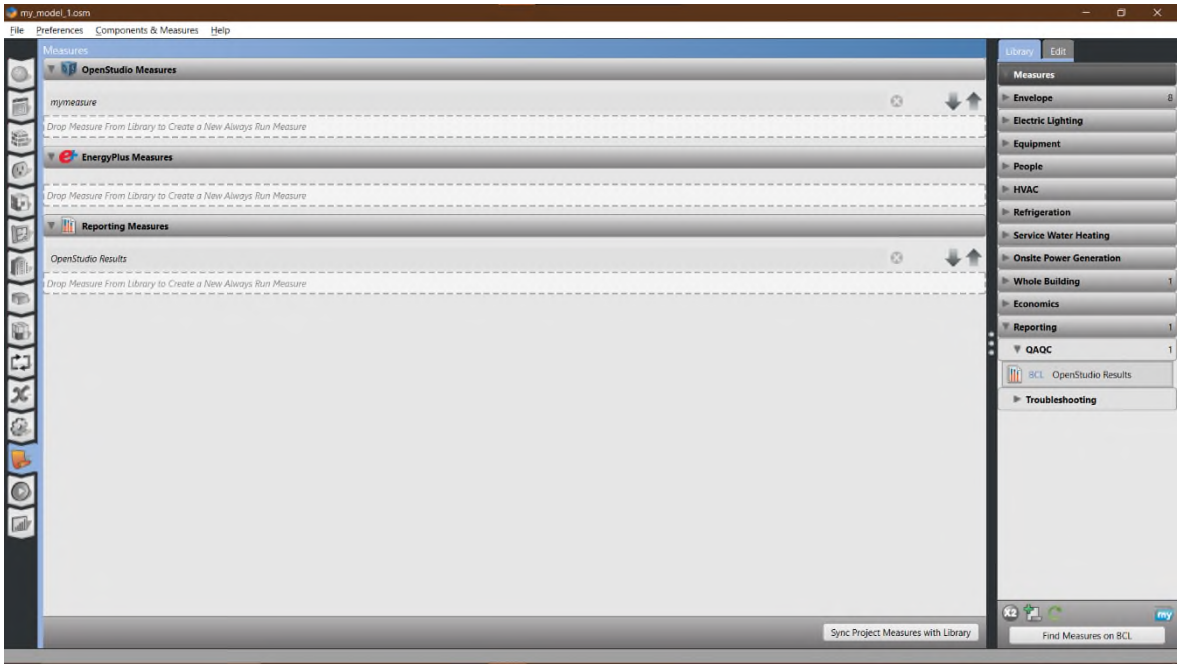

Figure 53: Measures tab.

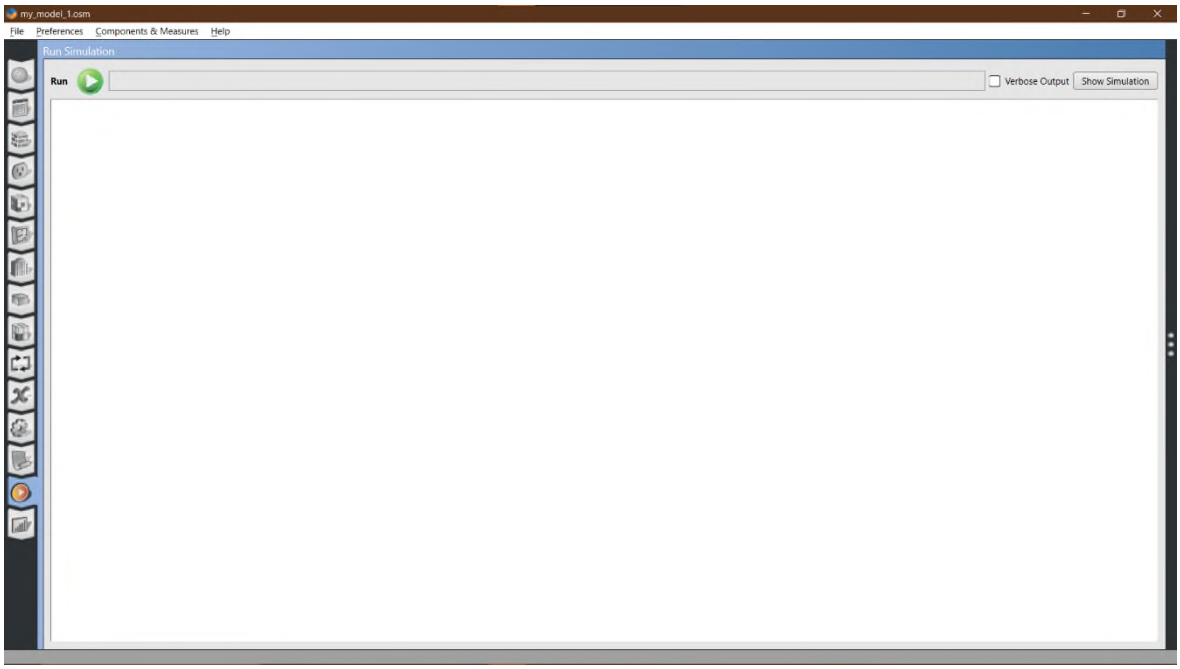

Figure 54: Run Simulation tab.

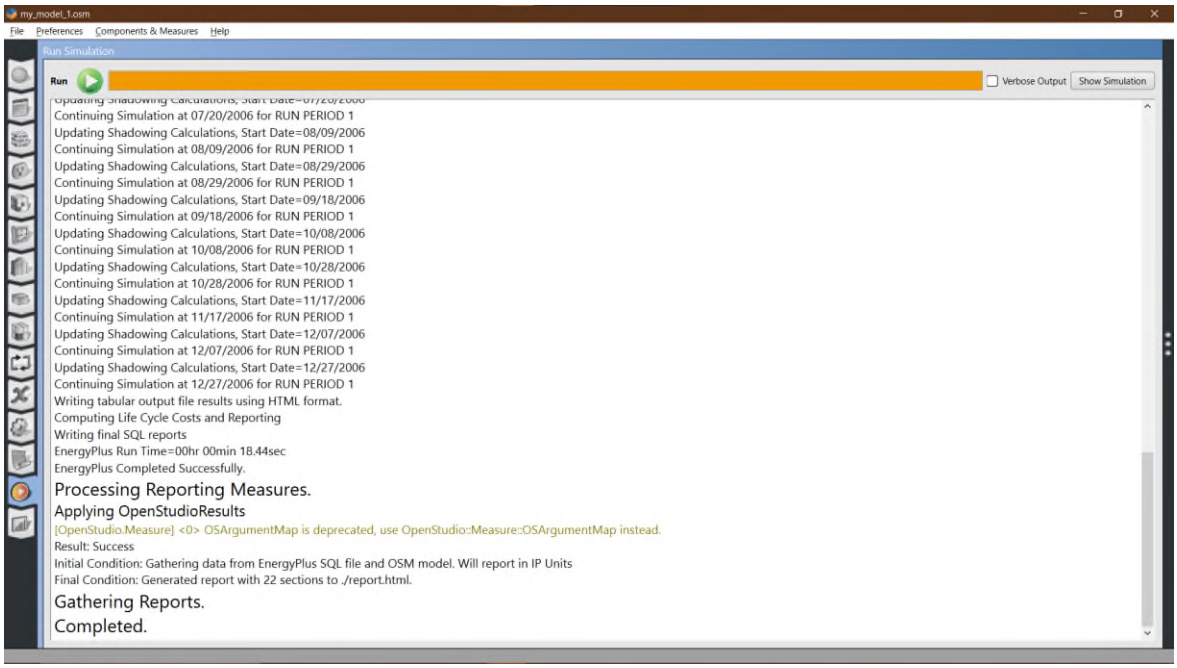

Figure 55: Result messages from a completed simulation run.

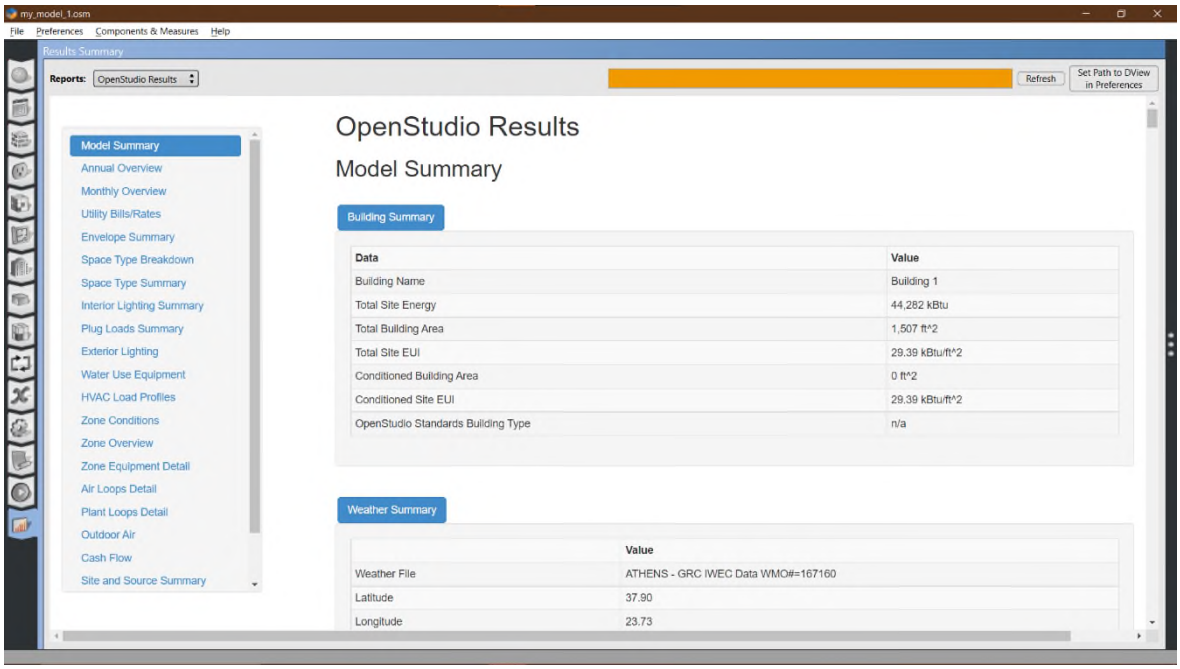

Figure 56(a): A part from the report of the simulation results for the Detailed model.

