Automation of Building Energy Efficiency Using BIM

A decision support BIM-based tool to optimize the EI of buildings

Personal Information

Author	**Yassin Radwan**
Student ID	**1368664**
Email	**Yassinmohamed70@icloud.com**

Institute

University	**Eindhoven University of Technology (TU/e)**
Faculty	**Faculty of the Built Environment**
Department	**Construction Management & Engineering (CME)**

Graduation Committee

Chairman	**prof. dr. ir. B. (Bauke) de Vries**
Graduation supervisor (TU/e)	**Ir: Jakko Heinen**
Graduation supervisor (TU/e)	**Ir: Luuk Wijnholts**

Date of Final Presentation

13-01-2021

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Table of Contents

Preface

This thesis outlines the results of the study I have conducted over the past six months as part of my graduation from the Eindhoven University of Technology (TU /e). I am proud to present the results after a period of hard work, growth of knowledge, and scientific discoveries through this thesis, which concludes the Masters of Science Construction Management and Engineering.

Several persons have contributed academically, practically, and with support to my master thesis. A significant part of this contribution belongs to my supervisors; I would therefore firstly like to thank my head supervisor Jakko Heinen who, from the very beginning, was closely involved in this project and in the moments when I most needed it, Jakko contributed his endless guidance and feedback, his attention to details helped me improve my research skills, and this thesis was improved sequentially because of his supervising. I was deeply satisfied with the constructive discussions about the research work.

Moreover, I'm sincerely thankful to Luuk Wijnholts who was always willing to provide guidance and feedback throughout the entire research. Furthermore, I'd like to thank Prof. Aant Van Der Zee for sharing his knowledge, experience, guidance, and offering his help.

Lastly, I'd like to thank my parents, my both brothers, and my fiancée for their support either practical or moral during the past six months, they were always by my side, encouraging me, believing in me, and sharing my moments of happiness and stress, without their contribution this work wouldn't have been done.

I hope you find this research interesting

Yassin Radwan

Eindhoven, November 2020

Summary

The construction industry accounts for around 50 percent of CO2 emissions. Also, it accounts for about 20-50 percent of all natural resources consumed, and 50% of all the produced solid waste, these problems takes place during the construction phase of the life cycle of the building, causing many environmental impacts (PULASKI, 2004; KHASREEN; BANFILL; MENZIES, 2009; PROBERT et al., 2010). These negative related environmental impacts underline the need for sustainable construction processes and effective ways of managing sustainability measures (TAM et al., 2006). This issue has led many researchers to apply many strategies to enhance sustainability in the construction sector. However, those studies were mainly focused on addressing the evaluation of the embodied impact of construction materials by exploring the various stages of the life cycle of a facility.

The environmental effects of the built environment can be minimized by using building performance simulation tools (BPS). The energy efficiency of buildings is predicted by the BPS tools. BPS tools' main objective is forecasting, evaluating, and demonstrating the energy performance of the facility. These tools were used in the late design phase simulation, but then their use was extended to the entire life cycle of the building (Coakley et al..2014). Since a large number of software tools are available to predict the energy and efficiency of buildings, number of questions were arose by many researchers, which is due to the regular use of these simulation tools as the activities and method involved in designing a simulation model using these BPS tools. Many limitations have been detected in using the BPS tools; one of these limitations is that these tools do not provide the user with sufficient assistance in making decisions. By using the simulation of the building during the design process, however, the situation becomes much more complicated since the designers need various kinds of data at the different design levels; the practitioners of the project have different backgrounds (Punjabi, 2005) and that made it difficult for architects to incorporate the energy analysis results throughout the design process (Attia, 2011).

Building Information Modeling (BIM) has shown many benefits in construction, for instance: It offers an integrated platform for various project participants in the lifecycle of a project. BIM models can simulate repairs or upgrading methods and thus, help to decrease the cost of facility management and enhance the maintenance activities, and also provide a reliable estimation of the cost of renovation; this can help in the design and construction phases to coordinate the procurement process (Schwegler & Nies, 2010). Moreover, many benefits in decision-making processes have been provided by the Building Information Modeling, as BIM provides all the engineering participants an opportunity to use a unified shared model to optimally achieve the project objectives. The data sharing offered by BIM between different team members allows for faster assessment and control of information (Qian, 2012).

Integrating BIM and BPS tools has solved many issues, as this research proposes a solution towards providing sustainable solutions in the designing process. Integrating BIM and BPS tools allows the possibility for creating building energy models which prevent the excessive processes, hence facilitating the creation of the energy models as the traditional process of the

creation of the energy models is time-consuming and may lead to many design errors (Jeong & Kim, 2016). Moreover, assisting the designers in the decision making processes, as data is automatically collected, the time consumption in the construction of the 3D model and the development of a BEP analysis is significantly reduced.

The main research objective of this graduation thesis is reducing the environmental impact caused by the built environment. This research proposes an integrated BIM-BPS tool that simulates the Energy Index of the building regarding the energy labeling earlier in the design phase. As the building performance simulation tools provide a prediction for the energy performance, this prediction helps the construction projects' participants to estimate the complexity of the building's performance, thus they would have the possibility to develop a sustainable solution in the early stage of the project as well as demonstrating a BIM-based energy evaluation method which allows designers to freely concentrate on the conceptual design while getting an insight into the possible consequences of design decisions at the same time.

The methodology of implementing the integrated decision support BIM-based tool is based on deriving two datasets. Firstly, the building's characteristics considering geometric values and thermal values, while the energy dataset is retrieved from the Dutch energy norm NTA 8800. The building' characteristics dataset is derived from the IFC file format of the 3D BIM model, the IFC file format enables the necessary information needed for the energy assessment to be derived and then the tool applies the EI calculations to be inserted based on the values and norms of NTA 8800 and check whether the building model complies with the EI required based on the NTA 8800 to optimize the energy labeling of the building. The developed tool is integrated by implementing a Python code script.

Lastly, as mentioned, the designed tool is a decision support tool, as the developed tool includes a feedback option that is associated with the final energy evaluations, which facilitates the ability to make sustainable decisions earlier in the design stage as it helps the designers of the project with sustainable design alternatives and get into the complexity of the designed model's energy performance. The developed method is therefore a decision support method that allows energy performance calculations to be automated towards sustainability and to reduce the negative environmental impact.

Samenvatting

De bouwsector is verantwoordelijk voor ongeveer 50 procent van de CO2-uitstoot. Het is ook goed voor ongeveer 20-50 procent van alle verbruikte natuurlijke hulpbronnen, en 50% van al het geproduceerde vaste afval.Deze problemen doen zich voor tijdens de bouwfase van de levenscyclus van het gebouw en veroorzaken veel milieueffecten (PULASKI, 2004; KHASREEN ; BANFILL; MENZIES, 2009; PROBERT et al., 2010). Deze negatieve milieueffecten onderstrepen de noodzaak van duurzame bouwprocessen en effectieve manieren om duurzaamheidsmaatregelen te beheren (TAM et al., 2006). Deze kwestie heeft ertoe geleid dat veel onderzoekers veel strategieën hebben toegepast om de duurzaamheid in de bouwsector te vergroten. Die studies waren echter vooral gericht op de evaluatie van de belichaamde impact van bouwmaterialen door de verschillende stadia van de levenscyclus van een faciliteit te onderzoeken.

De milieueffecten van de gebouwde omgeving kunnen worden geminimaliseerd door gebruik te maken van de Building Performance Simulation Tools (BPS). De energie-efficiëntie van gebouwen wordt voorspeld door de BPS-tools. Het hoofddoel van BPS-tools is het voorspellen, evalueren en aantonen van de energieprestaties van de faciliteit. Deze tools werden gebruikt in de simulatie van de late ontwerpfase, maar daarna werd het gebruik ervan uitgebreid tot de hele levenscyclus van het gebouw (Coakley et al..2014). Aangezien er een groot aantal softwaretools beschikbaar is om de energie en efficiëntie van gebouwen te voorspellen, zijn er door veel onderzoekers een aantal vragen gerezen, wat te wijten is aan het regelmatige gebruik van deze simulatietools als de activiteiten en methode die betrokken zijn bij het ontwerpen van een simulatiemodel met behulp van deze BPS-tools. Er zijn veel beperkingen ontdekt bij het gebruik van de BPS-tools; een van deze beperkingen is dat deze tools de gebruiker onvoldoende hulp bieden bij het nemen van beslissingen. Door tijdens het ontwerpproces gebruik te maken van simulatie van het gebouw, wordt de situatie echter veel gecompliceerder aangezien de ontwerpers verschillende soorten gegevens nodig hebben op de verschillende ontwerpniveaus; de uitvoerders van het project hebben verschillende achtergronden (Punjabi, 2005) en dat maakte het moeilijk voor architecten om de resultaten van de energieanalyse tijdens het ontwerpproces mee te nemen (Attia, 2011).

Building Information Modeling (BIM) heeft veel voordelen opgeleverd in de bouw, bijvoorbeeld: het biedt een geïntegreerd platform voor verschillende projectdeelnemers in de levenscyclus van een project. BIM-modellen kunnen reparatiemethoden of upgrademethoden simuleren en zo de kosten van facility management helpen verlagen en de onderhoudsactiviteiten verbeteren, en ook een betrouwbare schatting van de renovatiekosten opleveren; dit kan helpen in de ontwerp- en bouwfase om het inkoopproces te coördineren (Schwegler& Nies, 2010). Bovendien zijn er veel voordelen in besluitvormingsprocessen geleverd door de Building Information Modelling, aangezien BIM alle technische deelnemers de mogelijkheid biedt om een uniform gedeeld model te gebruiken om de projectdoelstellingen optimaal te bereiken. Het delen van gegevens dat door BIM wordt aangeboden tussen verschillende teamleden, zorgt voor een snellere beoordeling en controle van informatie (Qian, 2012).

De integratie van BIM- en BPS-tools heeft veel problemen opgelost, aangezien dit onderzoek een oplossing voorstelt om duurzame oplossingen te bieden in het ontwerpproces. De integratie van BIM- en BPS-tools biedt de mogelijkheid om energiemodellen voor gebouwen te creëren die buitensporige processen voorkomen, waardoor het creëren van energiemodellen wordt vergemakkelijkt, aangezien het traditionele proces van het creëren van energiemodellen tijdrovend is en tot veel ontwerpfouten kan leiden (Jeong& Kim, 2016). Bovendien wordt het assisteren van de ontwerpers bij de besluitvormingsprocessen, aangezien de gegevens automatisch worden verzameld, het tijdverbruik bij de constructie van het 3D-model en de ontwikkeling van een BEP-analyse aanzienlijk verminderd.

Het belangrijkste onderzoeksdoel van deze afstudeerscriptie is het verminderen van de milieu-impact van de gebouwde omgeving. Dit onderzoek stelt een geïntegreerde BIM-BPS-tool voor die de energie-index van het gebouw met betrekking tot de energie-etikettering eerder in de ontwerpfase simuleert. Aangezien de simulatietools voor gebouwprestaties een voorspelling geven voor de energieprestaties, helpt deze voorspelling de deelnemers aan bouwprojecten om de complexiteit van de prestaties van het gebouw in te schatten, zodat ze de mogelijkheid hebben om in een vroeg stadium van het project een duurzame oplossing te ontwikkelen. evenals het demonstreren van een op BIM gebaseerde energie-evaluatiemethode waarmee ontwerpers zich vrij kunnen concentreren op het conceptuele ontwerp en tegelijkertijd inzicht krijgen in de mogelijke gevolgen van ontwerpbeslissingen.

De methodologie voor het implementeren van de op BIM gebaseerde tool voor geïntegreerde besluitvorming is gebaseerd op het afleiden van twee datasets. Ten eerste de kenmerken van het gebouw, rekening houdend met geometrische waarden en thermische waarden, terwijl de energiedataset wordt opgehaald uit de Nederlandse energienorm NTA 8800. De karakteristieken van het gebouw zijn afgeleid van het IFC-bestandsformaat van het 3D BIM-model, het IFC-bestandsformaat maakt het mogelijk benodigde informatie nodig om de energiebeoordeling af te leiden en vervolgens past de tool de in te voeren EI-berekeningen toe op basis van de waarden en normen van NTA 8800 en controleert of het gebouwmodel voldoet aan de vereiste EI op basis van de NTA 8800 om de energie te optimaliseren etikettering van het gebouw. De ontwikkelde tool is geïntegreerd door een Python-codescript te implementeren.

Ten slotte, zoals gezegd, is de ontworpen tool een beslissingsondersteunend instrument, aangezien de ontwikkelde tool een feedbackoptie bevat die is gekoppeld aan de uiteindelijke energie-evaluaties, waardoor het gemakkelijker wordt om eerder in de ontwerpfase duurzame beslissingen te nemen, aangezien het de ontwerpers van de project met duurzame ontwerpalternatieven en verdiep u in de complexiteit van de energieprestaties van het ontworpen model. De ontwikkelde methode is daarmee een beslissingsondersteunende methode waarmee energieprestatieberekeningen kunnen worden geautomatiseerd richting duurzaamheid en om de negatieve milieu-impact te verminderen.

Abstract

Around 50 percent of CO2 emissions are generated by the building industry. It also accounts for about 20-50 percent of all natural resources consumed, and 50 percent of all solid waste generated, these problems occur during the building's life cycle construction process, causing many environmental impacts. This study explores the topic of construction sustainability by seeking to encourage designers to make more environmentally sustainable design choices to optimize the energy index of the building design and take into consideration the enhancing of energy labeling in buildings. Reducing the environmental damage generated by the built environment is the key research goal of this graduation thesis. This study proposes an integrated decision support BIM-BPS tool that simulates the building's energy index for energy labeling earlier in the design stage. The integration approach is based on deriving two datasets, the building's characteristics considering geometric values and thermal values, while the energy dataset is retrieved from the Dutch energy norm NTA 8800. The building' characteristics dataset is derived from the IFC file format of the 3D BIM model, the IFC file format enables the necessary information needed for the energy assessment to be derived and then the tool applies the EI calculations to be inserted based on the values and norms of NTA 8800 and check whether the building model complies with the EI required based on the NTA 8800 to optimize the energy labeling of the building. The developed tool is integrated by implementing a Python code script.

The developed tool includes a feedback option which is linked to the final energy assessments, this feedback facilitates the ability to make sustainable decisions earlier in the design phase as it assists the project's designers with sustainable design solutions and to get into the complexity of the energy performance of the designed model. Hence, the developed tool is a decision support tool that enables the automation of energy performance calculations towards sustainability and reducing the negative impact on the environment.

List of Abbreviations

3D	: Three Dimensional
AEC	: Architecture, Engineering, and Construction
BIM	: Building Information Modelling
BEP	: Building Energy Performance
BPMN	: Business Process Modeling and Notation
BPS	: Building Performance Simulation
IFC	: Industry Foundation Classes
GUID	: Globally Unique IDentifier
CO_2	: Carbon dioxide
EPC	: Energy Performance Coefficient
CHP	: Combined Heat and Power
HVAC	: Heat, Ventilation, and Air Conditioning
MEP	: Mechanical, Electrical, and Plumbing
STEP	: Standard for the Exchange of Product
ISSO	:Instituut voor Studie en Stimulering van Onderzoek (NL)
NEN	:Nederlands Normalisatie Instituut (NL)
NZEB	: Neutral Zero Energy Buildings
NTA	:Nederlandse Technische Afspraken
EI	: Energy Index
EPBD	: Energy Performance of Buildings Directive

List of Figures

List of Tables

List of Appendix tables

List of Appendix figures

Chapter 1: Introduction

1.1 Problem definition

1.1.1 Background

A detrimental influence on the environment has been reported with the noted development in the construction sector. Throughout the construction's entire life cycle, building products (and in particular those of commercial use) entail energy-intensive systems, containing energy-demanding properties and facility operations, as well as occupants who are the driving force of operations, conducting regular business processes and directly influencing overall energy consumption.

Three interrelated spatiotemporal groups of factors strongly depend on the energy efficiency of construction products during operation: construction properties and equipment, environmental conditions, and occupant's conduct. Besides, global building energy consumption accounts for 30 percent of CO2 emissions (Mardiana, 2015). It has been reported that buildings consume about one-third of the total primary energy resources, making it a significant goal for the implementation of energy efficiency steps (Mardiana, 2015).

The use of energy in office buildings is a major source of carbon emissions and in such environments, it is highly dependent on human presence and behavior. Human actions and occupant desires are significant explanations for the difference between the energy efficiency of the expected and actual building. This leads to the issue of the effect of users and their actions on the use of operations and the option of the design process that can help improve their use of operations. Comfort standards differ greatly between households and households, including in circumstances where households have the same environmental context or access to similar facilities. People do not behave or have the same level of comfort according to their gender, age, social status, etc. As a consequence, there is substantial variation in household lifestyles (Paone & Bacher, 2018).

Many researchers have addressed the impact of the AEC industry on the environment and claimed that the AEC industry is a main source of environmental pollution *(Morledge and Jackson, 2001; Ball, 2002; Chen et al., 2004; Lam et al., 2011)*. Various methods and strategies have been proposed to reduce the negative impact which is caused by the construction industry on the environment (Umar et al., 2013; Jin et al. 2019; van Gemert, 2019; Karvari, 2017), however, those studies were mainly focused on addressing the evaluation of the embodied impact of construction materials by exploring the various stages of the life cycle of a facility.

Moreover, other studies have proposed to the energy efficiency of commercial and home buildings by taking into account different environmental factors (Langevin et al., 2014; Liao et al., 2012; Azar et al. 2012), however, these studies were only focused on addressing the human behavior of occupants in the buildings to detect the energy peak of the buildings. The models used in their studies still faces a lack of interdisciplinary and data acquisition automation.

To conclude, most of the methods and strategies used addressing the factors affecting the environment used the traditional way of involving the BPS tools in the project, as these tools can enhance the building's energy efficiency if they're involved in the early design phases of the project. The early involvement of BPS tools in the design phase can assist the designers to make sustainable decisions earlier and build sustainable and therefore reducing the environmental impact. The figures below show the difference between the Traditional and integrated BPS tool's involvement in the design process.

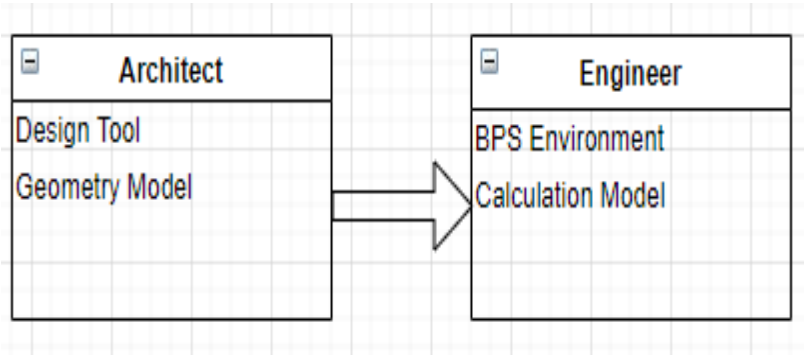

Figure 1 Traditional BPS tools involvement

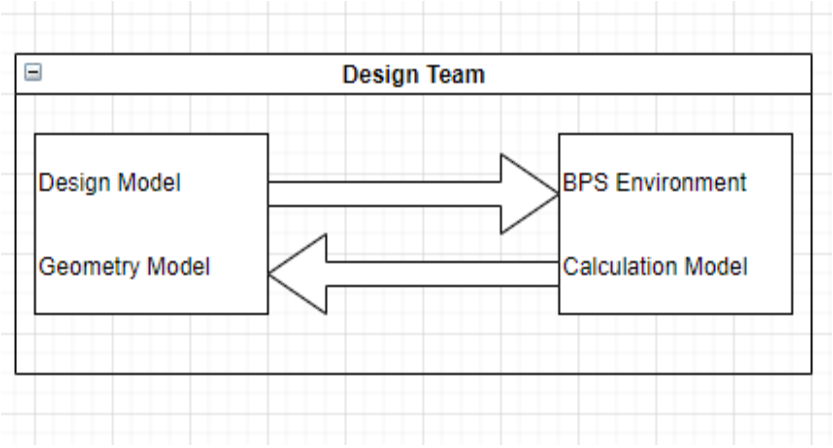

Figure 2 Integrated Design Process

1.1.2 Building Performance Simulation Tools

The environmental effects of the built environment can be minimized by using building performance simulation tools (BPS). The energy efficiency of buildings is predicted by the BPS tools. BPS tools' main objective is forecasting, evaluating, and demonstrating the energy performance of the facility. These tools were used in the late design phase simulation, but then their use was extended to the entire life cycle of the building (Coakley et al..2014).

BPS tools include a collection of inputs and variables that are placed into simulation engines to provide users with an output so that the user can gain insight into the effect of the energy efficiency of the building and could provide the possibility to make decisions based on the

outcome. Researchers have been addressing the possibility of whether these tools could be used in the early design phase and provide the user with inputs to meet the decision-making process requirements of the designers (Attia et al., 2013; Morbitize et al., 2016).

1.1.3 Importance of the early involvement of BPS tools

It is generally accepted that it is much more effective and economical to anticipate and evaluate potential actions in advance than to address issues resulting from the design process while the building is in the use. Nevertheless, the adoption of building performance modeling is surprisingly limited in current building design practice. In general, the actual implementation is limited to the final phases of the building design.

Involving BPS tools during the early design phases could predict the efficiency of design alternatives in terms of comfort, energy consumption, the total cost of the life cycle, etc., which can therefore lead to better-informed design decisions, can also lead to a better understanding of the actions of different climate agents and thus provide confidence in design. They are also particularly relevant for the development of a preliminary assessment of complex design strategies. Moreover, designs of buildings that rely on energy assessments from the early design stages of building design, efficiency, and environmental impact can sometimes exceed code requirements by more than 50 percent and even reduce initial costs at the same time (Punjabi, 2005).

1.1.4 BIM-BPS tools integration

Concerning the coming new building standards, it is becoming increasingly important to conduct Building Performance Simulation (BPS) as the demands for sustainable buildings are increasing. Technological and environmental factors are significantly considered to design sustainable buildings; this enables the advent of BIM in the AEC industry to produce designs that increase sustainability in the construction sector. BIM generates design and displays semantic information. Building Information Modeling (BIM) includes ICT frameworks and technologies that can support stakeholders' collaboration over projects life-cycle by facilities to insert, extract, update, or modify information in the BIM model. BIM applications produce more usable data and information for visualizations and simulations than the traditional and separate project application tools. Studies connected to BIM have moved from basically the functions to store, link, and exchange the project-based technical information to cover all data/information/knowledge analysis of the whole project lifecycle that benefits all stakeholders.

BIM developments have enabled its use in the BPS practice, as this integration has resolved data issues related to modeling software, this integration has lead to more accurate analysis that could be performed to a design even in the early design phases by providing input on the design's energy efficiency; participants of the project would gain a deep understanding of the

energy performance of the building. Hence, the main goal of integrating the BIM-BPS technique is stated in the following points:

a) Provides insights into the strategies of modeling and simulation of building performance and their application to the performance-based design and the operation of buildings and their systems.

b) Provides readers with the essential concepts of performance-based design and operation computational support.

c) It provides examples of how to use building simulation techniques, their limitations, and future direction for practical design, management, and operation.

1.2 Research objectives

The main research objective of this graduation thesis is reducing the environmental impact caused by the built environment. This research proposes an integrated BIM-BPS tool that simulates the Energy Index of the building regarding the energy labeling earlier in the design phase. As the building performance simulation tools provide a prediction for the energy performance, this prediction helps the construction projects' participants to estimate the complexity of the building's performance, thus they would have the possibility to develop a sustainable solution in the early stage of the project, however, these BPS tools are not able to send feedback to the participants, therefore connecting BIM with BPS is important for this point, as BIM generates the design and displays the semantic information. The main objective of this research is to demonstrate that a BIM-based energy evaluation method will allow designers to freely concentrate on the conceptual design while getting an insight into the possible consequences of design decisions at the same time. (Gkatzios , 2019), has developed a similar approach, however, in his paper, he only addressed the heat demand of the building, while this thesis focuses on the whole building factors regarding the thermal energy assessment of the design model.

1.3 Research questions

The principal research question to be addressed in this study, based on the problem definition and the objectives described above is as follows:

How can the integration of 3D Building Information Models and the Building Performance Simulation approach be used to optimize the EI regarding the thermal energy performance in buildings?

The main research question will be supported through the following sub-questions:

i. *How can BIM contribute to building performance simulation of a building project during the conceptual phase?*
ii. *How can BIM-BPS tools be early involved in the project and how this integration can affect the decision making process?*
iii. *What is the added value to the industry by applying the BIM-Building Performance Simulation approach?*

1.4 Research Model

As shown in figure (3) below, this research study is divided into four separate phases. (i) Literature Review, (ii) Development Approach, (iii) Tool Development, (iv) Conclusion. In the literature review, the impact of the built environment will be discussed, followed by a discussion on the benefits of BIM and BPS tools, the existed tools in the industry, and how BIM and BPS could be integrated, and lastly, the Literature review will discuss the energy standards regarding the residential building in the Netherlands. Moreover, the development approach will discuss the methodology of the implementation of the integrated BIM-BPS tool. Moreover, the Tool Development seeks to explain the details of the tool implementation, regarding the energy requirements, tool limitations, and interface. Lastly, the Conclusion will discuss a short overview of the steps taken throughout the research and a future research recommendation will be mentioned.

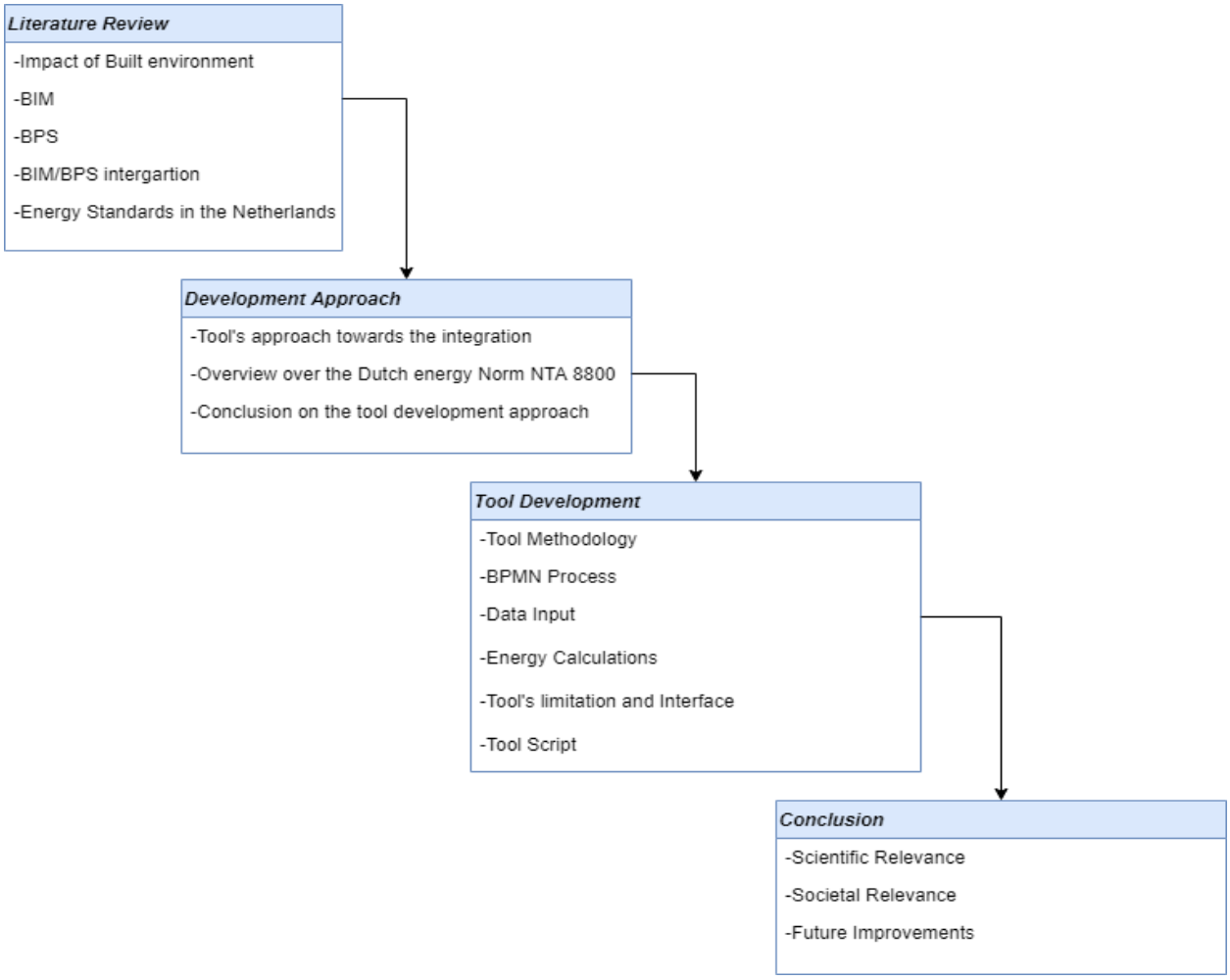

Figure 3 Research model

1.5 Scientific importance

The scientific importance of this research takes into consideration the complexity of energy assessment during the design phase. As many researchers noted the negative impact that may be caused by the built environment, the developed tool seeks to enhance the building's energy performance. The main goal is achieved as the designed tool perform an energy analysis for the building model and send feedbacks to the designers earlier in the design phase of the project, this process will assist the designers to get a deep insight into the complexity of the energy performance of the building and hence, make sustainable decisions and enhance the quality of the design.

Moreover developing integrated BIM-BPS tools is hard to be implemented, however, this research provides innovative technologies to implement this integration, as the outcome is intended to promote and enable future users to take into consideration the environmental effect of their design and to find ways of improving it to improve sustainability. This is

accomplished through direct integration and through sending feedback to the designers about each decision made in the building model and how it can affect the environment, which is a significant scientific achievement, of 3D building models and energy evaluation data.

1.6 Reading Guide

The following chapters outline the implementation of the research model. The literature review which is organized around the environmental impact of the AEC industry and sustainable examples, the Building Information modeling (BIM) and the Building Performance Simulation tools (BPS), existed tools and how BIM and BPS can be integrated is outlined and discussed in Chapter 2. Chapter 3 outlines the Development approach, by explaining the tool's approach towards the integration as well as an overview of the Dutch energy norm (NTA 8800) has been introduced. In chapter 4, the actual tool development has been discussed, including the data inputs, energy calculations, the tool's interface, and limitations. Lastly, chapter 5 outlines conclusions and future recommendations.

Chapter 2: Literature Review

A review and evaluation of the literature and the previous studies related to the graduation project are given in this chapter. The research problem of this research concerns how the designers can be motivated by using the energy simulation tools in the early project's phase in making sustainable decisions earlier in the project. Substantial observations of the benefits and drawbacks associated with each are described by insights into the various methods followed by the research community. The literature has guided the methodology of this research and the developed tool which is based on the previous outcomes seeking for sustainable decisions and solutions.

This chapter is divided into five main parts related to the topic of this research, these parts include the environmental impact of the AEC industry, Building information modeling (BIM), Building Performance Simulation tools (BPS) including its history, existing tools, benefits, and drawbacks, the existing integrated BIM-BPS tools and lastly a detailed discussion on the Dutch energy standards.

2.1 Environmental impact of AEC industry

In this section, the impact of Architecture Engineering and the construction Industry on the environment is discussed. Examples of implementing sustainable concepts are discussed based on this impact.

2.1.1 Environmental Impact of Architectural, Engineering and construction industry

In recent years the challenge for reducing the environmental impact by the building construction field has been significantly increasing, along with their economic existence and growing quality of life. Achieving sustainable construction is important for achieving these goals, as construction plays an important role in sustainable growth. Moreover, It has been stated that buildings consume most of the unrecoverable resources and produce large amounts of waste, and buildings make up half of the total carbon dioxide (Kamar et al., 2010).

In recent years there has been a growing amount of research on sustainable building and architecture. The core principle of sustainability is to concentrate on environmental factors to create a built product with optimum internal environmental qualities so that the negative aspects of such constructions can be minimized (Zabihi and Habib, 2012).

It has been also stated that the construction industry accounts for around 50 percent of CO2 emissions. Also, it accounts for about 20-50 percent of all natural resources consumed, and 50% of all the produced solid waste, these problems takes place during the construction phase of the life cycle of the building, causing many environmental impacts (PULASKI, 2004; KHASREEN; BANFILL; MENZIES, 2009; PROBERT et al., 2010). These negative related environmental impacts underline the need for sustainable construction processes and effective ways of managing sustainability measures (TAM et al., 2006).

Many researchers have applied many strategies to enhance sustainability in the construction sector. For instance, (Kibert, 2016) has developed a model in which the principles of sustainable construction are applied to any services needed during the construction life cycle at all project phases. Based on his model, control measures can be taken to enhance the construction process and reduces the impact on the consumption of natural resources and ecological systems.

Furthermore, to assist in formulating a strategy for the planning and design of green site offices to promote green building design and construction, an assessment was provided by (Hui and Law, 2002) for the construction site offices. The study has laid out a theme for addressing and investigating green design and site office construction. It aims to obtain key information and assess practical solutions to improve the performance of the building.

2.1.2 Sustainability in construction

Sustainable building encompasses a systematic approach to restore and preserve harmony between the built and natural environments so that people can live in a sustainable environment (CIB and UNEP-IETC, 2002). To achieve better quality, efficiency and a healthy environment should be implemented, sustainable construction must be able to enhance the environmental targets and align them with the social and economic considerations (Abd Jamil and Fathi, 2016). Furthermore, sustainable measurements in construction draw attention to reductions in the use of building resources, during the construction process and the life of the buildings' operations (Ismail et al., 2017), concerning the used materials (Oke et al., 2017; Aghimien et al., 2019) and waste production (Abd Jamil and Fathi, 2016).

The term Sustainable Building was originally proposed to explain the construction industry's goals to address sustainability. Sustainability initiatives therefore also take into consideration the issues of health, quality, productivity, and minimization of waste (Hall and Purchase, 2006; Koranda et al., 2012; Abd Jamil and Fathi, 2016), in accordance with the ecological, social, and economic considerations of a construction project (Kibert, 2008; Shurrab et al., 2019).

According to Agyekum-Mensah et al. (2012), sustainability in the construction industry has been implemented over the years, from the focus on how to deal with the issue of the short supply of resources, particularly energy, to technical consequences such as materials, building components, building technologies and energy-related design concepts called 'eco-build' and 'green construction'. Also, many researchers emphasize that the most effective construction industry is directed towards achieving social, environmental, and economic success (Agyekum-Mensah et al., 2012; Ndlangamandla and Combrinck, 2019).

A successful example of implementing sustainability in the construction industry is the Infosys residential building in India, the Infosys BPO building is one of India's largest platinum-rated towers. The building was designed and built-in five domains with a comprehensive approach to achieving sustainability, these domains are sustainable development of the site, water-saving, energy efficiency, selection of materials, and environmental quality of the interior. As per the specifications of ASHRAE 90.1-2004, this building has succeeded to utilize 30 percent less energy consumption. High-performance building envelope made up of isolated walls and spectrally designed windows with a low window-to-wall ratio that minimized the average heat gain in the building and also provided a lighting design efficiency of 0.65 watts per square foot increase by 40 percent over conventional designs. Besides, the air-conditioning system is fitted with multi-stage air handling systems that work during nights and winter in free cooling, evaporative cooling, and air-conditioning mode which led to an increase of 30 percent more efficiency than a traditional system. 13% of the total used materials were recycled, 80 percent of the total material was generated locally and more than 59 percent of this content has also been collected regionally, thus reducing transportation emissions. More than 57 interconnected recharge wells were installed around the campus to collect rainwater. Low-flow dual-flush toilets, sensor-based urinals, and other water-efficient installations were provided, reducing water consumption by more than 40%. Furthermore, Sewage water has been treated

and reused for flushing and air conditioning at a state-of-the-art Membrane Bio-Reactor (MBR) facility (Smiciklas et al., Sustainable Buildings 2012).

2.2 Building Information Modeling

This section discusses the Building Information Modeling (BIM) context, and specifically on its Background, its benefit on the AEC industry. Moreover, an insight into the data model of Industry Foundation Classes (IFC) is also provided.

2.2.1 Building Information Modeling BIM Background

In the building industry, BIM is a new shift in design methodology and documentation. Building Information Modeling (BIM) has been defined by Boukara and Naamane (2015) as the generation and management of building information during its life cycle, using three-dimensional, real-time, dynamic building modeling software to reduce time consumption and resources in building design and construction. This technique yields the Building Information Modeling (also abbreviated as BIM), which includes the geometry of construction, spatial relationships, geographical descriptions, and quantities and properties of building components, including life-cycle processes of building and facility operation.

While BIM definitions vary depending on the organization or the researcher, the general concept is that BIM is a method that utilizes BIM models during the service life of a building to generate, extract and re-use digital building information, it was reported that BIM is all-construction information and a complete collection of design documents collected in an integrated database. All the data is interconnected and parametric (Schwegler & Nies, 2010).

In the early 1970s, 3D modeling started based on CAD technologies founded in many industries, and the building industry implemented 2D design to use CAD. In the early 2000s, the idea of BIM was established from field research on the development of new regulations and techniques for collaborative data exchange to improve construction-specific CAD (Penttilä, Rajala, & Freese, 2007). This begun to bring the ability of architects and engineers to add information to designed objects (in terms of properties, materials, lifecycle, and other data) into the generated functional design. Since the start of BIM, it has been a focal point for AEC throughout the building lifecycle (Ghaffarianhoseini & al., 2017).

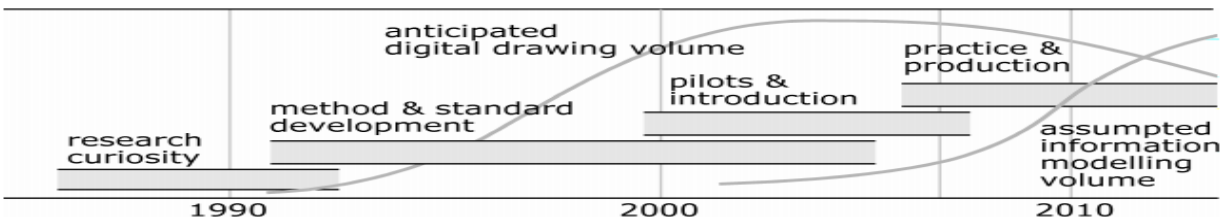

Figure 4 evaluation of building information modeling framework as a trend or phenomena within AEC-field Source: (Penttilä , Rajala and Freese, 2007)

The key difference between BIM and traditional 2D CAD is that the building in the 2D drawing of a building is essentially a depiction of the actual, built form detached into plans, sections, elevations, and often perspective views. BIM permits the building and its components to be simulated in 3D. This simulation goes beyond showing how different components of buildings can be combined within the project. Collisions are predictable and display environmental variables on various construction projects (Schwegler & Nies, 2010).

2.2.2 Main Benefits of BIM in the AEC industry and challenges

Many researchers have agreed on the numerous benefits of BIM in the AEC industry. Ghaffarianhoseini et al. (2017) has indicated many benefits of BIM in his study, for instance, he has stated the important role of BIM in offering technical advances to the traditional CAD, for instance, BIM has provided CAD with more capacity for intelligence and interoperability. Besides, BIM has improved the digital representation of traditional CAD, it has offered many functional features that allow the process of information exchange through different platforms, both within an organization and more broadly within a multidisciplinary team.

Moreover, BIM has shown many benefits in construction, for instance: It offers an integrated platform for various project participants in the lifecycle of a project. BIM models can simulate repairs or upgrading methods and thus help to decrease the cost of facility management and enhance the maintenance activities, and also provide a reliable estimation of the cost of renovation, this can help in the design and construction phases to coordinate the procurement process (Schwegler & Nies, 2010). Displaying simulation of buildings to developers can also reduce time consumption on the job site. Also, the contractor can utilize BIM to define areas of a project that conventional documents do not allow to visualize quickly. On-site, BIM can be used in breaking down the project's activities into separate phases within regular time intervals (Schwegler & Nies, 2010). Besides, BIM could be used in the implementation of the facility management; As BIM allows files' exchange which provides standardized processing of information from external facility management (Döllner and Hagedorn, 2007).

Furthermore, BIM has shown many benefits in decision-making processes, as BIM provides all the engineering participants an opportunity to use a unified shared model to optimally achieve the project objectives. The data sharing offered by BIM between different team members allows for faster assessment and control of information (Qian, 2012). Moreover, BIM provides an accurate comparison of different architectural designs, which encourages the development of more efficient, cost-effective, and sustainable solutions. BIM can also enhance the analysis and comparison of different energy performance alternative solutions for the designer to make sustainable decisions.

Researchers believe that there are many reasons behind the difficulties of adopting BIM, including costs and interoperability issues. A major concern is the lack of software interoperability and non-user-friendly format coupled with the absence of knowledge and previous experience. Many BIM users experience low added values; this discourages the

construction firms to implement the BIM techniques in the projects. Smaller firms tend to suffer the most as they are less engaged in BIM projects and thus less experienced. (Ghaffarianhoseini et. al., 2017).

2.3 Industry Foundation Classes

2.3.1 Introduction

It has been stated that the development of the IFC classes was implemented in the spring of 1993 (Bazjanac, Crawley, 1997). The development of the Industry Foundation Classes has begun as some of the major construction companies in the building industry of the United States started discussing ways of implementing modern information technology to the building industry. These groups have formed the Industry Alliance for Interoperability in the early summer of 1994 and revealed interoperability among a group of simulation tools at the AEC system shows in Atlanta Georgia in June 1995. The Alliance became a public organization, open to any member of the industry, in September 1995 and formally became a global organization in May 1996. At that point, the name was changed to the International Alliance for Interoperability (IAI) (Bazjanac, Crawley, 1997). The first version of Industry Foundation Classes was released in 1996, with version IFC 1.0, which was introduced to the market to bring a neutral model to the AEC industry. Moreover, the first version was followed by further updates on IFC 1.5, IFC 1.5.1, and IFC 2.0. In October 2000, the oldest version still in use, the main focus of IFC 2.0 was on the increase of the stability of the platform and the published information.

