Master thesis Construction Management and Engineering

BIM based whole-building energy analysis towards an improved interoperability

A conversion from the IFC file format to a validated gbXML file format

```
#3015= IFCWINDOW('0czX8gRY1CHurVdm6jjrx9',#41,'M_Fixed:0406 x 1220mm:307793',$,'0406 x 1220mm',#5849,#3009,
#3018= IFCMATERIALLIST((#2877,#2887));
#3020= IFCPROPERTYSINGLEVALUE('ThermalTransmittance',$,IFCTHERMALTRANSMITTANCEMEASURE(3.6886),$);
#3021= IFCPROPERTYSET('0czX8gRY1CHurVbF_jjrx9',#41,'Pset_WindowCommon',$,(#2152,#2915,#3020));
#3023= IFCRELDEFINESBYPROPERTIES('3MMizFCmz65RoitCtYUt2a',#41,$,$,(#3015),#3021);
#3027= IFCQUANTITYLENGTH('Height','',$,1220.);
#3028= IFCQUANTITYLENGTH('Width','',$,406.);
#3029= IFCQUANTITYAREA('Area','area measured in geometry',$,1.83710800000007);
#3030= IFCELEMENTQUANTITY('0rYE8VgtD6qesYftay0GiY',#41,'BaseQuantities','',$,(#3027,#3028,#3029));
#3032= IFCRELDEFINESBYPROPERTIES('1sfV2Hmwr80Qb5NEaBuHnG',#41,$,$,(#3015),#3030);
#3035= IFCPROPERTYSINGLEVALUE('Level',$,IFCLABEL('Level: Level 1'),$);
#3036= IFCPROPERTYSINGLEVALUE('Sill Height',$,IFCLENGTHMEASURE(915.000000000005),$);
#3037= IFCPROPERTYSINGLEVALUE('Area',$,IFCAREAMEASURE(1.83710800000007),$);
#3038= IFCPROPERTYSINGLEVALUE('Volume',$,IFCVOLUMEMEASURE(0.0418291920000012),$);
#3039= IFCPROPERTYSINGLEVALUE('Mark',$,IFCTEXT('3'),$);
#3040= IFCPROPERTYSINGLEVALUE('Head Height',$,IFCLENGTHMEASURE(2135.),$);
#3041= IFCPROPERTYSET('0czX8gRY1CHurVcHMjjrx9',#41,'Constraints',$,(#3035,#3036));
#3043= IFCRELDEFINESBYPROPERTIES('0czX8gRY1CHurVc1Mjjrx9',#41,$,$,(#3015),#3041);
#3046= IFCPROPERTYSET('0czX8gRY1CHurVcGEjjrx9',#41,'Dimensions',$,(#3037,#3038));
#3048= IFCRELDEFINESBYPROPERTIES('0czX8gRY1CHurVc0Ejjrx9',#41,$,$,(#3015),#3046);
#3051= IFCPROPERTYSET('0czX8gRY1CHurVcG2jjrx9',#41,'Identity Data',$,(#3039));
#3053= IFCRELDEFINESBYPROPERTIES('0czX8gRY1CHurVc02jjrx9',#41,$,$,(#3015),#3051);
#3056= IFCPROPERTYSET('2Mzvy8ITXFCwH3rzeyi2D5',#41,'Other',$,(#2935,#2936,#2937,#2939,#2940,#2941,#3040));
#3058= IFCRELDEFINESBYPROPERTIES('3xUksQwBTBCRaqTYg1fFq7',#41,$,$,(#3015),#3056);
#3061= IFCPROPERTYSET('0czX8gRY1CHurVcGwjjrx9',#41,'Phasing',$,(#2078));
#3063= IFCRELDEFINESBYPROPERTIES('0czX8gRY1CHurVc0wjjrx9',#41,$,$,(#3015),#3061);
```

M.G. (Maarten) Visschers
July 8, 2016
Eindhoven

Technische Universiteit
Eindhoven
University of Technology
Where innovation starts

Construction Management and Engineering
Department of the Built Environment
Master of Science (MSc)

Colophon

Title:
Building Information Modeling (BIM) based whole-building energy analysis towards an improved interoperability

Subtitle:
A conversion from the IFC file format to a validated gbXML file format.

Author: M.G. (Maarten) Visschers
University student number: 0845720
E-mail: maartenvisschers@hotmail.com
Thesis defense date: July 8, 2016

University: Eindhoven University of Technology (TU/e)
Faculty: Department of the Built Environment
Master program: Construction Management and Engineering (CME)

In collaboration with:
Company: Arcadis Nederland
Division: Buildings
Department: Performance Driven Design

Graduation committee:
Chairman (TU/e): Prof. dr. ir. B. (Bauke) de Vries
Graduation supervisor (TU/e): Assistant prof. dr. dipl-ing. J. (Jakob) Beetz
Graduation supervisor (TU/e): Doctoral candidate (PhD Stud.) T.F. (Thomas) Krijnen
External supervisor (Arcadis): Ing. T.C. (Tom) Borst

Contact details:
Den Dolech 2
5612 AZ Eindhoven, The Netherlands
T: +31 (0)40 247 4311
E: secretariaat.b@bwk.tue.nl

"I am convinced that whoever builds a clean energy economy, whoever is at the forefront of that, is going to own the 21st -century global economy."

- Barack Obama -
President of the U.S. (at Meeting with Governors on Energy Policy)
February 3, 2010

Preface

This master thesis is the result and final project of my time at the Eindhoven University of Technology. After six months I finished my graduation project which is representing the end of my master study Construction Management and Engineering. The project is carried out in collaboration with the Eindhoven University of Technology and the engineering consultancy company Arcadis.

By conducting research in the field of Building Information Modeling (BIM) and Building Performance Simulation (BPS) I aim to contribute positively to the increasing demand of sustainable solutions in the construction sector of today. I consider my graduation project as a unique opportunity to learn from and develop myself of which I will have profit in my future career.

While working on my graduation project I had a lot of interesting conversations with inspiring people and I would like to thank all who helped me during the research. Special thanks go to my university supervisor Jakob Beetz for his support, guidance and moments of discussion. In addition, I want to thank Tom Borst (Arcadis) for his valuable input and pointing me to the right connections within the company. Then, many thanks to Thomas Krijnen who helped me with his expertise in programming and crucial technical input. Last but not least, I want to thank my family, friends and fellow students for all their help and support.

I hope you will enjoy reading this thesis as much as I enjoyed conducting the research.

Maarten Visschers

Eindhoven, July 2016

Summary

The building sector contributes up to 30% of global annual greenhouse gas (GHG) emissions and is responsible for approximately 40% of the total energy consumption. Unless actions are undertaken, GHG emissions from the building sector this will be more than doubled in the next 20 years. As a result, the European Parliament and the Council of the European Union created an action plan for energy efficiency which identifies the significant potential for cost-effective energy savings in the building sector. In this plan it is stated that all member states need to ensure that by the year 2018 new buildings occupied and owned by public authorities have to be nearly zero-energy buildings. Also, by the year 2020 all member states have to make sure that new buildings are realized as nearly zero-energy buildings. In addition, tighter budgets and higher customer expectations are responsible for more pressure on project participants to control the life cycle costs (LCC). This results in an increasing attention for methods which are able to create proper trade-offs between LCC and an improved sustainability of the building sector.

In recent years, Building Information Modeling (BIM) had become a popular approach used for sustainable building design. BIM is able to deliver relevant building information which is required for Building Performance Simulation (BPS). If this information is used appropriately BIM can save significant amount of time and effort in preparing input data for BPS; while simultaneously reducing human error. The communication and interoperability between software is mainly depending on data exchange formats and their mutual compatibility. However, in many research literature it is stated that a smooth integration between software packages is typically lacking. Creating a file in one program may result in an insufficient, unreadable or wrong input for another program, this leads to time consuming or even crashing simulations. This lacking compatibility results in the manual input of building information into BPS software, often a process of re-entering information that already exists in different forms or data bases.

Consequently, this research aims to create an improved interoperability between BIM and BPS while making use of open BIM standards such as Industry Foundation Classes (IFC) and Green Building eXtensible Markup Language (gbXML). Hence, the main research problem of this study concerns the translation of the open exchange format IFC to a validated gbXML file format; no standalone mapping or conversion tool between these formats exists today. As a result the following central research question is formulated: *"How can the use of Building Performance Simulation software be implemented during an early phase of Building Information Modelling in an approachable manner?"*. In order to answer this question, the entire research is divided into 3 phases. The first phase covers a literature study to background information and related work. This is broken down into the design process (BIM), simulation process (BPS) and the use of open BIM formats (IFC and gbXML). Then, the second phase aims on the analysis of current and desired business processes that occur between design and simulation parties. Moreover, the IFC file format and the input requirements for BPS are studied. Lastly, the third phase of the research covers the actual development of the proposed conversion tool, requirements are set and the process of scripting and matching of gbXML elements with IFC entities is started. Relationships between gbXML elements and corresponding IFC entities are created and linkages are established by iterating over multiple IFC entities.

Additionally, when performing scientific research a crucial part is the validation process, especially in the field of software development. The conducted research resulted in a "proof of concept" which introduce limitations, large sets of operating instructions and lacking user interfaces. By studying the capabilities, barriers and opportunities it is possible to eventually establish a fully automated and well-functioning tool which is usable in practice. Therefore, the conversion tool is tested by the means of 5 test cases that each differ in their complexity. This diverse complexity implies the use of differing project sizes, building elements, shapes and materials. Each of the pilot projects are drawn in design software (Autodesk Revit 2016 in this case) and then exported to the IFC file format. These IFC models are studied and then converted to the gbXML file format while making use of the conversion tool. The gbXML model is then validated according to the official schema and also optical tested and compared with the original IFC model. This optical test provides differences between the two models and their extent of information. Finally, the simulation software DesignBuilder is used to perform whole-building energy analysis.

The tool development and subsequently the validation process revealed that gbXML models can be successfully created and comply with the official gbXML schema; most essential elements are included. Moreover, the following geometric IFC entities of building models can be converted to gbXML elements: *IfcWallStandardCase*, *IfcSlab* (only floors), *IfcWindow* and *IfcCovering*. Likewise, building opening elements and thermal properties of materials are successfully included as well. Testing indicated that complex building models cause issues with regard to included building geometry or the extraction of thermal properties from the IFC file. The manner in which projects are created in design software is key to this issue.

Finally, the central research question can be answered by proposing the developed tool which handles the integration between BIM and BPS. Project costs and the duration of early-design analysis can be brought to a minimum, while improving the quality of building projects. This graduation research initiates the potential and possibilities of converting the IFC file format to a validated gbXML file format. The found results indicate the capabilities and shortcomings of the current "proof of concept" and subsequently function as foundation for future research. By conducting further research, blind spots in business processes might be eliminated in the future and human error can be brought to a minimum. Ultimately an improved conversion tool can contribute to a more developed BIM environment and comply with the increasing demand of sustainable solutions in the construction sector of today.

Dutch summary

De bouwsector draagt voor 30% bij aan de jaarlijkse wereldwijde uitstoot van broeikasgassen (BKG) en is bovendien verantwoordelijk voor circa 40% van het totale energie verbruik. Wanneer er geen maatregelen genomen worden zal de uitstoot van BKG in de bouwsector de komende 20 jaar meer dan verdubbelen. Dit is de reden dat het Europees Parlement en de Raad van de Europese Unie een actieplan heeft opgesteld ten behoeve van energie-efficiënte. Dit plan geeft het aanzienlijke potentieel weer voor het creëren van kosten efficiënte energiebesparingen in de bouwsector. Er wordt gesteld dat in 2018 alle lidstaten ervoor dienen te zorgen dat door de overheid beheerde en gebruikte nieuwbouw gerealiseerd moet worden als nagenoeg energieneutraal. In 2020 dienen bovendien alle lidstaten ervoor te zorgen dat alle nieuwbouw wordt gerealiseerd als bijna energieneutraal. Daarnaast zorgen krappere budgetten en hogere verwachtingen van de klant voor een toenemende druk op projectdeelnemers om de life cycle costs (LCC) te beheersen. Dit resulteert in een toenemende vraag naar methoden, die in staat zijn om correcte afwegingen te maken tussen LCC en een verhoging van de duurzaamheid in de bouwsector.

In de afgelopen jaren is het gebruik van Building Information Modeling (BIM) een nuttige manier gebleken om bij te dragen aan een duurzame bouwsector. Met BIM kan men relevante gebouwinformatie leveren voor Building Performance Simulation (BPS). Wanneer deze informatie op een juiste manier wordt gebruikt kan BIM zowel kosten, tijd en arbeid besparen tijdens de invoer van gegevens in BPS. Bovendien kan op deze manier menselijke falen worden beperkt. De communicatie en interoperabiliteit tussen software is voornamelijk afhankelijk van hun formaten ten behoeve van gegevensuitwisseling en de onderlinge compatibiliteit. In diverse studies wordt echter aangegeven dat een probleemloze integratie tussen verschillende softwarepakketten veelal ontbreekt. Het creëren van een bestand in het ene programma kan resulteren in een ontoereikende, onleesbare of verkeerde invoer in een ander programma. Dit heeft tijdrovende of zelfs incorrecte simulaties tot gevolg. Het ontbreken van een complete integratie resulteert in handmatige invoer van gebouwinformatie in BPS software. Vaak is dit een proces van het opnieuw invoeren van informatie, die al bestaat in een ander formaat.

Volgend uit de gestelde problematiek richt dit onderzoek zich op het tot stand brengen van een verbeterde interoperabiliteit tussen BIM en BPS, dit door gebruik te maken van open BIM standaarden als Industry Foundation Classes (IFC) en Green Building eXtensible Markup Language (gbXML). De probleemstelling van deze studie betreft de vertaling van het open uitwisselingsformaat IFC naar een gevalideerd gbXML bestand. Op dit moment bestaat er geen eenzijdige vertaling of conversie tool. Dit resulteerde in de volgende onderzoeksvraag: *"Hoe kan het gebruik van Building Performance Simulation software laagdrempelig geïmplementeerd worden in een vroeg stadium van Building Information Modeling?"*. Om deze vraag te beantwoorden is het onderzoek verdeeld in 3 fasen. De eerste fase betreft een literatuurstudie naar achtergrondinformatie. Dit onderdeel is onderverdeeld in het ontwerpproces (BIM), het simulatie proces (BPS) en het gebruik van open BIM standaarden (IFC en gbXML). Vervolgens richt de tweede fase zich op het analyseren van de huidige en gewenste bedrijfsprocessen. Hierbij zijn het IFC uitwisselingsformaat en de nodige invoervereisten van BPS bestudeerd. Tot slot omvat de derde fase van het onderzoek de daadwerkelijke tool ontwikkeling. Eisen worden opgesteld, waarna het proces van scripten

en het koppelen van gbXML elementen met IFC entiteiten begonnen kan worden. Relaties tussen gbXML elementen en corresponderende IFC entiteiten worden gecreëerd en verbanden worden vastgesteld door het itereren over meerdere IFC entiteiten.

Een cruciaal onderdeel van wetenschappelijk onderzoek, met name op het gebied van software ontwikkeling, is het validatie proces. Het verrichte onderzoek is een zogenaamd "proof of concept" en brengt verschillende beperkingen en gebruiksaanwijzingen met zich mee. Door de capaciteiten, knelpunten en mogelijkheden te bestuderen is het mogelijk om uiteindelijk een volledig geautomatiseerde, functionerende en gebruiksklare tool te ontwikkelen. Op grond hiervan is de conversie tool getest aan de hand van 5 test cases welke ieder verschillen in hun complexiteit wat betreft de projectomvang, gebouwelementen, vormen en materialen. Iedere test case is gemodelleerd in ontwerp software (Autodesk Revit 2016 in dit geval) en vervolgens geëxporteerd naar het IFC uitwisselingsformaat. Deze IFC modellen zijn bestudeerd en vervolgens omgezet naar het gbXML uitwisselingsformaat met behulp van de conversie tool. De gbXML modellen zijn gevalideerd volgens het officiële schema en bovendien optisch vergeleken met het originele IFC model. Zo is inzicht verkregen in de verschillen tussen de twee modellen en de mate van informatie. Tot slot zijn met de resulterende gbXML modellen in het simulatie programma DesignBuilder energie simulaties uitgevoerd.

Tijdens de ontwikkeling van de tool en het hierbij horende validatieproces is gebleken dat gbXML modellen op een succesvolle manier gecreëerd kunnen worden en bovendien voldoen aan het officiële gbXML schema; de meest essentiële gebouwelementen zijn inbegrepen. Bovendien kunnen de volgende geometrische IFC entiteiten van modellen worden omgezet naar gbXML elementen: *IfcWallStandardCase*, *IfcSlab* (alleen vloeren), *IfcWindow* en *IfcCovering*. Hetzelfde geldt voor opening elementen en thermische eigenschappen van materialen. Daarnaast is gebleken dat er geometrie gerelateerde problemen ontstaan wanneer meer complexere modellen geconverteerd worden. Ook het achterhalen van thermische eigenschappen in het IFC bestand geeft problemen bij deze modellen. Deze problemen komen voort uit de manier waarop projecten gemodelleerd worden in ontwerp software.

Tot slot, de centrale onderzoeksvraag kan beantwoord worden met de ontwikkelde tool die de integratie tussen BIM en BPS handhaaft. Zowel de kosten als doorlooptijd van vroegtijdige simulaties kunnen worden geminimaliseerd, terwijl de kwaliteit van bouwprojecten naar een hoger niveau wordt gebracht. Dit afstudeeronderzoek initieert het potentieel van het converteren van het IFC uitwisselingsformaat naar een gevalideerd gbXML bestand. De resultaten brengen de capaciteiten en tekortkomingen van het huidige prototype in kaart en functioneren bovendien als fundament voor toekomstig onderzoek. Door het uitvoeren van vervolgonderzoek kunnen in de toekomst zogenaamde "blinde vlekken" in bedrijfsprocessen worden geëlimineerd en het menselijk falen kan geminimaliseerd worden. Uiteindelijk kan een verbeterde conversie tool positief bijdragen aan een meer ontwikkelde BIM omgeving en de toenemende vraag naar duurzame oplossingen in de hedendaagse bouwsector.

Table of Contents

Abbreviations

AEC	*Architecture, Engineering and Construction*
BIM	*Building Information Modeling*
BPMN	*Business Process Model and Notation*
BPS	*Building Performance Simulation*
B-rep	*Boundary Representation*
bsDD	*BuildingSMART Data Dictionaries*
CAD	*Computer-Aided Design*
CSG	*Constructive Solid Geometry*
CE	*Concurrent Engineering*
ER	*Exchange Requirements*
GBS	*Green Building Studio*
gbXML	*Green Building eXtensible Markup Language*
GUI	*Graphical User Interface*
GUID	*Global Unique Identifier*
IBD	*Integrated Building Design*
IDM	*Information Delivery Manual*
IFD	*International Framework for Dictionaries*
IFC	*Industry Foundation Classes*
IFD	*Industry Foundation Dictionaries*
ISO	*International Standardization Organization*
LOD	*Level of Detail*
MVD	*Model View Definition*
STEP	*Standard for the Exchange of Product model data*
XML	*Extensible Markup Language*
UML	*Unified Modeling Language*

1. Introduction

The awareness and implementation of energy efficiency in the built environment is getting progressively more entrenched in the public policy. The European Parliament and the Council of the European Union created an action plan for energy efficiency which identifies the significant potential for cost-effective energy savings in the building sector. It is stated that all member states need to ensure that by the year 2018 new buildings occupied and owned by public authorities have to be nearly zero-energy buildings. Furthermore, by the year 2020 all member states have to make sure that new buildings are realized as nearly zero-energy buildings (EBPD, 2010). Moreover, the building sector contributes up to 30% of global annual greenhouse gas (GHG) emissions and is responsible for approximately 40% of the total energy consumption. In 2009 the United Nations Environment Programme (UNEP) declared that GHG emissions from the building sector will be more than doubled in the next 20 years unless actions in order to minimize the emissions are undertaken (UNEP, 2009). Additionally, tighter budgets and higher customer expectations are responsible for more pressure on project participants to control the life cycle costs (LCC). This results in an increasing attention for methods which are able to create proper trade-offs between LCC and an improved sustainability of the building sector (Liu, Meng, & Tam, 2015).

According to the Intergovernmental Panel on Climate Change (2008) the building sector has the largest potential for delivering long-term, significant and cost-effective solutions regarding GHG emissions. Estimated for the year 2030, the potentials and costs of GHG mitigation for each economic sector are set out in figure 1.1. Hereby, the concerned bottom-up studies are divided into four categories; based on membership of the Organisation for Economic Co-operation and Development (OECD).

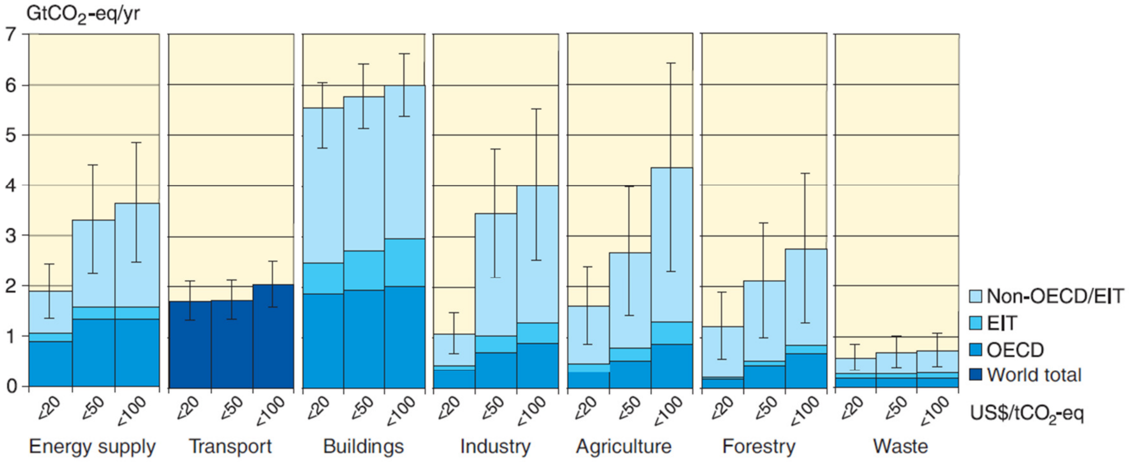


Figure 1.1: Estimated economic mitigation potentials by sector in 2030 from bottom-up studies (IPCC, 2008)

Nowadays, there are numerous building certificates and initiatives in the Architecture, Engineering and Construction (AEC) industry that call for life cycle optimization and approaches for realization of sustainability. A shift from the traditional design process towards a more integrated planning practice has been recognized as a necessary step in the direction of an energy efficient built environment. Integrated Building Design (IBD) is advocated as a suitable approach for this new aim. This IBD method supports early collaboration of project stakeholders and therefore an optimization of performance in early phases of a project. Interactions between project stakeholders on multiple levels, while using virtual environmental ICT tools, supports building processes in such a way that the transfer of information and the creation of new knowledge are attributing to innovation in its entirely (Kovacic & Müller, 2014). Furthermore, as stated by Wong & Zhou (2015) the developments of Computer-Aided Design (CAD) software and Building Information Modelling (BIM) have changed the traditional design formats and communication patterns of the AEC industry over the last two or three decades. BIM generates a systematic approach to managing the essential information for building design and project data in digital formats throughout the life cycle of a building.

In recent years, BIM had become a popular approach used for sustainable building design. BIM is able to deliver relevant building information which is required for Building Performance Simulation (BPS). If this information is used appropriately BIM can save significant amount of time and effort in preparing input data for BPS; while reducing human error. According to Ryan et al. (2012) BPS can save significant amounts of costs, time and labor if an accurate simulation is made during the early design phase (Ryan & Sanquist, 2012). In addition, a project member needs to interact between various design and simulation tools during multiple phases in a building project. By doing this it is possible to predict the performance of the future design. This interacting makes the interoperability among different software tools a necessity. Moreover, a large and growing body of literature has reported that the seamless integration between these software tools is typically lacking today (Rahmani, Zarrinmehr, Bergin, & Yan, 2015).

Augenbroe (2002) indicated two major movements started in parallel with similar goals in mind to address the above described issue: (1) a collective effort by industry, governmental and research organizations to establish data exchange standards for the building industry such as Industry Foundation Classes (IFC) or Green Building eXtensible Markup Language (gbXML); and (2) researches and the industry attempt to address the existing interoperability by scripting interfaces and self-developed applications between the design and performance simulation tools. The first aims to remove inefficiencies in data sharing by representing the relevant data within each program to a generic common data model that contains the required information by all other programs, the second focuses to functionally create a connection among two or more design and performance simulation software tools for specific goals (Augenbroe, 2002). Combining both statements suggests the value of an application which connects BIM and BPS, this while making use of open data exchange standards to increase and automate interoperability.

2. Research approach

2.1. Problem definition

In order to tackle upcoming new building requirements, the AEC industry is currently moving towards a more developed BIM environment. As mentioned in the introduction part of this thesis, project members within the AEC industry need to be able to interact between multiple design and simulation software while acting in different phases of a building project. For this reason it is needed to ensure interoperability between the different software tools. In many research literature it is stated that a smooth integration between these software packages is typically lacking (Rahmani et al., 2015).

Subsequently, todays companies and project members are making use of varying design software and multiple simulation software throughout building projects that are subject to change. Communication and interoperability between software is mainly depending on data exchange formats and their mutual compatibility. Creating a file in one program may result in an insufficient, unreadable or wrong input for another program. Moreover, files derived from design software are often unnecessarily complex and too large for simulation software to operate, this leads to time consuming or even crashing simulations. This lacking compatibility often results in the manual input of building information into simulation software, this process is time consuming and in addition sensitive for human error. As indicated by Bazjanac & Crawley (1999), a major reason for extra costs and the duration of early analysis is the process of entering building information that is needed to perform analysis. Additionally, this is a process of re-entering information that already exists in different forms in other documents or data bases. Approximately 80% or more of the resources needed to perform whole-building energy analysis is advocated by this current way of transferring building information. Figure 2.1 shows a schematic representation of the described process concerning the lacking communication and interoperability.

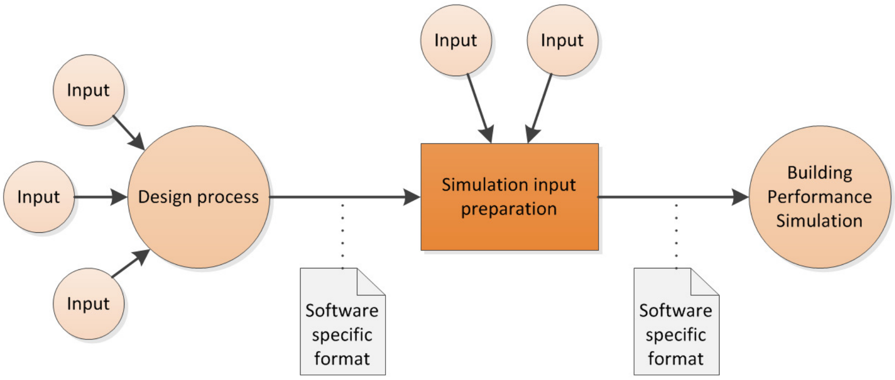


Figure 2.1: Schematic representation of the problem definition

Considering (1) the nowadays limited interoperability among design and simulation tools, (2) the broad potentials of BIM, energy performance and open data formats, and (3) the possibilities to use BIM in order to supply energy performance software with the necessary information, it seems valuable to study the process of creating a mapping and automatic conversion between BIM and BPS. Hence, the main research problem of this study concerns the translation of the open exchange format IFC to the open exchange format of gbXML. No standalone mapping or conversion tool between these formats exists today.

2.2. Purpose and limitations

The described concerns within the AEC industry are asking for a seamless integration between BIM and energy performance software. This research aims to create an improved interoperability between those aspects. The main purpose is to create a clear mapping and translation between open exchange formats of design software (IFC) and building performance software (gbXML). By developing an automatic data converter, which handles the communication and interoperability, it is possible to minimize human error during the input of BPS. In addition, as stated in the introduction by Ryan et al. (2012), significant amounts of cost, time and labor can be saved if an accurate simulation is made during an early design phase.

In addition to the explained main purpose there are several sub-objectives which need to be accomplished first. A broad understanding in terms of BIM, BPS and data formats such as IFC and gbXML is needed. Moreover, necessary input information concerning energy simulations and possible export options regarding to the design software need to be known. Eventually, scripting skills and knowledge are required in order to develop the conversion tool. Most important during this development is the relationship between information in IFC and gbXML; necessary information for gbXML needs to be identified in IFC. Finally, after validating and testing the developed conversion tool the main objective will be reached. Results can contribute to a more efficient and sustainable AEC industry and eventually to reaching the required new building standards in the year 2020. Figure 2.2 shows a schematic representation of the desired process from design software to simulation software.

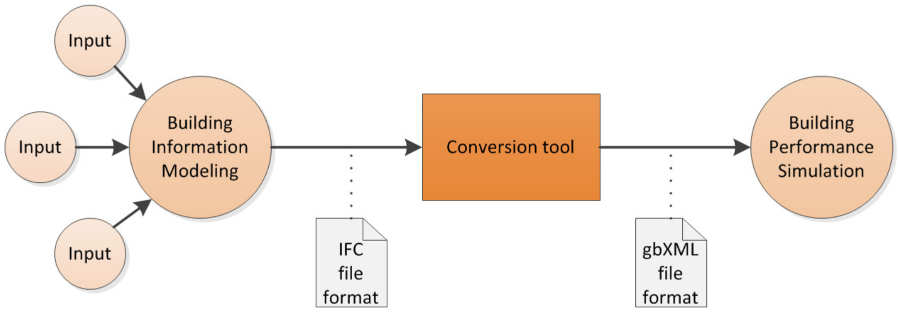


Figure 2.2: Schematic representation of the research purpose

While reaching for the goal of this research several limitations are concerned in order to scope the work possible in a MSc thesis. Ensuring those limitations will contribute to a well-defined and elaborated tool which provides valuable results. During the research a theoretical framework that functions as background information will be followed up by a practical study which includes interviews with relevant experts in the field of building performance and BIM. This last part specifically aims on current methods and knowledge in the AEC industry and more particularly on the organization within Arcadis. Involving other parties from the field may result in differing results.

Furthermore, in order to do in depth studies to the translation of data formats it is needed to make choices regarding the concerned software and used file formats. These choices are made based on commonly used software and file formats in the Dutch market. Other software or formats may lead to differing results. Limitations with regard to the developed tool will be discussed in the validation phase of this research. Finally, a more obvious but important limitation is the restricted period of time; hard deadlines are involved.

2.3. Research questions

In order to elaborate on the previous stated research problem a central research question is formed. Providing answer to this question will contribute to the outlined problem. The central research question of this thesis is:

"How can the use of Building Performance Simulation software be implemented during an early phase of Building Information Modelling in an approachable manner?"

Besides the main question, several sub-questions are formulated in order to provide a well-grounded approach to eventually answer to the central stated question. These sub-questions each contribute to the central question in their own field. The questions are numbered according to the research structure. The formulated sub-questions together with their involved subjects and actions are stated below.

1.1. What design software and file formats are commonly used during BIM and what are their features?

- Examining the AEC industry, integration of BIM and commonly used file formats. This includes interoperability, maturity levels and the data exchange file format IFC. Scientific literature will be examined and analyzed in order to gain the needed input.

1.2. What simulation software and file formats are commonly used during BPS and what are their features?

- Studying existing BPS software, their input information and used file formats. This involves the importance of early simulations, capability to be linked with BIM and the exchange file format gbXML. Scientific literature will be examined and analyzed to obtain the needed knowledge and input for further analysis in this thesis.

2.1. How is the current and desired business process mapping between the design parties and BPS organized?

- Mapping the current business process between design parties and the simulation consultant. This overview is needed to gain knowledge about errors and bottlenecks. Hereafter, the same business mapping is created for the desired process. Interviews with experts in the field of BIM and BPS are concerned.

2.2. What design model-parameters are needed to comply with early BPS?

- This concerns the specific need of information for early energy simulations. Concerned are principles of BPS, the input specification and building parameters with their effect during performance simulations. This involves the export of IFC files, geometry and the use of space boundaries. Also, earlier obtained knowledge functions as input during this sub study.

3.1. What are the needed characteristics of a conversion tool between BIM and BPS?

- Gathering information and skills which are needed in order to create an automatic data converter between BIM and BPS. Python programming is concerned and the characteristics of the tool are established based on the MoSCoW method.

3.2. What specific information needs can be converted to comply with BPS input?

- Studying (1) the input specification of gbXML and (2) the export possibilities of IFC. To establish a relationship between both formats the specific information for gbXML needs to be identified in IFC. Knowledge of both exchange formats, earlier obtained skills of Python language and the established characteristics of the tool are required.

3.3. What is the potential of the conversion tool after validating its capabilities?

- Analyze the results and potential of the developed conversion tool. Whole-building energy analysis are done during this process in order to test the contents and functioning of the established gbXML file. By doing this it is possible to provide future recommendations and point out area for improvement.

2.4. Research model and methods

The entire research approach is displayed in figure 2.3 below; sub-questions can be seen with their corresponding project development and project activities.