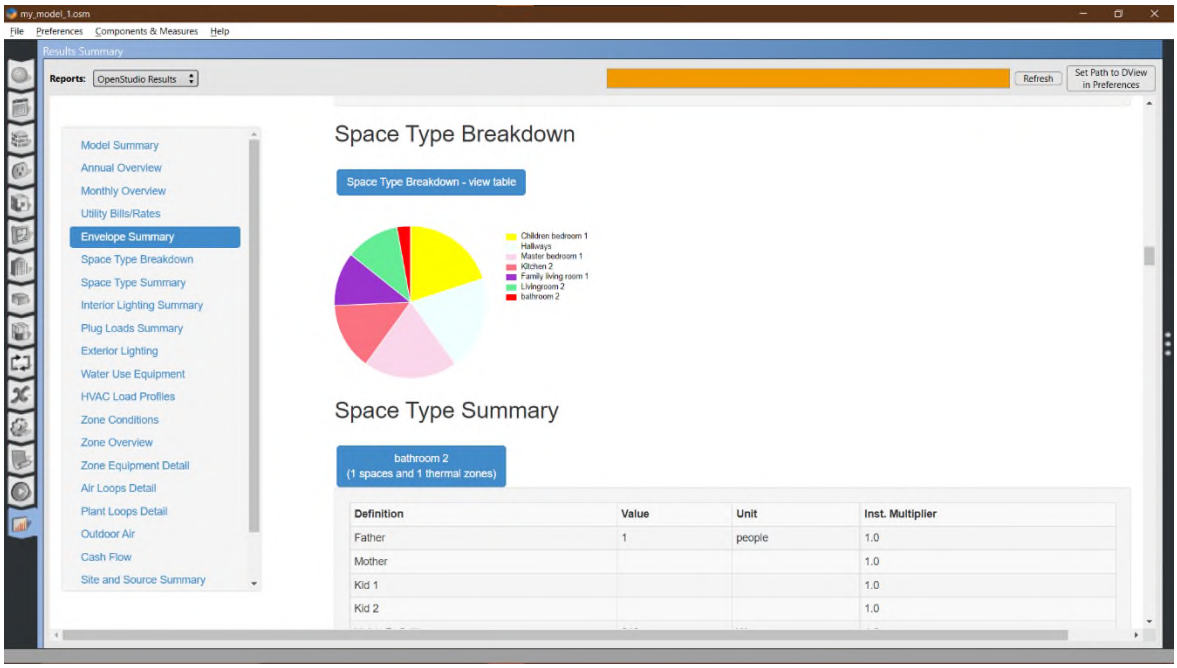

Figure 56(b): A part from the report of the simulation results for the Detailed model.

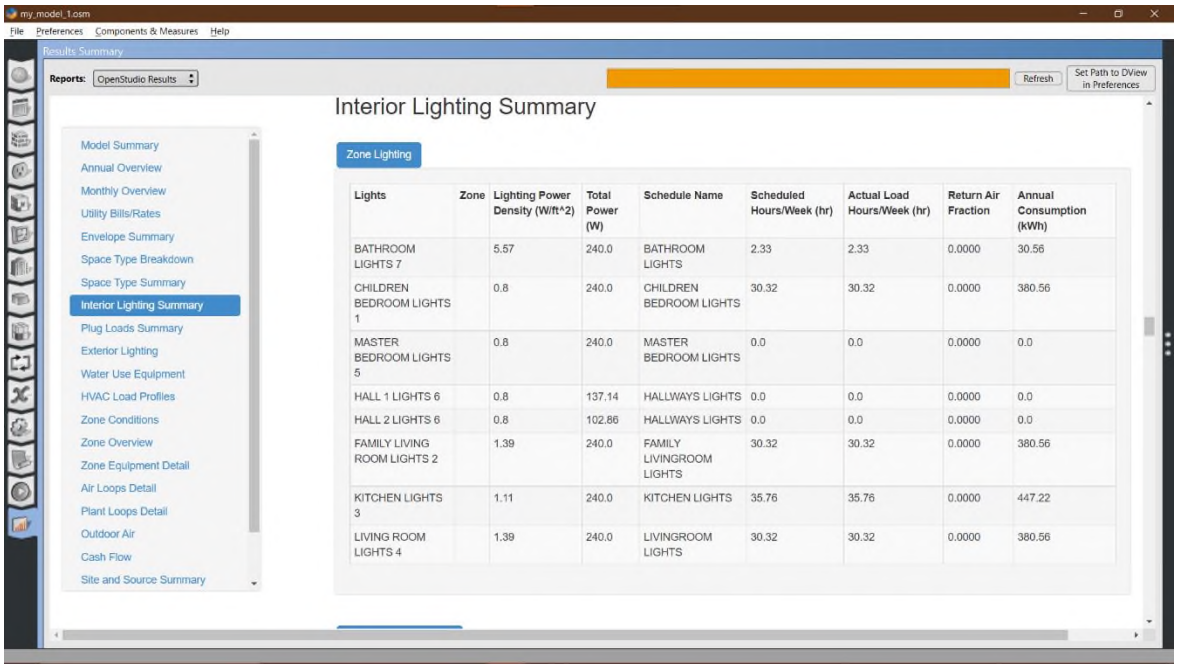

Figure 56(c): A part from the report of the simulation results for the Detailed model.

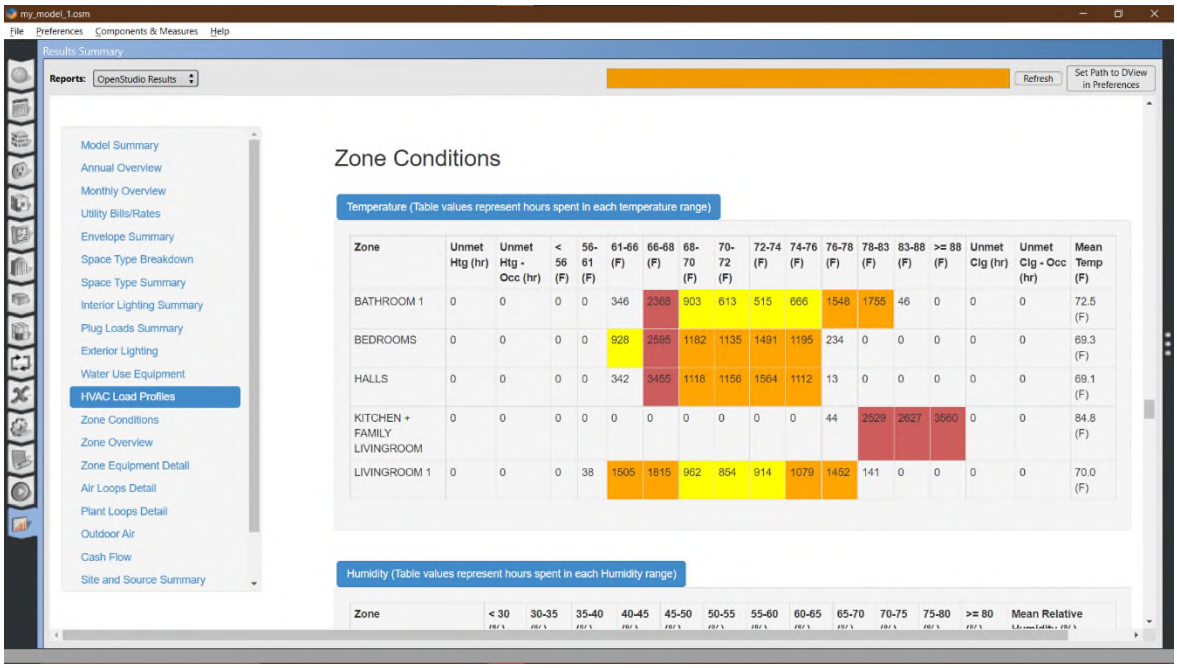

Figure 56(d): A part from the report of the simulation results for the Detailed model.

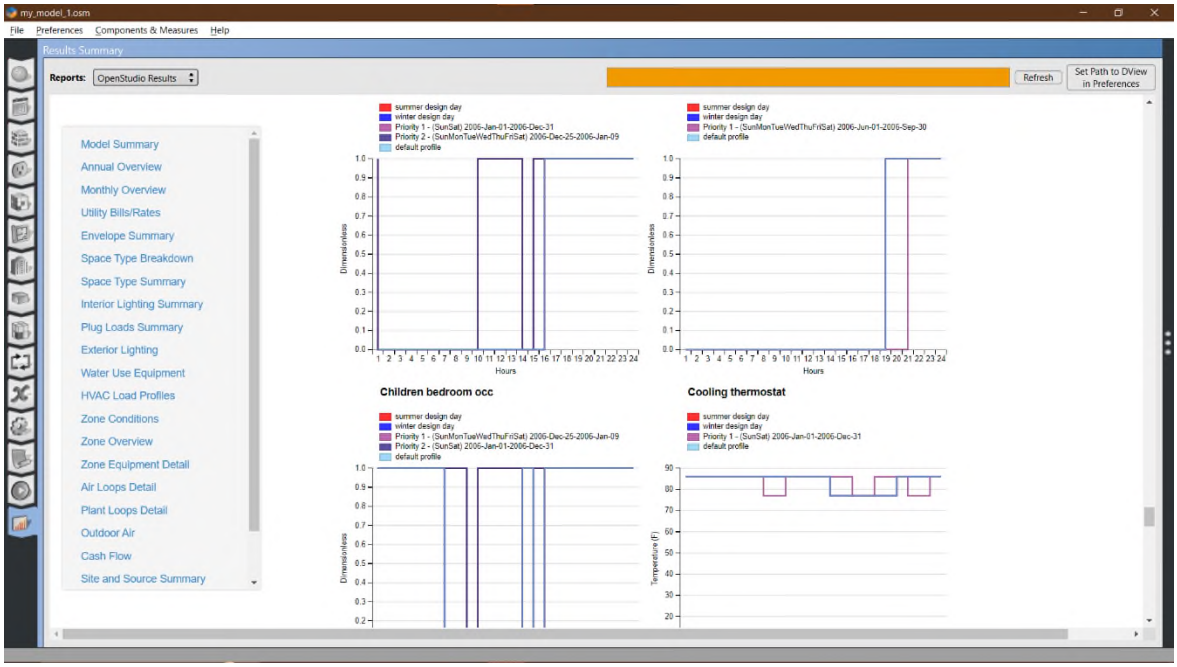

Figure 56(e): A part from the report of the simulation results for the Detailed model.