Besides, another version was released that demonstrated the possibility of IFC certifications, accompanied by several updates that improved the extension's capabilities and compatibility. The most common version is the 2007 2×3 version, which has implemented many performance and quality enhancements, plus bug fixes from previous versions. IFC 4.0 (originally known as IFC 2×4) was the newest release in 2013 and brought new ways of documentation, as well as support for new platforms, structures, and facilities for construction. Addendum 2, was released in July 2016, which brought changes and corrections, is the most recent edition of IFC 4.0. For future improvements, IFC 5.0 is already in its early stages of growth, promising to offer benefits to infrastructure staff (starting with alignments and then going on to highways, tunnels, bridges, and railways) and to ensure greater flexibility and capability by parameterizing the models of all disciplines. Figure 5 below shows the growth history timeline of IFC.

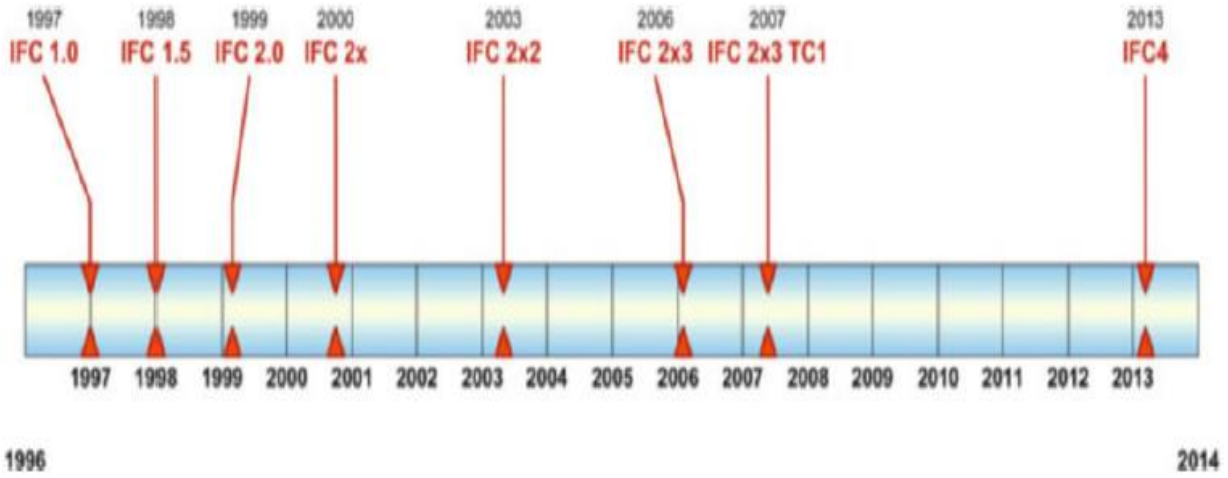

Figure 5 history of IFC (Source: Borrmann et al., 2018)

Although the development of the IFC evolved independently of the ISO standardization body and the STEP procedure, it shares much of the same underlying technology, most notably the data modeling language EXPRESS. EXPRESS is a declarative language which allows the possibility of defining object-oriented data models (Schenck and Wilson 1993). That means it follows the object-oriented principles for instance: the abstraction of objects in the real world into classes (called entities in EXPRESS) which can have attributes and be related to other classes. EXPRESS utilizes the formation of an entity type as an equivalent to classes in object-oriented theory. Moreover, EXPRESS offers the possibility to define algorithmic conditions using an optional WHERE block as a means of describing rules for data consistency. The WHERE block contains Boolean expressions that have to evaluate as true for the special instance to be valid (Borrmann, Beetz, Koch, Liebich and Muhic, 2018). An example of the most important entities of the IFC data model is shown below in figure 6.

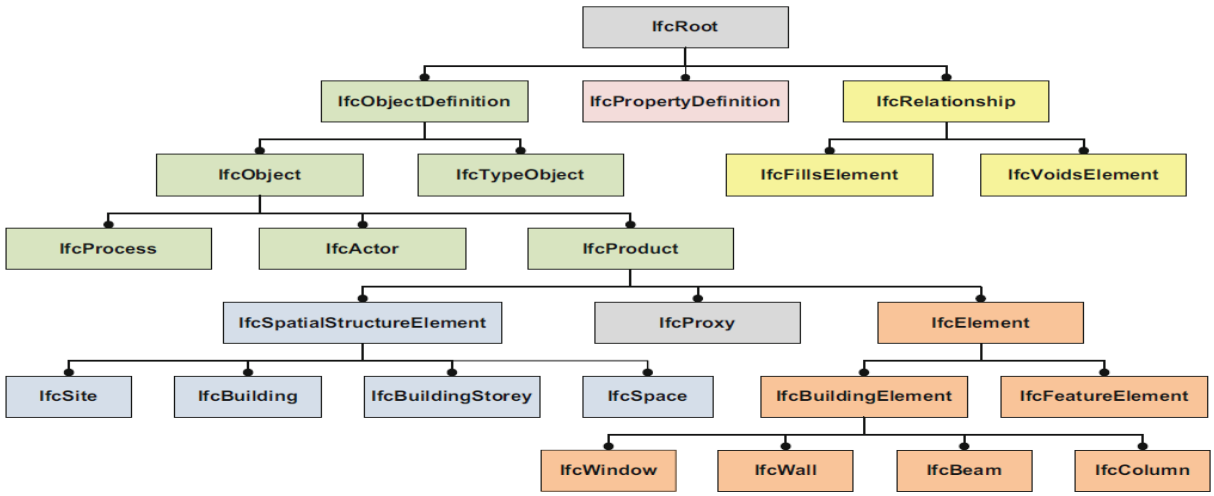

Figure 6 Part of the IFC data model showing the most important entities Source: (Borrmann, Beetz, Koch, Liebich and Muhic, 2018)

2.3.2 IFC Benefits and Drawbacks

It has been stated that IFC has shown many advantages and added value to the building industry, as IFC is a common data scheme is used for describing building information model data which aims to establish a standard method to store and represent data, so that all kinds of software can import and export building data in this format, thus enhancing the ability of data sharing among different project's participants and different software throughout the whole life cycle. IFC standards can also be used to unify the format of information generated by different types of software to realize the free conversion of building information (BuildingSMART, 2020). Many researchers have implemented the openBIM in the information exchange process, in which the most used openBIM standard is IFC; which is implemented in building life cycle, including all the project's processes (Jiang et al., 2019). Besides, Solihin et al. has tested multiple representations for 3D BIM geometry data checking for related queries, and it has been proved that IFC acted an important role in the exchange of information process (Solihin et al., 2017). In addition to the rule of IFC in the design process, many researchers have proved the role of IFC in the construction phase, Park et al. has implemented the IFC file format to record the information of the BIM model entity to the 4D BIM database, as results, Park et al. has realized the real-time construction progress information shared and visualized of the daily 4D BIM (Park et al., 2017). Moreover, many researchers have proved the role of IFC in the operation and maintenance phase, as Lu et al. has developed a semiautomatic image-driven system to build the original BIM objects in IFC from the image of the existing buildings in the operation and maintenance phase, as a result, the research was able to prove the ability to convert the identified objects to IFC BIM objects (Lu et al., 2017). Also, Hu et al. has proved the support of IFC in the collaborative management with multiple functions among the MEP project's participants in the phase of operation and management as the research has used IFC to represent and exchange information among different BIM applications (Hu et al., 2016). Moreover, many researchers have proved the role of IFC in the building lifecycle besides its importance in the design, construction, and maintenance and operation phase. Vanlande et al. has implemented IFC as a model to define the elements and relations of the construction projects, hence easing the information exchange process in the building project's lifecycle (Vanlande et al., 2008), the role of IFC in the information exchange process is explained below in figure 7.

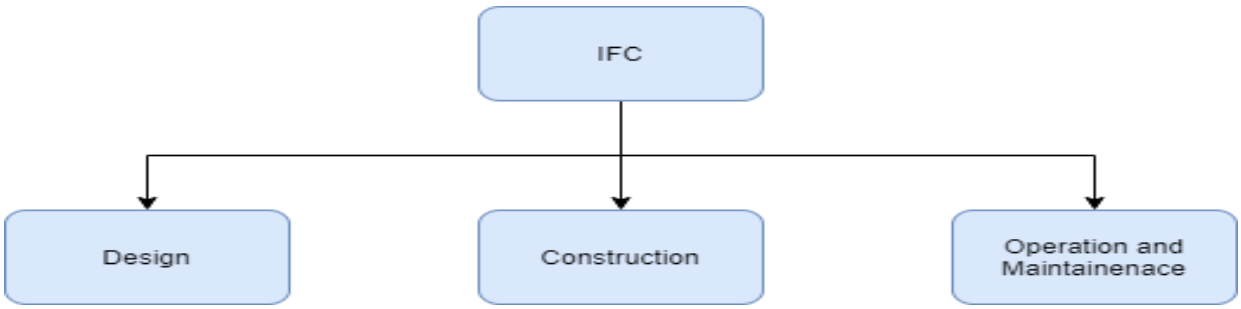

Figure 7 the role of IFC in Information exchange

Despite the benefits of IFC, it has been stated that IFC specification has many limitations and drawbacks. For instances, firstly, exporting an IFC from a modeling software such as Revit to import it in another software; has been always reported the loss of model information, hence it has been always recommended that tests should always be carried out to ensure that the IFC can be used in each specific case (BIM Experts, 2020). Moreover, sending and receiving a building model from other participants, many errors could still happen these errors could be syntactic, semantic and design errors, even if the project's participants applied the neutral format (Jiang et, al., 2019).

Besides, it has been reported that software vendors and end-users should not specify the requirements for exchanges made from their domain or manually validate IFC files against the validation platform, because the IFC schema must comply with the EXPRESS language specifications, if the IFC schema does not comply with the schema definitions, the file will not be correctly parsed and read. In addition, the IFC file must match the types and relationships of the IFC project's entities and attributes, because semantic problems may occur which may lead to translation errors and other primary defects if the IFC does not meet the specified specifications (Jiang et, al., 2019).

In addition to the importing drawbacks, the exchange of information between participants is also a stated problem; as the software vendors encode the design information into the IFC formats to transfer the required information, thus the participant's information exchange must comply with the needed conditions and rules, hence, according to the exchange information rule, the exchange of BIM data requires validation of semantic and syntactic compliance with the IFC file, otherwise, errors will take place (Jiang et, al., 2019).

2.4 Building Performance Simulation

This section provides an insight into the building performance simulation, by giving an introduction to BPS's background, history, advantages, and existed BPS tools. Despite the BPS tools' benefits limitations have been also mentioned in this section.

2.4.1 Introduction

Building Performance simulation of buildings is a vital method for enhancing both architectural design quality and efficiency. Moreover, the use of BPS tools is important to achieve an optimized and efficient design for architects. BPS is a technology with major implications that enables the competing cost and performance attributes of a proposed design to be quantified and compared in a logical manner and at relatively low cost and effort. Moreover, it has been stated that it may be the only method to trace operational robustness at the design stage. Also, simulation offers a way of comparing calculated output with design purpose during the

operating process, checking system for deployment and operating faults, and deduction of successful control sequences (Clarke, 2015).

Moreover, Hensen and Lamberts have described the building performance simulation as a Computational building performance modeling. Moreover, it assumes dynamic boundary conditions and is usually based on numerical methods aimed at providing an effective solution to a specific complex model in the real world (Hensen and Lamberts, 2012).

Besides, Building Performance Simulation (BPS) is also defined as Building Simulation, Building Energy Modeling, or Energy Simulation – has performed an important role in the design and operation in reducing the energy consumption, high-performance buildings, and policy formulation driving the achievement of the above-mentioned low energy consumption objectives (Hong et. al, 2018). BPS is defined as the use of mathematical computational models to describe a building's physical characteristics and to monitor energy performance strategies (Hensen and Lamberts, 2012).

2.4.2 History

Detecting the building's performance simulation using digital computers has been an active field of research since the 1960s and became widely known in the 1970s (Hong et al., 2000). Building performance simulation tools have been developed in the late 70s and continued through the 80s, efforts have been made to validate and test different methods for simulation tool codes (Augenbroe et al., 2004). Technical researchers and building scientists developed simulation tools targeting to solve the problems of engineers (Attia,2010). It was until the 1990s that the building simulation discipline achieved a certain maturation stage, offering a variety of Building performance simulation tools (Hensen, Lamberts, et al . 2002).

The growing global concern for the conservation of the environment was seen at the start of the 1990s. Global warming and thinning of the protective ozone layer are blamed for wasteful use of fossil fuels and the use of toxic material, since the situation changed then a growing global concern for environmental protection was observed in the early 1990s. The challenge facing experts in the building industry at that time is to build a safer and more sustainable built environment with better use of resources and minimized adverse impacts on the environment, the challenge for professionals in the construction industry was to build a safe and pleasant built environment with lower energy usage and reduced negative effects on the environment. The demand for 'green' buildings has made it a must, rather than a need, to apply building simulation hence, BPS tools have gained acceptance (Hong et al., 2000).

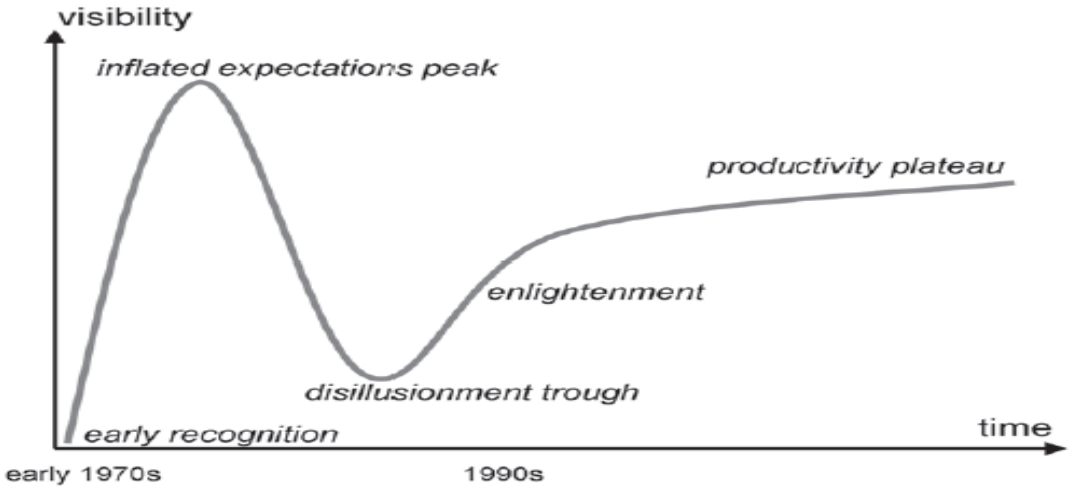

Figure 8 hype cycle for building performance simulation (source: Hensen and Lamberts, 2012)

2.4.3 Advantages

The BPS simulation has been widely used in the building sector. Many researchers have stated and proved the advantages of BPS in many stages during the design projects. Firstly, it has been stated that BPS has shown an effective impact on Modeling building operations, controls, and retrofits. For instance, a big amount of energy of a building is consumed during its operating period; thus, it is important that during this process building simulation techniques be implemented to analyze and understand the impact of energy-saving technologies. For energy retrofit projects, comprehensive energy models are generated using BPS tools that can be used to analyze the energy conservation measures (Hong et. al, 2018). Besides, BPS has shown a positive impact in controlling and monitoring the building consumption, as it is used to forecast thermal loads in buildings and provide feedback on energy consumption and the optimal control strategies to reach comfort (Hong et. al, 2018).

Moreover, the use of building performance simulation has several long-term benefits, as the use of BPS decreases the environmental, operating, and maintenance costs and increases energy efficiency in the building sector. As well as figure 9 below shows that BPS provides a quantitative evaluation of potential designs of different degrees of complexity to assist the designer in the decision-making process. Lastly, Building Performance Simulation could be used to design the building according to local building regulations, codes, or standards and subsequently support energy monitoring to test the energy efficiency of the building as built (Hong, Chou, and Bong, 1999).

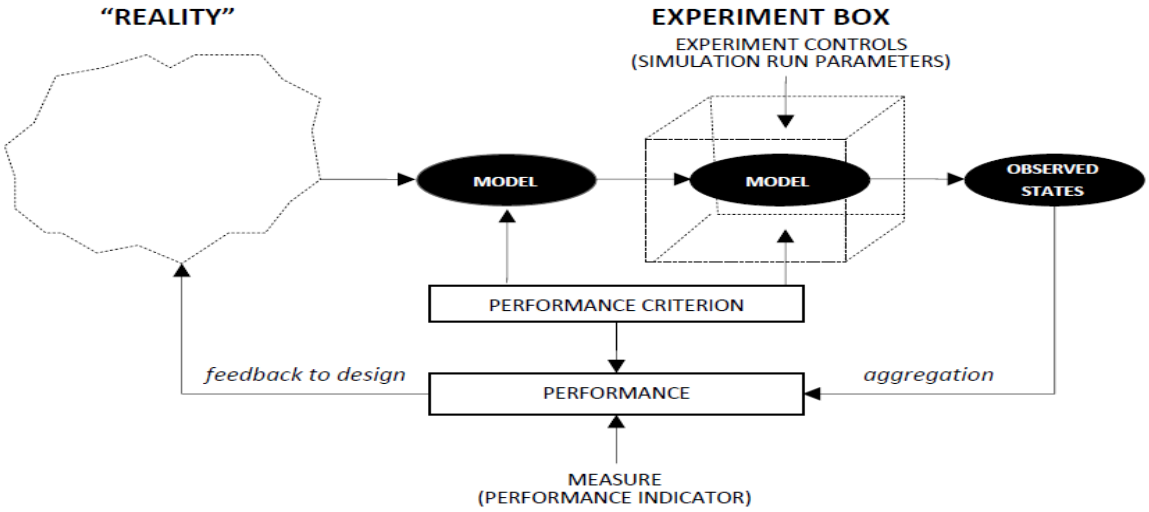

Figure 9 the role of BPS in the decision making process, Source: (Hensen and Lambert, 2002)

2.4.4 Existed tools

Most building simulation tools are composed of two separate elements, the engine, and the user graphical interface. Although academic or research institutions are typically the developers of these simulation engines, the user interfaces are implemented by private software vendors. The simulation engine uses an input file in a specified format, which includes a data description (Maile, Fischer, and Bazjanac, 2007). A simulation-based on this input is performed by the engine. While the output files include simulation results, they also contain information such as warning messages or additional input evaluation details about the simulation run itself.

In the industry, there are different numbers of BPS tools, one of these BPS tools is BLAST; it predicts the energy usage efficiency and cost and the energy output in buildings. Three major sub-programs are included in BLAST: space load estimation, air systems simulation, and central plant. The hourly space loads are being calculated in the Space Loads Prediction process. Besides, BLAST can be used to determine the energy efficiency of new or retrofit options (Crawley, Hand, Kummert, and Griffith) for building design. BLAST incorporates algorithms for predicting the usage of energy and the performance and cost of energy systems in buildings (BLAST, 2003). The heat balance scheme is used in actual thermodynamic equations in BLAST, which provides better and more detailed results.

Another tool is DeST, (Crawley, Hand, Kummert, and Griffith, 2007) claimed in their paper that it allows for a detailed study of the construction of thermal processes and the efficiency of HVAC systems. DeST includes many different functions in a separate management module: Medpha (weather data), VentPlus (natural ventilation), Bshadow (external shading), Lighting (lighting). It is possible to apply this BPS tool to complex buildings with up to 1000 spaces. It performs hourly indoor air temperature and heating load measurements.

In addition, one of the most widely used engines for thermal simulation is the DOE-2.1E engine. The Lawrence Berkeley National Laboratory has developed the DOE-2.1E engine. The DOE-2 engine can simulate the thermal behavior of spaces in a building where heat loads such as solar gain, equipment loads, people loads, lighting loads, and air conditioning systems can be modeled and simulated with the engine (Maile, Fischer and Bazjanac, 2007). DOE-2.1E also predicts the hourly energy use and expense of a building, including the hourly weather information, a geometric and HVAC overview of the building, and utility rate structure (Crawley, Hand, Kummert, and Griffith, 2007).

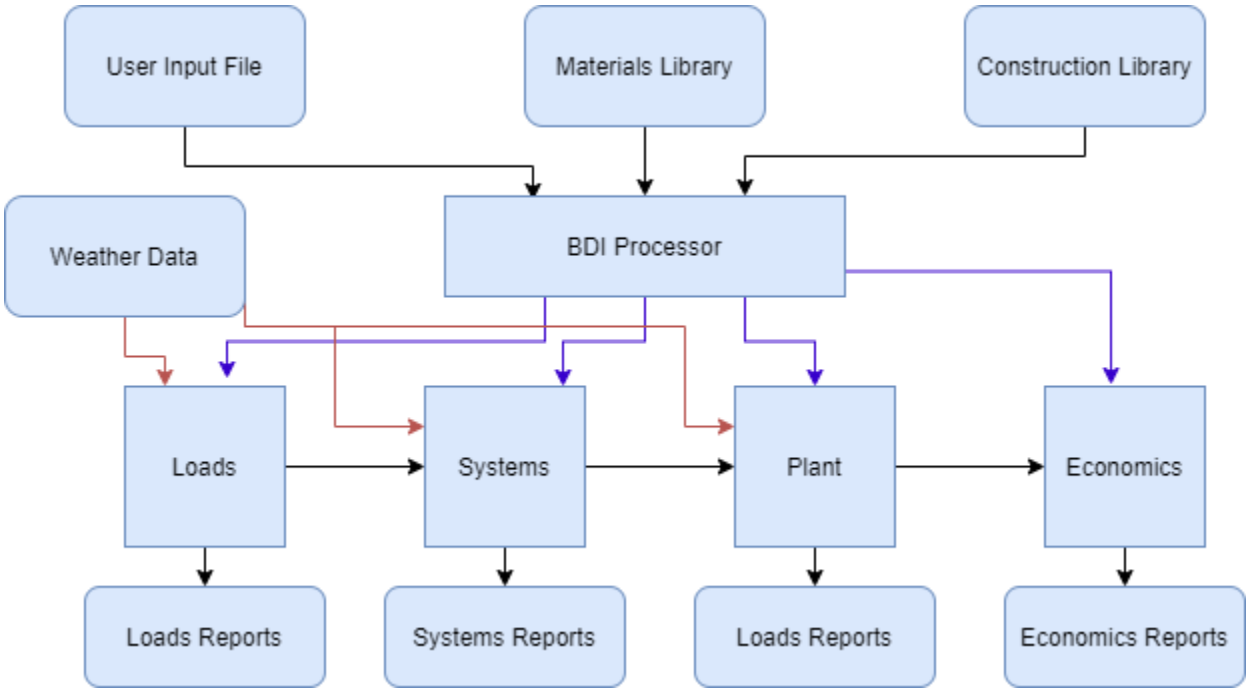

Figure 10 The DOE-2.1E engine's flow (source: Birdsall et al. 1990)

Besides, EnergyPlus (Version 2.1) is a simulation tool that uses the various advantages of DOE-2 and BLAST tools. EnergyPlus relies on the load and system simulation methodology, resulting in more reliable measurements of space temperatures and, thus, a better evaluation of the various relevant parameters. In the load calculation method, EnergyPlus is based on ASHRAE's heat-balanced approach. Simulation with EnergyPlus requires text file input, and this method consumes more effort compared to graphical user interface engines to sort out the input data (EnergyPlus, 2007).

Lastly, James Hirsch created eQUEST (Version 3.60) as an open server tool, eQUEST can measure and quantify the building's energy efficiency. A sustainable building model that represents the demands of the energy analysis of the building could be created by eQuest, it also enables alternative designs to be compared based on a building parameter sensitivity

analysis and provides a concise and effective description of the impact of parameter changes on energy consumption and comfort.

2.4.5 BPS tools' Limitations

Since a large number of software tools are available to predict the energy and efficiency of buildings, number of questions were arose by many researchers, which is due to the regular use of these simulation tools as the activities and method involved in designing a simulation model using these BPS tools.

While these BPS tools have ongoing processes of creation and extensions of modeling capabilities, their core ideas and basic software design do not change (Loonen et al., 2017). Also, during simulation run-time, the building structure and material properties in such software are usually unchangeable, restricting the modeling choices for dynamic building structure systems.

Besides, most modern BPS tools have a graphical user interface (GUI) as a front end to interact with the users. However, (Attia et al. 2012) suggested that the users have limited flexibility to improve usability through the non-modifiable user interface to model dynamic building envelopes and limited access to the programming language of the simulation tool. This is particularly the case for simulation tools that are targeted towards the needs of architects.

In addition, the BPS tools normally use constant coefficients at the beginning of the simulation and determined only once for each aspect of the building components, hence these tools don't involve different choices in the characteristics of thermophysical materials, this aspect leads to considering only heat power and thermal conductivity properties of the materials and ignore the other thermophysical properties (C.M.A.D, 2003).

Furthermore, (Punjabi, 2005) noted that most of the widely available advanced simulation programs do not provide the user with sufficient assistance in making decisions. By using the building simulation tools during the design process, however, the situation becomes much more complicated. Since the designers need various kinds of data at the different design levels, the practitioners of the project have different backgrounds.

Furthermore, most of the simulation instruments available were originally developed by researchers for experimental purposes and were often noted for their complexity (Attia, 2011). Since these tools involve detailed data from specific buildings, these data are usually stored in a specific language and structures. Besides, since the output data is primarily numerical, the functionality of these tools is typically complicated, and the input data involves mechanical engineering data that is applied in the later design phase of the architectural design process, making it difficult for architects to incorporate the energy analysis results throughout the design process.

In addition, (Ellis and Mathews, 2002) reported that most of the BPS methods which are commonly used are considered to be time-consuming or difficult to implement general design problems. Thus, overlap with the design process itself, impacting the design and consuming a lot of time and energy that designers would prefer to spend on the design instead. In addition, the latest tools do not always fit into the design of the architectural model well (Punjabi, 2005). These tools can aid architects dramatically in the design of energy-efficient buildings; they need detailed design information, however. These instruments are also usually intended to be incorporated in the later phase of the design since the design model must have been finalized by the architect to use the method.

Most of the existing modeling methods are unable to provide adequate input on the capacity or comfort of passive and active design and technology to meet such environmental conditions (Crawley et al . 2008). Many researchers stated that the existing tools are ineffective to design high-energy buildings during the early stages and are not appropriate for architects and during early design phases, no existing tool discusses optimized design for architects for constructing a high-performance building (Attia, 2011).

2.5 BIM-BPS Integration

This section covers the implementation of BIM and BPS technique, as previously mentioned both BIM and BPS offers individually an approach towards sustainability, however, BIM cannot provide the project's designers with feedbacks, however, the implementation of BIM and BPS can provide the designers with better options in the decision making process. In this section, the benefits of integrating BIM and BPS will be discussed as well as various examples of BIM-BPS implementation techniques will be covered.

2.5.1 BIM-BPS integration benefits

The integration of BIM-BPS tools has shown many benefits towards sustainability and the assistance of designers in the decision making processes. As data is automatically collected, the time consumption in the construction of the 3D model and the development of a BEP analysis is significantly reduced. The integration of data between BIM and BPS tools also reduces the probability of human errors. Also, the quick and precise reconstruction of the 3D model in energy tools enables BIM design alternatives to be evaluated.

Moreover, BIM-BPS integration allows the possibility for creating building energy models that prevent excessive processes, for instance re-entering building data from the design model. Hence BIM-BPS integrated tools ease and facilitate the creation of the energy models as the traditional process of the creation of the energy models is time-consuming and may lead to many design errors (Jeong & Kim, 2016).

Besides, another issue has been stated in the literature review which is the collaboration issues between project's participants, as the construction project involves several engineers with different backgrounds, which usually causes misrepresentation and communication between the project's participants, according to Negendahl (2015), the integration of BIM-BPS approach will solve the conventional collaboration methods within the project's participants, as the early involvement of BIM-BPS tools in the design phases will facilitate the design process and assist the architects in designing building models compiling with the energy requirements earlier in the design phase, hence saving time and overcome the design errors in the designed models.

2.5.2 BIM-BPS integration examples

Many researchers have implemented the BIM-BPS technique in their researches, P.M Pelken et al. have developed a software interface, the introduced tool for the building design and its energy efficiency has been implemented, this tool supports numerous operator interactions and the whole lifecycle design. Many researchers have built a 3D matrix based on this tool, including multi-design teams, phases, and variables, and have applied the tool in the entire design process of the building, resulting in the implementation of the simulation from the point of view of the user.

Moreover, S.Attia et al. has built a decision support method aimed at simulating a zero-energy building, which is only usable for tropical climates in the early design phase of the project. Quick feedback can be given in this method to assist decision-making as it is based on an integrated model and database of standards and many building energy data can be entered as default values depending on the environment of the site chosen. Besides, this technique is limited to its own rectangular standardized single-zone library, with few input building alternatives. Users will get a selection of alternatives in a short time after a sensitivity study of the technique.

In addition, E. Ochoa and I.G have implemented and integrated tool "NewFacades", this tool was technically designed to be implemented in the early design phases to provide alternative smart façade combinations, the methodology provided in their research was implemented via a list of text inputs, including, for example, the concepts of architecture, location, orientation, scale, and setting, this tool forms a model of building geometry. By integrating EnergyPlus, the energy modeling engine, NewFacades provides detailed and comparative evaluations of energy and visual comfort for each alternative. In addition, since the tool focuses on the early phase of the project, the performance of NewFacades will typically be used in an EnergyPlus-compatible format in more advanced design stages.

Furthermore, a tool for a semi-automated IFC-BEP simulation was also implemented by (Bazjanac,2008), the fundamental objective of the integrated technique is to remove all unnecessary human intervention that results in subjective and arbitrary decisions affecting BEP simulation. This is achieved by automating every part of the automated process of modeling and simulation of energy efficiency. The tool reads an IFC file, extracts geometry construction

and related data, and transforms extracted data as required to meet the geometry input data requirements of EnergyPlus.

Lastly, an integrated tool has been developed by (Petersen and S. Svendsen, 2010) in the early design phase of the project which enhanced the decision making process earlier in the project, this tool is an iDbuild-based method of informed decision-making. In two stages, “parameter variants” and “informed concept proposal,” the suggested extension of the construction design workflow extended sequentially. One case study illustrated the method’s ability to evaluate parameter variations. Therefore, iDbuild can be used for parameter variations, offering an overview of how performance-deciding parameters affect performance requirements.

2.6 Energy Performance Standards in the Netherlands

In this section, an overview of the Dutch energy standards for residential buildings will be discussed. The topics which are involved in this section are an introduction to the Dutch energy standards and an overview of the energy efficiency measures applied in the Netherlands.

2.6.1 Introduction

Since 2008, the EPC system has been introduced in the Netherlands and over 3.5 million EPCs (> 50 percent of the total building stock) have been registered so far. The government agreed in December 2012 to introduce a new, much more consumer-friendly scheme for residential owners. The new system was established in 2013 and 2014 and has been in operation since January 2015. The new labeling scheme law became effective on 1 January 2015. Changes in the certification scheme for experts were also introduced in January 2015 for the new EPC for residential buildings (Eck, 2018).

In September 2013, more than 40 market participants and other stakeholders signed a national Energy Agreement (‘Energie Akkoord’). In this agreement, the energy quality improvement goals and the usage of RES in buildings are under the criteria of the EPBD. This agreement has implemented various measures:

a) Upgrade 300,000 current homes to two energy efficiency groups on the energy label;
b) Redeveloping the building stock of social housing to energy class B standard (on average);
c) Boost 80% to a minimum of energy class C of private rental houses.

As a result of so-called “tightening experiments,” the transition towards more demanding criteria took place. These studies involved a market growth study of energy efficiency measures, solutions for renewable energy, and energy-efficient generators for heating and cooling. The cost-effectiveness of these interventions and their effects on the indoor environment and the satisfaction of occupants have also been taken into account. The tightening studies were published by consulting firms and were supervised on behalf of the Ministry of the Interior (BZK) by the Dutch Agency for Enterprises (RVO). Participants were

aware of the findings during the studies and were able to comment on them to ensure that a realistic experience with energy-saving measures was taken into account (Eck, 2018). The figure below, figure 11 indicates the overall improvements in the energetic quality of buildings over time.

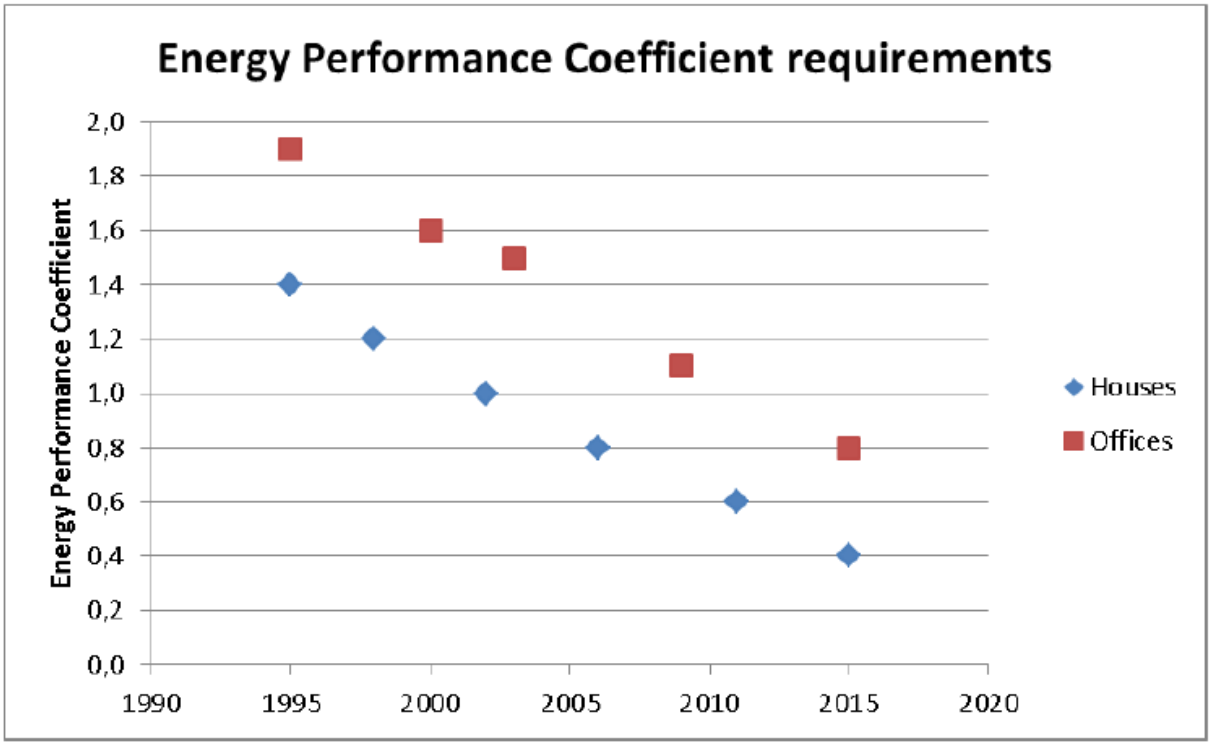

Figure 11 the energetic quality improvement of Dutch buildings over time, source: (Eck, 2018)

2.6.2 Energy efficiency measures regarding the residential buildings

The energy performance coefficient ('energieprestatiecoefficient' in Dutch), which sets minimum energy performance (MEP) for new buildings, is the key criterion for the energy performance of new buildings. This indicator is focused on a set of metrics based on the approximate overall primary energy usage of a building, e.g. heating, ventilation, and lighting, optimized for the useful floor area and the renewable energy emitted by the building.

For all new buildings and for major renovations of houses and offices, the measurement of the energy efficiency coefficient is necessary as the measurement of the energy efficiency coefficient is part of the application for building approval. To obtain a construction permit for a new building or a major renovation, a project developer needs to show complete compliance with the energy efficiency requirements. Permits are reviewed and given before building by local governments (Eck, 2018).

Moreover, during construction, municipalities are responsible for checking the compliance of the construction process with the energy requirements. They issue a "cease-work" order in the building that doesn't meet the energy requirements, which remains effective until the conditions are met. Buildings that do not comply do not get installed, and the building is stopped until it complies with the energy permit if builders deviate (Eck, 2018).

Furthermore, a sample is drawn each year by the RVO to verify if all permits are in line with the legal requirements. The RVO submits these samples to the municipalities that will take legal action if the permits are not in accordance with the legal requirements.

On 1 January 2015, the coefficient of energy efficiency was tightened as an intermediate step towards achieving the NZEB standard. The next step will be to determine the primary energy consumption criteria and the share of renewables up to the NZEB standard. The first draft of these criteria for new buildings was shared with stakeholders in March 2015 and submitted to Parliament for approval in July 2015 (Eck, 2018).

In the Netherlands, the triple NZEB specifications are referred to as Trias Energetica and are graphically displayed as shown in figure 12 below.

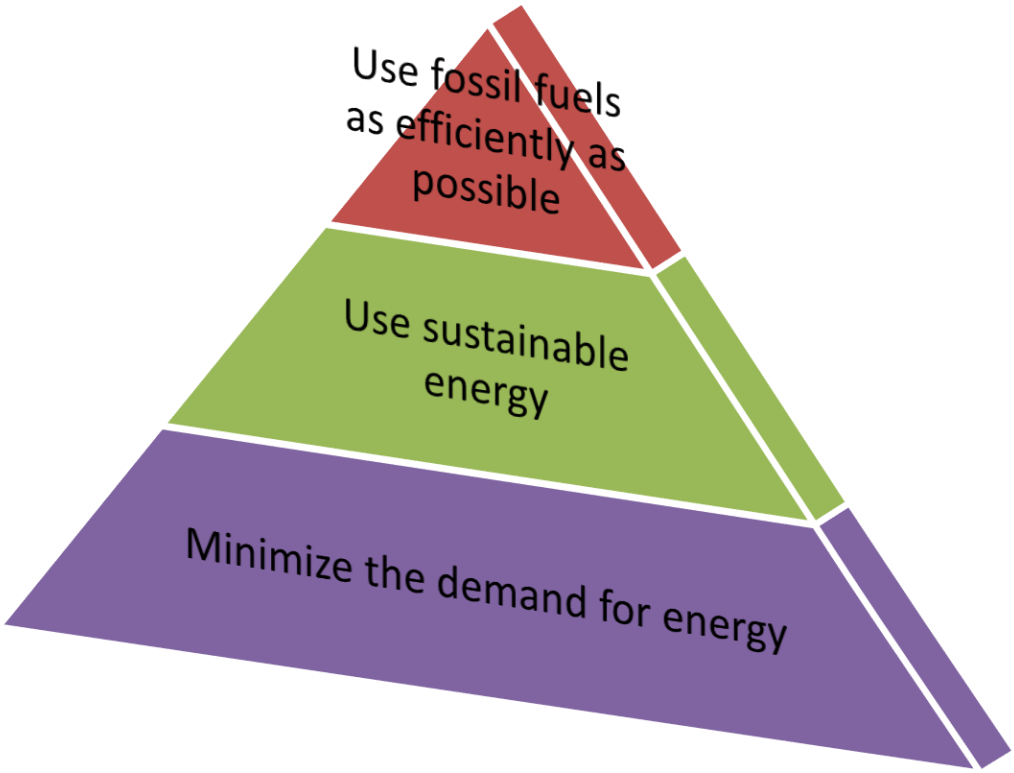

Figure 12 Trias Energetica, source: (Eck, 2018)

2.7 Overview on the literature review

In this section, an overview on the previously discussed literature review will be covered. The literature review has covered various topics, starting with the role of the AEC industry towards the environment, as well as the role of construction projects to positively impact the environment.

Besides, the literature review has proved the benefits of BIM and IFC towards the construction industry, as BIM has shown an effective impact on solving designing issues by producing semantic knowledge models that enhance the project's design, planning, operation, and maintenance. Moreover, BIM can provide the BPS tools with semantic data which will improve the design and assist the project's participants with the decision-making process. The main reason for BIM being effective on the BPS tools is the IFC Formats, as IFC is a data framework which is used to define building information model data that seeks to create a standard data storage and representation method such that all types of software can import and export building data in this format, thus improving the ability to exchange data between participants of different projects and various software over the entire life cycle.

In addition, BIM has demonstrated many advantages in decision-making processes, as BIM offers an incentive for all project members to use a single shared system to achieve the project goals optimally. The information-sharing provided by BIM between various team members allows for faster information assessment and control (Qian, 2012). In addition, BIM offers an accurate comparison of various architectural designs, promoting the creation of solutions that are more functional, cost-effective, and sustainable.

However, despite the advantages that BIM has shown, many researchers think that there are many factors behind the difficulties with BIM adoption, including problems with costs and interoperability. The lack of software interoperability and non-user-friendly format, combined with the lack of expertise and prior experience, is a major concern. Many BIM users experience low added values, which discourages construction companies from incorporating BIM methods in projects (Ghaffarianhoseini et. al., 2017).

Moreover, the literature review has a range of long-term advantages to the use of building performance simulation tools, as the use of BPS lowers environmental, operational, and maintenance costs and improves energy efficiency in the building sector. Also, it has the ability to assist the designer in the decision-making process by offering a quantitative assessment of possible designs of varying degrees of complexity.