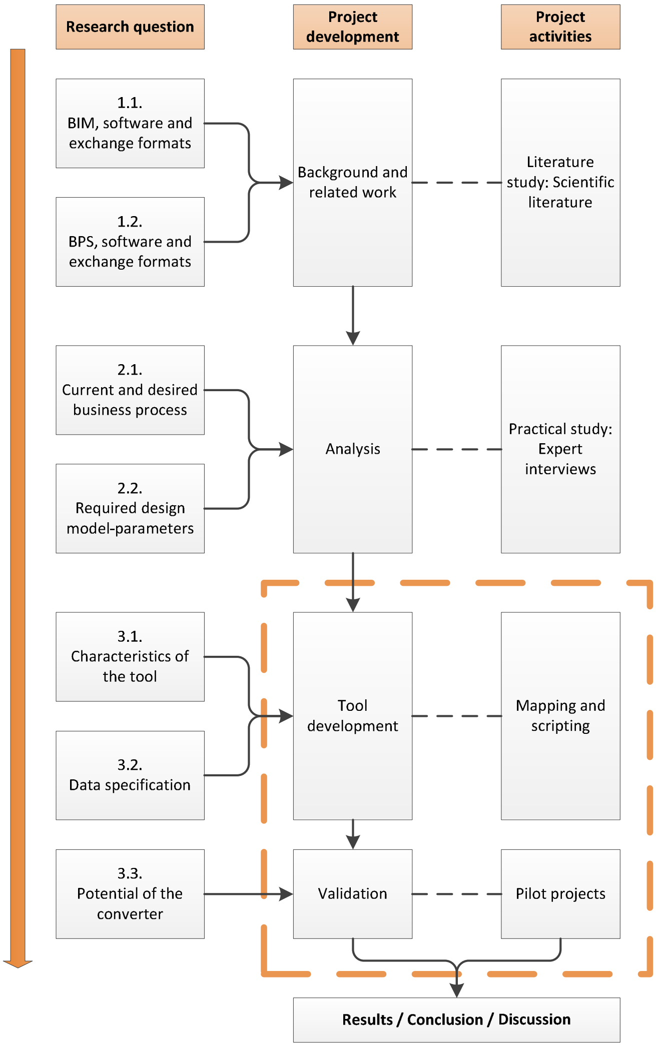

Research question
Project development
Project activities
1.1. BIM, software and exchange formats
1.2. BPS, software and exchange formats
Background and related work
Literature study: Scientific literature
2.1. Current and desired business process
2.2. Required design model-parameters
Analysis
Practical study: Expert interviews
3.1. Characteristics of the tool
3.2. Data specification
Tool development
Mapping and scripting
3.3. Potential of the converter
Validation
Pilot projects
Results / Conclusion / Discussion

Figure 2.3: Research model of the thesis

As obtained from figure 2.3, in total 3 phases can be distinguished during this research. The first phase is a literature study which is making use of scientific literature, then the second phase is an analysis phase with practical studies, and the third phase is a validation phase.

The literature study is required in order to obtain the necessary information and knowledge needed for the next phases of the research. This literature study is making use of scientific journals and research literature. During this phase sub-questions 1.1 and 1.2 are concerned. Hereafter, the analysis phase is concerned in order to describe real world phenomena and obtain needed knowledge and data according to the Dutch market. Besides earlier obtained information this phase concerns in-depth interviews with experts from the field. In this part the sub-questions 2.1 and 2.2 are examined.

With the input of both the literature and the practical study the last phase can be carried out. During this last phase the development of the new conversion tool is elaborated and validated by the means of a pilot projects. At this point sub-questions 3.1, 3.2 and 3.3 are studied. Finally, all sub-questions are discussed allowing the central research question to be answered and future recommendations to be made. In order to clarify the research method it is helpful to show its schematic representation, see figure 2.4.

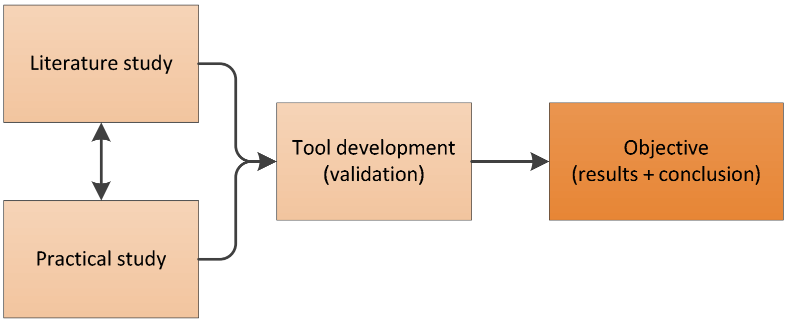


Figure 2.4: Methodological justification

2.5. Expected results

By conducting this research it is expected to provide well-grounded answers to the formulated main and sub-questions of this thesis. By doing this, this study will contribute to the search for new solutions concerning the upcoming new building requirements. The results are estimated to improve the interacting between design software and building performance software while making use of open BIM standards.

By creating a mapping and an automatic conversion tool between the data exchange formats of BIM (IFC) and simulation software (gbXML) it is expected to decrease the lacking compatibility and increase the communication and interoperability. In practice this leads to less manual input of building information to simulation software, which saves time and takes away the factor of human error.

This study will perform whole-building energy analysis to test and validate the potential of the established tool. By the means of this validation process conclusions can be made and recommendations for future research can be provided.

3. Background and related work

In order to provide answers to research questions 1.1 and 1.2 a literature study is conducted. This study provides necessary information which functions as foundation of the research. A broad understanding of the Architecture, Engineering and Construction (AEC) industry and its background is created in chapter 3.1. Discussed issues are the known inefficiencies and the needed interoperability of today. In chapter 3.2 the term Building Information Modeling (BIM) and its capabilities are being explained. Hereafter, the process of Building Performance Simulation (BPS) is studied and the need of early design simulations is explained in chapter 3.3. The final part of the literature study, chapter 3.4, concerns open exchange formats; the application of open standards and their added value are set out.

3.1. Architecture, Engineering and Construction industry

Originally, the AEC industry has been identified as fragmented. To get to the bottom of this it is useful to compare the construction sector with component manufacturing industries such as electrical and automotive engineering. Many studies within these industries are done with regard to produce models of buyer-supplier relations. According to Cox & Thompson (1997) those relations are different within the construction sector. Repetition is rare in the construction sector and work is produced mostly on the construction site. This causes cost inefficiencies for the stakeholders and a new learning curve is climbed every time a project is started.

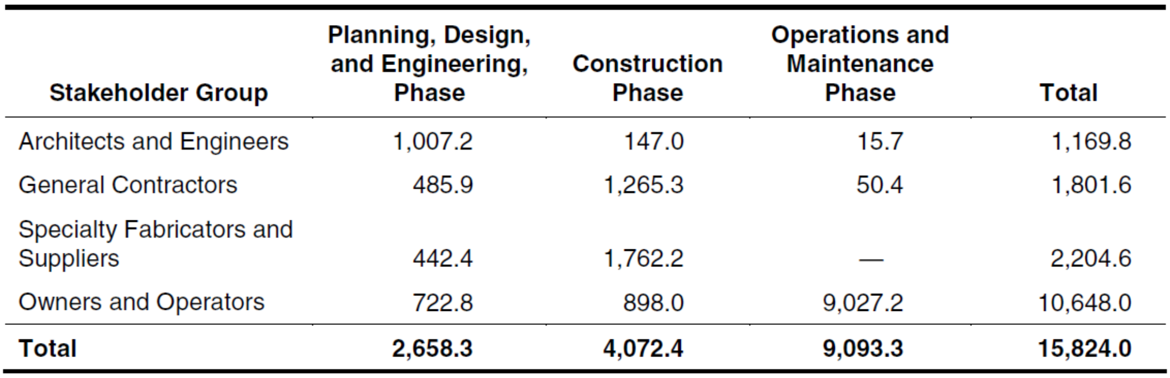
Over the past years multiple methods are introduced with the aim of decreasing the known inefficiencies of the construction sector. In the year 1989 the term Concurrent Engineering (CE) was used for the first time in the Unites States. The aim of this new method was to establish an environment where companies reduce cost and minimize product development lead time while simultaneously improving the quality of their products (Sohlenius, 1992). Moreover, CE was developed caused by rapidly decreasing product lifecycles. The time to market process needed to be improved (Koufteros, Vonderembse, & Doll, 2001). Pennell & Winner (1989) define CE as follows: *"A systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support. This approach is intended to cause the developers, from the outset, to consider all elements of the product life cycle from conception through disposal, including quality, cost, schedule, and user requirements"*. Consequently, a shift from the traditional division of tasks and phases, where minimum interaction between phases was usual, towards the integration of design stages was introduced. This integration implies an improved communication and collaboration between the conceptual design stage and process- and production- design phases (Sohlenius, 1992).

Nowadays, the management of knowledge and information exchange is getting more and more essential in AEC teams; there is a growing recognition of the benefits of knowledge management to construction firms (Carrillo & Chinowsky, 2006). Challenges during collaboration are often intensified by geographical separation and team members' cultural differences. Project members who receive visualizations and models reinterpret through their own domain lens and their disciplinary expertise. This results in conversations where meaning is partly made based on the exchange of interpretations, perspectives and knowledge. This unplanned and surprising occurrence supports brainstorming and mutual discovery and is often known as "messy talking". Moreover, when visualizations and models are created in a collaboration that is multidisciplinary, new ways of identifying, analyzing and synthesizing problems can occur (Dossick, Anderson, Iorio, Neff, & Taylor, 2012).

To this day, reasons for the decrease in construction productivity are not completely understood. However, the efficiencies achieved in the manufacturing industry through automation, the use of information systems, better supply chain management and improved collaboration tools are known. As stated earlier, these efficiencies have not been achieved in the AEC industry yet. Incompatibility between systems often nourishes the lack of information that is being shared rapid and accurate. This causes numerous problems, including added costs (Eastman, Teicholz, Sacks, & Liston, 2011a). The National Institute of Standards and Technology (NIST) performed a study to these added costs. Based on interviews and survey responses, $15.8 billion interoperability costs were quantified for the U.S. while focusing on construction taking place in the year 2002. Table 3.1 shows the additional costs for stakeholders by life-cycle phase which occur as a result of the described lacking interoperability in the construction sector. Subsequently, table 3.2 represents the category of the costs. The NIST distinguished the following categories of costs:

- Avoidance, redundant computer systems, inefficient business process management, redundant IT support staffing;
- Mitigation, manual reentry of data, request for information management;
- Delay, costs for idle employees and other resources.

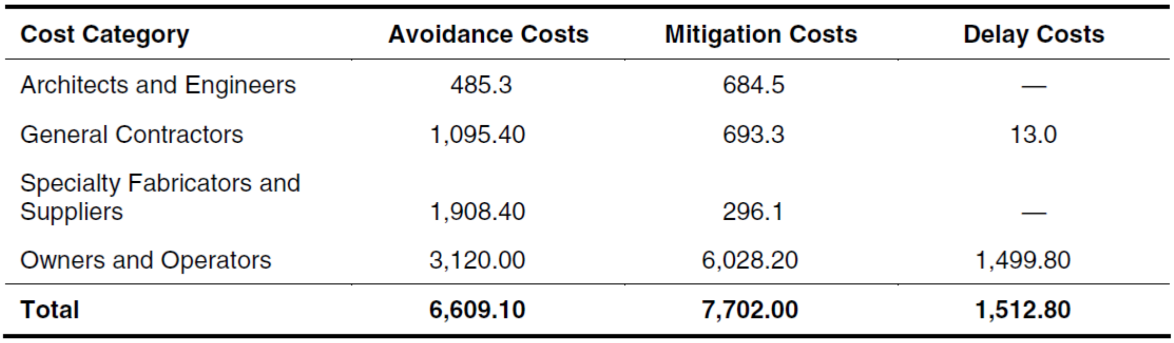
Table 3.1: Costs of inadequate interoperability by stakeholder group and life-cycle phase of the construction sector, 2002 (in $Millions)



Stakeholder Group	Planning, Design, and Engineering, Phase	Construction Phase	Operations and Maintenance Phase	Total
Architects and Engineers	1,007.2	147.0	15.7	1,169.8
General Contractors	485.9	1,265.3	50.4	1,801.6
Specialty Fabricators and Suppliers	442.4	1,762.2	—	2,204.6
Owners and Operators	722.8	898.0	9,027.2	10,648.0
Total	**2,658.3**	**4,072.4**	**9,093.3**	**15,824.0**

Source: (Gallaher, O'Conor, Dettbarn, & Gilday, 2004)

Table 3.2: Costs of inadequate interoperability by stakeholder group and cost category, 2002 (in $Millions)



Cost Category	Avoidance Costs	Mitigation Costs	Delay Costs
Architects and Engineers	485.3	684.5	—
General Contractors	1,095.40	693.3	13.0
Specialty Fabricators and Suppliers	1,908.40	296.1	—
Owners and Operators	3,120.00	6,028.20	1,499.80
Total	**6,609.10**	**7,702.00**	**1,512.80**

Source: (Gallaher et al., 2004)

Over the last two or three decades the development of Computer-Aided Design (CAD) software and Building Information Modeling (BIM) are contributing to a more integrated AEC industry (Wong & Zhou, 2015). An integration between different project development phases and multiple stakeholders supports building processes in such a way that knowledge and information exchange are attributing to a more efficient construction sector. BIM and related digital innovations can serve as a catalyst for more transparency, tighter integration and increased productivity in the AEC industry. However, many project teams struggle to fully exploit the benefits and potential of BIM (Merschbrock & Munkvold, 2015).

3.2. Building Information Modeling

Since November 2011 the Rijksgebouwendienst in the Netherlands stimulates the use of Building Information Modeling (BIM) when making use of integrated contract types such as DBGMO & DBM (Design, Build, Finance, Maintenance and Operate & Design, Build, Maintenance). The Rijksgebouwendienst (part of Rijksvastgoedbedrijf since 2014) is one of the largest real estate owners in the Netherlands and owns approximately 7 million square meters in 2000 objects such as offices, prisons, courts and museums. Director-general Peter Jägers assigns increasing failure costs and complexity of buildings and installations as foundation for the use of BIM (Rijksoverheid, 2011).

3.2.1. Diversity of definitions

Throughout the AEC industry there exists a diversity of BIM definitions when it comes to the implementation. The degree of applying BIM differs among companies and projects. Although much is written about BIM no self-contained definition among scientific literature exists. Being able to make correctly use of BIM within this research requires a broad and grounded understanding of the definition and application. Several definitions are discussed to provide insights in the differences and similarities. Eventually, a clear definition can be formed which will be leading throughout this thesis.

According to Eastman et al. (2011) the term BIM describes an activity (building information modeling), rather than an object (the building information model). It is stated that BIM is not a thing or a type of software but a human activity that ultimately involves broad process changes in design, construction and facility management. Furthermore, he describes BIM as one of the most promising developments in the AEC industry which facilitates a more integrated design and construction process. This process results in an improved quality of buildings at lower cost and reduced project duration when adopted right.

Additionally, Migilinskas, Popov, Juocevicius, & Ustinovichius (2013) mention the consistently changing definition of BIM. It is stated that this occurrence is the result of rapidly developing information technologies in the AEC industry. Implementing the technology of BIM will contribute to greater efficiencies in the construction industry through increased collaboration between different project stakeholders. Besides that, less collisions occur and the learning curve does not need to be climbed every time a project is started. Moreover, Succar (2009) describes BIM as an emerging technological and procedural shift within the AEC industry which currently is the most common denomination for a new way of approaching the design, construction and maintenance of building projects. He explains BIM as a set of interacting policies, processes and technologies generating a *"methodology to* *manage the essential building design and project data in digital format throughout the buildings' life-cycle"*.

When implementing the philosophy of BIM during this research it is helpful to explore the BIM definition formulated by Arcadis as well. This definition reads as follows: *"The processes and collaborative behaviors associated with the creation and sharing of object orientated databases of an asset in its environment, relevant to all stages of the asset's life cycle including design, construction and operation"*.

BuildingSMART International (formerly the International Alliance for Interoperability, IAI) is a platform for sharing knowledge regarding to BIM. Their technical vision on BIM is defined as a digital representation of physical and functional characteristics of a facility. Additionally, this representation functions as a shared knowledge resource for information about a facility forming a reliable basis for decisions during its life-cycle. This for existing structures and projects in the earliest conception or demolition phase (buildingSMART, 2014).

3.2.2. Maturity levels and implementation difficulties

Extensive research is done to the added value of BIM to companies; new business models are formed and based on the use of the BIM philosophy. Maturity levels and developments are widely discussed among scientific journals and research literature by various authors. In order to stimulate the implementation in the AEC industry its maturity needs to be studied and analyzed (Jayasena & Weddikkara, 2013).

The BIM industry Working Group visualized the BIM maturity levels for the British Government Construction Client Group (GCCG). This scheme is more and more adopted throughout Europe. In fact the scheme, see figure 3.1, represents a growth model for the implementation of BIM. This model is devised in order to ensure a clear understanding of the possible gradations. Levels from 0 to 3 are defined to the extent of supporting standards and their relationships to each other when applied in practice (BIM Industry Working Group (BIWG), 2011).

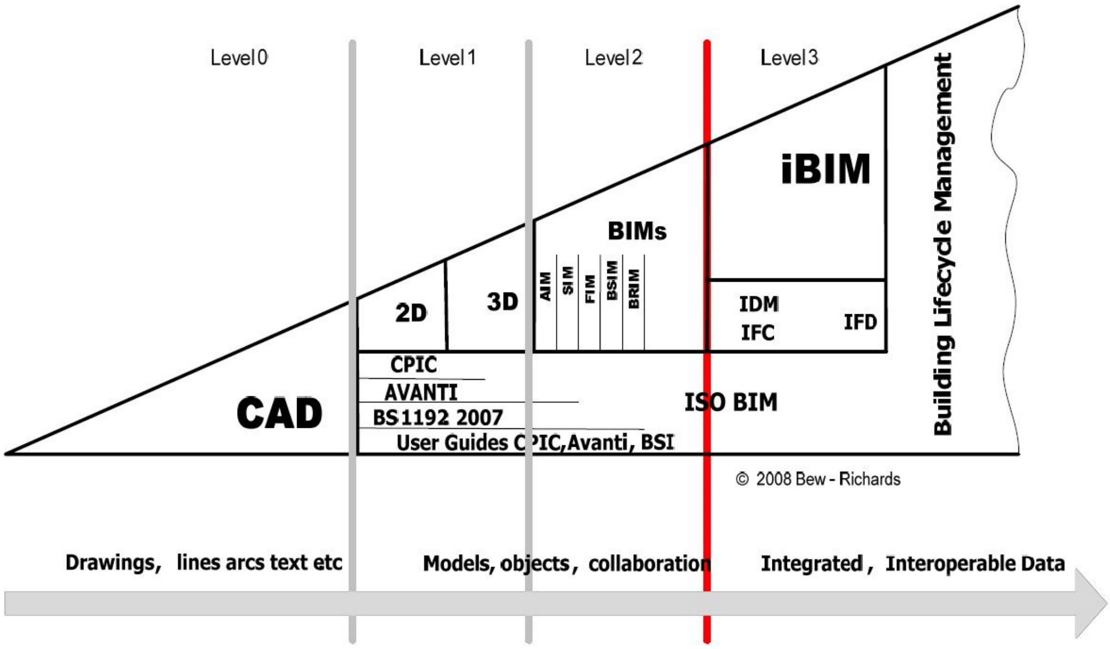

Level 0
Level 1
Level 2
Level 3
CAD
2D
3D
BIMs
AIM
SIM
FIM
BSIM
BRIM
iBIM
IDM
IFC
IFD
Building Lifecycle Management
CPIC
AVANTI
BS1192 2007
User Guides CPIC, Avanti, BSI
ISO BIM
© 2008 Bew - Richards
Drawings, lines arcs text etc
Models, objects, collaboration
Integrated, Interoperable Data

Figure 3.1: Maturity model by the UK BIM Working Group (BIM Industry Working Group (BIWG), 2011)

As mentioned in previous section, the maturity model is widely used today. When fully implementing BIM in practice all 4 levels need to be followed. The Bouw Informatie Raad in the Netherlands pointed out the key issues for each level. Level 0, also known as document oriented, is making use of non-intelligent information. Parties make use of CAD drawings or Excel based calculations, no digital objects are applied. Level 1 is object oriented, the first step towards the implementation of BIM. Objects are made in 3D design software, though there is no interaction between other disciplines such as planning or cost calculations. Next, level 2, also known as "little BIM", makes it possible to exchange the previous made objects. All models can be combined in one view-model and planning and cost calculations can be added to the object. The final stage is level 3, also known as "big BIM". In this level information is shared between parties by making use of open BIM standards. A strong relation with facility management and asset management is created (BIR, 2008).

3.2.3. Interoperability

The described final stage of BIM (maturity level 3) is reached when cooperation is based on the use of open standards and strong relations with other applications are established. This makes the interoperability among software packages and relations between multiple applications throughout construction projects a necessity. Eastman et al. (2011) describes interoperability in the AEC industry as the ability to exchange data between applications, which eases workflows and sometimes facilitates their automation.

Most commonly used methods to ensure interoperability among software applications is to exchange files using common standards. This can be done by: (1) using proprietary formats which are limited to programs of the same suite, or (2) by making use of open and neutral standards such as IFC. The buildingSMART consortium developed several technologies based on this second method. Examples are the BuildingSMART Data Dictionaries (bsDD), Model View Definition (MVD) and the Information Delivery Manual (IDM), more information about these terms is provided later in this report. In order to improve interoperability buildingSMART aims to keep improving the IFC schema, shortcomings from previous versions are removed or updated (Costa & Madrazo, 2015).

Additionally, the number of software manufacturers is skyrocketing, the amount of organizations that support IFC is estimated by buildingSMART to be around 75. Moreover, these organizations are providing the market with at least 150 BIM software packages that are IFC compatible. Packages are often costly and out of reach for individuals and small sized companies. The lack of knowledge about different BIM software packages could be seen as major barrier towards interoperability and reaching the final maturity level (Abanda, Vidalakis, Oti, & Tah, 2015).

3.2.4. Level of Development (LOD)

Collaboration during BIM requires contractual agreements and a clear understanding concerning the Level of Development (LOD) of each design phase. At any time in the process involved parties need to know the desired degree of development of their work. For this reason, the LOD is a commonly discussed subject within the philosophy of BIM.

The term LOD is used to indicate that detailing is not only about geometry (mass modeling) but also about the non-graphical information (Operation & Maintenance) of a model (Boton, Kubicki, & Halin, 2015). A study elaborated by Choi, Kim, & Kim (2015) analyzes the levels, ranging from LOD100 to LOD500 with increases of 100. Here, LOD100 represents the conceptual design phase where non-geographic data or line work and areas are used. LOD200, the schematic design phase, contains elements shown in 3D. Approximate dimensions, shape, location, orientation and quantity are provided. In LOD300, the detailed design stage, specific elements with accurate (non-geometric) information are modelled in 3D. The next level, construction stage, LOD400 models to assembly details including quantity, quality, material, texture, color, etc. The final level, LOD500, concerns information based on Operation & Maintenance (O&M) level. At this level objects are modeled as-built actual, updates and changes during the construction are included.

To conclude, Eastman et al. (2011) explains the importance of LOD by putting it in context. An architect may build a highly detailed wall system to support rendering for comparing materials. While a contractor may elect to represent the system using a single component, this because he is only interested in the building as a whole. On the other hand, an engineer who is making use of a sophisticated structural earthquake system may require a more detailed model.

3.3. Building Performance Simulation

With regard to the coming new building standards it is becoming increasingly necessary to conduct Building Performance Simulation (BPS); demands for sustainable buildings are increasing. Fulfilling the transition from conventional to sustainable buildings is studied by Biswas, Wang, & Krishnamurti (2006). Stated is the importance of the factors: technological, environmental, economic and social. Moreover, the technological and environmental factors are expected to be most significant. This enables the advent of BIM in the AEC industry to produce designs which increase sustainability in the construction sector.

However, the integration of BPS often is a problematic aspect of BIM. Today, multiple authors are developing data models to be used by BPS, which goes against the standardization aim of buildingSMART (Prada-Hernandez, Rojas-quintero, Vallejo-Borda, & Ponz-Tienda, 2015). Realizing an improved interoperability between BPS and BIM is a goal yet hard to achieve (Hitchcock & Wong, 2011).

3.3.1. Importance of early involvement

Throughout the existence of CAD the importance of designing with BPS feedback in an early stage has always existed. But then again, the fragmentation in the AEC industry complicates this involvement. Performance feedback is mainly depending on manually (re)modeling the design in dedicated BPS tools or manually importing and exporting geometry (Negendahl, 2015). Since this process is known as time consuming and error-prone, designers and energy experts have limited options when it comes to the examination of design alternatives. This last issue eventually results in non-optimized design solutions (Asl, Bergin, AdamMenter, & Yan, 2014). The traditional and the integrated design process of receiving BPS feedback is illustrated by Negendahl (2015), see figure 3.2 for the traditional process and figure 3.3 for the integrated design process.

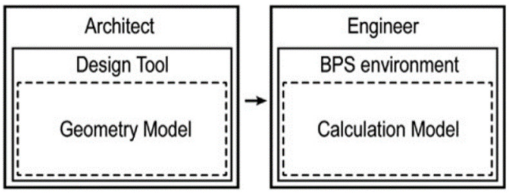


Figure 3.2: Traditional BPS feedback (Negendahl, 2015)

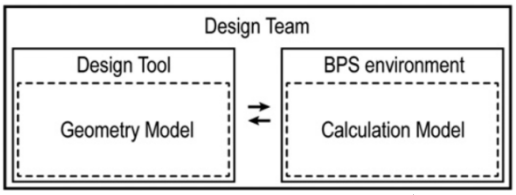


Figure 3.3: Integrated design process (Negendahl, 2015)

Early project phases do have a crucial impact on the future performance of a building throughout its lifecycle. Up to 80% of operational costs as well as environmental impacts are determined in an early design phase. Furthermore, operational costs exceed construction costs by a multiple (Kovacic & Zoller, 2014). The described impact of early design decisions on the overall functionality, costs and benefits of a construction project is being emphasized by Eastman et al. (2011), see figure 3.4. The decreasing value and effect of decisions made during the life-cycle of a project (line 1) is set out against the growth of costs when making changes during a project (line 2). Early design changes are able to bring a high functionality alongside minimal costs. Additionally, the relationship between design effort and time for both the traditional design process and the BIM process are visualized in the graph.

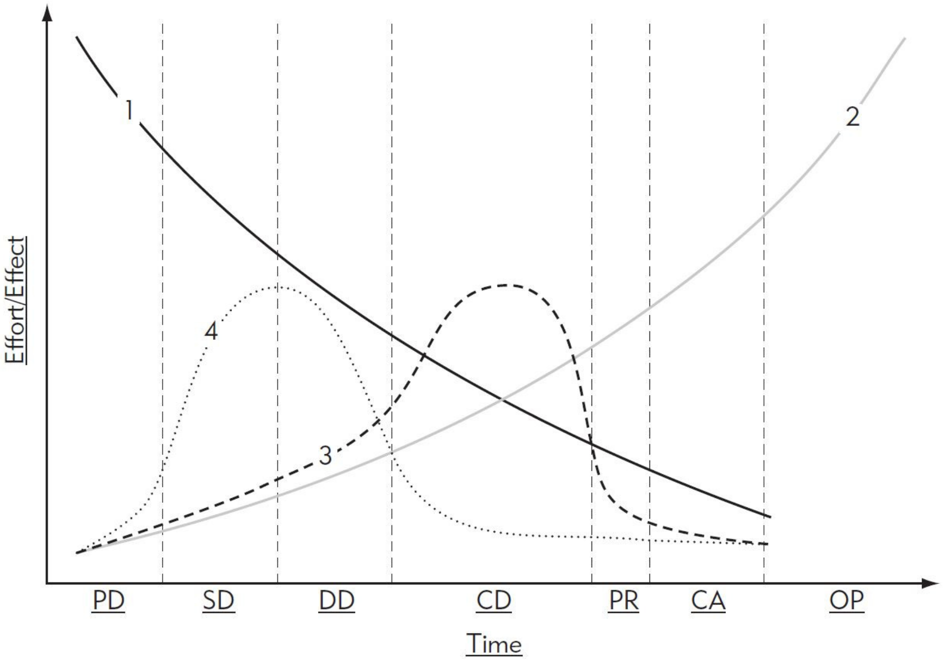


Figure 3.4: Costs of design changes throughout different design stages (Eastman et al., 2011)

1: Ability to impact cost and functional capabilities
2: Cost of design changes
3: Traditional design process
4: Building Information Modeling design process

PD: Pre-design
SD: Schematic design
DD: Design development
CD: Construction documentation
PR: Procurement
CA: Construction Administration
OP: Operation

3.3.2. Existing building performance software

As a result of the intended interoperability and the known potentials of early performance simulations various building energy simulation programs are studied throughout scientific literature (e.g. Kim, Kim, & Seo (2012); Crawley, Hand, Kummert, & Griffith (2005); Moon, Choi, Kim, & Ryu (2011); Clarke & Hensen (2015); J. B. Kim, Jeong, Clayton, Haberl, & Yan (2015)). Moreover, the absence of a standard exchange format and limitations in synchronization between data formats and software interfaces often cause users to be distrustful of energy assessment results. Each software packages delivers different results derived from the same building information (Crawley et al., 2005).

In order to get insights in the use of simulation tools it is valuable to study the nowadays market (April 2016) of commonly used BPS software and their characteristics. The U.S. Department of Energy (DOE) is maintaining an up-to-date listing of BPS software on the Building Energy Software Tools Directory (BESTD) website. This list (with currently 133 listed programs) is ranging from research to commercial software (DOE, 2016). As a result of this study to nowadays BPS software an organized overview is established and can be found in Appendix I. During this research all simulation packages are studied based on their data exchange formats, interoperability with CAD software and their known capabilities. Furthermore, including a certain software package in this list depends on its potential of being linked with BIM. According to J. B. Kim et al. (2015) linking BIM and energy simulation is mainly depending on their support of standard data schemes such as IFC and gbXML. As a result, diverse simulation packages are included; all widely known throughout the market of today. Moreover, VABI Elements is included due to the fact that this software package is used by Arcadis and other large companies throughout the Dutch market.

Additionally, building performance software mainly consists of two elements, an engine and an user interface. The engine contains all thermodynamic concepts in the form of equations, while the user interface eases the input and displays results. Furthermore, the user interface is able to provide multiple options to comply with the different needs of users (Maile, Fischer, & Bazjanac, 2007). Due to the fact that the Graphical User Interface (GUI) of DOE-2 and EnergyPlus are thought to be quite complex, their engine is used in combination with the GUI of other simulation tools. The general data flow principle between a simulation engine and GUI is illustrated in figure 3.5.

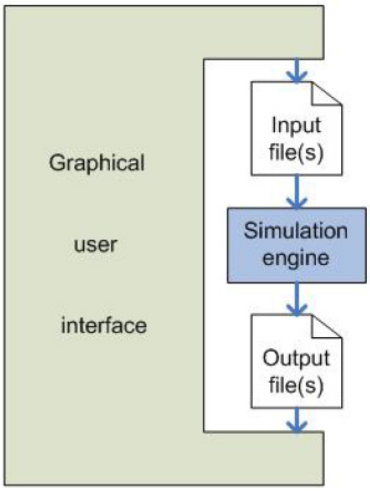


Figure 3.5: Connection between GUI and engine (Maile et al., 2007)

3.3.3. Comparison of exchange formats

Given the alternating use of the data exchange format IFC and gbXML it seems necessary and useful to study their differences and similarities when it comes to building simulation. The advantage of the gbXML schema is the support of many BIM and energy simulation tools (Moon et al., 2011). However, the IFC schema can be identified as the most developed data model for buildings in the AEC industry of today (Eastman et al., 2011). Moreover, according to Diaz-Vilarino, Laguela, Armesto, & Arias (2013) BIM based open standards such as IFC and gbXML are used to optimize the communication between the various stakeholders involved in different stages of the lifecycle of a building. Both standards represent building information, IFC represents the whole building project while gbXML only supports the information which is necessary for energy analysis. When comparing the level of information of IFC and gbXML it can be said that there is a downsize of information. IFC usually contains more information than gbXML does, while gbXML focuses only on the necessary information to comply with BPS. Additionally, when lowering the amount of building information it is important to keep in mind the specific information needs and the loss of information as well. A more in-depth look of this information requirements is provided in chapter 4 of this report.

Both the exchange formats IFC and gbXML are having their advantages and disadvantages when it comes to BPS. Large and complex IFC files may be rich and valuable, but on the other hand cause time consuming simulation runs or even crashing simulation software. The more compact gbXML schema may be widely used as exchange format, but may not be the most optimal format when it comes to building information. Moreover, the XML schema (gbXML) has a lower effort to implement, the schema is making use of a more familiar language and technology that EXPRESS (IFC) does. Dong, Lam, Huang, & Dobbs (2007) elaborated a comparative study of the IFC and gbXML informational infrastructures for data exchange. Both formats are studied in terms of geometry and their level of data representation. It is found that in terms of geometry, the IFC schema has the ability to represent any shape of a building, while gbXML only accepts rectangular shapes. Furthermore, the IFC schema makes use of a “top-down” and relational approach. This includes a relative complex data representation and often results in a large data file size (more information about data file sizes can be found in chapter 6). Subsequently, the gbXML schema embraces a “bottom-up” approach, which is accessible and flexible. This last approach has proven to be successful in offering web-based simulation services. More information on the file formats IFC and gbXML can be found in chapter 3.4.1 and 3.4.2 of this report.

3.4. Open data exchange

One of the problematic aspects of BIM is the interoperability among the diverse software packages in the different aspects of a design. For this reason open data exchange models were created to be used as standards for BIM software. Considering the extensive use of those standards in this thesis it is valuable to study the background, commonly used formats and their application in practice.

3.4.1. buildingSMART International

As mentioned before, buildingSMART is an international non-profit organization that develops open standards for open BIM. Their mission is to contribute to the sustainable built environment through smarter information sharing and communication using open international standards in the building and construction sector, private and public. BuildingSMART closely collaborates with the International Standardization Organization (ISO) for developing international, regional and national standards (buildingSMART, 2015). Concerned with the standards are data models, process definitions and dictionary terms, see the "Standard Triangle" in figure 3.6.

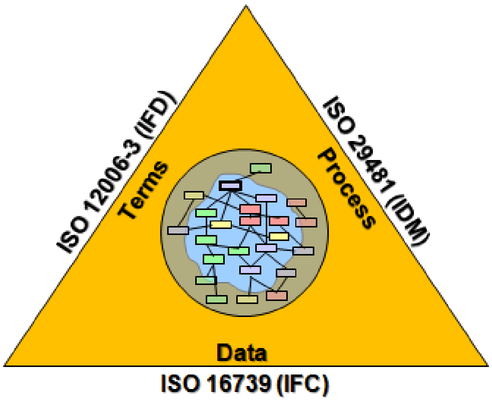


Figure 3.6: buildingSMART Standard Triangle (buildingSMART, 2015)

This "Standard Triangle" can be specified with the following terms:

- IFC (ISO-16759) – main buildingSMART data model standard;
- IDM (ISO-29481) – main buildingSMART process definition standard;
- IFD (ISO-12006-3) – main standard for buildingSMART dictionary terms.