CHAPTER 3 WRITING THE MEASURE ENABLING THE FUNCTIONALITY OF OUR SIMULATION TOOL

As we previously discussed, we can apply Measures to our models to make permanent changes or add them to the workflow of a simulation. A distinguishing quality of OpenStudio and essential to the platform's extensibility are these Ruby scripts. Hence their name, Measures are most frequently employed to carry out model modifications that map to Energy Efficiency (EE) measurements [23]. They may, however, be utilized in several ways to query and alter OpenStudio models and related data [23]. OpenStudio Measures may access elements in the Object Model of OpenStudio and are created in the fully complete programming language Ruby [23]. OpenStudio Measures are not restricted to simulating the application of EE technologies to buildings because Ruby is such a powerful programming language [23]. They may also work with an empty model, creating models from scratch using Ruby code and the OpenStudio Standards Gem [23]. The "OpenStudio Results" Measure is an example of an OpenStudio Measure that focuses on querying the energy Model's related data to generate specialized reports or data exports [23]. We can search for specific Measures at the BCL website. There are many Measures stored in the libraries that satisfy different tasks, that are also immediately accessible, free for use and implementation to other user generated Measures. This ensures that if we require a Measure for a certain function, we can simply download it and integrate it in our own Measure code. It is a good practice to credit the source of the code in our work.

The presented work provides the user with three main functionalities. First, the user can change the dimensions of the model on the three axes but leave the Sub-Surfaces such as windows and doors untouched. Second, the user can place windows to any side of the building. Third, control of the R-value of the walls’ and roofs’ insulation. We discuss the capabilities of each Measure in their specific sections.

3.1 Examples of Measure application and input acquirement

First, we explain how we use a Measure within OpenStusio and show how the Ruby code of a Measure looks like. For that purpose, we show how we gather the inputs and how we pass them to a function.

3.1.1 Applying a Measure

To apply a Measure, we can be on any tab we want except from the Geometry tab. We also must go to the Preferences on the toolbar and select the option “Change My Measures Directory” and specify the directory where we have saved our Measure (Figure 57). Next, we go to Components & Measures, select the apply Measure option. Then we go to the category where the desired Measure belongs. For example, the Measure we create belongs to the category “Envelope”, then “Form”. We select the Measure, change or fill in any input we want and finally click apply Measure (Figure 58). Depending on the Measure, we can be greeted by a window that displays any results or messages from the Measure (Figure 59). Here we can accept the changes, go back to enter different inputs or discard the application of the Measure altogether. Selecting the option “accept changes” applies any changes done by the Measure to the model directly and saving them, makes them permanent.

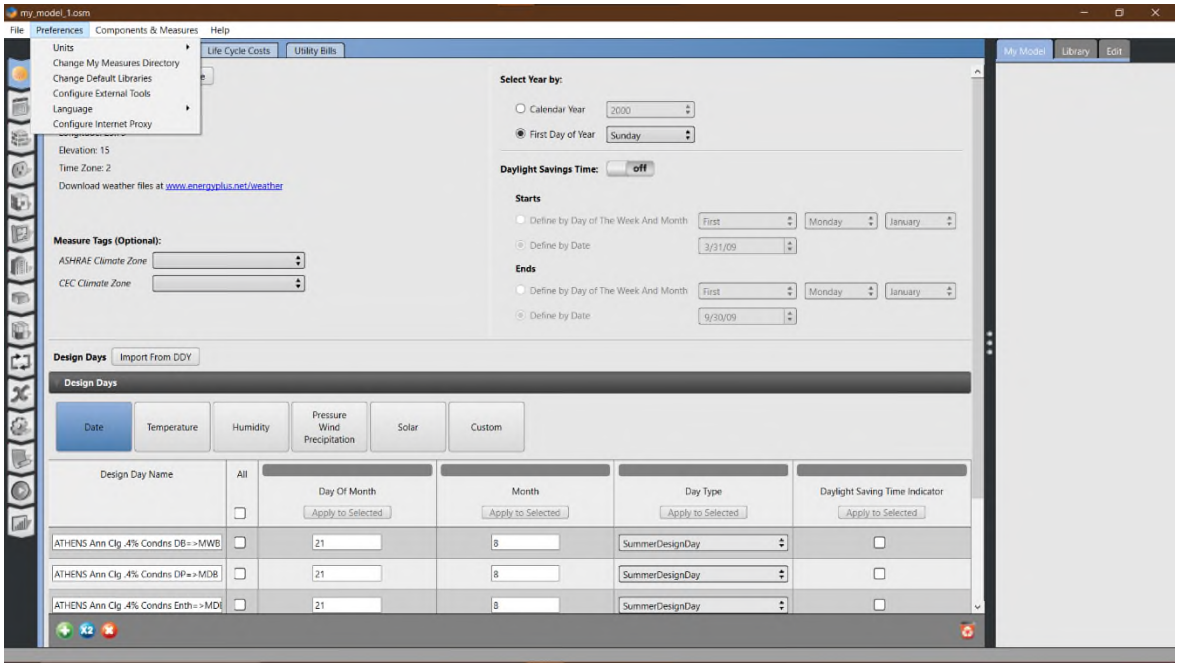

Figure 57: Specifying the directory where our Measure is saved at.

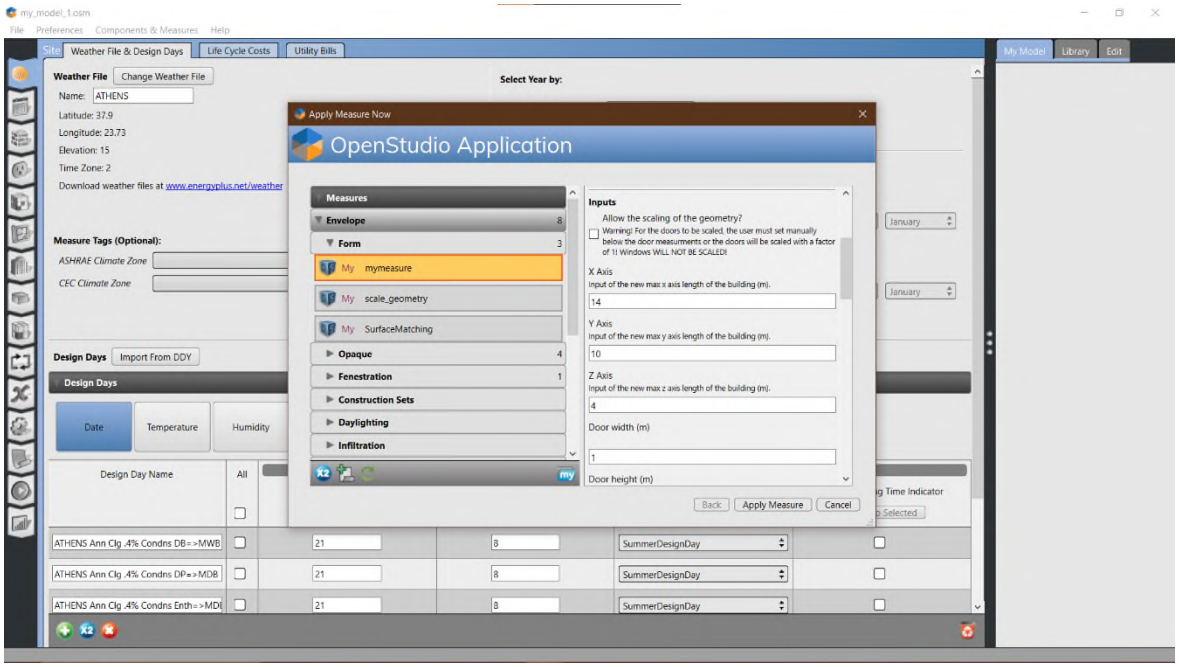

Figure 58: Applying our Measure.

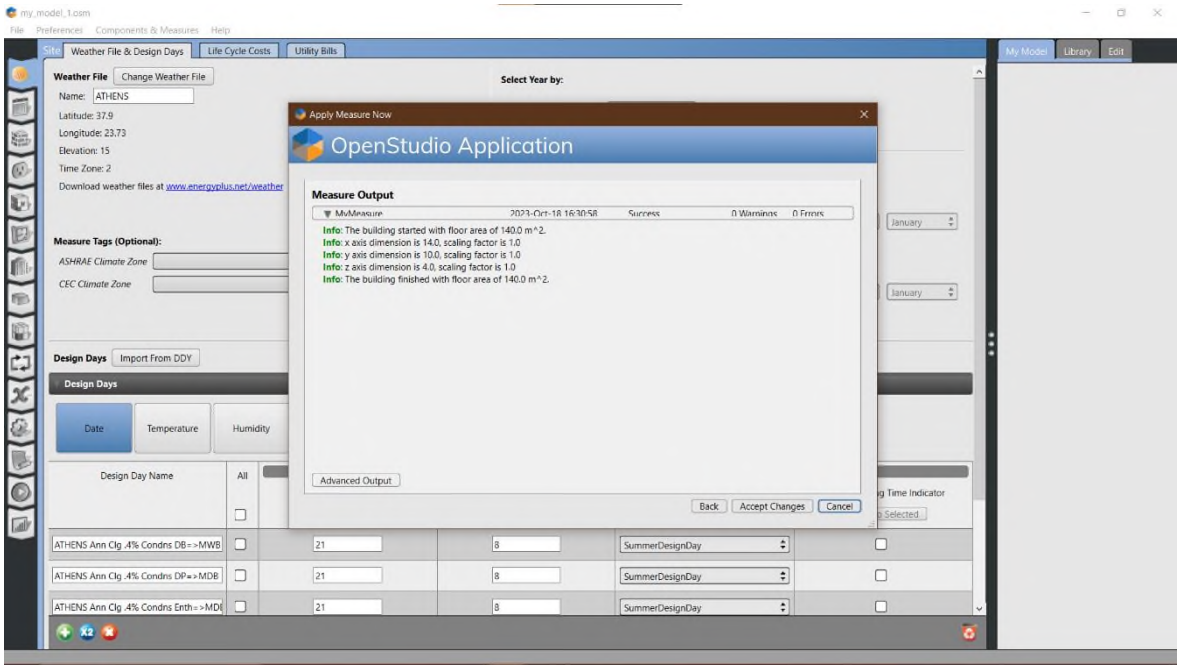

Figure 59: Measure results window.