Despite the benefits that BPS tools provide to the construction industry, (Attia et al. 2012) indicated that users have limited flexibility to enhance usability by modeling dynamic building envelopes through the non-modifiable user interface and limited access to the simulation tools. Besides, in making choices, the commonly available advanced simulation programs do not provide the user with adequate assistance. The situation becomes even more complicated by

using the simulation of the building during the design process, however. Since designers need various kinds of data at different levels of design as there are a different number of project participants with different backgrounds.

Furthermore, the literature review has shown the benefits of BIM-BPS integration and provided examples of how many researchers have used BIM and BPS to increase the sustainability of the designing process. As BIM and BPS integration allows the possibility of designing energy models for buildings that avoid unnecessary procedures, such as re-entering building data from the design model. Therefore, integrated BIM-BPS models enable and facilitate the development of energy models, as the traditional method of creating energy models takes time and can lead to many design errors.

Lastly, the energy standards regarding the residential buildings in the Netherlands have been introduced in the literature review, as the literature review revealed that the Dutch government is taking effective measures towards increasing the energy efficiency in the buildings, as the government developed a scheme to improve decrease the energy consumption in the residential buildings which led to the upgrade of a large number of homes regarding the energy labeling, redeveloping the social housing to energy class B standard and lastly boosting the private rental houses to a minimum of energy class C. Based on the energy measures which have been taken by the Dutch government, effective outcomes have been introduced to the building sector in the Netherlands.

Chapter 3: Development approach

Insights into the integration of Building Information Modeling (BIM) and Building Efficiency Simulation (BPS) were provided by the functionally integrated instruments presented in the Literature Review. The theoretical approach to the development of the instrument was built based on this integration.

This chapter represents the following steps. Firstly an overview on how BIM and BPS are integrated through the tool (3.1), followed by an overview of the Dutch energy norm NTA 8800 (3.2), and lastly a conclusion on the tool development approach (3.3).

3.1 The tool's approach towards BIM-BPS integration

As previously mentioned this thesis seeks to develop an integrated tool to promote sustainability in construction in the early design phases of the project, this aim should be achieved by effectively supporting the decision-making during the early design process with regard to reducing the energy demand of a building, as the early design stage of the project defines the degree of energy consumption of the building.

Based on the literature review, it has been stated that the integration BIM-BPS approach has shown effective results in achieving this goal. The Building performance simulation tool will track the IFC model to achieve this goal by performing energy analysis to the 3D model and send feedbacks with the results, and this process should be able to guide the project's designers to make sustainable decisions earlier in the project phase.

In the literature review, it has been seen that the BIM model is always used as input data for the BPS tools, however, it has been also proved that BIM models and BPS tool interaction could be established. This interaction could be developed as the 3D models will supply the BPS tools with the required information, hence the energy analysis could be established easily in the early phases, providing accurate, sustainable design models, preventing time consumption, and avoid design errors.

The applied approach throughout this graduation thesis is stated below in figure 13, the tool will be designed on the basis of the determination methods defined by the Dutch energy performance standards for buildings (NTA 8800), with this set of data, the tool will be able to extract and use IFC data to generate a BEP analysis. The energy efficiency is demonstrated to the architect, and feedback based on the built model will be offered, hence an optimized design model will be demonstrated.

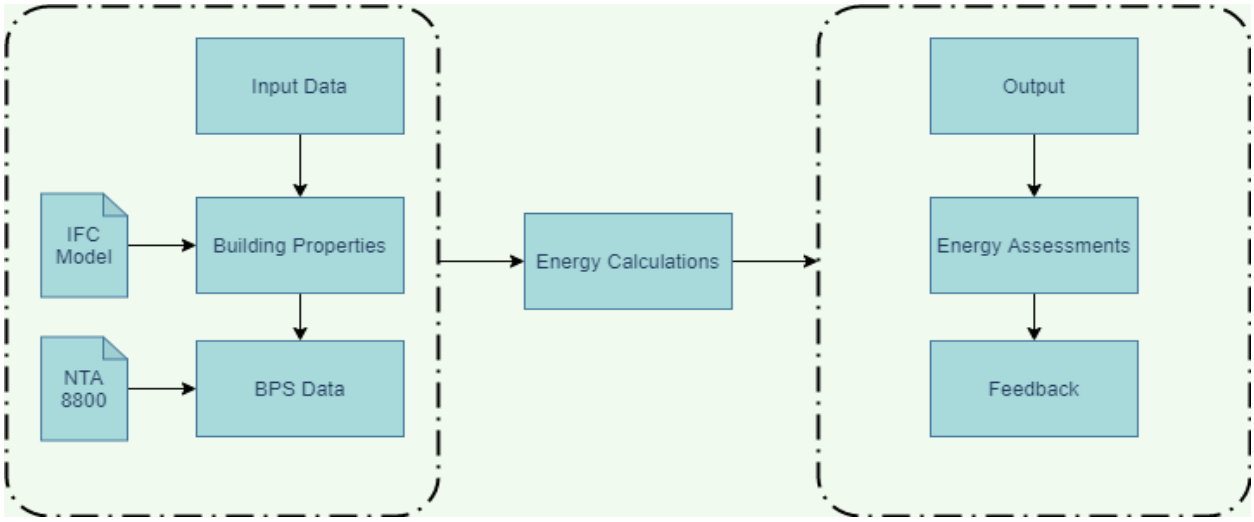

Figure 13 BIM-BPS integrated tool interpretation

3.2 Dutch energy normative NTA 8800

3.2.1 Goal

NTA 8800 aims to be a transparent and policy-free determination method for the energy performance of buildings, based on the EU's EPBD (Energy Performance of Buildings Directive). The NTA 8800 determination method has been developed within preconditions so that it can be used for setting energy performance requirements in Dutch building regulations. The design of the tool needs a reliable source of Energy Performance Indicator standards and values. These norms and values can be extracted from Dutch Standard NEN 8800, which lays down the maximum allowable coefficient of energy efficiency (EPC) for buildings. The system used aligns with the Building Regulations requirements and represents the Dutch energy performance of buildings.

3.2.2 Scope of the determination method

NTA 8800 is intended for determining the numerical values of the energy performance of buildings, as referred to in Dutch regulations. This also includes caravans and floating structures, insofar as (building) regulations set requirements for energy performance. The method describes requirements for new buildings to be built for all user functions and makes agreements with the market about existing buildings with regard to the energy performance to be realized and to which the energy label obligation applies.
NTA 8800 uses the standards developed within the framework of the EPBD and published by CEN and NEN. NTA 8800 is the implementation of these standards in Dutch building regulations. For the sake of the readability and usability of NTA 8800, the number of normative references is limited, so that NTA 8800 becomes independently readable and usable for the market as much as possible.

NTA 8800 includes the calculation rules for determining the relevant variables, such as the calculation of the energy performance indicators. NTA 8800 concerns a determination method for building-related energy use, as stated in the EPBD 2010/31 / EU + 2018/844. To determine energy requirements, a distinction is made in the building regulations by the function of use. The determination method is suitable for all user functions, in both new and existing buildings, for which an energy performance requirement is set in building regulations, or for which there is a requirement to demonstrate the energy performance, such as for an energy label.

Moreover, NTA 8800 provides a determination method for the following quantities, in accordance with the policy-based definition:

a) Energy requirement in kWh / m2.
b) Primary fossil energy use in kWh / m2.
c) Percentage of renewable energy in %.
d) Net heat demand of a building in kWh / m2.

e) Risk of too high indoor temperature during summer.

3.2.3 Determination method

NTA 8800 provides one determination method. This is a method with a fixed degree of accuracy (detail), intended for all calculations of new and existing buildings. The number of input parameters of the detail level is not limited to absolute numbers. The number of input parameters is the same for new and existing buildings, based on the 'as-built' assessment of new buildings. NTA 8800 provides fixed values that are intended, among other things, for situations in which certain input parameters cannot be determined or can only be determined with very time-consuming or destructive actions, such as with existing buildings.
In NTA 8800, the energy performance of a building is calculated on the basis of the monthly method, for instance, the monthly average value of parameters is determined or used. The effects of the dynamic behavior of building parts, installations, and installation components and the climate are included in these averages by means of utilization factors. Moreover, fixed values may not be adjusted, but may be replaced by a value derived from a quality declaration for an applied product, material, or system.

3.3 Overview over the tool development

The main objective of the research is to develop a tool that can be effectively used in the early stage of the design phase. The calculation approach is based on the calculation methods stated in the Dutch energy normative NTA 8800, the information flow process is stored in the developed tool based on the flow of BIM-BPS information from the data found in the BIM model and the NTA 8800 Dutch standard database to provide accurate and efficient results to the consumer to get insight into the energy analysis of the building components, these results will be sent to the user as feedback from the tool. Moreover, the tool will be provided with what-if options, to assist the user by providing him/her with extra information on which values should be modified to get accurate results.

Input includes data collection from two input sources: the BIM model and the NTA 8800 Dutch energy standard database. The quantity and dimensions of the building elements of the 3D model as well as the classification codes of individual building components are included in the BIM model input. Information is accessed via IFC, the international standard file format for carrying and delivering building-related data between different software applications, to leverage the building information nested in the BIM model and to be relatively independent of vendor-specific software.

The building model was specifically designed in Autodesk Revit 2019-generated IFC extracts, one of the key design tools used in the early design phases. Inputs from the EI and energy efficiency measures are extracted related to the Dutch energy normative NTA 8800. The BIM-BPS integrated tool utilizes NTA 8800 data since this is the required database of thermal energy

performance output to be used to generate energy assessments. Since normally the BIM model may not contain all the necessary data in most current architectural designs, particularly in the early design phases, some data were manually stored in the IFC model to analyze the energy performance in the developed tool.

Moreover, the research includes the estimation, following the determination method approach of environmental performance measures stated in the Dutch energy normative.
Finally, results include displaying the measured energy performance metrics in the form of a detailed tabular, displayed on an image, and feedback on the results. These numerous outputs provide the participants in the project with the ability to make more informative decisions towards sustainable design and optimized energy efficiency.

Chapter 4: tool development

This chapter outlines the criteria of the developed tool. The tool approach aims to provide a method that offers information and formulas about how to implement a model of sustainable design that meets the energy labeling criteria. The tool will be built on the basis of the methods of determination established by the Dutch Building Energy Efficiency Standards (NTA 8800). The tool will be able to extract and use IFC data to produce a BEP analysis using this collection of data.

The energy efficiency is demonstrated to the architect, and feedback based on the built model will be offered. The aim of the established tool is to effectively support decision-making during the early design process with regard to reducing the energy demand of a building, as the early design stage of the project defines the degree of energy consumption of the building. The design can be changed after this stage, without producing any remarkable results in energy optimization. Therefore, it is considered important that these decisions should be endorsed and tested in the early design phase so that optimal sustainable solutions can be applied. The following chapter is structured for the conduct of a systematic review. The activities the tool engages in are defined in detail in this section. It further explains and describes the design of the designed tool and its functionality.

4.1 Tool Methodology

The developed tool's functionality offers three main activities; these activities are offered in sequence as follow:

a) Creating building characteristics
b) Performing the energy assessments
c) Analyzing the results
d) Displaying the results associated with feedback

4.1.1 BPMN Process Map

Figure 14 explains the functionality of the developed tool in the early design phase of the project. The BPMN process map is divided into three lanes, one actor (Architect) and the developed tool. As shown in the BPMN process map, the architect is involved in two sets of activities; the first set of activities is designing the 3D model, checking the design errors, setting IFC export options, exporting the IFC model, and importing it into the newly developed tool. The type and the documentation of each activity are given in the Process Map (Appendix III).

These steps should be applied in a sequential process, firstly the Architect has to design the 3D model and check for the design errors, then the Architect has to export it as an IFC file, after modifying some options to the IFC option file to follow the validity of the IFC file, after checking the validity of the IFC file, the IFC file will be imported into the developed tool, once the IFC file is imported in the tool, it will be automatically activated for checking the energy demand of the designed model.

Secondly, after importing the IFC file into the developed tool and analyzing the energy demand of the designed model, the output results will be analyzed and a new tab window will appear with feedback of the BEP analysis, this feedback will be sent to the architect.

Once the architect receives the BEP analysis feedback, the second set of activities will be applied and the architect will proceed in the decision-making process, as the architect considers the BEP analysis output in this set of activities, and he/she modifies the design that seeks sustainable solutions. Thus, two decisions may be involved in this process, modifying the 3D design model, exporting the new 3D model and re-importing the modified model to the developed tool, or ending the process and keeping the design model without modifications. The process map activities and change requirements are further explained in Appendix III.

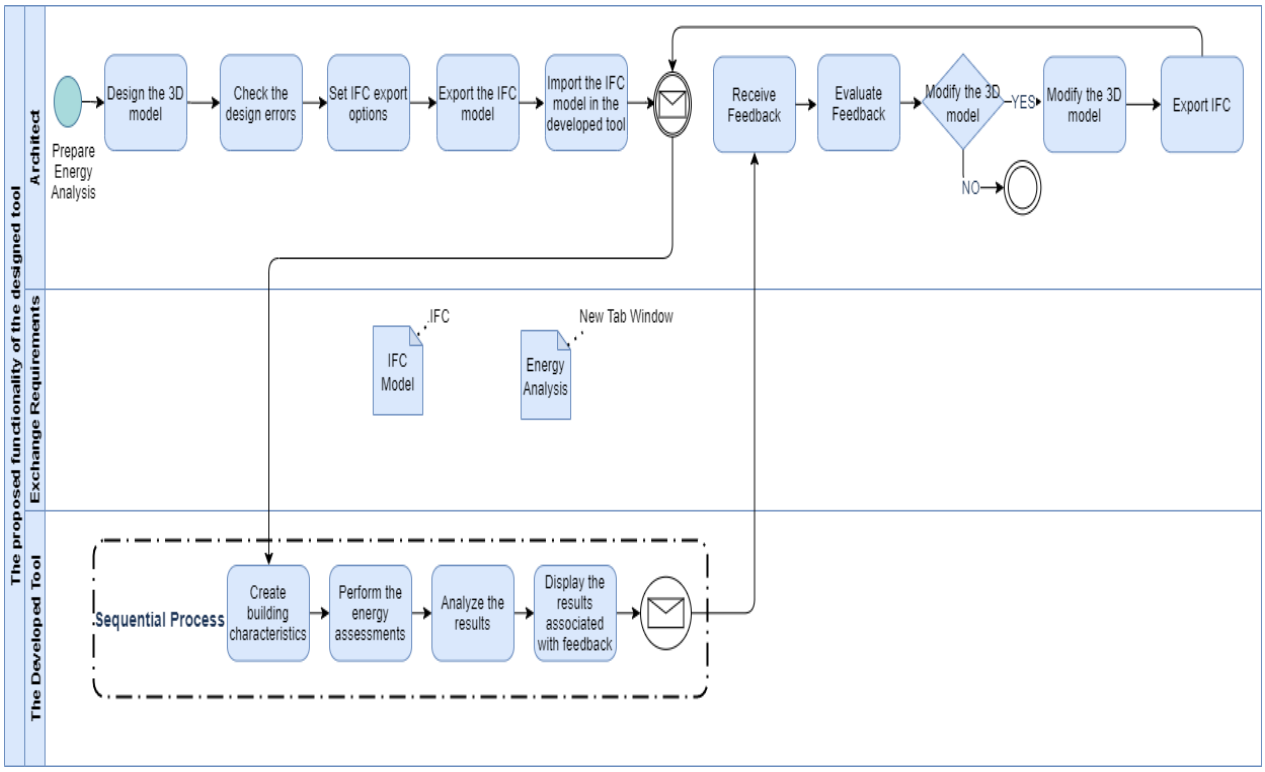

Figure 14 BPMN process of the Designed tool's functionality

4.2 Data Input

The integrated BIM-BPS tool requires data from two different sources, the first source is from Revit 2019 (BIM software) which stores its data in an IFC file and the second source is from the Dutch energy norm NTA 8800 which provides data on measurements of energy efficiency and EI. The BIM model provides a comprehensive design of the details and quantities needed for the measurement of the energy performance of the building elements, as well as classifications for the identification of the building components, while the NTA 8800 database provides information on the energy performance of the building components. In a nutshell, BIM data is input data for the BPS data to perform the energy assessments in the BIM-BPS integrated tool. Figure 15 shows the data input approach to perform the energy analysis in the developed tool.

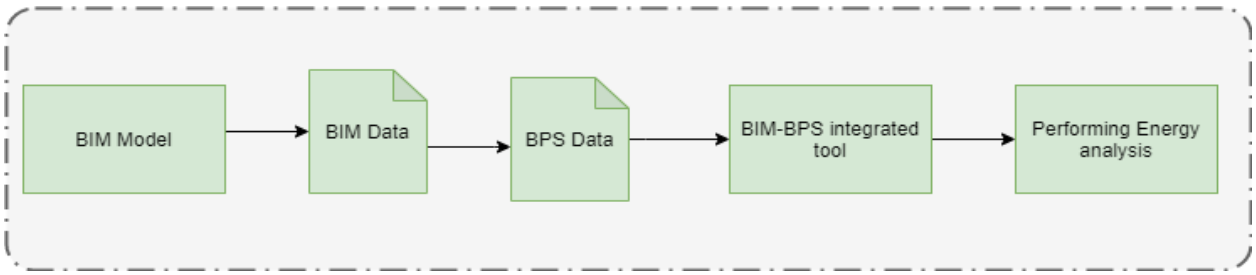

Figure 15 Data input approach

4.2.1 BIM data input

Developing the BIM-BPS integrated tool requires inputs from the design model, however as previously mentioned some BIM data are not available in the design model and not located in the right location based on the buildingSMART library, these data are explained in details how they're stored in the building model and how they should be stored based on the buildingSMART library for each building component. Hence, this section will explain the accurate way to derive the building data, and how to insert the unavailable data. To perform the energy assessments for the building model, some data are required to be extracted based on the building components, these data are deeply explained in this section for some building components. Figure 16 shows a schematic overview of the required data needed from the building model to perform the energy analysis.

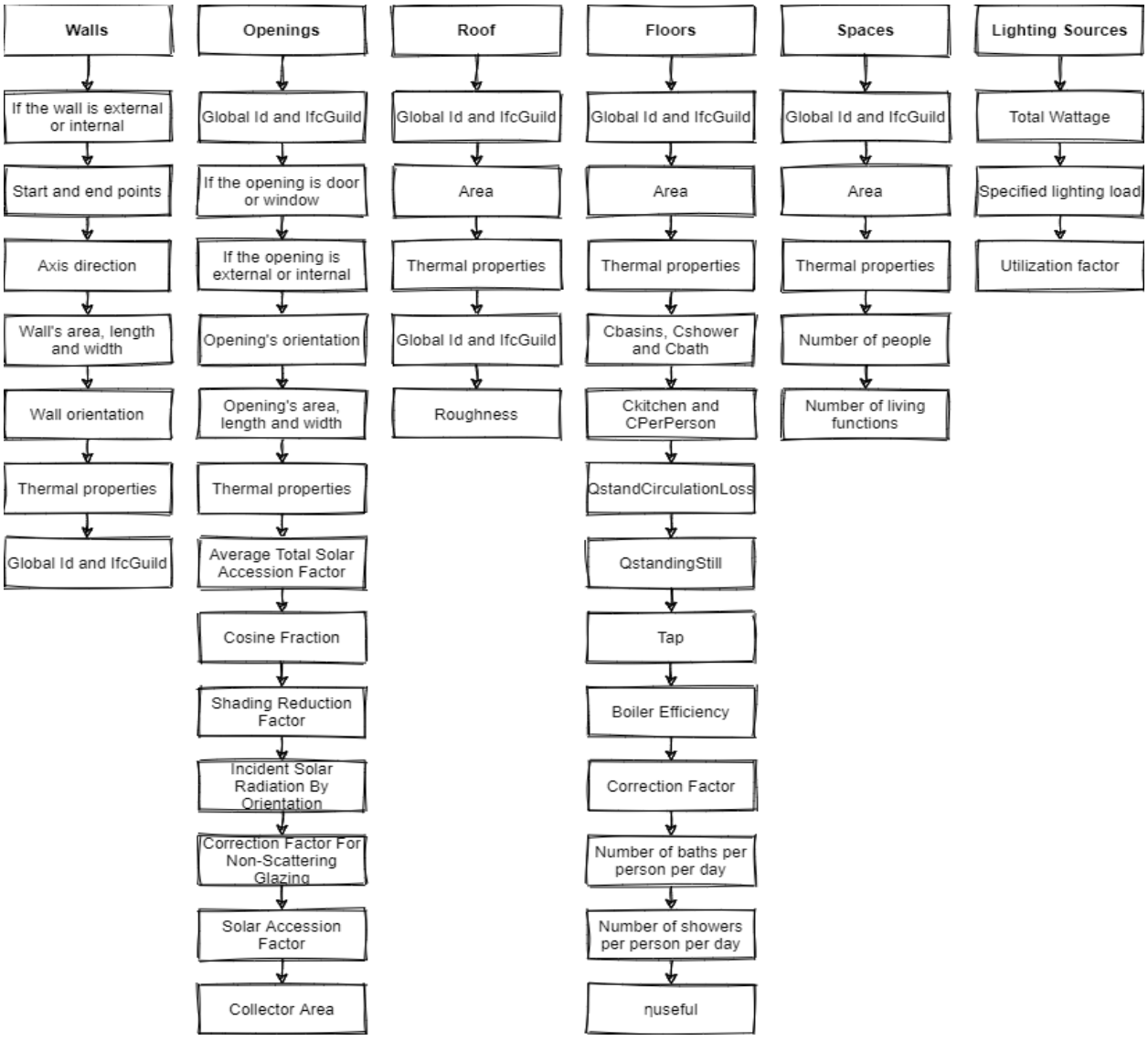

Figure 16 The building's data required for energy calculations

4.2.1.1 Walls

Based on the energy calculations, walls will be used to calculate the heat transfer through transmission. Based on the Dutch energy norm NTA 8800, calculating the heat loss through transmission requires Walls' areas, thermal transmittance, and heat transfer coefficient. Based on the buildingSmart library, the most accurate way to extract the element's quantity is to extract it from the IfcElementQuantity Property set. **IfcElementQuantity** Property set stores the object's physical data which can be allocated to an object, for instance: Area, Volume, Length and Width, these physical properties are assigned to the building's object via **IfcRelDefinesbyProperties** Relationship. These values are stored in an entity called Quantities, the assigned values in Quantities attribute are interpreted through common attributes, so for extracting Areas, there is an attribute called Area, similarly for extracting lengths and widths, there are attributes called Length and Width. Figure 17 shows how these values are stored in the IFC file format based on the buildingSMART international standards and figure 18 shows how these values are derived from the IFC file in PYTHON.

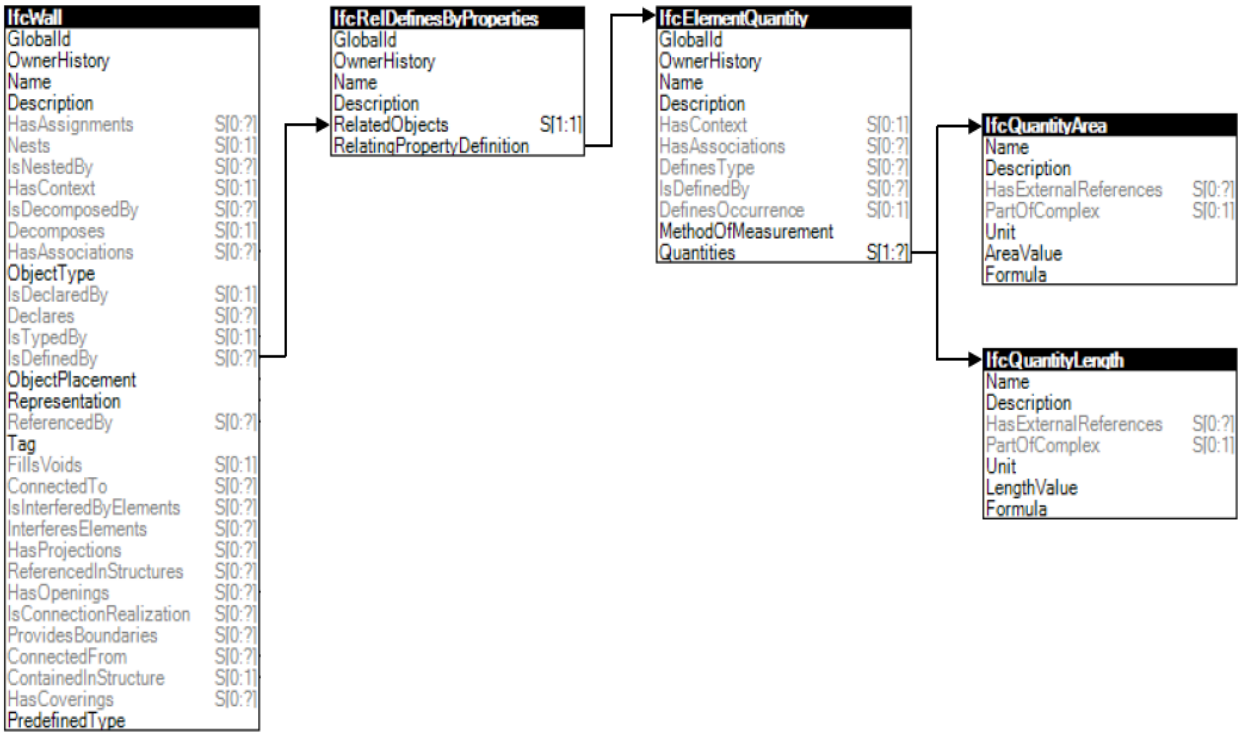

Figure 17 IFC schema from IFC wall shows how the quantities are mapped based on BuildingSMART library's international standards

```
690         def getWallLengthWidthHeight(wall, lengthOrWidthOrHeightOrAreaOrVolume):
691             width = 0
692             length = 0
693             height = 0
694             volume = 0
695             Area = 0
696             for relDefinesByProperties in wall.IsDefinedBy:
697                 if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
698                     if relDefinesByProperties.RelatingPropertyDefinition.is_a("IfcElementQuantity"):
699                         for properties in relDefinesByProperties.RelatingPropertyDefinition.Quantities:
700                             if properties.is_a("IfcQuantityLength"):
701                                 if properties.Name == "Width":
702                                     width = properties.LengthValue
703                                 if properties.Name == "Length":
704                                     length = properties.LengthValue
705                                 if properties.Name == "Height":
706                                     height = properties.LengthValue
707                                 if properties.Name == "Volume":
708                                     volume = properties.LengthValue
709                                 if properties.Name == "Area":
710                                     Area = properties.LengthValue
711                     elif relDefinesByProperties.RelatingPropertyDefinition.is_a("IfcPropertySet"):
712                         for properties in relDefinesByProperties.RelatingPropertyDefinition.HasProperties:
713                             if properties.Name == "Width":
714                                 width = properties.NominalValue.wrappedValue
715                             if properties.Name == "Length":
716                                 length = properties.NominalValue.wrappedValue
717                             if properties.Name == "Height":
718                                 height = properties.NominalValue.wrappedValue
```

Figure 18 deriving wall's quantities in Python

Moreover, other values are needed for calculating the heat Loss through transmission as Thermal transmittance, this value based on buildingSMART should be stored in **Pset_WallCommon**, A wall is correctly classed as an IfcWall entity, wall elements contain different properties, for instance: the IFC property IsExternal is placed based on the buildingSmart international standards in a property set called **Pset_WallCommon**, wall's thermal transmittance value is also placed in the **Pset_WallCommon** property set. However, these values were not located in my model as stated, hence a new property set has been created named Wall Schedule 2, in Wall Schedule 2, all the needed values which are not located in **Pset_WallCommon** has been inserted in this property set. Table 1 indicates the wall's properties placed based on the buildingSMART international standards and how the missing properties are stored based on the current IFC model.

Table 1 wall properties placement based on the IFC international standards and the current IFC model

Wall Properties	Property sets as stated based on the IFC international standards	Occurrence based on the IFC international standards	Property sets based on the current IFC model	Occurrence
IsExternal	Pset_WallCommon	✔	Pset_WallCommon	✔
ThermalTransmitance	Pset_WallCommon	✔	Pset_WallCommon	✔
Area	IfcElementQuantity	✔	IfcElementQuantity	✔
Width	IfcElementQuantity	✔	IfcElementQuantity	✔
Length	IfcElementQuantity	✔	IfcElementQuantity	✔

4.2.1.2 Openings

Based on NTA 8800, some data are required based on the building's components to calculate the heat loss through ventilation and heat gain by solar radiation. In this thesis, these data are derived based on the openings' properties. For calculating the heat loss through ventilation and heat gain by solar radiation Openings' areas, thermal transmittance, and heat transfer coefficient are required. Similarly as previously mentioned these data are stored based on the buildingSmart library in the IfcElementQuantity Property set. **IfcElementQuantity** Property set stores the object's physical data which can be allocated to an object, for instance: Area, Volume, Length and Width, these physical properties are assigned to the building's object via **IfcRelDefinesbyProperties** Relationship. These values are stored in an entity called Quantities, the assigned values in Quantities attribute are interpreted through common attributes, so for extracting Areas, there is an attribute called Area, similarly for extracting lengths and widths, there are attributes called Length and Width. In addition to the openings' physical properties, thermal values are required to perform the energy calculations, these values are stated below in figure 20 as follow:

I. Average total solar accession factor
II. Cosine fraction
III. Incident solar radiation
IV. Solar accession factor
V. Shading coefficient
VI. Correction factor

These values are stored as stated in the buildingSmart international library based on the opening's glazing type and shading type. The values which are dependent on the glazing type should be located in the entities IfcDoor or IfcWindow with a property set called **Pset_DoorWindowGlazingType** with property names SolarHeatGainTransmittance, GlazingAreaFraction, BeamRadiationTransmittance, and CorrectionFactor, these properties are assigned to the building objects via **IfcRelDefinesbyProperties** Relationship, table 1 indicates the property set definition reference for the glazing type. While the values which are dependent on the opening's shading type should be located in the entities IfcDoor or IfcWindow with a property set called **Pset_DoorWindowShadingType** with property names ExternalShadingCoefficient, this coefficient is also assigned to the building objects via **IfcRelDefinesbyProperties** Relationship, table 2 indicates the property set definition reference for the shading type. However, these values were not located in the building model and were manually inserted; a new property set was created called **Window_SolarRadiation** where the previously mentioned values are located, figure 19 below shows the created property set in the building model and table 2 shows an overview on the property sets and property names placements in the IFC based on the buildingSMART international standards library and their placements based on the current IFC model.

Table 2 Opening's properties placement based on the IFC international standards and the current IFC model

Opening's Properties	Property sets as stated based on the IFC international standards	Property Name	Occurrence based on the IFC international standards	Property sets based on the current IFC model	Current Property Name	Occurrence
Area	IfcElementQuantity	Area	✔	IfcElementQuantity	Area	✔
Width	IfcElementQuantity	Width	✔	IfcElementQuantity	Width	✔
Length	IfcElementQuantity	Length	✔	IfcElementQuantity	Length	✔
Internal or External	Pset_WindowCommon	IsExternal	✔		IsExternal	✔
Average solar accession factor	Pset_DoorWindowGlazingType	SolarHeatGainTransmittance	✗	Window_SolarRadiation	Average solar accession factor	✔
Cosine fraction	Pset_DoorWindowGlazingType	GlazingAreaFraction	✗	Window_SolarRadiation	Cosine fraction	✔
U-Value	Pset_WindowCommon	ThermalTransmittance	✔	Pset_WindowCommon	ThermalTransmittance	✔
Incident solar radiation	Pset_DoorWindowGlazingType	BeamRadiationTransmittance	✗	Window_SolarRadiation	Incident solar radiation	✔
Solar accession factor	Pset_DoorWindowGlazingType	SolarHeatGainTransmittance	✗	Window_SolarRadiation	Solar accession factor	✔
Correction factor	Pset_DoorWindowGlazingType	CorrectionFactor	✗	Window_SolarRadiation	Correction factor	✔
Shading	Pset_DoorWindowShadingType	ExternalShadingCoefficient	✗	Window_SolarRadiation	Shading coefficient	✔

coeffi cient

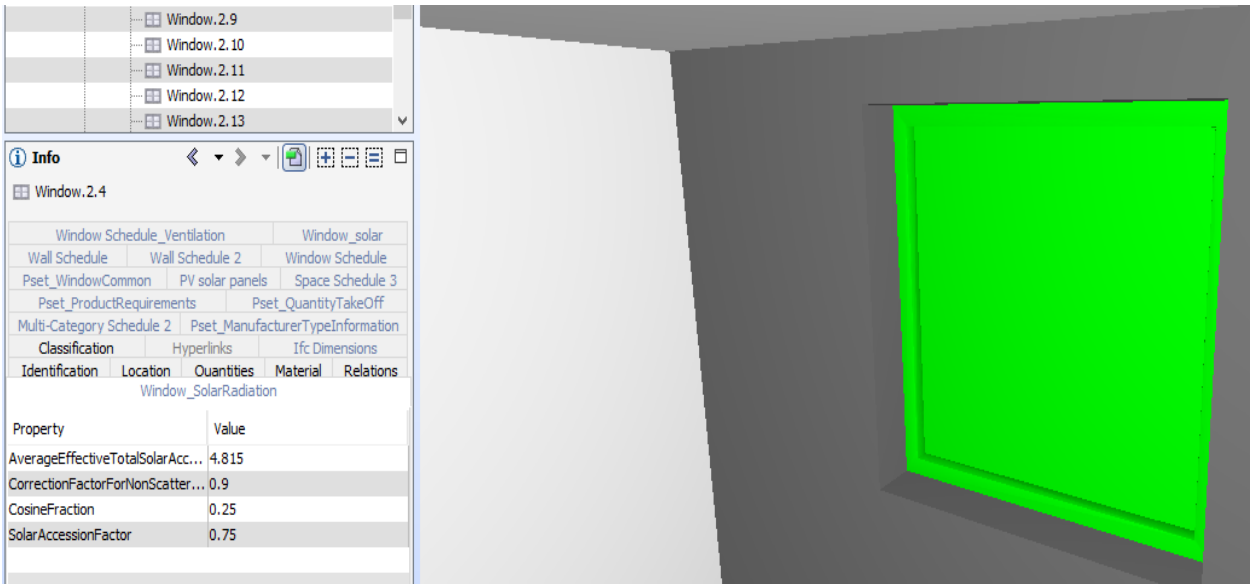

Figure 19 the Window_SolarRadiation property set in the building model

4.2.1.3 Floors

Floors have been used for calculating the heat gain by hot tap water based on the Dutch energy normative NTA 8800. Calculating the heat gain by hot tap water requires values from the BIM model, these values include floor area and other values for calculating the heat gain by hot tap water, as boiler efficiency, Qstanding still, Qstand circulation loss, ηuseful, Ckitchen, Cbasins, Cper person, Cshower, saving shower head (F), number of showers /person/day (D), Cbath, and number of baths /person/day. According to the buildingSmart international library, the previously mentioned values should be located in **IfcSanitaryTerminal**, these properties are assigned to the building objects via **IfcRelDefinesbyProperties** Relationship. The properties related to the kitchen are stored in a property set called **Pset_SanitaryTerminalTypeKitchen**, this predefined property is **IfcPropertySingleValue**, and all the object-related properties should be stored in this property set. Similarly, for basins, shower, saving shower head, and bath; the related properties are assigned to the building object via **IfcRelDefinesbyProperties** Relationship. The properties; related to basins are stored in a property set called **Pset_SanitaryTerminalTypeBasin**, all the values related to basins should be stored in this property set. To derive the values related to the shower, all the shower's properties' values should be stored in a property set called **Pset_SanitaryTerminalTypeShower.** According to the buildingSmart international library, all the related properties' values regarding type ''Bath'' should be stored in a property set called Pset_SanitaryTerminalTypeBath and these related

properties are assigned to the building object via **IfcRelDefinesbyProperties** Relationship. Figure 20 shows an IFC schema from the IFC sanitary terminal to show how the sanitary values are stored in IFC based on the IFC buildingSmart international library.

To derive the floor's area, similarly as mentioned in the previous section, by extracting the area from the **IfcElementQuantity** Property set, Entity Quantities, and attribute Area. For extracting the boiler efficiency value, the most accurate way is that these values should be stored in a property set called **Pset_BoilerTypeCommon** with an entity **IfcBoilerType**, within the entity IfcBoilerType there should the attribute values related to the Boiler stored, in this case, there should be **NominalEfficiency**, figure 22 shows a schematic overview over how the IFC boiler's values are stored based on the buildingSmart library. However in the used model, these values were not stored as stated, hence a new property set was created called floor schedule, this property set all the previously mentioned attributes were stored with values related to NTA 8800. Figure 21 shows the created property set in the IFC file format and tables 3 and 4 provide an overview of how the floor's properties are placed based on the IFC international standards and the current IFC model.