IDMs can be used by documenting existing or new processes to describe the needed exchange information between parties. Those parties are able to work efficient if they know which and when different kind of information needs to be communicated (Karlshoj, 2011). Additionally, the process within buildingSMART for documenting implementation standards for the IFCs is to create a Model View Definition (MVD) for supporting a specific business process. These MVD are defining subsets of the IFC data model together with a software requirement specification (Hitchcock & Wong, 2011).

The International Framework for Dictionaries (IFD) is also known as the bsDD. This dictionary contains objects and attributes and is used to identify objects in the built environment with properties regardless of language. Similar to all buildingSMART applications the dictionary is open and international. To point out the value of the dictionary it is useful to look into a practical example. For example, the meaning of "Door" is not the same in all languages when it comes to the elements which are included (e.g. window frames, glass or wall-openings). The dictionary makes sure that the different definitions are known and miscommunication is prevented (BuildingSMART, 2014). This is guaranteed by assigning a Global Unique Identifier (GUID) to all concepts. By doing this it is possible to take away duplicates and synonyms so that multiple objects with the same meaning are not created. Figure 3.7 describes how an object (e.g. a window) can be described by a set of characteristics in IFD.

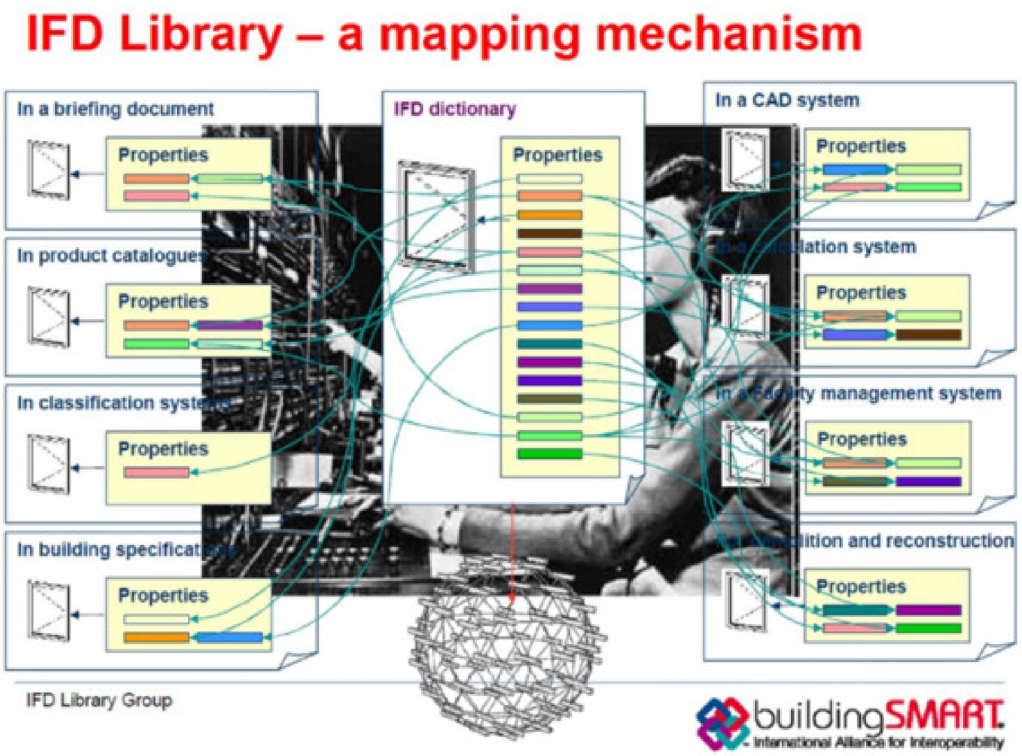


Figure 3.7: Graphic illustration of describing an object in IFD

3.4.2. Industry Foundation Classes (IFC)

The Industry Foundation Classes (IFC) are an open international standard for BIM data which is shared among numerous software applications used by various parties throughout the lifecycle of a building project within the AEC industry. In order to match the information needs of this industry, the early development stages of IFC were driven by the slow evolution of product data representation and exchange (ISO 10303) standard, which is informally known as STEP, Standard for the Exchange of Product Model Data. A part of the team that developed STEP participated in the definition of the IFC data model, but there is no precise analysis on the relation between the two standards (Borgo & Sanfilippo, 2015).

Furthermore, the specification of IFC includes terms, concepts and data specification items that are derived from various disciplines within the AEC industry. Terms and concepts are making use of plain English words and the specific data items within the data specification are following a naming convention, which is explained below. IFC has been structured into four conceptual layers which in total contain about 800 entity definitions, thousands of attributes and even more standardized properties augmented to the model schema in external property sets to represent information (Zhang, Beetz, & Weise, 2015). Figure 3.8 shows the data schema architecture with conceptual layers.

The naming convention is structured as follow according to buildingSMART (2015):

- The data item names for types, entities, rules and functions start with the prefix "Ifc" and continue with the English words in CamelCase naming convention (no underscore, first letter in word in upper case);
- The attribute names within an entity follow the CamelCase naming convention with no prefix;

- The property set definitions that are part of this standard start with the prefix "Pset_" and continue with the English words in CamelCase naming convention;
- The quantity set definitions that are part of this standard start with the prefix "Qto_" and continue with the English words in CamelCase naming convention.

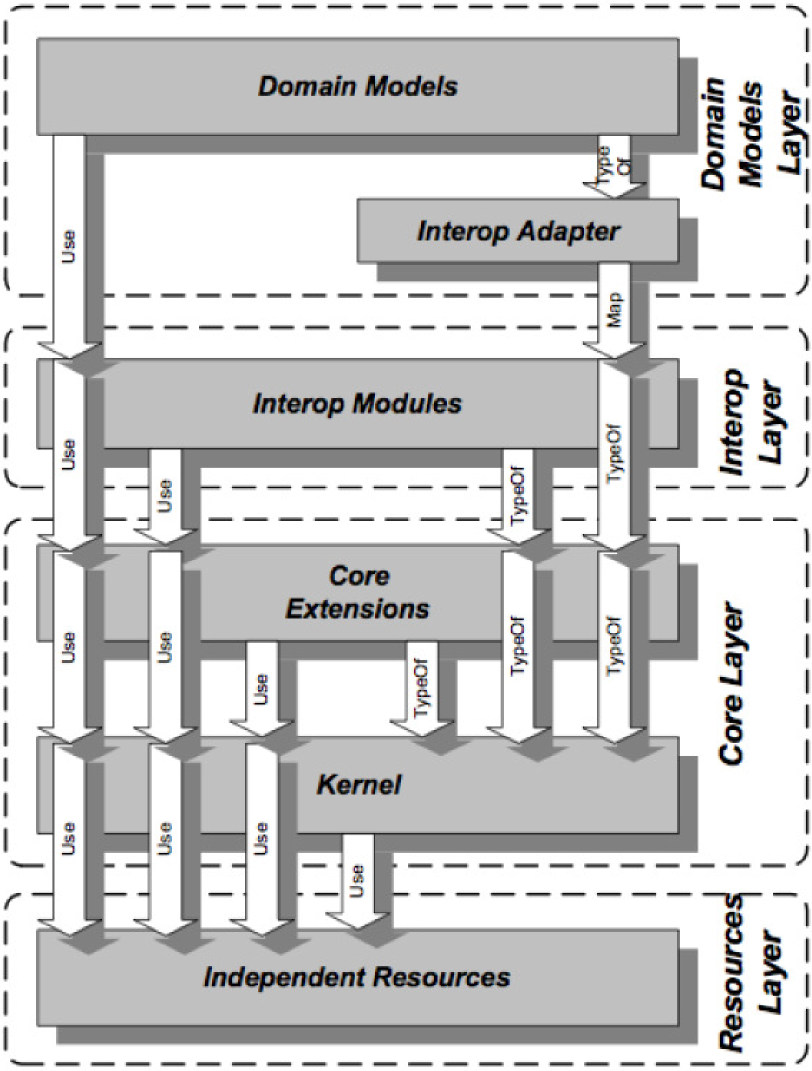


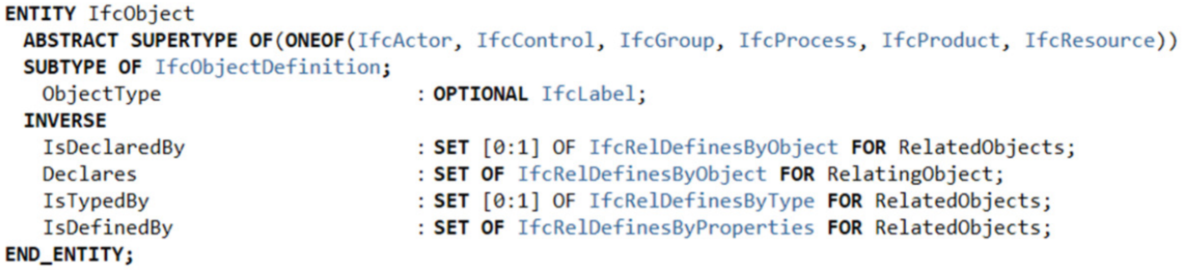
Figure 3.8: Data schema architecture with conceptual layers (Borgo & Sanfilippo, 2015)

As obtained from figure 3.8 above the current IFC release (IFC 4) distinguishes four conceptual layers, which each are subdivided in different schemas (buildingSMART, 2015):

- Resource layer: this is the lowest layer and includes all individual schemas containing resource definitions, those definitions do not include an globally unique identifier (GUID) and shall not be used independently of a definition declared at a higher layer;
- Core layer: this layer includes the kernel schema and the core extension schemas, containing the most general entity definitions, all entities defined at the core layer, or above carry a GUID and optionally owner and history information;
- Interoperability layer: this layer includes schemas containing entity definitions that are specific to a general product, process or resource specialization used across several disciplines, those definitions are typically utilized for inter-domain exchange and sharing of construction information;

- Domain layer: this is the highest layer and includes schemas containing entity definitions that are specializations of products, processes or resources specific to a certain discipline, those definitions are typically utilized for intra-domain exchange and sharing of information.

The IFC data model is making use of the EXPRESS language, which is developed within STEP. EXPRESS is platform independent and allows taxonomical classification, via classes, of the domain entities that share certain attributes. Within this language an IFC concept is introduced via the entity construct and classified with respect to other classes via “SUPERTYPE OF” and “SUBTYPE OF” partial ordering relation. To further explain this ordering, the example of the entity *IfcObject* is being set out in figure 3.9.



```
ENTITY IfcObject
 ABSTRACT SUPERTYPE OF(ONEOF(IfcActor, IfcControl, IfcGroup, IfcProcess, IfcProduct, IfcResource))
 SUBTYPE OF IfcObjectDefinition;
  ObjectType                    : OPTIONAL IfcLabel;
 INVERSE
  IsDeclaredBy                  : SET [0:1] OF IfcRelDefinesByObject FOR RelatedObjects;
  Declares                      : SET OF IfcRelDefinesByObject FOR RelatingObject;
  IsTypedBy                     : SET [0:1] OF IfcRelDefinesByType FOR RelatedObjects;
  IsDefinedBy                   : SET OF IfcRelDefinesByProperties FOR RelatedObjects;
END_ENTITY;
```

Figure 3.9: IfcObject specification in EXPRESS (Borgo & Sanfilippo, 2015)

Studying this entity shows that *IfcObject* is a subtype of *IfcObjectDefinition* and is the supertype of *IfcActor*, *IfcControl*, *IfcGroup*, *IfcProcess*, *IfcProduct*, and *IfcResource*. These subclasses are disjoint due to the use of the ONEOF construct. Furthermore, *ObjectType* is a direct attribute, while *IsDeclaredBy*, *Declares*, *IsTypedBy*, and *IsDefinedBy* are INVERSE attributes. In these last definitions the SET OF … FOR construct specifies the INVERSE attributes by providing a collection of elements, sometimes with indication of their minimum and maximum number and the related attribute (Borgo & Sanfilippo, 2015).

Lastly, the latest version of IFC is the IFC4 Addendum 1 version, released in July 2015 as the buildingSMART final standard. This version is an improved form of the popular IFC 2x3 version. The IFC4 Add1 is able to describe more specific details of a model. In context of building physics the IFC4 Add1 offers the possibility to describe different space boundaries. Moreover, the storage of HVAC information is improved as well (Remmen et al., 2015). However, this research will be making use of the IFC2x Edition 3 version. This release is being widely supported with scientific research and used by global software vendors.

3.4.3. Green Building XML (gbXML)

The increasing added-value of open data formats when improving the interoperability between BIM applications and energy simulation tools is acknowledged by multiple scientific researchers (e.g. Guzmann Garcia & Zhu (2015); Ham & Golparvar-Fard (2014); Cemesova, Hopfe, & McLeod (2015); Cheng & Das (2014)). In 1999 the development of Green Building XML (gbXML) was started by Green Building Studio (GBS). This resulted in a first version in June 2000 and more hereafter; the latest version (gbXML 6.01) was launched in November 2015. Furthermore, the gbXML schema is making use of XML (eXtensive Markup Language). This enables web based applications to automatically interact with each other, with no or minimal human intervention. When documenting in XML both computers and humans are able to interpret the information. This brings the opportunity to exchange information via internet easily and efficient (Cheng & Das, 2014). Furthermore, XML allows people to create their own customized language for exchanging information within their domains of interest. Therefore, the implementation of the actual data model or schema, with their associated semantics, can vary significantly (Dong et al., 2007).

The gbXML format facilitates the data transfer of building information stored in a BIM towards engineering analysis tools. Building information like building geometry, schedules, weather data, HVAC systems, lighting and thermal zone related data can be accommodated. This supports the interoperability between design software and engineering analysis tools and eliminates the need for time consuming plan take-offs. Nowadays, the gbXML format is adopted in the AEC industry by leading CAD-vendors such as Autodesk, Graphisoft and Bentley (gbXML, 2016b).

To get into the structure of gbXML Ham & Golparvar-Fard (2014) describes the main structure of the elements in gbXML schema with a simplified chart, see figure 3.10.

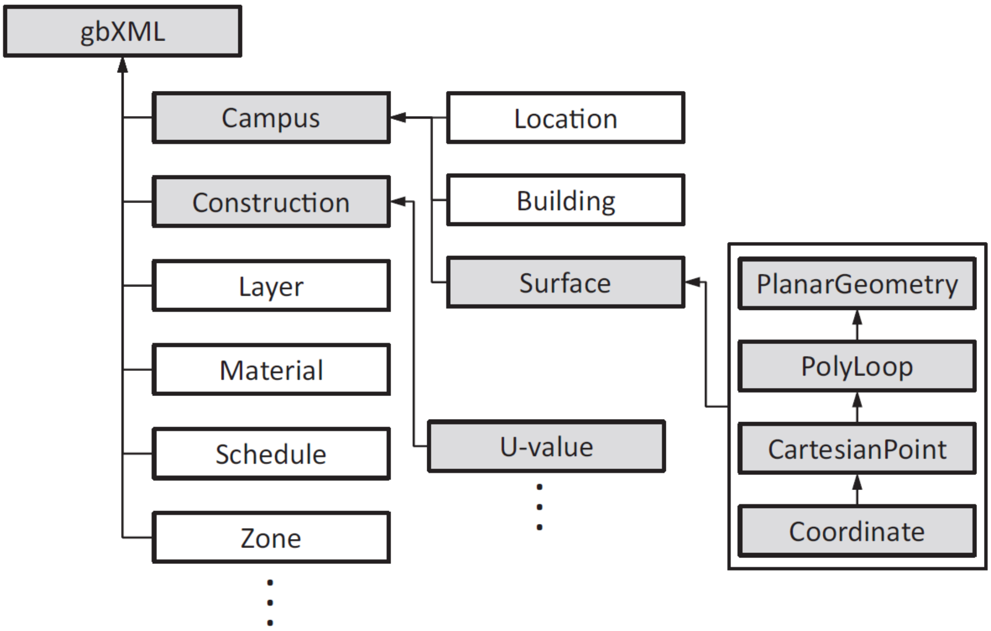


Figure 3.10: Structure of gbXML and main elements (Ham & Golparvar-Fard, 2014)

Firstly, all geometry information transferred from CAD tools are represented by the "Campus" element of gbXML. The child element "Surface" is representing all the surfaces in the geometry, those are each characterized with attributes "id" and "surfaceType" (e.g. surfaceType: "ExteriorWall"). Every "Surface" element has two representations of geometry, which are "PlanarGeometry" and "RectangularGoemetry". They are both carrying the same geometry information, this functions as a double-check whether the translation of geometry from the CAD tool is correct or not. The "PlanarGeometry" defines the area where the 3D surface polygon lies. The subtype, defined as the "PolyLoop" element, describes the polygon shape of the surface by making use of the "CartesianPoint" element and its multiple sub-elements "Coordinate". This last element contains the 3D coordinates (x, y, z) that shape the rectangular face (Ham & Golparvar-Fard, 2014). A more elaborated explanation of the gbXML structure, included elements (with IDs and Refs) and the geometry representation can be found in chapter 4.2.2, Appendix III and chapter 4.4 respectively.

4. Analysis

By conducting the literature study a foundation for the master thesis is established. The next phase of this research aims to analyze the concerned subjects in more detail; answer to sub-questions 2.1 and 2.2 will be provided. Within this chapter the current process between Building Information Modeling (BIM) and Building Performance Simulation (BPS) is mapped in order to point out bottlenecks, needed information and links. Hereafter, the desired process of this research is visualized; based on the current process. After studying this mappings the next step is to go more into detail regarding concerned model-parameters and BPS. The input specification for gbXML and model-parameters in IFC files are studied. When these analyzes are done it will be possible to start developing the suggested conversion tool.

4.1. Process mappings

In order to specify business processes both the current and desired process from design to simulation are studied. Two process maps are created by making use of the Business Process Model and Notation (BPMN). With this notation an understanding of internal business procedures is created in a graphical notation, this brings the opportunity to communicate these procedures in a standard manner.

4.1.1. Current method

First, the current process of design to simulation is studied and illustrated with BPMN. The entire scheme can be seen in figure 4.1. This mapping is created for a BIM project, having the desired process in mind. Distinguished are: Design parties, a BIM modeler and simulation consultant. Also, the concerned exchange formats are visualized alongside with the process.

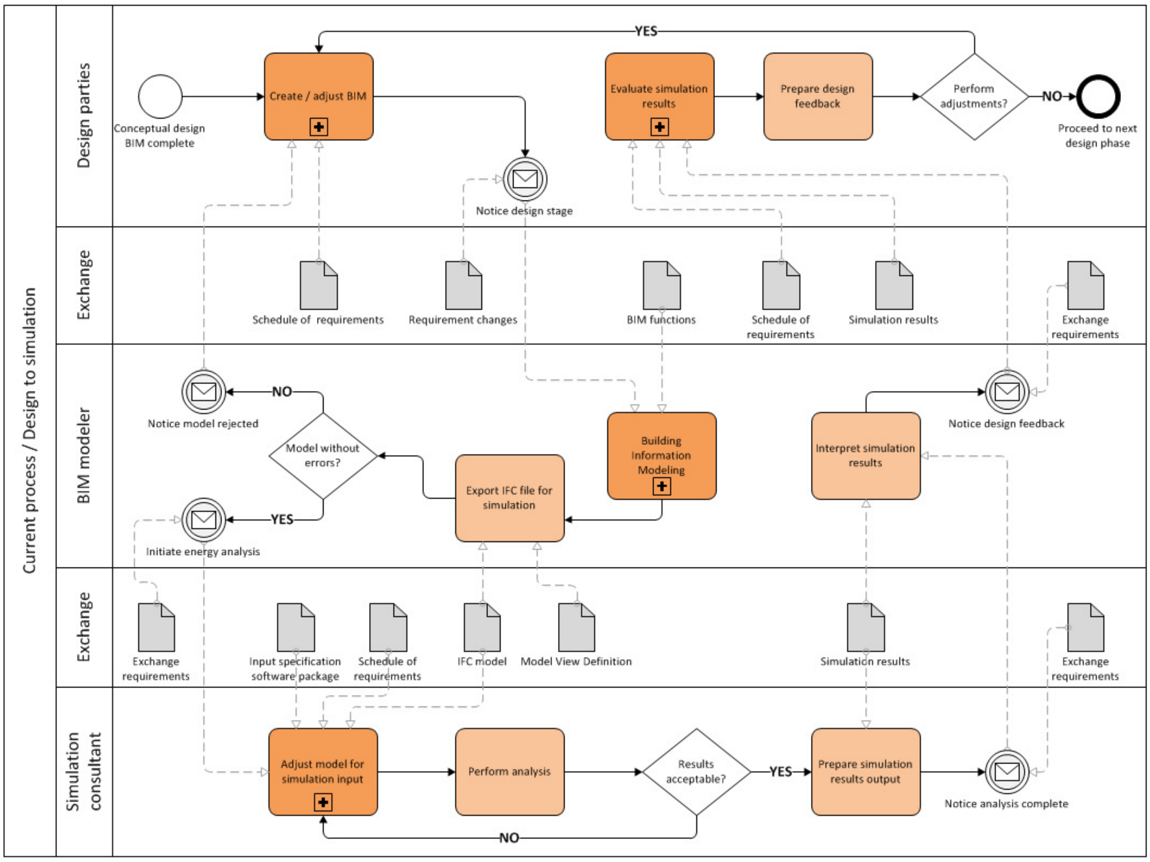


Figure 4.1: BPMN current process of design to simulation

The illustrated process describes the concerned tasks, stakeholders and information flows within the process of conceptual design and BPS. For this purpose, the intern business processes within the organization of Arcadis are used as a reference. Necessary information is gathered by interviewing multiple experts from the field. An overview of involved respondents and the used interview questionnaire can be found in Appendix II.

The visualized process starts with a conceptual stage of a BIM, created by involved designing companies within software of their preference. When starting the process of BPS a BIM modeler makes sure an appropriate IFC file is exported in a sufficient manner. Hereafter, the project is handed over to a simulation consultant by the means of exchange requirements (ER). These ER provides a description of the information in non-technical terms. Next, the simulation consultant starts specifying the input information for a selected BPS software of preference. For example, within the organization of Arcadis this input information is based on the needs of the simulation package Vabi Elements. During the input specification the simulation consultant is being supported by the earlier exported IFC file of the BIM modeler. Then, after completing the software input the simulation can be carried out. Subsequently, results are checked and prepared for exchange with other parties to function as design feedback. The BIM modeler interprets the simulation results and provides the designing companies with appropriate information. Finally, these designing companies receive the feedback and evaluate the simulation results. The design feedback eventually results in necessary changes to the design. Also, the whole process can be repeated as long as the design is exposed to changes.

Information obtained from expert interviews indicate that building geometry is regularly being re-modeled in the specific BPS software itself. Named reasons for this “habit” are often: Large IFC files and too much information with a too high complexity. An example provided by a practitioner (simulation consultant) from the field underpins this issue. At some point the specific consultant needed to perform early-design daylighting analysis. In order to avoid a unnecessary detailed and time consuming simulation he drew a few rooms and windows of the project in Vabi Elements and performed the analysis. This practical example supports the hypothesis elaborated in chapter 3.1. Here it is stated that simulation processes are partly based on knowledge and technical expertise of the concerned consultant, which brings the risk of “messy talking”.

4.1.2. Desired method

After mapping the current process of design to simulation it is possible to use this current business mapping to visualize the business process for the proposed method of this research; the method which is making use of the conversion tool. When comparing this mapping with the current process mapping similarities and differences will arise. For example, the participating stakeholders will not change in the desired process, but in return their tasks and exchange formats will. The main difference between de current process and the desired process will be the change of information exchange and input to the simulation software. The new created business process can be seen in figure 4.2. Additionally, while creating this new business process information is gathered by interviewing several experts. An overview of involved respondents and the used interview questionnaire can be found in Appendix II.

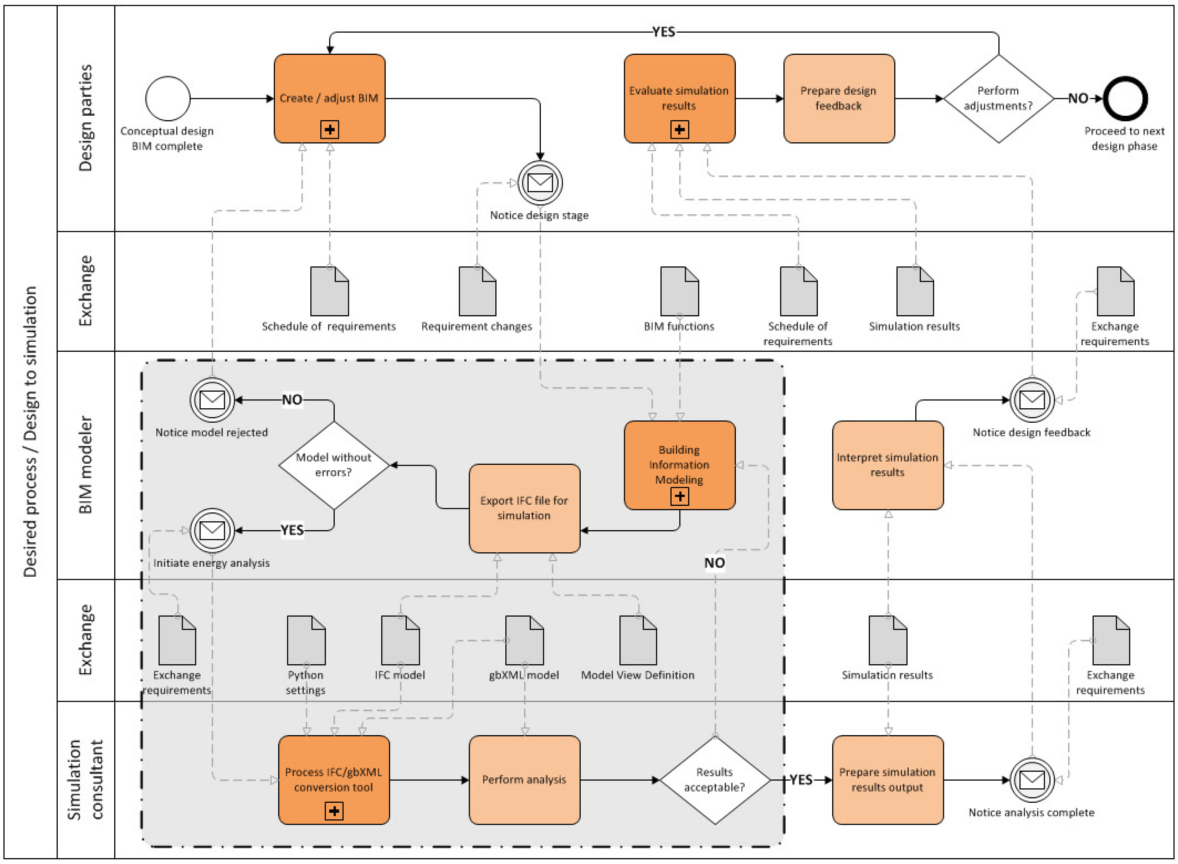


Figure 4.2: BPMN desired process of design to simulation

The use of the desired conversion tool is visualized within the grey box seen in figure 4.2. In the current process the input specification to BPS is done manually, but in this desired process the proposed IFC to gbXML conversion tool is used. This tool makes use of Python scripting and settings in order to specify the needed input. The program extracts the needed information from the IFC file and writes this out in gbXML format. Subsequently, this gbXML file functions as input for the specific simulation package (more information on this conversion is provided in chapter 5 of this thesis). By making use of the new conversion tool instead of manually input by the simulation consultant it is possible to eliminate the risk of “messy talking” as stated earlier.

In addition, after performing the analysis the results are checked on errors and relevance. If the results are not acceptable the process returns to the BIM modeler who started the process with exporting an appropriate IFC file. His task is to deliver a sufficient IFC file which can be understood and translated by the created Python script (see chapter 4.3.2 for more information on this). As a result the BPS software is provided with a valid gbXML file. The ER for this part are further explained in chapter 4.3. Then, as visualized in the current business process, the simulation can be carried out by the selected software package. The results of the simulation are checked and prepared for exchange with other parties to function as the intended design feedback. Again, the BIM modeler interprets the results and provides the designing companies with needed feedback. Moreover, the designing companies receive the feedback and evaluate the simulation results. This process results in necessary changes to the simulated design. Similar to the current business process, the above described process can be repeated as long as the design is subject to changes.

4.2. Principles of energy simulation and the gbXML schema

Predicting the energy performance and thermal comfort of buildings can today be done by making use of several BPS software packages. Each of those packages aim to create an understanding of how a given building operates according to certain criteria, also comparisons of different design alternatives are enabled. With regard to sub-question 2.2 it is necessary to create an understanding of the input data, basic principles of energy simulation and the information in the gbXML schema.

4.2.1. Simulation input data

As studied in chapter 3.3.2 of this report various simulation packages exist throughout the market of today. These packages each need input data in order to provide accurate energy simulation output; results can only be as accurate as the supplied input data is. For many studied simulation packages there can be said that they are making use of the same requirements with regard to simulation input data. The process of BPS is illustrated in figure 4.3 below. In this figure the needed input data is broken down into four types of data: Building geometry, HVAC systems, weather conditions and internal loads.

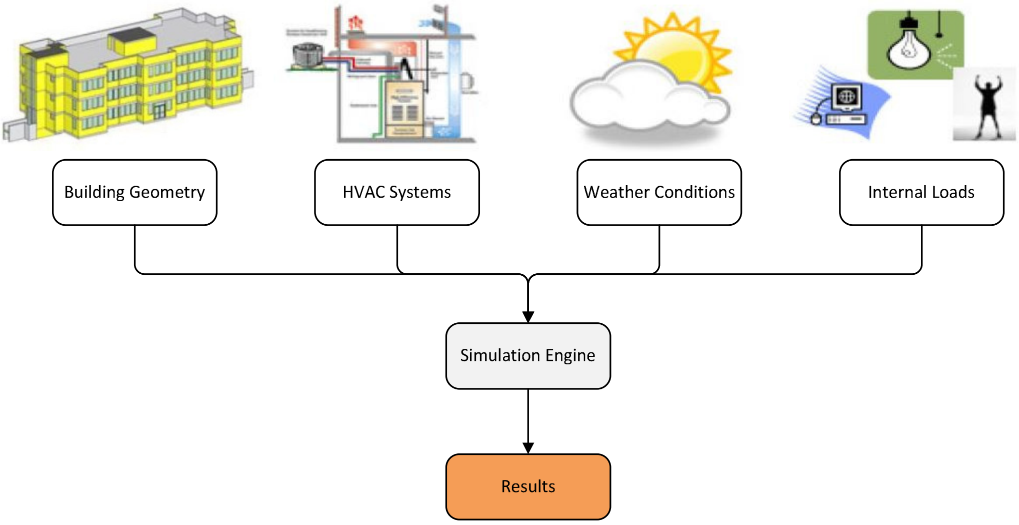


Figure 4.3: Data flow for energy simulation (Maile et al., 2007)

As clarified in figure 4.3, input data for energy simulations require not only a model of the building's geometry or HVAC systems (e.g. insulation, windows, foundation and walls), but also accurate environmental and occupant behavior data. This includes weather conditions such as humidity, wind speed and external temperatures over various time periods throughout the year. Furthermore, occupant behavior input data supports the building's internal electricity, heating and cooling loads such as lighting devices, electronic equipment caused by users.

Moreover, the basic input for energy simulations is building geometry which is created by CAD tools. Additionally, geometry of a building created by an architect is different from a model required for energy simulation. In an architectural model the spaces (e.g. rooms) are separated by walls, while in energy simulation modeling such spaces are known as thermal spaces and are separated by thermal space boundaries, which are not necessarily the same as the walls in the architectural model (see figure 4.4). Also, models for energy simulation are basically a simplified view of the architectural model. The main difference is the use of "space" and "wall" elements by architects and on the other hand the entities "boundary" or "thermal space boundary" in energy simulation models. Subsequently, it is possible to aggregate architectural spaces into thermal spaces (e.g. the division of large open offices into multiple thermal spaces). This division of architectural spaces is based on the thermal perspective where spaces with the same or very similar thermal characteristics are combined into one. In order to do so, the space boundaries define the interface between a thermal space and its surrounding boundaries for energy equations. Correspondingly, only the part of a wall that represents an actual boundary needs to be assigned to the space (Maile et al., 2007). The use and potential of space boundaries is studied in chapter 4.3.1.

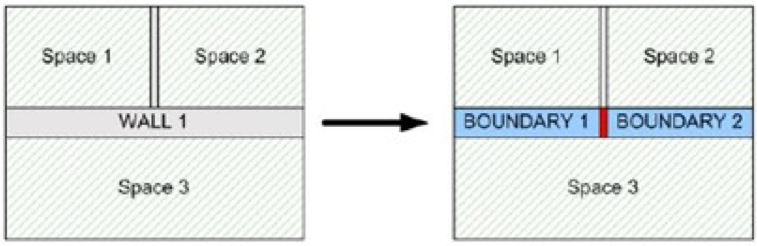


Figure 4.4: Difference architectural walls (left) and thermal space boundaries (right) (Maile et al., 2007)

Freestanding walls or columns can be ignored in thermal models most of the time. This is due to the fact that there is no difference in temperature between the exterior surfaces if those belong to the same thermal space. Furthermore, slabs and walls which are external, or which do not relate to a particular space, can be ignored from heat transfer perspective. They however need to be converted into shading objects if applicable, shading devices can influence an energy model if the solar loads in a space are effected. Lastly, curved surfaces can regularly not be represented in thermal simulation engines and are for that reason represented by a number of plane surfaces (Maile et al., 2007).