3.1.2 Input acquirement in a Measure

Figure 60 presents an example of how we create the user interface of a Measure. First, we define the user arguments section and set an argument variable that passes the inputs to the main part of the code. Then we establish set a variable for every argument, with a name, description and default value, if necessary. Lastly, in the main part of the code,

using the argument variable and the name of the input, we assign each input to a variable (Figure 61).

```
def arguments(model)
  args = OpenStudio::Measure::OSArgumentVector.new

  ########################################################################################################################################################
  #Input for scaling geometry
  allow_scale = OpenStudio::Measure::OSArgument.makeBoolArgument('allow_scale', true)
  allow_scale.setDisplayName('Allow the scaling of the geometry?')
  allow_scale.setDescription('Warning! For the doors to be scaled, the user must set manually below the door measurments or the doors will be scaled with a factor of 1
  allow_scale.setDefaultValue(false)
  args << allow_scale

  # x scale
  x_axis = OpenStudio::Measure::OSArgument.makeDoubleArgument('x_axis', true)
  x_axis.setDisplayName('X Axis')
  x_axis.setDescription('Input of the new max x axis length of the building (m).')
  x_axis.setDefaultValue(14.0)
  args << x_axis

  # y scale
  y_axis = OpenStudio::Measure::OSArgument.makeDoubleArgument('y_axis', true)
  y_axis.setDisplayName('Y Axis')
  y_axis.setDescription('Input of the new max y axis length of the building (m).')
  y_axis.setDefaultValue(10.0)
  args << y_axis

  # z scale
  z_axis = OpenStudio::Measure::OSArgument.makeDoubleArgument('z_axis', true)
  z_axis.setDisplayName('Z Axis')
  z_axis.setDescription('Input of the new max z axis length of the building (m).')
  z_axis.setDefaultValue(4.0)
  args << z_axis

  door_width = OpenStudio::Measure::OSArgument.makeDoubleArgument('door_width', true)
  door_width.setDisplayName('Door width (m)')
  door_width.setDefaultValue(1.0)
  args << door_width

  door_height = OpenStudio::Measure::OSArgument.makeDoubleArgument('door_height', true)
  door_height.setDisplayName('Door height (m)')
```

Figure 60: Establishing user argument variables.

```
def run(model, runner, user_arguments)
  super(model, runner, user_arguments)

  # use the built-in error checking
  if !runner.validateUserArguments(arguments(model), user_arguments)
    return false
  end

  # assign the user inputs to variables
  ########################################################################################
  #First for geometry scaling
  allow_scale = runner.getBoolArgumentValue('allow_scale', user_arguments)
  x_axis = runner.getDoubleArgumentValue('x_axis', user_arguments)
  y_axis = runner.getDoubleArgumentValue('y_axis', user_arguments)
  z_axis = runner.getDoubleArgumentValue('z_axis', user_arguments)
  door_width = runner.getDoubleArgumentValue('door_width', user_arguments)
  door_height = runner.getDoubleArgumentValue('door_height', user_arguments)
```

Figure 61: Passing the user arguments in the main part of the code.

3.2 The developed model Measure

We employed four existing Measures from [25][26] as templates for our work. The developed Measure is coded in four Ruby functions, which are described in the following paragraphs.

3.2.1 Geometry scaling

We use the "scale_geometry" Measure from [26] as template for the scaling geometry functionality. First, the choices the user can make are established. The user can choose to activate the geometry scaling to input new dimensions for the building (e.g., changing the height of the building) or to modify the width and height of the doors. The windows of the model are not scaled when changing the size of the model or the doors. The relative position of the windows and doors from the center of the wall they are installed on stays the same after the scaling. If the user activates scaling but leaves dimensions untouched, then there will be no changes to the building.

Scaling geometry uses six inputs, one for the activation, three for the x, y and z axes for the dimensions of the building and two for the width and height of the door. The first variable is a Boolean type with the default value as false, the three dimension-variables are of type Double with default values the measurements of the building, the two variables for the door are also of type Double with default values the measurements of the door.

After passing the arguments in the main part of the code, we calculate the scaling factors for each dimension of the model and the dimensions of the door if it exists. We calculate the scaling factors by dividing the input value with the default value, for each one of the dimensions. To maintain the relative position of windows and doors on the walls, first we find the centroids of the Sub-Surfaces, then the centroid of the wall Surface they rely on. Lastly, we find the distance between the Sub-Surface centroid and the Surface centroid, for each Sub-Surface and its parent Surface. Using mathematical operations with the vectors that describe the corners of a Surface and a Sub-Surface, we scale the walls and doors. Finally, using the centroid distance, we place the Sub-Surfaces at the scaled wall Surfaces.

In Figure 62, we present the resulting model when we apply the Measure, enable the "Geometry scaling" option and change the x-axis dimension from 14 to 50, which is a scaling factor of 50/14=3.571 for the x-axis.

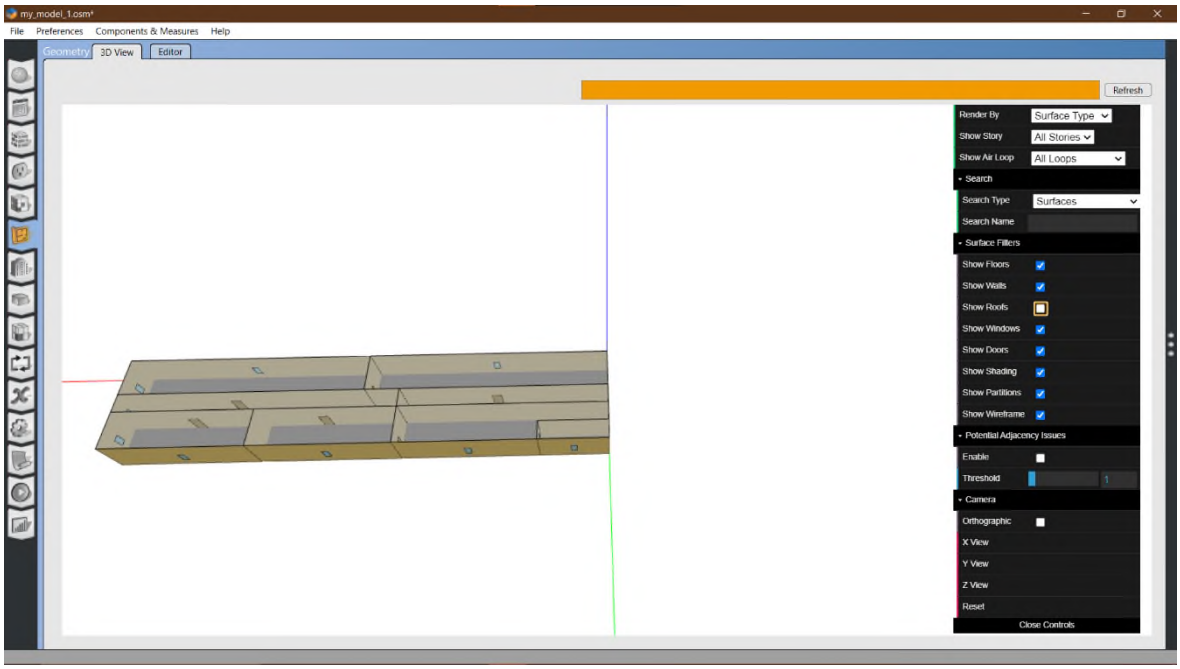

Figure 62: Resulting building model after applying the “Geometry scaling” option.

3.2.2 Window manipulation

For this part, the Measure “Set Window to Wall Ratio by Facade” [25] serves as a template. Like in the previous function, we establish the choices the user can make. The user can choose to activate the function of this part of the Measure. The user can choose to install a ribbon window or a set of windows and input a combination of Window to Wall Ratio (WWR), window height and window width. He also can input the number of windows a set can have, the sill height, the cardinal direction the windows will be facing and how to handle the doors (e.g., Do nothing, Remove doors). If the user activates the function but doesn’t input new measurements, default values are applied to the function.

So, for this part, we have nine inputs, for the activation, the window logic (ribbon window or windows set), cardinal direction and door logic and as well as window specific variables i.e., WWR, window height, window width, number of windows in a set and sill height.

If the user enters a value of 0 for the window to wall ratio and chooses the “Ribbon window” option, then any existing window is removed or no new window is added. The same happens when the user chooses the “Windows set” option and enters a value of 0 for the number of windows in a set. If the user chooses the “Windows set” option but the complementary data (window height, width etc.) results in the set taking more space than

the wall Surface, no change or modification is applied on the wall. A window or a set of windows is placed if the dimensions of the respective wall Surface allow it. The “Remove doors” option removes any door type Sub-Surface from an external wall in the specified cardinal direction and place windows. Otherwise, no change happens. The “Split Walls at Doors” option splits the wall Surface that has a door type Sub-Surface in three parts, two lanes that beside the door and a middle part that has the door.

A ribbon window is a group of windows arranged side by side to create a continuous band that spans a façade horizontally. In OpenStudio, it isn’t a set of windows but a continuous window. If the user has specified a WWR of 0, then we remove all window type Sub-Surfaces from the Surface. Otherwise, we create a new window, with size determined from the WWR value, and the input sill height.

We check if a materials’ Constructions set is assigned to every new window. If not, we check if there is a window Construction material in the prototype model, otherwise we create a new Simple Glazing System Window Material programmatically. The properties of this Construction material with U-factor value of 2.556, Solar Heat Gain Coefficient of 0.764 and Visible Transmittance value of 0.812.

In the Windows set case, there are three ways to specify the windows properties. In this part of the Measure, a Ruby code found on code sharing website Github [27] served as a template and was modified to suite the needs of our tool. First, we calculate how much area the windows cover by multiplying the Surface gross area with the WWR value and dividing it with the number of windows to find the area of an individual window. Next we perform several checks depending on which additional information the user has provided, whether it is the height, width or both for the windows. If the height or width of the window has been specified, we can find the other one by dividing the gross area of an individual window with the height or width. There are two checks we perform depending on which information was specified. If the width is known, we check if the combined width of the windows is greater than the width of the surface wall. If the height is known, we check whether the sill height combined with the height can get the window over the maximum height point of the surface wall. Then, if the surface wall has a door, we calculate the door height and remove any existing window type Sub-Surface. The door height is needed if we want to check whether we can put windows above the door.

Finally, we calculate the position of the new windows on the Surface, we place the windows and assign a Construction to them.

In Figure 63, we present the resulting model when we apply the Measure, enable the “Window to wall ratio” option, choose “Windows set, specify height & width” option, set the number of windows to 3, set the width and height of the windows to 3 and 1 respectively, set the sill height to 0.5, choose the option “All” and choose the option “Do nothing to doors”. As we can see windows have been installed only on Surfaces with measurements that support the installation.