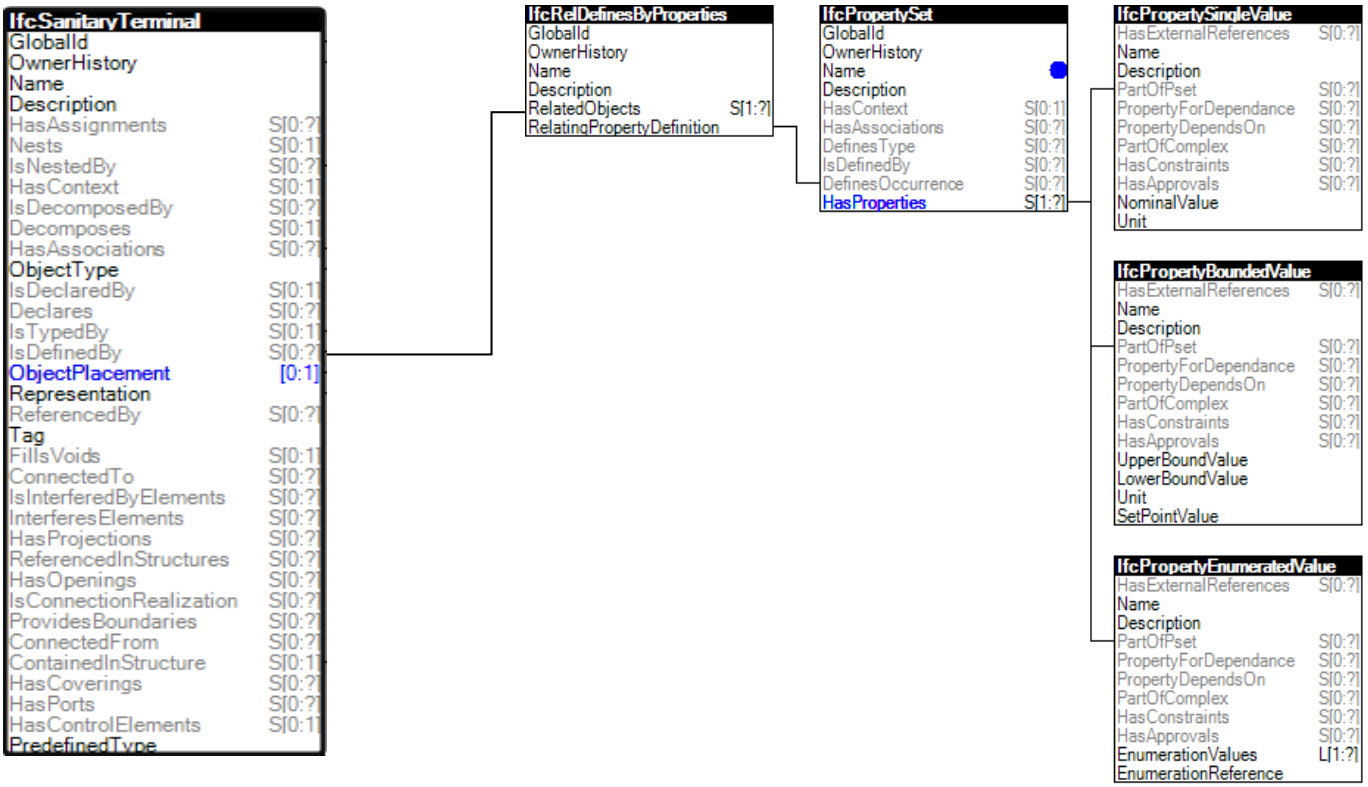

Figure 20 IFC schema from IFC sanitary terminal to show how the sanitary values are stored in IFC based on the IFC buildingSmart library (source: buildingSmart (2012))

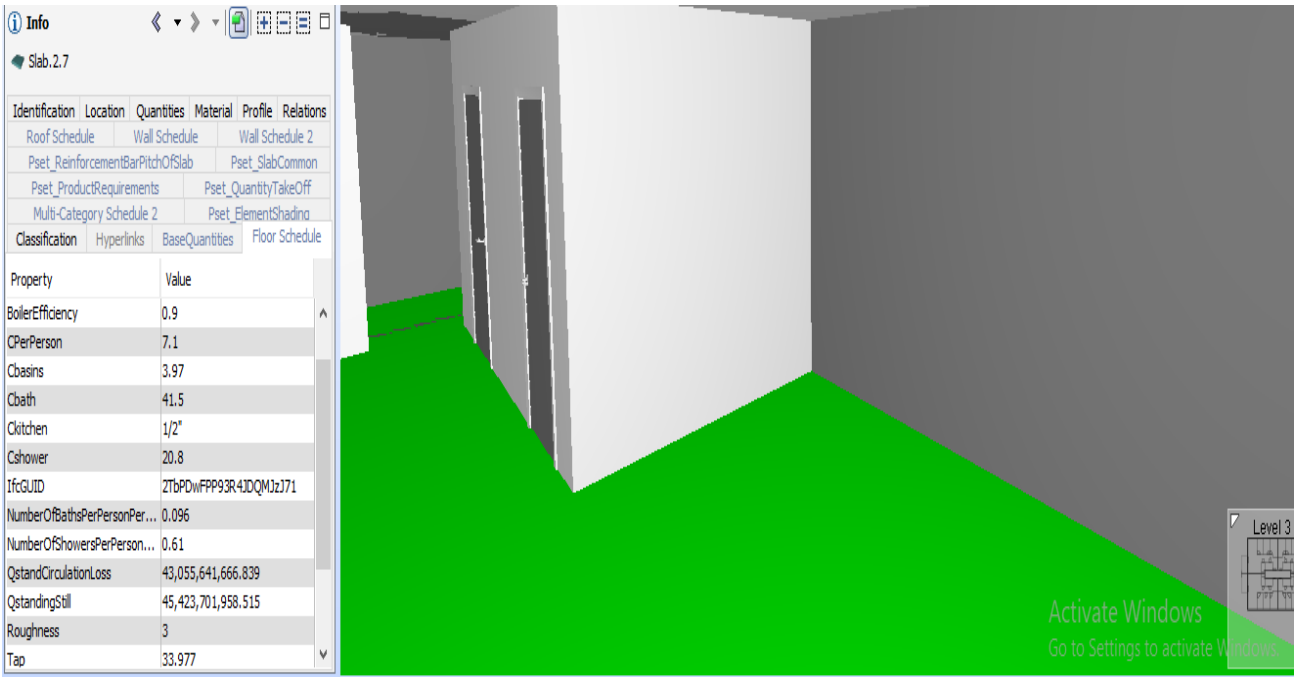

Figure 21 Values stored in Floor schedule property set

Table 3 Floor properties placement based on the IFC international standards and the current IFC model

Floor's Properties	Property sets as stated based on the IFC international standards	Property name	Occurrence based on the IFC international standards	Property sets based on the current IFC model	Property name	Occurrence based on the current IFC model
Area	IfcElementQuantity	Area	✔	IfcElementQuantity	Area	✔
Cbasins	Pset_SanitaryTerminalTypeBasin	NominalEfficiency	✗	FloorSchedule	Cbasins	✔
Cshower	Pset_SanitaryTerminalTypeShower	NominalEfficiency	✗	FloorSchedule	Cshower	✔
Cbath	Pset_SanitaryTerminalTypeBath	NominalEfficiency	✗	FloorSchedule	Cbath	✔
Ckitchen	Pset_SanitaryTerminalTypeKitchen	NominalEfficiency	✗	FloorSchedule	Ckitchen	✔
Tap	Pset_SanitaryTerminalTypeKitchen	Tap	✔	FloorSchedule	Tap	✔
Correction factor	Pset_SanitaryTerminalTypeKitchen	Correction factor	✔	FloorSchedule	Correction factor	✔

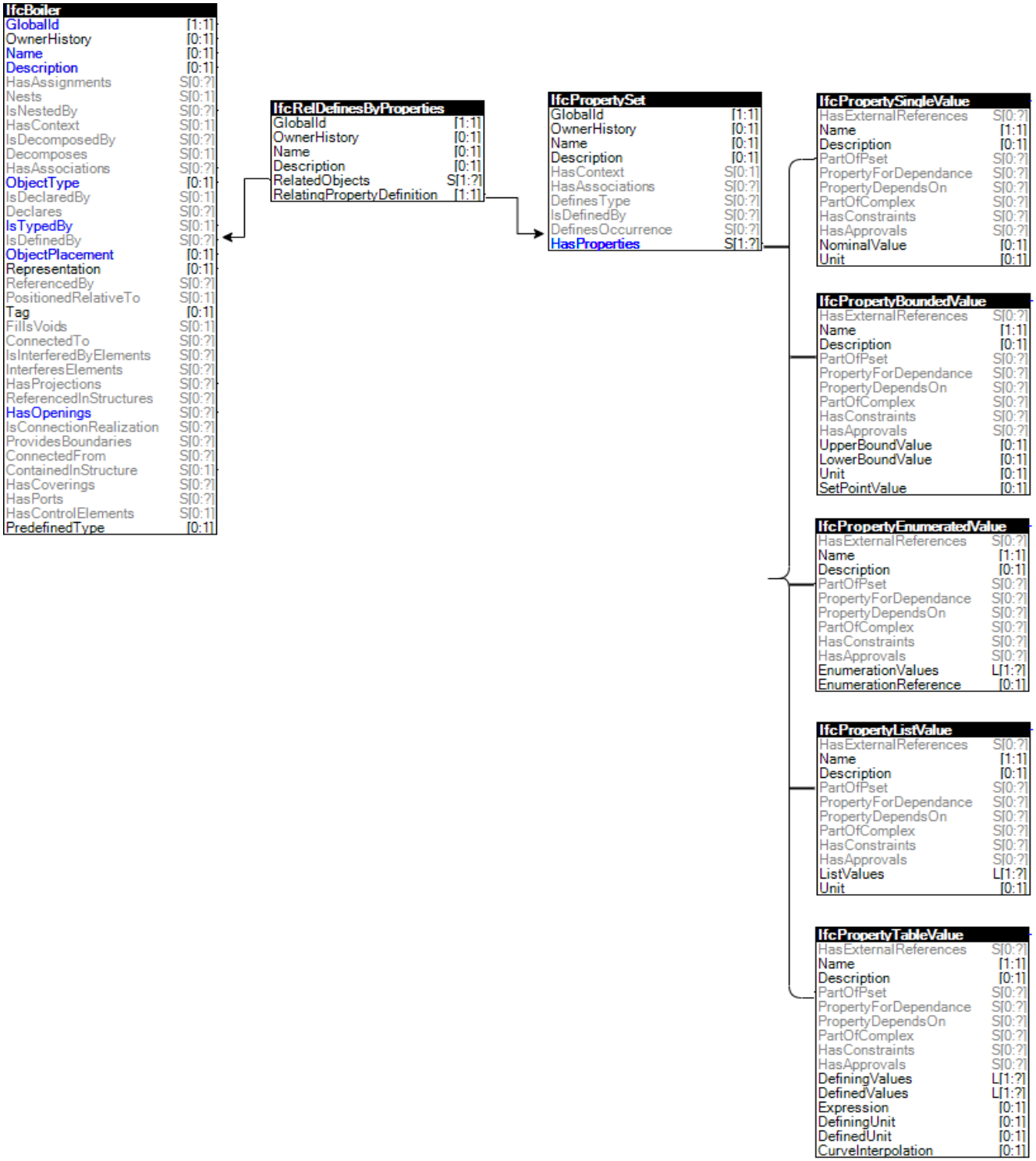

Figure 22 IFC schema from IFC boiler to show how the boiler values are stored in IFC based on the IFC buildingSmart library (source: buildingSmart (2012))

Table 4 Floor Boiler's properties placement based on the IFC international standards and the current IFC model

Floor's Properties	Property sets as stated based on the IFC international standards	Property name	Occurrence based on the IFC international standards	Property sets based on the current IFC model	Property name	Occurrence based on the current IFC model
Boiler efficiency	Pset_BoilerTypeCommon	NominalEfficiency	✗	FloorSchedule	BoilerEfficiency	✔

4.2.1.4 Spaces

Spaces have been used to detect the amount of heat gain by lighting sources. For calculating the heat gain by lighting based on the energy Dutch normative NTA 8800, the area of space is needed as well as the total wattage and utilization factor of the lighting source. To derive space's area, similarly as mentioned in the previous sections, by extracting the area from the IfcElementQuantity Property set, Entity Quantities, and attribute Area. As previously mentioned storing information in IFC should be object orientated, hence for extracting the lighting source's data, based on BuildingSmart library, the accurate way of deriving data from the lighting fixture is that data are stored in Pset_LightFixtureTypeCommon property set with entity IfcLightFixtureType, attributes are stored with a common name called, TotalWattage and UtilizationFactor, based on these attributes data should be extracted. Table 5 provides an overview of how space's properties are located based on the IFC international standards and the current IFC model and table 6 shows how the lighting fixtures' data are mapped in the IFC file. Figure 23 shows a schematic overview of how the lighting values are stored in the IFC light fixtures.

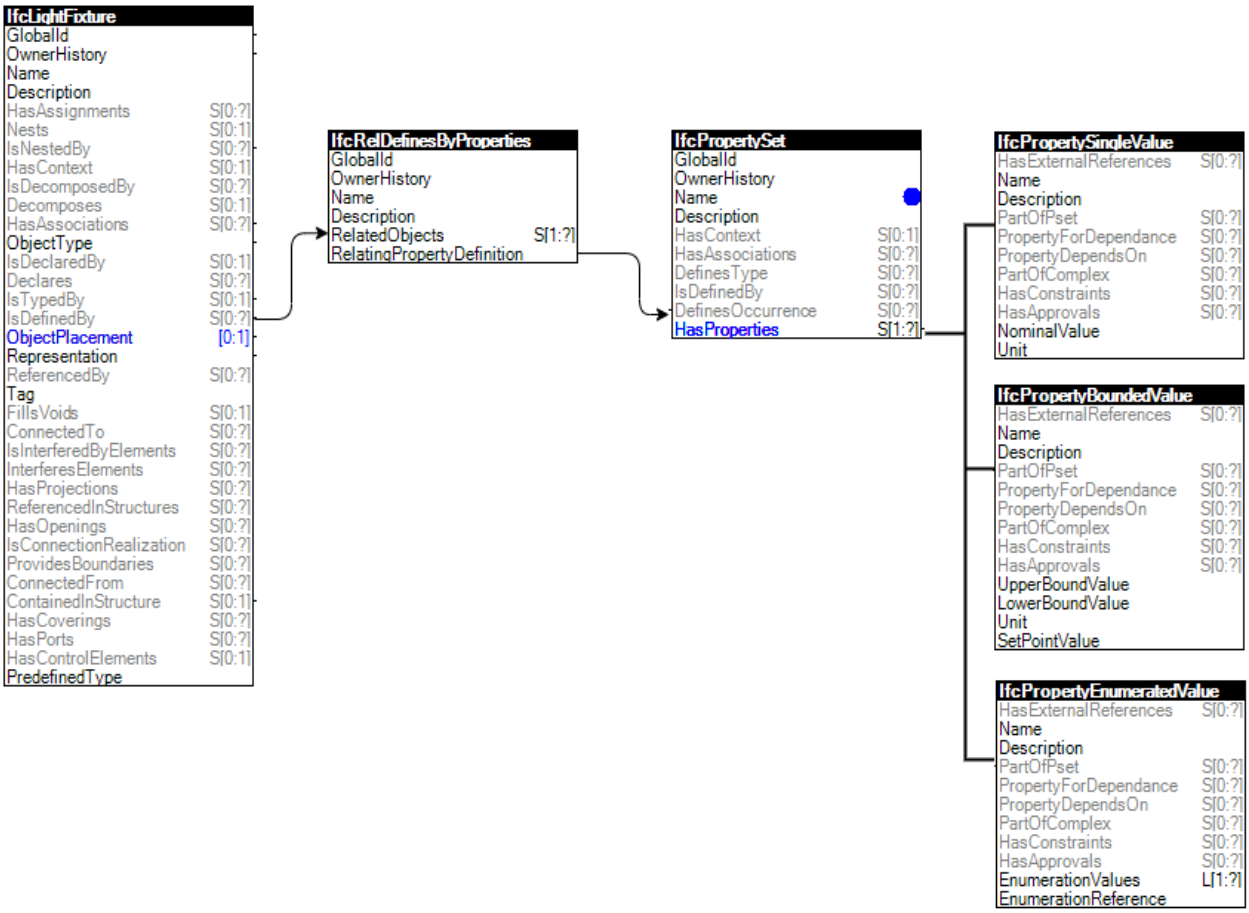

Figure 23 IFC schema from IFC Light fixture to show how the lighting values are stored in IFC based on the IFC buildingSmart library (source: buildingSmart (2012))

Table 5 Space properties placement based on the IFC international standards and the current IFC model

Space's Properties	Property sets as stated based on the IFC international standards	Property name	Occurrence based on the IFC international standards	Property sets based on the current IFC model	Property name	Occurrence based on the current IFC model
Area	IfcElementQuantity	Area	✔	IfcElementQuantity	Area	✔
Total Wattage	Pset_LightFixtureTypeCommon	Total Wattage	✔	Pset_LightFixtureTypeCommon	Total Wattage	✔

Table 6 deriving the lighting fixtures in IFC

	Existence 1	Value Type 2	Cardinality 3	Unit 4	IFCmapping
Type of Light source	M	String	n/a	n/a	Pset_LightFixtureTypeCommon
	M	String	n/a	n/a	Pset_LightFixtureTypeCommon
	M	String	n/a	n/a	Pset_LightFixtureTypeCommon
	M	String	n/a	n/a	Pset_LightFixtureTypeCommon
	M	String	n/a	n/a	Pset_LightFixtureTypeCommon
TotalWattage					
	M	Real	>=60	W	Pset_LightFixtureTypeCommon _IfcLightFixtureType_Property.Name = “TotalWattage”
	M	Real	>=60	W	Pset_LightFixtureTypeCommon _ IfcLightFixtureType _Property.Name = “TotalWattage”
	M	Real	>=60	W	Pset_LightFixtureTypeCommon _ IfcLightFixtureType_Property.Name = “TotalWattage”
	M	Real	>=60	W	Pset_LightFixtureTypeCommon _ IfcLightFixtureType _Property.Name = “TotalWattage”
Utilization					
	O	Integer	>=0.81	n/a	Pset_LightFixtureTypeCommon _ IfcLightFixtureType _Property.Name = “Utilization”
	O	Integer	>=0.71	n/a	Pset_LightFixtureTypeCommon _ IfcLightFixtureType _Property.Name = “Utilization”
	O	Integer	>=0.9	n/a	Pset_LightFixtureTypeCommon _ IfcLightFixtureType _Property.Name = “Utilization”
	O	Integer	n/a	n/a	Pset_LightFixtureTypeCommon _ IfcLightFixtureType_Property.Name = “Utilization”
	O	Integer	c	n/a	Pset_LightFixtureTypeCommon _ IfcLightFixtureType _Property.Name = “Utilization”

4.2.2 BPS data input

For calculating the EI, energy performance-related data is required. The energy performance data is utilized based on the Dutch Normative NTA 8800. NTA 8800 provides a determination method for calculating the energy performance of the building.

NTA 8800 is guidance on how to calculate the energy data based on the building characteristics as it provides a standardized way to measure the heat gain and heat loss of the building. These data is narrowed to the following factors:

- Heat loss through transmission
- Heat loss through ventilation
- Internal heat gain
- Heat gain by solar radiation
- Heat gain by hot tap water
- Heat gain by lighting
- Heat gain by PV cells
- Heat gain by Cogeneration

4.3 Energy Index calculation and energy efficiency measures

The Energy Index (EI), a dimensionless metric, ranges generally from A (extremely good performance) to G (extremely poor performance). The EI calculation method is described in NTA8800 (published by the Dutch Normalization Institute) and, to some degree, is also defined in ISSO Publication 82.3 – ISSO as an organization that offers approved guidance and specifications for technical systems to many institutions in the Netherlands, such as the Ministry of Interior, which is responsible for energy efficiency measures. The labels' primary purpose is to provide occupants and homeowners with knowledge about the thermal efficiency of their dwellings. In addition, the dwelling's theoretical energy use is also listed on all Dutch labels released after January 2010, expressed in kWh of electricity, m3 of gas, and GJ of heat, for district-heated dwellings (Majcen et al., 2013). The EI is associated with the total theoretical energy consumption of a Qtotal building or dwelling. According to the measurement norm for the EI (NTA 8800), as shown in Equation 1 See Appendix I, it is adjusted taking into consideration the dwelling floor area and the related heat transfer areas. A shape correction is often applied when determining the losses of infiltration within the demand for space heating, although the coefficient of air permeability relies on the shape factor of the buildings. The energy index corresponds directly with overall primary energy usage but is calibrated for the dwelling floor area and the associated heat transfer areas so as not to harm larger dwellings and dwellings with a larger portion of their heat envelope surrounding unheated spaces (different types of buildings) with certain insulation characteristics and efficiencies of the dwellings.

4.4 Calculation method

This section briefly explains how potential energy consumption is measured for Dutch dwellings and how the energy label is defined. Additionally, in the following equations, some assumptions have been put into consideration. The whole energy label measurement and determination process can be found under NTA 8800 (2020). Figure 25 shows a schematic overview of the energy calculation steps presented in the calculation part of the tool.

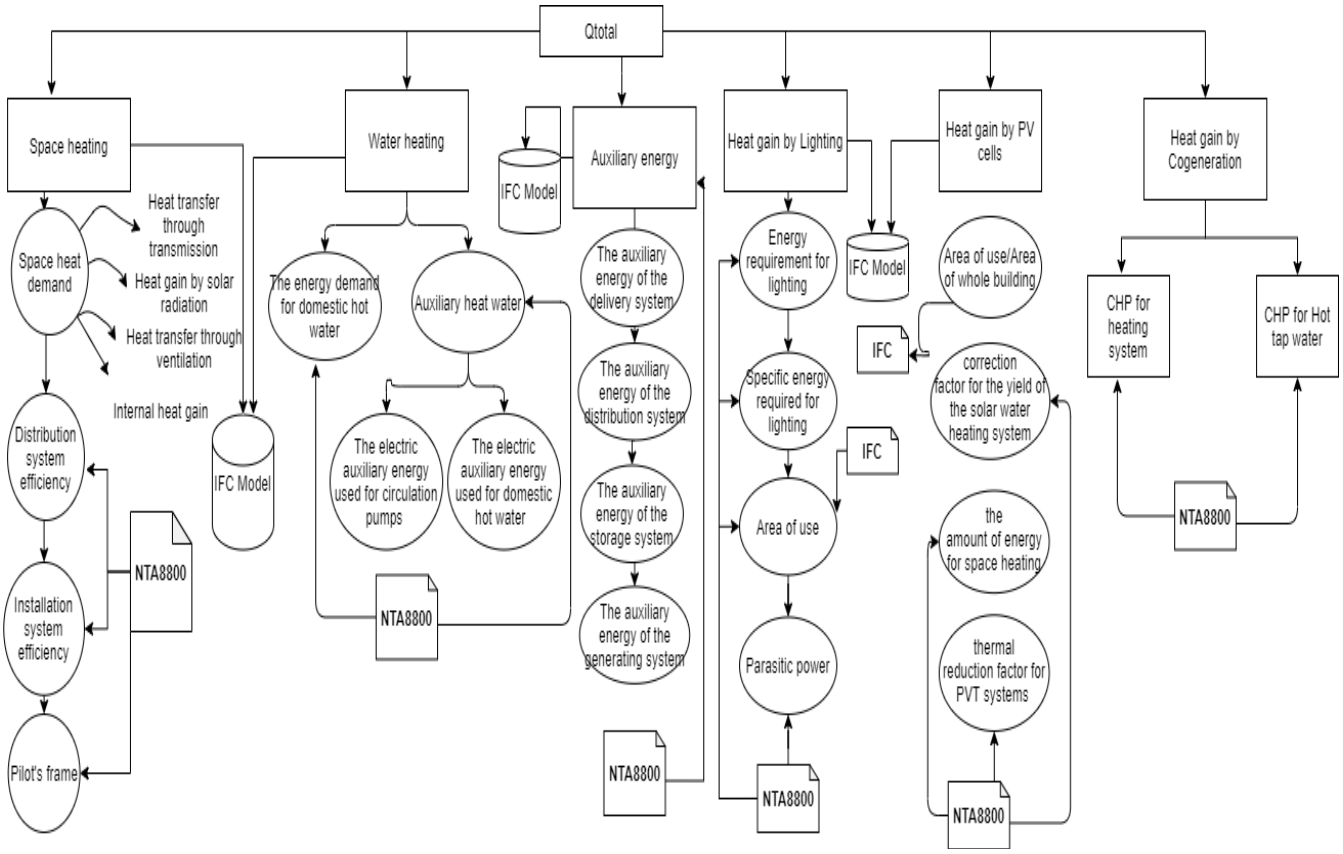

Figure 24 Schematic overview of the EI calculations' steps

4.4.1 Energy Index

The Energy Index (EI), a dimensionless metric, ranges generally from A (extremely good performance) to G (extremely poor performance). The EI calculation method is described in NTA8800 (published by the Dutch Normalization Institute) and, to some degree, is also defined in ISSO Publication 82.3 – ISSO as an organization that offers approved guidance and specifications for technical systems to many institutions in the Netherlands, such as the Ministry of Interior, which is responsible for energy efficiency measures. The labels' primary purpose is to provide occupants and homeowners with knowledge about the thermal efficiency of their dwellings. The calculation of the EI has been stated in Equation 1 based on the

determination method of the Dutch energy norm NTA 8800, see Appendix I. Based on NTA 8800, the total heat gain and loss by the building components are calculated based on various factors stated in Equation 2, Appendix I as follow:

- Space heating
- Water heating
- Auxiliary energy
- Heat gain by Lighting
- Heat gain by PV cells
- Heat gain by cogeneration

4.4.1.1 Space heating

The space heating requirement is calculated based on NTA 8800 as stated in Equation 3 (See Appendix I). The distribution system efficiency ranges up to 1 for a residential building where the temperature setting is optimal, the distribution system efficiency is assumed to be equal to 1. Installation system efficiency (ηinstallation) is assumed to be 1. Besides, the potential contribution of a solar boiler has been assumed to be 0. The energy needed for the pilot's flame is estimated to be 2,500 J.

4.4.2 The space heat demand

Heat transfer is an engineering discipline that covers the heat (thermal energy) generation, usage, conversion, and exchange across physical systems. Usually, heat transfer is categorized into different processes such as heat conduction, heat convection, and thermal radiation. Convection of heat relies on mass movement from one area of space to another. In addition, heat conduction, also known as diffusion, takes place in a body or contact between two bodies, while thermal radiation is heat transfer by electromagnetic radiation, such as sunlight, without the need for the matter to be included in the space between bodies (Connor, 2019). Based on the Dutch energy norm NTA8800, the space heat demand has been calculated based on the following methods (Equation 3.1), see Appendix I.

- Heat transfer through transmission
- Heat transfer through ventilation
- Internal heat gain
- Heat gain by solar radiation

4.4.3 Heat transfer through transmission

Based on the Dutch normative (NTA8800), the heat by transmission has been identified by the conduction of heat between the inside atmosphere and the outside. Thermal conduction, also called heat conduction, arises in contact within a body or between two bodies, without mass transfer and mixing being involved. It is the direct microscopic exchange of particle kinetic

energy through the boundary between two systems. Heat transfer by conduction depends on the temperature difference driving "energy" and the thermal conductivity (or heat transfer resistance) (Connor, 2019). The thermal conductivity depends on the existence of the heat transfer medium and its dimensions. Heat by transmission is calculated based on NTA 8800 as stated in Equation 3.1.1, see Appendix I. With regard to temperatures and the related length of time, NTA 8800 provides certain values for each month. The inside temperature varies depending on the functionality of the building. Mostly, the indoor temperature is fixed and placed at 20 ° C for the winter months, while the temperature rises to 24 ° C from May to September. In (Equation 3.1.1.), the heat transfer through transmission equation is simplified, transmission rates are determined based on an annual heating cycle and a constant average indoor temperature of 22° C. The floor area which is used in the calculation are limited to the heated rooms (bedrooms, dining room, kitchen), plus some rarely heated areas (halls, bathroom, laundry room, storage). Moreover, heat transfer coefficient is a constant value which is derived from NTA 8800, the choice of the heat transfer coefficient value may vary based on different conditions as stated in table 4, Appendix II, also the stated calculations are assumed to be in June, the time length is stated in table 5, Appendix II.

4.4.4 Heat transfer through Ventilation

The Dutch Normative has described heat by ventilation as the heat convection, which is correlated by natural means with air movement through a structure. This is determined primarily by the heat transfer coefficient of ventilation which is a function of air heat capacity, air density, a correction factor of supply temperature of air volume, the dynamic correction factor for air volume flow and the air volume flow, and also the ventilation heat transfer coefficient varies based on the building functionality, table 6 shows the difference in ventilation heat transfer coefficient based on the building functionality, Appendix II. Equation 3.1.2 shows the heat transfer through ventilation calculation based on NTA 8800, Appendix I. The total heat transfer coefficient through ventilation for calculation zone is calculated based on the stated formula (Equation 3.1.2.1), this calculation is limited to the case of natural ventilation the standard condition, and the ventilation heat transfer coefficient is distinguished based on the type of ventilation and the infiltration rates.

4.4.5 Internal Heat Gain

According to the Dutch norm NTA8800 the internal heat gain refers to the contribution of internal sources to heat management rather than the conscious supply of heat or cold for space heating, space cooling, or preparation of hot water. The average number of residents per calculation zone per residential function depends on the average usable area per function. In the calculation only the internal heat gain is used in the calculation zone itself represented by Equation 3.1.3, Appendix I.

4.4.6 Heat gain by solar radiation

The Dutch norm NTA 8800 has defined the heat gain from striking solar radiation (solar heat gain) as the contribution to the building's heat control due to on-site solar radiation, receiving surface orientation, permanent and mobile sun protection and sun propagation, light absorption, and heat transfer properties of the receiving planes. The evaluation process also involves a shading correction due to external obstructions associated with the plot itself. Furthermore, a correction is made to radiate to the atmosphere. Moreover, NTA 8800 has assumed in the calculation processes that the constructions considered are transparent constructions such as glazing (including any incorporated or added sun protection device), non-transparent external separating constructions, and separating buildings behind a transparent cover or transparent insulation. Additionally, the gain from solar heat is charged via greenhouses. Calculating the total solar entry factor (corrected for the angle of incidence) (g):
This calculation has considered the following properties:

- Non-scattering glazing windows
- Double glazing windows
- Glass only

Moreover, for determining the cosine factor for windows, there are two methods for calculation:

- Method A: The surface of the glazing can be determined with the geometrical data or window dimensions
- Method B: it can be derived from a fixed chosen frame fraction

As the windows' frame fraction is unknown because fixed values for the heat transmission coefficient for windows are used, Method B has been used and the cosine fraction for windows has been assumed as 0.25. In addition, the convection correction factor is limited to a situation of weak convection, as the insulation is installed in such a way that air cavities can break through the insulation layer, but no air circulation is possible on the warm side of the insulation, in this situation the convection correction factor is assumed as 0.01.

These calculations are explained in detail in Appendix I (Equation 3.1.4), (Equation 3.1.4.2) and table 7 shows the values of the total solar access factor at perpendicular incidence for common types of glazing, Appendix II.

4.4.7 Water heating

The Dutch energy norm NTA 8800 has stated that one or more hot water systems may provide the hot tap water in a house, residential building, or utility building. Moreover, a hot water system consists of the heat intake,-delivery, storage, and generation interrelated components.

In addition, a hot water system does not always correlate with a calculation zone: one system may contain more calculation zones and more systems may contain one calculation zone. The energy consumption per hot water system shall be determined per the carrier of electricity. Deviating from this, the calculation per calculation zone for utility buildings may be limited to the type of tap water system to which most of the tapping points are connected (NTA 8800, 2020). Moreover, according to NTA 8800, the following elements have been considered while calculating the energy consumption of hot tap water the energy consumption for hot tap water is determined per hot water system by the following elements:

- The energy demand for domestic hot water
- electric auxiliary energy use

The energy demand for water heating takes into account the main installation of hot water and the auxiliary kitchen boiler (which is estimated to have a normal demand of 2267.81 KWH). Moreover, the auxiliary heat water requires some steps to observe the calculations and due to the short period needed to deliver the graduation thesis, it has been assumed that the auxiliary energy of water = 0, however, all the needed calculations which are needed for auxiliary heat water are explained in Appendix I. The following equations pertain to a condensing boiler. Standard hot water consumption as described in NTA 8800 is calculated on the basis of the national average (Equations 4 and 4.1). These calculations are stated in Appendix I.

4.4.7.1 Auxiliary heat water

The total auxiliary energy consumption for domestic hot water in a building per month is determined in NTA 8800 as follows:

- Determine per month mi the sum for all the hot water systems involved si (excluding the auxiliary energy for heat generation)
- Determine per hot water system si the sum for all involved storage vessels, generators gi (index generator), and solar water heaters.

The formulas based on NTA 8800 are explained in Appendix I (Equations 4.2, 4.2.1, and 4.2.2)

4.4.7.1.1 The electric auxiliary energy used for circulation pumps

The consumption of any circulation pumps present is determined by the pump power and the time that the circulation system is in use. If there is a circulation system with central heating water, the auxiliary energy for the period in which central heating is in operation has already been included in heating. Only the auxiliary energy for the period that the system is only in use for domestic hot water is determined in Appendix I (Equations 4.2.2, 4.2.2.1 and 4.2.2.2), table 8 shows the time that the circulation system is in operation for the distribution of heat water which is stated in Appendix II.

4.4.7.1.2 The electric auxiliary energy used for domestic hot water

The calculation below is limited to the case of a building or part of a building belonging to the residential building category with one hot water system for both the kitchen (s) and the bathroom (s), the following calculation rule applies for the amount of electric auxiliary energy used for domestic hot water, the calculation of the amount of electric auxiliary energy used for domestic hot water is stated based on the Dutch energy Norm NTA 8800, these calculations are explained in Appendix I, (Equations 4.2.2, 4.2.2.1, 4.2.2.1.1 and 4.2.2.2).

4.4.8 Auxiliary energy

This section describes the auxiliary energy for space heating of all related subsystems. The auxiliary energy is calculated according to NTA 8800 as follows (Equation 15). Since the auxiliary energy of the circulation pump is not an integral part of the total auxiliary energy, therefore based on NTA8800, the auxiliary energy of the distributed system is assumed to equal to 0. Moreover, Heat storage for space heating is only considered in combination with heat generation from solar energy systems, for this situation the auxiliary energy of the related storage system based on the Dutch energy norm NTA8800 is assumed to be 0. Lastly, as the heat storage for space heating is considered in combination with heat generation from solar energy systems, hence based on NTA 8800, the auxiliary energy of the related generating system is equal to 0; these calculations are explained in Appendix I in (Equations 5, 5.1, 5.2, 5.3, 5.4, 5.5 and 5.6). The auxiliary energy is calculated based on the related subsystems which are narrowed down to the following systems:

- the auxiliary energy of the delivery system
- the auxiliary energy of the distribution system
- the auxiliary energy of the storage system
- the auxiliary energy of the generating system

Moreover, the fan power is calculated in NTA 8800 as shown in table 9, Appendix II. The number of hours per day with a reduced set point temperature for heating and the total number of hours per weekend with reduced set point temperature for heating are stated in table 10, Appendix II.

4.4.9 Heat Gain by Lighting

The heat gain from lighting is the sensible and latent heat emitted by any light source within an internal space to be eliminated by air conditioning or ventilation, and/or contributes to a rise in temperature and humidity within the room. The electrical energy using any source of lighting is emitted as heat energy. The power is released by conduction, convection, or radiation. When the luminaire is switched on it absorbs some of the heat emitted by the lamp itself. Much of the heat will then be transferred to the surface of the house, depending on how the luminaire is installed. The radiation energy emitted from a lamp will only lead to heat gain in space after it is

absorbed by the room surfaces. According to the Dutch norm NTA 8800, a distinction is made between calculation zones with a residential function and calculation zones with utility functions. The energy requirement for lighting for calculation zones with a residential function is determined in (Equation 6), these calculations are explained in Appendix I. Moreover, according to NTA 8800 the energy requirement for lighting consists of:

- The electrical energy needs for lighting to provide the necessary lighting levels.
- The electrical energy requirement for parasitic power for charging the batteries of emergency lighting luminaries and for the total standby power for the automatic control of the luminaries.

4.4.10 Heat gain by PV cells

The energy gain from solar panels determines the contribution of the heat supplied by the solar energy system to hot water preparation on the basis of the energy demand for domestic hot water supplied to the distribution part (NTA 8800, 2020). A solar energy system is always linked to a tap water or heating system:

- a solar energy system provides a heat contribution to one tap water system or one heating system or both;
- a tap water system can be powered by several solar energy systems:
    - If these are different solar water heating systems, these must be specified separately within the DHW system.
    - When the tap water system is modeled as one large system with several identical physical generation systems (with the same generators of the same brand, type, and power and the same energy carriers; for instances, in a residential building with an individual generation system per dwelling), the system is also powered by several solar energy systems. The physical number of solar energy systems is then equal to the physical number of generating systems. When determining the heat to be supplied or supplied by the solar water heating system, the heat to be supplied per physical generating system must then be taken into account.

Based on the current model, it has been assumed that the building doesn't consume energy from PV cells; however, detailed calculations have been explained in (equation 7) in Appendix I. Moreover, the amount of energy for space heating supplied by thermal renewable energy generated on its plot by solar energy system depends on the correlation factor of the collector connected to the storage vessel and (NTA 8800, 2020), these correlation factors are mentioned in table 11 in Appendix II as well as the heat loss of the storage tank which is varied based on the energy label. It has been assumed that the solar energy system is linked to a water heating system with an integrated electrical

re-heating system, figure 28 shows how this system is modeled. See Appendix II.

4.4.11 Heat Gain by Cogeneration

cogeneration or combined heat and power (CHP) is defined as the sequential generation of two beneficial energy sources from a single primary renewable energy source, usually mechanical and thermal. According to NTA 8800, there are two methods for determining the energy consumption for heating and tap water for a building-related CHP, these two methods are determined as follow:

- A method based on fixed yields for gas-fired CHPs where no measurement data in accordance with NEN-EN 50465 are known.
- A detailed method based on measurement data in accordance with NEN-EN 50465 for appliances with a thermal capacity of up to 70 kW, with which the performance of a CHP can be taken into account.

Note: The primary energy gains from Cogeneration is calculated based on Method 1, the Dutch energy norm NTA8800 has indicated that the calculations based on Method 1 are taken from NEN 7120, which are calculated as follow in (Equations 8, 8.1 and 8.2), these equations are discussed in Appendix I and the system used in the calculation is based on H_2PS-5 , a Combined Heat & Power (CHP) system. It has a capacity of 5 KW electric power and 7 KW thermal powers to produce hot water. The average electrical conversion and the mean thermal conversion number of relevant cogeneration installations are determined based on the electrical power of the relevant cogeneration installation; these values are stated in table 13 in Appendix II.

4.5 Tool interface

This section explains the tool interface function involved in the developed tool's system. There are two layouts involved in the tool; the initial layout and the energy results layout, the procedures involved in these layouts are described below.

1. The initial layout:

In the initial layout is the standard layout which is displayed once the user utilizes the tool. In this layout, three steps are taken by the designer, based on these steps the energy assessment is generated.

The first step which is taken by the user is to load the IFC model, once the IFC model is loaded, the user will be required to choose the building functionality. The developed tool is only limited to the Residential building functionality, based on the various steps the energy assessment will be performed.

The execution of these steps is performed once, based on the generated energy assessment the user will check the results and hence be allowed to modify the IFC file. The user is allowed to load a modified IFC model, based on the loaded IFC model the tool will generate new energy assessments along with related feedbacks.

2. The results layout:

- Energy results widget: Once the energy button is clicked the developed tool generates the energy assessments' results. Hence the user will get a deep insight into the energy performance of the designed model.
- Feedback box: This option allows the designer to detect the energy performance resulted from the building elements. These feedbacks are only detected for two building elements as an instance (Walls and Openings)
- Illustration box: This option displays the energy results on an imported image. The energy requirements displayed on the image are Heating demand, heat loss, heat gain, internal heat, water heating, lighting, heat by cogeneration, and heat by PV. These values change once the user modifies the design model and import again to the model, based on this option the designer will be able to make sustainable decisions earlier in the design phase.
- What-if option: This option allows the user to determine the impact of his/her decision before being applied to the design model, based on this option the user can get an insight into the decision before being implemented to the design.

4.6 Tool's limitations

The axis orientation is determined by the three Cartesian points (x, y, z), which is included in the wall properties. Besides, the endpoint of the wall may not always be next to the initial point of the wall, so changing the axis directions involves equal values between the start and endpoints, and this detection is prevented due to the irregularity of axis direction, so the tool has confined the axis direction of the wall to the origin of the model.

Since the orientation of the IfcWall (south, north, west, and east) is absent in the IFC format, however, the orientation of a wall, is required for energy performance calculations, such as solar radiation. The direction of a wall can't be estimated by a particular equation. Instead, a fair approximation is based on three requirements: the position of the axis of a wall, the north direction is given, and the polarity of the total area. Hence the tool has assumed that if the wall is oriented to the North direction then its orientation is East, if the wall is oriented to the East direction then it's orientation direction is South, if the wall is oriented to the South direction then it's orientation direction is West and lastly if the wall is oriented to the West direction then it's orientation direction is North.

In addition, the time length required for energy calculations has been derived based on June, due to the large number of calculations used in the calculation part of the tool.

Moreover, the tool was developed based on a building model designed in Revit and constructed based on the IFC 2x3 specifications, this option may cause some errors if the imported IFC file is based on other versions of IFC specifications, as well as the property sets and attributes may differ if the model is designed in a different 3D design tool which may lead to the possibility of not extracting the required building properties, however, the tool is running efficiently if the model is designed in Revit.

Lastly, all the energy calculations are set based on NTA 8800 which is calculated based on the environment of the Netherlands, as well as due to the limited time of finalizing the graduation thesis, the analytical energy data are only inserted for a part of the building, these limited data may reduce the tool's validity.

4.7 Tool Script

The tool script section is further stated in Appendix V.

4.8 Test Case Results

A test case was performed to validate the tool. In order to direct the design decision-making, the test case describes the development of a 3D model in Revit, and its relation with the developed tool. A residential building model of 8 storeys was used to test the functionality of the integrated tool in Revit 2019. A 2.554 m2 double top swing window with a U value of 0.3 m2.K and a solid DEKO door with a U value of 1.3 m2.K were used. Besides, in the designed model, generic exterior walls of U value = 24.7 m2.K and generic basic interior walls of U value = 4.92 m2.K were used. In the building model, some data such as the lighting fixtures and the boilers were not supplied and were thus inserted manually. The designed model was then exported as an IFC file and imported to evaluate the energy efficiency of the designed model in the developed tool.

The figure below shows the energy results generated by the developed tool after clicking the energy results widget. Figure 26 shows the energy assessment results regarding the total energy consumption produced by the design model. Firstly, the feedback box shows the total energy consumption of the building with value equals 50862.41MJ, as well as it shows the energy consumed based on the building components, these building elements are only limited to the walls and openings as provided in the code script of the developed tool, based on the evaluated results, walls have consumed 1454.38 MJ/m2, while it's clearly shown that openings have consumed higher energy with values (4243.36, 5772.74, 5775.34, 6347.36 MJ/m2) for the north, east, south and west openings respectively. Moreover, as shown in the illustration box, the resulted HeatingDemand is 18625.32 MJ, HeatGain is 32237.09 MJ, HeatLoss is 14892 MJ,

SolarRadiation is 22138.80 MJ, WaterHeating is 8164.08 MJ, Lighting is 1934.21 MJ, HeatByCogeneration is 0.0 MJ, HeatByPv is 0.0 and finally the InternalHeat is 3732 MJ.

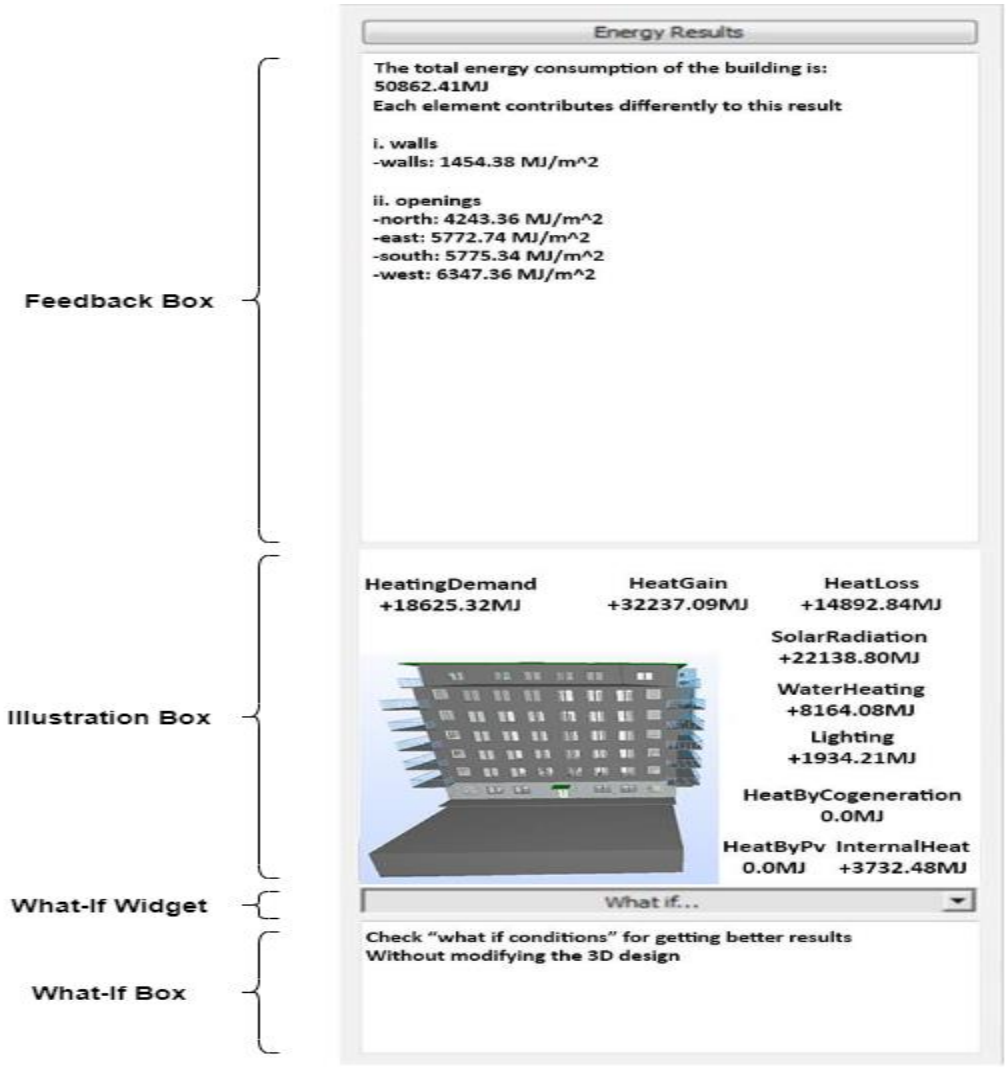

Figure 25 Energy results in the developed tool based on the designed model

To enhance the energy consumption by the building's components, some modifications have been made to the original model, some materials have been changed regarding the walls and openings with a reduced U value to decrease the energy consumption produced by walls and openings and the existed boilers have been exchanged with new boilers with higher efficiency values to reduce the energy consumption by hot tap water. A new IFC file has been exported to the developed tool to check the energy performance of the building model after modifying the design model. Figure 27 shows the new energy results after importing the new design model, as shown in the feedback box, the total energy consumption of the building has been decreased to

50632.16 MJ. Moreover, the energy consumption by walls has been reduced from 1454.38 MJ/m2 to 1423.16 MJ/m2 and the energy consumption by openings has been slightly decreased from (4243.36, 5772.74, 5775.34, 6347.36 MJ/m2) to (4242.16, 5643.43, 5775.14 and 6270.72 MJ/m2) for the north, east, south and west openings respectively. In addition, the illustration box shows a comparison between the old and the new results, the values with green colors show an improvement, while the red color shows a lapse. For instance, the modified model has shown better results compared to the previous assessment for HeatingDemand and WaterHeating, as the new result for HeatingDemand has been improved from 18625 MJ to 18215 MJ and the new result for WaterHeating shows that the energy consumption from hot tap water has been reduced from 8164.07 MJ to 7441.16 MJ. On the other hand, the energy consumption from SolarRadiation has been increased from 22138.80 MJ to 23042.44 MJ, while the other energy requirements have shown stable results.

