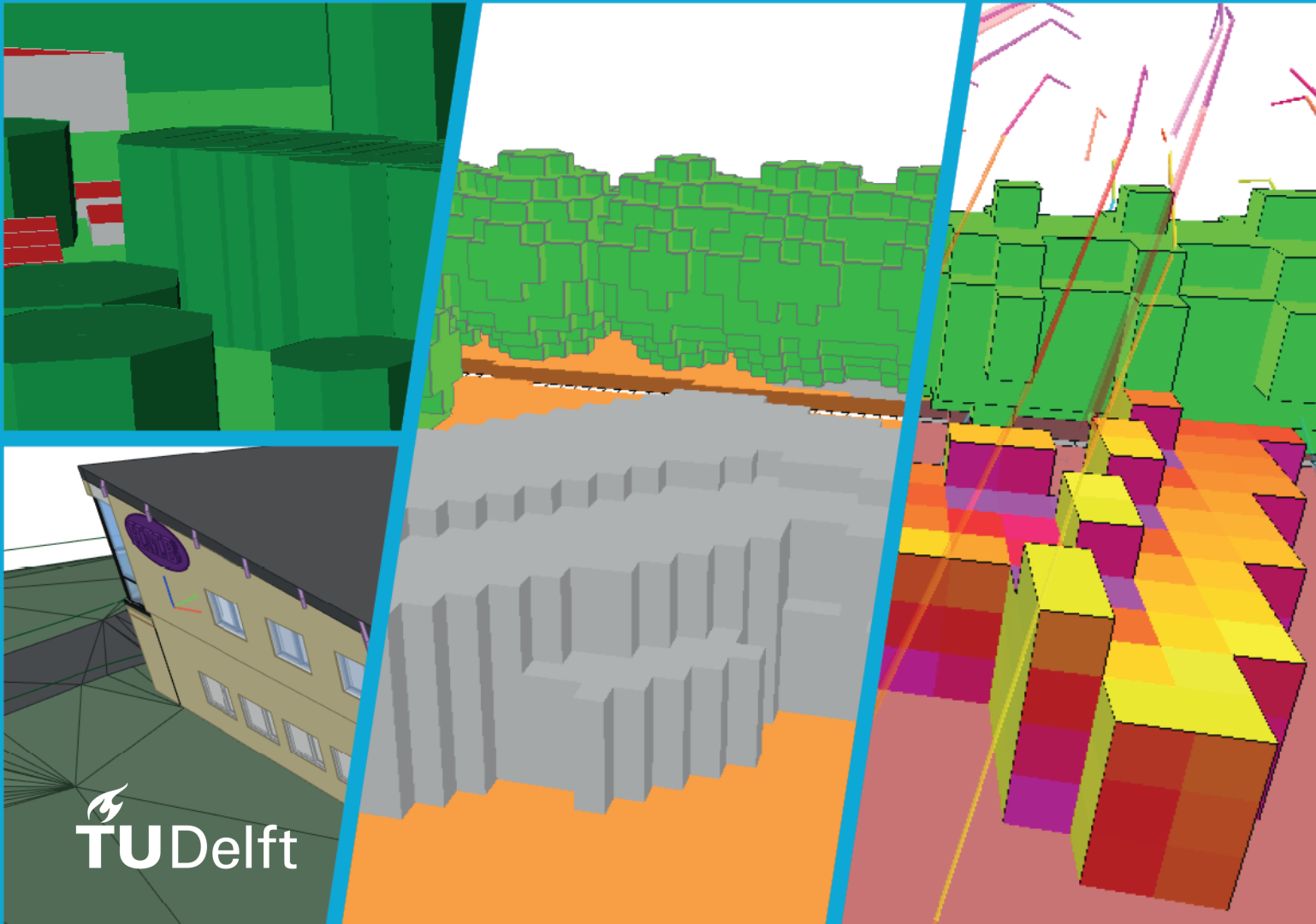


MSc thesis in Geomatics

BIM and 3D City Models as Input for Microclimate Simulation

Natasja van Heerden
2021



BIM AND 3D CITY MODELS AS INPUT FOR MICROCLIMATE SIMULATION

A thesis submitted to the Delft University of Technology in partial fulfillment
of the requirements for the degree of

Master of Science in Geomatics

by

Natasja van Heerden

July 2021

Natasja van Heerden: *BIM and 3D City Models as Input for Microclimate Simulation*
(2021)

© This work is licensed under a Creative Commons Attribution 4.0 International License. To view a copy of this license, visit
<http://creativecommons.org/licenses/by/4.0/>.

The work in this thesis was made in the:



3D geoinformation group
Department of Urbanism
Faculty of Architecture & the Built Environment
Delft University of Technology

Supervisors: Francesca Noardo
Marjolein Pijpers-van Esch
Co-reader: Hugo Ledoux

ABSTRACT

Climate change, and the effect it has on our lives and the world, has been at the centre of debate in the past decades. Also in architecture and urban planning fields, the effect of climate on buildings and cities has always had a role in the design process, nowadays especially when it comes to sustainability related topics. An interesting element within climate is the microclimate, which is a small area where climate circumstances are different then in the surrounding area. For example in an urban area the temperature difference between an area in a big city and the rural area around it can be up to 10°C, due to influences of surrounding buildings and materials. In the design process often more general weather data is used in calculation, this while they can differ a lot from reality. Modern tools make it possible to simulate the microclimate. One of these tools is ENVI-met. However, inputting 3D models into this software is mostly a manual process. This while a lot of information that is necessary is already available in existing architectural and urban design models. There is just no way to input this information into the software.

In this research a method is developed for combining detailed information from BIM models (IFC), with information about the surroundings from 3D City models (CityGML), and translating them to the ENVI-met format, so that these models can be used as input models for microclimate simulation in ENVI-met. This is done by creating a command line tool that extracts the necessary data from both input files, combining and converting it, and then writing it to a file in the ENVI-met format. Also guidelines and requirements for the input files will be established.

This is done by first establishing what information is necessary for microclimate simulation in ENVI-met and how this information needs to be represented, and then finding out where this information can be found in the intended input files, and how it is represented in there. From this can be concluded what information can be taken from which input file and the characteristics that are necessary for their correct use in the process can be established. Then the conversion tool itself can be developed, where the data is transformed to the same coordinate system and format, so that it can be combined and written to the ENVI-met format. In the last step the results are checked by doing a small case study and running the microclimate simulation.

This way, IFC and CityGML models can be used as input for microclimate simulation software ENVI-met, by using the conversion tool developed for this research and the provided guidelines.

ACKNOWLEDGEMENTS

I would like to thank everyone who has been involved in the research, and everyone who supported me during this time.

First of all I would like to express my sincere appreciation to my first supervisor, Francesca Noardo, who has guided and supported me from the beginning to the end of the project. She provided great advice, feedback and guidance, but also left room for me to explore and find my own path and solutions within the research, which is exactly what I needed. Thanks to her support, encouragements and patience throughout the project, especially when things got tough, I was able to finish it successfully. I would also like to thank my second supervisor, Marjolein Pijpers-van Esch, who has been there since the second half of my research, providing insight in ENVI-met and research in general. With her feedback and advice, and especially her positive attitude and support giving me a confidence boost after every meeting, she really helped me in the final stages of my thesis project. I would also wish to show my gratitude to Daniela Maiullari, who has been my second supervisor during the first half of my thesis project. With her knowledge and feedback she helped to set me on the path that ended in a successful research.

I would also like to thank the Geomatics staff and 3D geoinformation research group. I would like to recognize the invaluable assistance that you all provided during my study. I would especially like to thank Hugo Ledoux, who provided insightful feedback on my thesis as co-reader. Also as coordinator of the graduation project, he provided a clear overview of what was expected from me during the whole graduation process, which really makes a difference for me and has been very helpful.

Last but not least, I would like to thank my family and friends for their love and support. Especially my boyfriend Jarno Moree, who has been there for me throughout my whole study and thesis project. It was nice to always have someone that I could discuss encountered challenges with, and could help me see the bigger picture when I was stuck in the details. And since he has more experience with programming, he was always ready to answer my programming related questions, which gave me more confidence during the development of the conversion tool that was part of my thesis project.

CONTENTS

1	INTRODUCTION	1
1.1	Aim of the research	2
1.2	Thesis outline	4
2	THEORETICAL BACKGROUND AND RELATED WORK	5
2.1	Microclimate Simulations	5
2.1.1	Urban Climate	5
2.1.2	ENVI-met	6
2.2	3D City Model	8
2.2.1	CityGML Format and Information	8
2.2.2	CityGML for Microclimate Simulation	9
2.3	BIM	9
2.3.1	IFC Format and Information	9
2.3.2	IFC for Microclimate Simulation	10
2.4	GeoBIM	10
2.5	Importing existing 3D models in microclimate simulation tool ENVI-met	11
3	METHODOLOGY	13
3.1	Research Approach	13
3.1.1	Data inspection	13
3.1.2	Mapping	15
3.1.3	Data Conversion	15
3.1.4	Testing & Analysing	16
3.2	Test Data & Data Preparation	16
3.3	Conversion Approach	18
4	DATA REQUIREMENTS, CHARACTERISTICS AND IDENTIFIERS	21
4.1	Input files for microclimate simulation in ENVI-met	21
4.1.1	Input files relationship model	22
4.1.2	Area input file	24
4.1.3	Database system	27
4.1.4	Simulation file	28
4.2	Required information in IFC and CityGML	29
4.2.1	Representation in schemas and existing data	29
4.2.2	Format in IFC	30
4.2.3	Format in CityGML	32
4.3	Comparison	33
4.3.1	Differences	33
4.3.2	What information from which 3D model	34
4.4	Characteristics of the data for identification	34
4.4.1	Model Geometry	35
4.4.2	Location	35
4.4.3	Buildings	36
4.4.4	Vegetation	37
4.4.5	Terrain	38
5	CONVERSION OF IFC AND CITYGML MODEL DATA TO MICROCLIMATE SIMULATION SOFTWARE ENVI-MET	39
5.1	Modelling of the data in the conversion tool	40
5.1.1	Model design	40
5.1.2	Transformation	41
5.2	Model C: the data structure where all necessary information from data is stored	44
5.2.1	Model geometry and structure	44

5.2.2	Element Classes	44
5.2.3	From data structure C to ENVI-met model	45
5.3	Conversion of necessary data	45
5.3.1	Parsers	46
5.3.2	Conversions	46
6	RESULTS; PRODUCTS AND CASE STUDY	49
6.1	Guidelines and requirements for generating input data	49
6.1.1	Guidelines for IFC model creation	49
6.1.2	CityGML requirements	50
6.2	Workflow of the conversion tool	50
6.2.1	Setup	50
6.2.2	User input	51
6.2.3	Conversion tool	51
6.2.4	Verification and editing	52
6.3	Test study	52
6.3.1	Input models	52
6.3.2	Conversion results	53
6.3.3	Microclimate simulation results	55
7	CONCLUSIONS, DISCUSSION AND FUTURE WORK	57
A	DATA SHEET ENVI-MET AREA INPUT FILE	67
B	DATA SHEET ENVI-MET DATABASE FILE	73

LIST OF FIGURES

Figure 1.1	Basic concept of how detailed design information could be combined with information about the existing surrounding, for microclimate simulation	3
Figure 1.2	Research framework	3
Figure 2.1	Schematic showing the Urban Boundary Layer (UBL) its different layers on different scales, with amongst others the Urban Canopy Layer (UCL). (revised by Oke and Rotach after a figure in Oke, 1997).	6
Figure 2.2	Output of ENVI-met simulation in Leonardo for windspeed. .	7
Figure 3.1	The main steps in this research.	13
Figure 3.2	The steps in the data inspection phase	14
Figure 3.3	The steps in the mapping phase	15
Figure 3.4	The steps in the conversion phase	16
Figure 3.5	The steps in the testing and analysing phase	17
Figure 3.6	IFC models studied during this research	17
Figure 3.7	CityGML models studied during this research	17
Figure 3.8	Conversion approach where information is extracted from both input files, converted, combined, and inputted directly into the ENVI-met Area Input File format. This is the approach used in this research	19
Figure 3.9	Conversion approach where IFC is converted to CityGML and combined and then CityGML converted to the ENVI-met Area Input File format. This approach was chosen to not be used in this research	19
Figure 4.1	Input files for microclimate simulation in ENVI-met and how they are connected	22
Figure 4.2	Relationship between the ENVI-met Area Input File elements and databases	23
Figure 4.3	Illustrations clarifying the ENVI-met Area Input File elements and their attributes	27
Figure 4.4	Schema describing from which input file the required elements will be extracted	34
Figure 4.5	Diagram showing which attributes in IFC and CityGML can be used to determine the model geometry attributes for the ENVI-met Area Input File	35
Figure 4.6	Diagram showing which attributes in IFC and CityGML can be used to determine the location data attributes for the ENVI-met Area Input File	36
Figure 4.7	Diagram showing which attributes in IFC and CityGML can be used to determine the building and wall attributes for the ENVI-met Area Input File	37
Figure 4.8	Diagram showing which attributes in IFC and CityGML can be used to determine the 3D plants and simple plants attributes for the ENVI-met Area Input File	37
Figure 4.9	Diagram showing which attributes in IFC and CityGML can be used to determine the terrain 2D and 3D attributes for the ENVI-met Area Input File	38
Figure 5.1	transformation between the models	39
Figure 5.2	a schema how the different modules in the conversion tool co-operate	39

Figure 5.3	terminology in the model design	40
Figure 5.4	model space dimensions	41
Figure 5.5	model rotation in the different types of model representation	41
Figure 5.6	Explanation of reference point and (custom) projection systems	42
Figure 5.7	The transformation between two planar systems is dependent of three elements: translation, rotation and scale	43
Figure 6.1	user input - program call with arguments	51
Figure 6.2	feedback that the program gives while running, about what it is doing	51
Figure 6.3	Masterplan of Floriade site	52
Figure 6.4	IFC and CityGML input models for conversion tool	53
Figure 6.5	Resulting model in ENVI-met SPACES after combining and converting with the conversion tool on 0.5 meter resolution, displayed as a 2D map in ENVI-met SPACES	53
Figure 6.6	Details of resulting ENVI-met area input file displayed in ENVI-met SPACES	54
Figure 6.7	The IFC model after conversion in three different resolutions to the ENVI-met area input file displayed in ENVI-met SPACES	54
Figure 6.8	Details of resulting ENVI-met area input file displayed in ENVI-met SPACES	55
Figure 6.9	Microclimate simulation results for air temperature and wind visualised in ENVI-met LEONARDO in 2D. The colours display the potential air temperature at that location. The arrows display wind direction and speed. The longer the arrow, the higher the wind speed.	55
Figure 6.10	Microclimate simulation results for facade temperature and wind visualised in ENVI-met LEONARDO in 3D. The colours on the buildings represent the facade temperature. The tubes represent wind paths and their colour the wind speed.	56

LIST OF TABLES

Table 2.1	input and output files for ENVI-met simulation, with file format and description	7
Table 2.2	Required elements for ENVI-met simulation and the corresponding description of the element in ENVI-met data structure.	8
Table 4.1	Required input files for microclimate simulation in ENVI-met and a short description of them	22
Table 4.2	The three different databases in ENVI-met and a description of them	28
Table 4.3	Representation of required elements for microclimate simulation in ENVI-met in both official schemas and existing data	31

ACRONYMS

DEM Digital Elevation Model	50
PET Physiological Equivalent Temperature	1
GIS Geographic Information System	8
BIM Building Information Modelling	2
EML ENVI-met Markup Language	21
IFC Industry Foundation Classes	2
OGC Open Geospatial Consortium	8
GML Geography Markup Language	8
LOD Level of Detail	9
ADE Application Domain Extensions	9
TIN Triangulated Irregular Network	33
SRS Spatial Reference System	50
UBL Urban Boundary Layer	xi
UCL Urban Canopy Layer	xi
RSI The Roughness Sublayer	5
ISL Inertial Sublayer	5
ML Mixed Layer	5
MRT Mean Radiant Temperature	6
UTCI Universal Thermal Climate Index	6

The present and future impacts of global warming on the urban environment have been at the centre of debate in architecture and urban planning fields, during the last decades. More often than not, sustainability plays an important part in the design of an urban area or building. Also outdoor thermal comfort becomes more and more important. One of the major concerns in sustainability is the increase in ambient temperatures, and the related building energy demand, especially for cooling purposes. But this raise in temperature also influences how an urban area is perceived by the user, so that a designer may come to different designs when outdoor thermal comfort is considered. In order to address these, and other challenges, when developing liveable and sustainable cities, new tools and methods are necessary.

An example is energy demand calculations, where it is calculated how much energy is going to be used for heating and cooling of a building, based on the outside temperature. However, for these calculations, often general weather data from weather stations outside the city is used. But different elements in a city, like geometry, materials and vegetation, can influence climate factors like humidity, wind and also temperature. [Robitu et al., 2006]

Due to these influences, the temperature difference between an area in a city and the rural area around it can be up to 10°C [Santamouris et al., 2001]. This is called the urban heat island effect. Using general weather data, from weather stations outside the city, for calculations like the aforementioned, is therefore insufficient and gives unrealistic results. A simulation of the urban environment is necessary, to be able to map the climate inside a city and generate more realistic data for calculations. These so called microclimate simulations, can approach the climate on the micro scale, based on surrounding geometry, materials, and other urban influences and can therefore help generating more realistic climate data.

Simulation of the ambient temperature on microscale for energy demand calculations is only one of the many applications of microclimate simulations. The use of microclimate simulation can be beneficial for many different use cases concerning urban- and building design. For example in regards to urban comfort.

For outdoor thermal comfort, not only the ambient temperature plays a part, but also other aspects like the radiant temperature of surrounding surfaces, air movement and relative humidity have an effect on the human body and thus affect comfort. The air temperature is therefore not always a sufficient measure to describe how an environment can be perceived. In the Physiological Equivalent Temperature (PET) [Höppe, 1999], above mentioned factors are taken into account, and is a better representation of how comfortable the temperature of an area is actually perceived by a person. Water, vegetation and materials all affect the PET value. Water and vegetation can for example influence the humidity, which in turn affects the PET value. Also different materials have different thermal properties and can store and reflect heat, which can affect the radiant temperature. [Chatzidimitriou and Yannas, 2016]

Geometry of the buildings can influence the air flow and speed. A breeze can have a positive effect in an otherwise hot environment or climate, but on the other hand, placing the entrance of a building at a location where wind turbulence exists, because of a high surrounding building, could make for a bad design decision.

All these design decisions about geometry, facades, vegetation, street materials, and many more, affect the microclimate and with that outdoor thermal comfort, air temperature and even things like air quality [Chatzidimitriou and Yannas, 2016; Zhang and Gu, 2013]. The need for insight in the impact of these design decisions on the microclimate, makes microclimate simulation such a powerful tool in the process of designing sustainable and liveable cities, especially now the climate is changing and we have to adjust our cities accordingly.

One of the software packages often used for microclimate simulations is ENVI-met¹. In order to perform microclimate simulations in software like ENVI-met, 3D information about the buildings, materials and other elements, is essential. At this moment in time, collecting the required spatial data, and entering it in the software, is generally a manual process. This process is human-resource expensive and increases the probability of inaccuracy. The goal is to help designers, by making this process less cumbersome, and therefore microclimate simulations a more attractive design tool. By automating the creation of a 3D model containing all the necessary information for microclimate simulation, doing correct calculations for energy demand, outdoor thermal comfort and other analyses, will be made easier.

A lot of the 3D information, necessary for microclimate simulations, can be found in the existing 3D models, created by designers and urban analysts and -planners.

In the GIS domain, a 3D City Model is used for exchanging 3D information on city scale. CityGML is the open standard format of a 3D City Model. 3D City Models are on city level and the scale makes them very useful for microclimate simulations, since they contain a lot of information about the surrounding environment of a location [Arroyo Ohori et al., 2018a]. However, a 3D City Model often does not contain all the detailed information needed for microclimate simulations, such as detailed information about building materials.

On the other hand, a lot of this missing information can be found in the detailed models created by designers. Building Information Modelling (BIM) is widely used in the design world for designing on construction level, and Industry Foundation Classes (IFC) is the open standard. IFC models are very detailed, but because they are developed for other aims than automatic conversion, geometrical and topological issues can be found. They are very computation heavy because of the for microclimate simulations unnecessary information [Arroyo Ohori et al., 2018b]. Also the georeferencing component is often lacking, since many models make use of a local Cartesian reference system and a reference point is often missing or incorrect [Arroyo Ohori et al., 2017].

With the information from IFC models and CityGML combined, there is a lot of data about a building or design, and its environment, which can be used for microclimate simulations, as can be seen in figure 1.1. But at of this moment, there is no easy way to extract all this information from both models and input it automatically in the microclimate simulation software ENVI-met.

1.1 AIM OF THE RESEARCH

In this research, a method will be developed, that allows designers to be able to combine their building design in BIM (IFC) with a 3D City Model (CityGML), to run reliable microclimate simulations supporting the assessment of the energy performances of the new building and the effect on the urban climate. For this, a tool that generates a 3D model ready for microclimate simulations in ENVI-met based on the IFC and CityGML models, will be developed.

¹ <https://www.envi-met.com/>

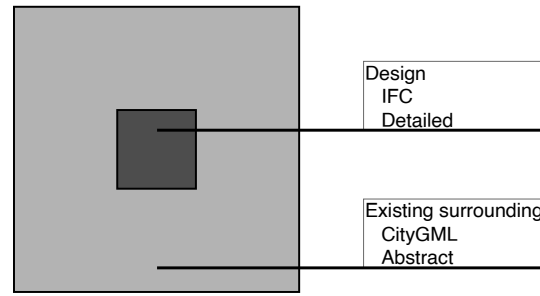


Figure 1.1: Basic concept of how detailed design information could be combined with information about the existing surrounding, for microclimate simulation

These three different models are very different in nature, since they are used in different domains, with different goals. One significant difference is the representation of the geometry. In CityGML, geometry is often a boundary representation for an object (like a building), while in IFC elements like walls and floors are represented separately commonly as solids based on extrusion. In the 3D model generated for ENVI-met, the geometry is entirely grid-based, which means elements are represented in a grid with a low resolution (between 0.5 and 10 meter).

To be able to combine information from these different models, into ENVI-met, not only a conversion between these different geometry representations is necessary, but also a method to combine these models in the right location. It is necessary to know what the exact location of the models is, in order to correctly combine them, but the different models use different coordinate systems. In the geomatics field, georeferencing is used to connect a model to real-world location. Georeferencing information from the models, can be used to know how the models lie relative to each other, so that correct transformations between the models can be applied.

Guidelines for the designer of the BIM model will be developed, including a subset of IFC entities that are expected to be used. This will help guide the designer in the process and make sure the IFC model meets the requirements for the conversion.

Figure 1.2 shows the framework of this research. Within this research an approach is developed for making microclimate simulation more accessible, so that it is easier to integrate in the design process, making it possible to do correct studies on energy and sustainability related topics, serving the wider social context.

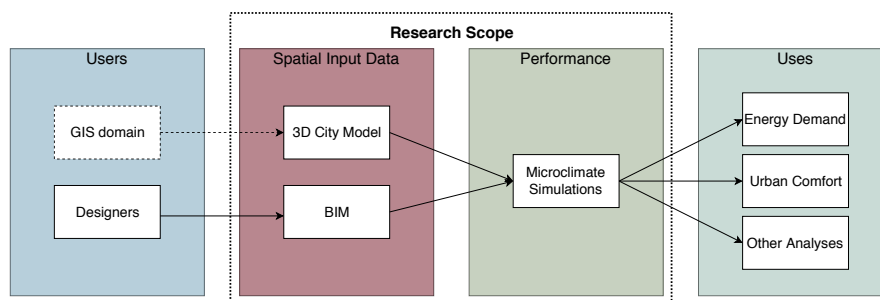


Figure 1.2: Research framework

In order to accomplish this, the following research question is formulated:

How can IFC and CityGML models be used as input for microclimate simulation software ENVI-met.