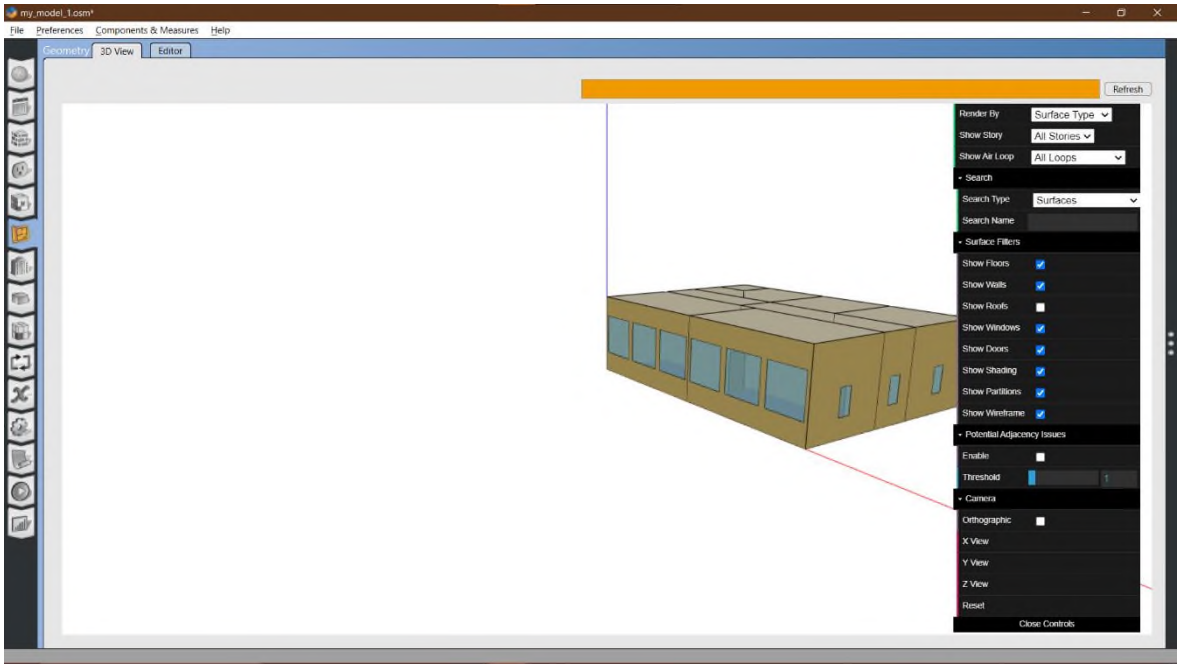

Figure 63: Resulting building model after applying the “Window to wall ratio” option.

3.2.3 Increasing the R-value of walls and roofs

For these functions, the Measures “Increase R-value of Insulation for Exterior Walls to a Specific Value” and “Increase R-value of Insulation for Roofs to a Specific Value” serve as templates [25]. The code for both Measures is the same, with the difference that the first code utilizes wall type Surfaces and the latter roof type Surfaces. So, we look at only one of the two, specifically for the exterior walls. Like in the previous functions, we establish the choices the user can make. The user can choose if he wants to activate the function of this part of the Measure. The user can input the desired insulation R-value in K*m^2/W units. Finally, the user can enable the option to decrease the R-value by inputting

an R-value lower than the current one. If the user activates the function but doesn't input a new value, default value is applied to the function if it conforms with the last requirement.

So, for this part, we have three inputs (three for the exterior walls version and three for the roofs), one for the activation, one input for the new R-value and one for the option to be able to lower the R-value. The first variable is a Boolean type with the default value as false. The R-value variable is of type Double and has default value 2.34 for exterior walls and 5.4 for roofs. Lastly, we have another Boolean type of input for the option to input a lower R-value. Using the specified values it changes the R-value of the insulation materials in the model.

In Figure 64 (a), (b) and (c), we present the resulting insulation material, wall Construction and Construction set if we enable the "R-value exterior Walls" option and set the insulation value to 3.

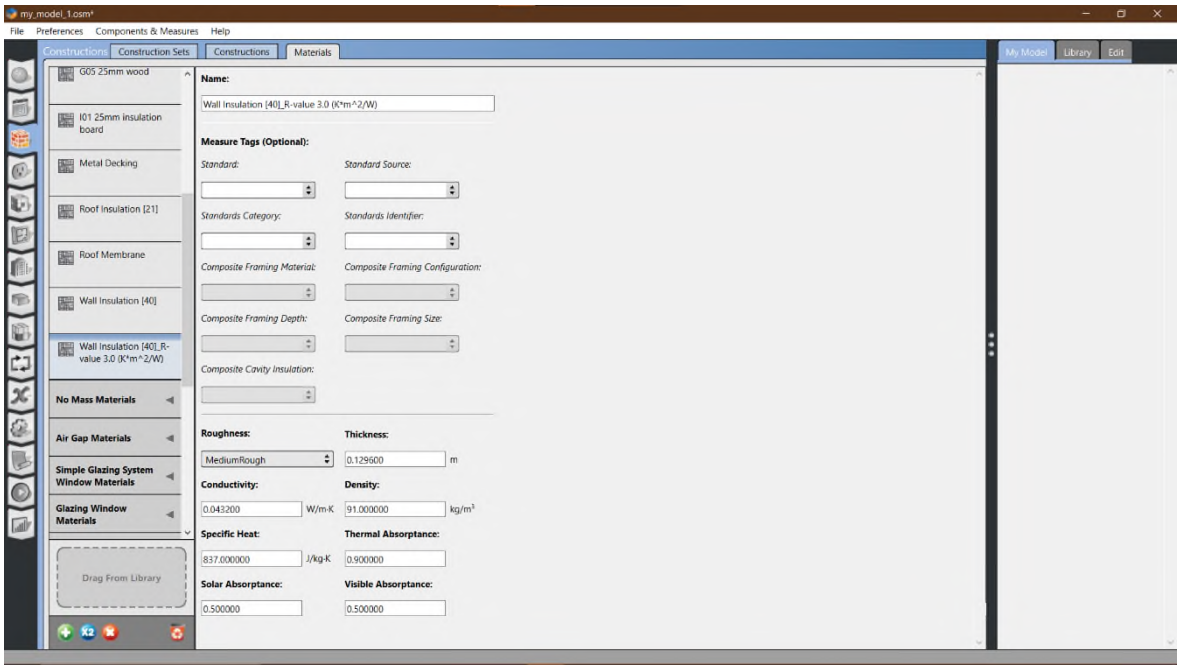

Figure 64(a): Resulting insulation material after applying the "R-value exterior Walls" option.

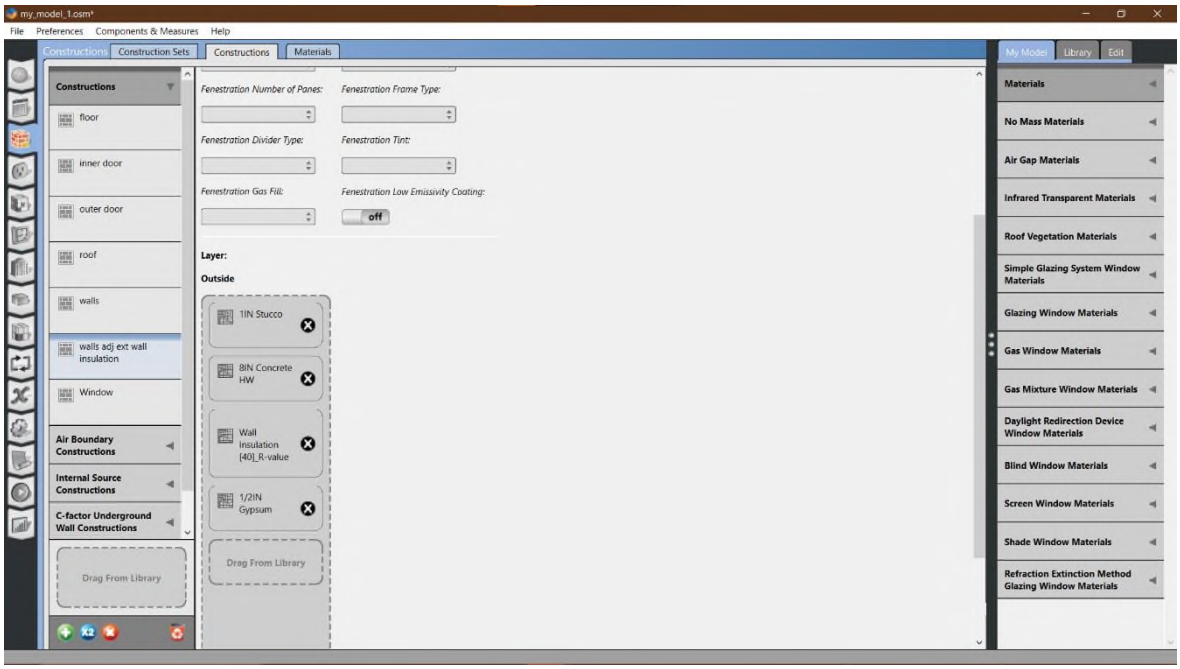

Figure 64(b): Resulting wall Construction after applying the “R-value exterior Walls” option.

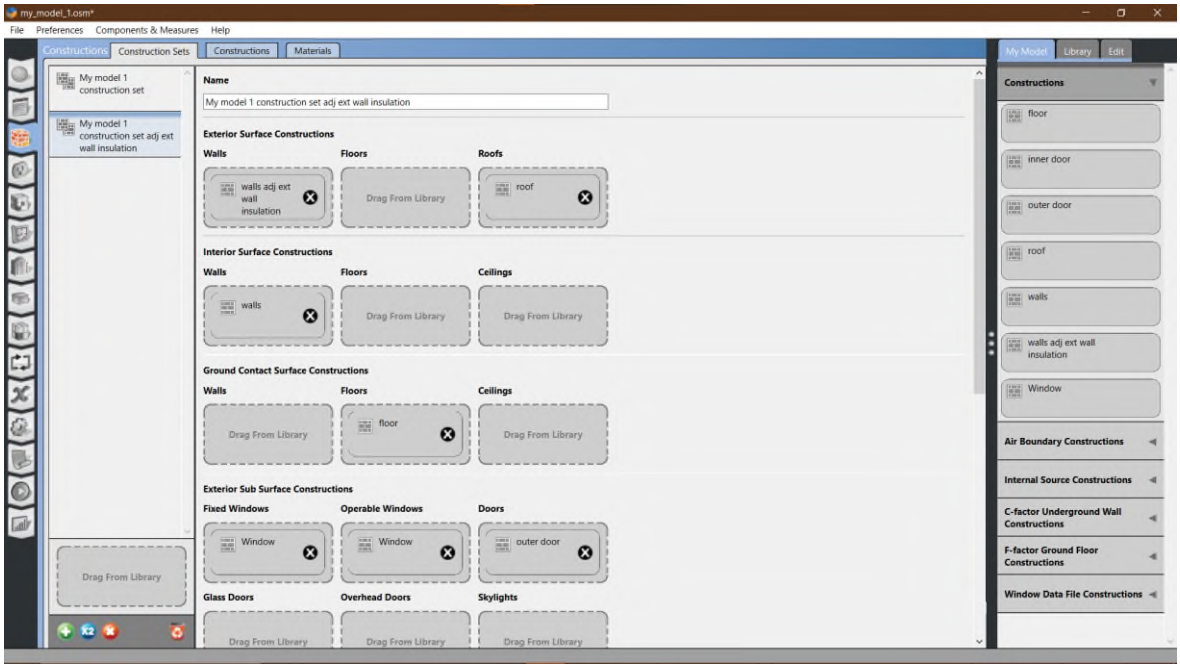

Figure 64(c): Resulting Construction set after applying the “R-value exterior Walls” option.