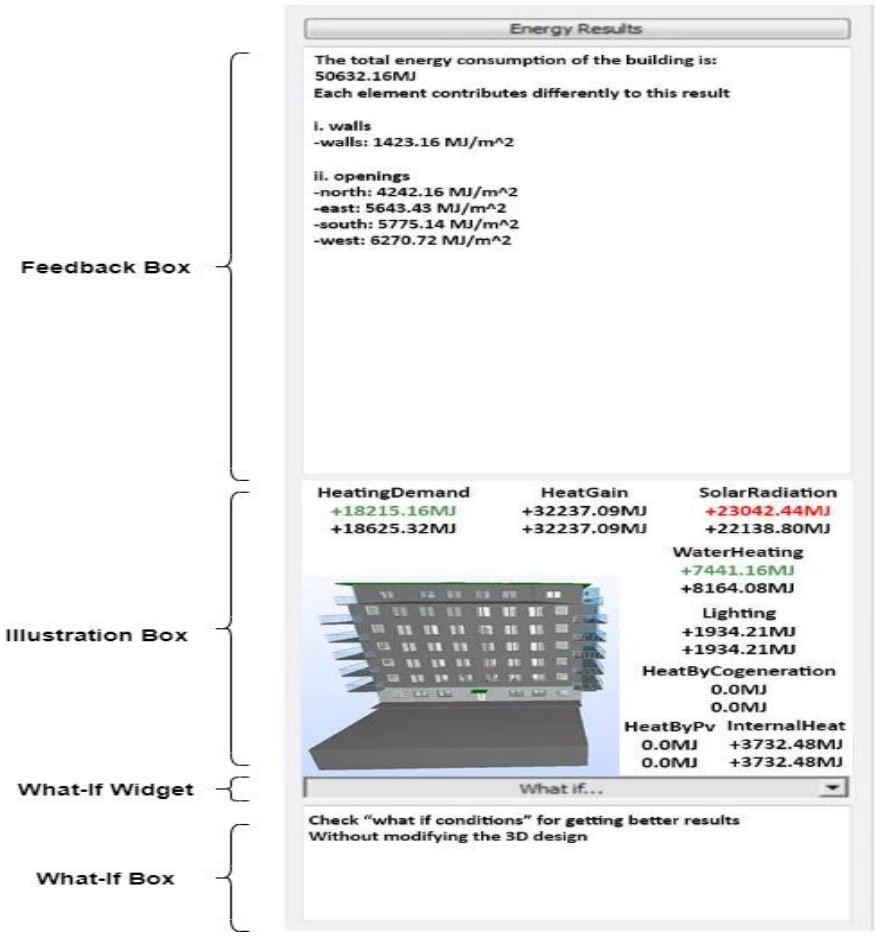

Figure 26 Energy results in the developed tool based on the modified design model

4.9 Tool Summary

Different steps have been followed in the tool development; these steps have been narrowed down as follow:

Step 1: Defining the building properties, these properties are different based on the data needed from each building element.

A) For Walls:

- If the wall is external or internal
- Start and endpoints
- Axis direction
- Wall's physical properties (Area, length, Width, Height)
- Wall orientation
- Thermal properties
- Global Id
- IfcGuid

B) For Openings:

- IfcGuid
- Global ID
- If the opening is Door or Window
- If the opening is external or internal
- Opening's orientation
- Physical properties (Area, Length, and Width)
- Thermal properties
- Average Effective Total Solar Accession Factor
- Cosine Fraction
- Shading Reduction Factor
- Incident Solar Radiation By Orientation
- Correction Factor For Non-Scattering Glazing
- Solar Accession Factor
- Collector Area

C) For Roof:

- Global ID
- Area
- Thermal Properties
- IfcGuid

- Roughness

D) For Floors:

- IfcGuid
- Global ID
- Area
- Thermal Properties
- Cbasins
- Cbath
- Ckitchen
- CPerPerson
- Cshower
- QstandCirculationLoss
- QstandingStill
- Tap
- Boiler Efficiency
- Correction Factor
- Number of baths per person per day
- Number of showers per person per day
- Huseful

E) For Spaces:

- Global Id
- IfcGuid
- Area
- Thermal Properties
- Number of people
- Number of living functions

F) For lighting sources:

- Total Wattage
- Specified lighting load
- Utilization factor

4.10 Future improvements

The method used in this study provides a way to bridge the lack of participation in the Building Performance Simulation tools in the early stage of the design. The steps that can be taken to strengthen this strategy and reduce the constraints include:

1. The data extraction process can be extended to include other IFC specifications.
2. In this thesis, only certain IFC entities are taken into account in the analysis, but other entities can be used to represent the building properties and the python script can be expanded further to include other entities.
3. Other internal data benefits can be applied to the enrichment BIM model and the python script extension can be introduced to extract those data. This will increase the precision of the effects of the simulation.
4. Taking into account how the building properties are mapped in the IFC formats related to different 3D design platforms.

Chapter 5: Conclusion

In this research study, an integrated decision support BIM-BPS tool that simulates the building's energy index for energy labeling earlier in the design stage is developed. Since the literature review has addressed the negative impact that is caused by the AEC industry on the environment, the developed tool seeks to reduce this negative impact by involving the integrated tool in the early design phases.

BPS tools have shown an effective impact on reducing energy consumption in buildings; however, the role of BPS tools is descriptive. The simulation tools provide insight into energy efficiency in this context, but they do not give any feedback to ensure that users make optimal decisions, and there comes the important role of BIM in overcoming this issue as BIM interaction with the BPS tools facilitated the possibility of sending feedback to the projects' participants which will help the designers to make better decisions in choosing the most suitable building parameters early in the design phase by sending comparisons of the current outcomes.

During this research several issues have been outlined, the main issue is the lack of an established methodology. Specifically, no structure explains how to develop integrated BIM-BPS tools, what their process is, or what kind of feedback they can generate. Therefore, only by assessing and criticizing existing integrated approaches and tools can the functionality and methodology of the BIM-BPS integration be found.

Besides, the developed tool was based on a building model designed in Revit and constructed based on the IFC 2x3 specifications, this option may cause some errors if the imported IFC file is based on other versions of IFC specifications, as well as the property sets and attributes may differ if the model is designed in a different 3D design tool which may lead to the possibility of not extracting the required building properties, hence, taking into account how the building properties are mapped in the IFC formats related to different 3D design platforms is an important aspect that the user must take into account before using the developed tool to prevent running errors. Also, some tool running issues may appear with large IFC models, so the smaller the IFC model the better the developed tool will perform.

Moreover, to perform the energy assessments, various values were extracted from the building model based on the building components, however, most of these values were not stored in the building model. Hence, the missing values were manually inserted in the building model based on the specifications of buildingSMART library.

Lastly, BIM has shown an effective impact during the decision making process, as once any modifications happen in the IFC model, it was easily detectable in the energy assessment outcomes in the developed tool. Therefore, BIM-BPS interaction has shown the ability to overcome the decision making issues stated in the literature review.

5.1 Research questions

The following section illustrates a discussion of the sub-questions; based on the sub-questions, the main research question is answered.

SQ.1 How can BIM contribute to building performance simulation of a building project during the conceptual phase?

The energy efficiency of buildings (BEP) is influenced by a dynamic interaction of several parameters, such as design, occupancy, and weather conditions. Building Information Modeling (BIM) enables design models to be created with integrated semantic information about each aspect of construction. The BIM data contains the geometric and thermal information of a building; these data are needed to be provided to the BPS tools to generated energy assessments. A various number of integrated BIM-BPS tools were developed by many researchers to generate sustainable designs; these tools receive data from the BIM models which have been disclosed in the Literature Review. The direct transfer of data solves problems of interoperability between BIM and BPS tools, enabling accurate designs, avoid human errors, and improve the collaboration between the project's participants.

SQ.2 How can BIM-BPS tools be early involved in the project and how this integration can affect the decision making process?

A user-friendly application was created to facilitate the usage of BPS tools in the early design stages, enabling designers to easily understand the building's thermal energy efficiency. In this research, the BIM-based BPS tool developed allows for an automated energy evaluation of the thermal energy efficiency of building elements during the early design phases, its functional criteria was defined by the literature review. In a way to inform designers about the workflow of the proposed BIM-BPS integration, a case study of a residential building design was presented. Regarding the impact of this approach on the decision making process, as the BIM-BPS integration will yield consistent outcomes quicker, from which a designer can assess and sustainably design the building efficiently. The developed tool can send feedback to the designer with the energy assessment of the designed model and provide him/her with What If options that assist the designer with possible modifications to the designed model earlier in the project phase. Therefore, sustainable decisions could be made faster and earlier in the design phase.

SQ.3 What is the added value to the industry by applying the BIM-Building Performance Simulation approach?

The Literature Review has shown that it is possible to boost the design process by incorporating BIM-BPS tools. In the early stages of the design process, simulation makes it possible to incorporate design changes by assumptive feedback and suggestive effect; the tool will start a

dialog with the architect that was not previously possible. A feedback loop arises between the architect and the developed tool by presenting relative prospects and relationships leading to an empowered design process. In addition, the construction project often requires multiple engineers of various backgrounds, the integration impact will be significant by incorporating BPS into the design phase as the communication between members of the project will be enhanced, and this integration will avoid loss of time and money over design errors.

MQ. How can the integration of 3D Building Information Models and the Building Performance Simulation approach be used to optimize the EI regarding the thermal energy performance in buildings?

Based on the previous sub-questions, it can be clearly shown that the BIM-BPS integration approach has shown effective results in optimizing the thermal energy performance in buildings. During the early design stages, the developed tool has facilitated the automation of the energy assessment of the thermal energy performance of building components. Moreover, via an assumptive input and suggestive influence, this incorporation makes it possible to implement design changes; the created tool has assisted the designers in the decision-making process that was not previously possible by providing relative prospects and relationships leading to an empowered design process, a feedback loop emerges between the architect and the created instrument. Hence, based on the mentioned prospective, the BIM-BPS integration approach has been successfully proved that its implementation can improve the AEC industry's impact on the environment and solve the issues facing the Architects in the construction project.

5.2 Scientific relevance

As stated in the literature review, the AEC industry is the main source of pollution towards the environment. The main scientific importance of this research is to reduce the negative impact caused by the AEC industry. User-friendly decision support BIM-based tool was created to facilitate the usage of BPS tools in the early design stages, enabling designers to easily understand the building's thermal energy efficiency. In this research, the BIM-based BPS tool developed allows for an automated energy evaluation of the thermal energy efficiency of building elements during the early design phases. The early involvement of BPS tools in the early design phase enables accurate designs, avoid human errors, and improve the collaboration between the project's participants. Moreover, BIM-BPS integration will yield consistent outcomes quicker, from which a designer can assess and sustainably design the building efficiently. The developed tool can perform energy assessments associated with feedbacks of the designed model and provide users with What If options that assist the designer with possible modifications to the designed model earlier in the project phase. Hence, sustainable decisions could be made faster and earlier in the design phase. Besides, in the early stages of the design process, simulation makes it possible to incorporate design changes by assumptive feedback and suggestive effect; the tool will start a dialog with the architect that was not previously possible. A feedback loop arises between the architect and the developed

tool by presenting relative prospects and relationships leading to an empowered design process.

5.3 Societal relevance

The social importance of this research arises from the environmental implications of the built environment, which require immediate intervention from the AEC industry's responsible stakeholders, to develop more effective and environmentally safe buildings for the population, by offering sustainable outcomes to the designers who can assess them and develop an effective and environmentally sustainable structure, the created tool promotes this action. In addition to the previously mentioned benefits, the tool is not limited to residential use only, but also could be used for different building functions.

5.4 Future recommendations

Based on the adopted methodology of the research, this section provides suggestions for the possibilities for further study. The suggested recommendations are presented based on the environmental impact aspects and the aspects of applications for BIM and Linked Data.

In future work, some broader aspects should be discussed, such as web integration, compliance with government building energy policies, a more detailed interface to other BIM applications, exchange of information from and to BPS tools should be addressed, compliance with IFC to facilitate OpenBIM standards, more direct information from and to BPS tools. Some potential issues should also be discussed, such as operational delay as these delays may cause issues to the integrated system.

In addition to the existing work of the developed tool, it may be useful to reduce the currently necessary human interaction required when implementing the instrument in operation. This could be developed by creating a plug-in to the BIMserver which can be a helpful addition. Creating such a plug-in will facilitate the access of feedback on the designed model, as the energy assessment’s results will be analyzed automatically when an IFC file is imported into the server. This user interface can be effective and useful when choosing preferences during data format conversion. , Moreover, as previously mentioned in the literature review, most of the currently available tools are complex to be used by the users and only easily usable by developers and researchers.

One path for further research will be to examine the most influential building parameters in the early phase of the project towards maximizing the efficiency of thermal energy. This requires an in-depth analysis of the structural aspect of the design to assign effective parameters that will serve the same function. This could be applied by specifying how uncertainty in output can be assigned to uncertainty in a process model's input parameters. Sensitivity analysis is useful to determine which parameters during model design require more attention and the input parameters that most affect the results of the simulation. Sensitivity analysis can assess the effect of the building materials and the number of occupants, so-called occupancy, on the room

temperature and incoming air ventilation temperature of a house. The methodology that could be used in this approach is to use Eppy, as it can be used In Python, a popular language with high-level features that support "one-liner" for many common programming idioms, Eppy offers low-level programmatic access to EnergyPlus inputs, e.g., extracting all objects that match a list of specifications (Eppy, 2018). For each corresponding EnergyPlus object, Eppy parses EnergyPlus IDF files and creates a Python object. Users will be able to modify object fields programmatically, remove objects, and construct new objects. Eppy writes the modified Python objects to a new IDF file for EnergyPlus.

Chapter 6: References

1. Abd Jamil, A. H., & Fathi, M. S. (2016). The integration of lean construction and sustainable construction: A stakeholder perspective in analyzing sustainable lean construction strategies in Malaysia. Procedia Computer Science, 100(1), 634-643.

2. Aghimien, D. O., Aigbavboa, C. O., & Thwala, W. D. (2019). Microscoping the challenges of sustainable construction in developing countries. Journal of Engineering, Design and Technology.

3. Agyekum-Mensah, G., Knight, A., & Coffey, C. (2012). 4Es and 4 Poles model of sustainability. Structural Survey.

4. Augenbroe, G., De Wilde, P., Moon, H. J., & Malkawi, A. (2004). An interoperability workbench for design analysis integration. Energy and buildings, 36(8), 737-748.

5. Attia, S. (2012). A tool for design decision making: zero energy residential buildings in hot humid climates. Presses univ. de Louvain.

6. Attia, S. (2010). Building performance simulation tools: selection criteria and user survey (No. 01/2010). Architecture et climat.

7. Attia, S. (2012). A tool for design decision making: zero energy residential buildings in hot humid climates. Presses univ. de Louvain.

8. Attia, S. (2011). State of the art of existing early design simulation tools for net zero energy buildings: a comparison of ten tools (No. 01/2011). Architecture et climat.

9. Attia, S., Gratia, E., De Herde, A., & Hensen, J. L. (2012). Simulation-based decision support tool for early stages of zero-energy building design. Energy and buildings, 49, 2-15.

10. Azar, Elie & Menassa, Carol. (2012). A comprehensive analysis of the impact of occupancy parameters in energy simulation of office buildings. Energy and Buildings. 55. 841–853. 10.1016/j.enbuild.2012.10.002.

11. Bazjanac, Vladimir & Crawley, Drury. (1997). The Implementation of Industry Foundation Classes in Simulation Tools for the Building Industry.

12. BuildingSMART - The Home of BIM. (2020, October 07). Retrieved November 03, 2020, from https://www.buildingsmart.org/

13. Borrmann, Andre & Beetz, Jakob & Koch, Christian & Liebich, T. & Muhic, Sergej. (2018). Industry Foundation Classes: A Standardized Data Model for the Vendor-Neutral Exchange of Digital Building Models: Technology Foundations and Industry Practice. 10.1007/978-3-319-92862-3_5.

14. Building Description. (n.d.). Retrieved from http://www.esru.strath.ac.uk/EandE/Web_sites/17-18/eebuilding/page9.html

15. Bazjanac, V., (2008), IFC BIM-Based Methodology for Semi-Automated Building Energy Performance Simulatio, Proceedings of the 25th International Conference on Information Technology in Construction, Available at: https://simulationresearch.lbl.gov/sites/all/files/919e.pdf

16. Ball, J. (2002). Can ISO 14000 and eco-labelling turn the construction industry green?. Building and environment, 37(4), 421-428.

17. Coakley, D., Raftery P. & Keane M., (2014), A review of methods to match building energy simulation models to measured data, Renewable and Sustainable Energy Reviews 37 (pp. 123–141), https://doi.org/10.1016/j.rser.2014.05.007

18. Crawley, D. B., Hand, J. W., Kummert, M., & Griffith, B. T. (2008). Contrasting the capabilities of building energy performance simulation programs. Building and environment, 43(4), 661-673.

19. Crawley, D. B., Hand, J. W., Kummert, M., & Griffith, B. T. (2008). Contrasting the capabilities of building energy performance simulation programs. Building and environment, 43(4), 661-673.

20. Connor, N. (2019, June 04). What is Heat Transfer - Definition. Retrieved from https://www.thermal-engineering.org/what-is-heat-transfer-definition/

21. Connor, N. (2019, June 04). What is Thermal Conduction - Heat Conduction - Definition. Retrieved from https://www.thermal-engineering.org/what-is-thermal-conduction-heat-conduction-definition/

22. Chen, Z., Li, H., & Hong, J. (2004). An integrative methodology for environmental management in construction. Automation in Construction, 13(5), 621-628.

23. Du Plessis, C. (2002). Agenda 21 for sustainable construction in developing countries. CSIR Report BOU E, 204, 2-5.

24. Döllner, J., & Hagedorn, B. (2007). Integrating urban GIS, CAD, and BIM data by servicebased virtual 3D city models. R. e. al.(Ed.), Urban and Regional Data Management-Annual, 157-160.

25. Experts, B. (2020, September 16). IFC (Industry Foundation Classes) and BIM Interoperability. Retrieved from https://www.e-zigurat.com/blog/en/ifc-and-bim-interoperability/

26. EnergyPlus. (2007). "EnergyPlus, Getting started with Energy Plus."http://www.eere.energy.gov/buildings/energyplus/documentation.html

27. Eck, H. V. (2018, May 22). Implementation of the EPBD in The Netherlands. Retrieved November 04, 2020, from https://epbd-ca.eu/ca-outcomes/outcomes-2015-2018/book-2018/countries/netherlands

28. Ellis, M. W., & Mathews, E. H. (2002). Needs and trends in building and HVAC system design tools. Building and environment, 37(5), 461-470.

29. Ghaffarianhoseini, A., Tookey, J., Ghaffarianhoseini, A., Naismith, N., Azhar, S., Efimova, O., & Raahemifar, K. (2017). Building Information Modelling (BIM) uptake: Clear benefits, understanding its implementation, risks and challenges. Renewable and Sustainable Energy Reviews, 75, 1046-1053.

30. Gkatzios. (2018). Integrating building information modeling and building performance simulation: A BIM based simulation tool generating automated building energy performance analysis with explanatory feedback at the early design stages. Eindhoven University of Technology. Retrieved from https://research.tue.nl/en/studentTheses/integrating-building-information-modeling-and-building-performance

31. Hui, S. C., & Law, M. A. Y. (2002). Green Design and Construction of Site Offices. Research Report for Gammon-Skanska Limited, Department of Mechanical Engineering, The University of Hong Kong, Hong Kong (http://web. hku. hk/~ cmhui/reportgreen_site_offices. pdf).

32. Hall, M., & Purchase, D. (2006). Building or bodging? Attitudes to sustainability in UK public sector housing construction development. Sustainable Development, 14(3), 205-218.

33. Hu, Z. Z., Zhang, J. P., Yu, F. Q., Tian, P. L., & Xiang, X. S. (2016). Construction and facility management of large MEP projects using a multi-Scale building information model. Advances in Engineering Software, 100, 215-230.

34. Hensen, J. L., & Lamberts, R. (Eds.). (2012). Building performance simulation for design and operation. Routledge.

35. Hong, T., Langevin, J., & Sun, K. (2018, October). Building simulation: Ten challenges. In Building Simulation (Vol. 11, No. 5, pp. 871-898). Tsinghua University Press.

36. Hong, T., Chou, S. K., & Bong, T. Y. (2000). Building simulation: an overview of developments and information sources. Building and environment, 35(4), 347-361.

37. Hirsch, J. J. (2006). eQUEST, the quick energy simulation tool. DOE2. com.

38. Ismail, F. Z., Halog, A., & Smith, C. (2017). How sustainable is disaster resilience?. International Journal of Disaster Resilience in the Built Environment.

39. Ing, C. A. M. D. (2003). Towards the Integration of Simulation into the Building Design Process.

40. Jiang, S., Jiang, L., Han, Y., Wu, Z., & Wang, N. (2019). OpenBIM: An Enabling Solution for Information Interoperability. Applied Sciences, 9(24), 5358.

41. Jeong, W., & Kim, K. H. (2016). A performance evaluation of the BIM-based object-oriented physical modeling technique for building thermal simulations: A comparative case study. Sustainability, 8(7), 648.

42. Jin, Ruoyu & Zhong, Botao & Ma, Ling & Hashemi, Arman & Ding, Lieyun. (2019). Integrating BIM with building performance analysis in project life-cycle. Automation in Construction. 106. 102861. 10.1016/j.autcon.2019.102861.

43. Kamar, K. M., Ismail, E., & Ismail, E. (2010). Abd. Hamid Z, Egbu C, Arif M., Mohd Zin, Mohd. MZN Ghani. K, Rahim. AH," Sustainable and Green Construction", Construction Industry Development Board (CIDB), Malaysia.

44. Khasreen, M. M., Banfill, P. F., & Menzies, G. F. (2009). Life-cycle assessment and the environmental impact of buildings: a review. Sustainability, 1(3), 674-701.

45. Kibert, C. J. (2016). Sustainable construction: green building design and delivery. John Wiley & Sons.

46. Koranda, C., Chong, W. K., Kim, C., Chou, J. S., & Kim, C. (2012). An investigation of the applicability of sustainability and lean concepts to small construction projects. KSCE Journal of Civil Engineering, 16(5), 699-707.

47. Kravari, M.N., (2017), Environmental impact assessment of building materials in BIM models Computation of embodied carbon in building materials with the use of Industry Foundation Classes and Semantic Web technologies, Eindhoven University of Technology, Available at: https://www.ofcoursecme.nl/?mdocs-file=3806

48. Lu, Q., Lee, S., & Chen, L. (2018). Image-driven fuzzy-based system to construct as-is IFC BIM objects. Automation in Construction, 92, 68-87.

49. Loonen, R. C., Favoino, F., Hensen, J. L., & Overend, M. (2017). Review of current status, requirements and opportunities for building performance simulation of adaptive facades. Journal of Building Performance Simulation, 10(2), 205-223.

50. Lam, P. T., Chan, E. H., Chau, C. K., Poon, C. S., & Chun, K. P. (2011). Environmental management system vs green specifications: How do they complement each other in the construction industry?. Journal of Environmental Management, 92(3), 788-795.

51. Liao, Chenda & Lin, Yashen & Barooah, P.. (2012). Agent-based and graphical modelling of building occupancy. Journal of Building Performance Simulation. 5. 5-25. 10.1080/19401493.2010.531143.

52. Morledge, R., & Jackson, F. (2001). Reducing environmental pollution caused by construction plant. Environmental Management and Health.

53. Maile, T., Fischer, M., & Bazjanac, V. (2007). Building energy performance simulation tools-a life-cycle and interoperable perspective. Center for Integrated Facility Engineering (CIFE) Working Paper, 107, 1-49.

54. Morbitzer, C., Strachan, P. A., Spires, B., Cafferty, D., & Webster, J. (2001). Integration of building simulation into the design process of an architectural practice.

55. Ndlangamandla, M. G., & Combrinck, C. (2019). Environmental sustainability of construction practices in informal settlements. Smart and Sustainable Built Environment.

56. Naamane, A., & Boukara, A. (2015). A Brief Introduction to Building Information Modeling (BIM) and its interoperability with TRNSYS. Renewable Energy and Sustainable Development, 1(1), 126-130.

57. Negendahl, K., (2015), Building Performance Simulation in the early design stage: An introduction to integrated dynamic models, Automation in Construction 54 (pp.39-53), https://doi.org/10.1016/j.autcon.2015.03.002

58. Oke, A. E., Aigbavboa, C. O., & Semenya, K. (2017). Energy savings and sustainable construction: examining the advantages of nanotechnology. Energy Procedia, 142, 3839-3843.

59. Ochoa, C. E., & Capeluto, I. G. (2009). Advice tool for early design stages of intelligent facades based on energy and visual comfort approach. Energy and buildings, 41(5), 480-488.

60. Pulaski, M. H., Horman, M., Riley, D., Dahl, P., Hickey, A., Lapinski, A., ... & Holland, N. (2004). Field guide for sustainable construction. Pentagon Renovation and Construction Program Office Safety Sustainability and Environment IPT and Washington Headquarters Services Defense Facilities Directorate, USA.

61. PROBERT, A. J. et al. Accounting for the Life Cycle Carbon Emissions of New Dwellings in the UK. In: INTERNATIONAL CONFERENCE ON NONCONVENTIONAL MATERIALS AND TECHNOLOGIES, 12., Cairo, 2010. Proceedings... Cairo, 2010.

62. Park, J., Cai, H., Dunston, P. S., & Ghasemkhani, H. (2017). Database-supported and web-based visualization for daily 4D BIM. Journal of Construction Engineering and Management, 143(10), 04017078.

63. Punjabi, S. A. (2005). Development of an integrated building design information interface (Doctoral dissertation, Texas A&M University).

64. Pelken, P. M., Zhang, J., Chen, Y., Rice, D. J., Meng, Z., Semahegn, S., ... & Ling, F. (2013, September). "Virtual Design Studio"—Part 1: Interdisciplinary design processes. In Building Simulation (Vol. 6, No. 3, pp. 235-251). Tsinghua University Press.

65. Petersen, S., & Svendsen, S. (2010). Method and simulation program informed decisions in the early stages of building design. Energy and buildings, 42(7), 1113-1119.

66. Paone, A., & Bacher, J. P. (2018). The impact of building occupant behavior on energy efficiency and methods to influence it: A review of the state of the art. Energies, 11(4), 953.

67. Penttilä, H., Rajala, M., & Freese, S. (2007). Building information modelling of modern historic buildings.

68. Qian, A. Y. (2012). Benefits and ROI of BIM for Multi-disciplinary Project Management. National University of Singapore, Mar.

69. Shurrab, J., Hussain, M., & Khan, M. (2019). Green and sustainable practices in the construction industry. Engineering, Construction and Architectural Management.

70. Smiciklas, J., De Jager, L., Cucchietti, F., Zeddam, A., & Ali, M. (2012, September 18). Sustainable Buildings. Retrieved June 25, 2020, from https://www.itu.int/oth/T0B1100001B

71. Schwegler, B. (2010). Green BIM: successful sustainable design with building information modeling by Eddy Krygiel and Bradley Nies. Journal of Industrial Ecology, 14(5), 859-860.

72. Solihin, W., Eastman, C., & Lee, Y. C. (2017). Multiple representation approach to achieve high-performance spatial queries of 3D BIM data using a relational database. Automation in Construction, 81, 369-388.

73. SB, Mardiana. (2015). Building Energy Consumption and Carbon dioxide Emissions: Threat to Climate Change. Journal of Earth Science & Climatic Change. s3. 10.4172/2157-7617.S3-001.

74. Tam, V. W., Tam, C. M., Zeng, S. X., & Chan, K. K. (2006). Environmental performance measurement indicators in construction. Building and environment, 41(2), 164-173.

75. Umar, U. A., Tukur, H., Khamidi, M., & Alkali, A. U. (2013). Impact of environmental assessment of green building materials on sustainable rating system. In Advanced Materials Research (Vol. 689, pp. 398-402). Trans Tech Publications Ltd.

76. Vanlande, R., Nicolle, C., & Cruz, C. (2008). IFC and building lifecycle management. Automation in construction, 18(1), 70-78.

77. V. Bazjanac, T. Maile, C. Rose, J.T.O. Donnell, E. Morrissey, B.R. Welle, AN ASSESSMENT OF THE USE OF BUILING ENERGY PERFORMANCE SIMULATION IN EARLY DESIGN, Proceedings of Building Simulation 2011: 12th Conference of International Building Performance Simulation Association, 2011, pp.1579-1585.

78. Zabihi, Hossein & Habib, Farah & Mirsaeedie, Leila. (2012). Sustainability in Building and Construction: Revising Definitions and Concepts. International Journal of Emerging Science. 2. 570-578.

79. Zabihi, H., & Habib, F. (2012). Sustainability in building and construction: revising definitions and concepts. International Journal of Emerging Sciences, 2(4), 570.

Appendix

Appendix I: Energy Calculations

Energy Index Equation as stated in NTA 8800 (Equation 1):

$$\mathbf{EI} = \frac{\mathbf{Qtotal}}{(\mathbf{155} \times \mathbf{Afloor}) + \mathbf{9560}} \quad \mathbf{(1)}$$

The total heat equation is stated below (Equation 2) Based on NTA 8800:

$$\mathbf{Qtotal} = \mathbf{Qspace\ heating} + \mathbf{Qwater\ heating} + \mathbf{Qaux.energy} + \mathbf{Qlighting} - \mathbf{Qpv} - \mathbf{Qcogeneration} \quad \mathbf{(2)}$$

Space heating calculation:

The space heating requirement is calculated as stated below in Equation 3

$$\mathbf{Qspace\ heating} = \frac{\frac{\mathbf{Qspace\ heat\ demand}}{\eta \mathbf{dist.system}} - \mathbf{Qsolar\ boiler}}{\eta \mathbf{installation}} + \mathbf{Qpilot\ flame} \quad \mathbf{(3)}$$

Space heat demand Calculation:

Equation 3.1 shows the calculation of the space heat demand as stated in NTA 8800.

$$\mathbf{Qspace\ heat\ demand} = \mathbf{Qtransmission\ loss} + \mathbf{Qventilation\ loss} - \mathbf{Qinternal\ gain} - \mathbf{Qsolar\ gain} \quad \mathbf{(3.1)}$$

Heat transfer through transmission Calculation:

Equation 3.1.1 shows the heat loss through transmission based on NTA 8800.

$$\mathbf{Qtransmission\ loss} = \mathbf{A} \times \mathbf{i} \times (\mathbf{Tin} - \mathbf{Tout}) \times \mathbf{0.001t} \quad \mathbf{(3.1.1)}$$

Where:

Q = heat transmission
A = area of each surface
i = Heat transfer coefficient (1.8 W/k)
Tin = inside temperature (22° C)
Tout = outside temperature
t = calculated value of the length of time in ms

Heat transfer through Ventilation Calculation:

Equation 3.1.2 shows the heat loss through transmission based on NTA 8800.

$$\mathbf{Qvent} = \mathbf{R} \times (\mathbf{tin} - \mathbf{tout}) \times \mathbf{0.001t} \qquad \textbf{(3.1.2)}$$

Where:

Q = heat transmission =1.296 kwh
R = ventilation heat transfer coefficient
tin = inside temperature
tout = outside temperature
t: time period length,h

Ventilation heat transfer coefficient Calculation:

$$\mathbf{R} = \boldsymbol{\rho}\mathbf{a} \times \mathbf{ca} \times \frac{\sum \mathbf{b} \times \mathbf{f} \times \mathbf{q}}{\mathbf{3600}} \qquad \textbf{(3.1.2.1)}$$

Where:

R: the total heat transfer coefficient by ventilation for heating for calculation zone (W / K)= 0.3
ρ a : the density of air, = 1.205 kg / m 3;
c a: the specific heat capacity of air, = 1.005 J / kgK;
b: the correction factor for supply temperature for air volume flow k=1
f: the dynamic correction factor for air volume flow k, with numerical value= 1;
q: air volume flow k, in m3 / h, the effective air volume flow due to natural supply of ventilation = 1

Internal Heat Gain Calculation:

$$\mathbf{Q} = \mathbf{180} \times \mathbf{Nliving} \times \mathbf{Np} \times \mathbf{0.001t} \qquad \textbf{(3.1.3)}$$

Where:

Q: the internal heat production
Nliving: the number of residential functions in the calculation zone
Np: the average number of residents per calculation zone per residential function
t: calculated value of the length of time in ms

Heat Gain by solar radiation Calculation:

$$\mathbf{Qsolar} = \mathbf{g} \times \mathbf{A} \times (\mathbf{1} - \mathbf{F1}) \times \mathbf{Fsh} \times \mathbf{I} \times \mathbf{0.001\ t} \quad \textbf{(3.1.4)}$$

Where:

g: dimensionless average effective total solar accession factor of windows per month
A: the effective collector area of the surface= 3.96 m2
F1: the cosine fraction of windows =0.25

Fsh: the dimensionless shading reduction factor of the external surface
I: the incident solar radiation of the surface determined by its orientation and slope
t: the calculated value of the length of time in ms

Calculating the total solar entry factor (corrected for the angle of incidence) (g):

This calculation has considered the following properties:

- Non-scattering glazing windows
- Double glazing windows
- Glass only

$$\mathbf{g = Fw \times Ggl} \qquad \textbf{(3.1.4.1)}$$

Where:

g: is the total solar entry factor (corrected for the angle of incidence)
$F_{w:}$ is a correction factor for non-scattering glazing, for which the following numerical value applies: Fw = 0.90;
$G_{gl:}$ is the solar accession factor at the perpendicular incidence of solar radiation =0.75

Water Heating:

$$\mathbf{Qwater\ heating = Qmain\ boiler + Qauxilary\ heat\ water\ (4)}$$

$$\mathbf{Qmain\ boiler = \frac{cf \times TAP}{\eta boiler} \times rtap + Qstanding\ still + Qstand\ circulation\ loss \times \frac{Afloor}{100} \times (1 - \eta useful)} \quad \textbf{(4.1)}$$

Where:

cf: conversion factor (MJ · day/l · year)
TAP: quantity of water (l · day)
η$boiler$: boiler efficiency − 0.9 in case of a condensing boiler
rtap : correction factor for short piping − 0.9 if < 5m, else 1
Qstanding still: 4220,2MJ in case of a condensing boiler
Qstand circulation loss: 10000MJ if non insulated, 4000MJ if insulated
ηuseful: used part of the loss (0.44)

$$\mathbf{TAP = Ckitchen + Cbasins + Npeople(Cper\ person + (Cshower \times F \times D) + (Cbath \times B \times B(yes\ or\ no))} \quad \textbf{(4.1.1)}$$

Where:

Ckitchen: 13.03 for a condensing boiler [l · day]
Cbasins: 3.97 for a condensing boiler [l · day]
Cper person: 7.1 for a condensing boiler [l · day]
Cshower: 20.8 for a condensing boiler [l · day]
F: saving shower head, if present 0.9 else 1
D: number of showers /person/day − 0.61
Cbath: 41.5 in case of condensing boiler [l · day]
B: number of baths /person/day − 0.096
B(yes or no): the presence of bath, if present 1 else 0

Auxiliary Heat Water Calculations:

$$\mathbf{Qaux\ heat\ water} = \mathbf{Qaux; ngen; si; mi+ Qaux;sto;si;mi+ Qaux; gen; si; gi; mi+ Qaux; soli; tot; ; si; mi\ (4.2)}$$

Where:

Qaux heat water: the amount of electrical auxiliary energy used for domestic hot water, in months in kWh.
Q aux, ngen, si, mi: the amount of electrical auxiliary energy used by the distribution system for domestic hot water, in system si, in month mi, in kWh
Q aux; sto, si, mi: the electrical auxiliary energy use for any circulators in the storage circuit charging circuit tough in system si, in month mi, in kWh = 0
Q aux; gene, si, gi, mi: the amount of electric auxiliary energy used for domestic hot water, for months mi, system si and the generator of this system gi, in kWh = 856 KWH
Q aux; soli, tot, si, mi: the total amount of electrical auxiliary energy taken from the solar energy system on its own plot soli for space heating and domestic hot water in system si, in month mi, in kWh.

Calculating the amount of electrical auxiliary energy used by the distribution system:

$$\mathbf{Q\ aux, ngen, si, mi} = \mathbf{Wdist.} \times \mathbf{Wconv.} \qquad \mathbf{(4.2.1)}$$

Where:

Q aux, ngen, si, mi: the amount of auxiliary electrical energy used by the distribution system for hot tap water, for month mi, (excluding the auxiliary energy for the heat generation), in kWh; = 35.77
Wdist.: the electrical auxiliary energy use of the distribution system for any available circulation pumps, in month mi, in kWh.
Wconv.: the electrical auxiliary energy use of an individual hot delivery set tap water, in a month in kWh.

$$\mathbf{Wdist.} = \boldsymbol{Pdist} \times \boldsymbol{\beta dist.} \times \boldsymbol{tcirc.} \times \boldsymbol{Fcorr.} \qquad \textbf{(4.2.1.1)}$$

In which:

P dis.: the hydraulic power of the circulation system pump, = 0.01 KW
βdis.: the partial load of the circulation system; = 1
Tcirc: the time that the circulation system is in operation for the distribution of heat for only hot water, in month mi, in h.
Fcorr.: the correction factor for the design conditions of the circulation system (f HB× f special) = 1.15

In which:

f HB: the correction factor for hydraulic balancing of circulation system si, with a fixed value f HB = 1.15
f special: a special factor in system design; f special, si = 1.