To answer this question and achieve the objectives of this research, the following sub questions will be answered:

- What data is needed for microclimate simulation? Where can this information be found in IFC and CityGML schemas and data from practice?
- What characteristics should the data have, in order to allow their suitable use in the process?
- How to convert and combine IFC and CityGML information effectively into the ENVI-met format?

1.2 THESIS OUTLINE

In this thesis, the process of the research will be discussed.

- In chapter 2, some theoretical background on the subject will be given. Also research on topics related to this research will be explained and discussed.
- In chapter 3, the methods that were used to find the answer to the research questions and why these methods were chosen will be explained
- In chapter 4, the data requirement for microclimate simulation in ENVI-met will be discussed. The input files of ENVI-met will be explained, and it will be discussed which of the necessary information for the input files can generally be found in CityGML and IFC models. The chapter will finish with an examination of what characteristics the CityGML and IFC models should have, for their suitable use in the process.
- In chapter 5 will be discussed how IFC and CityGML can be converted and combined effectively into the ENVI-met format. In this chapter the challenge of combining data with different reference systems will be tackled, in order to be able to actually combine and compare information at a location. In this chapter will also be discussed how the data can be extracted from the models and be converted and stored to a voxel based system that can be used in ENVI-met.
- Chapter 6 will explain the case study that is done, to demonstrate the resulting method, tool and workflow. Results will be shown and also the resulting tool and guidelines will be shortly discussed.
- Chapter 7 will conclude the research, the work and method will be discussed and possible future work will be suggested.

2

THEORETICAL BACKGROUND AND RELATED WORK

In order to generate a model containing all information needed for microclimate simulation, a method is going to be developed. In this method the needed information from existing BIM and 3D City Models is extracted and combined in the right format. But how can this be achieved? First it is important to understand the different aspects of the topics related to the problem. These topics are related to microclimate simulations, detailed 3D models on building level in BIM, and 3D City Models providing context on city level. In this chapter these topics will be discussed. Also existing research related to this research will be discussed.

2.1 MICROCLIMATE SIMULATIONS

To investigate the effect of certain design choices on the climate of the environment, the effects of building geometry, materials and vegetation, have to be simulated. There are different computational models and tools, able to perform microclimate simulations. In this research microclimate simulation software ENVI-met will be used, since it is one of the most complete tools. A further elaboration will be given in section 2.1.2. In 2.1.1 some aspects of the urban climate are explained.

2.1.1 Urban Climate

Weather stations are often situated in a rural area. However, values of air temperatures, humidity, and wind velocity among others are strongly influenced by urban settlements. Geometry and materials contribute to change climate conditions like airflow, humidity and shadow [Robitu et al., 2006]. These influence the climate from the microscale to the mesoscale. To be able to understand and look closer at the microclimate, a model is needed.

According to Oke et al. [2017], the urban environment can be seen as shown in figure 2.1. The Urban Boundary Layer (UBL) shows where the urban environment effects the atmosphere. On the local- and micro scale, the Urban Canopy Layer (UCL) can be seen. This is the space between the buildings, below rooflevel. The Roughness Sublayer (RSL) is where the flow is affected by the individual elements, and therefore needs to be modelled in 3D. It reaches from the ground till up to two to five times the building height and includes the UCL. The layers above, the Inertial Sublayer (ISL) and Mixed Layer (ML), barely vary in the horizontal direction, and temperature, humidity and wind speed and direction, change mostly in the vertical direction [Oke et al., 2017].

To describe the urban microclimate, different aspects are important to consider. A large number of studies investigates the magnitude of Urban Heat Island effect in cities [Bornstein, 1968; Debbage and Shepherd, 2015; Yang et al., 2016]. This phenomenon has been identified in urban environments, where air temperatures are significantly higher locally, compared to its surroundings. But air temperature is not the only factor when it comes to describing the urban climate, and especially when it comes to urban comfort, different factors have to be taken into account.

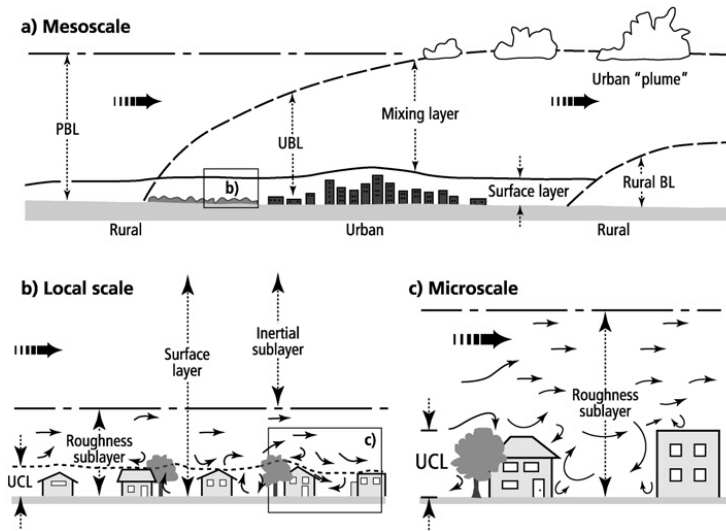


Figure 2.1: Schematic showing the UBL its different layers on different scales, with amongst others the UCL. (revised by Oke and Rotach after a figure in Oke, 1997).

With the 'wind chill equivalent temperature' the effect that wind speed has on heat loss of the human body is taken into consideration. Often used units and aspects for thermal comfort in urban design are the Mean Radiant Temperature (MRT), PET or Universal Thermal Climate Index (UTCI). The MRT considers the exchange of thermal radiation of human bodies, with their surroundings and is used as parameter for PET and UTCI [Kántor and Unger, 2011; Di Napoli et al., 2020]. These aspects are part of the microclimate, and can be considered in microclimate simulation.

2.1.2 ENVI-met

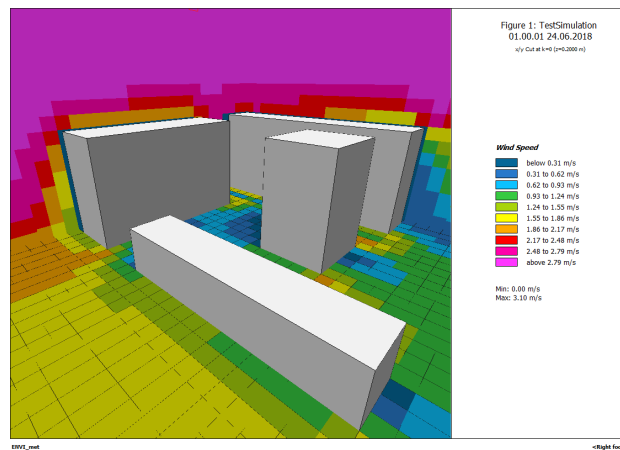
There are many different tools that simulate aspects of the microclimate. For the purpose of this research, ENVI-met is used as microclimate simulation tool. It is found to be a very complete tool that takes many different aspects of the climate into consideration. It includes computational fluid dynamics in the calculations, which is an important part of the urban climate. The completeness and adaptiveness of the tool make it very suitable for research.

The ENVI-met software suite includes different applications. Among others, a modelling application, a simulation application and an analysing application. The following features are included in the calculation of the ENVI-met simulation model: airflow, moisture and heat exchange processes between surfaces plants and air, building physics, vegetation, bioclimatology and pollution. ENVI-met can be used to simulate data about different topics, like temperatures, wind flow and turbulence, and radiation fluxes [ENVI-met GmbH, 2019].

The ENVI-met simulation tool needs a number of input files to run. An area input file, a simulation file and a database. The input and output files, with their format and description, are presented in table 2.1 [Salata et al., 2016; ENVI-met GmbH, 2019]. The results can be analysed in 2D, 3D or graphs, as can be seen for wind speed in 3D in figure 2.2.

Table 2.1: input and output files for ENVI-met simulation, with file format and description

File	Format	Description
Area input file	.inx	contains information about buildings, vegetation, soils, pollution, spatial resolution and model rotation and location
Simulation file	.simx	contains initial calculation parameters and boundary conditions, like meteorology information, simulation date and duration and computing options
Database	.edb	contains databases for Profiles, Soils, Materials, Walls, Single Walls, Plants, Sources, Greening and Plant3D. There is a System database, a User database and Project database
Output files	.edx, .edt	binary files that can be interpreted with the supplied analysing tool Leonardo

**Figure 2.2:** Output of ENVI-met simulation in Leonardo for windspeed.

The model space in ENVI-met is divided in an orthogonal 3D grid. Both the input model, as the simulation results will be based on this grid. The resolution of the grid is typically between 0.5 and 10 meters [Huttner, 2012]. The input geometry follows this grid, and is represented via 3D matrix. A position in this matrix is described via its index (i,j,k) , where i , j and k correspond to the index in the x , y and z direction of the model. A point in this matrix will also often be referred to as a voxel, in this research. The simulation will also be based on these voxels, with the same resolution.

To be able to properly simulate the urban environment, the model space should be at least twice as high as the highest building, because this is the layer where flow is affected by individual elements. Also the geometry should not be on the edge of the model space, because for wind simulation more space around it is needed to properly model this [ENVI-met GmbH, 2019].

A list of the required spatial elements for microclimate simulation in ENVI-met and how this data is described in the area input file, is presented in table 2.2. More detail about the exact elements and attributes of the ENVI-met input files will be explained in chapter 4.

Table 2.2: Required elements for ENVI-met simulation and the corresponding description of the element in ENVI-met data structure.

Required elements	Description
Buildings:	3D element
Geometry	Both in 2.5D (2D matrix with building height in grid) and in 3D (list of positions in 3D matrix that contain building)
Walls/roof	Either a single layer wall (Single Walls), a wall containing 3 layers (Walls) or a green facade or roof (Greening)
Material	List of positions in 3D matrix that contain wall, reference to a type of Wall in database. These have a reference to Materials.
Trees	3D element, with root location and reference to object in Plant3D database
Other vegetation	2,5D element in 2D matrix, with reference to object (with height) in Plants database
Soil	2D element in 2D matrix, with reference to Profiles in database (consisting of layers of Soils in database)
Infrastructure	Described in Soil
Water	Described in Soil
Location	LocationData (i.a. longitude and latitude)
Terrain height	Both as 2.5D element (2D matrix with terrain height) and 3D (list of positions in 3D matrix that contain terrain)

2.2 3D CITY MODEL

To be able to perform urban microclimate simulations, a model of the urban environment is necessary. 3D city models are on city level and useful for urban analysing purposes. They are mostly used in the Geographic Information System (GIS) domain and a commonly used open standard is CityGML. This section will give some theoretical background on CityGML and its format, as well as current research and how CityGML is currently used for sustainability and microclimate simulation purposes.

2.2.1 CityGML Format and Information

CityGML is an open standard by the Open Geospatial Consortium (OGC), for exchanging 3D City information in the GIS domain. For representation it uses Geography Markup Language (GML), which is an XML-based format [Gröger et al., 2012]. Besides 3D geometry, CityGML can also hold information about semantics, topology and appearance [Gröger and Plümer, 2012].

Generally, geometry like buildings are represented as watertight multisurfaces. These surfaces cannot overlap or intersect. In specific cases, the use of multisurface or surface or line geometries is also possible [Gröger and Plümer, 2012].

Each point in the representation is an absolute coordinate. When there is many copies of an object in a model, like for example trees, it is possible to represent it via translation instead of absolute coordinates [Gröger and Plümer, 2012].

CityGML consists of the following modules: Appearance, Bridge, Building, City-Furniture, CityObjectGroup, Generics, LandUse, Relief, Transportation, Tunnel, Vegetation and WaterBody.

In CityGML the model can be described in different Level of Detail (LOD) for suitable use in different applications. Multiple LODs can be stored in one model, so that for example for visualisation and analyses a different LOD can be used. The higher the LOD, the higher the accuracy and less generalised. In the enumeration below the five LODs and a brief description are given.

- **LOD₀** is a 2.5D digital terrain model where buildings are represented only as a footprint or roof surface.
- **LOD₁** represents buildings as block volumes.
- **LOD₂** adds different roof surfaces and building extensions.
- **LOD₃** adds facade and roof details, approaching an architectural model.
- **LOD₄** includes interior.

[Gröger et al., 2012; Arroyo Oñori et al., 2018a]

Biljecki et al. (2016) defined a refined set of 16 LODs, for a stricter specification and less ambiguity.

2.2.2 CityGML for Microclimate Simulation

There are many different use cases for CityGML, both for visualisation purposes as analytic purposes. For example estimation of solar irradiation, energy demand estimation and microclimate analyses [Biljecki et al., 2015].

In order to make CityGML more suitable for different use cases, Application Domain Extensions (ADE) can be used. These make additions to the CityGML data model, like new properties for existing classes or new object types [Gröger et al., 2012]. A useful extension for microclimate simulation can be Energy ADE, where some of the classes are extended and some new classes are defined, for the purpose of urban energy simulation [Nouvel et al., 2015; Benner, 2018]. This extension can help store data related to energy in CityGML, like thermal properties of materials.

2.3 BIM

A significant part of the more detailed required information for ENVI-met microclimate simulation, can be found in BIM data. BIM is widely used in the designers world. It is a 3D information system at a very high level of detail, intended for different disciplines in the building sector, to work together in the same model. The model can be imported and edited a variety of different software packages used by the different disciplines. The open standard for exchanging BIM information between different platforms is IFC, by buildingSMART. In this section the IFC format will be explained and use cases for sustainability and microclimate simulation will be noted.

2.3.1 IFC Format and Information

The IFC standard contains many different entities, that order elements and types in useful categories, like walls and windows or geometry types. Each entity can have its own unique attributes and representations, as well as more general inherited attributes.

In practice designers only use a subset of these entities and do not always follow the prescribed guidelines in the standard. Therefore in this research both official IFC schemas and existing IFC models will be examined, to see what entities are available and how they are used. This can then be considered when developing the and guidelines.

IFC2x3 and IFC4 are the most used versions, at this moment in time. IFC4 is the newest version and has several improvements when it comes to the GIS and green buildings domains. There's a better interoperability with GIS domain schemas, like CityGML, and more energy related and environmental entities. Also is it more efficient, since less lines are used for the same object and there are more options for 4D and 5D modelling [Liebich, 2013; Cheng et al., 2014]. Even though the advantages of IFC4 are present, especially taking into account the topics concerning this research, GIS and climate, IFC2x3 is still commonly used, and therefore both versions will be considered in this research.

IFC models are mainly used for designing purposes. For this purpose, the geometrical and topological correctness and georeferencing are less important. Therefore these models are more likely to contain issues like self-intersections, which can result in issues during conversion of the data. Also georeferencing is often done incorrectly [Arroyo Ogori et al., 2017].

2.3.2 IFC for Microclimate Simulation

Architectural designers and engineers use BIM for designing, computing and visualizing their project, and also for topics like planning, construction and installations. Because all information is available in one model, detecting possible conflicts is much easier. Also for cost estimation and facilities management after completion of the build, the BIM model can be used [Azhar, 2011].

The work of [Krygiel and Nies, 2008] describes how BIM can be used for sustainable design, for example: daylighting, energy modelling or using sustainable materials. BIM on itself is not really used for microclimate simulation. The LadyBug tool extension for Rhinoceros'Grasshopper do give some options for microclimate related topics.

Kim et al. (2016) developed a method for better building energy analysis in BIM. Cheng et al. (2014) evaluated the usefulness of the latest IFC standard, IFC4, for GIS and the green building domains.

2.4 GEOBIM

A lot can be learned from work done on GeoBIM, since they are working on a similar goal: combining geodata from city models and bim models in one model. Therefore they will run into similar challenges and problems, like feature selection, topological errors and working with location data.

The idea of GeoBIM is to intergrate BIM an Gis, so that the more detailed BIM data can enrich GIS data, and GIS data can provide context for BIM data. The problem is, these two domains work in a completely different way, using different technology, standards and syntax. Therefore their semantics, geometry and level of detail differ significantly, which makes conversion between the models difficult [Arroyo Ogori et al., 2018b; de Laat and van Berlo, 2011].

Different research has been done on this subject, with different approaches and goals. The most commonly seen goals being:

- getting detailed BIM data in the GIS environment
- getting context from GIS data in the BIM environment

In this research data will be extracted and combined from both BIM and GIS data, but will be stored in a different, third format.

In the GeoBIM project, [Arroyo Ogori et al. \(2018b\)](#) developed a methodology for the integration of IFC and CityGML. In this methodology IFC is transformed to CityGML, via IfcOpenShell¹ and CGAL². Also modelling guidelines for IFC are proposed. They found that existing IFC models contain many geometrical and topological errors, which makes conversion difficult. Also developing a conversion method containing all entities of IFC is unrealistic and not necessary, since mostly only a subset of entities is used. Therefore developing a subset containing the needed entities is essential, and for this thesis research a similar approach of creating an entity subset will be implemented, but with focus on the entities necessary for conversion to ENVI-met.

Also other research on the subject of GeoBIM is done. While many GeoBIM research projects are focussing on importing/converting data from one format into the other format, [de Laat and van Berlo \(2011\)](#) focusses on the development of an ADE for CityGML, where semantic IFC data can be stored in CityGML.

Other relevant research could be the work of [Kardinal Jusuf et al. \(2017\)](#), who researched the interoperability between IFC and CityGML geometry and the work of [Zhu et al. \(2019\)](#), where IFC geometry is transformed to Shapefiles. [Ma and Ren 2017](#) inventoried the work that is already done on the integrated application of BIM and GIS.

2.5 IMPORTING EXISTING 3D MODELS IN MICROCLIMATE SIMULATION TOOL ENVI-MET

In this research, information from existing 3D models will be imported in the ENVI-met ecosystem. Specifically, information from IFC and CityGML will be extracted, combined and converted to the ENVI-met area input file format. In GeoBIM projects could already be seen examples of how information from IFC and CityGML can be integrated. In this section research projects focussed on importing existing 3D models in ENVI-met will be discussed.

ENVI-met offers some suggestions for using existing models for simulation. A plugin is developed in the Rhinoceros' Grasshopper Ladybug tools, that can convert Rhino 3D designs to ENVI-met models areas and run simulations. Also in Envi-met Monde, which works with a vector based approach instead of a grid based approach for editing ENVI-met models. In here existing vector models (like shape files) and open worldwide databases (like OpenStreetMap) can be imported (ref.).

[Arapakis \(2019\)](#) has made some first steps in coupling CityGML with ENVI-met. He developed a method that creates simulation-ready ENVI-met input models from CityGML models and that can store the generated simulation results back in CityGML. A lot can be learned from this approach, especially for elements like trees and the general approach of voxelisation, etc, that will probably have a similar approach in this research. However, in this research more classes will be mapped and information that realistically is usually not found in CityGML or that can be more detailed, will be extracted from IFC models. This gives the extra element of combining the data, which element could best be used from which source. Also this research will have a more 3D focussed approach. The combination of these two projects could also be very interesting, since the more detailed simulation results could be stored back into CityGML to use the results in the GIS environment. More return attributes could be added to this. Antonello Di Nunzio developed a plugin for Sketchup, that can export the model as an ENVI-met Area Input File in 2.5D (<https://github.com/AntonelloDN/Envimet-inx>). This is very useful when a user is

¹ <http://ifcopenshell.org>

² <http://cgal.org>

used to creating Sketchup models or has the 3D model already available in a by Sketchup supported format. However, existing designs and 3D city models can be in a different format.

At this moment in time 3D models are mostly modelled by hand directly in the ENVI-met software.

3 | METHODOLOGY

In this chapter the methodology will be described. First, the main steps that were taken will be explained. After that the data that is used in the different stages of the research will be explained, and it will be explained what was done to prepare this data to be able to properly use it. Last the method used for conversion will be discussed and it will be explained why is chosen for this method, compared to others.

3.1 RESEARCH APPROACH

To answer the main research question and sub question, the following methodology was developed. In figure 3.1 the main steps in this methodology are shown. These steps were de following:

- **Literature review:** studying related topics, literature and existing research on the subject.
- **Data inspection:** acquiring data and analysing data requirements (3.1.1).
- **Mapping:** localizing required data and data characteristics in IFC and CityGML (3.1.2).
- **Conversion:** developing a conversion tool combining and converting IFC and CityGML to the ENVI-met file format (3.1.3).
- **Analysing and testing:** testing if the conversion results can be used for micro-climate simulation in ENVI-met (3.1.4).

The final steps were an iterative process, per subject, until the desired results were achieved. The four main steps are further discussed in the following subsections.

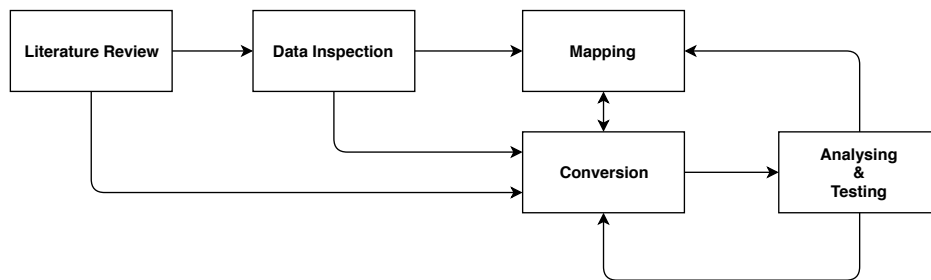


Figure 3.1: The main steps in this research.

3.1.1 Data inspection

In the data inspection phase is looked into what information is required for micro-climate simulation in ENVI-met, and if this information is generally available in the envisioned input models IFC and CityGML. In figure 3.2 the steps in these phase are further extended upon. The following processes are part of this phase:

- *Study ENVI-met data files and databases:* Both the documentation of ENVI-met and files generated in test runs of ENVI-met are studied. By going through the ENVI-met workflow and looking into the databases, and examining resulting data files in a text editor, insight in how the data is stored in text is achieved. It can be examined how changing parameters in the software, changes the files, to help understand how they work. The focus is mostly on the area input file, since this is where most of the data from IFC and CityGML will be stored, but also the other ENVI-met files will be looked into. The result is a list of the required data and how this data is formatted.
- *Representation in IFC:* With the list of required data can be looked into if this information can be extracted from IFC models. To see if the data is available is looked into literature, the official schemas of IFC and existing IFC models from practice. In the official schemas can be found if the information can be stored in an IFC model and can be found in very much detail how the data could be formatted. In the existing models can be checked if the information is generally recorded in reality and if this is done as expected or not. The existing IFC models are examined by going through the text file and by examining the 3D model in a 3D model viewer. Also, looking through the datastructure created by python library IfcOpenShell in debugging mode, can give a lot of insight. The result is list of data that can not be found in the IFC schemas, and data that is often missing or misrepresented in existing models.
- *Representation in CityGML:* A similar approach as for IFC can be used for CityGML. With more of a focus on the official schemas, since CityGML has a clear structure based on classes and LOD, which makes it easy to recognize if information is going to be there. Existing files are studied in both a text editor and a 3D model viewer. The result is a list of data that is generally not found in CityGML models.
- *Comparison IFC and CityGML:* By comparing which information is missing in both the input models and comparing how information is represented when it is available in both models, it can be established which information will be used from which model, and therefore which information needs to be represented in each model.