CHAPTER 4 DEVELOPMENT OF THE WEB APPLICATION

So far, we have created two building models for simulation and a Measure with four core functions that let us manipulate our models and experiment with simulations. The problem is how to distribute them for people to use them. We can obviously use a cloud sharing app like Google drive and provide a link from a website or even from this very thesis. The new problem that arises is that now the user must install the OpenStudio Software Development Kit (SDK), the OpenStudio application, the EnergyPlus application and download the correct weather files like we did, to be able to use what we created. As we mentioned in the beginning, we want this work to be oriented to people who have an average relationship with technology and energy simulation products, that are conscious about the climate problems and the energy crisis and don't want to contribute to the problems. Since almost everyone knows how to use web browsers and applications, the best solution in our opinion is to create a web application. In more detail, we create a web form, where the user can modify both models, simulate them and get the simulation results. During the development of the web application, we use the XAMPP software server package, we write the front end of the page in HTML, the back end in PHP and we use CSS to give a style to our page.

4.1 Developing the simulation form on HTML

We started the development of our web page by creating the form where the user can submit the inputs to the simulation, like when using the conventional Measure from OpenStudio (Figure 65). The title of the page is "Simulation page", the title of the menu "Measure inputs" and the set language is English.

The inputs of each function are grouped together in the form of dropdown menus, with the name of the function as the menu tag. First, we have the "Geometry scaling" inputs. The first input is the function activation and is a checkbox type. The rest of the input fields are of textbox type and have the same labels as those form our Measure. Zero is the minimum value and the user can specify up to three decimal digits. At the top we provide short usage instruction (Figure 66).

The next group of input fields regards the Window Manipulation providing inputs for the window logic (Ribbon window or Windows set), the cardinal direction and the door logic.

The final part of the form corresponds to the functionality that controls the R-value of the building's walls and roof. It includes four chackboxes and two input fields for the new R-values, just like in the Measure (Figure 67). The two menus are exactly the same.

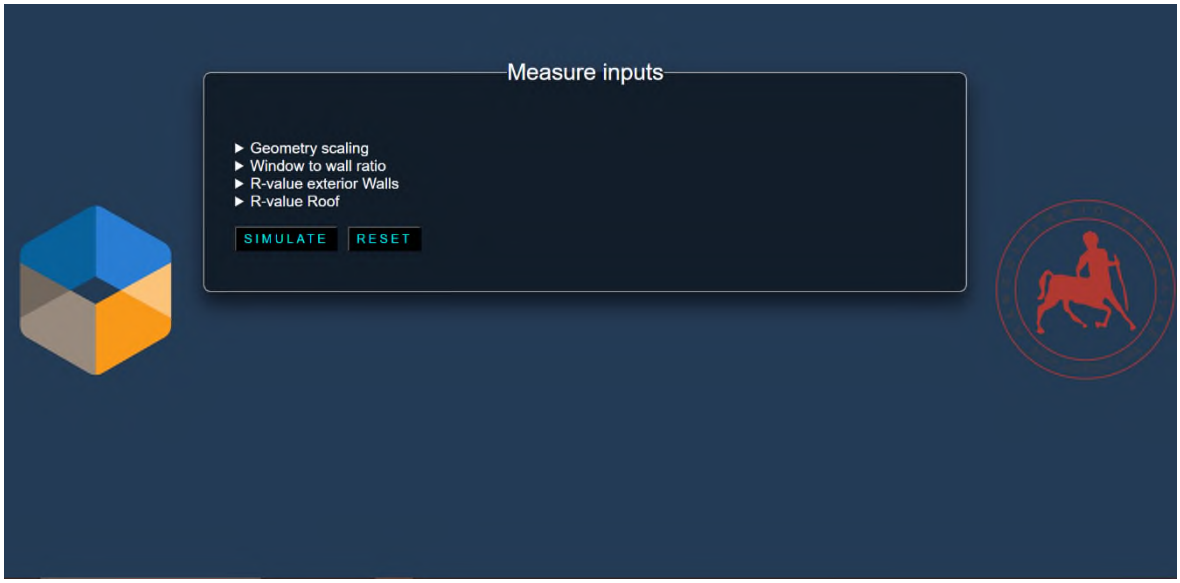

Figure 65: The first look of the web app.

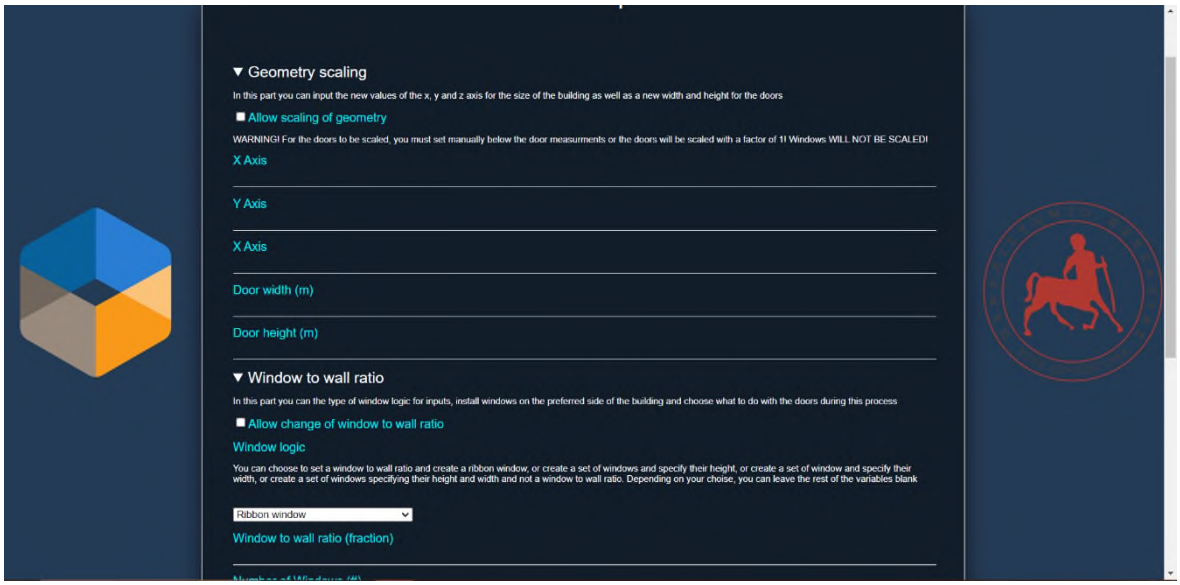

Figure 66: "Geometry scaling" dropdown menu.

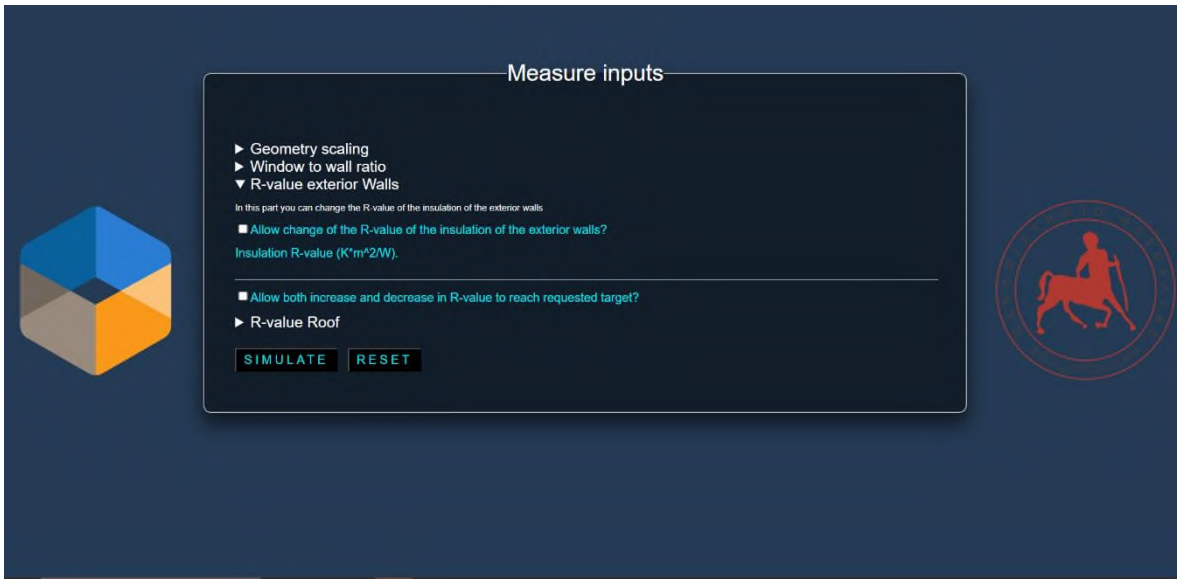

Figure 67: "R-value exterior Walls" dropdown menu.

At the end of the form there are two buttons for submission and reset. The "SIMULATE" button submits the inputs to the server and the "RESET" button deletes any input the user has given.

Lastly, after the inputs have been submitted and the simulations have been finished, two links become available entitled "SIMULATION RESULTS FOR DETAILED MODEL" and "SIMULATION RESULTS FOR SIMPLE MODEL" (Figure 68). These links redirect the user to the result reports for the two models.

The web application supports responsive design for convinience use in smartphones and tablets as the Figure 69 showcases.