During early design phases both the architectural and thermal model can sometimes be close to identical. However, differences increase while the level of a design progresses and more details are known. Besides the building geometry the input specification for simulation software consists of HVAC systems, weather data and internal loads. All of this information is expected to be present in a developed phase of the design process (usually not in the conceptual stage). The internal loads include energy consumption by occupants and represent the actual usage of a space within a building and the behavior of its occupants. The modeling of HVAC systems is accommodated significantly by all energy simulation packages. Modeling a representation of real HVAC systems within a building can be challenging, so the operating schedule is a key input for this purpose. Operating schedules define the behavior of HVAC components (e.g. business hours, night-time or peak hours). Subsequently, the weather conditions are having impact on the simulation results as well. This weather data includes basic location information such as for example: Name, country, latitude, longitude, time zone, elevation and peak hot and cold temperatures. This data does not reflect conditions of a specific year, but provide statistical reference for the typical weather parameters of a specific location. Weather data files are being created for design purposes for an increasing number of cities and regions around the world; files can be retrieved with BPS software during simulation input. The internal loads and occupant behavior are strongly related with weather information and climate. Also, assumptions have to be made with regard to the quantity of internal loads in a given space during performance simulations (Maile et al., 2007).

The previous mentioned parameters provide basic input data for performing whole-building energy simulations. Different simulation engines or software packages may require additional input parameters to enhance accuracy. Additionally, the type of BPS software may influence the results of a specific simulation as well; different simulation engines or BPS software can result in diverse outcomes. Key to this discrepancy are internal calculations of simulation engines and fixed settings of the BPS software.

4.2.2. The gbXML schema

After specifying the input data for BPS the next step is to have a more in-depth look at this building information. As studied in chapter 3.3.3 the proposed open exchange format for data transfer towards simulation software is the gbXML schema. The next chapter studies the representation of building information in this schema. This is done by making use of scientific literature and a conducted interview with Stephen Roth (President of the gbXML Board of Directors), see Appendix II.

Elaborating on the gbXML schema, a study by Ham & Golparvar-Fard (2014) aims to automate the association and updating of energy performance information with BIM elements in the gbXML schema. This is done by using real-world buildings and automatically associate actual thermal properties with BIM elements and update those in the gbXML file. By doing so, the gap between architectural information in BIM and the actual data needed for energy performance simulation can be shortened. Technically updating the gbXML file needs extensive understanding of the objects and properties within the schema. Therefore, the study makes use of the XML Document Object Model (DOM). DOM is a language-neutral interface which enables dynamic access and updating of the content of a XML document. This presents XML documents as tree-structures and defines the objects and properties of all concerned elements, and a related interface to access and manipulate them. An example of their proposed method is being described, this resulted in an updated “U-value” within the “Construction” element. Here, the method “getElementsByTagName()” is used to search for all elements in the gbXML sub-tree which have a certain entry. This can be an entry such as the element “Construction” based on the attribute “constructionIdRef” of the gbXML-element “Surface”. Additionally, within the gbXML schema the geometrical information and thermal properties for wall and door components are located in the “Surface” and “Construction” elements respectively. In case of window components the geometrical and thermal information is located in the “Surface” and “WindowType” elements respectively. To find the element “Construction” that is associated with exterior building elements, first the attribute “constructionIdRef” of the element “Surface” that is characterized by “ExteriorWall” for attribute “surfaceType” needs to be extracted. Next, the element “Construction” that its attribute “id” is identical to the attribute “constructionIdRef” of the element “Surface” needs to be located. Lastly, in the queried element “Construction” the content of the sub-element “U-value” is changed to the actual value that is derived earlier.

With the aimed purpose of this research and the concerned sub-questions, the associated gbXML elements with their properties need to be further clarified. In order to do so an overview of included gbXML elements, with “id” and “ref” attributes is created in Appendix III. In this structure, corresponding “id” and “ref” attributes are indicated with the same color. This enables the ability to link for example a “space” to its relating “buildingStorey” or properties to a building element. To conclude, together the described “id” and “ref” attributes establish an important feature of the gbXML schema.

4.3. Space boundaries and IFC model parameters

After introducing the value of implementing thermal space boundaries in IFC models for energy analysis, the next step is to further elaborate on this and position the practice within the aim of this thesis. Therefore, this chapter discusses different definitions, levels and representations of space boundaries by making use of scientific literature. Besides space boundaries, this chapter zooms in on IFC model parameters and the export of those from CAD applications. Necessary IFC elements are set out and studied with regard to building performance simulation (BPS) by making use of Unified Modeling Language (UML). During this process Autodesk Revit 2016 is used as standard, this because of the use of Autodesk Revit 2016 within the organization of Arcadis.

Building information is stored differently throughout various construction domains and their software, each define their own data structures and file formats. CAD software describe building information in an architectural view where exterior and interior shapes of walls are composing spaces. Besides the CAD tools there are energy simulation tools which define geometry information in a thermal view, see figure 4.5b. Simulation tools sub-divide architectural view spaces into several zones for energy analysis, see figure 4.5a. Simulation software usually makes use of perpendicular heat transfer directions, this means that two or three dimensional heat transfer is being ignored most of the time. Therefore, building performance software simplifies geometric information of a space into two-dimensional planes. Lastly, the use of space boundaries play an important role in linking the architectural and thermal views, see figure 4.5c (Ahn, Kim, Park, Kim, & Lee, 2014a).

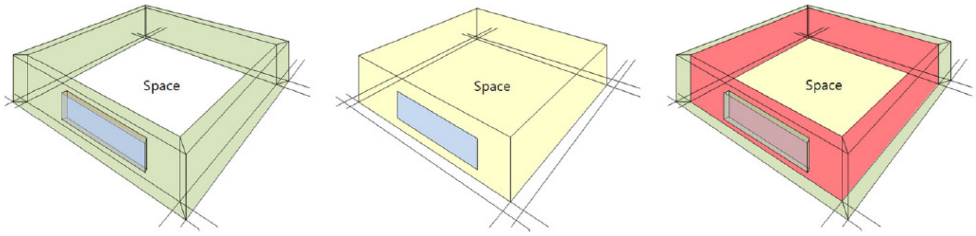


Figure 4.5: Architectural view (a) and thermal view (b) of a space and space boundaries (c) (Ahn, Kim, Park, Kim, & Lee, 2014b)

4.3.1. Types of space boundaries

A space boundary defines boundaries for spaces and relationships between spaces and the building elements (Ahn et al., 2014b). In general space boundaries can be of support for various tasks: Energy calculation, lighting calculation, indoor navigation, quantity take-off, and facility management. Therefore, several kind of space boundaries can be distinguished, also known as “levels”. All levels should be defined as simple, clearly and redundant free as possible. The known levels are: 1st level space boundaries and 2nd level space boundaries. Besides those there are some special types of 2nd level space boundaries known as well. The 1st level space boundary does not take into account heat flow between adjoining spaces and makes use of the architectural view (figure 4.5a). On the other hand, the 2nd level space boundary makes use of the thermal view and considers the heat flow (figure 4.5b) (Ahn et al., 2014b).

Weise, Liebich, See, Bazjanac, & Laine (2011) studied the application of space boundaries and especially their use for energy performance analysis. With the “Implementation Guide” it is intended to provide guidance to software vendors looking to support the import or export of space boundaries for energy analysis in an IFC model. Here, space boundaries are being described as: "*Virtual objects used to calculate quantities for various forms of analysis related to spaces or rooms in buildings*". These space boundaries provide information needed for calculating the energy flow between a space and other spaces or the outside air. Moreover, differences between space boundaries are caused and influenced by "what is on the other side". Below the distinguished types are set out; based on the "Implementation Guide" by Weise et al. (2011).

1st level space boundaries (no influence)

First level boundaries are used for quantity take-off, facility management, describing surfaces of elements, and cannot be directly used for thermal analysis. This level does not include the influence of surrounding spaces, see figure 4.6.

- 1st level space boundaries do not consider any change of material in the bounding building elements, or different spaces behind a wall or slab;
- 1st level space boundaries are differentiated in two ways: Virtual or physical and internal or external, or undefined (partly inside and outside);
- 1st level space boundaries create a closed shell around spaces if the space is enclosed, and include overlapping boundaries representing openings (filled or not) in the elements.

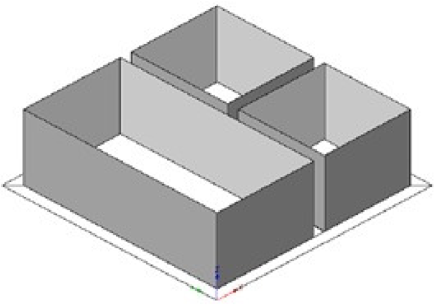


Figure 4.6: Space boundary level 1 (Häfele, 2010)

2nd level space boundaries (influence)

The second level boundaries are used by multiple BPS software packages. Those packages require a surface view of the building that can be transformed into various simple topological models. This type of space boundary is mainly used for energy analysis, lighting analysis, and fluid dynamics software. Subsequently, this level includes the influence of surrounding spaces, see figure 4.7.

- 2nd level space boundaries take into account differences in materials or material assemblies (e.g. a wainscot or paneling on the lower portion of a wall);
- 2nd level space boundaries include differences in spaces or zones on the other side of the building element (or virtual boundary) represented by the space boundary (e.g. two different spaces on the other side of a wall);
- 2nd level space boundaries are differentiated in two ways: Virtual or physical and internal or external, an element which is both internal and external is split into segments;
- 2nd level space boundaries include both sides of a heat transfer through a building element. Therefore, this level can be used for thermal analysis software, but require that two adjacent surfaces are found and combined to form a single heat transfer surface;

- 2^{nd} level space boundaries create a closed shell around spaces if the space is enclosed, curved surfaces need to be segmented.

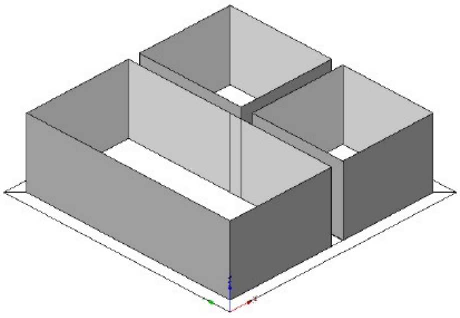


Figure 4.7: Space boundary level 2 (Häfele, 2010)

Special types of 2^{nd} level space boundaries

The second level of space boundaries can be subdivided into the following types:

- Type 2a, this type arises if there is a space on the opposite side of the building element providing the space boundary, see figure 4.8;

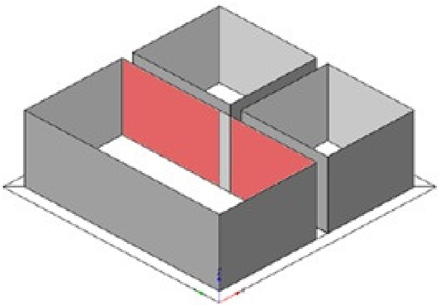


Figure 4.8: Space boundary level 2a (Häfele, 2010)

- Type 2b (also called 3^{rd} level), this type occurs when there is a building element on the opposite side of the building element which is providing the space boundary. Such boundaries are ignored in heat transfer calculations because the transfer is negligible, see figure 4.9;

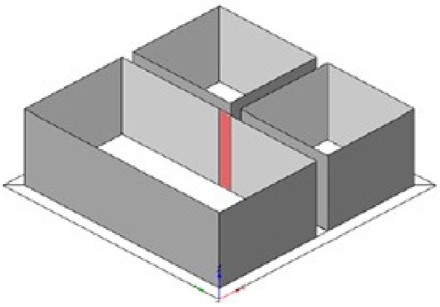


Figure 4.9: Space boundary level 2b (Häfele, 2010)

- Type 2c, this type occurs when building elements intersect and the two sides have different lengths. The extra length on one side or the other is defined to be type 2c. Those boundaries are ignored as well in heat transfer calculations because the transfer is negligible.

4.3.2. IFC model parameters

In order to have a sufficient IFC model as foundation for BPS it is necessary to identify required information and establish an optimal export from the CAD application, which is

Autodesk Revit 2016 during this research. Thus, building information needs to be exported from the CAD application in order to perform a sufficient energy simulation. However, not all information can be exported from CAD tools unfortunately. More information about the specific information flow between IFC and gbXML data is provided in chapter 5 of this report.

Additionally, an overview of the used export settings in Autodesk Revit 2016 can be found in Appendix IV. During this export all relevant elements with regard to building performance are extracted from the CAD model. Unfortunately, the current IFC schema is not yet able to include all information that is needed for energy analysis. For example, detailed information about mechanical systems applied to dynamic simulation tools (including EnergyPlus) is still unstructured in the IFC format (Ahn et al., 2014b). For more information about this last matter the graduation thesis of Groeneveld (2015) can be studied.

For the purpose of this research it is necessary to further identify IFC elements that need to be exported with regard to BPS. Studied by Hitchcock & Wong (2011) this information transformation consists of:

- Geometry;
- Thermal zoning;
- Internal loads and schedules;
- Construction and material thermal and optical properties;
- Shading surfaces;
- HVAC systems and components.

The required information for BPS can be set out by creating a Model View Definition (MVD). According to Jeong, Kim, Clayton, Haberl, & Yan (2014) this MVD defines a subset of the IFC schema that is needed to satisfy the Exchange Requirements (ER). Furthermore, MVDs can consist of a process model and a class diagram. The process model demonstrates the object mapping between BIM and BPS, and facilitates the required information during the translation. The class diagram represents required information during model translations using UML and aims on the information and object relationships. For the purpose of this research an IFC building model is presented by making use of the UML standard notations. The relationship of a building element and its placement can be seen in figure 4.10.

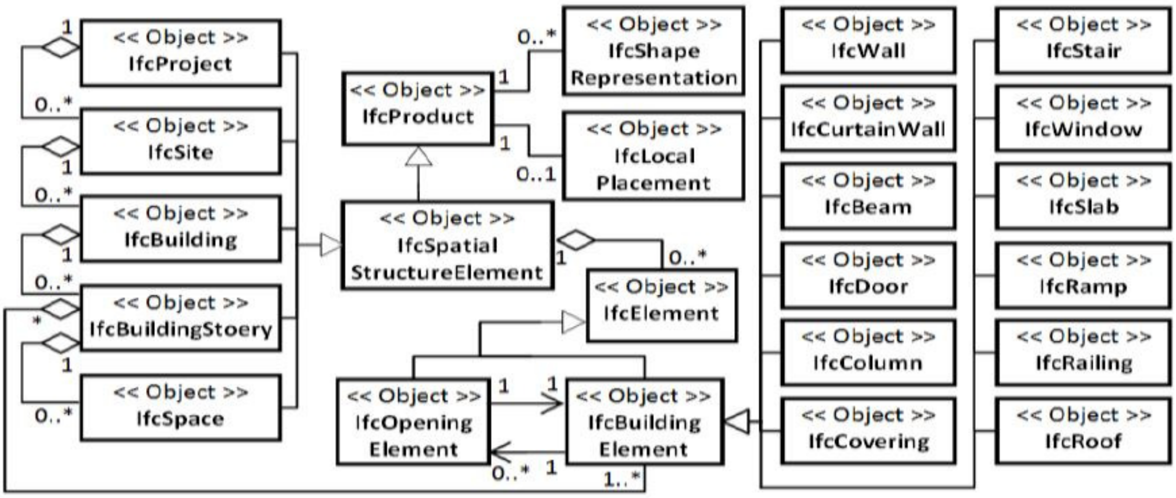


Figure 4.10: UML diagram of the IFC building model (M El-Mekawy, Östman, & Hijazi, 2012)

4.4. Geometric information

As described in chapter 4.2.1, the basic input for energy simulations is building geometry, which is created in (project differing) CAD tools. Hereafter, the specific CAD tool can export an IFC file which contains the modelled geometry. In the IFC 2x3 schema all geometric information can be represented, also there are many types of modelling representations.

4.4.1. Representing 3D objects in IFC

As stated above many types of modelling representations exist in IFC 2x3. Moreover, in order to model 3D solid models three main categories can be distinguished, see figure 4.11.

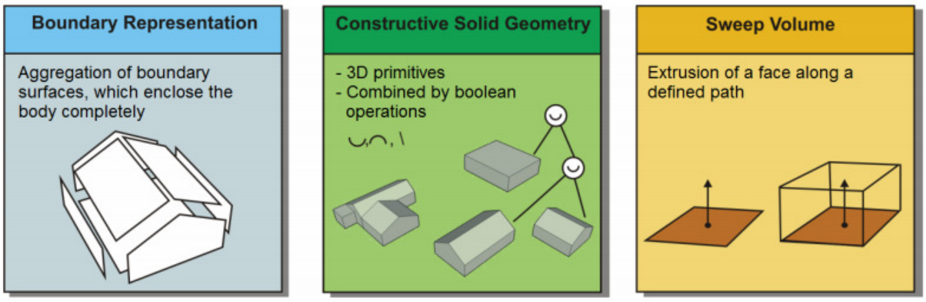

Boundary Representation
Aggregation of boundary surfaces, which enclose the body completely
Constructive Solid Geometry
- 3D primitives
- Combined by boolean operations
∪,∩,\
Sweep Volume
Extrusion of a face along a defined path

Figure 4.11: Three categories to represent 3D solids (Donkers, 2013a)

The first category visualized in figure 4.11 (left) is the Boundary Representation, often referred to as B-rep. This method is representing solids by making use of planar faces. A 3D solid is completely enclosed by boundaries and each boundary has its own surface. These surfaces mark the border between what is inside and outside the solid. The second method in figure 4.11 (middle) is the Constructive Solid Geometry (CSG). This method creates 3D solids by one or many Boolean operations. The operations needed to create a new solid can be represented with a tree structure where the "leafs" represent the base solids. An example is provided by Donkers (2013), where Boolean operations between a cube and a sphere are modelled, see figure 4.12.

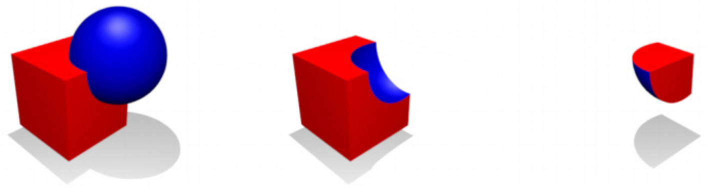


Figure 4.12: Boolean union (∪), difference (−) and intersection(∩) (from left to right) (Donkers, 2013b)

The third method to represent 3D objects in IFC is the Sweep Volume. This method uses a 2D profile and a path to compute the actual geometry of the body; see figure 4.11 (right). Here the 2D profile can be a primitive shape such as a circle, rectangle, or a polygon which contains holes. Next, the sweep can be a linear extrusion or a rotational sweep. Concluding, the last two methods are both representing implicit geometry, which means that parameters are provided to generate geometry in software packages. For example, an IFC viewer has to apply sweeping and CSG computations in order to visualize the elements in IFC. Finally, in practice most IFC models are built using the CSG or Sweep Volume method (Mohamed El-Mekawy & Östman, 2010).

4.4.2. Implicit and explicit geometry

As stated by El-Mekawy & Östman (2010) most IFC models are making use of CSG or sweep volumes where geometric information is stored implicitly. This implicit geometry of IFC is known by buildingSMART (2007) as attribute driven geometric representation. The method of attribute driven geometry defines location, orientation and dimensions of building elements that have shape (such as e.g. walls and windows). Object volumes are created through extrusion, revolution and cross section based sweep operations. Additionally, when making use of implicit geometry, elements and their location are not determined by three coordinates (x, y, z). Moreover, making use of these coordinates is called the three-dimensional Euclidean space (see figure 4.13), this technique provides explicit geometry.

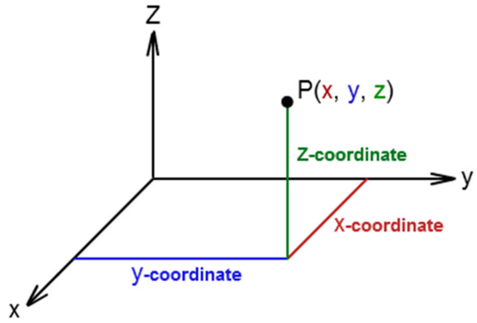


Figure 4.13: Three-dimensional Euclidean space (TutorVista, 2016)

For the purpose of this research it is needed to convert implicit geometric information in IFC files into explicit geometry that any CAD software or modelling package can understand; the gbXML schema in this case. As obtained in chapter 4.2.2, the gbXML schema is making use of the three-dimensional Euclidean space. Moreover, a study performed by Paul & Borrmann (2009) presents concepts for dealing with topological information in BIM. Here, the decomposition of a closed 3D object is being described by making a subdivision into a finite number of cells which create a topological space; *n-cell* is a cell of dimension *n.* So, a "*0-cell*" is representing exactly one point, also called a "vertex", an "*1-cell*" is called an "edge", a "*2-cell*" is called a "face" and a "*3-cell*" is called a "volume". See figure 4.14 for the cellular decomposition of a closed 3D object while using this technique.

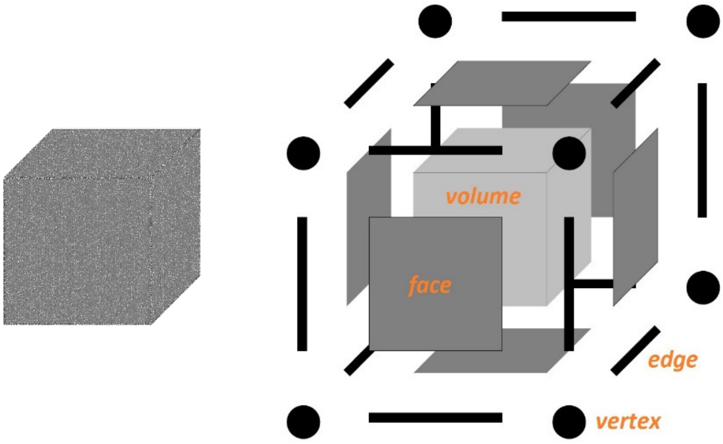


Figure 4.14: Cellular decomposition of a 3D object (Paul & Borrmann, 2009)

Now we know how a 3D object is described at a topological level, with topological information, it is possible to find corresponding elements in geometry. With these elements it possible to compose any shape. For example one "solid" will always consist of one "shell", six "faces", six "wires", twelve "edges" and eight "vertices". For an overview of the relationships between various geometric entities see table 4.1.

Table 4.1: Interlocking relationship of various geometric entities

Entity		Definition	
Solid		Part of 3D space bound by a shell.	
		Property	*Volume*
Surface	Shell	A collection of “faces” connected by some of the “edges” of their “wire” boundaries.	
	Face	Part of a surface bounded by a closed “wire”.	
		Property	*Area*
Curve	Wire	A sequence of “edges” connected by their “vertices”.	
	Edge	A shape corresponding to a curve, and bound by a “vertex” at each extremity.	
		Property	*Length*
Vertex		A zero-dimensional shape corresponding to a point in geometry.	
		Property	*Coordinate*

As stated earlier, the objective of this thesis is to develop a conversion tool which is able to translate IFC data into gbXML data. At this point it is known that the proposed tool needs to extract the implicit geometric information from IFC and convert it into explicit geometric information for gbXML; with three coordinates (x, y, z). Chapter 5 describes the process which eventually results in a new conversion tool, all necessary requirements and technical functions of the tool are visualized and explained in detail.

5. Tool development

After conducting both the literature and practical study the foundation for the last phase is complete. This final phase concerns the development and validation of the conversion tool, hereby making use of information obtained in previous phases and additionally Python scripting. First, the tool requirements and functions are set out in chapter 5.1. Hereafter, in chapter 5.2, the identification of necessary information for gbXML from the IFC schema is done; entity relationships and gbXML linkages are made clear. Then, in chapter 5.3, the created tool is validated so that finally its results and potential can be measured in chapter 5.4. While doing so answer is provided to sub-questions 3.1, 3.2 and 3.3.

As mentioned, the final part of this chapter is the validation of the conversion tool. This will be done by performing whole-building energy simulations for several pilot projects where a created gbXML file functions as input. The simulations are done with the building simulation software DesignBuilder (DesignBuilder, 2016). As described in Appendix I, multiple simulation packages exist in the market of today, in this case only one package is used in order to compare the project results mutually. During the whole process the below stated software and modules are used; divided by the development and validation part.

Development part

- Python - 2.7.11:
  - Python OCC - 0.16.3;
  - IfcOpenShell for Python 2.7 – Most recent developed version.

Validation part

- Schema GreenBuildingXML – Version 6.01;
- Autodesk Revit 2016 – 16.0.490.0;
- Revit IFC Exporter - Alternate UI 16.4.0.0;
- FZK Viewer x64 - 4.5;
- DesignBuilder - 4.7.0.027 (trial version);
- EnergyPlus - 8.3.0.001.

5.1. Requirements and scripting

Before being able to script the conversion tool and its functions it is needed to create an overview of the desired requirements. After establishing this overview it will be possible to elaborate on scripting and the core functions of the tool.

The functioning of the desired tool can be separated in four main parts, (1) reading the IFC file, (2) identifying necessary information for gbXML in the IFC file, (3) the conversion of implicit to explicit information, and (4) writing information according to the official gbXML schema. The conversion of implicit to explicit geometry is handled by IfcOpenShell, which for geometric computations relies on the Open CASCADE Community Edition, an open source 3D modelling kernel. Open CASCADE converts the implicit geometry in IFC files into explicit geometry that any software CAD or modelling package can understand (Dhillon, Jethwa, & Rai, 2014).

5.1.1. Tool requirements

Reaching a full understanding of requirements of a certain tool is in software development often handled with the MoSCoW Analysis. This is prioritization technique, of which the term itself is an acronym derived from the first letter of each of the four prioritization categories, divides requirements into the categories: “Must”, “Should”, “Could” and “Won’t”. These categories each describe their own requirements. “Must” describes a requirement that must be satisfied in order for it to be a success. “Should” represents a high-priority item that should be included if it is possible. Often a critical requirement, but one which can be satisfied in other ways if necessary. “Could” describes a requirement that is considered desirable but not necessary, only included if time and resources permit. Finally, “Won’t” represents a requirement that will not be implemented, but may be considered for future releases (Simues, 2009).

For the purpose of this thesis the main requirements of the desired tool are described by making use of the outlined MoSCoW method.

- **Must**:
    - Be compatible with IFC2x Edition 3 version files (commonly used today);
    - Process basic IFC building elements (e.g. walls and floors);
    - Convert implicit to explicit geometric information (to comply with gbXML);
    - Write valid information according to the official gbXML schema.
- **Should**:
    - Create gbXML files which are suitable for simulation in DesignBuilder;
    - Include gbXML construction types with each their layer and materials;
    - Process IFC window and opening elements;
    - Include thermal properties (e.g. IFC analytical property sets).
- **Could**:
    - Add the new conversion tool as a plugin to BIMserver (to create automatic building performance feedback after an IFC revision);
    - Create a GUI for the conversion tool (e.g. with PyQt4);
    - Create gbXML files which are suitable for varying BPS software packages;
    - Process a large set of IFC building element types.
- **Won’t**:
    - Be compatible with more recent editions of IFC (e.g. IFC4 Add1);
    - Create geometry if rooms are not assigned and 2nd level space boundaries are not included;
    - Include building information on O&M level.

5.1.2. Script documentation

After determining the main requirements of the tool it is possible to start scripting in Python programming language. During this process all desirable and necessary information for the new gbXML file is mapped and created based on the IFC file, this by iterating through IFC entities. Important to note is that each new created gbXML element needs to be valid according to the official schema; in this case the latest version (6.01) is used (gbXML, 2015). The matching between gbXML elements and IFC entities is explained in detail in chapter 5.2. Additionally, the functioning of the tool is set out by making use of two flowcharts. Each step of the conversion tool is visualized which ultimately results in a complete overview of the capabilities and functioning of the new tool, see Appendix V. Moreover, the earlier established tool requirements are used as objective during the process of scripting, specifically the categories “Must” and “Should”. As a result of this developing process the script of the new tool can be found in Appendix VI. The functioning and core capabilities of the script are being further explained in this section.

To start with, as one can obtain from the script and the first flowchart (Appendix V) the “gbXML” element represents the root element of the file. Hereafter, each gbXML element is separately created and appended as a child to earlier established elements. As long as the official gbXML schema allows it, elements are able to have multiple attributes attached and hold string nodes. Subsequently, attributes often represent an “id” string, which needs to be unique in the document. These “id” attributes are used in gbXML to refer to relating child or parent elements. Besides this, other attributes are for example “surfaceType”, “constructionIdRef”, “materialIdRef” or “unit”.

One of the core parts of the elaborated script is the conversion of implicit to explicit geometry; visualized in the second flowchart (Appendix V). As stated in the introduction part of this chapter, this conversion is handled by IfcOpenShell. This module uses OpenCascade to convert the implicit geometry in IFC (B-rep, CSG or Sweep Volumes) into explicit geometry (x, y, z coordinates), which is required in the gbXML schema. IfcOpenShell is capable of extracting “faces” and hereafter identify “wires”, “edges” and finally “vertices”. In this case geometry is derived by using the *IfcRelSpaceBoundary* entity. The reason for using this entity is that while developing the tool there has been found that exporting IFC with 2nd level space boundaries in Autodesk Revit 2016 results in “faces” positioned at the precise centerline of building elements. This phenomenon is important in order to make sure that created building elements together create a closed and water tight geometry. On the other hand, exporting IFC with 1st level space boundaries results in “faces” positioned at the outside (outer face) of building elements. For this reason, the use of 1st level space boundaries does not result in water tight geometry. Moreover, the *IfcRelSpaceBoundary* entity finds its origin while assigning “rooms” to the building model in Autodesk Revit. The borders of a room are determined by surrounding building elements; a room covers the area between building elements. Subsequently, an assigned room in Autodesk Revit automatically defines its associated boundaries, which finally can be exported as *IfcRelSpaceBoundary* entities to the IFC file. Concluding, an *IfcRelSpaceBoundary* entity basically represents a physical building element.

To look more specifically at the process of converting geometry, the script iterates from *IfcRelSpaceBoundary* via *IfcConnectionSurfaceGeometry* to *IfcCurveBoundedPlane*. This last entity represents geometry of which vertices are derived. As stated earlier, a “face” consists of “wires”, which consists of “edges”, which consists of “vertices”; each “vertex” represents a 3D point (x, y, z). Subsequently, the script uses Python OCC to iterate from a “face” to its “vertices”. This Python wrapper makes use of the “TopExp_Explorer” function to explore underlying topology. For “wires” it is however more convenient to use “BRepTools_WireExplorer”, this function takes into account the ordering and directly navigates from “wire” to “vertices”. Also, it prevents the tool of including “vertices” twice. Namely, the first and last “vertex” of each “edge” overlap each other. In conclusion, the scripting process results in “vertices” which represent the explicit coordinates needed in the gbXML schema as studied in chapter 4.4.2.

Besides the geometrical information another core function of the tool is to include analytical values which are wrapped inside for example an *IfcPropertySet* entity. Several properties are added to the gbXML file, these properties are valuable when carrying out whole-building performance simulations. In this case analytical properties are included based on the input specification of DesignBuilder. Examples of added properties are the: Volume, area, R-value, material thickness, U-value and absorbance of building elements. Moreover, the R-value represents the thermal resistance, while the U-value is the overall heat transfer coefficient. R-values of materials are established by dividing the thickness by the U-value. A more elaborated explanation of iterating through IFC entities can be found in chapter 5.2.1.

5.2. Entity relationships and gbXML linkages

Previous chapter provides the Python script with corresponding flowcharts of the new developed conversion tool. During the process of scripting, numerous relationships between gbXML elements and IFC entities are created. As mentioned earlier, accessing the needed information in IFC often results in iterating through multiple IFC entities. In order to elaborate on this, the following sections are visualizing the entity relationships in IFC and eventually providing an overview of each specific relationship with corresponding gbXML elements.

5.2.1. IFC entity relationships

As explained with scientific literature in chapter 3.4.2 of this report, the IFC data schema is making use of entities with each their attributes and inverse attributes. The IFC schema divides all entities in rooted and non-rooted entities. Rooted entities derive from *IfcRoot* and each have a GUID, on the other hand non-rooted entities do not. Also, these non-rooted entities only exist in a building model if referenced from a rooted instance directly or indirectly. By the means of this entity hierarchy it is possible to refer (back and forward) through all (related and relating) objects. This principle can be visualized by making use of figure 5.1 and figure 5.2.

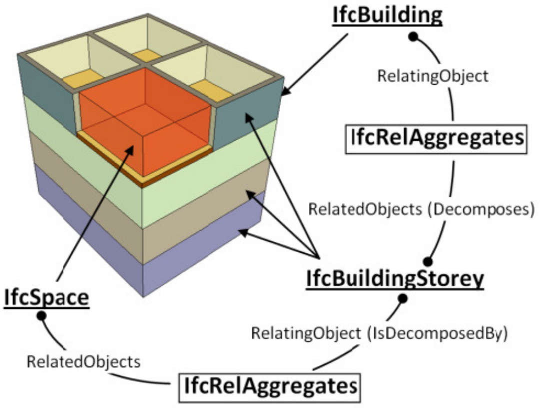


Figure 5.1: IFC building structure relationships (M El-Mekawy et al., 2012)

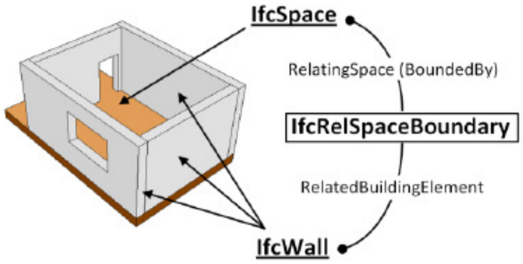


Figure 5.2: IFC building spaces relationships (M El-Mekawy et al., 2012)

As illustrated in figure 5.1 an *IfcSpace* decomposes *IfcRelAggregates*, this last entity holds the relating object *IfcBuildingStorey*. From this entity we can iterate to the *IfcBuilding* in the same way. Furthermore, chapter 5.1 of this report pointed out that the *IfcSpaceBoundary* entity plays a key role when it comes to the creation of explicit geometry within the tool. To elaborate on this, a space and its walls are obviously related to each other and can be referred to by making use of the discussed *IfcRelSpaceBoundary* entity, see figure 5.2.

Consequently, an *IfcSpace* is bounded by multiple *IfcRelSpaceBoundary* entities and each of these entities have their own related building element, which in this case is *IfcWall*.

Besides the relationships between discussed building elements such as *IfcBuilding, IfcBuildingStorey*, *IfcSpace* and *IfcWall*, more complex relationships are found as well. During the process of scripting, those relationships occurred while assigning properties to building elements. In this case the example of adding thermal properties to an *IfcBuildingElement* is used. In IFC files the Heat Transfer Coefficient (U) is stored as *NominalValue* in the *IfcPropertySingleValue* entity. The mapped relationship between the *IfcBuildingElement* and the desired *IfcPropertySingleValue* is visualized in figure 5.3.