The results can be found in chapter 4.

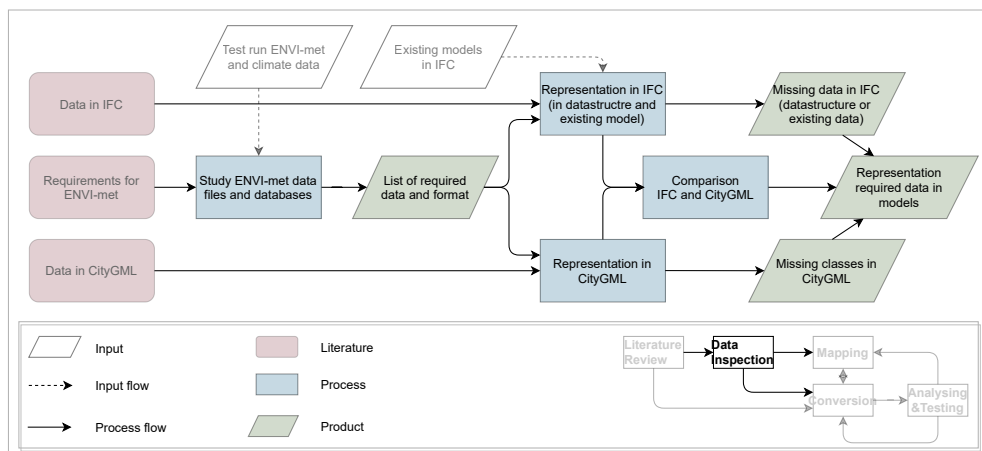


Figure 3.2: The steps in the data inspection phase

3.1.2 Mapping

In the mapping phase, it is decided which exact entities and attributes will be used to be able to get the required data to generate the ENVI-met area input file. The more detailed steps in this phase can be seen in figure 3.3. The steps are:

- *Mapping*: A mapping is done, where the translation between the elements in IFC, CityGML and ENVI-met format are established. Mapping is the process where the necessary data fields from the source files, are linked to the data fields in the destination file. In this phase, the necessary entities from IFC and CityGML, that are needed to generate the required elements in the desired format, will be established.
- *Entity selection*: The mapping leads to a subset of IFC entities and CityGML classes that will be used. The subset can then be altered based on results from later phases.
- *Establishing guidelines*: Since there is more than one way to describe a model in IFC and also entities are not always used by designers as described in the data schemas, or not used at all, the information that was required by looking into the existing data, will be used to complete the subset and establish guidelines for designers. For example, when an important entity is often not used in models, a rule can be added to the guidelines the entity needs to be handled. Also the requirements for the CityGML model will be explained to the user.

The results can be found mostly in chapter 4.4, and the guidelines in section 6.1.

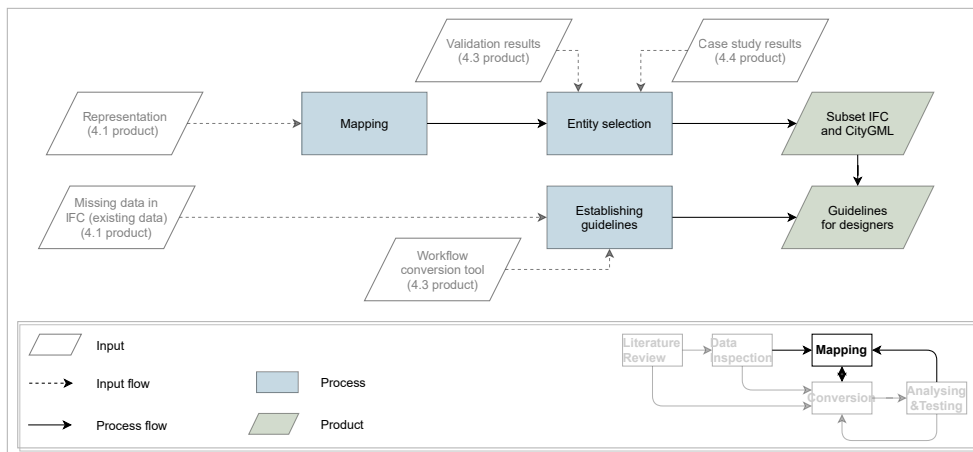


Figure 3.3: The steps in the mapping phase

3.1.3 Data Conversion

In the data conversion phase, the conversion tool, that generates an ENVI-met area input file based on input models in IFC and CityGML, is developed. The steps in this process are visualised in figure 3.4. These steps are:

- *Decide upon conversion method*: The options for this are already investigated in the literature study and some decisions are already made in the earlier steps in the process. But the exact method and tools should be definite at this point. More information about the conversion approach can be found in 3.3.
- *Extract information from IFC and CityGML*: During this process it will be researched how the information as stated in the mapping and the entity subset, will be extracted from the IFC and CityGML models with Python.

- *transform and combine data*: It will be examined how the data can be transformed, since they are in different spatial coordinate systems and different formats, and how the data can be combined. This will be done in Python.
- *write to ENVI-met file format*: The data will be written to the ENVI-met area input file format, with Python. The data structure of this file is already examined and described in data inspection phase. How the data will be adjusted to the correct format will be examined in this step.

All the above steps will result in a conversion tool, that can automatically generate the ENVI-met area input file, with as input IFC and CityGML models. How exactly this is done can be found in chapter 5.

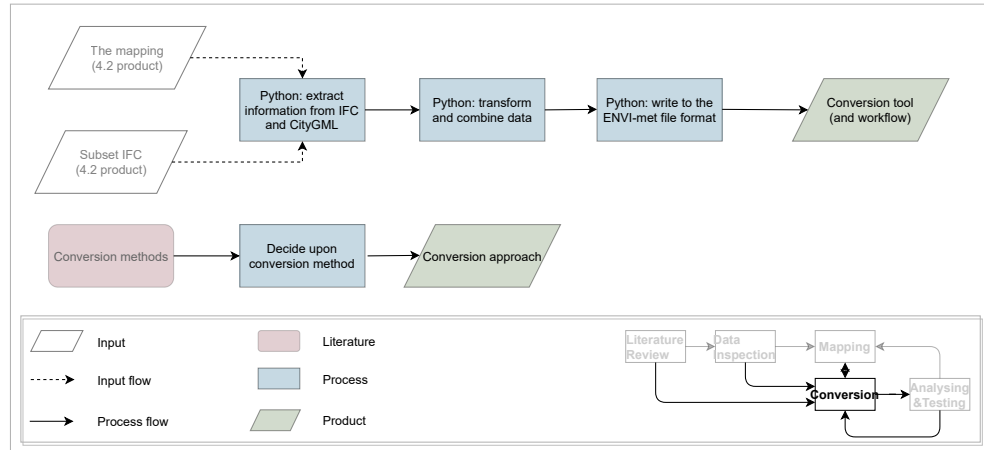


Figure 3.4: The steps in the conversion phase

3.1.4 Testing & Analysing

In the testing and analysing phase, the conversion tool is tested, to see if and how it works in practice. The steps in these phase can seen in figure 3.5 and are explained below:

- *Conversion*: With the created conversion tool, an area input file is generated. The resulting file can be inspected in ENVI-met Spaces. Based on the results the conversion tool and mapping can be adjusted accordingly.
- *Test study ENVI-met*: A test simulation will be done in ENVI-met with the generated Area input file as input. The microclimate simulation results can be analysed to check for any undesirable behaviour due to the input. Based on the results the conversion tool and mapping can be adjusted accordingly.

The steps in testing, the workflow, and resulting guidelines, can be found in chapter 6.

3.2 TEST DATA & DATA PREPARATION

In these research several IFC and CityGML models are used. Both in the data inspection phase and the testing phase. In the inspection phase is looked at actual models from practice, to see if information is generally available in the models and how it is represented, since this can deviate from the official schemas. Appropriate models are selected for the actual test study.

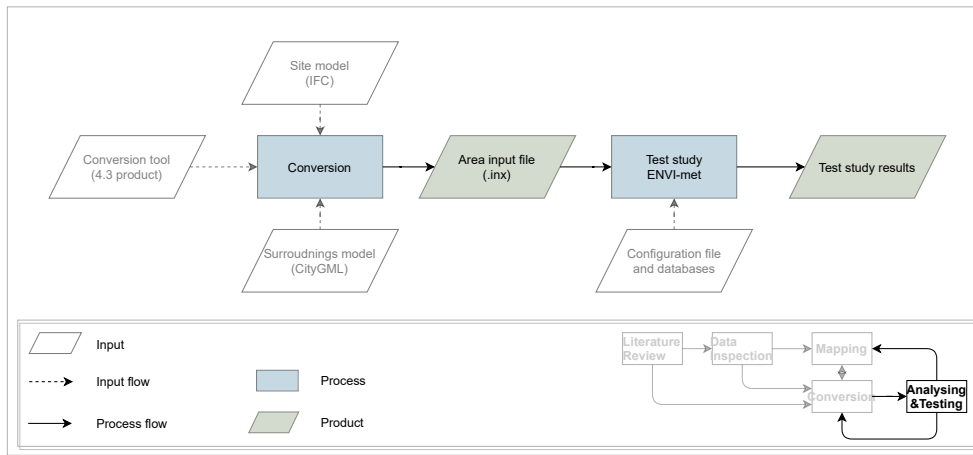


Figure 3.5: The steps in the testing and analysing phase

The following IFC models have been used during this research:

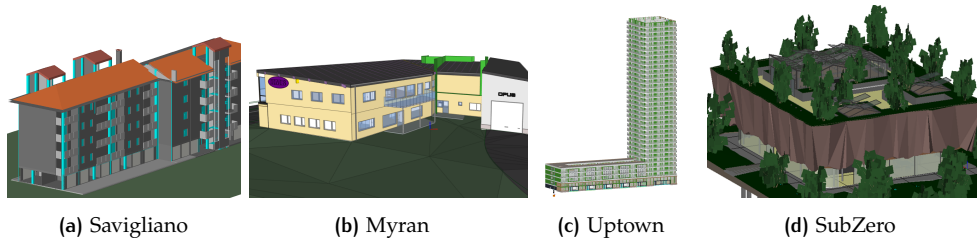


Figure 3.6: IFC models studied during this research

- *Savigliano*¹: model of residential building in Savigliano in Italy, IFC version 4
- *Myran*²: model of office building in Falun in Sweden, IFC version 2X3
- *Uptown*³: model of a residential building in Rotterdam in the Netherlands, IFC version 2X3
- *SubZero*: model of pavilion for the Floriade 2022 expo in Almere in the Netherlands, IFC version 2X3

The following CityGML models have been used during this research:

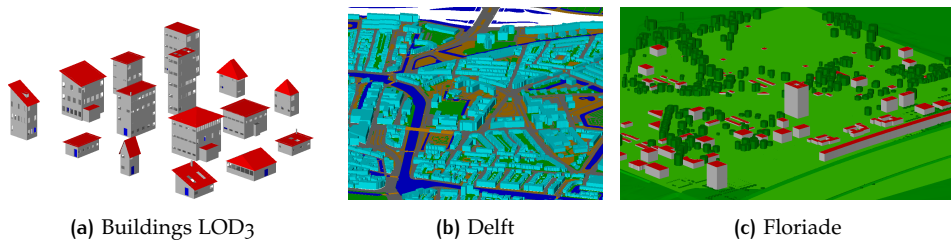


Figure 3.7: CityGML models studied during this research

¹ <https://3d.bk.tudelft.nl/projects/geobim-benchmark/savigliano.html>

² Mondo arkitekter <https://mondo.se/>

³ <https://3d.bk.tudelft.nl/projects/geobim-benchmark/uptown.html>

- *buildings LOD3*: generated buildings, only buildings, LOD3
- *Delft*: model of part of Delft, based on BGT and AHN3 data, all classes, LOD1. Also looked a little at models from other cities.
- *Floriade*: Floriade site in Almere in the Netherlands. Classes: Relief, Buildings, Vegetation and Landuse. LOD2

For the test study will be looked at the Floriade project in Almere. The CityGML of Floriade will be used, however a different IFC model will be used then the the SubZero pavilion. This is because this model does not make use of the appropriate entities like IfcWall and IfcRoof, and instead puts all elements in the entity IfcBuildingElementProxy. This makes it difficult to extract specific element, and certainly makes not regular conditions for a test.

Therefore is chosen to use for the Myran model for the test study, since this model represents more normal conditions that you would commonly see in these kind of models. To be able to use the model, it is manipulated by changing the location of the reference point to a location within the Floriade site.

3.3 CONVERSION APPROACH

There are different approaches for the conversion of the data. One aspect in this is the order in which data is extracted and converted. But also in the tools that are used can be different approaches.

In this is research is chosen to convert information from IFC and CityGML directly to the area input file of ENVI-met (figure 3.8). Another approach is converting and combining the IFC model to CityGML and converting from CityGML to the area input file of ENVI-met (figure 3.9).

Although the second approach seems a more versatile, since it is a more generic approach, that could be deployed in a variety of cases and software, instead of only ENVI-met. However, some specific detailed information from IFC might be difficult to store properly in CityGML, without losing some of the information. Via ADE and other tools it is possible to store this information, but this would make the solution less generic, so this approach loses some of its advantage.

Also, ENVI-met needs specific information and for other use cases, or even other microclimate simulation software, other information might be important.

Besides, the area input file of ENVI-met actually has a relative simple data structure with low resolution, and trying to store the information in a more complex data structure first, that probably needs more information that is redundant later, can simply overcomplicates things.

Therefore is chosen to extract and convert directly to the area input file of ENVI-met, as an effective method without overcomplicating things. The necessary information will be extracted from both the input models, and converted and combined in the ENVI-met area input file data structure.

The process off extracting, converting and combining of the data itself, will be done via a python 3 program, developed for this research. More details can be found in chapter 5.

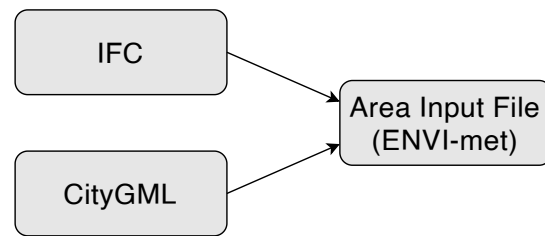


Figure 3.8: Conversion approach where information is extracted from both input files, converted, combined, and inputted directly into the ENVI-met Area Input File format. This is the approach used in this research



Figure 3.9: Conversion approach where IFC is converted to CityGML and combined and then CityGML converted to the ENVI-met Area Input File format. This approach was chosen to not be used in this research

4

DATA REQUIREMENTS, CHARACTERISTICS AND IDENTIFIERS

The microclimate simulation tool ENVI-met, focuses on the simulation of surface-plant-air interactions. As an output of the simulation, it gives information about the state of the atmosphere, data along the buildings facades and roofs, 1D inflow model representing surroundings of the model domain, pollutant concentration in the atmosphere, radiation fluxes, time series and profile data at defined receptor points, state of the soil, solar access, soil surface and state of the vegetation. Most of this output data is in a binary format and can be interpreted with the supplied analysing tool Leonardo.

To be able to generate these outputs, ENVI-met needs certain input data for its calculations. Therefore it is necessary to establish what exact information is needed as an input for the microclimate simulation in ENVI-met, and where this information can be found in IFC and CityGML schemas and data from practice. By doing this, a mapping can be made, showing what required information can be found in what data. In this chapter will be discussed where required elements can generally be found in the IFC and CityGML data and what exact characteristics the input data should have for their proper use in the conversion, and which exact entities will be used.

4.1 INPUT FILES FOR MICROCLIMATE SIMULATION IN ENVI-MET

In order to run the simulation, ENVI-met (version 4.4.6) needs the following information as input data:

- Information about geometry and its location at the site
- Information about the properties of the objects
- The parameters of the simulation

This information is input via the area input file, databases and simulation file. The required input files, and a short description of them, can be seen in table 4.1. In this table, the type of data, the file type and format this information can be found in, and a short description of what data these files contain, can be seen. A more detailed explanation and the relationship between the different input files can be found in the following subsections.

All ENVI-met files are written in the ENVI-met Markup Language (EML) format. This is an XML-based file format. It supports a subset of XML features, but also adds extra features like: matrix-data tags, spare-matrix-3D tags, bitmap tags, collections tag and color encoding. The root node of the file is <ENVI-MET_Datafile> and the header contains: filetype, version, revisiondate, remark, checksum and encryptionlevel. The data in the file follows in any order, after the header [ENVI-met GmbH, 2017].

Table 4.1: Required input files for microclimate simulation in ENVI-met and a short description of them

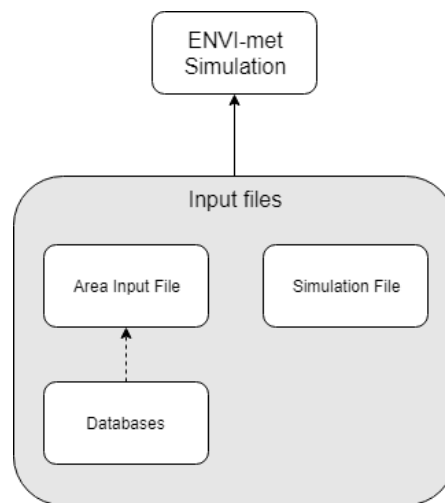
Input data	File Format	Description
Geometry	Area input file (.inx)	Contains the grid-based model geometry and the location of buildings, vegetation, soil profiles and pollution, in a 2D or 3D grid, with references to objects in the database (more detailed description in 4.1.2)
Object properties	Database (.edb)	Consisting of a System Database, a User Database and a Project Database, all containing object properties for Profiles, Soils, Materials, Walls, Single Walls, Plants, 3D Plants, Sources and Greening (more detailed description in 4.1.3)
Parameters of simulation	Simulation file (.simx)	Contains initial calculation parameters and boundary conditions, like meteorology information, simulation date and duration, and computing options (more detailed description in 4.1.4)

4.1.1 Input files relationship model

As briefly discussed before, three files are necessary to be able to perform the microclimate simulation in ENVI-met, the area input file, the database file and the simulation file. In this paragraph the relationship between the different elements in the input files are discussed.

Figure 4.1 shows the input files necessary for the microclimate simulation in ENVI-met. These are the area input file, the databases and the simulation file. There is a relationship between the area input file and the databases. The area input files contains locations of certain elements and a reference to corresponding objects in the database. These references are made via Ids.

For example: in the area input file the element 3Dplants can be found, which is a tree. In this element the root location of the tree is described and an ID is given. This ID corresponds with an ID of a tree species in the plant3D database. In this database entry, the properties of the tree species are stored.

**Figure 4.1:** Input files for microclimate simulation in ENVI-met and how they are connected

There are many more references between the area input file and database. In figure 4.2, a complete overview of the relationships between area input file elements and the database tables, are described. The elements of the area input files, and what they represent, are displayed in green. The database tables are shown in blue. The dotted lines are references to a database entry via an ID. In this diagram is also shown what attributes are present for each element or table. This also shows which information can be found in which file. The input files, their elements, and the attributes, will be explained in more detail in the next subsections.

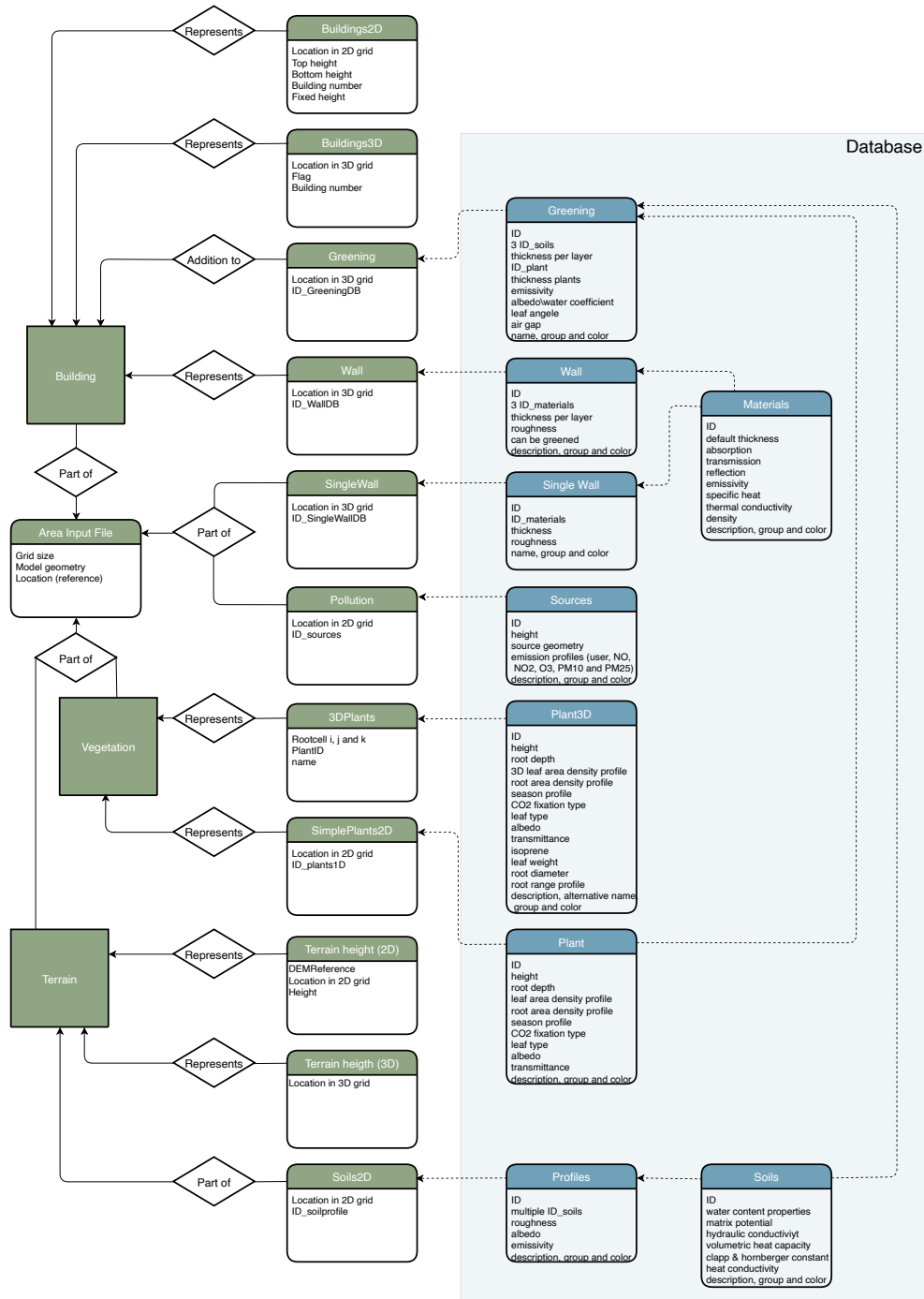


Figure 4.2: Relationship between the ENVI-met Area Input File elements and databases

4.1.2 Area input file

To run a microclimate simulation for an area, information about the site location and objects present at the location, is necessary. This is, among other things, information about terrain, buildings and vegetation. The location model is described in the area input file. As mentioned in the last paragraph, the location of certain elements is described and a reference ID to an object from the database, like a tree or material, at that location is given.

Just like all other files in ENVI-met version 4, the area input file is formatted in EML, the ENVI-met Markup Language, and therefore begins and ends with the `<ENVI-MET_Datafile>` start- and end-tag, and start with the `<Header>`.