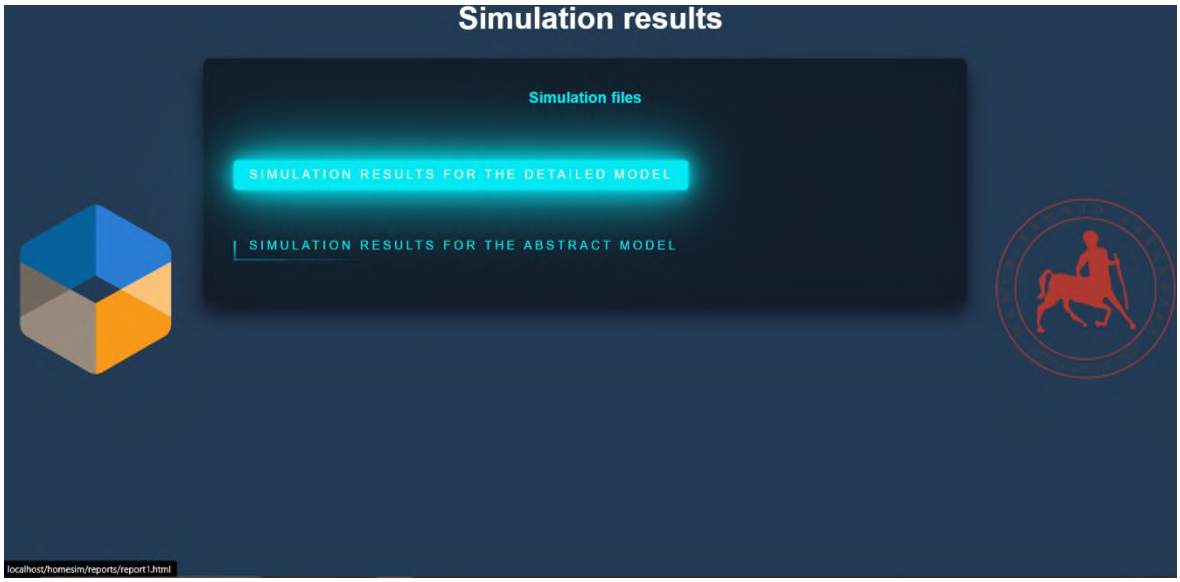

Figure 68: Simulation results page, with links to the reports.

Figure 69: Responsive design example.

4.2 Writing the PHP code for the server side

We developed a PHP script that processes the user input in the server side, invokes simulations and generates the appropriate links to result pages. OpenStudio encodes all required information to launch a simulation to an OSW file (Figure 70 (a), (b), (c)). The file conatins information about the model (.osm file), and the applied Measures along with their respective input values. The PHP code gets user input, builds a new OSW file based on this input, and invokes OpenStudio execution with the new OSW file.

So, to pass the inputs of the user to the simulation, we need to just pass them to the workflow file. Figure 71 lists the user inputs processing code in the server side resulting in the supplied values stored in an array. The workflow files are encoded in JSON format. Therefore, we use a JSON decode function to store the contents of each model's OSW file in an array. We pass user input in the respective models' arrays, we encode arrays to JSON and write back the JSON to the OSW files as depicted in Figure 72.

```
{
   "created_at" : "20230627T124000Z",
   "seed_file" : "../my_model_1.osm",
   "steps" :
   [
      {
         "arguments" :
         {
            "allow_reduction_ew" : false,
            "allow_reduction_rf" : false,
            "allow_scale" : false,
            "allow_set_rv_extwall" : false,
            "allow_set_rv_roof" : false,
            "allow_set_wwr" : false,
            "door_height" : 2.0330159999999999,
            "door_width" : 1,
            "facade" : "North",
            "height_of_window" : 3,
            "number_of_windows" : 2,
            "r_value_ew" : 2.3399999999999999,
            "r_value_rf" : 5.4000000000000004,
            "sillHeight" : 1,
            "split_at_doors" : "Do nothing to Doors",
            "width_of_window" : 3,
            "window_type" : "Ribbon window",
            "wwr" : 0.20000000000000001,
            "x_axis" : 14,
            "y_axis" : 10,
            "z_axis" : 4
         },
```


Figure 70(a): Contents of a workflow.osw file.


```
   "description" : "Combines several measures into one, namely: scale geometry,
                   set window to wall ratio, change the R value of the roof and wall insulations separately",
   "measure_dir_name" : "mymeasure",
   "modeler_description" : "It gives the option to change the scale of the geometry without scaling windows
                            but with the option to scale any doors. Option to place windows on the different sides of the building.
                            Able to change the R-value of the insulation of external walls and roof separately",
   "name" : "mymeasure"
},
```


Figure 70(b): Contents of a workflow.osw file.


```
    {
      "arguments" :
      {
         "units" : "SI"
      },
      "description" : "This measure creates high level tables and charts pulling both from model inputs and EnergyPlus results.
                       It has building level information as well as detail on space types, thermal zones, HVAC systems, envelope characteristics, and economics.
                       Click the heading above a chart to view a table of the chart data.",
      "measure_dir_name" : "OpenStudioResults",
      "modeler_description" : "For the most part consumption data comes from the tabular EnergyPlus results,
                               however there are a few requests added for time series results. Space type and loop details come from the OpenStudio model.
                               The code for this is modular, making it easy to use as a template for your own custom reports.
                               The structure of the report uses bootstrap, and the graphs use dimple js.
                               The new measure warning section will show warnings generated by upstream measures.
                               It will not show forward translation warnings, EnergyPlus warnings, or warnings that might be reported by this measure.",
      "name" : "OpenStudio Results"
    }
  ],
  "updated_at" : "20231027T105003Z",
  "weather_file" : "GRC_Athens.167160_IWEC.epw"
}
```


Figure 70(c): Contents of a workflow.osw file.

```php
<?php
    $temp_array = array_keys($_GET);

    $ini_array = array(
        "allow_reduction_ew" => false,
        "allow_reduction_rf" => false,
        "allow_scale" => false,
        "allow_set_rv_extwall" => false,
        "allow_set_rv_roof" => false,
        "allow_set_wwr" => false,
        "door_height" => 2.033016,
        "door_width" => 1,
        "facade" => "South",
        "height_of_window" => 3,
        "number_of_windows" => 2,
        "r_value_ew" => 2.34,
        "r_value_rf" => 5.4,
        "sillHeight" => 1,
        "split_at_doors" => "Split Walls at Doors",
        "width_of_window" => 3,
        "window_type" => "Ribbon window",
        "wwr" => 0.2,
        "x_axis" => 14,
        "y_axis" => 10,
        "z_axis" => 4,
    );

    for ($i = 0; $i < count($temp_array); $i++) {
        if ($_GET[$temp_array[$i]] != '') {
            $ini_array[$temp_array[$i]] = $_GET[$temp_array[$i]];
        }
    }
```

Figure 71: Passing the input arguments to the server side.

```
$temp_array1 = array_keys($ini_array);

$workflowContent1 = file_get_contents($filename1);
$workflowData1 = json_decode($workflowContent1, true);
$workflowContent2 = file_get_contents($filename2);
$workflowData2 = json_decode($workflowContent2, true);
for ($i = 0; $i < count($temp_array1); $i++) {
   $workflowData1['steps'][0]['arguments'][$temp_array1[$i]] = $ini_array[$temp_array1[$i]];
   $workflowData2['steps'][0]['arguments'][$temp_array1[$i]] = $ini_array[$temp_array1[$i]];
}
$updatedWorkflowContent1 = json_encode($workflowData1, JSON_PRETTY_PRINT);
$updatedWorkflowContent1 = str_replace('"arguments": []', '"arguments": {}', $updatedWorkflowContent1);
file_put_contents($filename1, $updatedWorkflowContent1);
$updatedWorkflowContent2 = json_encode($workflowData2, JSON_PRETTY_PRINT);
$updatedWorkflowContent2 = str_replace('"arguments": []', '"arguments": {}', $updatedWorkflowContent2);
file_put_contents($filename2, $updatedWorkflowContent2);
```

Figure 72: Passing the inputs inside the workflow files.

To run the simulation via a command prompt, we set a file path to the OpenStudio executable, a file path to our model's workflow and combine them in the command: "file path to exe" run - -workflow "path to workflow file". We run the command with the shell_exec() function, rename the result files and copy them to the application file (Figure 73).

```
   $openStudioPath = 'C:\openstudioapplication-1.5.0\bin\openstudio.exe';

   $modelPath1 = 'C:\Users\Neon_Demiurge\Desktop\my_model_1\my_model_1\workflow.osw';
   $command1 = "$openStudioPath run --workflow $modelPath1";

   $modelPath2 = 'C:\Users\Neon_Demiurge\Desktop\my_model_2\my_model_2\workflow.osw';
   $command2 = "$openStudioPath run --workflow $modelPath2";

   shell_exec($command1);
   shell_exec($command2);

   rename($filename3, $filename5);
   rename($filename4, $filename6);

   copy($filename5, $filename7);
   copy($filename6, $filename8);
?>
```

Figure 73: Creating the commands and gathering the result files.

4.3 Creating a CSS file to add style to the web app

The style of the web application was inspired by the style of a CSS style found in a website providing CSS templates for free [28]. With the CSS, we change the letter fonts, colors, sizes, backround images etc. For the backround color we used a dark blue color and

added two images, the logo of the OpenStudio Coalition logo on the left and the logo of the University of Thessaly on the right, which are in fixed positions. We set the form inside a box with a very dark grey backround color, a shadow around the box of the same color and specify the position in the page. We put the title of the form in the center on the top of the box, we set the font size and give it a white color. We set the color white for every menu tag and description and a neon blue for any other text. We add a line below the spaces where the user can fill-in an input. We give a glow effect to the two buttons as well as the two result links. Finally, we created an animation of a line that moves on a rectangular path, around each link that provides the results of the simulation.

CHAPTER 5 CONCLUSION

In this final chapter, we summarize everything we have done in the thesis, we discuss some conclusions we have come to, how our work can benefit the public and lastly how it can be improved.

5.1 Recap

First, we researched and overviewed how the world came in the situation with the energy crisis and the distancing from green energy sources. As we saw, except from the obvious two culprits, the COVID-19 pandemic and the Russo-Ukrainian war, the deep-rooted reliance in fossil fuels and the high cost – low return of green energy methods come into play. While the two crises created a greater need for energy and elevated fuel prices, some countries' refusal to move away from fossil fuels has put their citizens in a difficult financial position. The combination of the bigger energy demand and the buildings' energy inefficiency plays a big role in the financial stability of the citizens. One of the best solutions to those problems is the buildings' energy simulation. But most people don't have access to simulation tools, and even if they had, they would need to have a high-level knowledge of energy simulation. Our solution to this problem is the creation of an easily accessible application, so that the user won't need to have extensive knowledge of the subject.

With that in mind, we created two energy models of a residence for four family members. One model has a detailed depiction of the building's interior and the residents' everyday lives. The second model is a Simple model that consists of one room and a very general depiction of the residents' lives. To create the models, we used the application OpenStudio that also provides easier access to the simulation tool EnergyPlus, an energy simulation application.