The electrical auxiliary energy use of an individual (Wconv.):

$$\mathbf{Wconv} = \frac{\mathbf{Pw}}{\mathbf{1000}} \times \mathbf{t} \qquad \textbf{(4.2.1.2)}$$

Where:

Wconv.: the electrical auxiliary energy use of an individual hot delivery set tap water
Pw: it is the auxiliary energy use of electronic delivery set by the individuals during standby of the generating device = 10 W
t: the calculated value of the length of time in h

The auxiliary energy consumption is calculated according to the following formula:

$$\mathbf{Q\ aux;\ soli, tot, si, mi} = \mathbf{Psol.} \times \frac{\mathbf{taux.sol.}}{\mathbf{1000}} \qquad \textbf{(4.2.2)}$$

Where:

Q; aux; sol, mi [kWh] is the auxiliary energy use for the contribution of the solar water heater to the energy consumption for hot tap water, in month mi;
Psol. [W] is the effective capacity of the collector pump for domestic hot water, in month mi

$$\mathbf{Psol.} = \mathbf{fuse.} \times \mathbf{psol.\,pmp} \qquad \textbf{(4.2.2.1)}$$

Where:

F use.: is the dimensionless factor for hot tap water for distribution of the solar energy system between hot tap water and heating = 1

Psol. pmp: the power of the collector pump

$$\mathbf{Psol.pmp} = \mathbf{Psol.pmp.std} + \mathbf{Psol.pmp.area} \times \mathbf{Asol.mod.} \times \mathbf{Ncol.} \quad \textbf{(4.2.2.1.1)}$$

Where:

Psol. pmp.std: fixed factor = 10W
Psol.pmp.area: fixed value = 1 m2
Asol.mod.: the area of the collector pump
Ncol: the number of used collectors
t aux.sol. [h]: the monthly operating time of the collector pump, calculated according to formula (4.2.2.2).

$$\mathbf{taux.sol} = \frac{\mathbf{Isol.}}{\sum_{\mathbf{12}}^{\mathbf{m=1}} \mathbf{Isol.}} \times \mathbf{taux.sol} \qquad \textbf{(4.2.2.2)}$$

Where:

Sol [W / m 2]: the striking solar radiation in the month
t aux [h]: the annual operating time of the collector pump, fixed value t aux = 2000 h

Auxiliary energy:

$$\mathbf{Qaux} = \mathbf{W.H\ aux;\ del.} + \mathbf{W.H\ aux\ Dist} + \mathbf{W.H\ aux.sto.} + \mathbf{W.H\ aux\ gen.} \quad (\mathbf{5})$$

Where:

W. H; aux; del.: the auxiliary energy of the related delivery system.

W. H; aux. Dist.: the auxiliary energy of the related distribution system.

W. H; aux. sto.: the auxiliary energy of the related storage system.

W. H; aux. gen.: the auxiliary energy of the related generating system.

The auxiliary energy of the related delivery system's calculation:

The following equation (Equation 5.1) determines the auxiliary energy for improved delivery in the room (including the fan, regulation, and control of a delivery system in a room such as a fan convector).

$$\mathbf{W.H;\ aux;\ del.} = \mathbf{Wfan} \qquad \textbf{(5.1)}$$

In which:

W. H; aux; del.: the auxiliary energy of the related delivery system in (KWH)= 0.2 Kwh

Wfan: The energy consumption of fans in (KWH)

$$\mathbf{Wfan} = \frac{\sum \mathbf{Pfan} \times \mathbf{nfan} \times \mathbf{tfan}}{\mathbf{1000}} \qquad \textbf{(5.2)}$$

In which:

Pfan: the electrical power of the fans in the calculation zone (W) =10
nfan: the total number of fans in rooms =42
tfan: the operating time of the system per month in (h)

$$\mathbf{tfan} = \mathbf{tH} \times \mathbf{fH\ red.} \times \mathbf{fH\ op.} \quad \textbf{(5.3)}$$

In which:

tH: is the time in a calculation zone in which the outside air temperature is lower than the heating limit

$$\boldsymbol{fH\ red.} = \mathbf{1} - \boldsymbol{fH\ red.Day} - \boldsymbol{fH\ red.weekend} \qquad \textbf{(5.4)}$$

In which:

fH red.: is the reduction factor for discontinuous heating= 0.6
fH red. Day: the relative part of the time (day) with reduced setpoint for heating
fH red. Weekend: the relative part of the time (weekend) with reduced setpoint for heating

$$\boldsymbol{fH\ red.Day} = \frac{\boldsymbol{tH\ red.day} \times \left(7 - \frac{\boldsymbol{tH\ red.weekend}}{24}\right)}{24 \times 7} \quad \textbf{(5.5)}$$

$$\boldsymbol{fH\ red.weekend} = \frac{\boldsymbol{tH\ weekend}}{24 \times 7} \quad \textbf{(5.6)}$$

In which:
tH red. Day: the number of hours per day with reduced setpoint temperature for heating
tH red. Weekend: the total number of hours per weekend with reduced setpoint temperature for heating

Heat Gain by Lighting:

$$\boldsymbol{QL} = \boldsymbol{Wl} + \boldsymbol{Wp} \textbf{ (6)}$$

$$\boldsymbol{Wl} = \boldsymbol{Wl;spec} \times \boldsymbol{Ag} \textbf{ (6.1)}$$

$$\boldsymbol{Wp} = \mathbf{0} \textbf{ (6.2)}$$

Where:

Wl: is the energy requirement for lighting to provide the necessary lighting levels per year, in kWh;
Wl; spec: is the specific energy requirement for lighting to provide the necessary lighting levels, in kWh / m2. For the calculation of the energy performance indicators of a building, a

value of 5 kwh/m2 is used for this
Ag: is the area of use of the considered calculation zone in m2
Wp: is the energy requirement for parasitic power, in kWh.

Heat Gain by PV cells:

$$\mathbf{Qpv = fbuilding \times fsol. \times Q_H;\ ren, si, mi\ \times fPVT; th} \quad \textbf{(7)}$$

Where:

Qpv: is the energy gained by solar panels

fbuilding: is the dimensionless ratio between the area of use of the building part for which the energy performance for the hot tap water function is determined and the area of use of the building as a whole that is wholly or partly connected to the collective building system si for the hot tap water function = 1

fsol.: is the dimensionless correction factor for the yield of the solar water heating system under practical conditions, with the fixed value $f_{prac;\ sol} = 0.95$

$Q_{H;\ ren,\ si,\ mi}$: is the amount of energy for space heating, in month mi, supplied by thermal renewable energy generated on its own plot by solar energy system soli, for system si, determined in kWh

$f_{PVT;\ th}$: is the thermal reduction factor for PVT systems if there is no PVT system or if the PVT system has been tested in accordance with NEN-EN-ISO 9806, then fPVT applies; th = 1.

Heat Gain by Cogeneration:

$$\mathbf{Qcogenereation} = \sum \mathbf{Eel;\ CHP\ ; for\ heating\ system} + \sum \mathbf{Eel;\ CHP\ ; for\ hot\ tap\ water} \quad \textbf{(8)}$$

Where:

Qcogeneration: the electricity produced on their own plot is connected to all buildings cogeneration plants for heating and hot tap water, in kWh

$Eel;\ CHP;\ for\ heating\ system$: The electricity produced on the own plot comes from the generator for each heating system, in kWh

Eel; chp; for hot tap water: The electricity produced on the own plot from the generator for each system for hot tap water, in kWh

For calculating the electricity produced for each heating system:

$$\mathbf{Eel;\ CHP\ ;\ for\ heating\ system} = \mathbf{QH\ gen;gi;mi;out} \times \frac{\boldsymbol{\varepsilon\ chp;el;si}}{\boldsymbol{\varepsilon\ chp;th;si}} \quad \textbf{(8.1)}$$

Where:

Q_H; gen; gi; mi; out: is the amount of thermal energy supplied by the cogeneration plant gi = chp, for the heating energy function, in months mi, in kWh,= 3454 kwh

ε chp; el; si: is the dimensionless annual average electrical conversion figure of the relevant cogeneration installation gi = chp, for each system si for heating, at the upper value

ε chp; th; si: is the dimensionless annual mean thermal conversion number of the relevant cogeneration installation gi = chp, for each system si for heating, at the upper value

For calculating the electricity produced for hot tap water:

$$\mathbf{Eel;\ CHP\ ;\ for\ hot\ tap\ water} = \mathbf{QW\ gen;gi;mi;out} \times \frac{\boldsymbol{\varepsilon\ chp;el;si}}{\boldsymbol{\varepsilon\ chp;th;si}} \textbf{(8.2)}$$

In which:

Eel; CHP; for hot tap water: The electricity produced on the own plot comes from the generator, for each system si for domestic hot water, in kWh

QW; gen; gi; mi; out: is the amount of thermal energy supplied by the cogeneration plant gi = chp, for the tap water energy function, in month mi, in kWh.

ε chp; el; si: is the dimensionless annual average electrical conversion figure of the relevant cogeneration installation gi = chp, for each system si for tap water, at the upper-value

ε chp; th;si: is the dimensionless annual mean thermal conversion number of the relevant cogeneration installation gi = chp, for each system si for tap water, at the upper value

$$\mathbf{QW;\ gen;\ gi;\ mi;\ out} = \mathbf{fbuilding} \times \mathbf{Ffunc.} \times \mathbf{Pnom.} \times \mathbf{tmi} \quad \textbf{(8.2.1)}$$

In which:

Q W; gene; gi, mi : the maximum amount of energy that can be delivered by the device based on assets, in months mi, in kWh

f building: the dimensionless ratio between the usable area of the building part for which the energy performance for the hot tap water function is determined and the usable area of the building as a whole that is fully or partially connected to the collective building system si for the domestic hot water function

Ffunc.: the dimensionless time fraction is that generator gi maximum operation for domestic hot water. For devices in large systems (a g, si> 500 m 2) applies f func = 0.6. This applies to all other devices f func = 1.0;

P. nom is the nominal power of the appliance as stated by the supplier or as stated on the nameplate, in kW; =7Kw for hot water

t mi is the calculation value for the length of the considered month mi, determined in h;

Appendix II: Tables & Figures

Table 7 heat transfer coefficient for heat loss through transmission (source: NTA 8800, 2020)

Pipe Type	Heat transfer coefficient
Uninsulated Pipe through the thermal shell	1.8
Insulated Pipe through the thermal shell	0.5
no penetrations through the thermal shell	0

Table 8 Time length of the Month (source: NTA 8800, 2020)

Month	Time length of the Month (h)
January	744
February	672
March	744
April	720
May	744
June	720
July	744
August	744
September	720
October	744
November	720
December	744

Table 9 Ventilation transfer coefficient values based on the building functions (source: NTA 8800, 2020)

Natural Ventilation	Residential function	Other used functions
Standard	1	0.95
air pressure controlled supply $\Delta <= 1Pa$	0.85	0.81
air pressure controlled supply 1 Pa < Δp <= 5 Pa	0.9	0.85
air pressure controlled supply 1 5 Pa < Δp <= 10 Pa	0.93	0.88

Table 10 Fixed values for the total solar access factor at perpendicular incidence, Ggl for common types of glazing (source: NTA 8800, 2020)

Type	ggl
Only glass	0.85
Double glass	0.75
Double glass with spectral (low) selective and low-emissive coating (HR ++)	0.60
Triple glass without or with one spectral (low) selective and low-emissive coating	0.50
Triple glass with two spectral (low) selective and low emissive coatings	0.40
Single glass with single glass front window or rear window without coating	0.75

Table 11 the time that the circulation system is in operation for the distribution of heat water (source: NTA 8800, 2020)

Months	Tcirc. (h)
December t/m February	0
March/ April/ October and November	0.6t
May t/m September	Tcirc= t

Table 12 Fixed values for electrical power of fans for air circulation in the space (source: NTA 8800, 2020)

Fan Decisive Properties	Power (W)
Fan coil unit	10
Electric heating	10
Local dynamic heat storage	12

Table 13 The number of hours per day and weekend with reduced set point temperature for heating (source: NTA 8800, 2020)

Building function	The number of hours per day with a reduced set point temperature for heating (h)	the total number of hours per weekend with reduced set point temperature for heating (h)
Day-care	14	48
Other meeting function	14	48
Cell Function	8	0

Healthcare function with bed area	8	0
Other Healthcare function	14	48
Office function	14	48
Accommodation function	13	24
Educational function	14	48
Sport function	14	48
Shopping function	13	24
Residential function	10	0

Table 14 The correlation factors of the collector connected to the storage vessel (source: NTA 8800, 2020)

Factor	Correlation factor
A	1.029
B	-0.065
C	-0.0245
D	0.0018
E	0. 0215
F	0

Table 15 Maximum heat loss by storage tanks based on the energy label (source: NTA 8800, 2020)

Energy label	Heat Loss of the Storage Tank (W)
A+	5.5 + 3.16 * V
A	7.0 + 3.7 * V
B	10.25 + 5.09 * V
C	14.33 + 7.13 * V
D	18.83 + 9.33 * V
E	23.5 + 11.99 * V
F	28.5 + 15.16 * V
G	31 + 16.66 * V

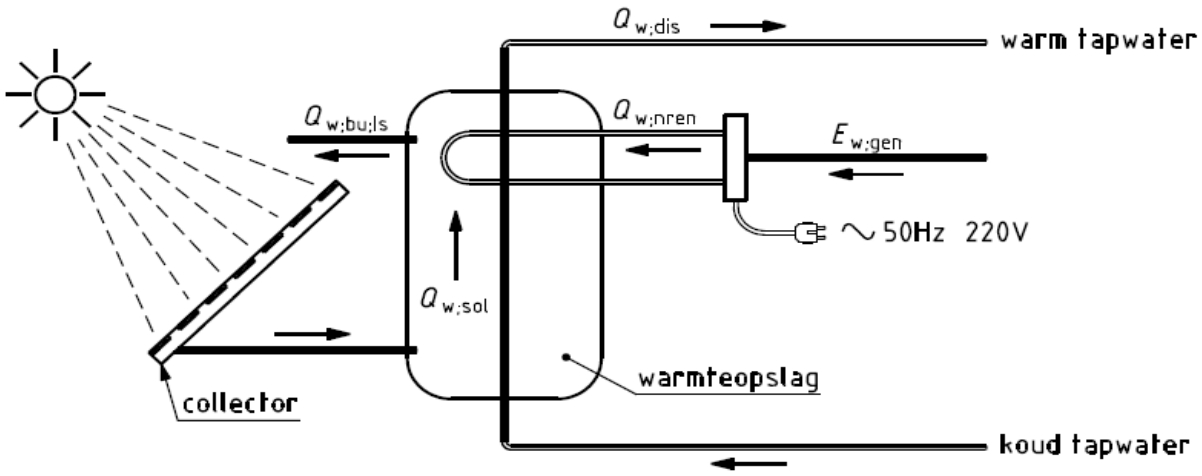

Figure 27 Water heating system with integrated electrical re-heating (source: NTA 8800, 2020)

Table 16 For calculating the monthly contribution of CHP for each system si for hot tap water, in kWh: (source : NTA 8800, 2020)

Building year	Up to and start from 2006		After 2006	
Electrical power of cogeneration installation	ε chp; th	ε chp; el	ε chp; th	ε chp; el
P <= 2kw	N/A	N/A	0.86	0.86
2kw < P <= 20 kw	0.57	0.26	0.57	0.55
20 kw < p <= 200 kw	0.54	0.27	0.51	0.49
200 kw < p <= 500 kw	0.50	0.32	0.52	0.50
500 kw < p <= 1000 kw	0.44	0.35	0.46	0.44
1000 kw < p <= 25 MW	0.4	0.36	0.41	0.39

Appendix III: Process Map

Overview

Within this IDM process map template, the process of the interaction of the developed tool in the early design phase of the project is shown. It will discuss the exchange of information between the architect and the developed tool. In addition, it will also show the exchange of requirements which is needed to create a model exchange.

IDM Process Map

All the processes will be stated below. Tasks, events, gateways, and reiterations are discussed. There are 3 Lanes introduced in the BPMN process map, which are: Architect, ER, and the developed tool.

Name	The functionality of the developed tool in the early design phase

Identifier	Tool_development

Change Log		
2020-09-09	Created	y.m.g.radwan@student.tue.nl
2020-09-09	Adding task description	y.m.g.radwan@student.tue.nl
2020-09-10	PM and text description	y.m.g.radwan@student.tue.nl
2020-09-10	Modified and finished	y.m.g.radwan@student.tue.nl

Exchange Requirements	IFC Model
	Energy Analysis

4.2.3 Tasks

Lane: Architect
Design 3D Model

Type	Task
Name	Design 3D Model
Documentation	In this task, the Architect is required to design a detailed model of the building.

Lane: Architect
Check the design errors

Type	Task
Name	Check the design errors
Documentation	In this task, the Architect is required to check the designed model and make it clear from the design errors to make it ready for the energy assessments.

Lane: Architect
Set IFC export options

Type	Task
Name	Set IFC export options
Documentation	This task can be implemented whether before or after the design process, as in this task the Architect is required to set the IFC export options so the building components can be all exported.

Lane: Architect
Export the IFC model

Type	Task
Name	Export the IFC model
Documentation	In this task, the Architect is required to export the designed model as IFC to be checked and analyzed regarding energy consumption.

Lane: Architect
Import the IFC model in the developed tool

Type	Task
Name	Import the IFC model in the developed tool
Documentation	In this task the IFC model should be imported into the developed tool to allow the tool to perform the energy assessments for the imported model.

Lane: Architect
Receive Feedback

Type	Task
Name	Receive Feedback
Documentation	In this task, the Architect receives the energy assessments associated with feedback based on the energy analysis performed by the developed tool.

Lane: Architect
Evaluate Feedback

Type	Task
Name	Evaluate Feedback
Documentation	In this task, the Architect evaluates the energy assessments and based on the performed analysis he/she decides whether to keep the designed model or modify it.

Lane: Architect
Modify 3D Model

Type	Task
Name	Modify 3D Model
Documentation	In this task, the Architect is required to modify the 3D detailed model if the received feedback from the developed tool states that the IFC designed model doesn't fit the required energy consumption.

Lane: Architect
Export IFC

Type	Task
Name	Export IFC
Documentation	In this task, the Architect is required to export the modified model to be checked and analyzed regarding the energy requirements.

Lane: The developed tool
Create building characteristics

Type	Task
Name	Create building characteristics
Documentation	After the 3D model being imported in the developed tool, the tool collects the needed information and generates information that cannot be contained in the IFC format.

Lane: The developed tool
Make the energy analysis

Type	Task
Name	Make the energy analysis
Documentation	In this task, the developed tool is required to prepare the energy analysis based on the detailed 3D model prepared by the architect.

Lane: The developed tool
Analyze the results

Type	Task
Name	Analyze the results
Documentation	In this task, the developed tool is required to analyze the results of the energy consumption of the designed model, to check whether the model fits the energy calculations required by the Energy Dutch Norm (NTA 8800).

Lane: The developed tool
Send feedback

Type	Task
Name	Send feedback
Documentation	In this task, the developed tool is required to send feedbacks to the Architect with the energy analysis results.

4.2.3 Data objects

Lane: ER
IFC Model.IFC

Type	Data object
Name	IFC Model.IFC
Documentation	The designed 3D model which has been prepared by the Architect is saved as IFC with the needed data and building characteristics.

Lane: ER
Energy analysis

Type	Data object
Name	Energy analysis
Documentation	This document includes all the energy analysis prepared by the developed tool.

4.2.4 Reiteration

Lane: Architect
The 3D Model Modification

Type	Reiteration
Name	The 3D Model Modification
Documentation	The Architect is responsible for the modification of the 3D model after receiving feedback from the developed tool with the energy analysis, the architect is required to check whether the 3D model contributes to the energy requirement, if yes then the architect is not required to modify the design model, if no, hence the architect is required to modify the model and export it as IFC and import it back in the developed tool.

Appendix IV: Lighting exchange requirements

Exchange Requirement				
Element	Property concept	Property name	Definition	Example and explanation test using a rule in pseudo-code
Lighting fixtures				
General Lighting requirements				
Identification	A unique ID for the identification of each wall		ID number of a total of 22 numbers and/or symbols and/or letters	
Specific lighting requirements				
Type	The type of light source The type of light source The type of light source The type of light source The type of light source		Standard Arm extension Exterior Ceiling light Troffer light	
Power density	Lighting power density Lighting power density Lighting power density Lighting power density Lighting power density		60 W 100 W 300 W 277 W 120 W	
Utilization	Coefficient of the utilization of the lighting source Coefficient of the utilization of the lighting source Coefficient of the utilization of the lighting source		0.811 0.712 0.997	
Area	The area of the light source The area of the light source		300*1200 mm² 600*1200 mm²	
Placement	The Location of the lighting sources		spaces	

1 Either one of M - Mandatory / O – Optional / R – Recommended / N - Shall not be used
2. Either one of Boolean, Integer, Real, String, Relation, Object type, etc.
3. Where applicable: Number of items per property such as <=2 (less or equal to two items per property)
4. Where applicable: (e.g. m, m_2, m_3, etc.)