To be able to describe 2D and 3D data, EML adds the special items 'matrix data' and 'spare-matrix-3D', to the XML syntax. With the matrix-data type, an I,J matrix can be defined, with the attributes dataI, being the number of elements in each row, and dataJ, the number of rows. The data on each row, represents all X data with the same Y index, and the data on a row is comma separated.

With the spare-matrix-3D type, 3D information can be described. A matrix is defined with the number of cells in the X, Y and Z direction. Also a default value is defined. Therefore, only the data different to the default needs to be noted. The rest of the matrix is filled with the default value. The format of each data element is the location in the matrix, followed by the value(s), separated by commas, with each data point on a new line [ENVI-met GmbH, 2017].

These ways to describe 2D and 3D matrices, are used in the area input file to describe geometry and store location specific information, like for buildings or soils.

Each area input file has a `<baseData>` tag, containing `<modelDescription>`, `<modelAuthor>` and `<modelcopyright>`. It also needs to have the elements `modelGeometry`, `nesting`, `location data`, and `default settings`. These elements and also the elements of the actual containments of the model, like buildings and greenery, are explained in the list below.

In this list, each element that is found in the area input file and its attributes is explained and their corresponding tags given. Also in appendix A, an extensive table can be found, explaining all elements and attributes and the exact tags and format of the data (type and unit). This data sheet is created to know exactly what data is necessary for the area input file and in what exact format it is expected. So that it can be used when selecting IFC and CityGML entities and especially when programming the area input file generation tool. The list below is more of a summary, describing the elements and explaining the most important characteristics for understanding and finding the necessary information in IFC and CityGML.

Model Geometry `<modelGeometry>`

In this element information about the model space can be found, like the size of space that needs to be modelled and the size of a grid cell.

- **Number of grids** `<grids-I>`, `<grids-J>`, `<grids-Z>`: The number of grid cells in each direction (Figure 4.3 a).
- **Grid size** `<dx>`, `<dy>`, `<dz-base>`: The size of one grid cell in each direction (Figure 4.3 a).
- **Splitting** `<useSplitting>`: If True, the dz of the lowest gridbox is divided in 5 subcells, to create more detail in the processes near ground level (Figure 4.3 b).
- **Telescoping** `<useTelescoping_grid>`: If True, dz increases with height. It can be used when a model contains higher buildings, but the processes at upper part don't require the same level of detail (Figure 4.3 b).

- **Stretch factor** <verticalStretch>: The telescoping factor. The factor with which the height of each grid cell grows (Figure 4.3 b).
- **Stretch start** <startStretch>: The height where telescoping starts (Figure 4.3 b).
- **3D** <has3DModel>, <isFull3DDesign>: Two attributes concerning if the model contains elements described in 3D and if the model is fully described in 3D.

Location data <locationData>

In this element information about the real world location and orientation of the model is stored (Figure 4.3 c).

- **Rotation** <modelRotation>: Model rotation out of grid North.
- **Projection system** <projectionSystem>: The reference system.
- **Reference point** <realworldLowerLeft_X>, <realworldLowerLeft_Y>: Two attributes giving the real world coordinates of the lower left corner of the model.
- **Location** <location.Longitude>, <location.Latitude>: The longitude and latitude of the location.
- **Time zone** <locationTimeZone_Name>: The name of the time zone.
- **Time zone location** <locationTimeZone_longitude>: The longitude of the reference time zone.

Nesting <nestingArea>

The nesting area refers to grid cells surrounding the model space. These can be used to create area around the model, so that simulation is more reliable at the border of the 3D model. This can also be achieved by making the 3D model grid bigger, without adding 3D elements (adding empty gridcells around the model), so that no 3D elements are close to the border of the model. But in case of nesting, the further from the core of the model, the larger the size of the grid cell, therefore less computation heavy, but it is less precise since they can only have two soil profiles in chessboard pattern (Figure 4.3 d).

- **Number of nesting grids** <numberNestinggrids>: The number of gridcells moving outward from 3D model.
- **Soil profiles** <soilProfileA>, <soilProfileB>: Two soil profiles can be chosen (the same or different ones), which will be in a chessboard pattern.

Buildings (2D) <buildings2D>

Information about buildings, described via 2.5D matrices (Figure 4.3 e).

- **Tops of buildings** <zTop>: I,J matrix, with at each point the height of the top of the building.
- **Bottoms of buildings** <zBottom>: I,J matrix with at each point the height of the bottom of the building.
- **Building number** <buildingNr>: Each building gets assigned a number. This attribute contains an I,J matrix with at each point the building number. When no building is present, the value is 0.
- **Fixed height** <fixedheight>: I,J matrix where each point says if the height at this point is absolute (meaning not following terrain). Otherwise height values are added to the terrain heights.

Buildings (3D) <buildings3D>

Information about buildings, but described in a different way, via points in a 3D matrix (Figure 4.3 f).

- **Building voxels** <buildingFlagAndNumber>: Attribute with information about if a voxel contains building and the building number. Only the voxels that are actually containing building need to be noted. Each voxel is described as (I,J,Z,f,nr) and can be explained as follows:
 - *Grid location (I,J,Z)*: The location of the voxel in the 3D matrix.
 - *Building flag (f)*: if the voxel contains building.
 - *building number (nr)*: The number corresponding to the building.

Building walls and roofs <WallDB>

Information about walls and roof, which can contain multiple different layers of material, described via points in a 3D matrix. This element is used to describe walls and roofs of buildings (Figure 4.3 g).

- **Wall voxels** <ID_wallDB>: Contains the information of the walls used for buildings for each voxel. Only voxels containing wall have to be noted. Each voxel is described as (I,J,Z,ID,ID,ID) and can be explained as follows:
 - *Grid location (I,J,Z)*: The location of the voxel in the 3D matrix.
 - *Wall ID vertical I direction*: References to an object in the 'wall database', for the vertical I direction of the voxel.
 - *Wall ID vertical J direction*: References to an object in the 'wall database', for the vertical J direction of the voxel.
 - *Wall ID horizontal direction*: References to an object in the 'wall database', for the horizontal direction of the voxel.

Single walls and roofs <SingleWallDB>

Information about walls and roofs with a single layer of material, described via points in a 3D matrix. Used for (semi) enclosed spaces that do not behave like a building, for example a shed or a bus stop.

- **Single wall voxels** <ID_singlewallDB>: Works similar to wall voxels, only the IDs reference to the 'single wall database' instead of the 'wall database'.

Green walls and roofs <GreeningDB>

Information about walls and roofs that have greening added to it, described via points in a 3D matrix

- **Greening voxels** <ID_greeningDB>: Works similar to wall voxels, only the IDs reference to the 'greening database' instead of the 'wall database'.

3D plants <3Dplants>

This element contains information about plants that can only be described in 3D, like trees (Figure 4.3 h).

- **Root grid cells (i,j,k)** <rootcell.i>, <rootcell.j>, <rootcell.k>: The root location of the plant.
- **Plant ID** <plantID>: An ID that references to an object in the 'plant3D database'. In this database the shape and physiological properties of the plant are stored.
- **Name** <name>: The name of the plant
- **Observe plant** <observe>: True if marker is set

Simple plants <simpleplants2D>

In this element information is found about plants that can be described in 2.5D, like grass and hedges (Figure 4.3 i).

- **Plants** <ID_plants1D>: I,J matrix with at each point that contains plants, an ID that references to the object in the 'plants database'.

Soils <soils2D>

The soil element holds information about soil materials, expressed in soil profiles. These include also infrastructure and water bodies. The information is stored in a 2D matrix (Figure 4.3 j).

- **Soil profiles** <ID_soilprofile>: I,J matrix with at each point an ID that references to the object in 'profiles database'.

Pollution <sources2D>

Information about pollution sources, described in a 2D matrix (Figure 4.3 k).

- **Sources** <ID_sources>: I,J matrix with at each point that contains pollution sources, an ID that references to the object in the 'sources database'.

Elevation (2D) <dem>

This element contains information about the terrain height in 2.5D (Figure 4.3 l).

- **Reference point elevation** <DEMReference>: The elevation of the reference point above sea level
- **Terrain height** <terrainheight>: I,J matrix, with at each point the terrain height relative to the reference height.

Elevation (3D) <dem3D>

This element also contain information about the terrain height, but described via points in a 3D matrix (similar to buildings in 3D).

- **Terrain** <terrainflag>: The location in the 3D matrix with terrain flag (I,J,Z,f), where the terrain is elevated. So if a voxel is part of the terrain.

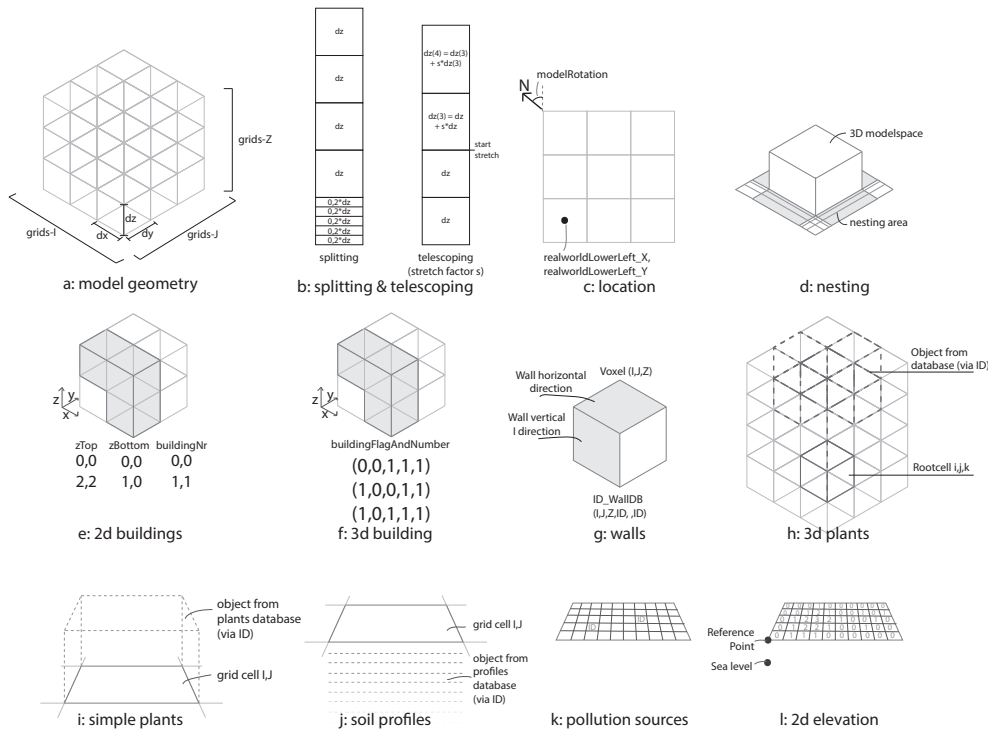


Figure 4.3: Illustrations clarifying the ENVI-met Area Input File elements and their attributes

4.1.3 Database system

Objects and their properties are stored in a database system. ENVI-met comes with a predefined series of objects for trees, materials, soils, etc. These are stored in

the *system database*. There is also the possibility for the user to add new objects, or change the properties of existing objects. These will then be stored in the *user database*, or in the *project database* if specified for the project. The user database is where the user can add their objects and change properties, which will then be used for all their projects. The project database is specific for a project and only used for the assigned project.

The three databases are formatted and structured in exactly the same way. The database used for a project simply exist of an addition of all objects in the system and user database, or the system and project database. When an object in the system database and an object in the user/project database, have the same ID, the object from the user/project database will be used.

In table 4.2, the three different databases are described.

Table 4.2: The three different databases in ENVI-met and a description of them

Database	Description
System database	Database provided with ENVI-met and maintained by model developers. Not supposed to be edited by user.
User database	The user's personal additions and changes to the database, available from all project, overrules item from system database when same ID is used.
Project database	Data specific to a project, especially helpful when the user wants to adjust certain properties of objects, but does not want this for all projects. Overrules items from system database when same ID is used.

The ENVI-met database exists of different database tables, namely profiles, soils, material, wall, single wall, plants, sources, greening and plant3D. Each entry in the database has an ID, name/description, color and group. Each table also has specific properties for each entry. In appendix B, a table where each database table and their attributes are shortly explained can be found. This information is important to be able to link information from CityGML or IFC, to the correct object from the database. This will be further explained in chapter 4.4, where also the most important attributes that can be used for linking and why these are usable will be explained.

4.1.4 Simulation file

In the simulation file, the configuration of the simulation can be found. This includes the date, time and duration of the simulation, weather data, and also computing options.

The following elements can be found in a simulation file:

- main data:
 - name and simulation output location
 - Area input file used for simulation
 - start date, start time and simulation duration
 - wind speed and wind direction
 - roughness length at measurement site
 - temperature
 - humidity
- simple forcing or full forcing data

- parallel, how many CPU cores can be used for the simulation
- ENVIguide, optional parameters for very specific advanced settings like custom soil temperature, plant conditions, specific adjustment factors, use of different models or methods etc.

For modelling the weather with custom data two options are possible: simple forcing and full forcing.

Simple forcing is quicker and relies on less input data, but is less precise and simulates only a short period of time. Only temperature of the atmosphere and relative humidity (on 2m height) are forced over a time period of 24 hours with data for every hour. Also Wind speed measured on 10m height, wind direction and roughness length at measurement site are taken into account.

For full forcing more data is necessary. A data entry is necessary for every 30 minutes and can be for up to a year. Each entry needs a date, time, direct radiation or low clouds, diffuse radiation or medium clouds, long wave radiation or high clouds, absolute temperature, relative humidity, wind speed and wind direction. The measurement height for air temperature and humidity and for wind speed and -direction, can be defined, just as the roughness length.

Overall, simple forcing is okay for estimations for a short period of time, but full forcing is more detailed and precise, especially for long term simulations.

The weather data necessary for micro climate simulation can be found in data 'verzameld' by weather stations. For the Netherlands this data is provided by the KNMI. For simple forcing, all the necessary data can be found in 'KNMI uur data', which gives i.a. wind direction, wind speed, temperature and humidity. It even includes information needed for full forcing, but since it is only for every hour, it is only useful for simple forcing. Information for full forcing can be found in datafiles per category for every 10 minutes.

More details about the exact format of the data and where exactly and in what format it can be found in the KNMI data, can be found in appendix

The ENVI-met simulation file can be generated with the tool ENVIguide, which is included in the ENVI-met suite. The tool leads the user through different steps, to input all the data. A choice for simple forcing or full forcing can be made and leads to a step to input the weather data necessary for these. In the tool created for this project, a small tool will be included that can extract the necessary data for simple forcing from the 'KNMI uur data' for a date and location chosen by the user.

4.2 REQUIRED INFORMATION IN IFC AND CITYGML

A lot of the required information for microclimate simulation in ENVI-met, can be found in existing IFC and CityGML models. Information about geometry and location of elements like buildings, vegetations, etc. can be found in these models and are necessary for generating the 'area input file'. In this section will be discussed if these elements exist in the model schemas, but also in real data. Which data is actually usable and which exact characteristics this data should have to be able to successfully generate the area input file and link to the right database entries, will be discussed in section 4.4.

4.2.1 Representation in schemas and existing data

The representation of the required information in both the official schemas and existing models, will be checked. In the official schemas can be seen what entities and attributes are available for describing certain elements, and how they are supposed

to be used. However, in practice, these entities could be used wrong, or certain elements could not be very common to be represented in the model, so could often simply be missing, even though the official schemes give the option to represent the element. Therefore models from practice will also be inspected, to see if required elements can generally be found in the data and if they are represented correctly.

In table 4.3 can be seen if the objects that need to be available for the ENVI-met area input file, are generally represented in the IFC and CityGML models.

Model geometry and nesting area are not included as required elements in the table, since these contain only parameters specific to ENVI-met modeling and simulation, so they are not (physical) objects that can or need to be found in 3d models. The ENVI-met element soils is split up in soils, infrastructure and water, in the table. Even though these are all described in one element in ENVI-met, it is important to know if all this separate information is available in the 3D models, and not just one part.

In the column 'representation in schema' is shown if entities and attributes are described in the official schemas, to represent the object. A check-mark (✓) means elements and attributes are available to represent the object. A tilde (~) means no specific entities and attributes exist to represent the object, but some information about the object can still be subtracted from it. An X-mark (X) means no entities and attributes to represent the object are available.

In the column 'representation in existing data' is shown if the objects can actually be found in existing IFC and CityGML models. A check-mark (✓) means information about the object can always be found in the model, using the right entities and attributes in the way they were intended. A tilde (~) means information about the object can sometimes be found in existing data and/or is often represented in the wrong way. An X-mark (X) means information about the object can generally not be found in existing models.

In the table can be seen that CityGML has a module for many of the elements, but it is possible to only use a selection of these modules, which is why the existing CityGML models do not necessarily contain all the required data. There are no specific modules or attributes to store material specific information for walls or soil, it might be able to derive some material information indirectly. An IFC model contains mostly information about the building, including material information. It can also have information about terrain. Other information like vegetation and roads can be in the models, but might be more difficult to derive and are often not represented. Location data is sometimes not applied how it is intended.

In subsection 4.2.2 and 4.2.3 will be explained how exactly these elements are represented in respectively IFC and CityGML.

4.2.2 Format in IFC

In the last subsection could be seen if some of the required information for the ENVI-met area input file, could be found in IFC models. In this subsection will be elaborated on where this information can be found and how it is represented.

Location data

Most information about the real life location of the model can be found in the Entity `IfcSite`. IFC models work with a custom coordinate system with a reference point to the real world. The real world location of this reference point can be stored in the attributes `RefLatitude` and `RefLongitude` in the WGS84 coordinate reference system. The reference height is relative to sea level and can be stored in the attribute `RefElevation`. This reference point will then be as the origin point of the model. All other coordinates in the IFC model are then relative to this reference point.

It is also possible for a translation of the reference point to be in `IfcSite` attribute `ObjectPlacement`.

Table 4.3: Representation of required elements for microclimate simulation in ENVI-met in both official schemas and existing data

Required elements	Representation in IFC schema	Representation in existing IFC data	Representation in CityGML schema	Representation in existing CityGML data
Location data	✓	~	✓	✓
Building 2D/3D	✓	✓	✓	✓
Wall/single wall	✓	✓	~	~
Greening	~	~	X	X
3D Plants	✓~	~	✓	~
Simple plants	✓~	~	✓	~
Soils	Soils	✓~	~	X
	Infra	✓~	✓	~
	Water	~	✓	~
Sources	X	X	X	X
DEM 2D/3D	✓	~	✓	~

The model rotation is also an important part for georeferencing, to know the orientation of the custom cartesian system that is used. This information can be found in IfcProject attribute RepresentationContexts, which contains attribute TrueNorth.

A reference point is optional in IFC version 2x3, and required in version 4. IFC version 4 also adds more entities for more exact georeferencing or referencing to other coordinate systems, IfcCoordinateReferenceSystem and IfcMapConversion.

In practice location data is often missing, incorrect or imprecise or implemented incorrectly. Therefore it is very important to have clear guidelines for georeferencing, since knowing the exact real world location is crucial when combining different 3D models

Building geometry

All elements containing information about building geometry can be found in the abstract entity IfcBuildingElement. Children of this entity are i.a. the entities IfcWall, IfcRoof and IfcSlab. IfcBuildingElement itself can not be instantiated. When a building element can not be described by one of its children, IfcBuildingElementProxy can be used.

The geometry can be found in the attribute representation (which is described with the entity IfcProductRepresentation(gives list of representations) subtype/child IfcProductDefinitionShape, and is inherited from/defined by IfcProduct). Can be represented as axis 2D geometry, surface geometry, body swept solid geometry or body clipping geometry.

Walls and materials

Information about walls and materials can be found around the same place as the building geometry. So also in the children entities of IfcBuildingElement, but in the IfcProductRepresentation subtype IfcMaterialDefinitionRepresentation. Here, materials (IfcMaterial) can be assigned to building elements, although this is more for visual representation.

To attach a material or even different material layers with various thicknesses, to building elements, IfcMaterialLayerSet can be used. It can be attached directly to the element or via relationship with IfcRelAssociatesMaterial...

Some of the properties like isExternal and thermalTransmittance could be useful.

Vegetation

Vegetation in IFC can be, if available, found in the entity `IfcBuildingElementProxy` (in version 2x3) or `IfcGeographicElement` (in version 4). Attributes are i.a. Name, Description, ObjectType, ObjectPlacement and Representation. `IfcBuildingElementProxy` is basically a residual group for elements that do not have a definition in IFC. `IfcGeographicElement` is a generalization of all elements within a geographical landscape, like trees or terrain. These could be used to seek out vegetation from the model, but it might be hard to derive which ones are actually vegetation.

Soils

The IFC2x3 schematics do not really provide specific entities to store soil information. In IFC4 the entity `IfcGeographicElement` could contain information about the terrain and roads can be found in `IfcCivilElement`. The entity `IfcCivilElement` is a stub for future extensions, that will have more specific entities for civil elements.

Some information might be able to be derived from the appearance material of the top layer of the soil, but this is highly subjective.

Generally not a lot of information about soil is found in existing IFC models.

Terrain

The terrain model can be found in `IfcProductDefinitionShape` in `IfcSite`. It can be represented as a footprint (curve set), with survey points (point set), facetation (usually result of triangulation of survey points, surface model), or body representation (brep or surface model).

4.2.3 Format in CityGML

In this subsection will be elaborated on where the required information for microclimate simulation could be found in the classes and subclasses of CityGML and how it is represented.

Location data

In contrast to in IFC, in CityGML generally does not work with a custom cartesian coordinate system, but with real world coordinates in an existing coordinate reference system. Each point in an object is then described in real world coordinates. (although it is possible to define your own coordinate reference system within the same CityGML document *ref). The coordinates reference system can be defined per object, but can also be defined globally in `gml:Envelope`.

Building geometry

In CityGML, buildings have their own class `Building`. In the subclass `bldg:Building` semantic data like year of construction, function, usage, measured height and number of storeys above and below ground, can be found, as well as geometric data. Geometry can be described as footprint and roofedge for `lodo`, solid or multisurface for `lod1` to `4` and multicurve for `lod 2` to `4`. Also terrain intersection for `lod1` to `4`. Recommended is solid for `lod1` and solid from bounding surfaces for `lod2` to `4`.

Walls and materials

In the `Building` class can also be found information about the geometry that makes the building like walls and roofs. These can be found in `bldg:boundedBy`. Wall surfaces, roof surfaces, ground surfaces, outer floor and outer ceiling are all useful here. Also `bldg:door` and `bldg:opening` could be used. (`Lod 2` to `4` (multi)surfaces (`lod1` possible when looking at surface normal))

Information about some single wall geometry could be found in some specific objects in the classes `Bridge`, `Building` and `Cityfurniture`.

For building materials can be looked at the `Appearance` class, in subclasses `Appearance`, where colour or texture is assigned per theme, and `X3Dmaterial`, where

colour and textural elements can be linked to a surface. These are mainly for appearance purposes. Some information can be subtracted from this, but that's very ambiguous. With energy ADE (material and construction module), information about material(layers), thickness and other material properties can be stored.

Vegetation

Information about vegetation can be found in the class *Vegetation*. The subclass *SolitaryVegetationObject* corresponds to the 3D plants in ENVI-met and has Elements for species, height, trunk diameter, crown diameter and geometry. The geometry can be either explicitly, with absolute coordinates, or implicitly, by reference to a shape and transformation (lod 1 to 4). In the subclass *PlantCover* the equivalent to simple plants can be found. Elements are average height and geometry. The geometry can be either described as multi-surface, or multi-solid for lod 1 to 4.