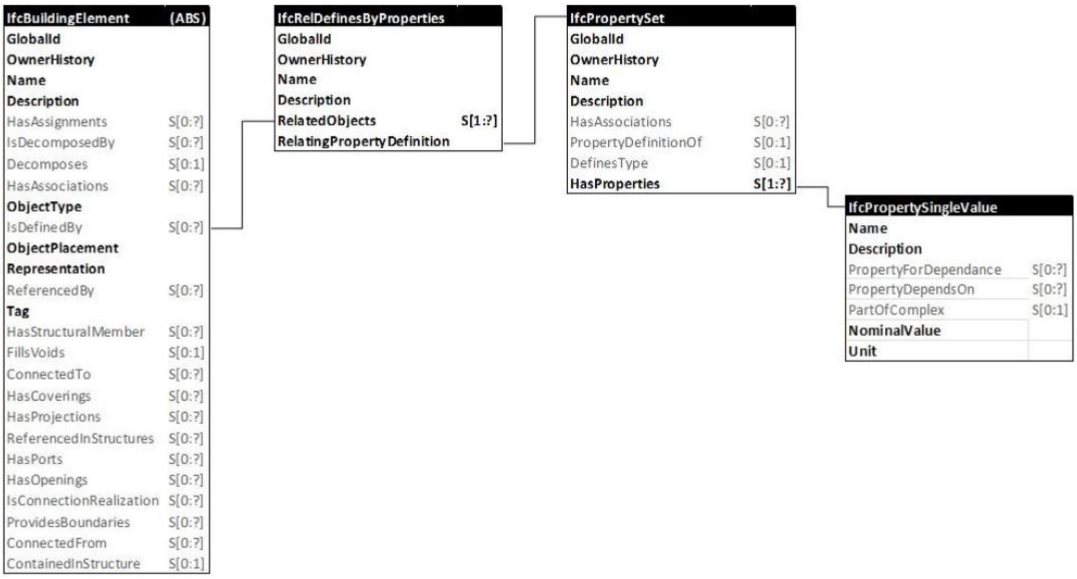


Figure 5.3: Mapping between IfcBuildingElement and IfcPropertySingleValue (buildingSMART, 2007b)

In addition, testing of the tool revealed that not all properties were stored in the *IfcRelDefinesByProperties* entity as stated above. Some building elements are linked using the *IfcRelDefinesByType* entity and one extra entity named *IfcTypeObject* (e.g. *IfcWindowStyle* or *IfcWallType*). These relationships are illustrated in figure 5.4 where the *IfcPropertySet* and *IfcPropertySingleValue* are linked to *IfcTypeObject* respectively.

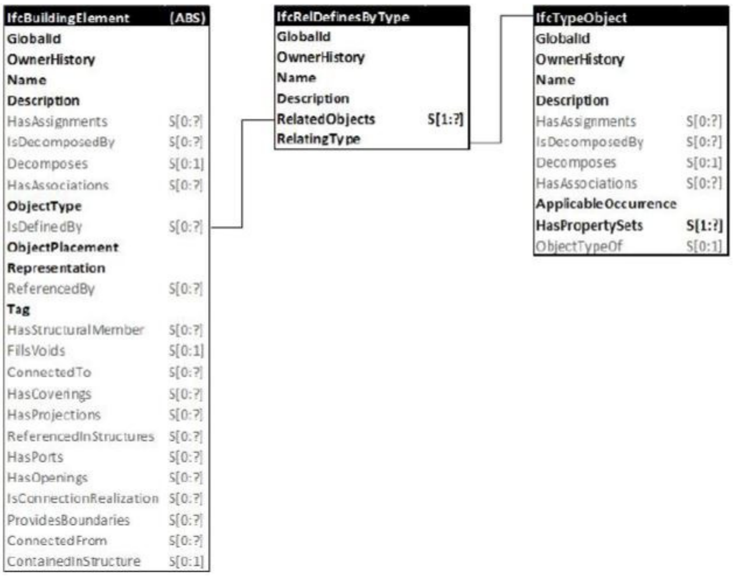


Figure 5.4: Mapping between IfcBuildingElement and IfcTypeObject (buildingSMART, 2007b)

5.2.2. Matching gbXML and IFC

At this point in the development internal relationships between IFC entities are studied. Also, the functioning of the developed tool is explained with the support of flowcharts and a description of its core functions. In addition, during the process of scripting many relationships are created between IFC entities and gbXML elements. The next section provides insights into the information source of each new created gbXML element. An organized way to do this is to create a table and point out how each gbXML element is matched with its corresponding IFC entity. Below in table 5.1 the overview of this matching process is provided. In this table all main gbXML parent elements are set out with each their corresponding IFC entity, this together with a brief description.

Table 5.1: Matching gbXML elements and IFC entities

gbXML element	IFC entity	Description
<Campus>	- *IfcSite*	- With an “id” attribute which is created by iterating to the *IfcSite* entity. From here the GlobalId (GUID) is extracted. - Child elements are: “Location”, “Building” and “Surface”.
<Location>	- *IfcSite* - *IfcPostalAddress*	- Child elements are: “Longitude”, “Latitude”, “Elevation”, “ZipcodeOrPostalCode” and “Name”; - Values of the child elements are created by iterating to the *IfcSite* and *IfcPostalAddress* entity. From these corresponding IFC attributes are extracted.
<Building>	- *IfcBuilding* - *IfcPostalAddress*	- With an “id” attribute which is created by iterating to the *IfcBuilding* entity. From here the GUID is extracted; - With a “type” attribute which is set to “unknown” in this case (multiple values are supported by the official schema, examples are “Office”, “Hotel”, “Courthouse”; - Child elements are: “StreetAddress”, “BuildingStorey” and “Space” (*see table 5.2 for more information*).
<Surface>	- *IfcRelSpaceBoundary* - *IfcRelAssociatesMaterial* - *IfcBuildingElement*	- With an “id” attribute which is created by iterating to the *IfcRelSpaceBoundary* entity. From here the GUID is extracted; - With a “constructionIdRef” attribute which is created by iterating to the *IfcRelAssociatesMaterial* entity. From here the GUID is extracted; - With a “surfaceType” attribute which is set in relation to the *IfcBuildingElement* entity. Based on the type of this element the “surfaceType” attribute is automatically set; - Child elements are: “Name”, “AdjacentSpaceId”, “PlanarGeometry”, “CADObjectId” and if applicable “Opening” (*see table 5.3 for more information*).

<WindowType>	- *IfcWindow* - *IfcReldefinesByProperties* - *IfcPropertySet* - *IfcRelDefinesByType* - *IfcWindowStyle*	- With an "id" attribute which is created by iterating to the *IfcWindow* entity. From here the GUID is extracted; - Child elements are: "Name", "Description", "U-value", "SolarHeatGainCoeff" and "Transmittance"; - Values of the child elements are created by iterating through multiple IFC entities. From these corresponding IFC attributes are extracted.
<Construction>	- *IfcRelAssociatesMaterial* - *IfcReldefinesByProperties* - *IfcPropertySet* - *IfcMaterialLayerSet*	- With an "id" attribute which is created by iterating to the *IfcRelAssociatesMaterial* entity. From here the GUID is extracted; - Child elements are: "U-value", "LayerId", "Absorptance" and "Name"; - Values of the child elements are created by iterating through multiple IFC entities. From these corresponding IFC attributes are extracted.
<Layer>	- *IfcRelAssociatesMaterial* - *IfcMaterialLayer* - *IfcMaterial*	- With an "id" attribute which is created by iterating to the *IfcRelAssociatesMaterial* entity. From here the GUID is extracted; - Child elements are multiple "MaterialId" elements, one for each *IfcMaterial* in the specific layer.
<Material>	- *IfcMaterial* - *IfcMaterialLayer* - *IfcRelDefinesByType* - *IfcWallType* - *IfcReldefinesByProperties* - *IfcPropertySet*	- With an "id" attribute which is created by iterating to the *IfcMaterial* entity. From here the IFC line id (#number) is extracted to be used as "id"; - Child elements are: "Name", "Thickness" and "R-value"; - Values of the child elements are created by iterating through multiple IFC entities. From these the corresponding IFC attributes are extracted or calculated.
<DocumentHistory>	- *IfcPerson* - *IfcApplication*	- Child elements are: "ProgramInfo", "CreatedBy" and "PersonInfo"; - Values of the child elements are created by iterating through the *IfcPerson* and *IfcApplication* entity. From these the corresponding IFC attributes are extracted.

Not all created gbXML elements are included, also the official gbXML schema contains more elements than handled by the developed tool. The reason for this "restricted" level of information is that during the tool development the focus has been set on the gbXML elements which are particularly needed in order to perform (early design) whole-building energy analysis; as explained in chapter 5.1.

Subsequently, in chapter 3.3.3 it is stated that the gbXML schema embraces a "bottom-up" approach. This means that each individual element is created and thereafter connected with each other by making use of "id" and "ref" attributes. Resulting in a hierarchical tree structure where child elements are appended to earlier created parent elements. In table 5.1 only elements which are in a "high" hierarchical position are included. While at the same time, needed and also (for BPS) important elements are nested in "lower" positions of the hierarchical tree. To elaborate on, those key elements are often nested within child elements of the “Building” and “Surface” element. Moreover, both of these elements have explicit geometry attached. So, in order to have a more in depth look at these for whole-building energy analysis key elements, table 5.2 is created.

Table 5.2: Matching key elements of the gbXML schema

gbXML element	IFC entity	Description
<BuildingStorey>	- *IfcBuildingStorey*	- With an “id” attribute which is created by iterating to the *IfcBuildingStorey* entity. From here the GlobalId (GUID) is extracted; - Child elements are: “Name” and “Level”; - Values of the child elements are created by iterating to the *IfcBuildingStorey* entity. From here the corresponding IFC attributes are extracted.
<Space>	- *IfcSpace* - *IfcBuildingStorey*	- With an “id” attribute which is created by iterating to the *IfcSpace* entity. From here the GUID is extracted; - With an “buildingStoreyIdRef” attribute which is created by iterating to the *IfcBuildingStorey* entity. From here the GUID is extracted; - Child elements are: “Area”, “Volume”, “Name” and “SpaceBoundary”.
<Area>	- *IfcReldefinesByProperties* - *IfcPropertySet*	- Values are created by iterating to their corresponding *IfcPropertySet*. From here the IFC wrapped value attribute is extracted.
<Volume>		
<PlanarGeometry>	- *IfcRelSpaceBoundary* - *IfcCurveBoundedPlane* - *IfcBuildingElement*	- Contains one “PolyLoop” element with four “CartesianPoint” elements, with each three “Coordinate” elements; - The coordinate values are created by iterating to the *IfcCurveBoundedPlane*. From here the Python IfcOpenShell (with Python OCC) module derives explicit coordinates.
<Opening>	- *IfcRelSpaceBoundary* - *IfcWindow*	- With an “id” attribute which is automatically created by the tool; - With an “openingType” attribute which is automatically created by the tool; - With an “windowTypeIdRef” attribute which is created by iterating to the *IfcWindow* entity. From here the GUID is extracted.

As stated, table 5.2 includes elements which have geometry attached. This explicit geometry is nested in the “PlanarGeometry” element of gbXML. This element contains one “PolyLoop” element with four “CartesianPoint” elements. Subsequently, each “CartesianPoint” element contains three “Coordinate” elements. As explained in chapter 5.1.2, within this research coordinates are created by iterating to the *IfcCurveBoundedPlane* entity. However, other entities are able to represent the geometry as well. Furthermore, testing revealed that relations between “Space” and “Building” elements is handled only on a logical level. For the developed tool this results in coordinates located at building level 0 (with often the z value zero as base). This issue is solved with scripting and so manipulating the z value while iterating from the IfcSpace entity to the IfcLocalPlacement, IfcAxis2Placement3D and IfcCartesianPoint entity. This last entity represents the placement of an IfcSpace, of which the value of the height can be added to the z coordinate in gbXML.

5.3. Tool validation

After studying created links between the scripted gbXML elements and their corresponding IFC entities the next step is to validate the capabilities and potential of the conversion tool. When performing scientific research a crucial part is the validation process, especially in the field of software development. Subsequently, new developed tools are usually first released as prototype and meant for scientific use only. Clearly such prototypes introduce limitations, large sets of operating instructions and lacking user interfaces. Moreover, by studying capabilities, barriers and opportunities it is possible to eventually deliver clear and relevant research results. Based on those results, conclusions can be drawn and recommendations for future research projects can be made.

5.3.1. Validation approach

As a result of the elaborated research a conversion tool which translates the IFC file format into a valid gbXML file format is developed. This tool is one of the first of its kind which increases the importance of conducting a broad validation process. No comparable other tools or relevant scientific research is found which it can be compared to. Therefore, the following three validation steps are prepared:

- **Step 1** (file creation):
    - Creation of Autodesk Revit projects with differing complexity;
        - 5 building models are tested;
        - The tool requirements are kept in mind.
    - Export the building model to an IFC2x Edition 3 version file;
        - With the in Appendix IV elaborated export options.
    - View the IFC file to study exported building information;
        - Make use of the FZKViewer (IFC import);
        - A search for included entities and an optical test.
    - Conversion tool: Convert the IFC file to a new gbXML file.
        - Run the developed Python script.
- **Step 2** (file validation):
    - Validate the gbXML file according to the official gbXML schema;
        - Make use of the online validator on the gbXML website;
        - Properly formed gbXML file and water tight geometry (gbXML, 2016a).
    - View the gbXML file to study exported building information.
        - Make use of the FZKViewer (gbXML import);
        - A search for included entities / properties and an optical test.
- **Step 3** (simulation performance):
    - Perform whole-building energy analysis;
        - Make use of DesignBuilder with the EnergyPlus engine;
        - Include thermal properties during simulations.

By means of these steps the validation process is divided into 3 parts: The file creation, file validation and lastly the simulation performance. These steps are proceeded for each of 5 tested building projects, resulting in products which can be found in Appendix VII. Summarized those products are: (1) the optical IFC test with the FZKViewer, (2) the optical gbXML test with the FZKViewer and (3) images, graphs and tables conducted from energy simulations in DesignBuilder (with EnergyPlus).

5.3.2. Test cases

As explained above, the developed conversion tool is a prototype at this point in time. An important reason for this is to scope the work possible in a MSc thesis; aimed objectives and introduced limitations are maintained during the project. Certainly it is important to identify the precise capabilities of the tool, this is done during the validation process. Subsequently, in order to have a broad and detailed look at the capabilities it is needed to make use of IFC file test cases that each differ in their complexity, see Appendix VII. This diverse complexity implies the use of differing building elements, shapes, materials, project sizes and entity relationships. Based on those characteristics the project complexity is labeled as "low", "medium" or "high". As a result, the following test cases are distinguished:

Test case 1:

- New created building model;
- Low complexity, 218 KB;
- 58 IFC entities, 477 relations;
- 2 storey building (2x ±56m^2), 10 rooms;
- Elements: Wall, floor and window;
- Basic materials.

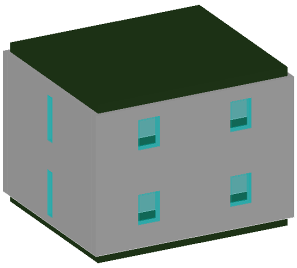


Figure 5.5: Test case 1

Test case 2:

- New created building model;
- Medium complexity, 307 KB;
- 95 IFC entities, 705 relations;
- 3 storey building (1x ±72m^2, 2x ±40m^2), 11 rooms;
- Elements: Wall, floor, window, roof and ceiling;
- Varying materials.

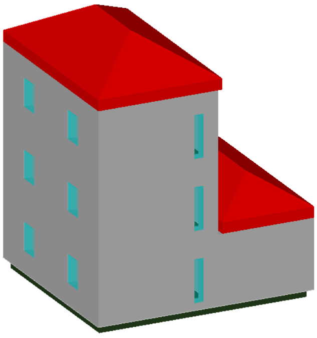


Figure 5.6: Test case 2

Test case 3:

- New created building model;
- Medium to high complexity, 543 KB;
- 98 IFC entities, 816 relations;
- 2 storey building with angled walls (2x ±187m^2), 12 rooms;
- Elements: Wall, floor, window;
- Varying materials.

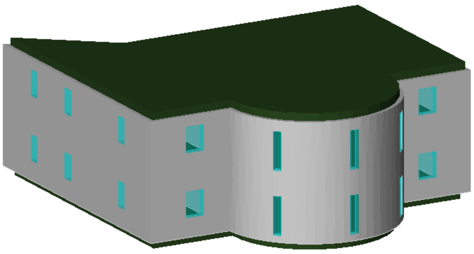


Figure 5.6: Test case 2

Test case 4:

- Office building model (architectural);
- High complexity, 13,0 MB;
- 1703 IFC entities, 15397 relations;
- 4 storey building (3x ±714m^2, 1x 200m^2), 67 rooms;
- Multiple building elements;
- Varying materials;
- Source: Arcadis.

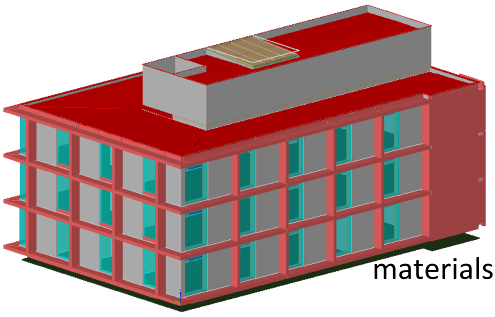


Figure 5.7: Test case 3 (Arcadis model)

Test case 5:

- Office building model (architectural);
- High complexity, 60,8 MB;
- 5943 IFC entities, 34615 relations;
- 3 storey building (3x ±1677m^2), 116 rooms;
- Multiple building elements;
- Varying materials;
- Source: (Autodesk, 2016) (rac_advanced_sample_project.rvt).

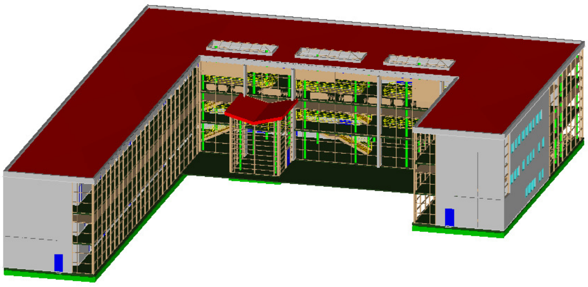


Figure 5.8: Test case 4 (Autodesk, 2016)

5.4. Results and shortcomings

The final part of the tool development covers the study of both conversion and simulation results. Each part of the validation process is discussed while making use of test case related examples. In addition, a comparison between the IFC and gbXML file is made. Thereafter, the shortcomings of the new tool are studied in order to indicate the potential, capabilities, shortcomings and opportunities. Moreover, the validation results are analyzed by making use of the content in Appendix VII. Lastly, during this study to shortcomings the tool is placed back in a broad perspective; the objectives and research questions of the research are cited.

5.4.1. Validation results

Previous section provides the validation approach and used test cases. Each of these pilot projects are exported to IFC from Autodesk Revit 2016, which results in the shown IFC models. These IFC models are studied and then converted to gbXML while making use of the new conversion tool. The new gbXML file is then validated according to the official schema and also optical tested and compared with the original IFC model. This optical test provides differences between the two models and their extent of information. Finally, when an appropriate gbXML file is created, the simulation software DesignBuilder is used to perform whole-building energy analysis. After proceeding this validation process the following results can be identified.

File creation and validation

To start with, within the scope of this research project it is important to appropriately create Autodesk Revit models and so correctly supply IFC and indirectly BPS. Moreover, testing revealed that the modeling process in Revit is a crucial part for the functioning of the conversion tool, without an appropriately created Revit model it is currently not possible to apply the tool. To clarify, in this case a correct Revit model means that (1) it is needed to assign rooms to the created spaces in the model. By appointing these rooms Revit automatically defines the corresponding space boundaries of each room; those are not directly visible in the software. Furthermore, (2) walls need to be connected up to the bottom of its upper floor, otherwise space boundaries are not sufficiently included and overlap with adjacent building elements. Then, (3) when including ceilings in the model it is needed to do this prior to assigning rooms; ceilings do represent space boundaries when a room is appointed. Moreover, the space between a ceiling and the upper floor is not part of the room.

Hereafter, an IFC2x Edition 3 version file can be exported from Autodesk Revit for each of the test cases. These exports resulted in widely divergent complexities and data file sizes, more information about those files is discussed in the file comparison section below. The subsequent step was to use the new developed conversion tool to convert the IFC file to a gbXML file. For Test case 1, Test case 2 and Test case 3 this process was successful; Test case 4 and Test case 5 introduced some bottlenecks which are discussed later on. Subsequently, each of the 5 created gbXML files were checked with the online validator on the gbXML website. This validator checked the file according to the latest schema and for valid water tight geometry. The tests *passes*, but still indicate some (less essential) *fails* with regard to the use of unique identifiers and the absence of optional gbXML elements (e.g. ShellGeometry definitions). To clarify, unique identifiers are supported by the conversion tool, but the GUIDs from IFC can be quite similar to each other. For example, the identifiers from the spaces consist of 27 characters of which only the last 3 are different, which the validator considers to be not unique.

File comparison

The FZKViewer is used to study the created gbXML file and perform a comparison with its original IFC file. For this comparison Test case 2 is used as example because apparently building information is being lost in Test case 3, Test case 4 and Test case 5, see Appendix VII. From the IFC file of Test case 2 the following entities can be handled by the tool: *IfcWallStandardCase*, *IfcSlab* (only floors), *IfcWindow* and *IfcCovering*. From these entities the conversion tool created gbXML “space” elements which are enclosed by gbXML walls with windows, floors and ceilings. Roof elements are not supported, mainly due to the use of complex (not rectangular) shapes. Moreover, the exported IFC entities can be found as “surface” elements in gbXML. Each of the created “surface” elements are having thermal properties attached, which are included while performing whole-building energy analysis. Those properties are located in “material” elements of the gbXML schema, which are each linked with its corresponding “surface” element through a “construction” and “layer” element and the use of “id” and “ref” attributes, this structure can be seen in Appendix III.

Additionally, when comparing both file formats there can be obtained that a large data size reduction has occurred. For example, Test case 2 introduces a reduction in size from 307 KB to 167 KB. This reduction is a result of the scoped information needs in gbXML files in contrast to the wide range of information in IFC files. Also, the degree of reduction increases when IFC files are becoming larger. Subsequently, when comparing the number of entities and relationships the created gbXML file has several more entities but way less relationships than the IFC file. Those extra entities in gbXML may be a result of using the *IfcRelSpaceBoundary* entity in the tool. For example, a wall across 3 building storeys is in IFC described as 1 object, while in gbXML this same wall is divided into 3 elements (each for every building storey). The downsize of relationships can again be explained by the scoped information needs in gbXML in contrast to the wide range of information (and thus relationships) in IFC files.

Simulation performance

After analyzing the concerned open data exchange files IFC and gbXML another important part of the validation process is the performance of whole-building energy analysis in DesignBuilder (with EnergyPlus as simulation engine). Previous section provided information on which elements are included in the gbXML file of Test case 2; the file contains both geometry and thermal properties. In the DesignBuilder software a new project is created, a geographical location (weather conditions) is selected and the gbXML file is imported. During this import it is required to select the option of "include thermal properties". By doing so the software includes gbXML "construction", "layer" and "material" elements as well. Then, whole-building energy analysis, shadow visualizations and computational fluid dynamics analysis can be done. This results in simulations of which for example the energy consumption, amount of CO2, velocity, predicted percentage of dissatisfied or operative temperature can be conducted. Such analysis contribute to an increased building performance and occupant comfort level. Finally, EnergyPlus simulation reports can be exported and design feedback can be provided.

5.4.2. Tool shortcomings

As mentioned in previous section, after testing the 5 test cases some shortcomings and barriers of the developed tool are introduced. Key to these issues is the fact that the tool currently is a "proof of concept" and still needs development to be fully functional. The found shortcomings are explained below. Especially Test case 3, Test case 4 and Test case 5 introduce the most crucial failures. Testing revealed the following main shortcomings:

- Failures with regard to included building geometry (some geometry is missing);
- Barriers while extracting thermal properties from the IFC file (thermal properties are not always included).

The first stated issue is a result of not properly enclosing spaces and the use of diverse building elements and materials. Key to this problem is the manner in which building models are created in Autodesk Revit. Rooms might not be assigned correctly or consistent, which results in lacking *IfcRelSpaceBoundary* entities (which are necessary for the tool to create spaces). Also, Test case 3 consists of angled (circle shaped) walls which are not supported by the tool; rectangular faces are needed. However, it was possible to perform simulations in DesignBuilder with this test case because the software apparently fixes the lacking (circle shaped) walls by creating multiple rectangular faces instead, see the figures in Appendix VII. In addition, the use of diverse building elements and materials also contributes to the found problem stated above. At this moment only the building elements *IfcWallStandardCase*, *IfcSlab* (floors only), *IfcCovering* and *IfcWindow* are supported by the conversion tool. The result of this limited coverage of information is that spaces which are enclosed by not supported building elements cannot be completely created in gbXML with the conversion tool. This results in missing spaces or a lack of surfaces, see Test case 5 and Test case 6 in Appendix VII. Clearly, whole-building energy analysis cannot be fully performed if this is the case. The second stated shortcoming arises due to the use of varying building elements and materials as well. During script development the entity relationships are based on the in chapter 5.2.1 elaborated relations; the script executes the in chapter 5.2.2 explained matching of IFC entities and gbXML elements. However, testing reveals that some IFC entities in Test case 4 and Test case 5 do not comply with those relationships, for example relating *IfcPropertySet* entities of some *IfcCovering* entities cannot be found with the iterating process conducted by the tool. This shortcoming results in lacking thermal properties, causing simulation software (DesignBuilder in this case) to show errors. Testing revealed that DesignBuilder minimal needs the material properties “R-value” and “thickness”. When using other BPS software the needed thermal properties can differ.

Finally, choices are made with regard to the used simulation software. At this moment the tool only creates gbXML files which are a valid input for DesignBuilder, tool development is needed to create files which are suitable for varying BPS software packages. Moreover, previous to the validation process several simulation software packages are tested (e.g. DesignBuilder, Green Building Studio (GBS) and OpenStudio). This process revealed that each package requires specific and often differing input information. The OpenStudio software for example requires the material thermal property “SpecificHeat”, while DesignBuilder needs the “R-value” and “Thickness” of a certain material. Subsequently, testing with GBS revealed that this web application needs additional information which cannot be directly extracted from IFC, but which is included when exporting a gbXML file via Autodesk Revit. An example of this information is the gbXML element “RectangularGeometry” with child elements “Azimuth”, “Tilt”, “Width” and “Height”. Also, GBS requires two “AdjacentSpaceId” elements for each floor element instead of one like other simulation software do. Insights to this “added” information by Autodesk Revit can be gained when exporting a gbXML file directly from Autodesk Revit. Lastly, the DesignBuilder software is used because of its worldwide application in the construction sector and the reliance on EnergyPlus for thermal simulations.

6. Conclusion and future work

Nowadays, Building Information Modeling (BIM) is subject to lots of development parallel with a broad implementation in the Architecture, Engineering and Construction (AEC) sector. Managing building information and particularly information between different project phases and software is becoming more and more of interest. Project members need to interact between different software tools, which calls for an ensured interoperability. However, in the problem definition it is stated that many research literature are indicating a lacking interoperability. This inconvenience results in extra project costs and an increased duration of early-design analysis due to the re-entering of building information. The process of re-entering information is both time consuming and sensitive for human error.

Therefore, the objective of this thesis is to create a mapping and automatic conversion tool between BIM and Building Performance Simulation (BPS). Hence, the main research problem of this study concerns the translation of the open exchange format IFC to a validated gbXML file format; no standalone mapping or conversion tool between these formats exists today. In order to do so, the following central research question is formulated:

"How can the use of Building Performance Simulation software be implemented during an early phase of Building Information Modelling in an approachable manner?"

Hereafter, a total of 7 sub-questions are formulated which each are appointed to the corresponding research phase, explained in chapter 2.4. Distinguished phases are: (1) Background and related work, (2) Analysis, (3) Tool development and (4) Validation. The stated research questions can be answered by discussing the results of each phase.

6.1. Conclusion

To start with, this graduation project contributes to the current movement of the AEC industry towards a more developed BIM environment. The concerns within the AEC industry are asking for a seamless integration between diverse BIM tools. Interoperability of different project phases and also varying software packages is becoming increasingly important. The development and application of new ICT tools that handle the integration between those aspects are contributing to a highly evolved BIM environment and ultimately a more automated industry. At this moment in time multiple ICT tool exist, often a result of scientific research and with deficiencies included. This thesis aims to create a tool which handles the integration between BIM and BPS, in order to do so the following main steps can be distinguished: (1) specification of required input information, (2) specification of needed BIM output information, and (3) the process of creating a clear mapping between both data sets. In general, these steps can be used when developing any new ICT tool that handles the integration between diverse BIM tools today.

Within this graduation project the first phase of the research concerns the study to background information and related work. Based on the stated research problem this phase is broken down into the design process (BIM), simulation process (BPS) and the use of open BIM formats (IFC and gbXML). Thereafter, the second phase aims on the analysis of current and desired business processes that occur between design and simulation parties. Moreover, the IFC file format and the input requirements for BPS are studied. Lastly, the third phase of the research covers the actual tool development, requirements are set and the process of scripting and matching of gbXML elements with IFC entities is started. Here, relationships between gbXML elements and corresponding IFC entities are created and linkages are established by iterating over multiple IFC entities. A core function of the tool is the translation of implicit geometry in IFC into explicit geometry in gbXML. The tool handles this process by making use of the *IfcRelSpaceBoundary* entity; IfcOpenShell and PythonOCC are used to generate explicit geometry from this entity. The *IfcRelSpaceBoundary* entity finds its origin when assigning rooms to a building project in Autodesk Revit 2016. Then, exporting the project to an IFC file with 2nd level space boundaries results in geometry positioned at the precise centerline of building elements. Another core function of the tool is the capability to include thermal properties and assign those properties to corresponding building elements. During whole-building energy analysis the properties can be used to supply the simulation software. Moreover, assigning properties to a gbXML element is done by iterating over *IfcPropertySet* entities and making use of “id” and “ref” attributes in the gbXML schema. Also, the tool is making use of the GUIDs from IFC to establish this function and reference connection.

Additionally, when performing scientific research a crucial part is the validation process, especially in the field of software development. The focus of this research has been set on the information which is needed to perform (early design) whole-building energy analysis. Obviously, more information can exist in the gbXML schema but choices are made in order to ensure depth and quality of the results. Then, the validation process indicates the precise capabilities and barriers of the tool by testing 5 pilot projects. This process revealed that gbXML files can be successfully created and comply with the official gbXML schema; most essential elements are included. Moreover, the following geometric IFC entities of building models can be converted to gbXML elements:

- *IfcWallStandardCase*;
- *IfcSlab* (only floors);
- *IfcWindow*;
- *IfcCovering*.

Subsequently, opening elements and thermal properties can be included as well. Thereafter, the resulting new gbXML models are used for whole-building energy simulation in DesignBuilder. More complex building models cause issues with regard to included building geometry or the extraction of thermal properties from the IFC file. The manner in which projects are created in design software is key to this issue. Rooms might not be assigned correctly or consistent, which results in lacking *IfcRelSpaceBoundary* entities. Lastly, the tool is not capable of progressing all types of building elements and shapes, which results in missing spaces or a lack of surfaces.

To conclude, the central research question can be answered by proposing the developed conversion tool which handles the integration between BIM and BPS. Project costs and the duration of early-design analysis can be brought to a minimum, while improving the quality of building projects. This tool is making use of the open BIM standards to create an automated translation and so implement BPS during an early phase of BIM. The research initiates the potential and possibilities of converting the IFC file format to a validated gbXML file format. Obviously the current version of the tool is a prototype which should at this moment be used in a scientific environment only. Development and further research is required to fully take advantage of the proposed tool. Found results indicate the possibilities and shortcomings of the tool and subsequently function as foundation for future research. As explained above there are in general 3 steps that can be used when developing any new ICT tool that handles the integration between diverse BIM tools. By doing so, blind spots in business processes can be eliminated in the future and human error can be brought to a minimum.

Finally, in the created action plan of the European Parliament and the Council of the European Union it is stated that all European member states need to ensure that by the year 2018 new buildings occupied and owned by public authorities have to be nearly zero-energy buildings. Furthermore, by the year 2020 all member states have to make sure that new buildings are realized as nearly zero-energy buildings. This research contributes to the achievement of these goals and at the same time create a proper trade-offs between life cycle costs and an improved sustainability of the built environment.

6.2. Future work

Prior to this graduation study several objectives are set together with their associated limitations. The conducted research resulted in a “proof of concept” and is meant for scientific use only. Clearly these kind of prototypes introduce limitations, large sets of operating instruction and lacking user interfaces. After studying the capabilities, barriers and opportunities of this tool it was possible to deliver clear and relevant results. Based on those results conclusions are drawn and subsequently the last step of the study is to point out areas of improvement, this can be both theoretical and practical. Those areas of improvement are important because of their functioning as foundation for future research. By the means of this future work it might be possible to eliminate barriers and establish a fully automated and well-functioning tool which is usable in practice. Therefore, it is important to clearly document the found insights and formulate relevant recommendations for future researchers. This chapter discusses those areas of improvement based on the research results.

Design software dependency

One of the main areas for improvement of the tool is the current dependency on how the building model is created in design software. Geometry for the gbXML file format is derived by making use of the 2nd level space boundaries in IFC, these find their origin when assigning rooms to the model in Autodesk Revit. Being less dependent of those 2nd level space boundaries will increase the functionality of the tool and influence the application of it positively. For this reason, future work can focus on creating geometry from specific IFC entities (building elements) itself, instead of using their corresponding space boundary entities. Then, from this recommendation a second area of improvement directly arises, namely processing a larger set of geometry related IFC entities. At this moment the tool only handles the entities I*fcWallStandardCase*, *IfcSlab* (only floor), *IfcWindow* and *IfcCovering*. When more IFC entities are handled by the tool it will be possible to convert more complex building models and have less lacking geometry, which ultimately results in a fully converted building model. An example of possible geometry related IFC entities are roof elements, this specific building element is in IFC described by the *IfcSlab* entity. To conclude, the above stated recommendations can be met with a more elaborated research to IFC entity relationships and a decent script development to match the corresponding gbXML elements.