Appendix V: Code Script

```
#IMPORTS
import sys
sys.path
import os,time
import win32file
import win32con

import ifcopenshell
path = r'E:/courses/graduation/Automation of building energy efficiency assessment using BIM/Revit models/Ifc_Model.ifc'
file = ifcopenshell.open(path)
from ifcopenshell import geom
settings = ifcopenshell.geom.settings()
settings.set(settings.USE_PYTHON_OPENCASCADE, True)
#import datetime
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtGui import *
from PyQt5.QtGui import QFont, QPixmap
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QWidget, QLabel, QHBoxLayout, QComboBox, QMainWindow, QApplication, QPushButton, QFileDialog, QTextBrowser, QTableWidget, QVBoxLayout, QSplitter
from OCC.Display.backend import load_backend
load_backend("qt-pyqt5")
import OCC.Display.qtDisplay
from OCC.Display.qtDisplay import qtViewer3d
from OCC.Core.gp import *
import OCC.Core.Bnd, OCC.Core.BRepBndLib

from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.backends.backend_qt5agg import NavigationToolbar2QT as NavigationToolbar
from matplotlib.figure import Figure
import matplotlib.pyplot as plt; plt.rcdefaults()
import numpy as np
#import pandas as pd

from PIL import Image, ImageDraw, ImageFont
from PIL.ImageQt import ImageQt

#INITIATE SELECTION
guid_selection = None
class ProductViewer(qtViewer3d):
    def __init__(self, *args):
        qtViewer3d.__init__(self)
        self.objects = {}

        @staticmethod
        def Hash(shape):
            return shape.HashCode(1 << 30)

        displayed_shapes = {}
        def Show(self, key, shape, color=None):
            self.objects[ProductViewer.Hash(shape)] = key
            qclr = OCC.Quantity.Quantity_Color(.35, .25, .1, OCC.Quantity.Quantity_TOC_RGB)
            ais = self._display.DisplayColoredShape(shape, qclr)
```

```
53.             self.displayed_shapes[key] = ais
54.             self._display.FitAll()
55.
56.         def Color(self, key):
57.             ais = self.displayed_shapes[key]
58.             qclr = OCC.Quantity.Quantity_Color(1, 0, 0, OCC.Quantity.Quantity_TOC_RGB)

59.             ais.GetObject().SetColor(qclr)
60.
61.         def ColorBack(self, key):
62.             ais = self.displayed_shapes[key]
63.             qclr = OCC.Quantity.Quantity_Color(.35, .25, .1, OCC.Quantity.Quantity_TOC_
    RGB)
64.             ais.GetObject().SetColor(qclr)
65.
66.         def ColorWhenElementAcceptable(self, key):
67.             try:
68.                 ais = self.displayed_shapes[key]
69.                 qclr = OCC.Quantity.Quantity_Color(0, 0.7, 0, OCC.Quantity.Quantity_TOC
    _RGB)
70.                 ais.GetObject().SetColor(qclr)
71.             except KeyError as e:
72.                 print( e )
73.
74.         def ColorWhenElementNotAcceptable(self, key):
75.             ais = self.displayed_shapes[key]
76.             qclr = OCC.Quantity.Quantity_Color(1, 0, 0, OCC.Quantity.Quantity_TOC_RGB)

77.             ais.GetObject().SetColor(qclr)
78.
79.         def mouseReleaseEvent(self, *args):
80.             qtViewer3d.mouseReleaseEvent(self, *args)
81.             if self._display.selected_shape:
82.                 global guid_selection
83.                 global selected_shape
84.                 selected_shape = self._display.selected_shape
85.                 guid_selection = (self.objects[ProductViewer.Hash(self._display.selecte
    d_shape)])
86.
87. #MAIN CLASS OF THE APPLICATION (GKZATIOS, 2017)
88. class initUI(object):
89.     def __init__(self, *args):
90.         #(App developing)
91.         app = QtWidgets.QApplication(sys.argv)
92.         # Viewer initialization
93.         self.main = Main(self)
94.         self.main.show()
95.         self.main.canvas.InitDriver()
96.         self.main.statusBar()
97.         self.display = self.main.canvas._display
98.         # Raise a system exit
99.         sys.exit(app.exec_())
100.
101.        #MAIN CLASS OF THE GRAPHICAL USER INTERFACE
102.        class ResultsDialog(QWidget):
103.            def __init__(self, parent=None):
104.                  QtWidgets.QWidget.__init__(self, None)
105.                  self.setGeometry(100, 100, 800, 500)
106.        class PopUpDialog(QWidget):
107.            def __init__(self, parent=None):
108.                QtWidgets.QWidget.__init__(self, None, QtCore.Qt.WindowStaysOnTopHint)
```

```
109.                self.setGeometry(1250, 50, 300, 800)
110.                self.setWindowTitle("Energy Calculations - Feedback")
111.
112.
113.        class Main(QMainWindow):
114.            def __init__(self, parent=None):
115.                self.parent = parent
116.                QtWidgets.QMainWindow.__init__(self)
117.
118.        # Instantiating the tabs
119.                global filename
120.                self.filename = None
121.                self.PopUpDialog = PopUpDialog(self)
122.                self.ResultsDialog = ResultsDialog(self)
123.                self.tabs = QtWidgets.QTabWidget()
124.                self.setCentralWidget(self.tabs)
125.                #VIEWER TAB
126.                self.viewerTab = QtWidgets.QWidget()
127.                self.tabs.addTab(self.viewerTab, "3D IFC-EC integrated Viewer")
128.                # Implementing the OCC viewer
129.                self.canvas = ProductViewer(self)
130.                self.setGeometry(100, 100, 400, 400)
131.                self.setWindowTitle("Energy Performance - IFC Viewer")
132.                # Calling the tab
133.                self.tab3dView()
134.            @QtCore.pyqtSlot(str)
135.            def directory_changed(self, path):
136.                print('Directory Changed')
137.                self.refreshIfcFile()
138.            @QtCore.pyqtSlot(str)
139.            def file_changed(self, path):
140.                print('File Modified')
141.                self.refreshIfcFile()
142.            def tab3dView(self):
143.                # Initializing a split-view layout
144.                font = QtGui.QFont("Calibri", 9, QtGui.QFont.Bold, True)
145.                sizePolicy = QtWidgets.QSizePolicy(QtGui.QSizePolicy.Fixed, QtGui.QSizeP
    olicy.MinimumExpanding)
146.                self.visualizationBox = QtWidgets.QTextBrowser()
147.                self.visualizationBox.setFont(font)
148.                self.visualizationBox.setSizePolicy(sizePolicy)
149.
150.                self.whatIfBox = QtWidgets.QTextBrowser()
151.                self.whatIfBox.setFont(font)
152.                self.visualizationBox.setSizePolicy(sizePolicy)
153.                self.whatIfBox.setFixedHeight(100)
154.
155.                self.feedbackBox = QtWidgets.QTextBrowser()
156.                self.feedbackBox.setFont(font)
157.                self.feedbackBox.setSizePolicy(sizePolicy)
158.
159.                #for the diagram
160.                self.figure = Figure()
161.                self.figureCanvas = FigureCanvas(self.figure)
162.                self.figureToolbar = NavigationToolbar(self.figureCanvas, self)
163.                self.figureCanvas.setFont(font)
164.                self.figureCanvas.setSizePolicy(sizePolicy)
165.
166.                #for the image
167.                self.label = QtWidgets.QLabel()
168.                self.pixmap = QtGui.QPixmap()
```

```
169.
170.                 #for the table
171.                 self.table = QtWidgets.QTableWidget()
172.
173.                 # Define a widget for the 3D viewer
174.                 center1 = QtWidgets.QWidget()
175.                 center2 = QtWidgets.QWidget()
176.                 center3 = QtWidgets.QWidget()
177.                 # Define and set layout
178.                 mainLayout = QtWidgets.QHBoxLayout(center1)
179.                 viewerFeedbackResults = QtWidgets.QVBoxLayout()
180.                 viewerEnergyResults = QtWidgets.QVBoxLayout()
181.
182.                 #BUTTONS
183.                 self.openIfcButton = QtWidgets.QPushButton("Load IFC", self)
184.                 self.openIfcButton.clicked.connect(self.openIfcFile)
185.                 self.openIfcButton.setFixedHeight(20)
186.
187.
188.                 # self.refreshIfcButton = QtWidgets.QPushButton("Refresh IFC", self)
189.                 # self.refreshIfcButton.clicked.connect(self.refreshIfcFile)
190.                 # self.refreshIfcButton.setFixedHeight(20)
191.
192.                 self.calculateEnergyPerformanceButton = QtWidgets.QPushButton("Calculate
     Energy Performance", self)
193.                 self.calculateEnergyPerformanceButton.clicked.connect(self.calculateEner
    gyPerfomance)
194.                 self.calculateEnergyPerformanceButton.setFixedHeight(20)
195.
196.                 self.closeIfcButton = QtWidgets.QPushButton("Close IFC", self)
197.                 self.closeIfcButton.clicked.connect(self.closeIfcFile)
198.                 self.closeIfcButton.setFixedHeight(20)
199.
200.                 self.viewEnergyResultsButton = QtWidgets.QPushButton("Energy Results", s
    elf)
201.                 self.viewEnergyResultsButton.clicked.connect(self.viewEnergyResults)
202.                 self.viewEnergyResultsButton.setFixedHeight(20)
203.
204.                 self.buildingFunctionalityButton = QtWidgets.QComboBox ()
205.                 self.buildingFunctionalityButton.setEditable(True)
206.                 self.buildingFunctionalityButton.lineEdit().setAlignment(QtCore.Qt.Align
    Center)
207.                 self.buildingFunctionalityButton.setFixedHeight(20)
208.                 self.buildingFunctionalityButton.setStyleSheet("background-
    color: #f0f0f0")
209.
210.                 self.buildingFunctionalityButton.addItem("Building Functionality")
211.                 self.buildingFunctionalityButton.addItem("Residence")
212.                 self.buildingFunctionalityButton.currentIndexChanged.connect(self.getBui
    ldingFuctionality)
213.
214.                 self.whatIfButton = QtWidgets.QComboBox()
215.                 self.whatIfButton.setEditable(True)
216.                 self.whatIfButton.lineEdit().setAlignment(QtCore.Qt.AlignCenter)
217.                 self.whatIfButton.setFixedHeight(20)
218.                 self.whatIfButton.setStyleSheet("background-color: #e3e3e3")
219.
220.                 self.whatIfButton.addItem("What if...")
221.                 self.whatIfButton.addItem("Change wall area")
222.                 self.whatIfButton.addItem("Change opening area")
223.                 self.whatIfButton.addItem("Change orientation")
```

```
224.                self.whatIfButton.currentIndexChanged.connect(self.whatIf)
225.
226.                splitterVer1 = QtWidgets.QSplitter(QtCore.Qt.Vertical)
227.                splitterVer2 = QtWidgets.QSplitter(QtCore.Qt.Vertical)
228.                splitterVer3 = QtWidgets.QSplitter(QtCore.Qt.Vertical)
229.
230.                splitterVer1.addWidget(self.table)
231.                splitterVer1.addWidget(self.figureCanvas)
232.                splitterVer1.addWidget(self.figureToolbar)
233.
234.                splitterVer2.addWidget(self.canvas)
235.                splitterVer2.addWidget(self.openIfcButton)
236.                splitterVer2.addWidget(self.buildingFunctionalityButton)
237.                splitterVer2.addWidget(self.calculateEnergyPerformanceButton)
238.                splitterVer2.addWidget(self.closeIfcButton)
239.
240.                #splitterVer3.addWidget(self.refreshIfcButton)
241.                splitterVer3.addWidget(self.viewEnergyResultsButton)
242.                splitterVer3.addWidget(self.feedbackBox)
243.                splitterVer3.addWidget(self.label)
244.                splitterVer3.addWidget(self.whatIfButton)
245.                splitterVer3.addWidget(self.whatIfBox)
246.
247.                mainLayout.addWidget(splitterVer2)
248.                secondLayout = QtWidgets.QVBoxLayout(center2)
249.                secondLayout.addLayout(viewerFeedbackResults)
250.                viewerFeedbackResults.addWidget(splitterVer3)
251.                thirdLayout = QtWidgets.QVBoxLayout(center3)
252.                thirdLayout.addLayout (viewerEnergyResults)
253.                viewerEnergyResults.addWidget(splitterVer1)
254.
255.                self.viewerTab.setLayout(mainLayout)
256.                self.PopUpDialog.setLayout(secondLayout)
257.                self.ResultsDialog.setLayout(thirdLayout)
258.
259.                #To make buttons unavailable until IFC file is loaded
260.                if not self.filename:
261.                    self.calculateEnergyConsumptionButton.setEnabled(False)
262.                    self.closeIfcButton.setEnabled(False)
263.                    self.viewEnergyResultsButton.setEnabled(False)
264.                    self.buildingFunctionalityButton.setEnabled(False)
265.                    self.buildingFunctionalityButton.setStyleSheet("background-
    color: #f0f0f0")
266.                    self.whatIfButton.setEnabled(False)
267.                    self.whatIfButton.setStyleSheet("background-color: #f0f0f0")
268.                self.count = 0
269.                self.rdf_graph = None
270.
271.                #Load IFC file
272.                def openIfcFile(self):
273.                    self.counter = 1
274.                    self.filename = QtWidgets.QFileDialog.getOpenFileName(self, 'Open fi
    le', ".", "Industry Foundation Classes(*.ifc)")
275.                    self.fs_watcher = QtCore.QFileSystemWatcher([self.filename])
276.                    self.fs_watcher.connect(self.fs_watcher, QtCore.SIGNAL('fileChanged(
    const QString &)'), self.file_changed)
277.
278.                    if self.filename:
279.                        self.parent.display.EraseAll()
280.                        self.visualizationBox.clear()
281.                        self.whatIfBox.clear()
```

```
282.                         self.feedbackBox.clear()
283.                     self.buildingFunctionalityButton.setEnabled(True)
284.                     self.buildingFunctionalityButton.setStyleSheet("background-
    color: #e3e3e3")
285.                     self.closeIfcButton.setEnabled(True)
286.
287.                     #creating energy results and feedback lists
288.                     self.energyResults = []
289.                     self.FeedbackResults = []
290.                     self.parse_ifc(self.filename)
291.
292.                     #WHAT IF Scenarios
293.                     def whatIf(self):
294.                         self.whatIfBox.clear()
295.                         if self.whatIfButton.currentText()== "What if..." :
296.                             self.whatIfBox.append(r'Check "what if conditions" for optimi
    zing the results')
297.                             self.whatIfBox.append("without modifying the 3D deisgn")
298.                         if self.whatIfButton.currentText() == "Change wall area":
299.                             self.getWhatIfChangeWallArea()
300.                         if self.whatIfButton.currentText() == "Change opening area":
301.                             self.getWhatIfChangeOpeningArea()
302.                         if self.whatIfButton.currentText() == "Change orientation":
303.                             self.getWhatIfChangeOrientation()
304.                     def getWhatIfChangeWallArea(self):
305.                         self.whatIfBox.append("Changes in wall area")
306.                         for wall in self.wallList:
307.                             wall.area = wall.area * 1.049
308.                         for roof in self.roofList:
309.                             roof.area = roof.area * 1.1
310.                         self.totalExternalArea = self.totalExternalArea * 1.1
311.                         self.getAllEnergyResults()
312.                         self.tipResultsSaved()
313.
314.                         oldResults = self.tipResults[-2]
315.                         newResults = self.tipResults[-1]
316.
317.                         differences = tuple(x -
     y for x, y in zip(newResults, oldResults))
318.                         if differences[3] >= 0:
319.                             self.whatIfBox.append((" -
    " + "External area increase by 10%:" + " " + "+" + str(differences[3] + "MJ")))
320.                         else:
321.                             self.whatIfBox.append((" -
    " + "External area increase by 10%:" + " " + str(differences[3] + "MJ")))
322.                             for wall in self.wallList:
323.                                 wall.area = wall.area / 0.9486
324.                             for roof in self.roofList:
325.                                 roof.area = roof.area / 0.9
326.                             self.totalExternalArea = self.totalExternalArea / 0.9
327.
328.                             del self.tipResults[1:]
329.
330.                             def getWhatIfChangeOpeningArea(self):
331.                                 self.whatIfBox.append("Changes in opening area")
332.                                 for opening in self.openingList:
333.                                     opening.area = opening.area * 0.9
334.                                     unique = True
335.                                     for wall in self.wallList:
336.                                         if unique:
```

```
337.                                             if wall.orientation == opening.orientation:
338.                                                 wall.area = wall.area + ((opening.area / 0.9) - opening.area)
339.                                                 unique = False
340.                             self.getAllEnergyResults()
341.                             self.tipResultsSaved()
342.                             oldResults = self.tipResults[-2]
343.                             newResults = self.tipResults[-1]
344.
345.                             differences = tuple(x - y for x, y in zip(newResults, oldResults))
346.                             if differences[3] >= 0:
347.                                 self.whatIfBox.append((" - " + "External area of openings decrease by 10%:" + " " + " +" + str(differences[3])))
348.                             else:
349.                                 self.whatIfBox.append( (" - " + "External area of openings decrease by 10%:" + " " + str(differences[3])))
350.                             for opening in self.openingList:
351.                                 temporaryValue = opening.area
352.                                 opening.area = opening.area/0.9
353.                                 unique = True
354.                             for wall in self.wallList:
355.                                 if unique:
356.                                     if wall.orientation == opening.orientation:
357.                                         wall.area = wall.area -(opening.area - temporaryValue)
358.                                         unique = False
359.                             del self.tipResults[1:]
360.                             def getWhatIfChangeOrientation(self):
361.                                 self.whatIfBox.append ("Changes in orientation")
362.                                 for index in range(1, 4):
363.                                     for wall in self.wallList:
364.                                         if wall.orientation == "North":
365.                                             wall.orientation = "East"
366.                                         elif wall.orientation == "East":
367.                                             wall.orientation = "South"
368.                                         elif wall.orientation == "South":
369.                                             wall.orientation = "West"
370.                                         elif wall.orientation == "West":
371.                                             wall.orientation = "North"
372.                                     for opening in self.openingList:
373.                                         if opening.orientation == "North":
374.                                             opening.orientation = "East"
375.                                         elif opening.orientation == "East":
376.                                              opening.orientation = "South"
377.                                         elif opening.orientation == "South":
378.                                             opening.orientation = "West"
379.                                         elif opening.orientation == "West":
380.                                             opening.orientation = "North"
381.                                     self.getAllEnergyResults()
382.                                     self.tipResultsSaved()
383.
384.                                     oldResults = self.tipResults[(-1 - index)]
385.                                     newResults = self.tipResults[-1]
386.
387.                                     if index == 1:
388.                                         rotation = "90 degrees: "
389.                                     elif index == 2:
```

```
                                        rotation = "180 degrees: "
                                    elif index == 3:
                                        rotation = "270 degrees: "

                                    differences = tuple(x -
     y for x, y in zip(newResults, oldResults))
                                    if (differences[3] + differences[4]) <= 0:
                                        self.whatIfBox.append( (" -
    " + "Rotate building clockwise by" + " " + rotation + "+" + str(differences[3] + differ
    ences[4])))
                                    else:
                                        self.whatIfBox.append( (" -
    " + "Rotate building clockwise by" + " " + rotation + "-
    " + str(differences[3] + differences[4])))

                                    for wall in self.wallList:
                                        if wall.orientation == "North":
                                            wall.orientation = "East"
                                        elif wall.orientation == "East":
                                            wall.orientation = "South"
                                        elif wall.orientation == "South":
                                            wall.orientation = "West"
                                        elif wall.orientation == "West":
                                            wall.orientation = "North"
                                        for opening in self.openingList:
                                            if opening.orientation == "North":
                                                opening.orientation = "East"
                                            elif opening.orientation == "East":
                                                opening.orientation = "South"
                                            elif opening.orientation == "South":
                                                opening.orientation = "West"
                                            elif opening.orientation == "West":
                                                opening.orientation = "North"
                                        del self.tipResults[1:]

                                         #BUILDING FUNCTIONALITY
                                        def getBuildingFuctionality(self):
                                            # set building functionality
                                            self.buildingFunctionality = ""
                                            if self.buildingFunctionalityButton.currentT
    ext() != "Building Functionality":
                                                self.buildingFunctionality = str(self.bu
    ildingFunctionalityButton.currentText())
                                            else:
                                                self.buildingFunctionality == ""
                                                #enable other buttons
                                            if self.buildingFunctionality != "":
                                                self.calculateEnergyConsumptionButton.se
    tEnabled(True)
                                                self.viewEnergyResultsButton.setEnabled(
    True)
                                                self.whatIfButton.setEnabled(True)
                                                self.whatIfButton.setStyleSheet("backgro
    und-color: #e3e3e3")
                                            elif self.buildingFunctionality == "":
                                                self.calculateEnergyConsumptionButton.se
    tEnabled(False)
                                                self.viewEnergyResultsButton.setEnabled(
    False)
                                                self.whatIfButton.setEnabled(False)
```

```
                                                self.whatIfButton.setStyleSheet("backgro
    und-color: #f0f0f0")

        #Setting values according to the building's functionality based on NTA8800
                                                #water heating demand


        #Close IFC
         def closeIfcFile(self):
             self.filename = None
             self.parent.display.EraseAll()
             self.visualizationBox.clear()
             self.whatIfBox.clear()
             self.feedbackBox.clear()
             self.figure.clf()
             self.figureCanvas.draw()
             self.calculateEnergyConsumptionButton.setEnabled(False)
             self.closeIfcButton.setEnabled(False)
             self.viewEnergyResultsButton.setEnabled(False)
             self.buildingFunctionalityButton.setCurrentIndex(0)
             self.buildingFunctionalityButton.setEnabled(False)
             self.buildingFunctionalityButton.setStyleSheet("background-
    color: #f0f0f0")
             self.whatIfButton.setCurrentIndex(0)
             self.whatIfButton.setEnabled(False)
             self.whatIfButton.setStyleSheet("background-color: #f0f0f0")

                #REFRESH IFC
         @QtCore.pyqtSlot(str)
         def refreshIfcFile(self):
         time.sleep(1)
         if self.filename:
            self.counter = self.counter + 1
            print (self.counter)
            self.parent.display.EraseAll()
            self.visualizationBox.clear()
            self.feedbackBox.clear()
            self.whatIfBox.clear()
            self.figure.clf()
            self.tipResults = []
            self.parse_ifc(self.filename)
            if self.calculateEnergyConsumptionButton.isEnabled():
               self.calculateEnergyConsumption()

               #PARSE IFC
            def parse_ifc(self, filename):
                self.created_shapes = {}
                self.ifc_file = ifcopenshell.open(filename)
                rooms = self.ifc_file.by_type("IfcBuildingElement")
                for room in rooms:
                    if room.Representation:
                        ifcgeom = ifcopenshell.geom.create_shape(settings, room).geometry

                        shp = self.canvas.Show(room.GlobalId, ifcgeom, None)
                        print ("IFC file successfully loaded")


                      #calculating energy performance
             def calculateEnergyConsumption(self):
                 self.PopUpDialog.show()
                 self.whatIfButton.setCurrentIndex(0)
```

```
                self.whatIfBox.clear()
                self.whatIfBox.append(r'Check "what if conditions" for getting better r
    esults')
                self.whatIfBox.append("without modifying the 3D design")

                        #Calculations
            def getWallForTotal(wall, areaOrVolume):
                forTotalVolume = 0
                forTotalArea = ((wall.startPoint[0] * wall.endPoint[1]) -
     (wall.startPoint[1] * wall.endPoint[0]))
                  if areaOrVolume == "area":
                      return (forTotalArea / 2)
                  elif areaOrVolume == "volume":
                      return (forTotalVolume / 2)
              def getWallForPart(wall):
                  for connect in wall.connectedTo:
                     if wall.connectedTo != []:
                         wall.forPart.append((((connect[1] * wall.startPoint[0]) -
     (connect[0] * wall.startPoint[1])) / 2))
                         counter = 0
                     if len(wall.forPart) > 1:
                        for index in range(1, len(wall.forPart)):
                            wall.forPart.append((wall.forPart[index] -
     wall.forPart[(index - 1)]))
                            counter = counter + 1
                         for index in range(0, counter):
                             wall.forPart.remove(wall.forPart[1])
                             wall.forPart.append(round(((wall.forTotal) -
     sum(wall.forPart)), 3))

                    class myWall(object):
                          def __init__(self, globalId, internalOrExternal, startPoint,
    endPoint, possibleStartPoint, possibleEndPoint, axisDirection, length, width, height, a
    xisOrientation, orientation, area, connectedTo, forTotal, forPart, thermalMass, thermal
    Resistance, heatTransferCoefficient, absorptance, ThermalTransmittance,solarHeatGainCoe
    fficient, Roughness, HeatLossThroughTransmission, IfcGUID, thermalTransmittance):
                                self.IfcGUID = IfcGUID
                                self.globalId = globalId
                                self.internalOrExternal = internalOrExternal
                                self.startPoint = startPoint
                                self.possibleStartPoint = possibleStartPoint
                                self.possibleEndPoint = possibleEndPoint
                                self.endPoint = endPoint
                                self.axisDirection = axisDirection
                                self.length = length
                                self.width = width
                                self.height = height
                                self.axisOrientation = axisOrientation
                                self.orientation = orientation
                                self.area = area
                                self.connectedTo = connectedTo
                                self.thermalMass = thermalMass
                                self.thermalResistance = thermalResistance
                                self.heatTransferCoefficient = heatTransferCoefficient
                                self.absorptance = absorptance
                                self.thermalTransmittance = thermalTransmittance
                                self.solarHeatGainCoefficient = solarHeatGainCoefficient

                                self.roughness = Roughness
                                self.HeatLossThroughTransmission = HeatLossThroughTransm
    ission
```

```
546.
547.                     class myOpening(object):
548.                             def __init__(self, globalId, doorOrWindow, internalOrExtern
    al,length, height, area, orientation, thermalMass, thermalResistance, heatTransferCoeff
    icient, absorptance, thermalTransmittance, solarHeatGainCoefficient, Rvalue, HeatLossTh
    roughVentilation, IfcGUID, AverageEffectiveTotalSolarAccessionFactor, CosineFraction, C
    orrectionFactorForNonScatteringGlazing, SolarAccessionFactor, CollectorArea, HeatGainBy
    SolarRadiation, IncidentSolarRadiationByOrientation, ShadingReductionFactor):
549.                                 self.globalId = globalId
550.                                 self.doorOrWindow = doorOrWindow
551.                                 self.internalOrExternal = internalOrExternal
552.                                 self.length = length
553.                                 self.height = height
554.                                 self.area = area
555.                                 self.orientation = orientation
556.                                 self.thermalMass = thermalMass
557.                                 self.thermalResistance = thermalResistance
558.                                 self.heatTransferCoefficient = heatTransferCoefficient

559.                                 self.absorptance = absorptance
560.                                 self.thermalTransmittance = thermalTransmittance
561.                                 self.solarHeatGainCoefficient = solarHeatGainCoefficien
    t
562.                                 self.Rvalue = Rvalue
563.                                 self.IfcGUID = IfcGUID
564.                                 self.HeatLossThroughVentilation = HeatLossThroughVentil
    ation
565.                                 self.AverageEffectiveTotalSolarAccessionFactor = Averag
    eEffectiveTotalSolarAccessionFactor
566.                                 self.CosineFraction = CosineFraction
567.                                 self.ShadingReductionFactor = ShadingReductionFactor
568.                                 self.IncidentSolarRadiationByOrientation = IncidentSola
    rRadiationByOrientation
569.                                 self.CorrectionFactorForNonScatteringGlazing = Correcti
    onFactorForNonScatteringGlazing
570.                                 self.SolarAccessionFactor = SolarAccessionFactor
571.                                 self.CollectorArea = CollectorArea
572.                                 self.HeatGainBySolarRadiation = HeatGainBySolarRadiatio
    n
573.
574.                         class mySlab (object):
575.                                 def __init__(self, globalId, area, thermalMass,thermalResi
    stance, heatTransferCoefficient, absorptance,thermalTransmittance,solarHeatGainCoeffici
    ent, Roughness, IfcGUID, Cbasins, Cbath, Ckitchen, CPerPerson, Cshower, QstandCirculati
    onLoss, QsatndingStill, Tap, BoilerEfficiency, CorrectionFactor, NumberOfBathsPerPerson
    PerDay, SavingShowerHead, NumberOfShowersPerPersonPerDay, ηuseful, HeatGainByHotWater,
    AuxilaryHotWater, QstandingStill):
576.                                 self.globalId = globalId
577.                                 self.area = area
578.                                 self.thermalMass = thermalMass
579.                                 self.thermalResistance = thermalResistance
580.                                 self.heatTransferCoefficient = heatTransferCoefficient

581.                                 self.absorptance = absorptance
582.                                 self.thermalTransmittance = thermalTransmittance
583.                                 self.solarHeatGainCoefficient = solarHeatGainCoefficien
    t
584.                                 self.IfcGUID = IfcGUID
585.                                 self.roughness = Roughness
586.                                 self.Cbasins = Cbasins
587.                                 self.Cbath = Cbath
```

```
                                    self.Ckitchen = Ckitchen
                                    self.CPerPerson = CPerPerson
                                    self.Cshower = Cshower
                                    self.QstandCirculationLoss = QstandCirculationLoss
                                    self.QsatndingStill = QstandingStill
                                    self.Tap = Tap
                                    self.BoilerEfficiency = BoilerEfficiency
                                    self.CorrectionFactor = CorrectionFactor
                                    self.NumberOfBathsPerPersonPerDay = NumberOfBathsPerPer
    sonPerDay
                                    self.SavingShowerHead = SavingShowerHead
                                    self.NumberOfShowersPerPersonPerDay = NumberOfShowersPe
    rPersonPerDay
                                    self.ηuseful = ηuseful
                                    self.HeatGainByHotWater = HeatGainByHotWater
                                    self.AuxilaryHotWater = AuxilaryHotWater


                           class mySpace (object):
                                 def __init__(self, globalId, area, thermalMass,thermalResist
    ance, heatTransferCoefficient, absorptance, thermalTransmittance, solarHeatGainCoeffici
    ent, Roughness, IfcGUID, NumberOfPeople, Qlighting, SpecifiedLightingLoad, InternalHeat
    Gain, EnergyRequirementForLighting, EnergyRequirementForParasiticPower, Nliving):
                                    self.globalId = globalId
                                    self.area = area
                                    self.thermalMass = thermalMass
                                    self.thermalResistance = thermalResistance
                                    self.heatTransferCoefficient = heatTransferCoefficient

                                    self.absorptance = absorptance
                                    self.thermalTransmittance = thermalTransmittance
                                    self.solarHeatGainCoefficient = solarHeatGainCoefficient

                                    self.IfcGUID = IfcGUID
                                    self.roughness = Roughness
                                    self.Numberofpeople = NumberOfPeople
                                    self.Qlighting = Qlighting
                                    self.SpecifiedLightingLoad = SpecifiedLightingLoad
                                    self.Nliving = Nliving
                                    self.InternalHeatGain = InternalHeatGain
                                    self.EnergyRequirementForLighting = EnergyRequirementFor
    Lighting
                                    self.EnergyRequirementForParasiticPower = EnergyRequirem
    entForParasiticPower


                        class myLightFixtureType (object):
                                 def __init__(self, IfcGUID, TottalWattage):
                                    self.IfcGUID = IfcGUID
                                    self.TottalWattage = TottalWattage

          def getNorthDirection(mainProject):
              northDirection = [0.0, 1.0]
              counter = 0
              for Project in mainProject:
                  for RepresentationContext in Project.RepresentationContexts:
                      if counter == 0:
                         if RepresentationContext.is_a("IfcGeometricRepresentationContext"
    ):
                            if RepresentationContext.TrueNorth != None:
```

```
                            northDirection[0] = round((RepresentationContext.TrueNorth.DirectionRatios[0]), 1)
                            northDirection[1] = round((RepresentationContext.TrueNorth.DirectionRatios[1]), 1)
                            counter = 1
                            return (northDirection)
          def getAllWidths(wallList):
               widths = []
               for wall in wallList:
                   width = wall.width
                   if width not in widths:
                       widths.append(width)
                       return (widths)



                      # FUNCTIONS FOR myWall CHARACTERISTICS #
          def getWallAxisDirection(wall):
              wallAxisDirection = [1.0, 0.0]
              if wall.ObjectPlacement.is_a("IfcLocalPlacement"):
                 if wall.ObjectPlacement.RelativePlacement.RefDirection != None:
                    wallAxisDirection[0] = wall.ObjectPlacement.RelativePlacement.RefDirection.DirectionRatios[0]
                    wallAxisDirection[1] = wall.ObjectPlacement.RelativePlacement.RefDirection.DirectionRatios[1]
                    return (wallAxisDirection)

          def getWallLengthWidthHeight(wall, lengthOrWidthOrHeightOrAreaOrVolume):
              width = 0
              length = 0
              height = 0
              volume = 0

              for relDefinesByProperties in wall.IsDefinedBy:
                   if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
                       if relDefinesByProperties.RelatingPropertyDefinition.is_a("IfcElementQuantity"):
                           for properties in relDefinesByProperties.RelatingPropertyDefinition.Quantities:
                               if properties.is_a("IfcQuantityLength"):
                                   if properties.Name == "Width":
                                       width = properties.LengthValue
                                       if properties.Name == "Length":
                                           length = properties.LengthValue
                                           if properties.Name == "Height":
                                               height = properties.LengthValue
                                               if properties.Name == "Volume":
                                                   volume = properties.LengthValue
                                               elif relDefinesByProperties.RelatingPropertyDefinition.is_a("IfcPropertySet"):
                                                  for properties in relDefinesByProperties.RelatingPropertyDefinition.HasProperties:
                                                     if properties.Name == "Width":
                                                        width = properties.NominalValue.wrappedValue
                                                        if properties.Name == "Length":
                                                           length = properties.NominalValue.wrappedValue
```

```
                                                                    if properties.Name ==
    "Height":
                                                                        height = properties
    .NominalValue.wrappedValue


                if properties.Name == "Volume":
                    volume = properties.NominalValue.wrappedValue
                    area = length * height
                    if lengthOrWidthOrHeightOrAreaOrVolume == "length":
                        return (length)
                        elif lengthOrWidthOrHeightOrAreaOrVolume == "width":
                             return (width)
                        elif lengthOrWidthOrHeightOrAreaOrVolume == "height":
                              return (height)
                        elif lengthOrWidthOrHeightOrAreaOrVolume == "area":
                              return (area)
                        elif lengthOrWidthOrHeightOrAreaOrVolume == "volume":
                              return (volume)

            def getThermalValues(element):
                 thermalMass = 0
                 thermalResistance = 0
                 heatTransferCoefficient = 0
                 absorptance = 0.8
                 solarHeatGainCoefficient = 0
                 Roughness = 3
                 HeatLossThroughTransmission = 25.3
                 for relDefinesByType in element.IsDefinedBy:
                     if relDefinesByType.is_a("IfcRelDefinesByType"):
                        if relDefinesByType.RelatingType.is_a("IfcTypeProduct"):
                           for PropertySets in relDefinesByType.RelatingType.HasPropertySet
    s:
                               if PropertySets.is_a("IfcPropertySet"):
                                   if PropertySets.Name == "Analytical Properties":

    for Properties in PropertySets.HasProperties:

        if Properties.Name == "Heat Transfer Coefficient (U)":

            heatTransferCoefficient = Properties.NominalValue.wrappedValue

        if Properties.Name == "Thermal mass":

            thermalMass = Properties.NominalValue.wrappedValue

        if Properties.Name == "Thermal Resistance (R)":

            thermalResistance = Properties.NominalValue.wrappedValue

        if Properties.Name == "Absorptance":

            absorptance = Properties.NominalValue.wrappedValue

        if Properties.Name == "Solar Heat Gain Coefficient":

            solarHeatGainCoefficient = Properties.NominalValue.wrappedValue

        if Properties.Name == "Roughness":

            Roughness = Properties.NominalValue.wrappedValue
```

```
733.
        if Properties.Name == "Heat loss through transmission":
734.
            HeatLossThroughTransmission = Properties.NominalValue.wrappedValue
735.
        elif relDefinesByType.is_a("IfcRelDefinesByProperties"):
736.
            if relDefinesByType.RelatingPropertyDefinition.is_a("IfcPropertySet"):
737.
                for Properties in relDefinesByType.RelatingPropertyDefinition.HasProperties
    :
738.
                    if Properties.Name == "Heat Transfer Coefficient (U)":
739.
                        heatTransferCoefficient = Properties.NominalValue.wrappedValue
740.


741.


742.
        if Properties.Name == "Thermal mass":
743.
            thermalMass = Properties.NominalValue.wrappedValue
744.
        if Properties.Name == "Thermal Resistance (R)":
745.
            thermalResistance = Properties.NominalValue.wrappedValue
746.
        if Properties.Name == "Absorptance":
747.
            absorptance = Properties.NominalValue.wrappedValue
748.
        if Properties.Name == "Solar Heat Gain Coefficient":
749.
            solarHeatGainCoefficient = Properties.NominalValue.wrappedValue
750.
        if Properties.Name == "Roughness":
751.
            Roughness = Properties.NominalValue.wrappedValue
752.
        if Properties.Name == "Heat loss through transmission":
753.
            HeatLossThroughTransmission = Properties.NominalValue.wrappedValue
754.
        if heatTransferCoefficient == 0:
755.
            heatTransferCoefficient = 0.533
756.
        if thermalResistance == 0:
757.
            thermalResistance = 1/0.533
758.


759.
        return (thermalMass, thermalResistance, heatTransferCoefficient, absorptance, solar
    HeatGainCoefficient, Roughness, HeatLossThroughTransmission)
760.


761.
        def getThermalTransmittance (opening):
```

```
762.
763.            thermalTransmittance = 1.3
764.            for relDefinesByProperties in opening.IsDefinedBy:
765.                if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
766.                    if relDefinesByProperties.RelatingPropertyDefinition.is_a("IfcPropertyS
    et"):
767.                        for properties in relDefinesByProperties.RelatingPropertyDefinition
    .HasProperties:
768.                            if properties.Name == "R Value":
769.                                thermalTransmittance = properties.NominalValue.wrappedValue

770.                                return (thermalTransmittance)
771.
def checkIfElementIsExternal(wall):

772.    external = False
773.    for relDefinesByProperties in wall.IsDefinedBy:
774.        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
775.            if relDefinesByProperties.RelatingPropertyDefinition.is_a("IfcPropertySet"):
776.                for properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
777.                    if properties.Name == "IsExternal":
778.                        if properties.NominalValue.wrappedValue == True:
779.                            external = True
780.                            return (external)

def getPossibleCoordinates(wall, allWidths):

781.     firstCoordinates = getFirstCoordinates(wall)
782.     isConnectedTo = getHowManyWallsIsConnectedTo(wall)
783.     for possible in allWidths:
784.         #assuming coordinates
785.         Case1x1 = round((firstCoordinates[0] + (wall.axisDirection[0] * possible / 2)), 10
    )
```

```
786.
        Case1y1 = round((firstCoordinates[1] + (wall.axisDirection[1] * possible / 2)), 10
    )
787.
        Case2x1 = round((firstCoordinates[0] -
     (wall.axisDirection[0] * possible / 2)), 10)
788.
        Case2y1 = round((firstCoordinates[1] -
     (wall.axisDirection[1] * possible / 2)), 10)
789.
        Case3x1 = round ((firstCoordinates[0]),10)
790.
        Case3y1 = round((firstCoordinates[1]), 10)
791.
        wall.possibleStartPoint.append([Case1x1, Case1y1, firstCoordinates[2]])
792.
        wall.possibleStartPoint.append([Case2x1, Case2y1, firstCoordinates[2]])
793.
        wall.possibleStartPoint.append([Case3x1, Case3y1, firstCoordinates[2]])
794.

795.
        #to get coordinates
796.
        Case1x2 = round( (firstCoordinates[0] + (wall.axisDirection[0] * (wall.length + (i
    sConnectedTo * possible / 2)))), 10)
797.
        Case1y2 = round( (firstCoordinates[1] + (wall.axisDirection[1] * (wall.length + (i
    sConnectedTo * possible / 2)))), 10)
798.
        Case2x2 = round( (firstCoordinates[0] + (wall.axisDirection[0] * (wall.length -
     (isConnectedTo * possible / 2)))), 10)
799.
        Case2y2 = round( (firstCoordinates[1] + (wall.axisDirection[1] * (wall.length -
     (isConnectedTo * possible / 2)))), 10)
800.
        Case3x2 = round( (firstCoordinates[0] + (wall.axisDirection[0] * wall.length)), 10
    )
801.
        Case3y2 = round( (firstCoordinates[1] + (wall.axisDirection[1] * wall.length)), 10
    )
802.

803.
        wall.possibleEndPoint.append([Case1x2, Case1y2, firstCoordinates[2]])
804.
        wall.possibleEndPoint.append([Case2x2, Case2y2, firstCoordinates[2]])
805.
        wall.possibleEndPoint.append([Case3x2, Case3y2, firstCoordinates[2]])
806.


def getFirstCoordinates(wall):

807.
    unique = True
808.
    for wall in wallStandardCaseList:
809.
        if unique:
```

```
810.
          if wall.globalId == wall.GlobalId:
811.
              x1 = round((wall.ObjectPlacement.RelativePlacement.Location.Coordinates[0])
    , 10)
812.
              y1 = round((wall.ObjectPlacement.RelativePlacement.Location.Coordinates[1])
    , 10)
813.
              z1 = round(((wall.ObjectPlacement.RelativePlacement.Location.Coordinates[ 2
    ]) + getHeightAfterLevel(wall)), 10)
814.
              unique = False
815.
    firstCoordinates = [x1, y1, z1]
816.
    return (firstCoordinates)

 def getHeightAfterLevel(wall):

817.
    level = 1
818.
    for relDefinesByProperties in wall.IsDefinedBy:
819.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
820.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("IfcPropertySet"):
821.
                for properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
822.
                    if properties.Name == "Base Constraint":
823.
                        baseConstraintName = properties.NominalValue.wrappedValue
824.
                        baseConstraintName = baseConstraintName.replace('Level: ', '')
825.
                        level = level + getLevelHeight(baseConstraintName)
826.
                    if properties.Name == "Base Offset":
827.
                        level = level + properties.NominalValue.wrappedValue
828.
    return (level)
829.

def getLevelHeight(baseConstraintName):

830.
     buildingStoreyList = self.ifc_file.by_type("IfcBuildingStorey")
831.
     for buildingStorey in buildingStoreyList:
832.
         if baseConstraintName == buildingStorey.Name:
833.
             baseConstraint = buildingStorey.Elevation
834.
             return (baseConstraint)
```

```
def getHowManyWallsIsConnectedTo(wall):

835.
    unique = True
836.
    for wall in wallStandardCaseList:
837.
        if unique:
838.
            if wall.globalId == wall.GlobalId:
839.
                counter = 0
840.
                for RelConnectsElements in wall2.ConnectedTo:
841.
                    if RelConnectsElements.is_a("IfcRelConnectsPathElements"):
842.
                        if RelConnectsElements.RelatedElement.is_a("IfcBuildingElement"):
843.
                            if RelConnectsElements.RelatedElement.is_a("IfcWallStandardCase
    "):
844.
                                if checkIfElementIsExternal(RelConnectsElements.RelatedElem
    ent):
845.
                                    counter = counter + 1
846.
        if counter == 2:
847.
            isConnectedTo = 1
848.
        else:
849.
            isConnectedTo = -1
850.
        unique = False
851.
        return (isConnectedTo)

def getConnectedToInCorrectOrder(wallList):

852.
    for wall in wallList:
853.
        if len(wall.connectedTo) > 1:
854.
            for index in range(0, (len(wall.connectedTo) - 1)):
855.
                if abs(wall.connectedTo[index][0] - wall.startPoint[0]) >= \
856.
                    abs(wall.connectedTo[index + 1][0] - wall.startPoint[0]) and \
857.
                    abs(wall.connectedTo[index][1] - wall.startPoint[1]) >= \
858.
                    abs(wall.connectedTo[index][1] - wall.startPoint[1]):
859.
                    (wall.connectedTo[index], wall.connectedTo[index + 1]) = ( wall.connecte
    dTo[index + 1], wall.connectedTo[index])
860.
    # FOR EXTERNAL WALLS startPoint, endPoint, orientation, AND totalExternalArea
```

```
 def defineFinalCoordinates_defineAxisOrientation_getTotalExternalArea(checklist, totalExt
ernalArea, totalExternalVolume):
861.
    finalWallList = [checklist[0]]
862.
    getFinalCoordinatesForExternalWall(finalWallList, externalWallList)
863.
    externalArea = 0
864.
    externalVolume = 0
865.
    for wall in finalWallList:
866.
        externalArea = externalArea + getWallForTotal(wall, "area")
867.
        externalVolume = externalVolume + getWallForTotal(wall, "volume")
868.
    orientation = externalArea
869.
    totalExternalArea = totalExternalArea + abs(externalArea)
870.
    totalExternalVolume = totalExternalVolume + abs(externalVolume)
871.

872.
    getOrientation(finalWallList, orientation)
873.
    externalOpeningList = []
874.
    for opening in self.openingList:
875.
        if opening.internalOrExternal == "external":
876.
            externalOpeningList.append(opening)
877.

878.
    def getOpeningOrientation(opening):
879.
        openingOrientation = []
880.
        for relFillsElement in opening.FillsVoids:
881.
            for relVoidsElement in relFillsElement.RelatingOpeningElement.VoidsElements:
882.
                if relVoidsElement.RelatingBuildingElement.is_a("IfcBuildingElement"):
883.
                    if relVoidsElement.RelatingBuildingElement.is_a("IfcWallStandardCase"):

884.
                        wallId = relVoidsElement.RelatingBuildingElement.GlobalId
885.
                        unique = True
886.
                        for wall in externalWallList:
887.
                            if unique:
888.
                                if wallId == wall.globalId:
```

```
                                    openingOrientation = wall.orientation
                                    wall.area = wall.area -
    (opening.OverallWidth *opening.OverallHeight)
                                    unique = False
                                    return (openingOrientation)
    def checkIfOpeningIsExternal(opening):
        openingLocation = "internal"
        for relFillsElement in opening.FillsVoids:
            for relVoidsElement in relFillsElement.RelatingOpeningElement.VoidsElements:
                if relVoidsElement.RelatingBuildingElement.is_a("IfcBuildingElement"):
                    if relVoidsElement.RelatingBuildingElement.is_a("IfcWallStandardCase"):
                        wallId = relVoidsElement.RelatingBuildingElement.GlobalId
                        unique = True
                        for wall in self.wallList:
                            if unique:
                                if wallId == wall.globalId:
                                    openingLocation = wall.internalOrExternal
                                    unique = False
                                    return (openingLocation)


def getFloorArea (slab):

    area = 0

for relDefinesByProperties in slab.IsDefinedBy:

    if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
        if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
            for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProperti
    es:
                if Properties.Name == "Area":
                    area = Properties.NominalValue.wrappedValue
                    return (area)
```

```
def getFloorCbasins (slab):

    Cbasins = 3.97
    for relDefinesByProperties in slab.IsDefinedBy:
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProperties:
                    if Properties.Name == "Cbasins":
                        Cbasins = Properties.NominalValue.wrappedValue
                        return (Cbasins)

def getFloorCbath (slab):

    Cbath = 41.5
    for relDefinesByProperties in slab.IsDefinedBy:
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProperties:
                    if Properties.Name == "Cbath":
                        Cbath = Properties.NominalValue.wrappedValue
                        return (Cbath)


def getFloorCKitchen (slab):

    Ckitchen = 0.5
    for relDefinesByProperties in slab.IsDefinedBy:
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProperties:
                    if Properties.Name == "CKitchen":
                        Ckitchen = Properties.NominalValue.wrappedValue
                        return (Ckitchen)