Soils

In ENVI-met the element soils consists of soilprofiles. Therefore these include information about infrastructure and waterbodies, since these are basically just a different profile. CityGML has its own classes for infrastructure and water, respectively *Transportation* and *Waterbody*. However, actual information about the soil itself is not found in CityGML.

In the *Transportation* class different subclasses are available for different kind of infrastructure. Parts of this infrastructure have attributes for function, usage and surfaceMaterial. From LOD1 the infrastructure is geometrically represented as a surface. From LOD2 this geometry can be divided in thematic areas (*TrafficAreas*), like cars, pedestrians, etc..

Waterbodies are from LOD2 represented as thematic surfaces outlining the waterbody, or can be a solid composed of surrounding thematic surfaces. Attributes are class, function and usage.

Terrain

Information about the terrain can be found in CityGML class *Relief*. It can be represented as a raster or grid (*RasterRelief*), a Triangulated Irregular Network (TIN) (*TINRelief*), breaklines (*BreaklineRelief*), or mass points (*MasspointRelief*).

4.3 COMPARISON

In order to decide what information can be extracted from which 3D model in a useful way, the above results are compared. The difference between the information from the different models is noted and it is decided which information will be extracted from which input model.

4.3.1 Differences

The following differences can be found about if and how information is represented between IFC and CityGML:

- Location data is present in both formats. In CityGML point in geometries are often represented via coordinates in a real-world coordinate system. In IFC usually a custom local coordinate system is used, with a reference point to a real-world coordinate. Sometimes in practice the referencing is done different or incorrectly.
- In both IFC and CityGML, information about buildings can be found. However in IFC this information is a lot more detailed. In IFC detailed information about building element structure and material is available.

- In CityGML vegetation is represented in its own class. In IFC the presence of vegetation is not common. Sometimes it can be found in the model, but in not very specific entities.
- Information about soil profiles is usually not found in either one of the formats. However in CityGML information about infrastructure (roads) and waterbodies are common, and have their own class. This can be used to at least define water and hardening like roads.
- Information about terrain height can be found in both models. The terrain model of the CityGML can easily be used, but sometimes there can also be terrain information available in the IFC file.

4.3.2 What information from which 3D model

In figure 4.4 can be seen from which input model the necessary information will be extracted. It is important to make this decision in order to focus on these specific elements for each model.

Location is a very important element, because this makes it possible to correctly combine the information from both input models. The building and its materials will be extracted from IFC and buildings surrounding this building will be extracted from CityGML. Vegetation and information about infrastructure and waterbodies can be extracted from CityGML, as well as a terrain model. Terrain and vegetation on the site location can be extracted from the IFC when it is available.

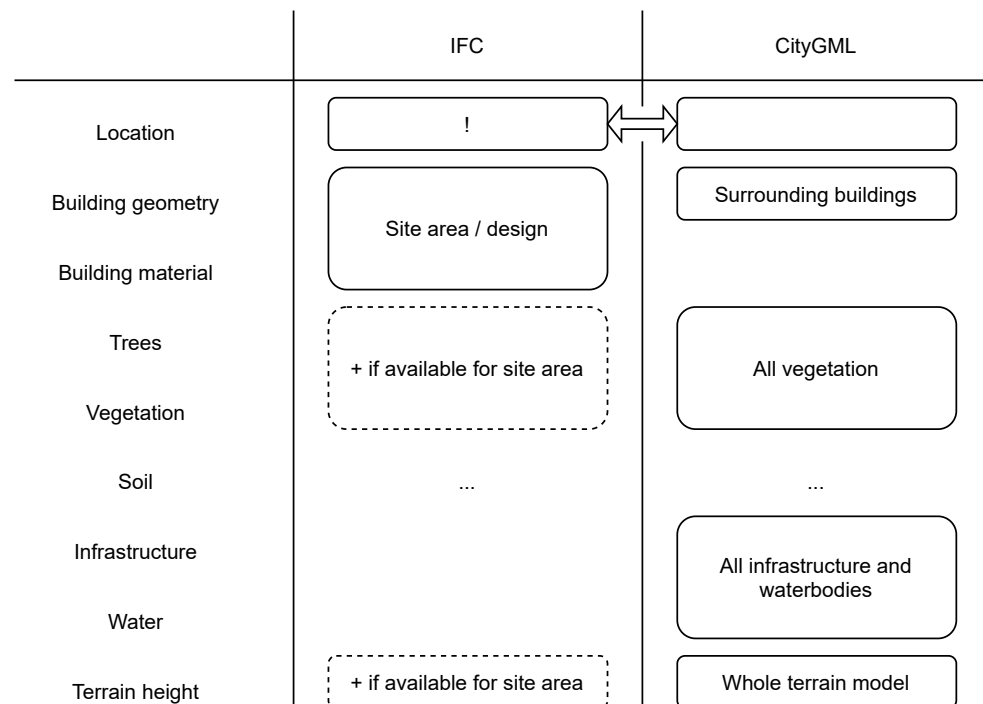


Figure 4.4: Schema describing from which input file the required elements will be extracted

4.4 CHARACTERISTICS OF THE DATA FOR IDENTIFICATION

Now we know if and where information about specific elements necessary for microclimate simulation in ENVI-met can be found in IFC and CityGML. But what

exact elements and attributes will be used and which identifiers can be used to assign objects and materials in ENVI-met? This chapter will focus on what characteristics of the data will be used by the conversion tool to identify the objects for ENVI-met.

4.4.1 Model Geometry

Model geometry is an element specific for the area input file for ENVI-met, and is therefore mostly based on user input and model design. More detailed information about the expected user input can be found in section 6.2.2 and about model design in 5.1.1. But for this conversion tool some of the model design decisions are based on the IFC and CityGML input models.

The most prominent information that is needed from the input files are their bounding boxes. These are needed to determine the size of the model, and to check if the input files are even within the same area.

In CityGML the bounding box can be extracted from the envelope. In the envelope, attributes can be found for the lower left corner and upper right corner of the bounding box that enclosed the whole model.

The bounding box of the IFC model is extracted by going through the points in the site representation and saving the lowest and heights x, y, and z values. If the site model is unavailable, the bounding box of the building could be used for this purpose.

In figure 4.5 a diagram showing all the related attributes:

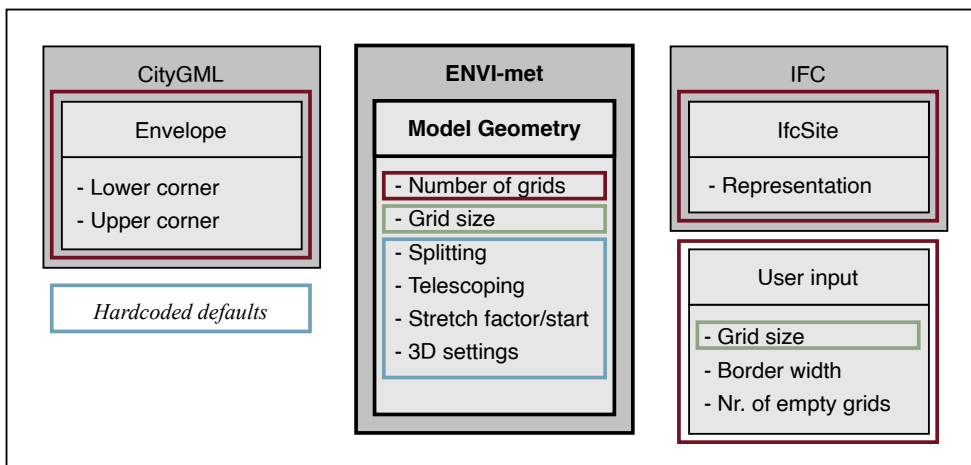


Figure 4.5: Diagram showing which attributes in IFC and CityGML can be used to determine the model geometry attributes for the ENVI-met Area Input File

4.4.2 Location

The location data is also information more specific for the area input file for ENVI-met. It is what links the model to a real world location. But in this project it is still mostly based on information from the IFC and CityGML input models.

The same projection system as the one that is used in CityGML will be used for the location data in the area input file. It can be found as an attribute called srsName in envelope.

For the ENVI-met model, the same rotation will be used as in the IFC model. The true North can be found in IfcProject. The reference point of the model can be based on the reference point from the IFC model and user input. Based on the user input and the bounding box of the IFC model, it can be calculated where the area input file referenc point would be within the IFC custom coordinate system. With

the reference point, rotation and units known, this point can be transformed to a real world location (more about transformations in 5.1.2).

A diagram showing the related attributes can be seen in figure 4.6.

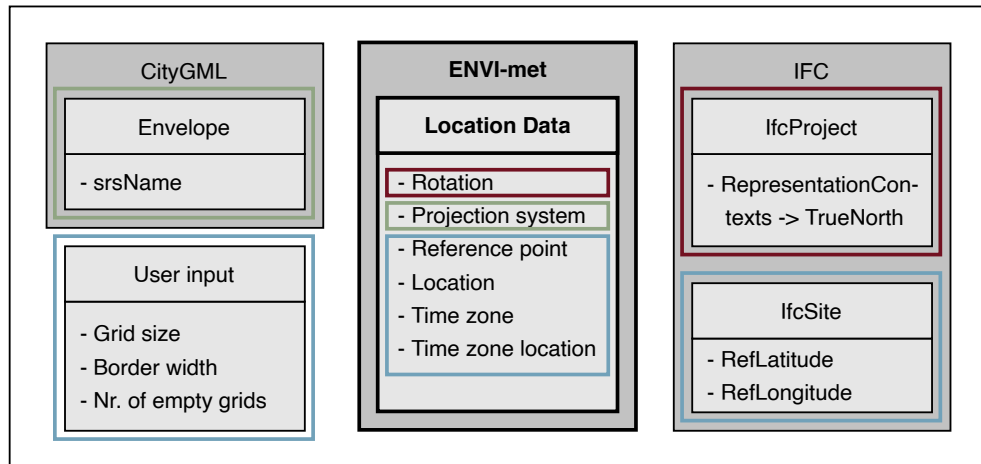


Figure 4.6: Diagram showing which attributes in IFC and CityGML can be used to determine the location data attributes for the ENVI-met Area Input File

4.4.3 Buildings

The building element is an element that has to be fully based on the provided IFC and CityGML input models. A building in ENVI-met is defined with multiple elements: building, walls (material), and eventual greening (green walls/roofs).

In ENVI-met element building3D the basic shape of the building is defined, by appointing the voxels that contain building.

In CityGML all information about the building can be found in the class Building. There are multiple different geometric representations possible. In this research the geometric information of each building will be extracted from lod2multisurface. Based on this list of surfaces, that forms a closed building, for each building, the building voxels can be determined via conversion.

In IFC the information can be a lot more complex. All information about the building can be found in a subelement of IfcBuildingElement. The building elements that can be used to base the building voxels on are: IfcWall, IfcSlab and IfcRoof, specifically those that have the property isExternal True. These are the outside surfaces of the building. The geometry of these can be found in representation. However, each element is usually a solid on its own. For proper use in the program they can be simplified to a sole surface.

The walls are basically the outside of the building, so the face between a building voxel and not building voxel. However, a specific wall from the ENVI-met database needs to be assigned to the wall. These contain the structure and materials of the wall.

In CityGML basic materials could be based on the type of surface, like walls, roofs, windows. In IFC more detail about the compositions of the building elements is available. However, the materials will not be implemented within this research.

The ENVI-met elements and attributes and their related attributes in IFC and CityGML are illustrated in figure 4.7.

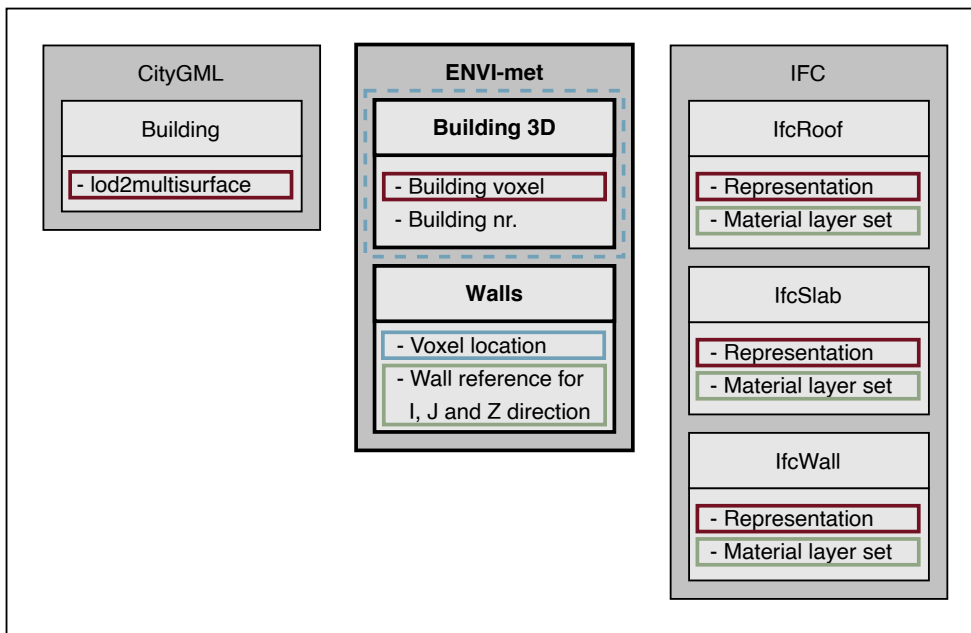


Figure 4.7: Diagram showing which attributes in IFC and CityGML can be used to determine the building and wall attributes for the ENVI-met Area Input File

4.4.4 Vegetation

Information about vegetation can be found in CityGML. CityGML class *SolitaryVegetationObject* link directly to ENVI-met element 3D plant (trees) and CityGML class *PlantCover* links directly to ENVI-met element simple plants (bushes/hedges). They contain respectively attributes height and average height, which can be linked to the height of entries in the 3Dplants database and plants database.

Sometimes information about vegetation can also be found in IFC, however identifying the object might be difficult, since it does not have its own entity. Locating the element can be based on the attribute 'name' in *IfcBuildingElementProxy* (in version 2x3) or *IfcGeographicElement* (in version 4), but this can be highly subjective. Once the object is identified, the height can be extracted from the bounding box.

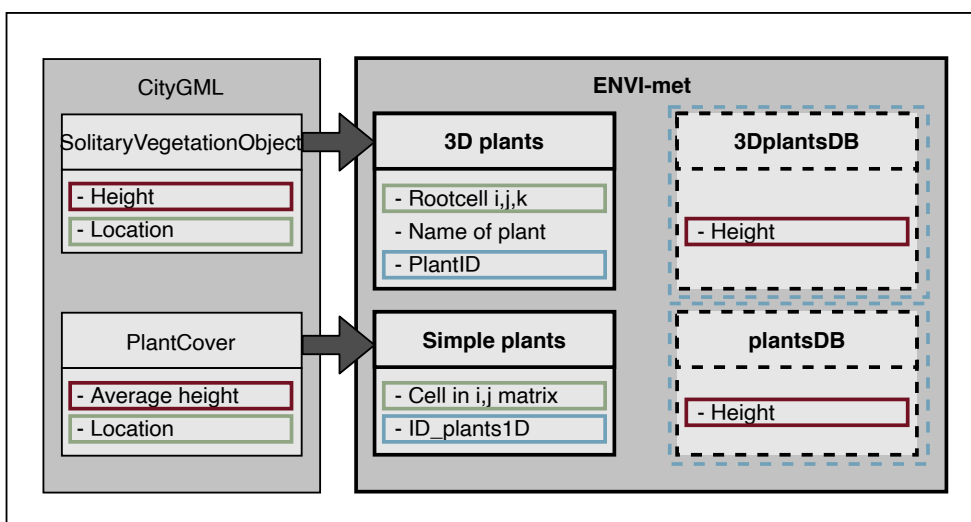


Figure 4.8: Diagram showing which attributes in IFC and CityGML can be used to determine the 3D plants and simple plants attributes for the ENVI-met Area Input File

4.4.5 Terrain

In ENVI-met the terrain height of each grid cell needs to be determined.

In CityGML information about the terrain height is found in the class Relief. In this research the [TIN](#) representation option will be used. This representation consist of multiple triangle surfaces forming a surface.

In IFC the surface model can sometimes be found as a representation attribute in IfcSite.

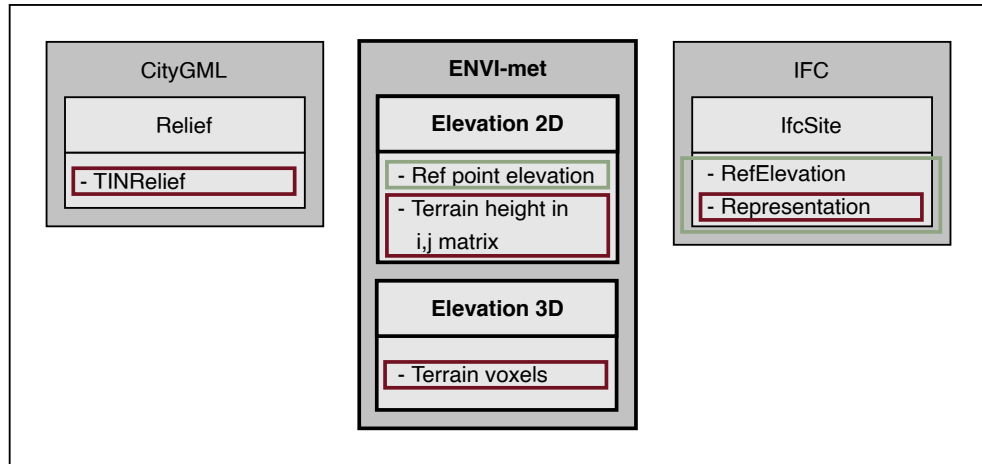


Figure 4.9: Diagram showing which attributes in IFC and CityGML can be used to determine the terrain 2D and 3D attributes for the ENVI-met Area Input File

5

CONVERSION OF IFC AND CITYGML MODEL DATA TO MICROCLIMATE SIMULATION SOFTWARE ENVI-MET

Knowing what information is necessary for microclimate simulation, and where this information can be abstracted and derived from, the next step is actually extracting and converting this data and putting it in the right format. This transformation of the data can be visualized in a few simple steps, as can be seen in figure 5.1. The necessary parts of the IFC model (A) and CityGML model (B) will be extracted, converted to one format and stored in a datastructure (C). From this datastructure the ENVI-met area input file (D) can be generated.

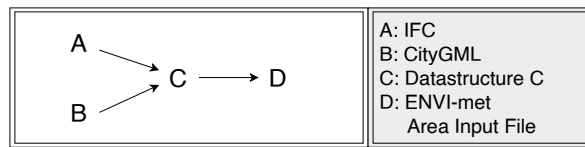


Figure 5.1: transformation between the models

To achieve this the conversion tool consist of the following modules, as can be seen in figure 5.2. In main the main modules are called and the user input is gathered, and supplied to these modules. The conversion module is the core of the tool. It uses IFC and CityGML parsers to extract information from the input modules. Converts the data using the transformation module that can transform between different spatial reference systems, and the elements module that provides an interface for the data structure where the information is stored. The finished data structure can than be used by the EML writer, to generate the ENVI-met area input file.

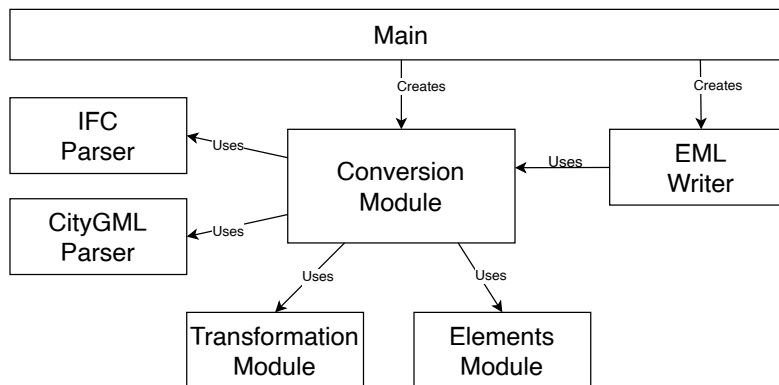


Figure 5.2: a schema how the different modules in the conversion tool co-operate

How all this is done, how it works and how the different modules work together, will be further explained in this chapter.

5.1 MODELLING OF THE DATA IN THE CONVERSION TOOL

In order to be able to make a model for ENVI-met from other input files, some choices and properties need to be recorded. For example the dimensions of the model, which parts of the input models will be used, what input is expected from the user, and how the transformation between the models will be handled, since they are all in different spatial reference systems. In this section will be elaborated on these subjects.

5.1.1 Model design

In order to make a model, some ground rules need to be established, about which parts will be used and what dimensions the model will be. In figure 5.3 some general terminology used throughout this section, is explained.

When the site or site area is mentioned, it refers to area of interest. This is the area where the user has an IFC file for, for example a building designed by an architect. Information about this area typically has a higher level of detail, and since it is the area of interest, it will be the centre of the model that will be created.

The surroundings or surrounding area is the area around the site. Information about this area can be extracted from a CityGML model. These are typically models containing information about a large area. A selection from this area will be used as surrounding area.

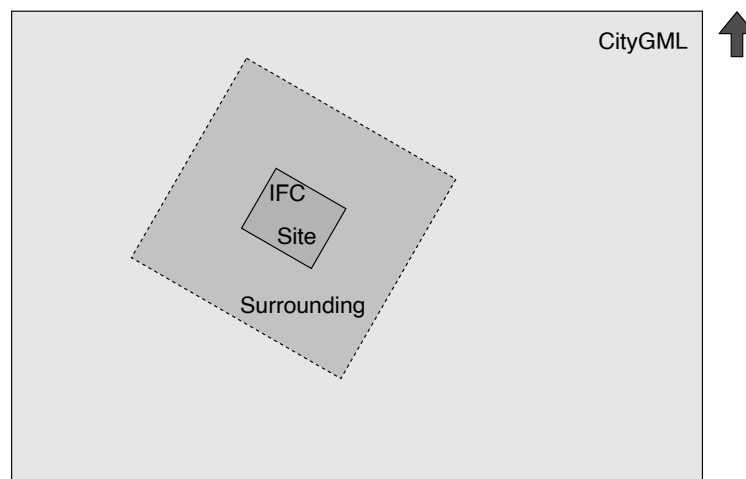


Figure 5.3: terminology in the model design

In order to generate the ENVI-met grid, the model dimensions need to be determined. The model space is shown in figure 5.4. The user needs to decide on the surrounding area by setting the border width. The border width is the space around the site in meters, as can be seen in figure 5.4a. Here, the empty grid cells can also be seen. These are a fixed number of empty grid cells that will surround the whole model. In this space, no 3D information like building or trees will be modelled, only digital elevation model and soils. This is to leave space for for example wind simulation in the ENVI-met simulation, to get better and more realistic results.

The size of the ENVI-met model in x and y directions is the size of the bounding-box of the site (IFC model) in both directions, plus twice the set border width, plus twice the grid cells. The height of the model will be twice as high as the highest building within the site and surrounding area, as can be seen in figure 5.4b. The models size can be expressen in number of grids cells/voxels in the x, y and z direction, based on the resolution set by the user. The number of cells will always be rounded up.