Whole-building energy analysis

Another possibility for improvement of the tool is to create gbXML files that are a valid validated input for varying BPS software packages throughout the market. The current "proof of concept" is only validated by making use of the simulation software DesignBuilder. However, in practice multiple BPS software exists throughout the market of today and companies each use the program of their own preference. Moreover, this graduation project revealed that each of the BPS software packages require different gbXML elements and more specifically varying thermal properties of materials in order to perform whole-building energy analysis. Creating valid gbXML files which are suitable for varying BPS software will increase the applicability of the conversion tool and its future potential.

Additionally, to ensure quality and accuracy of the outcome of whole-building energy analysis it is useful to test the outcome of simulations that are performed with the created gbXML file. In order to do so, an analytical verification and comparative diagnostic procedure is being developed by the International Energy Agency (IEA), named the Building Energy Simulation Test (BESTEST) method (Neymark et al., 2002). This method is mainly being used by developers as part of their standard quality control program. The BESTEST tests performance simulation software by comparing simulation results to prefixed analytical solutions which are developed for several test cases. Concluding, improvements related to whole-building energy analysis can be met through scientific research and a more developed script.

Human intervention

Besides technical improvements of the existing tool future work can be valuable to minimize the currently needed human intervention which is required when applying the tool in practice. A useful addition can be to add the conversion tool as plugin to the BIMserver. Creating such a plugin would make it possible to automatically receive building performance feedback when an IFC revision is made on the server. The interoperability of the design and simulation process will be established in a more approachable manner and design feedback can be received real-time without any human intervention. Another possibility to ensure a minimal human intervention is to create a clear Graphical User Interface (GUI). Such user interfaces can be helpful and valuable when selecting preferences during the conversion of data formats. In addition, a GUI might make the tool accessible for all kinds of users, instead of scientific users and developers only.

Finally, as described above the current "proof of concept" has multiple areas of improvement. Future work might be able to eliminate barriers and establish a fully automated and well-functioning tool which is usable in practice. Drawn conclusions and formulated recommendations in this graduation thesis can function as input during these future studies. Ultimately an improved conversion tool can contribute to a more developed BIM environment.

7. References

Abanda, F. H., Vidalakis, C., Oti, A. H., & Tah, J. H. M. (2015). A critical analysis of Building Information Modelling systems used in construction projects. *Advances in Engineering Software*, *90*, 183–201. http://doi.org/10.1016/j.advengsoft.2015.08.009

Ahn, K. U., Kim, Y. J., Park, C. S., Kim, I., & Lee, K. (2014a). BIM interface for full vs. semi-automated building energy simulation. *Energy and Buildings*, *68*(PART B), 671–678. http://doi.org/10.1016/j.enbuild.2013.08.063

Ahn, K. U., Kim, Y. J., Park, C. S., Kim, I., & Lee, K. (2014b). BIM interface for full vs. semi-automated building energy simulation. *Energy and Buildings*, *68*(PART B), 671–678. http://doi.org/10.1016/j.enbuild.2013.08.063

Asl, M. R., Bergin, M., AdamMenter, & Yan, W. (2014). BIM-based Parametric Building Energy Performance MultiObjective Optimization. *32nd eCAADe Conference*, *224*, 10. Retrieved from http://autodeskresearch.com/pdf/bimparametric.pdf

Augenbroe, G. (2002). Trends in building simulation, *37*, 891–902.

Autodesk. (2016). Revit Sample Project Files. Retrieved June 18, 2016, from http://help.autodesk.com/view/RVT/2016/ENU/?guid=GUID-61EF2F22-3A1F-4317-B925-1E85F138BE88

Bazjanac, V., & Crawley, D. (1999). Industry Foundation Classes and interoperable commercial software in support of design of energy-efficient buildings, (April).

BIM Industry Working Group (BIWG). (2011). Strategy Paper for the Government Construction Client Group From the BIM Industry Working Group. *Department of Business, Innovation and Skills, URN 11*, (March), 1– 107.

BIR. (2008). BIR Kenniskaart nr. 1 Nederlandse BIM Levels, (1).

Biswas, T., Wang, T.-H., & Krishnamurti, R. (2006). Integrating sustainable building rating systems with building information models. *13th International Conference on Computer Aided Architectural Design Research in Asia*, 193–200. Retrieved from http://cumincad.scix.net/data/works/att/caadria2008_24_session3a_193.content.pdf

Borgo, S., & Sanfilippo, E. M. (2015). Ontological Analysis and Engineering Standards : An Initial Study of IFC, (May 2016), 2–29. http://doi.org/10.1007/978-3-319-15326-1

Boton, C., Kubicki, S., & Halin, G. (2015). The Challenge of Level of Development in 4D/BIM Simulation Across AEC Project Lifecyle. A Case Study. *Procedia Engineering*, *123*, 59–67. http://doi.org/10.1016/j.proeng.2015.10.058

buildingSMART. (2007a). IFC2x Edition 3 Technical Corrigendum 1. Retrieved May 24, 2016, from http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/

buildingSMART. (2007b). IFC2x Edition 3 Technical Corrigendum 1. Retrieved from http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/index.htm

BuildingSMART. (2014). Data Dictionary. Retrieved March 17, 2016, from http://www.buildingsmart.org/standards/standards-library-tools-services/data-dictionary/

buildingSMART. (2014). Technical Vision. Retrieved March 10, 2016, from http://www.buildingsmart.org/

buildingSMART. (2015). IFC4 - Addendum 1 [Final Standard]. Retrieved March 17, 2016, from http://www.buildingsmart-tech.org/ifc/IFC4/Add1/html/

Carrillo, P., & Chinowsky, P. (2006). Exploiting Knowledge Management : The Engineering and Construction Perspective. *Journal of Management in Engineering*, (January), 2–10. http://doi.org/10.1061/(ASCE)0742-597X(2006)22:1(2)

Cemesova, A., Hopfe, C. J., & McLeod, R. S. (2015). PassivBIM: Enhancing interoperability between BIM and low energy design software. *Automation in Construction*, *57*, 17–32. http://doi.org/10.1016/j.autcon.2015.04.014

Cheng, J. C. P., & Das, M. (2014). a Bim-Based Web Service Framework for Green Building Energy Simulation and Code Checking. *Journal of Information Technology in Construction*, *19*(MAY 2014), 150–168.

Choi, J., Kim, H., & Kim, I. (2015). Open BIM-based quantity take-off system for schematic estimation of building frame in early design stage. *Journal of Computational Design and Engineering*, *2*(1), 16–25. http://doi.org/10.1016/j.jcde.2014.11.002

Clarke, J. A., & Hensen, J. L. M. (2015). Integrated Building Performance Simulation:Progress, Prospects and Requirements. *Building and Environment*, *91*, 294–306. http://doi.org/10.1016/j.buildenv.2015.04.002

Costa, G., & Madrazo, L. (2015). Connecting building component catalogues with BIM models using semantic technologies: An application for precast concrete components. *Automation in Construction*, *57*, 239–248. http://doi.org/10.1016/j.autcon.2015.05.007

Cox, A., & Thompson, I. (1997). “Fit for purpose” contractual relations: determining a theoretical framework for construction projects. *European Journal of Purchasing & Supply Management*, *3*(3), 127–135. http://doi.org/10.1016/S0969-7012(97)00005-1

Crawley, D. B., Hand, J. W., Kummert, M., & Griffith, B. T. (2005). CONTRASTING THE CAPABILITIES OF BUILDING ENERGY PERFORMANCE SIMULATION PROGRAMS A Joint Report by, (July).

DesignBuilder. (2016). DesignBuilder. Retrieved September 28, 2016, from http://www.designbuilder.co.uk/

Dhillon, R. K., Jethwa, M., & Rai, H. S. (2014). Extracting Building Data from BIM with IFC. *Int. J. on Recent Trends in Engineering and Technology*, *11*(1).

Diaz-Vilarino, L., Laguela, S., Armesto, J., & Arias, P. (2013). Semantic as-built 3d models including shades for the evaluation of solar influence on buildings. *Solar Energy*, *92*, 269–279. http://doi.org/10.1016/j.solener.2013.03.017

DOE. (2016). Building Energy Software Tools. Retrieved March 14, 2016, from http://www.buildingenergysoftwaretools.com/software-listing

Dong, B., Lam, K. P., Huang, Y. C., & Dobbs, G. M. (2007). A comparative study of the IFC and gbXML informational infrastructures for data exchange in computational design support environments Geometry information. *Proceedings: Building Simulation 2007*, 1530–1537.

Donkers, S. (2013a). Automatic generation of CityGML LoD3 building models from IFC models. *Zhurnal Eksperimental'noi I Teoreticheskoi Fiziki*, 127. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:No+Title#0

Donkers, S. (2013b). Automatic generation of CityGML LoD3 building models from IFC models. *Zhurnal Eksperimental'noi I Teoreticheskoi Fiziki*, 127.

Dossick, C., Anderson, A., Iorio, J., Neff, G., & Taylor, J. (2012). Messy Talk and Mutual Discovery: Exploring the Necessary Conditions for Synthesis in Virtual Teams. Retrieved fromhttp://www.epossociety.org/EPOC2012/Papers/Dossick_Anderson_Iorio_Neff_Taylor.pdf

Eastman, C., Teicholz, P., Sacks, R., & Liston, K. (2011a). *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. *Building* (Vol. 2). http://doi.org/10.1002/9780470261309

Eastman, C., Teicholz, P., Sacks, R., & Liston, K. (2011b). *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. *Building* (Vol. 2). http://doi.org/10.1002/9780470261309

EBPD. (2010). Directive 2010/31/EU of the European Parliament and the council of 19 May 2010 on the energy performance of buildings. *Official Journal of the European Union*, 13–35.

El-Mekawy, M., Östman, a, & Hijazi, I. (2012). An evaluation of ifc-citygml unidirectional conversion. *International Journal*, *3*(5), 159–171. http://doi.org/10.14569/IJACSA.2012.030525

El-Mekawy, M., & Östman, A. (2010). Semantic Mapping: an Ontology Engineering Method for Integrating Building Models in IFC and CITYGML. *Proceedings of the 3rd ISDE Digital Earth Summit*, 1–11. Retrieved from http://news.digitalearth-isde.org/Bulgaria/pdf/32_El-Mekawy_Sweden_paper.pdf

El-Mekawy, M., & Östman, A. (2010). Semantic Mapping: an Ontology Engineering Method for Integrating Building Models in IFC and CITYGML. *Proceedings of the 3rd ISDE Digital Earth Summit*, 1–11.

Gallaher, M. P., O'Conor, A. C., Dettbarn, J. L., & Gilday, L. T. (2004). Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry. *National Institute of Standards & Technology*, 1–210. http://doi.org/10.6028/NIST.GCR.04-867

gbXML. (2015). Schema GreenBuildingXML_Ver6.01. Retrieved from http://gbxml.org/schema/6-01/GreenBuildingXML_Ver6.01.xsd

gbXML. (2016a). gbXML Validator 2.0. Retrieved June 14, 2016, from http://www.controlsestimate.com/Validator/Pages/TestPage.aspx

gbXML. (2016b). Open Green Building XML Schema. Retrieved March 17, 2016, from http://gbxml.org/

Groeneveld, J. (2015). Building performance simulation from a BIM; EPG input for Vabi abstracted from Revit by IFC, (August).

Guzmann Garcia, E., & Zhu, Z. (2015). Interoperability from building design to building energy modeling. *Journal of Building Engineering*, *1*, 33–41. http://doi.org/10.1016/j.jobe.2015.03.001

Häfele, K.-H. (2010). IFC Implementation Agreement Space Boundary, (March).

Ham, Y., & Golparvar-Fard, M. (2014). Mapping actual thermal properties to building elements in gbXML-based BIM for reliable building energy performance modeling. *Automation in Construction*, *49*, 214–224. http://doi.org/10.1016/j.autcon.2014.07.009

Hitchcock, R. J., & Wong, J. (2011). Transforming IFC Architectural View BIMS for Energy Simulation: 2011. *Proceedings of Building Simulation 2011: 12th Conference of International Building Performance Simulation Association*, 1089–1095. Retrieved from http://www.ibpsa.org/proceedings/BS2011/P_1391.pdf

IPCC. (2008). *Climate Change 2007 Synthesis Report*. *Intergovernmental Panel on Climate Change [Core Writing Team IPCC*. http://doi.org/10.1256/004316502320517344

Jayasena, H. S., & Weddikkara, C. (2013). Assessing the BIM Maturity in a BIM Infant Industry. *The Second World Construction Symposium 2013: Socio-Economic Sustainability in Construction*, (June 2013), 62–69. Retrieved from http://www.suranga.net/publications/2013_bim_maturity.pdf

Jeong, W., Kim, J. B., Clayton, M. J., Haberl, J. S., & Yan, W. (2014). Translating building information modeling to building energy modeling using model view definition. *TheScientificWorldJournal*, *2014*(1), 638276. http://doi.org/10.1155/2014/638276

Karlshoj, J. (2011). Information Delivery Manuals. Retrieved March 17, 2016, from http://iug.buildingsmart.org/idms

Kim, I., Kim, J., & Seo, J. (2012). Development of an IFC-based IDF Converter for Supporting Energy Performance Assessment in the Early Design Phase. *Journal of Asian Architecture and Building Engineering*, *11*(2), 313–320. http://doi.org/10.3130/jaabe.11.313

Kim, J. B., Jeong, W., Clayton, M. J., Haberl, J. S., & Yan, W. (2015). Developing a physical BIM library for building thermal energy simulation. *Automation in Construction*, *50*(C), 16–28. http://doi.org/10.1016/j.autcon.2014.10.011

Koufteros, X., Vonderembse, M., & Doll, W. (2001). Concurrent engineering and its consequences. *Journal of Operations Management*, *19*(1), 97–115. http://doi.org/10.1016/S0272-6963(00)00048-6

Kovacic, I., & Müller, C. (2014). Challenges for the Implementation of Integrated Design in the Planning Practice. *Procedia - Social and Behavioral Sciences*, *119*, 529–538. http://doi.org/10.1016/j.sbspro.2014.03.059

Kovacic, I., & Zoller, V. (2014). Building life cycle optimization tools for early design phases. *Energy*, *92*, 409–419. http://doi.org/10.1016/j.energy.2015.03.027

Liu, S., Meng, X., & Tam, C. (2015). Building information modeling based building design optimization for sustainability. *Energy and Buildings*, *105*, 139–153. http://doi.org/10.1016/j.enbuild.2015.06.037

Maile, T., Fischer, M., & Bazjanac, V. (2007). Building Energy Performance Simulation Tools - a Life-Cycle and Interoperable Perspective. *Center for Integrated Facility Engineering*, (December), 1–49. Retrieved from cife.stanford.edu/sites/default/files/WP107.pdf

Merschbrock, C., & Munkvold, B. E. (2015). Effective digital collaboration in the construction industry - A case study of BIM deployment in a hospital construction project. *Computers in Industry*, *73*, 1–7. http://doi.org/10.1016/j.compind.2015.07.003

Migilinskas, D., Popov, V., Juocevicius, V., & Ustinovichius, L. (2013). The benefits, obstacles and problems of practical bim implementation. *Procedia Engineering*, *57*, 767–774. http://doi.org/10.1016/j.proeng.2013.04.097

Moon, H. J., Choi, M. S., Kim, S. K., & Ryu, S. H. (2011). Case Studies for the Evaluation of Interoperability Between a BIM Based Architectural Model and Building Performance Analysis Programs. *Building Simulation 2011: 12th Conference of International Building Performance Simulation Association*, 1521–1526. Retrieved from http://www.ibpsa.org/proceedings/BS2011/P_1510.pdf

Negendahl, K. (2015). Building performance simulation in the early design stage: An introduction to integrated dynamic models. *Automation in Construction*, *54*, 39–53. http://doi.org/10.1016/j.autcon.2015.03.002

Neymark, J., Judkoff, R., Knabe, G., Le, H. T., Dürig, M., Glass, A., & Zweifel, G. (2002). Applying the building energy simulation test (BESTEST) diagnostic method to verification of space conditioning equipment models used in whole-building energy simulation programs. *Energy and Buildings*, *34*(9), 917–931. http://doi.org/10.1016/S0378-7788(02)00072-5

Paul, N., & Borrmann, A. (2009). Geometrical and topological approaches in building. *Electronic Journal of Information Technology in Construction*, *14*(October), 705–723.

Pennell, J. P., & Winner, R. I. (1989). I . Winner Institute for Defense. *Concurrent Engineering*, 647–655.

Prada-Hernandez, A. V., Rojas-quintero, J. S., Vallejo-Borda, J. A., & Ponz-Tienda, J. L. (2015). Interoperability of Building Energy Modeling (Bem) With Building Information Modeling (Bim), (October), 519–526. http://doi.org/10.13140/RG.2.1.3807.3042

Rahmani, M., Zarrinmehr, S., Bergin, M., & Yan, W. (2015). BPOpt : A framework for BIM-based performance optimization. *Energy & Buildings*, *108*, 401–412. http://doi.org/10.1016/j.enbuild.2015.09.011

Remmen, P., Cao, J., Ebertsh@auser, S., Frisch, J., Lauster, M., Maile, T., … van Treeck, C. (2015). An open framework for integrated BIM-based building performance simulations using Modelica. *Proceedings of the 14th IBPSA Conference*, (DECEMBER). Retrieved from http://www.iea-annex60.org/downloads/p2384.pdf

Rijksoverheid. (2011). Rijksgebouwendienst gaat BIM voorschrijven bij geïntegreerde contracten. Retrieved March 10, 2016, from https://www.rijksoverheid.nl/actueel/nieuws/2011/09/27/rijksgebouwendienst-gaat-bim-voorschrijven-bij-geintegreerde-contracten

Ryan, E. M., & Sanquist, T. F. (2012). Validation of building energy modeling tools under idealized and realistic conditions. *Energy & Buildings*, *47*, 375–382. http://doi.org/10.1016/j.enbuild.2011.12.020

Simues, G. (2009). *A Guide to the Business Analysis Body of Knowledge ® (BABOK ® Guide)*.

Sohlenius, G. (1992). Concurrent Engineering, *41*, 645–655.

Succar, B. (2009). Building information modelling framework: A research and delivery foundation for industry stakeholders. *Automation in Construction*, *18*(3), 357–375. http://doi.org/10.1016/j.autcon.2008.10.003

TutorVista. (2016). Euclidean Space. Retrieved May 24, 2016, from http://math.tutorvista.com/algebra/Euclidean-space.html

UNEP. (2009). Buildings and climate change: a summary for decision-makers. *United Nations Environmental Programme, Sustainable Buildings and Climate Initiative*, 1–62. http://doi.org/ISBN: 987-92-807-3064-7 DTI/1240/PA

Weise, M., Liebich, T., See, R., Bazjanac, V., & Laine, T. (2011). Implementation guide: Space boundaries for energy analysis. *… (GSA) and Open …*, (August 2009), 1–62. Retrieved from http://www.blis-project.org/IAI-MVD/documents/Space_Boundaries_for_Energy_Analysis_v1.pdf

Wong, J. K. W., & Zhou, J. (2015). Enhancing environmental sustainability over building life cycles through green BIM: A review. *Automation in Construction*, *57*, 156–165. http://doi.org/10.1016/j.autcon.2015.06.003

Zhang, C., Beetz, J., & Weise, M. (2015). Interoperable validation for IFC building models using open standards. *Journal of Information Technology in Construction*, *20*(November 2014), 24–39.

8. Appendixes

Appendix I – Overview of BIM related simulation software

Appendix II – Interview questionnaire

Appendix III – Specification of included gbXML elements

Appendix IV – Export settings Autodesk Revit 2016

Appendix V – Tool flowcharts

Appendix VI – Python script of the tool

Appendix VII – Validation results

Appendix I – Overview of BIM related simulation software

Software package	Developer	BIM based import	About	Source
DOE-2	Lawrence Berkeley National Laboratory	Only indirectly	Advanced thermal simulation engine without user interface. Modeling geometry is not possible.	http://doe2.com/doe2/
EnergyPlus	U.S. Department of Energy	Only indirectly	Advanced thermal simulation engine without user interface. Modeling geometry is not possible.	https://energyplus.net/
eQuest	Part of the Energy Design Resources program	Only indirectly (via gbXML and GBS)	**GUI for DOE-2.** Needs GBS to export an DOE-2 input file.	http://doe2.com/equest/
RIUSKA	Granlund	IFC	**GUI for DOE-2.** Suitable for large projects but limited to four types of HVAC systems. IFC input information needs to be adjusted after input	http://www.granlund.fi/en/software/riuska/
DesignBuilder (used in thesis)	Design Builder Software Ltd	gbXML	**GUI for EnergyPlus.** Suitable for any design phase from conception to production. Modeling buildings rapidly without requiring special expertise	http://www.designbuilder.co.uk/
AECOsim Energy Simulator	Bentley	gbXML	**GUI for EnergyPlus.** Providing occupant productivity, comfort, and safety; controls lifecycle operational energy costs.	https://www.bentley.com/
OpenStudio	Open software, by NREL and U.S. Department of Energy	IFC; gbXML	**GUI for EnergyPlus.** Supports whole-building energy modeling using EnergyPlus and advanced daylight analysis using Radiance.	https://www.openstudio.net/
Autodesk Green Building Studio (GBS)	Autodesk	gbXML	Web-based energy analysis that determines building's total energy use and carbon footprint.	https://gbs.autodesk.com/GBS/
IES / Virtual Environment	Integrated Environmental Solutions	IFC; gbXML	Powerful software for building performance simulation which is capable of simulating advanced HVAC systems and passive building effects.	https://www.iesve.com/
IDA Indoor Climate and Energy (ICE)	EQUA Simulation AB	IFC	Studies the thermal indoor climate of thermal zones within a building and the energy consumption for the entire building.	http://www.equa.se/en/ida-ice
Vabi	Vabi Elements	IFC; gbXML	Simulates the performance of a building in all stages of the design, different scenarios can be applied and consequences on the overall performance can be detected.	http://www.vabi.nl/
TRACE	TRANE	gbXML	Design and analysis tool that helps HVAC professionals optimize the design of a building based on energy utilization and life-cycle cost.	http://www.trane.com/Index.aspx
TAS	Environmental Design Solutions Limited (EDSL)	gbXML	Accurately predict energy consumption, CO2 emissions, operating costs and occupant comfort.	http://www.edsl.net/main/

Appendix II – Interview questionnaire

Date: …	Interviewer: Maarten Visschers
Time: …	Respondent: …
Location: …	Job description: …

Interview opening

1) Description of the graduation project:
   a. Background information of the subject;
   b. The formulated problem definition;
   c. The research purpose and objectives;
   d. The proposed research plan and model;
   e. Expected results of the graduation project.
2) Respondents information:
   a. The company and department;
   b. The specific job description;
   c. Main activities and responsibilities.

Research related discussion

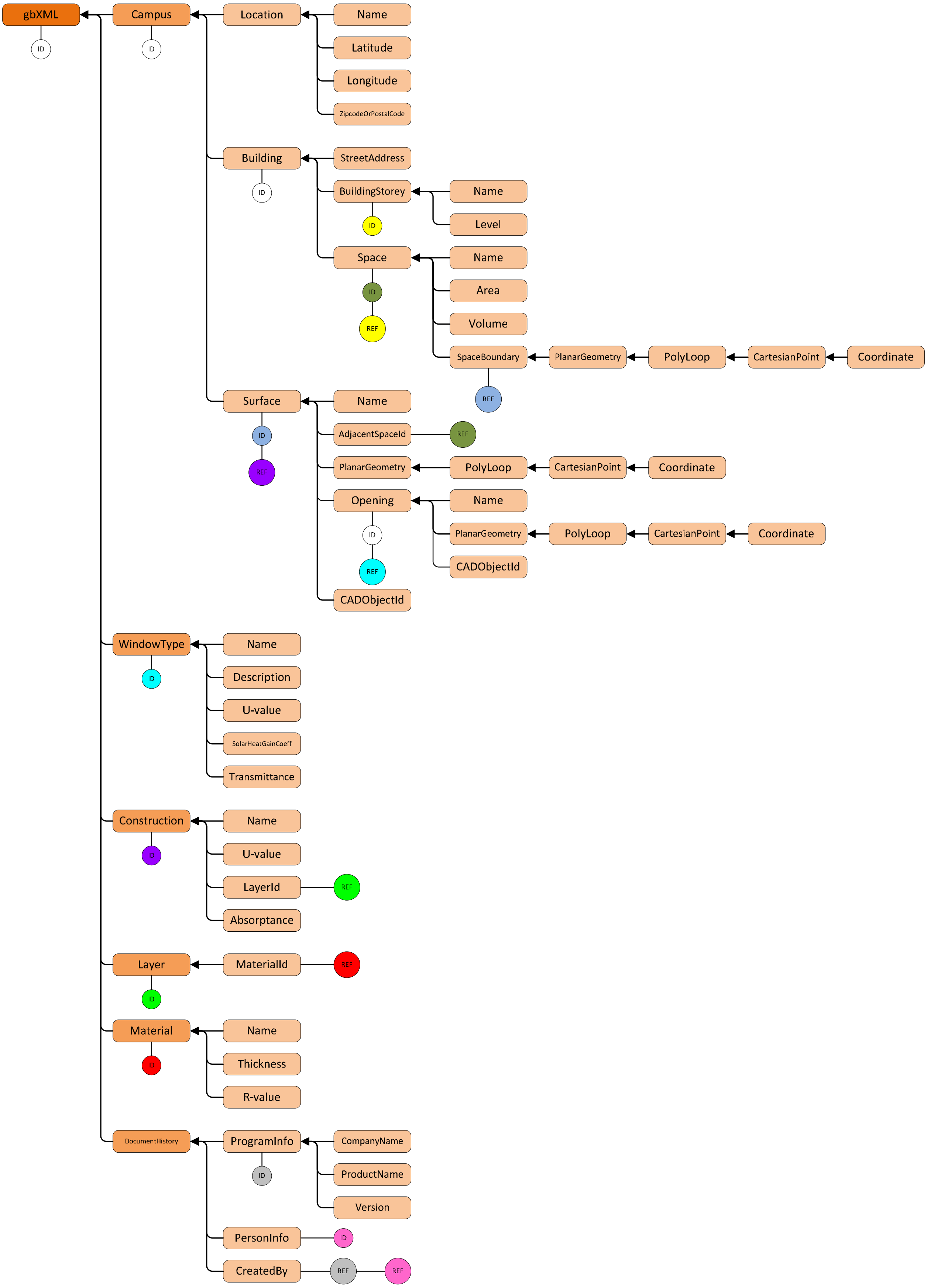
3) Discussion of the <u>current</u> building process about:
   a. Project involved parties and their responsibilities;
   b. The business process between design and simulation activities;
   c. The supply and exchange of data files within a project;
   d. Known bottlenecks and limitations of interoperability;
   e. (if applicable) The current way of performing whole-building energy analysis, including used software, data formats and the way of application in practice;
   f. (if applicable) Known bottlenecks, limitations and areas of improvement during Building Performance Simulation.
4) Discussion of the <u>desired</u> building process about:
   a. Respondents knowledge and support in the field of BIM (or why not);
   b. Respondents knowledge and support of open BIM standards such as IFC and gbXML (or why not);
   c. Respondents thoughts on or use of a central BIM server;
   d. Respondents thoughts on the proposed new conversion tool;
   e. Possible bottlenecks and limitations in the changed business process;
   f. Possible bottlenecks and limitations during the conversion of file formats.

Interview closure

5) Ending the interview:
   a. Respondents tips and ideas for improvement of the graduation project;
   b. Knowledge of possible other respondents which are of interest for this research.

<u>The following experts from the field are interviewed:</u>
(1) BIM Manager, (2) BIM IT Specialist, (3) BPS Consultant, (4) BIM Specialist, (5) Information Management Specialist, (6) BIM Consultant and (7) President of the gbXML Board of Directors Stephen Roth (by e-mail).

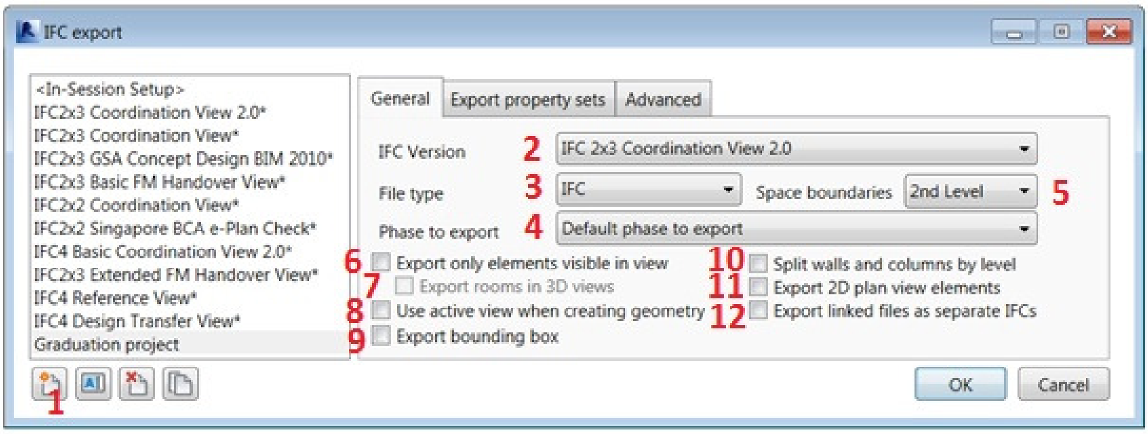
Appendix III – Specification of included gbXML elements



Overview of included gbXML elements (parent and child with “id” and “ref” attributes)

Appendix IV – Export settings Autodesk Revit 2016

Exporting building models to an IFC2x Edition 3 version file is performed with Autodesk Revit 2016 and thereby the Revit IFC Exporter - Alternate UI 16.4.0.0. The used settings during this export are shown below; the process is started by (1) creating a new template.