```

```
def getFloorCPerPerson (slab):

941.
    CPerPerson = 0.5
942.
    for relDefinesByProperties in slab.IsDefinedBy:
943.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
944.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
945.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
946.
                    if Properties.Name == "CPerPerson":
947.
                        CPerPerson = Properties.NominalValue.wrappedValue
948.                     return (CPerPerson)
949.


 def getFloorCshower (slab):

950.
    Cshower = 0.5
951.
    for relDefinesByProperties in slab.IsDefinedBy:
952.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
953.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
954.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
955.
                    if Properties.Name == "Cshower":
956.
                        Cshower = Properties.NominalValue.wrappedValue
957.                     return (Cshower)
958.



  def getFloorQstandacairculationLoss (slab):

959.
    QstandCirculationLoss = 4000000000.0
960.
    for relDefinesByProperties in slab.IsDefinedBy:
961.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
962.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
963.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
964.
                    if Properties.Name == "QstandCirculationLoss":
965.
                        QstandCirculationLoss = Properties.NominalValue.wrappedValue
966.                     return (QstandCirculationLoss)
```

```
967.


def getFloorQstandingStill (slab):

968.
    QstandindgStill = 4220000000.0
969.
    for relDefinesByProperties in slab.IsDefinedBy:
970.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
971.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
972.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
973.
                    if Properties.Name == "QstandingStill":
974.
                        QstandingStill = Properties.NominalValue.wrappedValue
975.                     return (QstandingStill)
976.

def getFloorTap (slab):

977.
    Tap = 0
978.
    for relDefinesByProperties in slab.IsDefinedBy:
979.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
980.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
981.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
982.
                    if Properties.Name == "Tap":
983.
                        Tap = Properties.NominalValue.wrappedValue
984.                     return (Tap)
985.

def getFloorBoilerEfficiency (slab):

986.
    BoilerEfficiency = 0.9
987.
    for relDefinesByProperties in slab.IsDefinedBy:
988.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
989.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
990.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
991.
                    if Properties.Name == "BoilerEfficiency":
```

```
992.
                        BoilerEfficiency = Properties.NominalValue.wrappedValue
993.                    return (BoilerEfficiency)
994.

def getFloorBoilerEfficiency (slab):

995.
    BoilerEfficiency = 0.9
996.
    for relDefinesByProperties in slab.IsDefinedBy:
997.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
998.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
999.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
1000.
                    if Properties.Name == "BoilerEfficiency":
1001.
                        BoilerEfficiency = Properties.NominalValue.wrappedValue
1002.                   return (BoilerEfficiency)
1003.


def getFloorBoilerEfficiency (slab):

1004.
    BoilerEfficiency = 0.9
1005.
    for relDefinesByProperties in slab.IsDefinedBy:
1006.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
1007.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
1008.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
1009.
                    if Properties.Name == "BoilerEfficiency":
1010.
                        BoilerEfficiency = Properties.NominalValue.wrappedValue
1011.                   return (BoilerEfficiency)
1012.

def getFloorCorrectionFactor (slab):

1013.
    CorrectionFactor = 0.9
1014.
    for relDefinesByProperties in slab.IsDefinedBy:
1015.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
1016.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
1017.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
```

```
1018.
                    if Properties.Name == "CorrectionFactor":
1019.
                        CorrectionFactor = Properties.NominalValue.wrappedValue
1020.                   return (CorrectionFactor)
1021.

def getFloorNumberOfBathsPerPersonPerDay (slab):

1022.
    NumberOfBathsPerPersonPerDay = 0.096
1023.
    for relDefinesByProperties in slab.IsDefinedBy:
1024.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
1025.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
1026.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
1027.
                    if Properties.Name == "NumberOfBathsPerPersonPerDay":
1028.
                        NumberOfBathsPerPersonPerDay = Properties.NominalValue.wrappedValue

1029.                   return (NumberOfBathsPerPersonPerDay)
1030.

def getFloorNumberOfShowersPerPersonPerDay (slab):

1031.
    NumberOfShowersPerPersonPerDay = 0.61
1032.
    for relDefinesByProperties in slab.IsDefinedBy:
1033.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
1034.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
1035.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
1036.
                    if Properties.Name == "NumberOfShowersPerPersonPerDay":
1037.
                        NumberOfShowersPerPersonPerDay = Properties.NominalValue.wrappedVal
    ue
1038.                   return (NumberOfShowersPerPersonPerDay)
1039.

1040.

def getFloorSavingShowerHead (slab):

1041.
    SavingShowerHead = 1
1042.
    for relDefinesByProperties in slab.IsDefinedBy:
1043.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
```

```
1044.
          if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
1045.
               for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
   erties:
1046.
                    if Properties.Name == " SavingShowerHead":
1047.
                         SavingShowerHead = Properties.NominalValue.wrappedValue
1048.                    return (SavingShowerHead)
1049.

def getFloorηuseful (slab):

1050.
   ηuseful = 0.44
1051.
   for relDefinesByProperties in slab.IsDefinedBy:
1052.
       if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
1053.
          if relDefinesByProperties.RelatingPropertyDefinition.is_a("Floor Schedule"):
1054.
               for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
   erties:
1055.
                    if Properties.Name == " ηuseful":
1056.
                         ηuseful = Properties.NominalValue.wrappedValue
1057.                    return (ηuseful)
1058.

def getSpaceArea (space) :

1059.
   Area = 0
1060.
   for relDefinesByProperties in space.IsDefinedBy:
1061.
       if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
1062.
          if relDefinesByProperties.RelatingPropertyDefinition.is_a("Quantities"):
1063.
               for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
   erties:
1064.
                    if Properties.Name == "Area":
1065.
                        Area = Properties.NominalValue.wrappedValue
1066.                    return (Area)
1067.
1068.                              # For (lighting)#

def getLightFixtureTypeTotalWattage (LightFixtureType):

1069.
   TotalWattage = 0
1070.
   for relDefinesByProperties in LightFixture.IsDefinedBy:
```

```
1071.
       if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
1072.
           if relDefinesByProperties.RelatingPropertyDefinition.is_a("Pset_LightFixtureTyp
   eCommon"):
1073.
               for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
   erties:
1074.
                   if Properties.Name == "TotalWattage":
1075.
                       TotalWattage = Properties.NominalValue.wrappedValue
1076.                  return (TotalWattage)
1077.

def getSpaceQlighting (space):

1078.
   Qlighting = 0
1079.
   for relDefinesByProperties in space.IsDefinedBy:
1080.
       if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
1081.
           if relDefinesByProperties.RelatingPropertyDefinition.is_a("Space(Qlighting)"):

1082.
               for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
   erties:
1083.
                   if Properties.Name == "Qlighting":
1084.
                       ActualLightingLoad = Properties.NominalValue.wrappedValue
1085.                  return (Qlighting)
1086.
```

```
                                #For Qinternal#
```

```
1087.

def getSpaceNumberOfpeople (space):

1088.
    NumberOfPeople = 0
1089.
    for relDefinesByProperties in space.IsDefinedBy:
1090.
       if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
1091.
           if relDefinesByProperties.RelatingPropertyDefinition.is_a("Qinternal (spaces)")
   :
1092.
               for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
   erties:
1093.
                   if Properties.Name == "NumberOfPeople":
1094.
                       NumberOfPeople = Properties.NominalValue.wrappedValue
1095.                  return (NumberOfPeople)
```

```
1096.

def getSpaceNliving (space):

1097.
    Nliving = 0
1098.
    for relDefinesByProperties in space.IsDefinedBy:
1099.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
1100.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Qinternal (spaces)")
    :
1101.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
1102.
                    if Properties.Name == "Nliving":
1103.
                        Nliving = Properties.NominalValue.wrappedValue
1104.                   return (Nliving)
1105.
1106.                       #For walls (heat loss through transmission)
1107.

def getWallArea (wall):

1108.
    Area = 0
1109.
    for relDefinesByProperties in space.IsDefinedBy:
1110.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
1111.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Quantities"):
1112.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
1113.
                    if Properties.Name == "Area":
1114.
                        Area = Properties.NominalValue.wrappedValue
1115.                   return (Area)
1116.

def getWallHeatTrasnferCoefficient (wall):

1117.
    HeatTransferCoefficient = 0
1118.
    for relDefinesByProperties in wall.IsDefinedBy:
1119.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
1120.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Wall Schedule 2"):
1121.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
```

```
1122.
                    if Properties.Name == "HeatTransferCoefficient":
1123.
                        HeatTransferCoefficient = Properties.NominalValue.wrappedValue
1124.                    return (HeatTransferCoefficient)
1125.
1126.                        #For window heat gain by solar radiation

def getWindowAverageEffectiveTotalSolarAccessionFactor (opening):

1127.
    AverageEffectiveTotalSolarAccessionFactor = 4.815
1128.
    for relDefinesByProperties in window.IsDefinedBy:
1129.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
1130.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Window_SolarRadiatio
    n"):
1131.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
1132.
                    if Properties.Name == "AverageEffectiveTotalSolarAccessionFactor":
1133.
                        AverageEffectiveTotalSolarAccessionFactor = Properties.NominalValue
    .wrappedValue
1134.                    return (AverageEffectiveTotalSolarAccessionFactor)
1135.

def getWindowCorrectionFactorForNonScatteringGlazing (opening):

1136.
    CorrectionFactorForNonScatteringGlazing = 0.9
1137.
    for relDefinesByProperties in window.IsDefinedBy:
1138.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
1139.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Window_SolarRadiatio
    n"):
1140.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
1141.
                    if Properties.Name == "CorrectionFactorForNonScatteringGlazing":
1142.
                        CorrectionFactorForNonScatteringGlazing = Properties.NominalValue.w
    rappedValue
1143.                     return (CorrectionFactorForNonScatteringGlazing)
1144.


def getWindowCosineFraction (opening):

1145.
    CosineFraction = 0.25
1146.
    for relDefinesByProperties in window.IsDefinedBy:
```

```
1147.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
1148.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Window_SolarRadiatio
    n"):
1149.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
1150.
                    if Properties.Name == "CosineFraction":
1151.
                        CosineFraction = Properties.NominalValue.wrappedValue
1152.                     return (CosineFraction)
1153.
1154.      def getWindowSolarAccessionFactor (opening):
1155.
     SolarAccessionFactor = 0.75
1156.
     for relDefinesByProperties in window.IsDefinedBy:
1157.
        if relDefinesByProperties.is_a("IfcRelDefinesByProperties"):
1158.
            if relDefinesByProperties.RelatingPropertyDefinition.is_a("Window_SolarRadiatio
    n"):
1159.
                for Properties in relDefinesByProperties.RelatingPropertyDefinition.HasProp
    erties:
1160.
                    if Properties.Name == "SolarAccessionFactor":
1161.
                        SolarAccessionFactor = Properties.NominalValue.wrappedValue
1162.                     return (SolarAccessionFactor)

def getOpeningLengthOrHeightOrArea (opening,lengthOrWidthOrHeightOrAreaOrVolume ):

1163.
    length= opening.OverallWidth
1164.
    height= opening.OverallHeight
1165.
    area= length * height
1166.
    if lengthOrWidthOrHeightOrAreaOrVolume == "length":
1167.
        return (length)
1168.
    if lengthOrWidthOrHeightOrAreaOrVolume == "height":
1169.
        return (height)
1170.
    if lengthOrWidthOrHeightOrAreaOrVolume == "area":
1171.
        return (area)
1172.


1173.
    def __init__(self, Main_Project = "E:/courses/graduation/Automation of building energy
    efficiency assessment using BIM/Revit models/Ifc_Model.ifc"):
1174.
        Main_Project = self.ifc_file.by_type("Ifc_Model")
```

```
1175.
        northDirection = getNorthDirection(Main_Project)
1176.
        print (northDirection)
1177.
    def __init__(self, wallStandardCaseList, openingsList, SpacesList, slabsList, IfcLightF
    ixtureTypesList):
1178.
        wallStandardCaseList = self.ifc_file.by_type("IfcWall")
1179.
        openingsList = self.ifc_file.by_type("IfcWindow") + self.ifc_file.by_type("IfcDoor"
    )
1180.
        SpacesList = self.ifc_file.by_type("IfcSpace")
1181.
        slabsList = self.ifc_file.by_type("IfcSlab")
1182.
        IfcLightFixtureTypesList = self.ifc_file.by_type("IfcLightFixtureType")
1183.
        self.wallList = []
1184.
        for wall in wallStandardCaseList:
1185.
            internalOrExternal = "internal"
1186.
            axisDirection = getWallAxisDirection(wall)
1187.
            length = getWallLengthWidthHeight(wall, "length")
1188.
            width = getWallLengthWidthHeight(wall, "width")
1189.
            height = getWallLengthWidthHeight(wall, "height")
1190.
            area = getWallLengthWidthHeight(wall, "area")
1191.
            (thermalMass, thermalResistance, heatTransferCoefficient, absorptance,solarHeat
    GainCoefficient) = getThermalValues(wall)
1192.
            thermalTransmittance = getThermalTransmittance(wall)
1193.
            forTotal = 0
1194.
        if checkIfElementIsExternal(wall):
1195.
            internalOrExternal = "external"
1196.
            temporaryWall = myWall \
1197.
                (wall.GlobalId, internalOrExternal, [], [], [], [], axisDirection, length,
    width, height, [], [], area, [], forTotal, [], thermalMass, thermalResistance, heatTran
    sferCoefficient, absorptance, thermalTransmittance, solarHeatGainCoefficient)
1198.
            self.wallList.append(temporaryWall)
1199.
            allWidths = getAllWidths(self.wallList)
1200.
            for wall in self.wallList:
1201.
                getPossibleCoordinates(wall, allWidths)
1202.
```

```
1203.
            externalWallList = []
1204.
            internalWallList = []
1205.
            for wall in self.wallList:
1206.
                if wall.internalOrExternal == "external":
1207.
                   externalWallList.append(wall)
1208.
            for wall in self.wallList:
1209.
                if wall.internalOrExternal == "internal":
1210.
                    internalWallList.append(wall)
1211.
            checklist = []
1212.
            for index, wall in enumerate(externalWallList):
1213.
                checklist.append(externalWallList[index])
1214.
                print (externalWallList)
1215.
                print (checklist)
1216.

1217.
                self.totalExternalArea = 0
1218.
                self.totalExternalVolume = 0
1219.
                (self.totalExternalArea, self.totalExternalVolume) = defineFinalCoordinates
    _defineAxisOrientation_getTotalExternalArea(checklist, self.totalExternalArea, self.tot
    alExternalVolume)
1220.

1221.
                for wall in externalWallList:
1222.
                    getExternalWallOrientation(northDirection, wall)
1223.
                getFinalCoordiatesForInternalWall(internalWallList, externalWallList)
1224.
                getInternalWallAxisDirection(internalWallList)
1225.
                getConnectedToInCorrectOrder(self.wallList)
1226.

1227.
                self.northWallArea = 0
1228.
                self.southWallArea = 0
1229.
                self.eastWallArea = 0
1230.
                self.westWallArea = 0
1231.
                for wall in externalWallList:
```

```
1232.
                    if wall.orientation == "North":
1233.
                        self.northWallArea = self.northWallArea + wall.area
1234.
                    if wall.orientation == "South":
1235.
                        self.southWallArea = self.southWallArea + wall.area
1236.
                    if wall.orientation == "East":
1237.
                        self.eastWallArea = self.eastWallArea + wall.area
1238.
                    if wall.orientation == "West":
1239.
                        self.westWallArea = self.westWallArea + wall.area
1240.

1241.
    #for the openings
1242.
            self.openingList = []
1243.
            for opening in openingsList:
1244.
                doorOrWindow = "door"
1245.
                if opening.is_a("IfcWindow"):
1246.
                    doorOrWindow = "window"
1247.
                    length = getOpeningLengthOrHeightOrArea(opening, "length")
1248.
                    height = getOpeningLengthOrHeightOrArea(opening, "height")
1249.
                    area = getOpeningLengthOrHeightOrArea(opening, "area")
1250.
                    orientation = getOpeningOrientation(opening)
1251.
                    internalOrExternal = checkIfOpeningIsExternal(opening)
1252.
                    (thermalMass, thermalResistance, heatTransferCoefficient, absorptance,
    solarHeatGainCoefficient, AverageEffectiveTotalSolarAccessionFactor, CorrectionFactorFo
    rNonScatteringGlazing, CosineFraction, SolarAccessionFactor) = getThermalValues(opening
    )
1253.
                    thermalTransmittance = getThermalTransmittance(opening)
1254.
                    temporaryOpening = myOpening(opening.GlobalId, doorOrWindow, internalOr
    External, length, height, area, orientation, thermalResistance, heatTransferCoefficient
    , absorptance, solarHeatGainCoefficient, AverageEffectiveTotalSolarAccessionFactor, Cor
    rectionFactorForNonScatteringGlazing, CosineFraction, SolarAccessionFactor)
1255.
                    self.openingList.append(temporaryOpening)
1256.
                    externalOpeningList = []
1257.
                    for opening in self.openingList:
1258.
                        if opening.internalOrExternal == "external":
```

```
                            externalOpeningList.append(opening)
                    self.northOpeningArea = 0
                    self.southOpeningArea = 0
                    self.eastOpeningArea = 0
                    self.westOpeningArea = 0
                    for opening in externalOpeningList:
                        if opening.orientation == "North":
                            self.northOpeningArea = self.northOpeningArea + opening.area
                        if opening.orientation == "South":
                            self.southOpeningArea = self.southOpeningArea + opening.area
                        if opening.orientation == "East":
                            self.eastOpeningArea = self.eastOpeningArea + opening.area
                        if opening.orientation == "West":
                            self.westOpeningArea = self.westOpeningArea + opening.area

            #Spaces
                self.spaceList = []
                for space in spacesList:
                    length = getSpaceLengthWidthHeight(space, "length")
                    width = getSpaceLengthWidthHeight(space, "width")
                    height = getSpaceLengthWidthHeight(space, "height")
                    area = getSpaceLengthWidthHeight(space, "area")
                    (thermalMass,thermalResistance, heatTransferCoefficient, absorptance, t
    hermalTransmittance, solarHeatGainCoefficient, Roughness, NumberOfPeople, Qlighting, Sp
    ecifiedLightingLoad, InternalHeatGain, EnergyRequirementForLighting, EnergyRequirementF
    orParasiticPower, Nliving) = getThermalValues(space)
                    thermalTransmittance = getThermalTransmittance(opening)
                    temporaryOpening = myOpening(space.globalId, area, IfcGUID, length, hei
    ght, area, orientation, thermalResistance, heatTransferCoefficient, absorptance, solarH
    eatGainCoefficient, AverageEffectiveTotalSolarAccessionFactor, CorrectionFactorForNonSc
    atteringGlazing, CosineFraction, SolarAccessionFactor)
                    self.openingList.append(temporaryOpening)
```

```
1286.
            #Floors
1287.
                self.slabList = []
1288.
                for slab in slabsList:
1289.
                    length = getSlabLengthWidthHeight(slab, "length")
1290.
                    width = getSlabLengthWidthHeight(slab, "width")
1291.
                    height = getSlabLengthWidthHeight(slab, "height")
1292.
                    area = getSlabLengthWidthHeight(slab, "area")
1293.
                    (thermalMass,thermalResistance, heatTransferCoefficient, absorptance, R
    oughness, Cbasins, Cbath, Ckitchen, CPerPerson, Cshower, QstandCirculationLoss, Qsatndi
    ngStill, Tap, BoilerEfficiency, CorrectionFactor, NumberOfBathsPerPersonPerDay, SavingS
    howerHead, NumberOfShowersPerPersonPerDay, ηuseful, HeatGainByHotWater, AuxilaryHotWate
    r, QstandingStill) = getThermalValues(slab)
1294.
                    thermalTransmittance = getThermalTransmittance(slab)
1295.
                    temporarySlab = mySlab(slab.globalId, area, IfcGUID, length, height, ar
    ea, thermalMass, thermalResistance, heatTransferCoefficient, absorptance, Roughness, Cb
    asins, Cbath, Ckitchen, CPerPerson, Cshower, QstandCirculationLoss, QsatndingStill, Tap
    , BoilerEfficiency, CorrectionFactor, NumberOfBathsPerPersonPerDay, SavingShowerHead, N
    umberOfShowersPerPersonPerDay, ηuseful, HeatGainByHotWater, AuxilaryHotWater, Qstanding
    Still)
1296.
                    self.slabList.append(temporaryOpening)
1297.

1298.
            #Light Fixtures
1299.

1300.
                self.LightFixtureTypeList = []
1301.
                for LightFixtureType in LightFixturesTypesList:
1302.
                    TotalWattage = getTotalWattage(LightFixtureType)
1303.
                    temporaryLightFixtureType = myLightFixrureType(lightfixturetype.totalwa
    ttage)
1304.
                    self.LightFixtureTypeList.append(temporaryLightFixtureType)
1305.

1306.
1307.         #heat calculations
1308.       self.getAllEnergyResults()
1309.
1310.        self.lastStep()
1311.
1312.        def getAllEnergyResults(self):
1313.
     Tlength = 720
1314.
     TransmissionHeatTransferCoefficient = 1.8
```

```
1315.
    Tin = 22
1316.
    Tout = 26
1317.
    WallHeatTransmissionCalculation = 0
1318.
    HeatThroughVentilation = 0
1319.
    SpaceInternalHeatGain = 0
1320.
    densityOfAir = 1.205
1321.
    heatCapacityOfAir = 1005
1322.
    FloorHeatGainByHotWater = 0
1323.
    AuxilaryHotWater = 0
1324.
    b = 1
1325.
    f = 1
1326.
    q = 1
1327.
    Npeople = 3
1328.
    VentilationHeatTransferCoefficient = densityOfAir * heatCapacityOfAir * ((b * f * q) /
    3600)
1329.
    Fbuilding = 1
1330.
    Fsolar = 0.95
1331.
    Fpvt = 1
1332.
    QH = 121.7
1333.
    self.WallHeatThroughTransmission = 0
1334.
    self.OpeningHeatThroughVentilation = 0
1335.
    self.SpaceInternalHeatGain = 0
1336.
    self.HeatGainByHotTapWater = 0
1337.
    self.SpaceHeatGainByLighting = 0
1338.
    self.OpeningSolarRadiation = 0
1339.
    self.northOpeningSolarRadiationCalculation = 0
1340.
    self.FloorTap = 0
1341.
    self.FloorHeatGainByHotWater = 0
1342.
    self.HeatGainMainBoiler = 0
1343.
    self.AuxilaryHotWater = 0
1344.
    self.EnergyRequirementForLighting = 0
```

```
        self.EnergyRequirementForParasiticPower = 0
        self.SolarBoiler = 0
        self.HeatGainByPV = 0
        self.HeatGainByCogeneration = 0
        self.distributionsystemefficiency = 1
        self.systemefficiency = 1
        self.PilotFlame = 2500



    #For Ventilation
    for opening in self.openingList:
        HeatThroughVentilation = VentilationHeatTransferCoefficient * (Tin -
    Tout) * Tlength
        HeatThroughVentilationCalculation = HeatThroughVentilation + VentilationHeatTransf
   erCoefficient * (Tin - Tout) * Tlength
        self.HeatLossThroughVentilation = self.OpeningHeatThroughVentilation + HeatThrough
   Ventilation

    #For heat loss by transmission
    for wall in self.wallList:
        HeatLossThroughTransmission = area * heatTransferCoefficient * (Tin -
    Tout) * (0.001 * Tlength)
        self.HeatLossThroughTransmission = self.WallHeatThroughTransmission +  HeatLossThr
   oughTransmission
    #create results
        self.wallCalculation.append(round(HeatLossThroughTransmission, 1))
        self.ventilationCalculation.append(round(HeatThroughVentilation,1))

   # internal heat gain
    for space in self.spaceList:
        InternalHeatGain = 180 * space.NumberOfPeople * space.Nliving
        InternalHeatGainCalculation = SpaceInternalHeatGain + InternalHeatGain
        self.InternalHeatGain = self.SpaceInternalHeatGain + InternalHeatGain
```

```
1372.
        self.InternalHeatCalculation.append(round(InternalHeatGain,1))
1373.

1374.
    # heat gain from hot tap water
1375.
     for floor in self.floorList:
1376.
        Tap = floor.Ckitchen + floor.Cbasins + floor.Npeople * (floor.CPerPerson + (floor.
    Cshower * floor.SavingShowerHead * floor.NumberOfBathsPerPersonPerDay) + (floor.Cbath *
     floor.NumberOfBathsPerPersonPerDay * floor.NumberOfShowersPerPersonPerDay ))
1377.
        TapCalculation = self.FloorTap + Tap
1378.
        HeatGainMainBoiler = ((TapCalculation / floor.BoilerEfficiency) * floor.Correction
    Factor) + floor.QstandindgStill + (floor.QstandCirculationLoss * (floor.area / 100))
1379.
        HeatGainMainBoilerCalculation = self.HeatGainMainBoiler + HeatGainMainBoiler
1380.
        HeatGainByHotWater = self.FloorHeatGainByHotWater + Tap + HeatGainMainBoiler + Aux
    ilaryHotWater
1381.
        HeatGainByHotWaterCalculation = HeatGainMainBoilerCalculation +  TapCalculation +
    self.AuxilaryHotWater + FloorHeatGainByHotWater
1382.
        self.FloorHeatGainByHotWater = self.HeatGainByHotWater + HeatGainByHotWaterCalcula
    tion
1383.
        self.HeatGainByHotWaterCalculation.append(round(HeatGainByHotWater,1))
1384.

1385.
    # Heat Gain From Lighting
1386.
     for space in self.spaceList:
1387.
        for LightFixtureType in self.LightFixtureTypeList:
1388.
            EnergyRequirementForLighting = space.area * LightFixtureType.TotalWattage
1389.
            EnergyRequirementForLightingCalculation = self.EnergyRequirementForLighting + E
    nergyRequirementForLighting
1390.
            EnergyRequirementForParasiticPower = 0
1391.
            EnergyRequirementForParasiticPowerCalculation = self.EnergyRequirementForParasi
    ticPower + EnergyRequirementForParasiticPower
1392.
            Qlighting = EnergyRequirementForLighting +  EnergyRequirementForParasiticPower

1393.
            QlightingCalculation = self.SpaceHeatGainByLighting + Qlighting
1394.
            self.SpaceHeatGainByLighting = self.Qlighting + Qlighting
1395.
            self.SpaceHeatGainByLightingCalculation.append(round(Qlighting,1))
1396.

1397.
```

```
1398.    # Heat gain by solar radiation (June)
1399.
1400.            northIncidentSolarRadiationByOrientaion = 81.6
1401.            eastIncidentSolarRadiationByOrientaion = 130.6
1402.            southIncidentSolarRadiationByOrientaion = 123.4
1403.            westIncidentSolarRadiationByOrientaion = 143.6
1404.            northShadowReductionFactor = 1
1405.            eastShadowReductionFactor = 0.85
1406.            southShadowReductionFactor = 0.9
1407.            westShadowReductionFactor = 0.85
1408.            heatTransferResistanceOutside = 0.04
1409.
1410.            self.northOpeningSolarRadiationCalculation = 0
1411.            self.southOpeningSolarRadiationCalculation = 0
1412.            self.westOpeningSolarRadiationCalculation = 0
1413.            self.eastOpeningSolarRadiationCalculation = 0
1414.
1415.            self.SolarRadiation = []
1416.            SolarRadiation = 0
1417.            self.northOpeningSolarRadiationCalculation = 0
1418.            self.southOpeningSolarRadiationCalculation = 0
1419.            self.westOpeningSolarRadiationCalculation = 0
1420.            self.eastOpeningSolarRadiationCalculation = 0
1421.
1422.
1423.            for opening in self.openingList:
1424.                if opening.orientation == "North":
1425.                    IncidentSolarRadiationByOrientaion = northIncidentSolarRadiation
1426.                    ShadingReductionFactor = northShadowReductionFactor
1427.                else:
```

```
1428.
                    CollectorArea = opening.absorptance * heatTransferResistanceOutside * o
   pening.heatTransferCoefficient * opening.area
1429.
                    HeatGainBySolarRadiation = (northShadowReductionFactor * CollectorArea
   * IncidentSolarRadiationByOrientation * opening.AverageEffectiveTotalSolarAccessionFact
   or * (1 - opening.CosineFraction) * (0.001 * Tlength))
1430.
                    print(HeatGainBySolarRadiation)
1431.
                    self.northOpeningSolarRadiationCalculation = self.HeatGainBySolarRadiat
   ion + HeatGainBySolarRadiation
1432.
                    SolarRadiation = SolarRadiation + HeatGainBySolarRadiation
1433.
                    print(self.HeatGainBySolarRadiation)
1434.
                    if opening.orientation == "East":
1435.
                        IncidentSolarRadiationByOrientaion = eastIncidentSolarRadiation
1436.
                        ShadingReductionFactor = eastShadowReductionFactor
1437.
                    else:
1438.
                        CollectorArea = opening.absorptance * heatTransferResistanceOutside
    * opening.heatTransferCoefficient * opening.area
1439.
                        solarRadiationCalculation = (eastShadowReductionFactor * CollectorA
   rea * IncidentSolarRadiationByOrientation * opening.AverageEffectiveTotalSolarAccession
   Factor * (1 - opening.CosineFraction) * (0.001 * Tlength))
1440.
                        self.eastOpeningSolarRadiationCalculation = self.eastOpeningSolarRa
   diationCalculation + solarRadiationCalculation
1441.
                        HeatGainBySolarRadiation = self.HeatGainBySolarRadiation + solarRad
   iationCalculation
1442.
                    if opening.orientation == "West":
1443.
                        IncidentSolarRadiationByOrientaion = westIncidentSolarRadiation
1444.
                        ShadingReductionFactor = westShadowReductionFactor
1445.
                    else:
1446.
                        CollectorArea = opening.absorptance * heatTransferResistanceOutside
    * opening.heatTransferCoefficient * opening.area
1447.
                        solarRadiationCalculation = (westShadowReductionFactor * CollectorA
   rea * IncidentSolarRadiationByOrientation * opening.AverageEffectiveTotalSolarAccession
   Factor * (1 - opening.CosineFraction) * (0.001 * Tlength))
1448.
                        self.westOpeningSolarRadiationCalculation = self.westOpeningSolarRa
   diationCalculation + solarRadiationCalculation
1449.
                        HeatGainBySolarRadiation = self.HeatGainBySolarRadiation + solarRad
   iationCalculation
1450.
                    if opening.orientation == "South":
```

```
1451.
                        IncidentSolarRadiationByOrientaion = westIncidentSolarRadiation
1452.
                        ShadingReductionFactor = southShadowReductionFactor
1453.
                    else:
1454.
                        CollectorArea = opening.absorptance * heatTransferResistanceOutside
     * opening.heatTransferCoefficient * opening.area
1455.
                        solarRadiationCalculation = (southShadowReductionFactor * Collector
    Area * IncidentSolarRadiationByOrientation * opening.AverageEffectiveTotalSolarAccessio
    nFactor * (1 - opening.CosineFraction) * (0.001 * Tlength))
1456.
                        self.southOpeningSolarRadiationCalculation = self.southOpeningSolar
    RadiationCalculation + solarRadiationCalculation
1457.
                        HeatGainBySolarRadiation = self.HeatGainBySolarRadiation + solarRad
    iationCalculation
1458.
            self.OpeningSolarRadiation.append(round((HeatGainBySolarRadiation),1))
1459.
            self.OpeningSolarRadiation = self.northOpeningSolarRadiationCalculation + HeatG
    ainBySolarRadiation * 0.0036
1460.
            self.eastOpeningSolarRadiationCalculation = self.eastOpeningSolarRadiationCalcu
    lation * 0.0036
1461.
            self.southOpeningSolarRadiationCalculation = self.southOpeningSolarRadiationCal
    culation * 0.0036
1462.
            self.westOpeningSolarRadiationCalculation = self.westOpeningSolarRadiationCalcu
    lation * 0.0036
1463.

1464.

1465.
    #Heat Demand
1466.

1467.
            self.HeatDemand = self.wallCalculation + self.ventilationCalculation + self.Int
    ernalHeatCalculation + (self.northOpeningSolarRadiationCalculation + self.eastOpeningSo
    larRadiationCalculation + self.southOpeningSolarRadiationCalculation + self.westOpening
    SolarRadiationCalculation)
1468.
1469.
    # Space heating Calculation
1470.

1471.
            self.SpaceHeating = (((self.HeatDemand / self.distributionsystemefficiency) -
     self.SolarBoiler) / self.systemefficiency) + self.PilotFlame
1472.

1473.
    # Qtotal
1474.
```

```
1475.
            self.Qtotal = self.SpaceHeating + self.FloorHeatGainByHotWater + self.SpaceHeat
    GainByLighting + self.HeatGainByPV + self.HeatGainByCogeneration
1476.

1477.
    #Energy Index calculation
1478.

1479.
            self.EnergyIndex = self.Qtotal / ((155 * floor.area ) + 9560)
1480.

1481.
    # Take Heat Demand
1482.

1483.
            self.HeatGains =[]
1484.
            self.HeatLosses =[]
1485.

1486.
            for row in range(0, len(self.OpeningSolarRadiation)):
1487.
                self.HeatGains.append((self.InternalHeatCalculation[row] + self.OpeningSola
    rRadiation[row] + self.HeatGainByHotWaterCalculation[row] + self.HeatGainByPV[row] + se
    lf.HeatGainByCogeneration[row] + self.SpaceHeatGainByLightingCalculation[row]))
1488.
                self.HeatLosses.append ((self.wallCalculation[row] + self.ventilationCalcul
    ation[row]))
1489.
            self.heatingDemand = []
1490.
            for row in range(0, len(self.heatGains)):
1491.
                if row >=4 and row <=8:
1492.
                    self.heatingDemand.append(0)
1493.
                else:
1494.
                    self.heatingDemand.append((self.heatLosses[row]+ self.heatGains[row]))

1495.

1496.
            print ("Heat Demand", self.heatingDemand)
1497.
            print ("Inernal Heat = ", self.internalHeatCalculation)
1498.
            print ("solar Radiation= ", self.OpeningSolarRadiation)
1499.
            print (" Wall Transmission", self.wallCalculation)
1500.
            print ("Window Ventilation", self.ventilationCalculation)
1501.
            print ("Water Heating", self.HeatGainByHotWaterCalculation)
1502.
            print ("Heat By Lighting", self.SpaceHeatGainByLightingCalculation)
```

```
1503.
            print ("Heat By Pv", self.HeatGainByPV)
1504.
            print ("Heat By Cogeneration", self.HeatGainByCogeneration)
1505.
            print ("heat gains = ", self.heatGains)
1506.
            print ("heat losses = ", self.heatLosses)
1507.
            print (sum(self.heatingDemand))
1508.

1509.
    #SAVE AND SHOW THE RESULTS
1510.
            def lastStep(self):
1511.
                self.resultsSaved()
1512.
                self.getResultsOnImage()
1513.
                self.getFeedback()
1514.

1515.
            def resultsSaved(self):
1516.

1517.
                #Final results are added to the final list
1518.
                self.energyResults.append((round(sum(self.internalHeatCalculation),2), roun
    d(sum(self.OpeningSolarRadiation) + sum(self.wallCalculation) + sum(self.ventilationCal
    culation),2), round(sum(self.HeatGainByHotWaterCalculation),2), round(sum(self.SpaceHea
    tGainByLightingCalculation),2), round(sum(self.HeatGainByPV),2), round(sum(self.HeatGai
    nByCogeneration),2), round(-(sum (self.heatingDemand)),2)))
1519.
                self.tipResultsSaved()
1520.

1521.
                def tipResultsSaved(self):
1522.
                    self.tipResults.append((round(sum(self.internalHeatCalculation),2), rou
    nd(sum(self.OpeningSolarRadiation) + sum(self.wallCalculation) + sum(self.ventilationCa
    lculation),2), round(sum(self.HeatGainByHotWaterCalculation),2), round(sum(self.SpaceHe
    atGainByLightingCalculation),2), round(sum(self.HeatGainByPV),2), round(sum(self.HeatGa
    inByCogeneration),2), round(-(sum (self.heatingDemand)),2)))
1523.

1524.
    #Draw results on the image  (GKZATIOS, 2017)
1525.
            def getResultsOnImage(self):
1526.
                image = Image.open('3D Model.png')
1527.
                path = r'E:/courses/graduation/Automation of building energy efficiency ass
    essment using BIM'
1528.
                draw = ImageDraw.Draw(image)
```

```
1529.
                font = ImageFont.truetype("arial.ttf", 30)
1530.
                colorBlack = 'rgb(0, 0, 0)'
1531.
                colorRed = 'rgb(255, 0, 0)'
1532.
                colorGreen = 'rgb(0, 128, 0)'
1533.
                text0 = "Heating Demand"
1534.
                (x10, y10) = (50, 50)
1535.
                text01 = "Heat Gain"
1536.
                (x101, y101) = (400,50)
1537.
                text1 = "Internal Heat"
1538.
                (x11, y11) = (320, 550)
1539.
                text2 = "Heat Loss"
1540.
                (x12, y12) = (680, 450)
1541.
                text3 = "SolarRadiation"
1542.
                (x13, y13) = (680, 150)
1543.
                text4 = "Water Heating"
1544.
                (x14, y14) = (680, 150)
1545.
                text5 = "Lighting"
1546.
                (x15, y15) = (680, 150)
1547.
                text6 = "Heat By Cogeneration"
1548.
                (x16, y16) = (680, 150)
1549.
                text7 = "Heat By Pv"
1550.
                (x17, y17) = (680, 150)
1551.

1552.
                draw.text((x10, y10), text0, fill=colorBlack, font=font)
1553.
                draw.text((x101, y101), text01, fill=colorBlack, font=font)
1554.
                draw.text((x11, y11), text1, fill=colorBlack, font=font)
1555.
                draw.text((x12, y12), text2, fill=colorBlack, font=font)
1556.
                draw.text((x13, y13), text3, fill=colorBlack, font=font)
1557.
                draw.text((x14, y14), text4, fill=colorBlack, font=font)
1558.
                draw.text((x15, y15), text5, fill=colorBlack, font=font)
```

```
1559.
                draw.text((x16, y16), text6, fill=colorBlack, font=font)
1560.
                draw.text((x17, y17), text7, fill=colorBlack, font=font)
1561.
1562.
            if len(self.energyResults) == 1:
1563.
                for row in self.energyResults:
1564.
                    (x1, y1) = (320, 600)
1565.
                    internalHeatText = "+" + " " + str(row[0]) + "J"
1566.
                    (x2, y2) = (680, 500)
1567.
                    heatGain = row[1]
1568.
                    if heatGain >= 0:
1569.
                        mark = "+"
1570.
                    else:
1571.
                        mark = ""
1572.
                        heatGainText = mark + " " + str(row[1]) + "J"
1573.
                        (x3, y3) = (680, 200)
1574.
                        solarHeatText = "+" + " " + str(row[2]) + "J"
1575.
                        (x4, y4) = (50, 100)
1576.
                        HeatingDemandText = "+" + " " + str(row[3]) + "J"
1577.
                        (x5, y5) = (400,100)
1578.
                        HeatLossText = "+" + " " + str(row[4]) + "J"
1579.
                        (x6, y6) = (400,100)
1580.
                        WaterHeatingText = "+" + " " + str(row[5]) + "J"
1581.
                        (x7, y7) = (150,100)
1582.
                        LightingText = "+" + " " + str(row[6]) + "J"
1583.
                        (x8, y8) = (300,100)
1584.
                        HeatByCogenerationText = "+" + " " + str(row[7]) + "J"
1585.
                        (x9, y9) = (400,100)
1586.
                        HeatByPv = "+" + " " + str(row[8]) + "J"
1587.
                        (x10, y10) = (150,100)
1588.
```

```
1589.
                        draw.text((x1, y1), internalHeatText, fill=colorBlack, font=font)
1590.
                        draw.text((x2, y2), heatGainText, fill=colorBlack, font=font)
1591.
                        draw.text((x3, y3), solarHeatText, fill=colorBlack, font=font)
1592.
                        draw.text((x4, y4), HeatingDemandText, fill=colorBlack, font=font)
1593.
                        draw.text((x5, y5), HeatLossText, fill=colorBlack, font=font)
1594.
                        draw.text((x6, y6), WaterHeatingText, fill=colorBlack, font=font)
1595.
                        draw.text((x7, y7), LightingText, fill=colorBlack, font=font)
1596.
                        draw.text((x8, y8), HeatByCogenerationText, fill=colorBlack, font=f
    ont)
1597.
                        draw.text((x9, y9), HeatByPv, fill=colorBlack, font=font)
1598.
1599.
                if len(self.energyResults)==2:
1600.
                    differences = tuple(x - y for x, y in zip(self.energyResults[-
    1], self.energyResults[-2]))
1601.
                    color1 = colorBlack
1602.
                    color2 = colorBlack
1603.
                    color3 = colorBlack
1604.
                    color4 = colorBlack
1605.
                    color5 = colorBlack
1606.
                    color6 = colorBlack
1607.
                    color7 = colorBlack
1608.
                    color8 = colorBlack
1609.
                    color9 = colorBlack
1610.
1611.
                    if differences[0] > 0:
1612.
                        color1 = colorGreen
1613.
                    elif differences[0] < 0:
1614.
                        color1 = colorRed
1615.
                    if differences[1] > 0:
1616.
                        color2 = colorGreen
1617.
                    elif differences[1] < 0:
```

```
                    color2 = colorRed
                if differences[2] > 0:
                    color3 = colorGreen
                elif differences[2] < 0:
                    color3 = colorRed
                    if differences[3] > 0:
                        color4 = colorGreen
                elif differences[3] < 0:
                    color4 = colorRed
                    if differences[4] > 0:
                        color5 = colorGreen
                elif differences[4] < 0:
                    color5 = colorRed
                    if differences[5] > 0:
                        color6 = colorGreen
                elif differences[5] < 0:
                    color6 = colorRed
                    if differences[6] > 0:
                        color7 = colorGreen
                elif differences[6] < 0:
                    color7 = colorRed
                    if differences[7] > 0:
                        color8 = colorGreen
                elif differences[7] < 0:
                    color8 = colorRed
                    if differences[8] > 0:
                        color9 = colorGreen
                elif differences[8] < 0:
                    color9 = colorRed

```

```
1648.
                    oldResults = self.energyResults[-2]
1649.
                    (x1, y1) = (320, 650)
1650.
                    internalHeatText = "+" + " " + str(oldResults[0]) + "J"
1651.
                    (x2, y2) = (680, 550)
1652.
                    heatGain = oldResults[1]
1653.
                    if heatGain >= 0:
1654.
                        mark = "+"
1655.
                    else:
1656.
                        mark = ""
1657.
                    heatGainText = mark + " " + str(oldResults[1]) + "J"
1658.
                    (x3, y3) = (680, 200)
1659.
                    solarHeatText = "+" + " " + str(oldResults[2]) + "J"
1660.
                    (x4, y4) = (50, 150)
1661.
                    HeatingDemandText = "+" + " " + str(oldResults[3]) + "J"
1662.
                    (x5, y5) = (400,150)
1663.
                    HeatLossText = "+" + " " + str(oldResults[4]) + "J"
1664.
                    (x6, y6) = (400,150)
1665.
                    WaterHeatingText = "+" + " " + str(oldResults[5]) + "J"
1666.
                    (x7, y7) = (150,150)
1667.
                    LightingText = "+" + " " + str(oldResults[6]) + "J"
1668.
                    (x8, y8) = (300,150)
1669.
                    HeatByCogenerationText = "+" + " " + str(oldResults[7]) + "J"
1670.
                    (x9, y9) = (400,150)
1671.
                    HeatByPv = "+" + " " + str(oldResults[8]) + "J"
1672.
                    (x10, y10) = (150,150)
1673.

1674.
                    draw.text((x1, y1), internalHeatText, fill=colorBlack, font=font)
1675.
                    draw.text((x2, y2), heatGainText, fill=colorBlack, font=font)
1676.
                    draw.text((x3, y3), solarHeatText, fill=colorBlack, font=font)
1677.
                    draw.text((x4, y4), HeatingDemandText, fill=colorBlack, font=font)
```

```
1678.
                draw.text((x5, y5), HeatLossText, fill=colorBlack, font=font)
1679.
                draw.text((x6, y6), WaterHeatingText, fill=colorBlack, font=font)
1680.
                draw.text((x7, y7), LightingText, fill=colorBlack, font=font)
1681.
                draw.text((x8, y8), HeatByCogenerationText, fill=colorBlack, font=font)
1682.
                draw.text((x9, y9), HeatByPv, fill=colorBlack, font=font)
1683.
1684.
                newResults = self.energyResults[-1]
1685.
                (x11, y11) = (320, 600)
1686.
                internalHeatTextNew = "+" + " " + str(newResults[0]) + "J"
1687.
                (x12, y12) = (680, 500)
1688.
                heatGainNew = newResults[1]
1689.
                if heatGainNew >= 0:
1690.
                    mark = "+"
1691.
                else:
1692.
                    mark = ""
1693.
                heatGainTextNew = mark + " " + str(newResults[1]) + "J"
1694.
                (x13, y13) = (680, 200)
1695.
                solarHeatTextNew = "+" + " " + str(newResults[2]) + "J"
1696.
                (x14, y14) = (50, 150)
1697.
                HeatingDemandTextNew = "+" + " " + str(newResults[3]) + "J"
1698.
                (x15, y15) = (400,150)
1699.
                HeatLossTextNew = "+" + " " + str(newResults[4]) + "J"
1700.
                (x16, y16) = (400,150)
1701.
                WaterHeatingTextNew = "+" + " " + str(newResults[5]) + "J"
1702.
                (x17, y17) = (150,150)
1703.
                LightingTextNew = "+" + " " + str(newResults[6]) + "J"
1704.
                (x18, y18) = (300,150)
1705.
                HeatByCogenerationTextNew = "+" + " " + str(newResults[7]) + "J"
1706.
                (x19, y19) = (400,150)
1707.
                HeatByPvNew = "+" + " " + str(newResults[8]) + "J"
```

```
                    (x20, y20) = (150,150)
                    draw.text((x11, y11), internalHeatTextNew, fill=colorBlack, font=font)
                    draw.text((x12, y12), heatGainTextNew, fill=colorBlack, font=font)
                    draw.text((x13, y13), solarHeatTextNew, fill=colorBlack, font=font)
                    draw.text((x14, y14), HeatingDemandTextNew, fill=colorBlack, font=font)
                    draw.text((x15, y15), HeatLossTextNew, fill=colorBlack, font=font)
                    draw.text((x16, y16), WaterHeatingTextNew, fill=colorBlack, font=font)
                    draw.text((x17, y17), LightingTextNew, fill=colorBlack, font=font)
                    draw.text((x18, y18), HeatByCogenerationTextNew, fill=colorBlack, font=
    font)
                    draw.text((x19, y19), HeatByPvNew, fill=colorBlack, font=font)
                else:
                    differences = tuple(x - y for x, y in zip(self.energyResults[-
    1], self.energyResults[-3]))
                    color1 = colorBlack
                    color2 = colorBlack
                    color3 = colorBlack
                    color4 = colorBlack
                    color5 = colorBlack
                    color6 = colorBlack
                    color7 = colorBlack
                    color8 = colorBlack
                    color9 = colorBlack

                    if differences[0] > 0:
                        color1 = colorGreen
                    elif differences[0] < 0:
                        color1 = colorRed
                    if differences[1] > 0:
                        color2 = colorGreen
```

```
1736.
                    elif differences[1] < 0:
1737.
                        color2 = colorRed
1738.
                    if differences[2] > 0:
1739.
                        color3 = colorGreen
1740.
                    elif differences[2] < 0:
1741.
                        color3 = colorRed
1742.
                        if differences[3] > 0:
1743.
                            color4 = colorGreen
1744.
                    elif differences[3] < 0:
1745.
                        color4 = colorRed
1746.
                        if differences[4] > 0:
1747.
                            color5 = colorGreen
1748.
                    elif differences[4] < 0:
1749.
                        color5 = colorRed
1750.
                        if differences[5] > 0:
1751.
                            color6 = colorGreen
1752.
                    elif differences[5] < 0:
1753.
                        color6 = colorRed
1754.
                        if differences[6] > 0:
1755.
                            color7 = colorGreen
1756.
                    elif differences[6] < 0:
1757.
                        color7 = colorRed
1758.
                        if differences[7] > 0:
1759.
                            color8 = colorGreen
1760.
                    elif differences[7] < 0:
1761.
                        color8 = colorRed
1762.
                        if differences[8] > 0:
1763.
                            color9 = colorGreen
1764.
                    elif differences[8] < 0:
1765.
                        color9 = colorRed
```

```
1766.
1767.                   oldResults = self.energyResults[-3]
1768.                   (x1, y1) = (320, 650)
1769.                   internalHeatText = "+" + " " + str(oldResults[0]) + "J"
1770.                   (x2, y2) = (680, 550)
1771.                   heatGain = oldResults[1]
1772.                   if heatGain >= 0:
1773.                       mark = "+"
1774.                   else:
1775.                       mark = ""
1776.                   heatGainText = mark + " " + str(oldResults[1]) + "J"
1777.                   (x3, y3) = (680, 200)
1778.                   solarHeatText = "+" + " " + str(oldResults[2]) + "J"
1779.                   (x4, y4) = (50, 150)
1780.                   HeatingDemandText = "+" + " " + str(oldResults[3]) + "J"
1781.                   (x5, y5) = (400,150)
1782.                   HeatLossText = "+" + " " + str(oldResults[4]) + "J"
1783.                   (x6, y6) = (400,150)
1784.                   WaterHeatingText = "+" + " " + str(oldResults[5]) + "J"
1785.                   (x7, y7) = (150,150)
1786.                   LightingText = "+" + " " + str(oldResults[6]) + "J"
1787.                   (x8, y8) = (300,150)
1788.                   HeatByCogenerationText = "+" + " " + str(oldResults[7]) + "J"
1789.                   (x9, y9) = (400,150)
1790.                   HeatByPv = "+" + " " + str(oldResults[8]) + "J"
1791.                   (x10, y10) = (150,150)
1792.
1793.                   draw.text((x1, y1), internalHeatText, fill=colorBlack, font=font)
1794.                   draw.text((x2, y2), heatGainText, fill=colorBlack, font=font)
1795.                   draw.text((x3, y3), solarHeatText, fill=colorBlack, font=font)
```

```
                draw.text((x4, y4), HeatingDemandText, fill=colorBlack, font=font)
                draw.text((x5, y5), HeatLossText, fill=colorBlack, font=font)
                draw.text((x6, y6), WaterHeatingText, fill=colorBlack, font=font)
                draw.text((x7, y7), LightingText, fill=colorBlack, font=font)
                draw.text((x8, y8), HeatByCogenerationText, fill=colorBlack, font=font)
                draw.text((x9, y9), HeatByPv, fill=colorBlack, font=font)

                newResults = self.energyResults[-1]
                (x11, y11) = (320, 600)
                internalHeatTextNew = "+" + " " + str(newResults[0]) + "J"
                (x12, y12) = (680, 500)
                heatGainNew = newResults[1]
                if heatGainNew >= 0:
                    mark = "+"
                else:
                    mark = ""
                heatGainTextNew = mark + " " + str(newResults[1]) + "J"
                (x13, y13) = (680, 200)
                solarHeatTextNew = "+" + " " + str(newResults[2]) + "J"
                (x14, y14) = (50, 100)
                HeatingDemandTextNew = "+" + " " + str(newResults[3]) + "J"
                (x15, y15) = (400,100)
                HeatLossTextNew = "+" + " " + str(newResults[4]) + "J"
                (x16, y16) = (400,150)
                WaterHeatingTextNew = "+" + " " + str(newResults[5]) + "J"
                (x17, y17) = (150,150)
                LightingTextNew = "+" + " " + str(newResults[6]) + "J"
                (x18, y18) = (300,150)
                HeatByCogenerationTextNew = "+" + " " + str(newResults[7]) + "J"
                (x19, y19) = (400,150)
```

```
                    HeatByPvNew = "+" + " " + str(newResults[8]) + "J"
                    (x20, y20) = (150,150)
                    draw.text((x11, y11), internalHeatTextNew, fill=colorBlack, font=font)
                    draw.text((x12, y12), heatGainTextNew, fill=colorBlack, font=font)
                    draw.text((x13, y13), solarHeatTextNew, fill=colorBlack, font=font)
                    draw.text((x14, y14), HeatingDemandTextNew, fill=colorBlack, font=font)
                    draw.text((x15, y15), HeatLossTextNew, fill=colorBlack, font=font)
                    draw.text((x16, y16), WaterHeatingTextNew, fill=colorBlack, font=font)
                    draw.text((x17, y17), LightingTextNew, fill=colorBlack, font=font)
                    draw.text((x18, y18), HeatByCogenerationTextNew, fill=colorBlack, font=
font)
                    draw.text((x19, y19), HeatByPvNew, fill=colorBlack, font=font)

                imageFirstConvertToQ = ImageQt(image)
                imageSecondConvertToQ = QImage(imageFirstConvertToQ)
                self.pixmap = QtGui.QPixmap(imageSecondConvertToQ)
                self.pixmap = self.pixmap.scaled(300, 300, QtCore.Qt.KeepAspectRatio)
                self.label.setPixmap(self.pixmap)
                self.label.show()

                # Creating feedbacks

                def getFeedback(self):
                    totalWallArea = 0
                    totalWindowArea = 0
                    for wall in self.wallList:
                        totalWallArea = totalWallArea + wall.area
                    for opening in self.openingList:
                        totalWindowArea = totalWindowArea + opening.area
```

```
1854.
1855.                       if self.HeatLossThroughTransmission == 0:
1856.                               WallConsumption = 0
1857.                   else:
1858.                       WallConsumption = self.heatLosses / wall.area
1859.
1860.
1861.                   if (self.northOpeningSolarRadiationCalculation + self.southOpeningSolar
    RadiationCalculation + self.eastOpeningSolarRadiationCalculation + self.westOpeningSola
    rRadiationCalculation) == 0:
1862.                       OpeningConsumption = 0
1863.                   else:
1864.                       openingConsumption = self.OpeningSolarRadiation / CollectorArea + h
    eatGains
1865.
1866.                   if self.northOpeningSolarRadiationCalculation == 0:
1867.                       northConsumptionOpening = 0
1868.                   else:
1869.                       northConsumptionOpening = self.northOpeningSolarRadiationCalculatio
    n / CollectorArea + heatGains
1870.
1871.                   if self.westOpeningSolarRadiationCalculation == 0:
1872.                       westConsumptionOpening = 0
1873.                   else:
1874.                       westConsumptionOpening = self.westOpeningSolarRadiationCalculation
    / CollectorArea + heatGains
1875.
1876.
1877.                   if self.southOpeningSolarRadiationCalculation == 0:
1878.                      southConsumptionOpening = 0
1879.                   else:
1880.                       southConsumptionOpening = self.southOpeningSolarRadiationCalculatio
    n / CollectorArea + heatGains
```

```
1881.
1882.           results = self.energyResults[-1]
1883.           self.feedbackBox.clear()
1884.           self.feedbackBox.append ("The total energy consumption of the building is: " +
    str(results[3]) + "J")
1885.           self.feedbackBox.append("Each element contributes differently to this result")
1886.           self.feedbackBox.append("")
1887.           self.feedbackBox.append("i. walls")
1888.           self.feedbackBox.append("Walls: " + str(round(WallConsumption,2)) + " J/m^2")
1889.           self.feedbackBox.append("")
1890.           self.feedbackBox.append("ii. openings")
1891.           self.feedbackBox.append("-
    north: " + str(round(northConsumptionOpening,2)) + " J/m^2")
1892.           self.feedbackBox.append("-
    east: " + str(round(eastConsumptionOpening, 2)) + " J/m^2")
1893.           self.feedbackBox.append("-
    south: " + str(round(southConsumptionOpening, 2)) + " J/m^2")
1894.           self.feedbackBox.append("-
    west: " + str(round(westConsumptionOpening, 2)) + " J/m^2")
1895.
1896.           def viewEnergyResults (self):
1897.               if not self.ResultsDialog.isVisible:
1898.                   self.ResultsDialog.show()
1899.                   self.getVisualGraphAsBarChart(self.heatGains, self.heatLosses)
1900.                   self.getTableResults()
1901.           def getTableResults(self):
1902.               results = ["Heat Transmission", "Heat Ventilation","Internal Heat Gain","So
    lar Radiation","Water Heating","Lighting","Heat By Cogeneration","Heat By Pv"]
1903.               horHeaders = []
1904.               verHeaders = []
1905.               rowList =[]
1906.               heatTransmission = []
1907.               for row in range(0, len(self.wallCalculation)):
```

```
1908.                     heatTransmission.append((self.wallCalculationPerMonth[row]))
1909.             self.table.setRowCount(0)
1910.             self.table.setColumnCount(0)
1911.             rowPosition = self.table.rowCount()
1912.             if rowPosition <= 3:
1913.                 self.table.insertRow(rowPosition)
1914.                 verHeaders.append(result)
1915.             for row in heatTransmission:
1916.                 rowList.append(row)
1917.             for row in self.ventilationCalculation:
1918.                 rowList.append(row)
1919.             for row in self.internalHeat:
1920.                 rowList.append(row)
1921.             for row in self.OpeningSolarRadiation:
1922.                 rowList.append(row)
1923.             for row in self.HeatGainByHotWaterCalculation:
1924.                 rowList.append(row)
1925.             for row in self.HeatGainByCogeneration:
1926.                 rowList.append(row)
1927.             for row in self.HeatGainByPV:
1928.                 rowList.append(row)
1929.             rowList = []
1930.             print (rowList)
1931.             self.table.setHorizontalHeaderLabels(horHeaders)
1932.             self.table.setVerticalHeaderLabels(verHeaders)
1933.             self.table.resizeColumnsToContents()
1934.             self.table.resizeRowsToContents()
1935.             self.table.setFixedHeight(140)
1936.             self.table.show()
1937.         init = initUI()
```