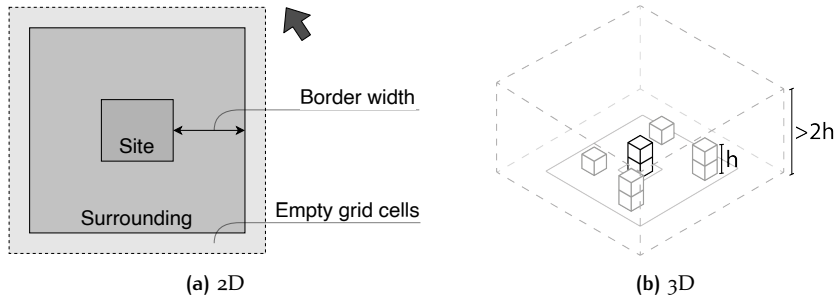


Figure 5.4: model space dimensions

The model rotation is an important factor to consider, both when combining different models and when expressing the direction of the true north.

In IFC, the true north is notated. This is expressed as a unity vector. In ENVI-met the model rotation required, which is how the model is rotated from the North, in degrees. Apart from the two being in a different format (unity vector and degrees), they are also exactly opposite of each other, as can also be seen in figure 5.5. In CityGML, rotation is not considered, since coordinates are usually expressed as real life location in existing spatial reference systems.

In this study, the rotation of the IFC model will be considered as the rotation for the output model for ENVI-met. This is done since in many cases the building in IFC will be in line with the axes of the custom coordinate system. As many straight lines as possible will give a little more accurate result when voxelizing the data, since diagonals will result in a saw pattern.

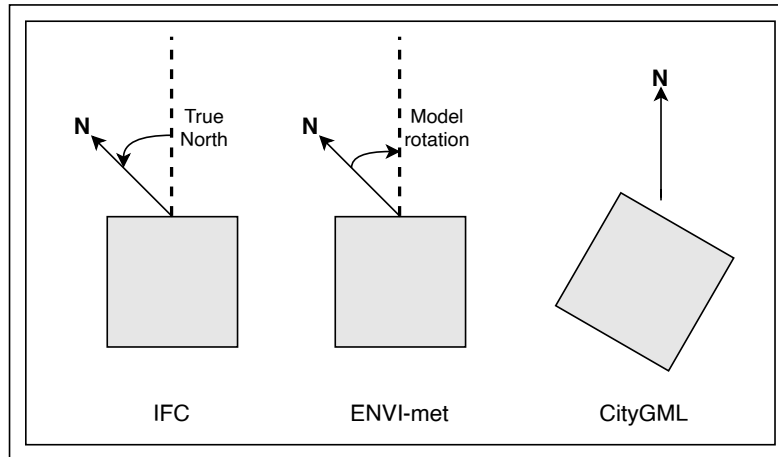


Figure 5.5: model rotation in the different types of model representation

5.1.2 Transformation

To be able to combine the different models, we need to know where the models are relative to each other. To do that, we need to be able to transform coordinates between the different spatial reference systems used in the different models. There are three different kind of transformation that can be encountered:

- Transforming a coordinate between existing spatial reference systems (EPSG)
- Transforming a coordinate between an existing spatial reference system and a custom system
- Transforming a coordinate between custom systems

For transforming between existing spatial reference systems the GDAL library is used (SpatialReference and CoordinateTransformation from osgeo.osr)¹.

An IFC model is expressed in a custom system, starting at (0, 0, 0) and axes in certain direction. To be able to link coordinates from this system to a real world location, the following information is needed:

- The coordinate to transform
- The coordinates of the reference point (the real world location of (0, 0, 0) on earth). In IFC this point is given in longitude, latitude and a height relative to level (WGS84/EPDG:4326).
- The true North
- The unit of measurement

Since the reference point in IFC is given as longitude, latitude, it means it is a point on a sphere approximating the shape of the earth. To be able to work with the data, a point on a projection is necessary. A projection system is a coordinate system representing the earth's surface by projecting it to a 2D plane (figure 5.6). In this study, the reference point will be transformed to the same projection system as used in the CityGML model via the GDAL library (for example: EPSG:28992, Amersfoort). This is done since the reference system used in the CityGML model should already be an appropriate representation (plane) for the location. Just a different rotation, origin point and scale.

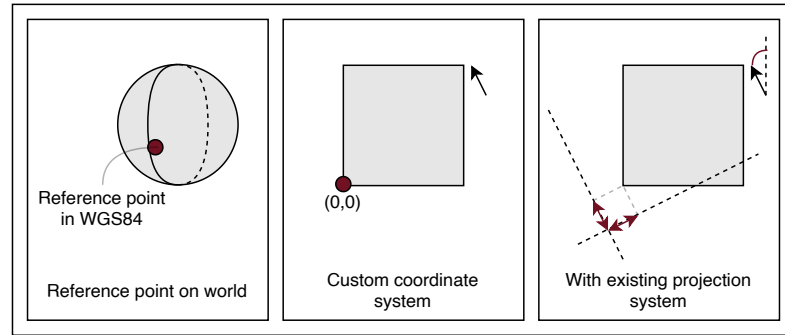


Figure 5.6: Explanation of reference point and (custom) projection systems

A transformation matrix can be applied in order to transform between the two planar systems. In figure 5.7 can be seen how the transformation between the two systems is dependent of three elements: translation, rotation and scale. The dotted lines represent the axes of the two different systems.

The transformation matrix (epsg to custom) consist of three elements:

- The rotation matrix R_z , which in this case is the rotation θ around the z-axis, based on the true North.

$$R_z = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- The translation matrix M_T , which is the distance between (0, 0) in the target system and the source system, which in this case is the distance between

¹ <https://gdal.org/python/osgeo.osr-module.html>

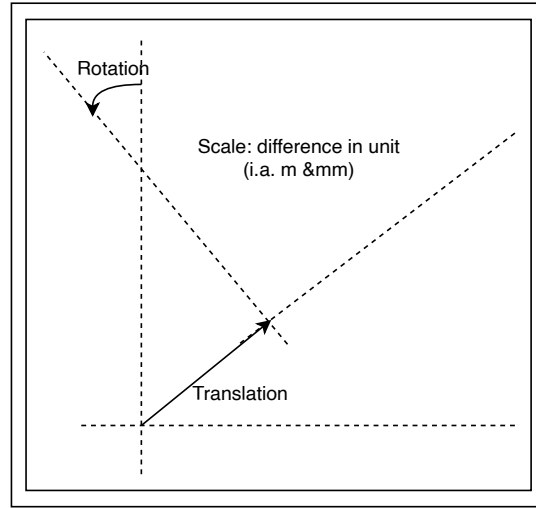


Figure 5.7: The transformation between two planar systems is dependent of three elements: translation, rotation and scale

(0, 0) in the real world coordinate system and our custom system, which is equal to the reference point available from the custom system expressed in the coordinate system of the real world coordinate system projection (from CityGML in this case).

$$M_T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- The scaling matrix M_S , containing the scaling factor in each direction. In this case, when for example the real world coordinate system is in meters, and the custom system in millimetre, the scaling factor custom to target would be 0.001.

$$M_S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

When combining the matrices it leads to transformation matrix T shown below. In this matrix θ is in radian ($\cos \theta$ and $\sin \theta$ can also be replace by the two values from the unity vector extracted from IFC). This is the transformation matrix from a custom system to an existing spatial reference system (could also be to another custom system, given you know what the coordinates of the reference point (0, 0) of the source are in the coordinate system of the target system). When the transformation the other way around is needed, the inverted matrix needs to be used.

$$T = \begin{bmatrix} 0.001 \cos \theta & -0.001 \sin \theta & 0 & R_x \\ 0.001 \sin \theta & 0.001 \cos \theta & 0 & R_y \\ 0 & 0 & 0.001 & R_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To transform a point, the dot product of the transformation matrix and the input coordinate as vector $x,y,z,1$ needs to be calculated.

5.2 MODEL C: THE DATA STRUCTURE WHERE ALL NECESSARY INFORMATION FROM DATA IS STORED

All the necessary information extracted from the input files, are temporally stored in a data structure. In this section it will be explained how this data structure is constructed and how this information will be written to an ENVI-met file in the end.

5.2.1 Model geometry and structure

The data structure will be a voxel based system. The same dimensions and coordinate system as the target model (ENVI-met area input file) will be used. The boundaries will be the same x and y coordinates. All z-information for these x and y coordinates will be stored and the z-border for the ENVI-met area input file will be based on this.

In the basis: for the elements (building, terrain, vegetation) that lie within the boundaries of the model, an object will be created, containing the information that is needed from this element. This objects can be linked to voxels in the voxel grid.

An empty VoxelGrid will be initialized. The created grid will have an empty voxel list that can be searched via a key (dictionary). To this dictionary, a voxel that contains an element can be added. The key will be de coordinate of the voxel (for example (1,2,1)). The value will be the linked element (for example a building or terrain).

Attaching an object to the voxel-grid: if an element (for example a building) is added to voxel: if voxel does not exist yet, it is created. If it does exist and does not contain the/a element yet, it is attached to the voxel. If the/a element already is associated with the voxel a consideration is made which one it should be, based on type of element and source file.

5.2.2 Element Classes

Different elements call for different information that needs to be stored. In model C different classes are available that can each facilitate unique attributes for that element. Each object in data structure C is an instance of one of the following classes:

- **Building** ... The building will only be created and added to the data structure, if it has geometry within the boundaries of the area of interest.
 - *Geometry*: The geometry of the building as boundary representation. Multiple surfaces forming an enclosed space. The coordinates of the vertices of the polygons will be in the coordinate system of the voxel system, but NOT floored to whole voxels. Some extra information might be stored for each polygon representing a building element, like material and wall thickness.
 - *bbox*: The bounding box of the building handy to have for knowing which voxel need to be checked if they lay inside the building. Voxels outside the bounding box can never be part of the building, so there is no reason to check them.
 - *building number*: A number is assigned to each building, to recognize each building (id) and to keep them separate in ENVI-met upon selection/deletion.
 - *Source file*: Which model does the information of this element come from, the IFC model or the CityGML model. This is important because in the

case that two buildings share the same space, one building from IFC and one from CityGML, the building from IFC needs to have priority.

- **Terrain**

- *Geometry/tin*: Surface model of the terrain.
- *Source file*: Which input model does the terrain information come from.

- **Vegetation**

- *Type* Simple plan (bushes/hedges) or 3D plant (trees).
- *Height* The height of the vegetation. This will be used to match it to the best matching element in the ENVI-met database
- *Source file* Which input model does the vegetation information come from.

5.2.3 From data structure C to ENVI-met model

When all necessary data is parsed and transformed to data structure C, the voxel grid is complete. The data can then be written to the area input file of ENVI-met in the EML data format. The data will be written in the format and tags as explained before in section 4.1.2 and appendix A. Most data can be extracted directly from the input files or data structure C, with only minimal formatting.

The number of voxels in the z-direction will be established at this point. Now that all data from the different input files is stored in one data structure, the maximum Z value of the highest building can be extracted. If the lowest Z-value is not a negative value, the maximum z value + 1 times two (+1 because count starts at zero) will be the number of grid cells in the z-direction in the Area Input File for ENVI-met. If the lowest Z-value is negative, all voxels will be shifted in the z-direction so that the lowest voxel will be zero.

The first time the writer asks for a list of voxels that contain a certain element, for example building, a list is created by traversing through the voxels that are registered to contain information in data structure C, and it is checked if they contain building. If so, the coordinate of that voxel is added to the list, and the building number of the corresponding building is also added to a list. This is done till it has gone through all the registered voxels. When the writer needs the list for a second or third time it can use the same list that was created earlier. These lists are then formatted for the corresponding elements.

The wall elements are derived from the building list. For each building voxel from the list, it is checked if any of the six neighbouring voxels also contains building or not (aka, is the neighbouring voxel in the list). If not, there should be wall between these two voxels and the corresponding material of the building or buildingpart is assigned to the right voxel (each voxel hosts three walls, the left face, front face and bottom face of the voxel).

5.3 CONVERSION OF NECESSARY DATA

The information extracted from the IFC and CityGML models is not always directly usable. A conversion of this data is necessary. Coordinate transformation is discussed in section 5.1.2. In the parser there is already some preselection and -ordering of the data, by extracting specific data from the input files and handing it to the conversion tool in a certain way or order. But sometimes more conversion is necessary, for example when going from boundary representation data to a voxel based data. The steps in these conversions is explained in this section.

5.3.1 Parsers

A parser is a tool that takes data string input and creates a meaningful data structure. In this case the parsing consist of two parts. In the first part the whole input file is converted to a datastructure that can easily be gone through to retrieve information. The second part is selecting, ordering and converting the data in a way that is meaningful for the conversion, so that when is asked for information about an element, it retrieves exactly the information it needs.

CityGML parser

Since CityGML is in an XML-format, an XML parser can be used to parse the data so that it can easily be traversed through. In this case the library `xml.etree.ElementTree`² is used, which orders the data in a tree structure.

For the second part an interface is created to interact with the data in the file. For example functions to get the bounding box of the model, or to extract all the buildings from the model.

IFC parser

To be able to create a data structure that can be easily traversed through from the IFC file, the library `IfcOpenShell`³ is used. The library not only creates a data structure, but already gives some interface functions to make it easier working with the data.

Since IFC models contain a lot information and a lot of small details, a selection will already be made when extracting the data. For example when extracting information about the building, only building elements noted as external elements will be extracted.

Other parsers

Another parser within the conversion tool is the knmi data parser. This is a small tool that can extract the information needed for simple forcing from 'KNMI uur data' file from the Dutch weather institute, for locations in the Netherlands. This makes collecting the information for the simulation file less cumbersome and less prone for errors.

Like this parser, also other parser could be added to the program in the future, either to collect data for the simulation file like the knmi data parser, or to use more input models for the area input file. For example a parser could be added for files that contain information about soil profiles. By creating a similar interface and adding the functionality to the conversion module, the program could this way be enriched.

5.3.2 Conversions

Most of the data that is extracted from the input files, can not be directly written to an ENVI-met file. A conversion is necessary to be able to use the data. The most prominent conversion in this research is the conversion from objects with boundary representation, like for example a building that is described as six polygons forming a box, to a voxel based system where is noted if a voxel is a building or not.

When conversing a point, for example the rootpoint of a tree, the point simply has to be converted via the transformation explained in 5.1.2. The resulting coordinate can then be floored, which results in the voxel that te point lies within.

But when going from a boundary representation to a voxel representation it becomes a little more complicated. Since the points that make the polygons are not the only places that are part of the building, more steps need to be taken then just transforming the points that represent the object. There are multiple ways to approach this problem. In this case is chosen for a approach where is checked if a

² <https://github.com/python/cpython/blob/3.9/Lib/xml/etree/ElementTree.py>

³ <http://ifcopenshell.org/>

voxel lies within the boundaries of the defined object, by defining a line fragment, and checking how many times it crosses a polygon of the building. This method explained below.

The line fragment is defined as follows:

- *the first point* is the middle of the voxel of interest. This will be $(x+0.5, y+0.5, z+0.5)$, in the voxel coordinate system. By using the middle point it is already very likely that if it turns out the point lies within the building, that the corresponding voxel contains a minimum of 50% of building. More points could be added to handle the corner cases.
- *the second point* needs to be a point that is for certain outside the building. This point will be very similar to the first point, only with the x coordinate being the x coordinate of the lower left corner of the bounding box of the building, in the voxel coordinate system.

The line fragment corresponding to each voxel are then checked how many of the polygons of the building they cross. The following method is used to check if the line crosses the polygon:

- First it is calculated where the ray through the line and the plane through the polygon intersect:
 - A ray through the line is defined. This is a point (the middle point of the voxel) and a direction $((-1,0,0)$, parallel to the x-axis).
 - The plane through the polygon is defined as a point and a normal vector.
 - First is checked if the ray and plane are parallel to each other, since they will never collide in that case anyway, so the process is aborted and can start over with the next polygon.
 - When they are not parallel, the intersection point will be calculated.
- Then it is checked if this intersection point lies within the polygon and on the line segment.
 - Since we already know the point will be on the plane and ray in 3D, we can simplify the check if it lies in the polygon and on the line segment to a 2D problem.
 - If the polygon is not parallel to the z-direction, the problem is solved with a projection to the x-y-plane.
 - If the polygon is parallel to the z-direction, the problem is solved with a projection to the y-z-plane.
 - When the point lies both in the polygon and on the line segment, the count goes one up.

When gone through all the polygons, and the count modulo 2 is equal to 1, this means the point lies within the building and the voxel can be set as building. This is done for all voxels that lie inside the bounding box of the building AND the bounding box of the surrounding area (surrounding area explained in [5.1.1](#)).

For IFC is chosen for a simpler 2.5d approach, to simplify the process. All horizontal surfaces from floors and roofs are extracted. For each voxel just outside the top of the bounding box, a ray going down is checked. It is noted when it crosses any one of the surfaces. The same is done for the voxels just outside the bottom of the bounding box, with a ray going up. The voxels inbetween these two point will be noted as building voxels.

A similar approach can be done for elevation. With a line upwards and checking when it crosses the surface model.

For converting trees from CityGML to ENVI-met, the reference point from the implicit geometry and the height are extracted from the CityGML. The reference point can be transformed and rounded to a voxel (provided that location transformation is 0). Based on the height, one of the following three ENVI-met trees is assigned to that voxel:

- *1.5 - 10 meter*: dense spherical deciduous tree, 5 meter
- *10 - 20 meter*: dense spherical deciduous tree, 15 meter
- *20+ meter*: dense spherical deciduous tree, 25 meter

Objects smaller than 1.5 meter will not be counted as trees.

The resulting product of this research is a conversion tool that will serve as a proof of concept for this research. It will show that information can be extracted from different input models and how it can lead to a working ENVI-met Area Input File, that can be used in microclimate simulation in ENVI-met. The guidelines and workflow accompanying the tool will be explained in this chapter, followed by a small case study, showing the results in practice.

6.1 GUIDELINES AND REQUIREMENTS FOR GENERATING INPUT DATA

Guidelines and requirements are set up for the IFC and CityGML input models, to make sure that they properly work with the conversion tool. For the aim of this research, a proof of concept is developed, rather than a fully functional tool. Therefore the requirements can be very specific, because not all different cases and representations are fully developed. In the future this tool could be expanded so that more cases could be handled, making the tool more versatile. The current guidelines and requirement will be elaborated on in this section, and a few small possible improvements shortly mentioned (more extensively discussed in the discussion and future work in chapter 7).

6.1.1 Guidelines for IFC model creation

The following guidelines and requirements are set up for the designer of the input IFC model:

- *Record a reference point to (0,0,0) in WGS84 in the latitude, longitude and elevation attributes of IfcSite. Also correctly note the rotation to North in the attribute TrueNorth found in RepresentationContext in IfcProject* - It is really important that the IFC model is georeferenced properly, to be able to use it in the conversion tool. Also very important is how the model is rotated with respect to the North, to be able to link the model to the real world.
- *Add some form of a terrain model to the attribute Representation in IfcSite* - The x- and y-coordinates of the bounding box of the site representation are used to decide on the size of the generated ENVI-met area input file. Therefore this needs to be defined in the model. It can be in the most simple form, for example a square.
- *Store only one building in the IFC File* - The IFC model includes one building. In case of multiple buildings they will be labelled as one building in ENVI-met.
- *Use millimetres (mm) as the unit in the IFC file* - The conversion tool only handles the file if it is in millimetres.
- *Use the building entities IfcSlab and IfcRoof for all horizontal surfaces like floors and roofs. Set the property isExternal for external floors and roofs.* - The building entities IfcSlab and IfcRoof are used to extract a building from the model.

These entities need to be properly used and assigned to the correct elements. The property `isExternal` helps knowing if an element is an external part of the building. When this property is correctly assigned, only these outside floors and roofs have to be taken into account when generating the building for ENVI-met.

6.1.2 CityGML requirements

The following classes are required to be included in the CityGML model:

- *Relief*: The Digital Elevation Model (DEM) of the CityGML is used to generate the DEM for the ENVI-met area input file.
- *Building*: The buildings from CityGML are used as surroundings/context in the ENVI-met area input file.

The next classes are nice to have in the CityGML model, and could provide for a great addition to the ENVI-met area input file (not implemented yet):

- *Transportation*: The information about roads and pavements could be used to enrich the soil model the ENVI-met area input file.
- *Vegetation*: Both information about simple plants like grass and bushes, as trees, could be extracted and added to the ENVI-met area input file.
- *Waterbody*: Could be used to enrich the soil model of the ENVI-met area input file.

At this point in time, the buildings can only be extracted from the `lod2multisurface`, so the buildings need to have this representation available. For the terrain height the TIN representation will be used. *Processing of other LODs and geometry representation could be added in the future.*

Also the elements in the CityGML model need to be in the same Spatial Reference System (SRS) as the `gml:Envelope`. And the unit in meters.

6.2 WORKFLOW OF THE CONVERSION TOOL

In this section the workflow of using the conversion tool will be described. The setup that is needed to run the conversion tool will be explained, and the process from user input to post processing will be described.

6.2.1 Setup

There is probably more than one way to set everything up, but the following setup works with all the needed libraries and dependencies.

To be able to run the program, the following is required. An Anaconda environment with python 3.8 installed and the following libraries and their dependencies: GDAL, pythonOCC-core, IfcOpenShell, shapely, lxml, reverse-geocode, time-zonefinder. Other needed modules like NumPy, xml or math will already be included in the Anaconda or Python installation.

The program can be run from a python IDE (integrated development environment) where an Anaconda (virtual) environment can be set up (like PyCharm), or can be run directly from the Conda command line, by running the file `main.py` from the conversion tool directory.

To be able to check the conversion result, edit them and run the simulation, the ENVI-met software package needs to be installed.

6.2.2 User input

The following information needs to be provided by the user, as can be seen in figure 6.1:

- *IFC input model*: Path to the input IFC model according to the guidelines.
- *CityGML input model*: Path to the input CityGML model, meeting the requirements.
- *Output file path*: Location and file name of the output file (.inx file/area input file).
- *Border width*: Size of the surrounding area around the IFC model, where information from CityGML will be used, in meters.
- *Resolution*: The resolution of the output model between 0.5 and 10 meters.
- *Border grid*: The number of empty gridcells surrounding the model

```
C:\Users\natas\anaconda3\envs\thesis-project-test\python.exe
E:/git/thesis-project-natasja/conversion_tool/main.py --ifc input/Myran2.ifc
--citygml input/floriade.gml --output output/final_reshalf.INX
--borderwidth 20.0 --resolution 0.5 --bordergrid 20
```

Figure 6.1: user input - program call with arguments

Some of the options that could be user input with a default when not defined, are now hardcoded defaults in the program, to simplify for the proof of concept. Examples of this are the telescoping grid and splitting, which are off (this could be added by the user by opening the output model in SPACES, via the model inspector tool). And for example the rotation is now always based on the rotation of the input IFC instead of user defined.

6.2.3 Conversion tool

After defining the input variables, the program runs by itself. It gives some feedback on what it is doing, as can be seen in figure 6.2. The conversion takes approximately a couple of minutes, but can take longer based on the size of the input models and the set resolution and border width. When it is done an .inx file will have been created on the output location defined by the user.

```
loading ifc [input/Myran2.ifc]...
loading citygml [input/floriade.gml]...
input files loaded
calculation parameters...
extracting buildings from input files...
converting IFC building...
converting CityGML building 18 of 18...
extracting and generating dem...
extracting and converting trees...
creating 'ENVI-met area input file'...
writing to file [output/final_reshalf.INX]...
```

Figure 6.2: feedback that the program gives while running, about what it is doing

6.2.4 Verification and editing

The created .inx file can be used directly in the ENVI-met simulation software or can be opened directly in ENVI-met SPACES for verification and editing. In this environment the model can be checked for any strange behaviour and can be edited by the user if desired.