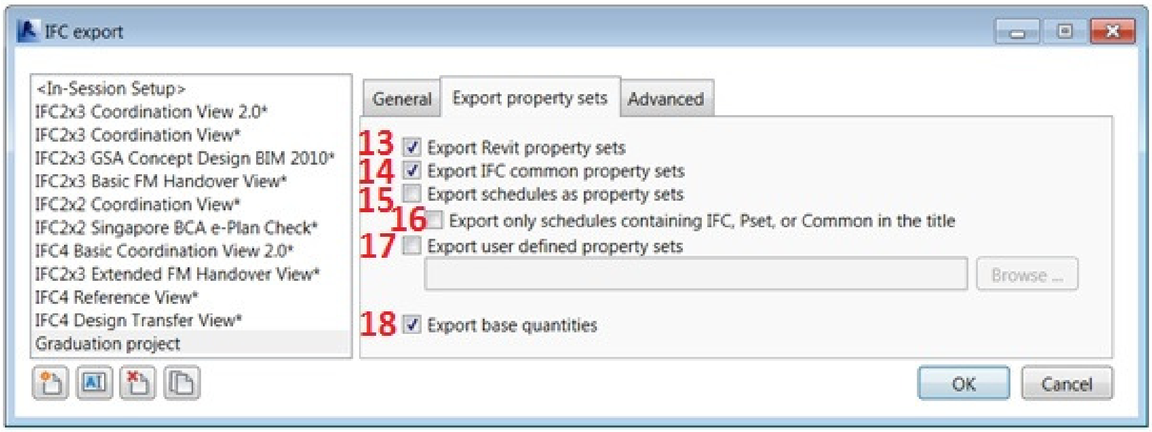
IFC export

<In-Session Setup>
IFC2x3 Coordination View 2.0*
IFC2x3 Coordination View*
IFC2x3 GSA Concept Design BIM 2010*
IFC2x3 Basic FM Handover View*
IFC2x2 Coordination View*
IFC2x2 Singapore BCA e-Plan Check*
IFC4 Basic Coordination View 2.0*
IFC2x3 Extended FM Handover View*
IFC4 Reference View*
IFC4 Design Transfer View*
Graduation project

General | Export property sets | Advanced

IFC Version 2 IFC 2x3 Coordination View 2.0
File type 3 IFC — Space boundaries 2nd Level 5
Phase to export 4 Default phase to export

6 □ Export only elements visible in view
7 □ Export rooms in 3D views
8 □ Use active view when creating geometry
9 □ Export bounding box
10 □ Split walls and columns by level
11 □ Export 2D plan view elements
12 □ Export linked files as separate IFCs

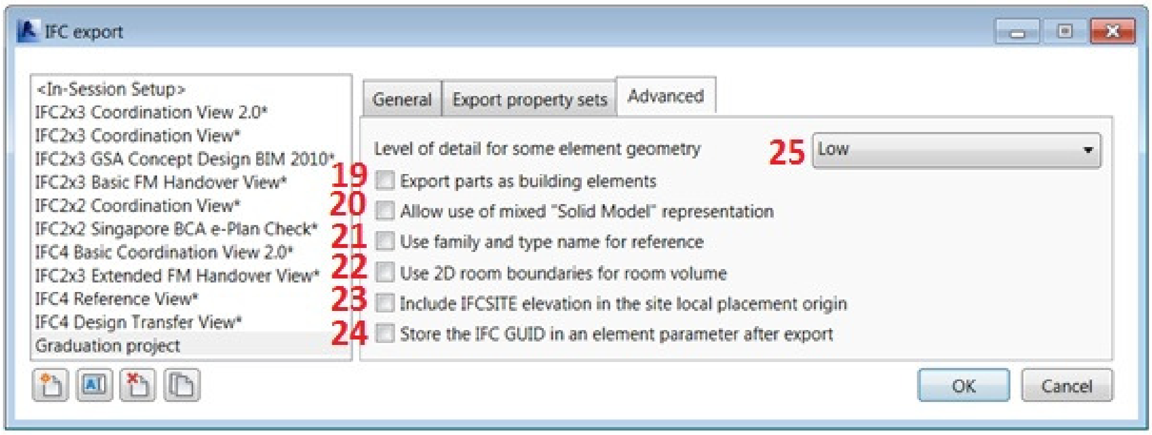


IFC export

<In-Session Setup>
IFC2x3 Coordination View 2.0*
IFC2x3 Coordination View*
IFC2x3 GSA Concept Design BIM 2010*
IFC2x3 Basic FM Handover View*
IFC2x2 Coordination View*
IFC2x2 Singapore BCA e-Plan Check*
IFC4 Basic Coordination View 2.0*
IFC2x3 Extended FM Handover View*
IFC4 Reference View*
IFC4 Design Transfer View*
Graduation project

General | Export property sets | Advanced

13 ☑ Export Revit property sets
14 ☑ Export IFC common property sets
15 □ Export schedules as property sets
16 □ Export only schedules containing IFC, Pset, or Common in the title
17 □ Export user defined property sets
18 ☑ Export base quantities



IFC export

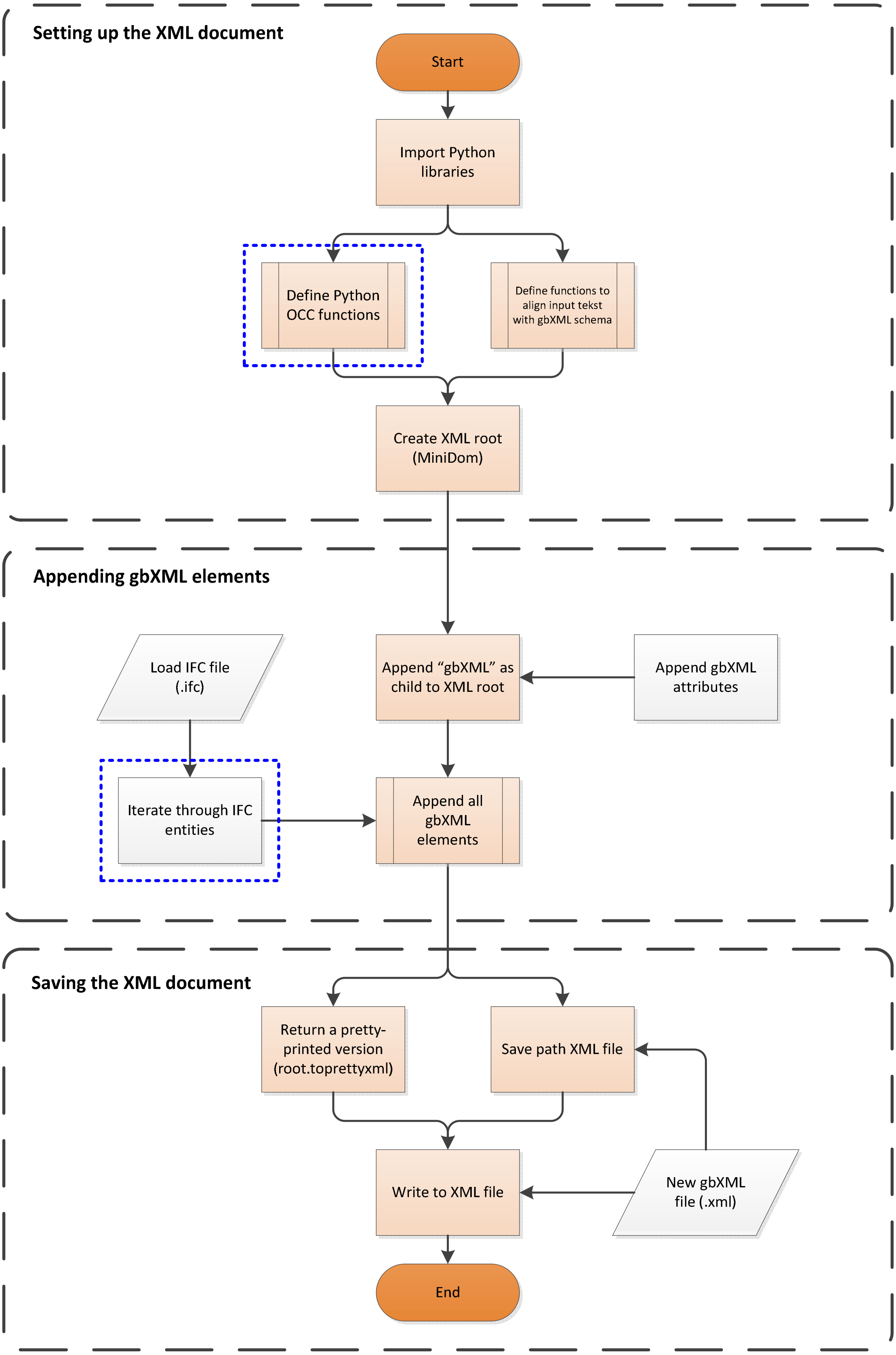
<In-Session Setup>
IFC2x3 Coordination View 2.0*
IFC2x3 Coordination View*
IFC2x3 GSA Concept Design BIM 2010*
IFC2x3 Basic FM Handover View*
IFC2x2 Coordination View*
IFC2x2 Singapore BCA e-Plan Check*
IFC4 Basic Coordination View 2.0*
IFC2x3 Extended FM Handover View*
IFC4 Reference View*
IFC4 Design Transfer View*
Graduation project

General | Export property sets | Advanced

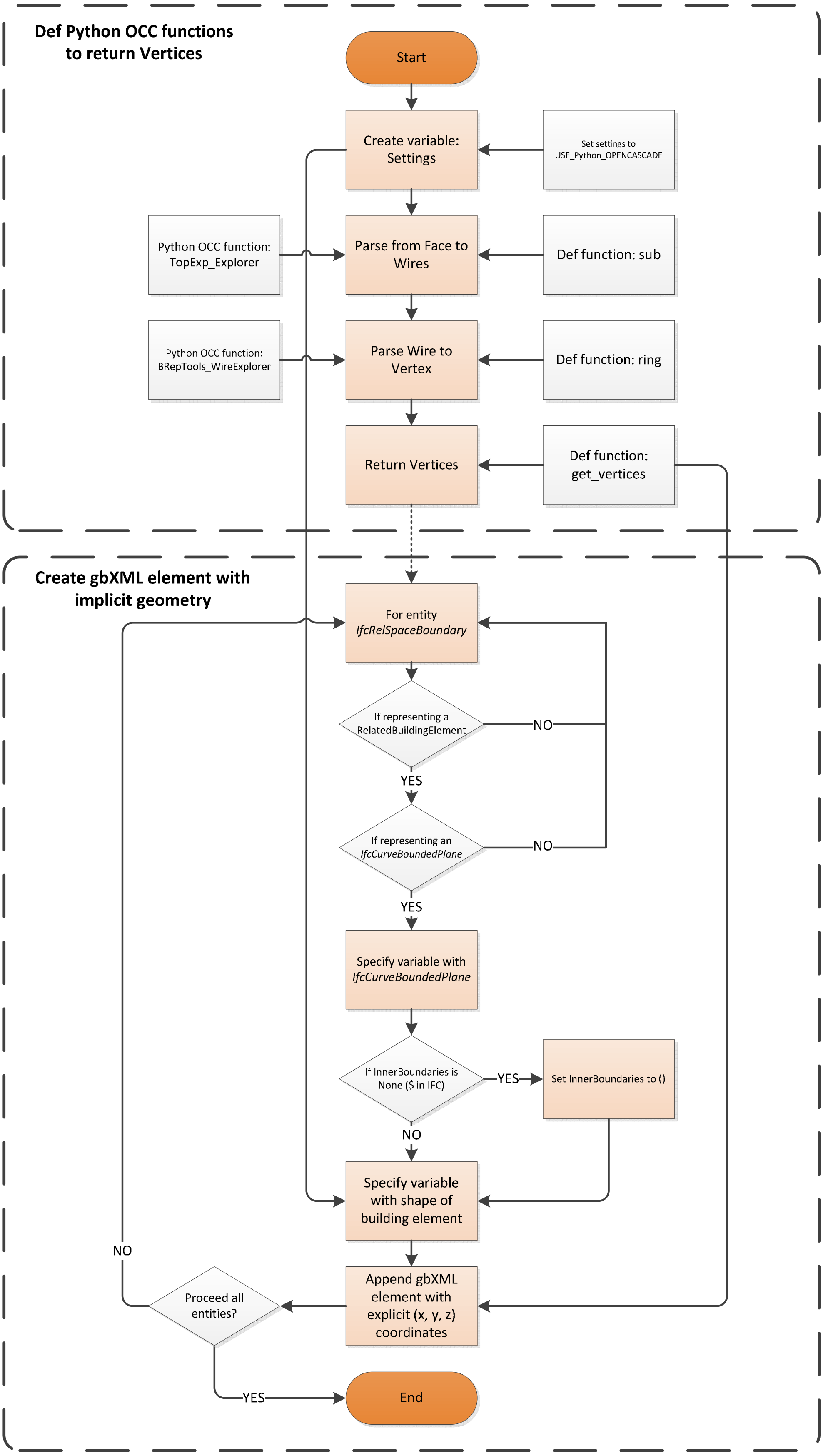
Level of detail for some element geometry 25 Low
19 □ Export parts as building elements
20 □ Allow use of mixed "Solid Model" representation
21 □ Use family and type name for reference
22 □ Use 2D room boundaries for room volume
23 □ Include IFCSITE elevation in the site local placement origin
24 □ Store the IFC GUID in an element parameter after export

2) Version IFC 2x3 Coordination View 2.0;
3) File type: IFC
4) Default phase to export;
5) 2nd level of space boundaries;
6) -
7) -
8) -
9) -
10) -
11) -
12) -
13) Revit property sets;
14) Standard IFC property sets;
15) -
16) -
17) -
18) IFC BaseQuantities;
19) -
20) -
21) -
22) -
23) -
24) -
25) -

Appendix V – Tool flowcharts


Setting up the XML document
Start
Import Python libraries
Define Python OCC functions
Define functions to align input tekst with gbXML schema
Create XML root (MiniDom)
Appending gbXML elements
Load IFC file (.ifc)
Append "gbXML" as child to XML root
Append gbXML attributes
Iterate through IFC entities
Append all gbXML elements
Saving the XML document
Return a pretty-printed version (root.toprettyxml)
Save path XML file
Write to XML file
New gbXML file (.xml)
End

First flowchart : General process of the tool (*blue parts are specified in the second chart*)


Def Python OCC functions to return Vertices
Start
Create variable: Settings
Set settings to USE_Python_OPENCASCADE
Python OCC function: TopExp_Explorer
Parse from Face to Wires
Def function: sub
Python OCC function: BRepTools_WireExplorer
Parse Wire to Vertex
Def function: ring
Return Vertices
Def function: get_vertices
Create gbXML element with implicit geometry
For entity *IfcRelSpaceBoundary*
If representing a RelatedBuildingElement
NO
YES
If representing an *IfcCurveBoundedPlane*
NO
YES
Specify variable with *IfcCurveBoundedPlane*
If InnerBoundaries is None ($ in IFC)
YES
Set InnerBoundaries to ()
NO
Specify variable with shape of building element
Append gbXML element with explicit (x, y, z) coordinates
Proceed all entities?
NO
YES
End

Second flowchart : Converting implicit to explicit geometry with Python OCC

Appendix VI – Python script of the tool

```
# Import necessary python libraries e.g. IfcOpenShell, PythonOCC and MiniDom
import ifcopenshell.geom
import OCC.BRep
import OCC.TopExp
import OCC.TopoDS
import OCC.TopAbs
import OCC.ProjLib
import OCC.BRepTools
import datetime
import time
from xml.dom import minidom

# Use IfcOpenShell and OPENCASCADE to convert implicit geometry into explicit geometry
# Each Face consists of Wires, which consists of Edges, which has Vertices
FACE, WIRE, EDGE, VERTEX = OCC.TopAbs.TopAbs_FACE, OCC.TopAbs.TopAbs_WIRE, OCC.TopAbs.TopAbs_EDGE, \
                           OCC.TopAbs.TopAbs_VERTEX

settings = ifcopenshell.geom.settings()
settings.set(settings.USE_PYTHON_OPENCASCADE, True)

def sub(shape, ty):
    F = {
        OCC.TopAbs.TopAbs_FACE: OCC.TopoDS.topods_Face,
        OCC.TopAbs.TopAbs_WIRE: OCC.TopoDS.topods_Wire,
        OCC.TopAbs.TopAbs_EDGE: OCC.TopoDS.topods_Edge,
        OCC.TopAbs.TopAbs_VERTEX: OCC.TopoDS.topods_Vertex,
    }[ty]
    exp = OCC.TopExp.TopExp_Explorer(shape, ty)
    while exp.More():
        face = F(exp.Current())
        yield face
        exp.Next()

def ring(wire, face):
    def vertices():
        exp = OCC.BRepTools.BRepTools_WireExplorer(wire, face)
        while exp.More():
            yield exp.CurrentVertex()
            exp.Next()

    return list(map(lambda p: (p.X(), p.Y(), p.Z()), map(OCC.BRep.BRep_Tool.Pnt, vertices())))

# Face to vertices
def get_vertices(shape):
    for face in sub(shape, FACE):
        for idx, wire in enumerate(sub(face, WIRE)):
            vertices = ring(wire, face)

            if idx > 0:
                vertices.reverse()
            return vertices

# Align the gbXML input according to the predefined official gbXML schema
def fix_xml_cmps(a):
    return 'campus' + a.replace('$', '').replace(':', '').replace(' ', '').replace('(',
'').replace(')', '')

def fix_xml_bldng(a):
    return 'building' + a.replace('$', '').replace(':', '').replace(' ', '').replace('(',
'').replace(')', '')

def fix_xml_stry(a):
    return 'storey' + a.replace('$', '').replace(':', '').replace(' ', '').replace('(',
'').replace(')', '')

def fix_xml_spc(a):
    return 'space' + a.replace('$', '').replace(':', '').replace(' ', '').replace('(',
'').replace(')', '')
```

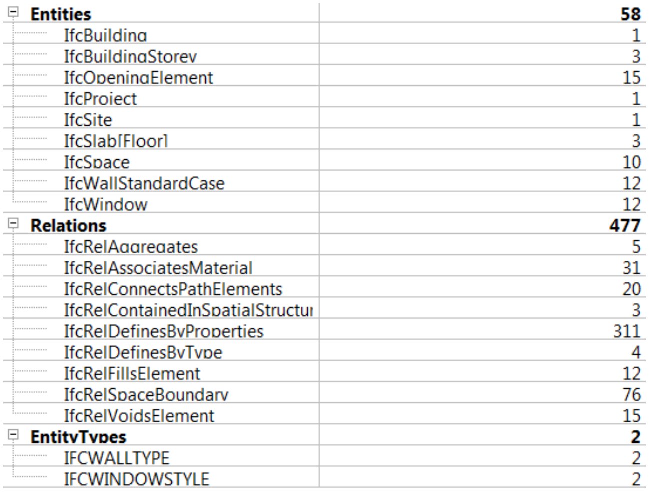
```
def fix_xml_id(a):
    return 'id' + a.replace('$', '').replace(':', '').replace(' ', '').replace('(',
'').replace(')', '')

def fix_xml_name(a):
    return 'object' + a.replace('$', '').replace(':', '').replace(' ', '').replace('(',
'').replace(')', '')

def fix_xml_cons(a):
    return 'construct' + a.replace('$', '').replace(':', '').replace(' ', '').replace('(',
'').replace(')', '')

def fix_xml_layer(a):
    return 'lyr' + a.replace('$', '').replace(':', '').replace(' ', '').replace('(',
'').replace(')', '')

# Access the specific IFC file; external directory:
ifc_file = ifcopenshell.open('Pilot project 1.ifc')

# Create the XML root by making use of MiniDom
root = minidom.Document()

# Create the 'gbXML' element and append it to the Root of the document
gbxml = root.createElement('gbXML')
root.appendChild(gbxml)

# Create attributes for the 'gbXML' element
gbxml.setAttribute('xmlns', 'http://www.gbxml.org/schema')
gbxml.setAttribute('temperatureUnit', 'C')
gbxml.setAttribute('lengthUnit', 'Meters')
gbxml.setAttribute('areaUnit', 'SquareMeters')
gbxml.setAttribute('volumeUnit', 'CubicMeters')
gbxml.setAttribute('useSIUnitsForResults', 'true')
gbxml.setAttribute('version', '0.37')

# Create a dictionary to store all gbXML element Id's
dict_id = {}

# Specify the 'Campus' element of the gbXML schema; making use of IFC entity 'IfcSite'
# This element is added as child to the earlier created 'gbXML' element
site = ifc_file.by_type('IfcSite')
for element in site:
    campus = root.createElement('Campus')
    campus.setAttribute('id', fix_xml_cmps(element.GlobalId))
    gbxml.appendChild(campus)

    dict_id[fix_xml_cmps(element.GlobalId)] = campus

    # Specify the 'Location' element of the gbXML schema; making use of IFC entities 'IfcSite' and
    # 'IfcPostalAddress'
    # This new element is added as child to the earlier created 'Campus' element
    location = root.createElement('Location')
    campus.appendChild(location)

    longitude = root.createElement('Longitude')
    longitudeValue = str(element.RefLongitude[0])
    longitude.appendChild(root.createTextNode(longitudeValue))
    location.appendChild(longitude)

    latitude = root.createElement('Latitude')
    latitudeValue = str(element.RefLatitude[0])
    latitude.appendChild(root.createTextNode(latitudeValue))
    location.appendChild(latitude)

    elevation = root.createElement('Elevation')
    elevation.appendChild(root.createTextNode(str(element.RefElevation)))
    location.appendChild(elevation)

address = ifc_file.by_type('IfcPostalAddress')
for element in address:
    zipcode = root.createElement('ZipcodeOrPostalCode')
    zipcode.appendChild(root.createTextNode(element.PostalCode))
    location.appendChild(zipcode)
    name = root.createElement('Name')
    name.appendChild(root.createTextNode(element.Region + ', ' + element.Country))
    location.appendChild(name)

# Specify the 'Building' element of the gbXML schema; making use of IFC entity 'IfcBuilding'
# This new element is added as child to the earlier created 'Campus' element
buildings = ifc_file.by_type('IfcBuilding')
for element in buildings:
    building = root.createElement('Building')
    building.setAttribute('id', fix_xml_bldng(element.GlobalId))
    building.setAttribute('buildingType', "Unknown")
    campus.appendChild(building)

    dict_id[fix_xml_bldng(element.GlobalId)] = building

for element in address:
    streetAddress = root.createElement('StreetAddress')
    streetAddress.appendChild(root.createTextNode(element.Region + ', ' + element.Country))
    building.appendChild(streetAddress)

# Specify the 'BuildingStorey' element of the gbXML schema; making use of IFC entity
# 'IfcBuildingStorey'
# This new element is added as child to the earlier created 'Building' element
storeys = ifc_file.by_type('IfcBuildingStorey')
storey_name = 1
for element in storeys:
    buildingStorey = root.createElement('BuildingStorey')
    buildingStorey.setAttribute('id', fix_xml_stry(element.GlobalId))
    building.appendChild(buildingStorey)

    dict_id[fix_xml_stry(element.GlobalId)] = buildingStorey

    name = root.createElement('Name')
    name.appendChild(root.createTextNode('Storey_%d' % storey_name))
    storey_name = storey_name + 1
    buildingStorey.appendChild(name)

    level = root.createElement('Level')
    level.appendChild(root.createTextNode(str(element.Elevation)))
    buildingStorey.appendChild(level)

# Specify the 'Space' element of the gbXML schema; making use of IFC entity 'IfcSpace'
# This new element is added as child to the earlier created 'Building' element
spaces = ifc_file.by_type('IfcSpace')
space_name = 1
for s in spaces:
    space = root.createElement('Space')
    space.setAttribute('id', fix_xml_spc(s.GlobalId))
    building.appendChild(space)

    dict_id[fix_xml_spc(s.GlobalId)] = space

    # Refer to the relating 'BuildingStorey' GUID by iterating through IFC entities
    space.setAttribute('buildingStoreyIdRef', fix_xml_stry(s.Decomposes[0].RelatingObject.GlobalId))

    area = root.createElement('Area')
    volume = root.createElement('Volume')

    properties = s.IsDefinedBy
    for r in properties:
        if r.is_a('IfcRelDefinesByProperties'):
            if r.RelatingPropertyDefinition.is_a('IfcPropertySet'):
                for p in r.RelatingPropertyDefinition.HasProperties:
                    if p.Name == 'Area':
                        valueArea = p.NominalValue.wrappedValue
                        area.appendChild(root.createTextNode(str(valueArea)))
                        space.appendChild(area)
                    if p.Name == 'Volume':
                        valueVolume = p.NominalValue.wrappedValue
                        volume.appendChild(root.createTextNode(str(valueVolume)))
                        space.appendChild(volume)

    name = root.createElement('Name')
    name.appendChild(root.createTextNode('Space_%d' % space_name))
    space_name = space_name + 1
    space.appendChild(name)

    # Specify the 'SpaceBoundary' element of the gbXML schema; making use of IFC entity 'IfcSpace'
    # This new element is added as child to the earlier created 'Space' element
    boundaries = s.BoundedBy
    for element in boundaries:

        # Make sure a 'SpaceBoundary' is representing an actual element
        if element.RelatedBuildingElement == None:
            continue

        # Specify the 'IfcCurveBoundedPlane' entity which represents the geometry
        boundaryGeom = element.ConnectionGeometry.SurfaceOnRelatingElement

        if boundaryGeom.is_a('IfcCurveBoundedPlane') and boundaryGeom.InnerBoundaries is None:
            boundaryGeom.InnerBoundaries = ()

        print boundaryGeom

        # Use IfcOpenShell and OPENCASCADE to attach geometry to the specified IFC entity
        space_boundary_shape = ifcopenshell.geom.create_shape(settings, boundaryGeom)

        # Create 'SpaceBoundary' elements for the following building elements
        if element.RelatedBuildingElement.is_a('IfcCovering') or
        element.RelatedBuildingElement.is_a('IfcSlab') or \
                element.RelatedBuildingElement.is_a('IfcWallStandardCase'):

            # Exclude Roof elements, those shapes cause some malfunction
            # Note: Roof elements have no priority(Ceilings and Floors are used to ensure water tight
            # geometry)
            if element.is_a('IfcSlab'):
                if element.PredefinedType != 'FLOOR':
                    continue

            spaceBoundary = root.createElement('SpaceBoundary')
            spaceBoundary.setAttribute('isSecondLevelBoundary', "true")

            # Refer to the relating 'SpaceBoundary' GUID by iterating through IFC entities
            spaceBoundary.setAttribute('surfaceIdRef', fix_xml_id(element.GlobalId))

            space.appendChild(spaceBoundary)

            planarGeometry = root.createElement('PlanarGeometry')
            spaceBoundary.appendChild(planarGeometry)

            # Specify the 'PolyLoop' element which contains 4 'CartesianPoint' elements with each
            # 3 explicit 'Coordinate' elements. Note: if geometry is not attached to the
            # 'SpaceBoundary' element, the relationship between 'Space' and 'Building' elements is
            # handled only on a logical level. If geometry is attached, it is given within the local
            # coordinate systems of the 'Space' and (if given in addition) of the 'Building' element.

            # Z-coordinates are extracted by iterating through IFC entities to the 'IfcCartesianPoint'
            # of the related 'IfcBuildingStorey'
            print 'SpaceBoundary'

            new_z = element.RelatingSpace.ObjectPlacement.PlacementRelTo.RelativePlacement.
                Location.Coordinates[2]
            new_z = new_z / 1000

            polyLoop = root.createElement('PolyLoop')

            for v in get_vertices(space_boundary_shape):
                x, y, z = v
                z = z + new_z
                print x, y, z

                point = root.createElement('CartesianPoint')

                for c in x, y, z:
                    coord = root.createElement('Coordinate')
                    coord.appendChild(root.createTextNode(str(c)))
                    point.appendChild(coord)
                polyLoop.appendChild(point)

            planarGeometry.appendChild(polyLoop)

# Specify the 'Surface' element of the gbXML schema; making use of IFC entity 'IfcRelSpaceBoundary'
# This new element is added as child to the earlier created 'Campus' element
boundaries = ifc_file.by_type('IfcRelSpaceBoundary')
opening_id = 1
for element in boundaries:
    # Make sure a 'SpaceBoundary' is representing an actual element
    if element.RelatedBuildingElement == None:
        continue

    # Specify the 'IfcCurveBoundedPlane' entity which represents the geometry
    if element.ConnectionGeometry.SurfaceOnRelatingElement == None:
        continue

    surfaceGeom = element.ConnectionGeometry.SurfaceOnRelatingElement

    if surfaceGeom.is_a('IfcCurveBoundedPlane') and surfaceGeom.InnerBoundaries is None:
        surfaceGeom.InnerBoundaries = ()

    print surfaceGeom

    space_boundary_shape = ifcopenshell.geom.create_shape(settings, surfaceGeom)

    # Specify each 'Surface' element and set 'SurfaceType' attributes
        if element.RelatedBuildingElement.is_a('IfcCovering') or
        element.RelatedBuildingElement.is_a('IfcSlab') or element.\
            RelatedBuildingElement.is_a('IfcWallStandardCase'):

        # Exclude Roof elements, those shapes cause some malfunction
        # Note: Roof elements have no priority(Ceilings and Floors are used to ensure water tight
        # geometry)
        if element.is_a('IfcSlab'):
            if element.PredefinedType != 'FLOOR':
                continue

        surface = root.createElement('Surface')
        surface.setAttribute('id', fix_xml_id(element.GlobalId))
        dict_id[fix_xml_id(element.GlobalId)] = surface

        if element.RelatedBuildingElement.is_a('IfcCovering'):
            surface.setAttribute('surfaceType', 'Ceiling')

        if element.RelatedBuildingElement.is_a('IfcSlab'):
            surface.setAttribute('surfaceType', 'InteriorFloor')

        if element.RelatedBuildingElement.is_a('IfcWallStandardCase') and element.\
                InternalOrExternalBoundary == 'EXTERNAL':
            surface.setAttribute('surfaceType', 'ExteriorWall')

        if element.RelatedBuildingElement.is_a('IfcWallStandardCase') and element.\
                InternalOrExternalBoundary == 'INTERNAL':
            surface.setAttribute('surfaceType', 'InteriorWall')

        # Refer to the relating 'IfcRelAssociatesMaterial' GUID by iterating through IFC entities
        surface.setAttribute('constructionIdRef', fix_xml_cons(element.RelatedBuildingElement.
                                                               HasAssociations[0].GlobalId))

        name = root.createElement('Name')
        name.appendChild(root.createTextNode(fix_xml_name(element.GlobalId)))

        surface.appendChild(name)

        adjacentSpaceId = root.createElement('AdjacentSpaceId')

        # Refer to the relating 'Space' GUID by iterating through IFC entities
        adjacentSpaceId.setAttribute('spaceIdRef', fix_xml_spc(element.RelatingSpace.GlobalId))
        surface.appendChild(adjacentSpaceId)

        planarGeometry = root.createElement('PlanarGeometry')
        surface.appendChild(planarGeometry)
        # Specify the 'PolyLoop' element which contains 4 'CartesianPoint' elements with each
        # 3 explicit 'Coordinate' elements. Note: if geometry is not attached to the
        # 'SpaceBoundary' element, the relationship between 'Space' and 'Building' elements is
        # handled only on a logical level. If geometry is attached, it is given within the local
        # coordinate systems of the 'Space' and (if given in addition) of the 'Building' element.

        # Z-coordinates are extracted by iterating through IFC entities to the 'IfcCartesianPoint'
        # of the related 'IfcBuildingStorey'
        print "Surface"

        new_z = element.RelatingSpace.ObjectPlacement.PlacementRelTo.RelativePlacement.
                Location.Coordinates[2]
        new_z = new_z / 1000

        polyLoop = root.createElement('PolyLoop')

        for v in get_vertices(space_boundary_shape):
            x, y, z = v
            z = z + new_z
            print x, y, z

            point = root.createElement('CartesianPoint')

            for c in x, y, z:
                coord = root.createElement('Coordinate')
                coord.appendChild(root.createTextNode(str(c)))
                point.appendChild(coord)

            polyLoop.appendChild(point)

        planarGeometry.appendChild(polyLoop)

        objectId = root.createElement('CADObjectId')
        objectId.appendChild(root.createTextNode(fix_xml_name(element.GlobalId)))
        surface.appendChild(objectId)

        campus.appendChild(surface)

    if element.RelatedBuildingElement.is_a('IfcWindow'):
        opening = root.createElement('Opening')

        # Refer to the relating 'IfcWindow' GUID by iterating through IFC entities
        opening.setAttribute('windowTypeIdRef', fix_xml_id(element.RelatedBuildingElement.GlobalId))
        opening.setAttribute('openingType', 'OperableWindow')

        opening.setAttribute('id', 'Opening%d' % opening_id)
        opening_id = opening_id + 1

        # If the building element is an 'IfcWindow' the gbXML element 'Opening' is added
        print 'Opening'
        planarGeometry = root.createElement('PlanarGeometry')
        opening.appendChild(planarGeometry)

        # Specify the 'PolyLoop' element which contains 4 'CartesianPoint' elements with each
        # 3 explicit 'Coordinate' elements. Note: if geometry is not attached to the
        # 'SpaceBoundary' element, the relationship between 'Space' and 'Building' elements is
        # handled only on a logical level. If geometry is attached, it is given within the local
        # coordinate systems of the 'Space' and (if given in addition) of the 'Building' element.

        # Z-coordinates are extracted by iterating through IFC entities to the 'IfcCartesianPoint'
        # of the related 'IfcBuildingStorey'
        polyLoop = root.createElement('PolyLoop')
        new_z = element.RelatingSpace.ObjectPlacement.PlacementRelTo.RelativePlacement.
                        Location.Coordinates[2]
        new_z = new_z / 1000

        for v in get_vertices(space_boundary_shape):
            x, y, z = v
            z = z + new_z
            print x, y, z

            point = root.createElement('CartesianPoint')

            for c in x, y, z:
                coord = root.createElement('Coordinate')
                coord.appendChild(root.createTextNode(str(c)))
                point.appendChild(coord)

            polyLoop.appendChild(point)

        planarGeometry.appendChild(polyLoop)

        name = root.createElement('Name')
        name.appendChild(root.createTextNode(fix_xml_name(element.RelatedBuildingElement.Name)))
        opening.appendChild(name)

        objectId = root.createElement('CADObjectId')
        objectId.appendChild(root.createTextNode(fix_xml_name(element.RelatedBuildingElement.Name)))
        opening.appendChild(objectId)

        surface.appendChild(opening)

    else:
        continue

# Specify the 'WindowType' element of the gbXML schema; making use of IFC entity 'IfcWindow'
# This new element is added as child to the earlier created 'gbXML' element
windows = ifc_file.by_type('IfcWindow')
for element in windows:
    window = root.createElement('WindowType')
    window.setAttribute('id', fix_xml_id(element.GlobalId))
    gbxml.appendChild(window)

    dict_id[fix_xml_id(element.GlobalId)] = window

    name = root.createElement('Name')
    name.appendChild(root.createTextNode(fix_xml_name(element.Name)))
    window.appendChild(name)

    description = root.createElement('Description')
    description.appendChild(root.createTextNode(fix_xml_name(element.Name)))
    window.appendChild(description)

    # Specify analytical properties of the 'IfcWindow' by iterating through IFC entities
    analyticValue = element.IsDefinedBy

    u_value = root.createElement('U-value')
    for r in analyticValue:
        if r.is_a("IfcRelDefinesByProperties"):
            if r.RelatingPropertyDefinition.is_a('IfcPropertySet'):
                for p in r.RelatingPropertyDefinition.HasProperties:
                    if p.Name == 'ThermalTransmittance':
                        valueU = p.NominalValue.wrappedValue
                        u_value.setAttribute('unit', 'WPerSquareMeterK')
                        u_value.appendChild(root.createTextNode(str(valueU)))
                        window.appendChild(u_value)

    solarHeat = root.createElement('SolarHeatGainCoeff')
    visualLight = root.createElement('Transmittance')
    for r in analyticValue:
        if r.is_a('IfcRelDefinesByType'):
            if r.RelatingType.is_a('IfcWindowStyle'):
                for p in r.RelatingType.HasPropertySets:
                    if p.Name == 'Analytical Properties(Type)':
                        for t in p.HasProperties:
                            if t.Name == 'Solar Heat Gain Coefficient':
                                valueSolar = t.NominalValue.wrappedValue
                                solarHeat.setAttribute('unit', 'Fraction')
                                solarHeat.appendChild(root.createTextNode(str(valueSolar)))
                                window.appendChild(solarHeat)

                            if t.Name == 'Visual Light Transmittance':
                                valueLight = t.NominalValue.wrappedValue
                                visualLight.setAttribute('unit', 'Fraction')
                                visualLight.setAttribute('type', 'Visible')
                                visualLight.appendChild(root.createTextNode(str(valueLight)))
                                window.appendChild(visualLight)
```

```
# Specify the 'Construction' element of the gbXML schema; making use of IFC entity
# 'IfcRelSpaceBoundary'
# This new element is added as child to the earlier created 'gbXML' element
listCon = []

for element in boundaries:
    # Make sure a 'SpaceBoundary' is representing an actual element
    if element.RelatedBuildingElement is None:
        continue

    if element.RelatedBuildingElement.is_a('IfcCovering') or
    element.RelatedBuildingElement.is_a('IfcSlab') or
    element.RelatedBuildingElement.is_a('IfcWallStandardCase'):

        # Exclude Roof elements, those shapes cause some malfunction
        # Note: Roof elements have no priority(Ceilings and Floors are used to ensure water tight
        # geometry)
        if element.is_a('IfcSlab'):
            if element.PredefinedType != 'FLOOR':
                continue

        # Refer to the relating 'IfcRelAssociatesMaterial' GUID by iterating through IFC entities
        constructions = element.RelatedBuildingElement.HasAssociations[0].GlobalId

        # Make use of a list to make sure no same 'Construction' elements are added twice
        if constructions not in listCon:
            listCon.append(constructions)

            construction = root.createElement('Construction')
            construction.setAttribute('id',
            fix_xml_cons(element.RelatedBuildingElement.HasAssociations[0].GlobalId))
            dict_id[fix_xml_cons(element.RelatedBuildingElement.HasAssociations[0].GlobalId)] =
            construction

            # Specify analytical properties of the 'Construction' element by iterating through IFC
            # entities
            analyticValue = element.RelatedBuildingElement.IsDefinedBy

            u_value = root.createElement('U-value')
            for r in analyticValue:
                if r.is_a('IfcRelDefinesByProperties'):
                    if r.RelatingPropertyDefinition.is_a('IfcPropertySet'):
                        for p in r.RelatingPropertyDefinition.HasProperties:
                            if element.RelatedBuildingElement.is_a("IfcWallStandardCase"):
                                if p.Name == 'ThermalTransmittance':
                                    valueU = p.NominalValue.wrappedValue
                                    u_value.setAttribute('unit', 'WPerSquareMeterK')
                                    u_value.appendChild(root.createTextNode(str(valueU)))
                                    construction.appendChild(u_value)

                            if p.Name == 'Heat Transfer Coefficient (U)':
                                valueU = p.NominalValue.wrappedValue
                                u_value.setAttribute('unit', 'WPerSquareMeterK')
                                u_value.appendChild(root.createTextNode(str(valueU)))
                                construction.appendChild(u_value)

            absorptance = root.createElement('Absorptance')
            for r in analyticValue:
                if r.is_a('IfcRelDefinesByProperties'):
                    if r.RelatingPropertyDefinition.is_a('IfcPropertySet'):
                        for p in r.RelatingPropertyDefinition.HasProperties:
                            if p.Name == 'Absorptance':
                                valueAb = p.NominalValue.wrappedValue
                                absorptance.setAttribute('unit', 'Fraction')
                                absorptance.setAttribute('type', 'ExtIR')
                                absorptance.appendChild(root.createTextNode(str(valueAb)))
                                construction.appendChild(absorptance)

            # Refer to the relating 'IfcRelAssociatesMaterial' GUID by iterating through IFC entities
            layerId = fix_xml_layer(element.RelatedBuildingElement.HasAssociations[0].GlobalId)

            layer_id = root.createElement('LayerId')
            layer_id.setAttribute('layerIdRef', layerId)
            construction.appendChild(layer_id)
            # Refer to the relating 'IfcMaterialLayerSet' name by iterating through IFC entities
            name = root.createElement('Name')
            name.appendChild(root.createTextNode(element.RelatedBuildingElement.HasAssociations[0].
            RelatingMaterial.ForLayerSet.LayerSetName))

            construction.appendChild(name)

            gbxml.appendChild(construction)

    else:
        continue

# Specify the 'Layer' element of the gbXML schema; making use of IFC entity 'IfcBuildingElement'
# This new element is added as child to the earlier created 'gbXML' element
buildingElements = ifc_file.by_type('IfcBuildingElement')
for element in buildingElements:
    if element.is_a('IfcWallStandardCase') or element.is_a('IfcCovering') or element.is_a('IfcSlab'):

        # Exclude Roof elements, those shapes cause some malfunction
        # Note: Roof elements have no priority(Ceilings and Floors are used to ensure water tight
        # geometry)
        if element.is_a('IfcSlab'):
            if element.PredefinedType != 'FLOOR':
                continue

        # Refer to the relating 'IfcRelAssociatesMaterial' GUID by iterating through IFC entities
        layerId = fix_xml_layer(element.HasAssociations[0].GlobalId)

        layer = root.createElement('Layer')
        layer.setAttribute('id', layerId)

        dict_id[layerId] = layer

        # Specify the 'IfcMaterialLayer' entity and iterate to each 'IfcMaterial' entity
        materials = element.HasAssociations[0].RelatingMaterial.ForLayerSet.MaterialLayers
        for l in materials:
            material_id = root.createElement('MaterialId')
            material_id.setAttribute('materialIdRef', "mat_%d" % l.Material.id())
            layer.appendChild(material_id)

            dict_id["mat_%d" % l.Material.id()] = layer

            gbxml.appendChild(layer)

    else:
        continue

# Specify the 'Material' element of the gbXML schema; making use of IFC entity 'IfcBuildingElement'
# This new element is added as child to the earlier created 'gbXML' element
listMat = []

for element in buildingElements:
    if element.is_a('IfcWallStandardCase') or element.is_a("IfcSlab") or element.is_a('IfcCovering'):

        # Exclude Roof elements, those shapes cause some malfunction
        # Note: Roof elements have no priority(Ceilings and Floors are used to ensure water tight
        # geometry)
        if element.is_a('IfcSlab'):
            if element.PredefinedType != 'FLOOR':
                continue

        materials = element.HasAssociations[0].RelatingMaterial.ForLayerSet.MaterialLayers

        for l in materials:
            item = l.Material.id()

            # Make use of a list to make sure no same 'Materials' elements are added twice
            if item not in listMat:
                listMat.append(item)

                material = root.createElement('Material')
                material.setAttribute('id', "mat_%d" % l.Material.id())
                dict_id["mat_%d" % l.Material.id()] = material
                name = root.createElement('Name')
                name.appendChild(root.createTextNode(l.Material.Name))
                material.appendChild(name)

                thickness = root.createElement('Thickness')
                thickness.setAttribute('unit', 'Meters')
                valueT = l.LayerThickness / 1000
                thickness.appendChild(root.createTextNode((str(valueT))))
                material.appendChild(thickness)

                rValue = root.createElement('R-value')
                rValue.setAttribute('unit', 'SquareMeterKPerW')

                # Specify analytical properties of the 'Material' element by iterating through IFC
                # entities
                thermalResistance = element.IsDefinedBy
                for r in thermalResistance:
                    if r.is_a('IfcRelDefinesByType'):
                        if r.RelatingType.is_a('IfcWallType'):
                            for p in r.RelatingType.HasPropertySets:
                                if p.Name == 'Analytical Properties(Type)':
                                    for t in p.HasProperties:
                                        if t.Name == 'Heat Transfer Coefficient (U)':
                                            valueU = t.NominalValue.wrappedValue
                                            valueR = valueT / valueU
                                            rValue.appendChild(root.createTextNode(str(valueR)))
                                            material.appendChild(rValue)

                                            gbxml.appendChild(material)

                    if r.is_a('IfcRelDefinesByProperties'):
                        if r.RelatingPropertyDefinition.is_a('IfcPropertySet'):
                            for p in r.RelatingPropertyDefinition.HasProperties:
                                if p.Name == 'Heat Transfer Coefficient (U)':
                                    valueU = p.NominalValue.wrappedValue
                                    valueR = valueT / valueU
                                    rValue.setAttribute('unit', 'SquareMeterKPerW')
                                    rValue.appendChild(root.createTextNode(str(valueR)))
                                    material.appendChild(rValue)

                                    gbxml.appendChild(material)

                    if element.is_a('IfcCovering'):
                        if r.is_a('IfcRelDefinesByProperties'):
                            if r.RelatingType.is_a('IfcPropertySet'):
                                for p in r.RelatingType.HasPropertySets:
                                    if p.Name == 'Analytical Properties(Type)':
                                        for t in p.HasProperties:
                                            if t.Name == 'Heat Transfer Coefficient (U)':
                                                valueU = t.NominalValue.wrappedValue
                                                valueR = valueT / valueU
                                                rValue.setAttribute('unit', 'SquareMeterKPerW')
                                                rValue.appendChild(root.createTextNode(str(valueR)))
                                                material.appendChild(rValue)

                                                gbxml.appendChild(material)

    else:
        continue

# Specify the 'DocumentHistory' element of the gbXML schema; making use of IFC entity 'IfcApplication'
# and 'IfcPerson'
# This new element is added as child to the earlier created 'gbXML' element
programInfo = ifc_file.by_type('IfcApplication')
docHistory = root.createElement('DocumentHistory')
for element in programInfo:
    program = root.createElement('ProgramInfo')
    program.setAttribute('id', element.ApplicationIdentifier)
    docHistory.appendChild(program)

    company = root.createElement('CompanyName')
    company.appendChild(root.createTextNode(element.ApplicationDeveloper.Name))
    program.appendChild(company)
    product = root.createElement('ProductName')
    product.appendChild(root.createTextNode(element.ApplicationFullName))
    program.appendChild(product)

    version = root.createElement('Version')
    version.appendChild(root.createTextNode(element.Version))
    program.appendChild(version)

personInfo = ifc_file.by_type('IfcPerson')
for element in personInfo:
    created = root.createElement('CreatedBy')
    created.setAttribute('personId', element.GivenName)

for element in programInfo:
    created.setAttribute('programId', element.ApplicationIdentifier)

    today = datetime.date.today()
    created.setAttribute('date', today.strftime('%Y-%m-%dT') + time.strftime('%H:%M:%S'))
    docHistory.appendChild(created)

for element in personInfo:
    person = root.createElement('PersonInfo')
    person.setAttribute('id', element.GivenName)
    docHistory.appendChild(person)

gbxml.appendChild(docHistory)

# Create a new XML file and write all created elements to it
xml_str = root.toprettyxml(indent="\t", encoding="UTF-8")

save_path_file = "New_Exported_gbXML.xml"

with open(save_path_file, "w") as f:
    f.write(xml_str)
```