Then we created a Measure written in Ruby programing language. This Measure is coded in four functions that change the models' properties and enable experimentation during simulations. The first function controls the models' size, meaning it can change the height length and width, also it can change the doors' size. The second function controls the windows' placement throughout the building. With this function the user can place a

ribbon window or a set of windows in the desired cardinal direction. Also, using the Measure, we can control the window to wall ratio, the windows' width and height, and sill height. Finally, the last two functions control the walls' and roofs' thermal resistances separately.

To wrap it all together, we created a web page where the user can submit his/her inputs to the Measure, simulate the two models, and view the results.

5.2 Conclusions from experiments

During our experiments and simulations, we compared the two models through the results from the simulations. We wanted to test whether we need to create detailed models to get accurate results, or if we can get the desired results from simple models. We compared the Simple model with two versions of the Detailed model. In the Detailed model's first version, we assumed that all the residents and electrical equipment are found only in the Kitchen Space, except for the lights. The schedules about their routine and placement we described during the model's construction were simplified to match the simple model. We call this version Simplified Detailed model. This change was made because in the Simple model we cannot simulate the residents and equipment placement in other rooms, since it is one big room. The second version is the Standard Detailed model we created. The simulations were conducted with OpenStudio application version 1.5.0, OpenStudio SDK 3.6.0 and EnergyPlus version 22.2.0. The device specifications are the following: AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz, 8GB RAM (7.32 GB usable), 64-bit operating system, x64-based processor, Windows 10 Home 22H2 version.

In Figure 74 (a), (b), (c) we compare the Building Summary of the Simplified Detailed model, Simple model and Standard Detailed model. In the three summaries, we are interested in the "Total Site Energy" and "Total Site EUI" values. They depict the total energy used to operate the building during a year and the energy usage intensity in a Space respectively.

Table 9 documents the percentages of electricity usage distribution, since it is the only energy source we have specified for the models. The percentages are derived from the OpenStudio simulation results for the models.

Building Summary

Data	Value
Building Name	Building 1
Total Site Energy	8,989 kWh
Total Building Area	140 m^2
Total Site EUI	64.21 kWh/m^2
OpenStudio Standards Building Type	n/a

Figure 74(a): Building Summary of the Simple model.

Building Summary

Data	Value
Building Name	Building 1
Total Site Energy	8,547 kWh
Total Building Area	140 m^2
Total Site EUI	61.05 kWh/m^2
OpenStudio Standards Building Type	n/a

Figure 74(b): Building Summary of the Simplified Detailed model.

Building Summary

Data	Value
Building Name	Building 1
Total Site Energy	6,661 kWh
Total Building Area	140 m^2
Total Site EUI	47.58 kWh/m^2
OpenStudio Standards Building Type	n/a

Figure 74(c): Building Summary of the Standard Detailed model.

Table 9: Annual End Use percentages for Simplified Detailed, Standard Detailed and Simple models.

End Use	**Consumption of Simplified Detailed model (kWh)**	**Consumption of Simple model (kWh)**	**Consumption of Standard Detailed model (kWh)**
Interior Equipment	4900	4900	4800
Interior Lighting	3100	3500	1600
Cooling	600	600	300
Summary	8600	9000	6700

We observe that by simplifying the Detailed model in terms of Schedule complexity, equipment placement, occupant activity in different Spaces and by focusing the activity on only the Kitchen space, we get roughly the same results from the two models. However, we observe a 1900 kWh difference between the Standard and the Simplified Detailed model. This is also observed by looking at the End Use percentages of the Detailed model's two versions, especially with the Interior Lighting percentages. The reason for this is mainly the difference in the two versions Schedules. In the Standard Detailed model, the occupants have multiple schedules that represent their presence in different Spaces throughout the day. Also, in the Standard version, the Lighting schedules are more precise and describe in detail when the lights are in use. This is in contrast with the Simplified version's generalized schedule where all the lights are in use from a specified hour and forward. It is evident that the Standard version gives a more accurate image of the energy consumption in the model.

All in all, we have come to three main conclusions. Firstly, it is more beneficial to create detailed models, since they offer a more accurate description of a complex building's behavior than simple models. Secondly, the energy consumption level of detail for a complex building model is affected mainly by the Schedules and by extension from the interior layout. The Schedules dictate the operation time and intensity of the electrical appliances, such as lights, which greatly affect the total energy consumption. Lastly, we can use simple models for buildings which we do not care about the schedules precision and want a rough estimate of the energy consumption. Also, we can utilize simple models for small buildings or apartments, since they have extremely simple interiors and do not require detailed Schedules to study them accurately.

5.3 Suggestions for improvement

There are a few ways that our tool could improve. The major way that can elevate the tool's functionality is to include a way for the user to change the materials used in the building. To fully implement such a feature, we would need to do research on the materials each country uses in constructions. We should also do research into the materials' prices and the general prices of companies that do building retrofits. That way, the user can have a good view of the cost of the construction or retrofit of the building. Another improvement

we could make is to include more weather files from other countries. We could also add a way to specify how many residents there can be in the house. Moreover, we can add an option to install HVAC equipment wherever the user wants. Finally, the biggest improvement we can make is to add a machine learning model to give suggestions to the user depending on the location.

References

1. Tufail, Muhammad, et al. "Does financial inclusion promote a green economic system? Evaluating the role of energy efficiency." *Economic research-Ekonomska istraživanja* 35.1 (2022): 6780-6800.
2. Chong, Cheng Tung, et al. "Post COVID-19 ENERGY sustainability and carbon emissions neutrality." *Energy* 241 (2022): 122801.
3. Berahab, Rim. "The energy crisis of 2021 and its implications for africa." *Policy* 1967 (2022).
4. Popkostova, Yana. "Europe's energy crisis conundrum." *European Union Institute for Security Studies* (2022).
5. Farghali, Mohamed, et al. "Strategies to save energy in the context of the energy crisis: a review." *Environmental Chemistry Letters* (2023): 1-37.
6. Drago, Carlo, and Andrea Gatto. "An interval-valued composite indicator for energy efficiency and green entrepreneurship." *Business Strategy and the Environment* 31.5 (2022): 2107-2126.
7. Simshauser, Paul. *The 2022 Energy Crisis: horizontal and vertical impacts of policy interventions in Australia's National Electricity Market*. No. 2307. Faculty of Economics, University of Cambridge, 2022.
8. Tracker, Climate Action. "Global reaction to energy crisis risks zero carbon transition." *Analysis of Government Responses to Russia's Invasion of Ukraine* (2022).
9. Vrana, Vasiliki, et al. "EU citizens' Twitter discussions of the 2022–23 energy crisis: A content and sentiment analysis on the verge of a daunting winter." *Sustainability* 15.2 (2023): 1322.
10. Chen, Yongbao, et al. "Physical energy and data-driven models in building energy prediction: A review." *Energy Reports* 8 (2022): 2656-2671.
11. Shtunder, I., et al. "SUSTAINABLE DEVELOPMENT OF THE ECONOMY IN THE CONDITIONS OF THE ENERGY CRISIS." *Scientific Bulletin of National Mining University* 4 (2022).
12. D'Agostino, Delia, et al. "How will future climate impact the design and performance of nearly zero energy buildings (NZEBs)?." *Energy* 240 (2022): 122479.
13. Matsumoto, Shigeru, Kenichi Mizobuchi, and Shunsuke Managi. "Household energy consumption." *Environmental Economics and Policy Studies* 24 (2022): 1-5.
14. Ozarisoy, B. "Energy effectiveness of passive cooling design strategies to reduce the impact of long-term heatwaves on occupants' thermal comfort in Europe: Climate change and mitigation." *Journal of Cleaner Production* 330 (2022): 129675.
15. Hong, Tianzhen, et al. "Commercial building energy saver: An energy retrofit analysis toolkit." *Applied Energy* 159 (2015): 298-309.

16. Sun, Kaiyu, et al. "A pattern-based automated approach to building energy model calibration." *Applied Energy* 165 (2016): 214-224.
17. Weaver, Evan, et al. *Rapid application development with Openstudio*. No. NREL/CP-5500-54998. National Renewable Energy Lab.(NREL), Golden, CO (United States), 2012.
18. Chen, Yixing, Tianzhen Hong, and Mary Ann Piette. "Automatic generation and simulation of urban building energy models based on city datasets for city-scale building retrofit analysis." *Applied Energy* 205 (2017): 323-335.
19. Τσαούσης, Ιωάννης. "Ενεργειακή σχεδίαση κτηρίου με στόχο τη μηδενική ή ελάχιστη ενεργειακή κατανάλωση. Μελέτη περίπτωσης: ενεργειακή σχεδίαση βασισμένη στην ευρωπαϊκή και ελληνική νομοθεσία λαμβάνοντας υπόψιν το μικροκλίμα της περιοχής." (2022).
20. Μηλιόπουλος, Ραφαήλ Β. *Ενεργειακή μοντελοποίηση κτηρίου χρησιμοποιώντας EnergyPlus και OpenStudio*. BS thesis. 2017.
21. Μυρωνίδης, Αλέξανδρος. *Μελέτη των Υλικών Αλλαγής Φάσης στα πλαίσια του ενεργειακού και περιβαλλοντικού σχεδιασμού για κατοικία στην πόλη της Θεσσαλονίκης*. Diss. Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, 2019.
22. Άρτεμις Μπαρδούτσου, "Σχεδιασμός και Προσομοίωση κτιρίων για τον ενεργειακό αυτοματισμό κτιρίων", Διπλωματική Εργασία, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πολυτεχνείο Κρήτης, Χανιά, Ελλάς, 2020
23. Brackney, Larry, et al. *Building energy modeling with OpenStudio*. New York: Springer International Publishing, 2018.
24. https://study.com/learn/lesson/sensible-heat-vs-latent-heat-function-differences-examples.html#:~:text=Latent%20heat%20is%20the%20energy,than%20in%20raising%20its%20temperature, (Last accessed: October 16th, 2023)
25. NREL, Building Component Library (BCL), https://bcl.nrel.gov/, (Last accessed October 27th, 2023)
26. BigOptiBase Research Project, T1EDK-04605, WP4.4, Radio Network Simulation Subsystem.
27. https://github.com/CITY-at-UMD/openstudio_measures/blob/master/individualWindows/SetIndividualWindows.rb, (Last day accessed: September 1st, 2023)
28. https://codepen.io/soufiane-khalfaoui-hassani/pen/LYpPWda, (Last accessed: October 19th, 2023)