Possible edits in SPACES that could be thought of are:

- Deleting buildings on the edge of the model, where only small parts or corners of the building have made it inside of the model, because the rest of the building lies outside the defined surrounding space. The buildings can be individually selected and deleted.
- Adding extra border grid cells if necessary, with the inspection tool of SPACES.
- Assigning materials to the building walls.
- Adding more elements like trees, infrastructure, etc.

When changes are made, the file can be saved. The file is now ready for microclimate simulation in ENVI-met.

6.3 TEST STUDY

To test the workflow and the resulting ENVI-met area input file, a small case study is set up. The case study location will be the Floriade 2022 site in Almere, in the Netherlands. Floriade is an international horticultural exposition and the theme of this project is Growing Green Cities. In figure 6.3 the masterplan of the project can be seen. In this section the process and (intermediate) results of the test study will be discussed.



Figure 6.3: Masterplan of Floriade site

6.3.1 Input models

The input models for this test study are the IFC model Myran¹ and the CityGML model Floriade (figure 6.4). The geodata of the Myran model is altered, so that it lies within the Floriade location.

Three conversions will be done with a surrounding border width of 20 meters. One with a resolution of 0.5 meter and 20 empty border grid cells, one with a resolution of 1 meter and 10 empty grid cells and one with a resolution of 2 meters and 5 empty border grid cells.

¹ Mondo arkitekter <https://mondo.se/>

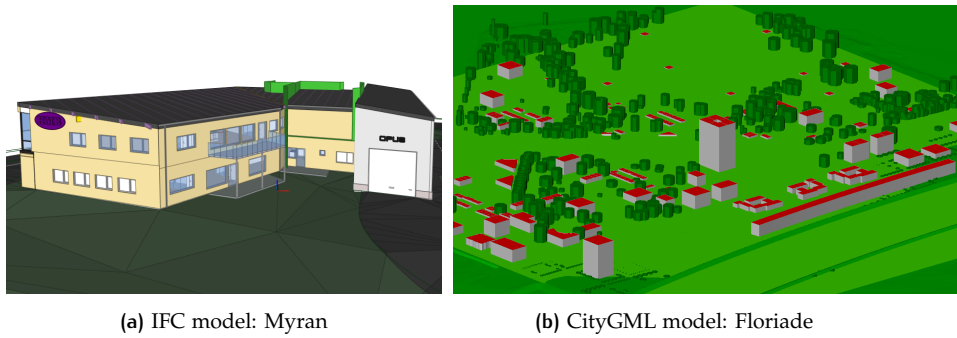


Figure 6.4: IFC and CityGML input models for conversion tool

6.3.2 Conversion results

In figure 6.5 the resulting ENVI-met area input file (.inx file) on 1 meter resolution can be seen, displayed in ENVI-met SPACES as a 2D map. When buildings with a different building number are right next to each other, this is visualized with the black line between the two buildings in SPACES. In the image can be seen that some of the building blocks contain multiple buildings.

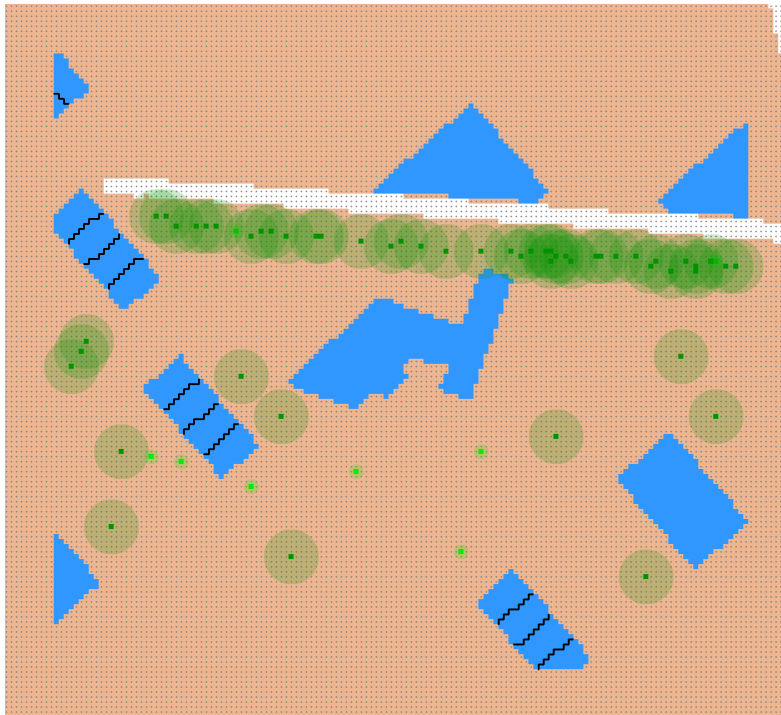


Figure 6.5: Resulting model in ENVI-met SPACES after combining and converting with the conversion tool on 0.5 meter resolution, displayed as a 2D map in ENVI-met SPACES

Figure 6.6b shows the conversion result in 3D. It can be compared to the same area in CityGML, shown in figure 6.6a. Obviously, the building in the centre cannot be found in the CityGML, since it is extracted from the IFC file. The buildings, big trees and terrain from the CityGML, are converted to the area input file, as can be seen in the images.

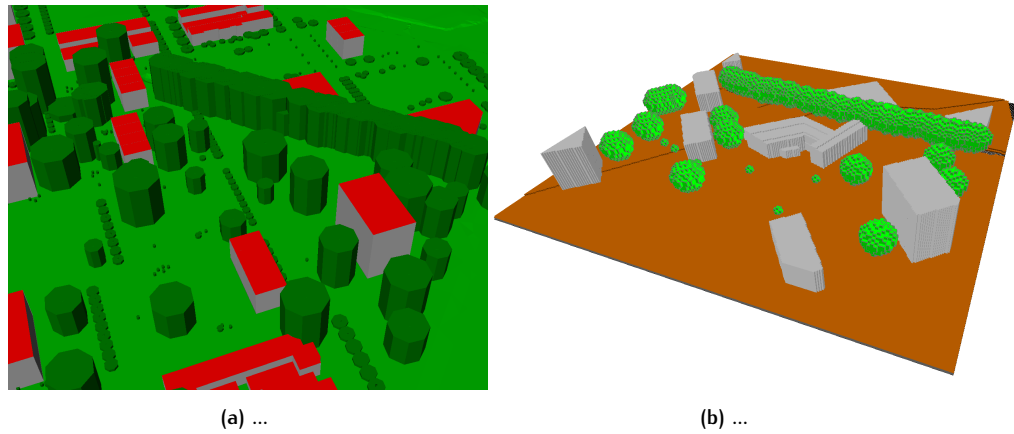


Figure 6.6: Details of resulting ENVI-met area input file displayed in ENVI-met SPACES

Figure 6.7 focusses on the converted building extracted from the IFC model. This detail is shown in three different resolution: 0.5 meter, 1.0 meter and 2.0 meter. This illustrates the effect that resolution has on the resulting building, especially for diagonal walls and sloped roofs.

In this image can also be seen what the effect is of the 2.5D approach that was used for extracting and converting the building from the IFC model. This can be seen when looking at the balcony. Because this approach only checks from the bottom and from the top when it crosses any horizontal surface, the space between the balcony and porch is not detected. Therefore, this area ends up looking more like a building extension after conversion. The 2.5D approach does work for overhangs.

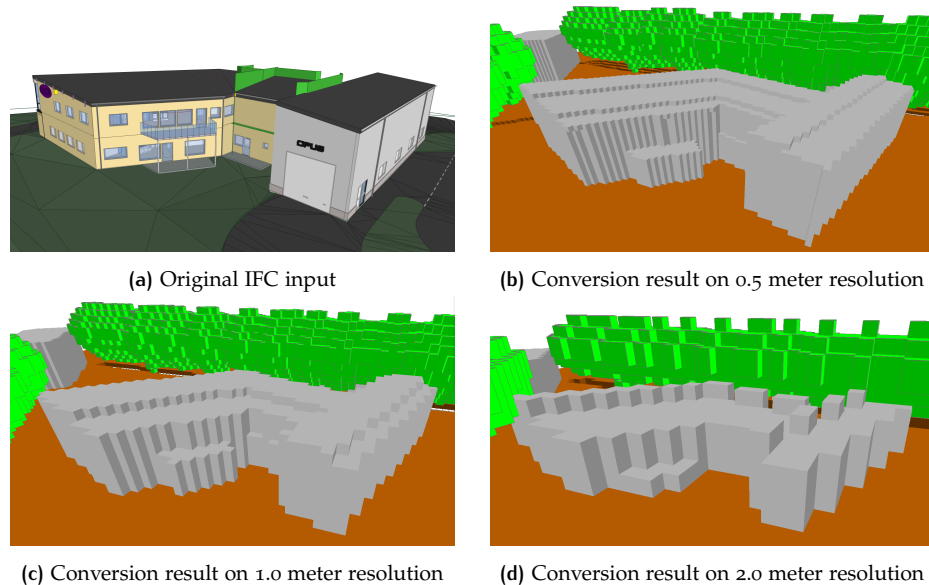


Figure 6.7: The IFC model after conversion in three different resolutions to the ENVI-met area input file displayed in ENVI-met SPACES

In figure 6.8 some details of the model are shown. In figure 6.8a can be seen how each building can be selected separately. This gives the possibility to for example delete a small cut off corner of a building that just made it into the model at the edge, but is not desirable in the simulation. Also materials and greening can be added. In figure 6.8b can be seen what the relief of the terrain looks like in the output model.

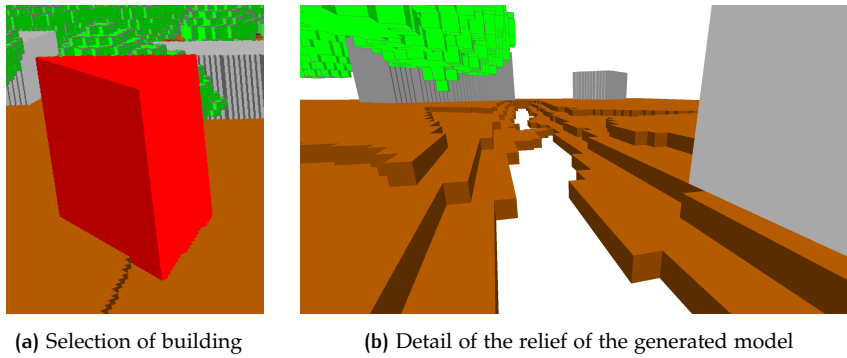


Figure 6.8: Details of resulting ENVI-met area input file displayed in ENVI-met SPACES

6.3.3 Microclimate simulation results

The resulting ENVI-met area input file will be tested by running a simulation in ENVI-met. The output file generated by the conversion tool has not been altered in SPACES but is directly used in the simulation. In order to suppress the time that the simulation takes, the area input file with a resolution of 2 meters, is used for the simulation. Other input files of this simulation are a simulation file, generated via ENVI-guide, and an ENVI-met database file. The simulation date is the 10th of August, 2019. The microclimate simulation in ENVI-met with the generated area input file was successful. Some images of the simulation results in ENVI-mets' simulation visualisation tool LEONARDO are shown below.

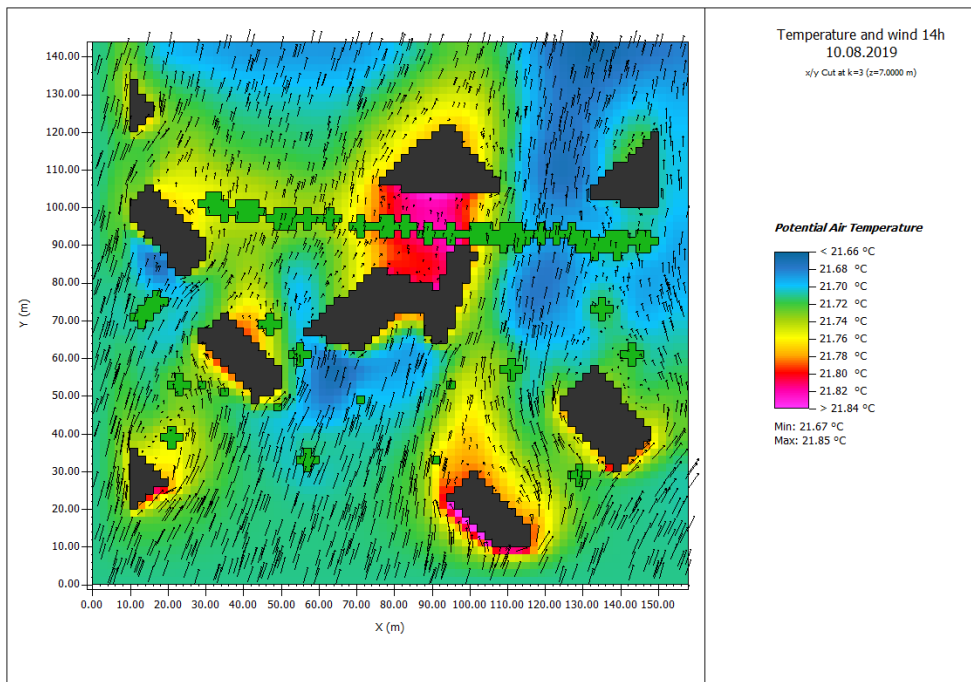


Figure 6.9: Microclimate simulation results for air temperature and wind visualised in ENVI-met LEONARDO in 2D. The colours display the potential air temperature at that location. The arrows display wind direction and speed. The longer the arrow, the higher the wind speed.

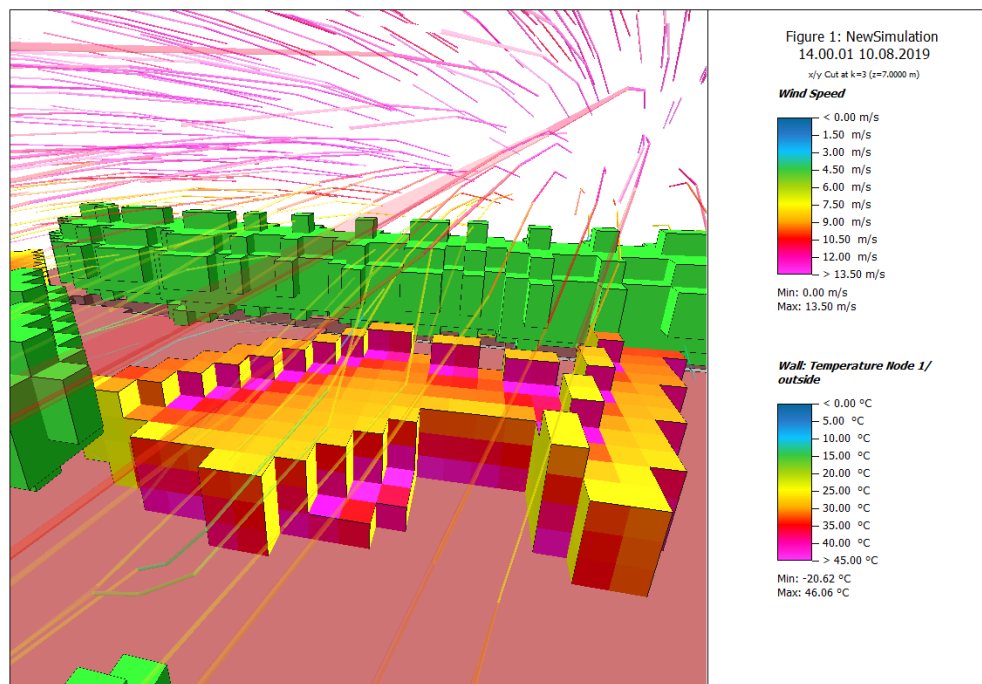


Figure 6.10: Microclimate simulation results for facade temperature and wind visualised in ENVI-met LEONARDO in 3D. The colours on the buildings represent the facade temperature. The tubes represent wind paths and their colour the wind speed.

In this thesis, a method is developed for using existing IFC and CityGML models in microclimate simulation software ENVI-met. This allows designers to combine their design in BIM (IFC) with a 3D city model (CityGML), and run reliable microclimate simulations of their design. The research question was: How can IFC and CityGML models be used as input for microclimate simulation software ENVI-met. The research resulted in the findings discussed in this chapter.

It was found that extracting the necessary data from both the input files, combining and converting it, and then writing it to a file in the ENVI-met format, would be an effective method to achieve the goal.

To do that, the ENVI-met simulation input files and their file structure were studied. To run a simulation in ENVI-met, the following files are needed: area input file (.inx), simulation file (.simx) and database file (.edb). The information about what elements can be found where is stored in the area input file. This is the file where the information from the IFC and CityGML have to be written to.

To generate the area input file, the information that is needed for this file needs to be known and the availability of this information in the anticipated input files needs to be checked. The following information is needed for the ENVI-met area input file: building geometry and material, vegetation/trees, terrain height and soil profile.

Most of this information can be found in the IFC and CityGML models. Detailed information about geometry and material of a building can be found in the IFC model. Sometimes the model also contains information about vegetation and terrain height. In CityGML models, information about building geometry, vegetation, terrain height can be found. Also, information about infrastructure and waterbodies can help for the soil model.

To be able to use the input models properly in the conversion, some characteristics are very important.

First, it is important that the data is properly georeferenced, otherwise the data can't be linked to the real world and combined with other data. This is especially important for the IFC files, since from practice can be learned that this is not always done correctly.

For IFC files the entities *IfcWall*, *IfcSlab* and *IfcRoof* need to be used correctly, together with the property *isExternal*. In the CityGML the classes *Building* and *Relief* need to be implemented. The classes *Vegetation*, *Waterbody* and *Transportation* are a nice addition.

To convert and combine IFC and CityGML information effectively into the ENVI-met format, a conversion tool is developed in Python. This tool extracts the required information from the input files and transforms the data to the same coordinate system via coordinate transformation tools and transformation matrices. The data is then combined and stored in an intermediate voxel based data structure. When all information is collected, it is written to an ENVI-met area input file in the ENVI-met format.

In conclusion, IFC and CityGML models can be used as input for microclimate simulation software ENVI-met. This can be done by using a conversion tool, developed for this research. This tool can extract the necessary information from the input models, combines the data and converts it to the right format, and write it to an ENVI-met area input file, so that ENVI-met can understand the data.

This way, designers can automatically convert their existing models (instead of manually drawing them in the software) and use them in the ENVI-met environment for microclimate simulations. Microclimate simulations are important to be able to test the performance of a building (heating and cooling of the building) with realistic numbers, instead of general weather data. Also the effect on the urban environment can be tested with microclimate simulation. With a conversion tool for using existing models, the process of using microclimate simulation for designers will be simplified, so that more realistic data can be used in sustainable design.

Discussion

In the development of the conversion tool, some shortcuts are taken, in order to simplify the process and save production time. That is why the conversion tool is referred to as a proof of concept. It shows that the concept of a tool that extracts information from IFC and CityGML, combining, converting and outputting it in the ENVI-met format, could work. Only some of the details are simplified or not implemented completely.

For example not all useful information that can be found in the input files, is now extracted and used in the conversion tool. For example information about infrastructure and waterbodies from CityGML and details about walls and materials from IFC. These elements could be added with a similar approach as some of the other elements that are implemented.

Also some of the simplification could be reversed and further developed. One of these simplifications is the 2.5D approach for converting the building from the IFC model to a voxelized model. The 2.5D approach gives a good approximation of the building, easier to implement and quick in computation time because many of the geometry in IFC can be ignored. However, a full 3D approach would give a more accurate result. For example, in the results of chapter 6 can be seen how the balcony of the IFC building ended up looking like a building extension, because the program only looks from the top and the bottom when it crosses any horizontal surface. This would not have happened in a full 3D approach.

For the elements where only one of the more common cases is handled, more cases could be added. Also the ability to process more different input models and types would be a good addition. The possibilities for this are further discussed in the future work section.

In this research and proof of concept for the tool, the open standard CityGML is used as input 3D city model. Another 3D city model format is CityJSON. The information stored in this data structure is very similar to CityGML, only stored differently, without some of the complexity and verbosity around it. This makes the data easier to parse and manipulate. However, it is not an official OGC standard, which makes for less availability of models in this format. It is however easy to convert CityGML to CityJSON, since it is a lot of the same information, just written down in a different format. But this would make for an extra step, that might be out of the comfort zone of some of the intended user groups.

In this project it is only needed to extract information from the CityGML, and this is not that difficult with an XML-parser, especially when you compare it with extracting information from IFC, which has proven to be a lot more complex. Therefore is chosen for CityGML as input in this prove of concept, since it is a commonly used open standard and extracting information from it is still fairly reasonable. A parser for CityJSON or other similar models could be added easily, since the infor-

mation is very similar, it is only stored in a different format.

Some things do not work exactly as expected. When the number of border grid cells or resolution changes, the IFC model and CityGML seem to slightly shift relative to each other, more than you should expect from rounding due to low resolution. The building is still roughly where you would expect it, just slightly shifted. This is probably due to a small bug somewhere in the conversion tool program. Most likely a missing multiplication or division with the resolution, when it comes to the border grid, or a wrong bounding box or corner point used (full model or surrounding bounding box) at one point. It is probably very easy bug to fix, once found, but it has not been detected yet where it is located.

One last thing that could have been better looking back, is the way the empty border grid cells are handled. It was realised very early in the research that the presence of these, would improve simulation results significantly, because unrealistic scenarios with i.a. wind would be simulated when a building is right next to the border of the model. Therefore was chosen to add the number of border grid cells as user input for the program, so that the user can estimate how many grid cells they consider necessary. However, this number could also be decided upon automatically by the program, since ideally the distance between a building and the edge of the model is the height of that building. And this can be calculated based on the data structure C that is created. However, in order to make this change, some things in the basis of the program would have to be changed, since the bounding box of the model is calculated right at the start of the program, and many things like referencing and the data structure are depending on this. So in order to implement this, the core code of the program might have to change. Also an additional part of the code could be added, that adds the extra voxels to the model at the end when it is necessary, and converts everything that depends on this accordingly. The same shifting system could be used for the minimum z value of the model, if a lower points needs to be represented, then originally calculated. These features could be added in future work.

The approach developed in this research shows the potential of automatic conversion of existing 3D models to allow microclimate simulations in ENVI-met, making microclimate simulations more accessible for designers and urbanist and therefore valuable in the social context. Since the conversion tool is only a proof of concept in this stage, it still needs some development and improvement, before it can be properly used in the field.

Future work

During the research some new ideas arose, that could be interesting for future research. These will be summarised underneath. Also possible addition that are now not part of the proof of concept for the conversion tool will be mentioned.

- *Effect of different data features on simulation results* - An interesting research could be how different data features effect the results of the simulation. For example how much does a higher resolution of the ENVI-met model or higher Level of Detail of the input CityGML model, improve the results. How do the results change and which steps cause significant improvement/differences. At what point starts increasing these, not outweighing computation time and complexity any more.
- *2.5D approach buildings IFC could be 3D* - A full 3D approach for converting the IFC building to ENVI-met was discontinued, because time was running out. Converting the IFC building was simplified via a 2.5D approach, to be able to finish it in time. It would be very interesting to work out a 3D approach, for more accurate results.

- *Extracting infrastructure and waterbodies from CityGML* - From the CityGML, information about infrastructure and waterbodies could be extracted, and used to create a better soil model.
- *Extracting more information from IFC models* - IFC models contain a lot of information that could be used, but is not used in the proof of concept of the conversion tool. For example information about materials and wall thickness. These could be extracted from IFC, linked to items in the ENVI-met database, and converted and coupled to the right voxels in the conversion tool. Also some information like elevation and vegetation information from the site could be used when it is available in the IFC model.
- *Addition of simplified material approach* - From IFC information about materials can be extracted, like described in the last point. A simplified approach could be assigning three different materials for walls, roofs and windows. If a geometry is a wall, roof or window can be derived directly from the entity (IfcWall, IfcRoof, IfcWindow). The same information is also available in CityGML (wall/roof surface from LOD2 and windows from LOD3). This way an interesting material like glass can be added in the simulation relatively simply. Even from CityGML where information about materials is generally not present. The three materials could be default that can be changed by the user.
- *Handling more cases for the elements that are implemented in the proof of concept* - In the proof of concept of the conversion tool, often only one of the more common representations is worked out. For example only lod2multisurface in CityGML for buildings. To make the tool more versatile, more cases with for example different lod or geometry could be handled.
- *Adding a third input for elements that IFC and CityGML are lacking* - Some information is generally not found in the existing 3D models. For example information about soil profiles. In CityGML and IFC sometimes information about the top layer can be found, for example a road. But no generally no data is available about the layers underneath, and the exact type of soil. An interesting little project could be finding where this information can be found, parsing the information and linking it to the ENVI-met database, so that it could be added to conversion tool. This could be done for soil data, but also for other information that the input models are lacking, but would be interesting to add in the ENVI-met simulation.
- *Adding parsers for other 3d model standards, to make tool more versatile* - The conversion tool now handles two input models: IFC for the detailed information of the area of interest, and CityGML for the information about the surroundings. Since the tool is "opgezet" mostly modular, parsers for other formats input models then IFC and CityGML could be added. Also, the program now expects one IFC file and one CityGML. The program could be tweaked so that it takes variations of this, for example multiple IFC files, or only one of the two input files.
- *More possibilities for user input* - In the proof of concept of the conversion tool some of ENVI-met functionality is not available, in order to simplify the tool. For example telescoping grid and splitting. This functionality could be added, so that it could be turned on if the user wishes so. Also making it possible for the user to be able to set some other things that are now defaults. As well as the actual defaults that can be set by the user, for example building material or soil, could be a simple but nice addition.
- *Improve voxel in geometry approach* - To check if a voxel lies within a closed 3D geometry, it is checked how many times a line from the middle of the voxel

crossed a surface. This approach will work in most cases, since in the most straightforward cases, when the middle lies within the shape, it most of the time means more than half the voxel lies within the shape. However in some corner cases, this could lead to unwanted results. The approach could be optimized by making a consideration with multiple points within the voxel.

- *automatic detection of number needed empty grid cells* - In the proof of concept, the user has to set the number of empty grid cells for around the model, themselves. However this could be done automatically by the program, based on the height of the buildings on the outer grid cells. The space between a building and the edge of the model is ideally at least the height of that building.
- *Improve user usability* - In order to improve usability, the tool should be a simple executable with a graphical user interface, in the end. Also a tool could be added for automating the generation of the simulation file, or maybe automatically extract the necessary data for full forcing from weather files. Within the conversion tool there already is a small program that can extract 24 hour temperature and humidity data for simple forcing from Dutch weather data file from KNMI, but this could also be improved and extended upon.

BIBLIOGRAPHY

- Arapakis, T. (2019). The use of digital models in microclimatic studies : First steps in coupling CityGML with ENVI-met.
- Arroyo Ohori, K., Biljecki, F., Diakit , A., Krijnen, T. F., Ledoux, H., and Stoter, J. (2017). Towards an integration of GIS and BIM data : what are the geometric and topological issues? In *12th 3D Geoinfo Conference, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, ISPRS, 26-27 October, Melbourne, Australia*.
- Arroyo Ohori, K., Biljecki, F., Kumar, K., Ledoux, H., and Stoter, J. (2018a). Modeling Cities and Landscapes in 3D with CityGML. pages 199–215. Springer International Publishing, Cham.
- Arroyo Ohori, K., Diakit , A., Ledoux, H., Stoter, J., and Krijnen, T. (2018b). GeoBIM project. resreport, Delft University of Technology and Eindhoven University of Technology.
- Azhar, S. (2011). Building Information Modeling (BIM): Trends, Benefits, Risks, and Challenges for the AEC Industry. *Leadership and Management in Engineering*, 11:241–252.
- Benner, J. (2018). *CityGML Energy ADE V.1.0 Specification*. Karlsruhe Institute of Technology.
- Biljecki, F., Ledoux, H., and Stoter, J. (2016). An improved LOD specification for 3d building models. *Computers, Environment and Urban Systems*, 59:25–37.
- Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., and   ltekin, A. (2015). Applications of 3d City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889.
- Bornstein, R. D. (1968). Observations of the urban heat island effect in New York City. *Journal of Applied Meteorology and Climatology*, 7(4):575–582.
- Chatzidimitriou, A. and Yannas, S. (2016). Microclimate design for open spaces: Ranking urban design effects on pedestrian thermal comfort in summer. *Sustainable Cities and Society*, 26:27–47.
- Cheng, J. C. P., Deng, Y., Das, M., and Anumba, C. (2014). Evaluation of IFC4 for the GIS and Green Building Domains.
- de Laat, R. and van Berlo, L. (2011). Integration of BIM and GIS: The Development of the CityGML GeoBIM Extension. In Kolbe, T. H., K nig, G., and Nagel, C., editors, *Lecture Notes in Geoinformation and Cartography*, Lecture Notes in Geoinformation and Cartography, pages 211–225. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Debbage, N. and Shepherd, J. M. (2015). The urban heat island effect and city contiguity. *Computers, Environment and Urban Systems*, 54:181–194.
- Di Napoli, C., Hogan, R. J., and Pappenberger, F. (2020). Mean radiant temperature from global-scale numerical weather prediction models. *International Journal of Biometeorology*, 64(7):1233–1245.
- ENVI-met GmbH (2017). The new EML fileformat. <https://envi-met.info/doku.php?id=filereference:fileformat>. Accessed: 2020-10-28.

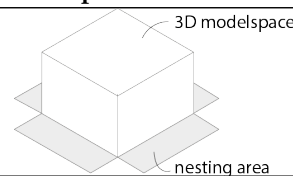
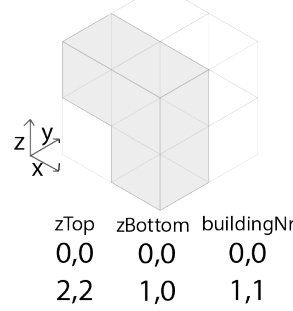
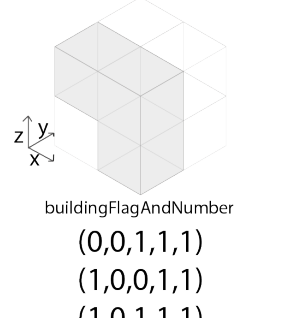
- ENVI-met GmbH (2019). ENVI-met 4. A holistic Microclimate Modelling System. <http://envi-met.net/hg2e/doku.php?id=root:start>. Accessed: 2019-04-24.
- Gröger, G., Kolbe, T. H., Nagel, C., and Häfele, K.-H. (2012). OGC City Geography Markup Language (CityGML) Encoding Standard. Technical report, Open Geospatial Consortium.
- Gröger, G. and Plümer, L. (2012). CityGML – Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 71:12–33.
- Huttner, S. (2012). Further development and application of the 3D microclimate simulation ENVI-met. *Johannes Gutenberg-Universität, Mainz*.
- Höppe, P. (1999). The physiological equivalent temperature – a universal index for the biometeorological assessment of the thermal environment. *International Journal of Biometeorology*, 43(2):71–75.
- Kardinal Jusuf, S., Mousseau, B., Godfroid, G., and Soh Jin Hui, V. (2017). Integrated modeling of CityGML and IFC for city/neighborhood development for urban microclimates analysis. *Energy Procedia*, 122:145–150.
- Kim, H., Shen, Z., Kim, I., Kim, K., Stumpf, A., and Yu, J. (2016). BIM IFC information mapping to building energy analysis (BEA) model with manually extended material information. *Automation in Construction*, 68:183–193.
- Krygiel, E. and Nies, B. (2008). *Green BIM: Successful Sustainable Design with Building Information Modeling*. John Wiley & Sons. Google-Books-ID: 2Nia6A4GouQC.
- Kántor, N. and Unger, J. (2011). The most problematic variable in the course of human-biometeorological comfort assessment — the mean radiant temperature. *Central European Journal of Geosciences*, 3(1):90–100.
- Liebich, T. (2013). IFC4 – the new buildingSMART Standard.
- Ma, Z. and Ren, Y. (2017). Integrated Application of BIM and GIS: An Overview. *Procedia Engineering*, 196:1072–1079.
- Nouvel, R., Bahu, J.-M., Kaden, R., Kaempf, J., Cipriano, P., Lauster, M., Häfele, K.-H., Munoz, E., Tournaire, O., and Casper, E. (2015). Development of the CityGML Application Domain Extension Energy for Urban Energy Simulation. In *Building Simulation 2015 - 14th Conference of the International Building Performance Simulation Association*, pages 559–564, Hyderabad, India.
- Oke, T. R., Mills, G., Christen, A., and Voogt, J. A. (2017). *Urban Climates*. Cambridge University Press.
- Robitu, M., Musy, M., Inard, C., and Groleau, D. (2006). Modeling the influence of vegetation and water pond on urban microclimate. *Solar Energy*, 80(4):435–447.
- Salata, F., Golasi, I., de Lieto Vollaro, R., and de Lieto Vollaro, A. (2016). Urban microclimate and outdoor thermal comfort. A proper procedure to fit ENVI-met simulation outputs to experimental data. *Sustainable Cities and Society*, 26:318–343.
- Santamouris, M., Papanikolaou, N., Livada, I., Koronakis, I., Georgakis, C., Argiroiu, A., and Assimakopoulos, D. N. (2001). On the impact of urban climate on the energy consumption of buildings. *Solar Energy*, 70(3):201–216.
- Yang, L., Qian, F., Song, D.-X., and Zheng, K.-J. (2016). Research on Urban Heat-Island Effect. *Procedia Engineering*, 169:11–18.
- Zhang, Y. and Gu, Z. (2013). Air quality by urban design. *Nature Geoscience*, 6(7):506–506.

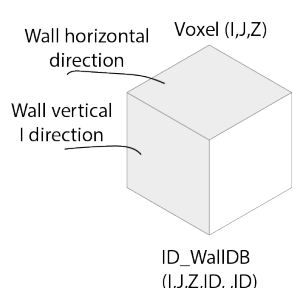
Zhu, J., Wang, X., Wang, P., Wu, Z., and Kim, M. J. (2019). Integration of BIM and GIS: Geometry from IFC to shapefile using open-source technology. *Automation in Construction*, 102:105–119.

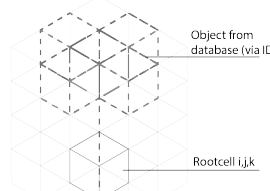
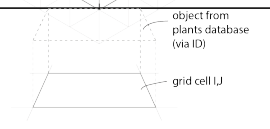
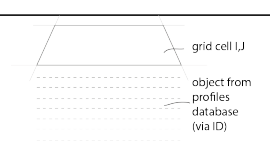

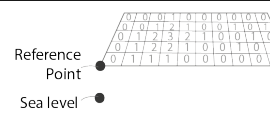


DATA SHEET ENVI-MET AREA INPUT FILE

Element	Element Description	Attributes	Attribute Description	Type	Unit	Example
modelGeometry	Information about the model space, like the size and the grid	grids-I, grids-J, grids-Z	Number of grid cells in each direction	Int	-	<p>The diagram shows a 3D grid box with dimensions dx, dy, and dz along the $grids-I$, $grids-J$, and $grids-Z$ axes respectively. Below the 3D box, two vertical stacks of boxes illustrate different vertical discretization methods. The left stack, labeled 'splitting', shows a single dz box divided into five smaller boxes, each labeled $0.2 * dz$. The right stack, labeled 'telescoping (stretch factor s)', shows a series of boxes where the height increases with altitude. The bottom box is labeled dz, and the boxes above it are labeled $dz(3) = dz + s * dz(3)$ and $dz(2) = dz + s * dz$. A 'start stretch' point is indicated between the bottom and middle boxes.</p>
		dx , dy , dz -base	The size of one grid cell in each direction	Float	meter	
		useSplitting	If the dz of lowest gridbox is divided in 5 subcells, to create more detail in the processes near the ground level	Bool	-	
		useTelescoping_grid	If dz increases with height. Can be used when model contains higher buildings, but the processes at upper parts don't require the same level of detail	Bool	-	
		verticalStretch	The telescoping factor. The factor with which the height of each grid cell grows	Float	%	
		startStretch	Height where telescoping starts	Float	meter	
		has3DModel, isFull3DDesign	Two attributes concerning if it has 3D, and if its fully 3D	Bool	-	
locationData	Information about the real world location of the model	modelRotation	Model rotation out of grid North	Float	degrees	<p>The diagram shows a 2D grid with a North arrow pointing towards the top-left. A point is marked at the bottom-left corner of the grid, labeled $realworldLowerLeft_X$ and $realworldLowerLeft_Y$. The grid is labeled $modelRotation$ at the top.</p>
		projectionSystem	The reference system	?		
		realworldLowerLeft_X, realworldLowerleft_Y	Two attributes giving the coordinates of the lower left corner of the model	Float	-	
		location_Longitude, location_Latitude	The longitude and latitude of the location	Float	degrees	
		locationTimeZone_Name	The name of the timezone	String	-	
		locationTimeZone_longitude	The longitude of the reference time-zone	Float	degrees	

Element	Element Description	Attributes	Attribute Description	Type	Unit	Example
nestingArea	Grid cells surrounding core of 3D model, the further from the core, the bigger their size	numberNestinggrids	Number of gridcells moving out-ward from 3D model	Int	-	
		soiProfileA, soiProfileB	Two soil profiles can be chosen (same or different types), which will be in a chessboard pattern	IDs	-	
buildings2D	Information about buildings, described via 2.5D matrices	zTop	I,J matrix, with at each point the height of the top of the building	Int	meter	
		zBottom	I,J matrix with at each point the height of the bottom of the building	Int	meter	
		buildingNr	I,J matrix with at each point the building number. When no building is present, the value is 0	Int	-	
		fixedheight	I,J matrix with at each point if the height is absolute (meaning not fol-lowing terrain). Otherwise height are added to terrain heights	Bool	-	
buildings3D	Information about buildings, described via points in a 3D matrix	buildingFlagAndNumber	Attribute with information about if a voxel contains building and the building number. Only the voxels containing building have to be noted. Separate elements explained below.	(I,J,Z,f,nr)	-	
		- I, J, Z (grid location)	The location in the 3D matrix (I,J,Z)	Int	-	
		- Building flag	Building flag (f), if voxel contains building	Bool	-	
		- Building number	The number given to the building (nr)	Int	-	

Element	Element Description	Attributes	Attribute Description	Type	Unit	Example
WallDB	Information about walls and roofs with multiple different layers of material, described via points in a 3D matrix. Used for buildings	ID_wallDB	Contains information about the walls used for walls and roof. Only voxels containing wall have to be noted	(I,J,Z, ID,ID,ID)	-	
		- I,J,Z (grid location)	The location in the 3D matrix (I,J,Z)	Ints	-	
		- ID wall vertical I dir.	References to an object in the 'wall database', for the vertical I direction of the voxel	ID	-	
		- ID wall vertical J dir.	References to an object in the 'wall database', for the vertical J direction of the voxel	ID	-	
		- ID wall horizontal direction	References to an object in the 'wall database', for the horizontal direction of the voxel	ID	-	
SingleWallDB	Information about walls and roofs with a single layer of material, described via points in a 3D matrix. Used for (semi) enclosed spaces that do not behave like buildings	ID_singlewallDB	. Only voxels containing single wall are noted	(I,J,Z, ID,ID,ID)	-	
		- I,J,Z (grid location)	The location in the 3D matrix (I,J,Z)	Ints	-	
		- 3x ID single wall for every direction	Three IDs that each reference to an object in the 'single wall database' , for every direction of the voxel	IDs	-	
GreeningDB	Information about walls and roofs that have greening added to a wall described via points in a 3D matrix	ID_greeningDB	. , and. Only voxels containing greening are noted	(I,J,Z, ID,ID,ID)	-	
		- I,J,Z (grid location)	The location in the 3D matrix (I,J,Z)	Ints	-	
		- 3x ID greening for every direction	Three IDs that each reference to an object in the 'greening database' , for every direction of the voxel	IDs	-	

Element	Element Description	Attributes	Attribute Description	Type	Unit	Example
3Dplants	Information about plants which have to be described in 3D, like trees	rootcell_i, rootcell_j, rootcell_k	The root location of the plant	Int	-	
		plantID	An ID that references to an object in the 'plant3D database'	ID	-	
		name	The name of the plant	String	-	
		observe	If marker is set	Bool	-	
simpleplants2D	Information about plants that can be described in 2.5D, like grass or hedges	ID_plants1D	I,J matrix, with at each point that contains plants, an ID that references to the object in the 'plants database'	IDs	-	
soils2D	Information about the soil profiles, including infrastructure and water bodies, in a 2D matrix	ID_soilprofile	I,J matrix, with at each point, an ID that references to the object in the 'profiles database'	IDs	-	
sources2D	Information about pollution sources, described in a 2D matrix	ID_sources	I,J matrix, with at each point that contains pollution sources, an ID that references to the object in the 'sources database'	IDs	-	
dem	Information about the terrain height in 2.5D	DEMReference	Elevation of reference point above sea level	Float	meter	
		terrainheight	I,J matrix, with at each point the terrain height relative to the reference height	Int	meter	
dem3D	Information about the terrain height, described via points in a 3D matrix	terrainflag	The location in the 3D matrix (I,J,Z), with terrain flag (f), where the terrain is elevated	(I,J,Z,f) Ints and float	-	

B | DATA SHEET ENVI-MET DATABASE FILE

DB Table	Attribute	Attribute Description	Type	Unit
Profiles	zo.Length	Microscale roughness length of surface	Float	meter
	soilprofil	Layers of soils via reference ID to soils database	IDs	-
	Albedo	Reflectivity for shortwave radiation	Float	Fraction
	Emissivität	Emissivity for longwave thermal radiation	Float	Fraction
	Irrigated	If the surface is irrigated	Bool	-
Soils	versiegelung	Type of material (0:natural, 1: artificial or 2: water)	Int	-
	ns	Volumetric water content at saturation point	Float	m ₃ (water)/m ₃ (soil)
	nfc	Volumetric water content at field capacity	Float	m ₃ (water)/m ₃ (soil)
	nwilt	Volumetric water content at wilting point	Float	m ₃ (water)/m ₃ (soil)
	matpot	Matrix potential at water saturation	Float	meter
	hydro.lf	Hydraulic conductivity at water saturation, how easily water can pass through	Float	m/s*10 ⁻⁶
	volumenw	Volumetric heat capacity	Float	J/(m ³ K) *10 ⁶
	b	Clapp & Hornberger constant b	Float	-
	waerme.lf	Heat conductivity of material	Float	W/(mK)
Materials	DefaultThickness	Default thickness of this material	Float	meter
	Absorption	Fraction of shortwave radiation absorbed by the material	Float	fraction
	Transmission	Fraction of shortwave radiation transmitted through the material	Float	fraction
	Reflection	Fraction of shortwave radiation reflected by the material (Albedo)	Float	fraction
	Emissivity	Emissivity for longwave thermal radiation	Float	fraction
	SpecificHeat	Specific Heat of the material	Float	J/(kg*K)
	ThermalConductivity	Thermal conductivity through the material on molecular basis	Float	W/(m*K)
	Density	Density of material	Float	Kg/m ₃
Wall	Materials	Layers of materials (outer, center and inner) via reference ID to materials database.	IDs	-
	ThicknessLayers	Thickness of each layer	Float	meter
	TypeID	Possible usage (0: Wall or Roof, 1: roof only or 2: wall only)	Int	-
	RoughnessLength	Aerodynamic Roughness of outer wall/roof layer	Float	meter
	CanBeGreened	If greening can be added to roof/wall	Bool	-

DB Table	Attribute	Attribute Description	Type	Unit
Single Wall	Material	Material of wall via materials database	ID	-
	RoughnessLength	Aerodynamic Roughness Length of wall surface	Float	meter
	Thickness	Thickness of wall	Float	meter
Plants	Planttype	CO2 fixation type (0: C3 or 1: C4)	Int	-
	Leaftype	Leaf type (0: grass, 1: deciduous or 2: conifer)	Int	-
	Albedo	Leaf albedo to shortwave radiation	Float	fraction
	Transmittance	Transmittance factor of leaf for shortwave radiation	Float	fraction
	rs_min	Minimum stomatal resistance	Float	\sqrt{s}/m
	Height	Height of the plant	Float	meter
	Depth	Depth of plant root zone	Float	meter
	LAD-Profile	Vertical LAD (leaf area density) profile (list of 10, 1D)	Floats	m ² /m ³
	RAD-Profile	Vertical RAD (root area density) profile (list of 10, 1D)	Floats	m ² /m ³
	Season-Profile	Dynamical growing factor of LAD per month (not implemented yet)	Floats	fraction
Sources	DefaultHeight	Default height for placing the source	Float	meter
	Sourcetype	Source geometry (0: point, 1: line or 2: area)	Int	-
	Emissionprofile_User	Emission rates user defined species for 24 hours of the day.	Floats	$\mu g/s$, $\mu g/(s * m)$, $\mu g/(s * m^2)$
	Emissionprofile_NO	Emission rates NO for 24 hours of the day	Floats	$\mu g/s$, $\mu g/(s * m)$, $\mu g/(s * m^2)$
	Emissionprofile_NO2	Emission rates NO2 for 24 hours of the day	Floats	$\mu g/s$, $\mu g/(s * m)$, $\mu g/(s * m^2)$
	Emissionprofile_O3	Emission rates ozone for 24 hours of the day	Floats	$\mu g/s$, $\mu g/(s * m)$, $\mu g/(s * m^2)$
	Emissionprofile_PM10	Emission rates particulate matter (10 μm class) for 24 hours of the day	Floats	$\mu g/s$, $\mu g/(s * m)$, $\mu g/(s * m^2)$
	Emissionprofile_PM25	Emission rates particulate matter (2.5 μm class) for 24 hours of the day	Floats	$\mu g/s$, $\mu g/(s * m)$, $\mu g/(s * m^2)$

DB Table	Attribute	Attribute Description	Type	Unit
Greening	HasSubstrate			
	SoilID			
	ThicknessLayers			
	subEmissivity			
	subAlbedo			
	subWaterCoeff			
	SimplePlantID			
	LAI			
	SimplePlantThickness			
	LeafAngleDistibution			
	Airgap			
Plant3D	Planttype			
	Leaftype			
	Albedo			
	Transmittance			
	isoprene			
	leafweigth			
	rs_min			
	Height, Width, Depth			
	RootDiameter			
	cellsize			
	xy_cells, z_cells			
	LAD-Profile			
	RAD-Profile			
	Root-Range-Profile			
	Season-Profile			
	PlantGroup			

COLOPHON

This document was typeset using L^AT_EX. The document layout was generated using the `arsclassica` package by Lorenzo Pantieri, which is an adaption of the original `classicthesis` package from André Miede.