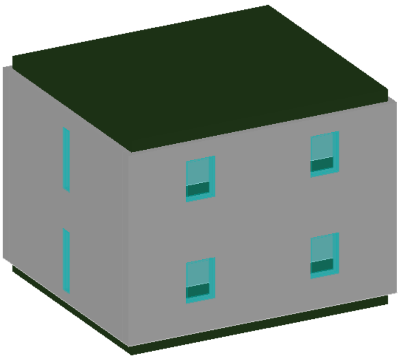
Appendix VII – Validation results

TEST CASE 1

(1) Optical IFC test with the FZKViewer:

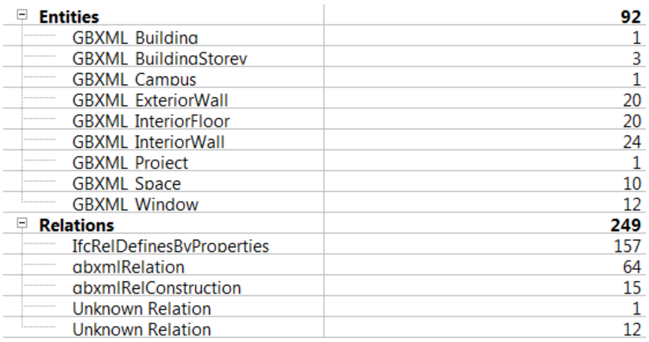


⊟ **Entities**	**58**
IfcBuildina	1
IfcBuildinaStorev	3
IfcOpeninaElement	15
IfcProiect	1
IfcSite	1
IfcSlab[Floorl	3
IfcSpace	10
IfcWallStandardCase	12
IfcWindow	12
⊟ **Relations**	**477**
IfcRelAaareaates	5
IfcRelAssociatesMaterial	31
IfcRelConnectsPathElements	20
IfcRelContainedInSpatialStructui	3
IfcRelDefinesBvProperties	311
IfcRelDefinesBvTvpe	4
IfcRelFillsElement	12
IfcRelSpaceBoundarv	76
IfcRelVoidsElement	15
⊟ **EntitvTvpes**	**2**
IFCWALLTYPE	2
IFCWINDOWSTYLE	2

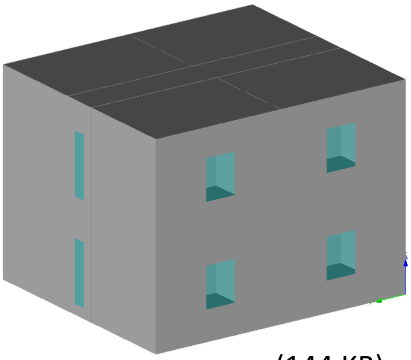


(218 KB)

(2) Optical gbXML test with the FZKViewer:

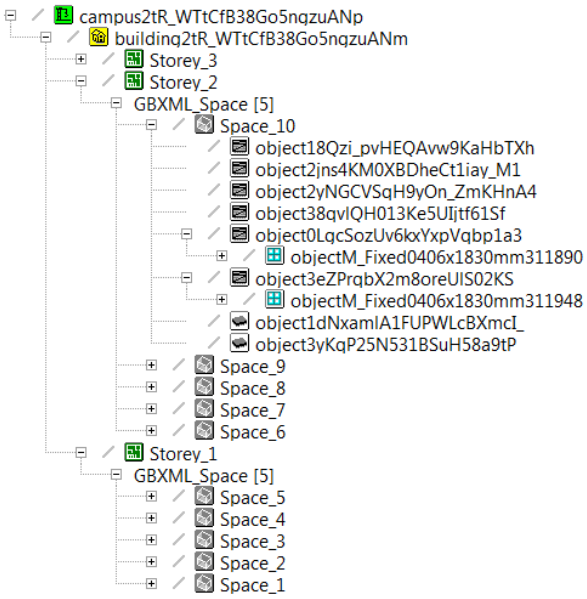


⊟ **Entities**	**92**
GBXML_Buildina	1
GBXML_BuildinaStorev	3
GBXML_Campus	1
GBXML_ExteriorWall	20
GBXML_InteriorFloor	20
GBXML_InteriorWall	24
GBXML_Proiect	1
GBXML_Space	10
GBXML_Window	12
⊟ **Relations**	**249**
IfcRelDefinesBvProperties	157
abxmlRelation	64
abxmlRelConstruction	15
Unknown Relation	1
Unknown Relation	12



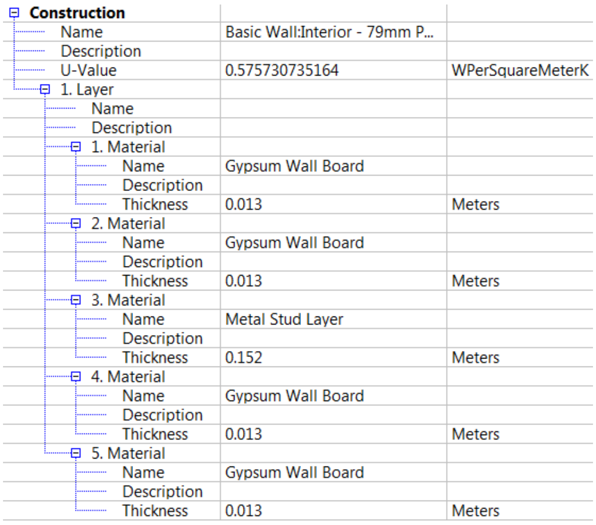
(144 KB)

gbXML campus tree structure



```
⊟ campus2tR_WTtCfB38Go5nqzuANp
  ⊟ building2tR_WTtCfB38Go5nqzuANm
    ⊞ Storey_3
    ⊟ Storey_2
      ⊟ GBXML_Space [5]
        ⊟ Space_10
            object18Qzi_pvHEQAvw9KaHbTXh
            object2jns4KM0XBDheCt1iay_M1
            object2yNGCVSqH9yOn_ZmKHnA4
            object38qvIQH013Ke5UIjtf61Sf
          ⊟ object0LqcSozUv6kxYxpVqbp1a3
            ⊞ objectM_Fixed0406x1830mm311890
          ⊟ object3eZPrqbX2m8oreUIS02KS
            ⊞ objectM_Fixed0406x1830mm311948
            object1dNxamIA1FUPWLcBXmcI_
            object3yKqP25N531BSuH58a9tP
        ⊞ Space_9
        ⊞ Space_8
        ⊞ Space_7
        ⊞ Space_6
    ⊟ Storey_1
      ⊟ GBXML_Space [5]
        ⊞ Space_5
        ⊞ Space_4
        ⊞ Space_3
        ⊞ Space_2
        ⊞ Space_1
```

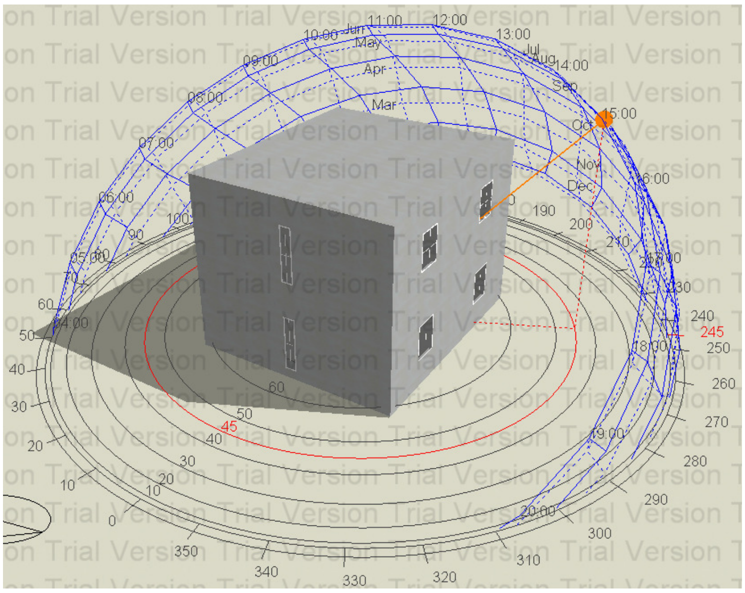
Property relationships (e.g. Interior Wall)



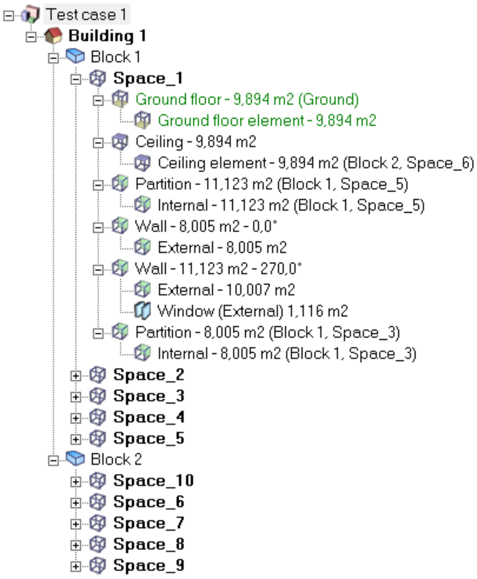
⊟ **Construction**		
Name	Basic Wall:Interior - 79mm P...	
Description		
U-Value	0.575730735164	WPerSquareMeterK
⊟ 1. Layer		
Name		
Description		
⊟ 1. Material		
Name	Gypsum Wall Board	
Description		
Thickness	0.013	Meters
⊟ 2. Material		
Name	Gypsum Wall Board	
Description		
Thickness	0.013	Meters
⊟ 3. Material		
Name	Metal Stud Layer	
Description		
Thickness	0.152	Meters
⊟ 4. Material		
Name	Gypsum Wall Board	
Description		
Thickness	0.013	Meters
⊟ 5. Material		
Name	Gypsum Wall Board	
Description		
Thickness	0.013	Meters

(3) Simulation in DesignBuilder:

Visualization (July 15 at 15:00 hrs)

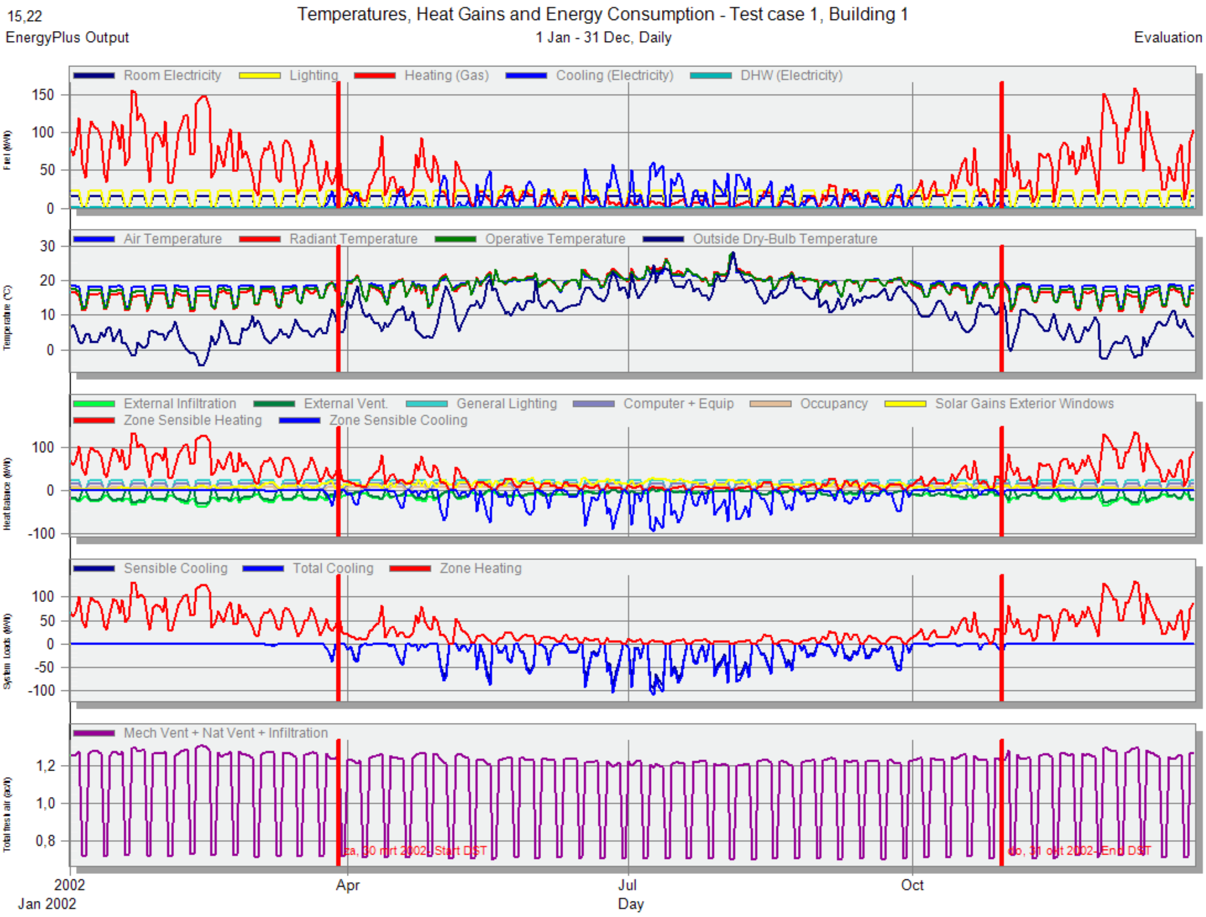


Tree structure

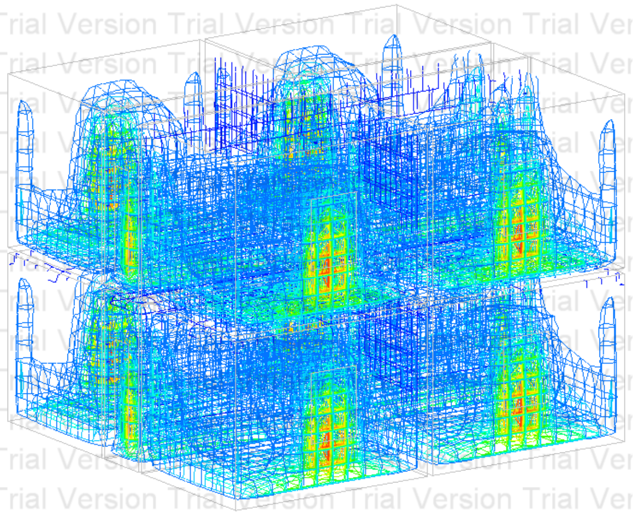


```
Test case 1
  Building 1
    Block 1
      Space_1
        Ground floor - 9,894 m2 (Ground)
          Ground floor element - 9,894 m2
        Ceiling - 9,894 m2
          Ceiling element - 9,894 m2 (Block 2, Space_6)
        Partition - 11,123 m2 (Block 1, Space_5)
          Internal - 11,123 m2 (Block 1, Space_5)
        Wall - 8,005 m2 - 0,0°
          External - 8,005 m2
        Wall - 11,123 m2 - 270,0°
          External - 10,007 m2
          Window (External) 1,116 m2
        Partition - 8,005 m2 (Block 1, Space_3)
          Internal - 8,005 m2 (Block 1, Space_3)
      Space_2
      Space_3
      Space_4
      Space_5
    Block 2
      Space_10
      Space_6
      Space_7
      Space_8
      Space_9
```

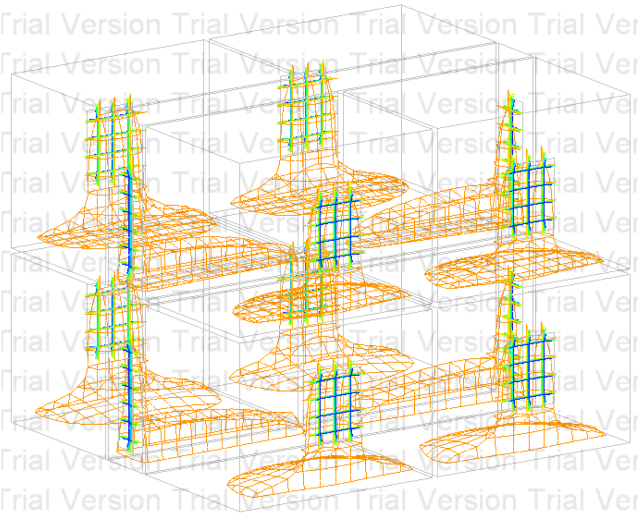
Simulation with EnergyPlus



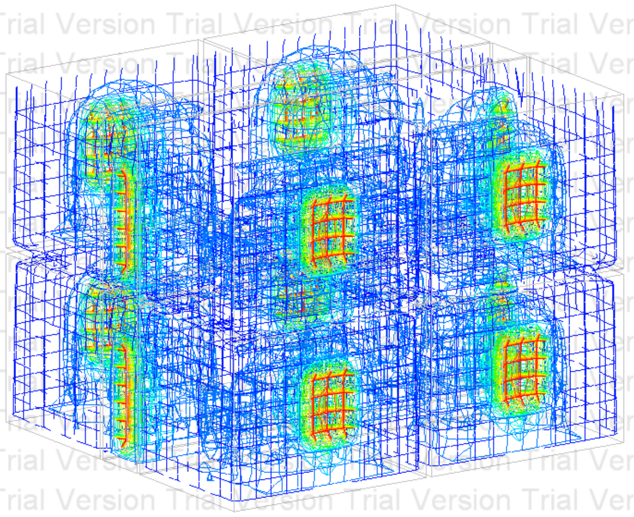
Computational Fluid Dynamics (CFD) analysis



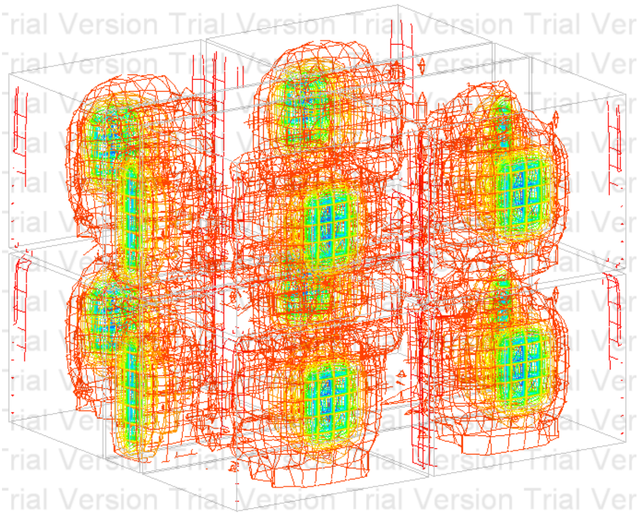
(velocity) (m/s)



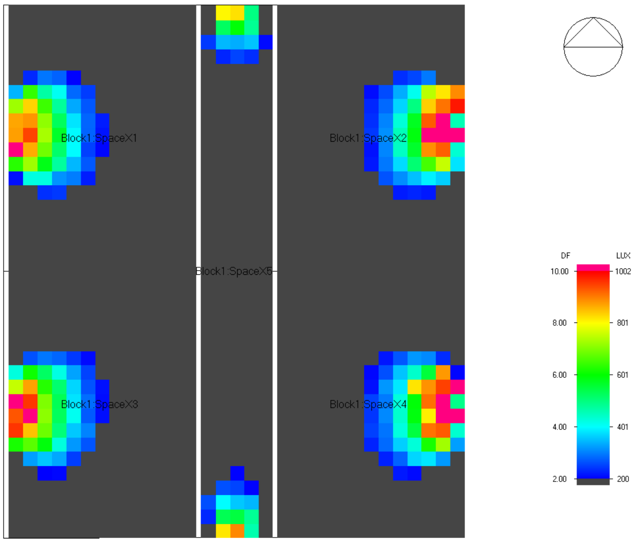
(temperature) (C)



(predicted Percentage of Dissatisfied) (%)



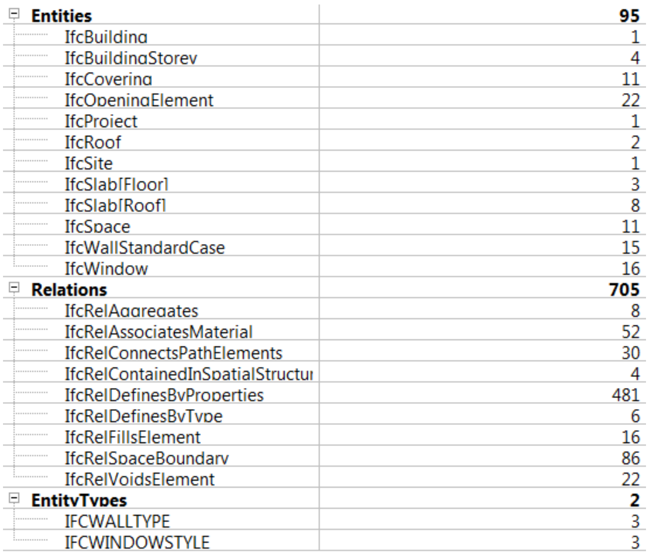
(operative temperature) (C)



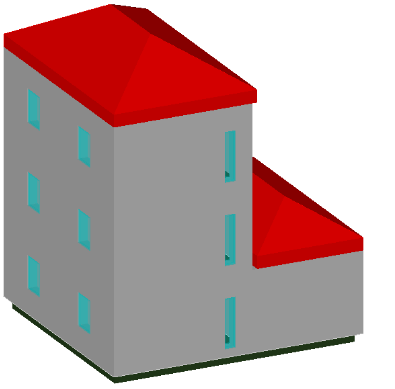
(daylighting) (LUX)

TEST CASE 2

(1) Optical IFC test with the FZKViewer:

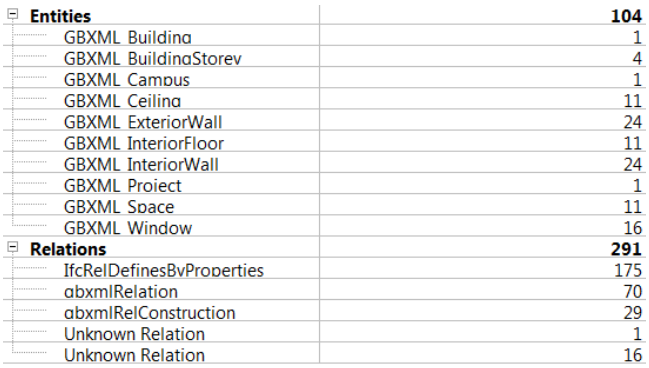


Entities	95
IfcBuilding	1
IfcBuildingStorey	4
IfcCovering	11
IfcOpeningElement	22
IfcProject	1
IfcRoof	2
IfcSite	1
IfcSlab[Floor]	3
IfcSlab[Roof]	8
IfcSpace	11
IfcWallStandardCase	15
IfcWindow	16
Relations	**705**
IfcRelAggregates	8
IfcRelAssociatesMaterial	52
IfcRelConnectsPathElements	30
IfcRelContainedInSpatialStructu	4
IfcRelDefinesByProperties	481
IfcRelDefinesByType	6
IfcRelFillsElement	16
IfcRelSpaceBoundary	86
IfcRelVoidsElement	22
EntityTypes	**2**
IFCWALLTYPE	3
IFCWINDOWSTYLE	3

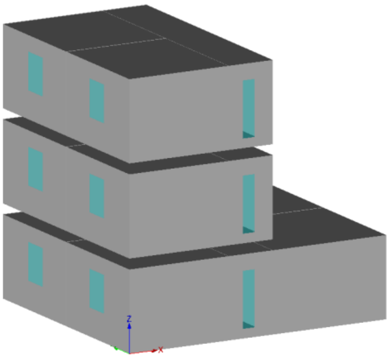


(307 KB)

(2) Optical gbXML test with the FZKViewer:

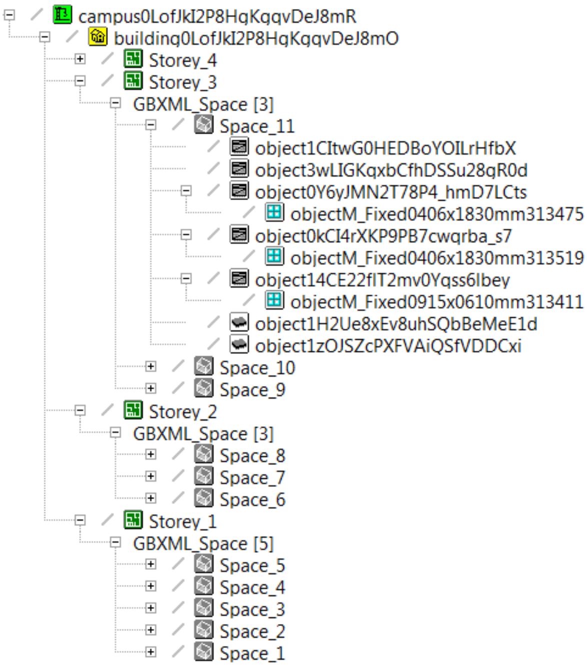


Entities	104
GBXML Building	1
GBXML BuildingStorey	4
GBXML Campus	1
GBXML Ceiling	11
GBXML ExteriorWall	24
GBXML InteriorFloor	11
GBXML InteriorWall	24
GBXML Project	1
GBXML Space	11
GBXML Window	16
Relations	**291**
IfcRelDefinesByProperties	175
gbxmlRelation	70
gbxmlRelConstruction	29
Unknown Relation	1
Unknown Relation	16



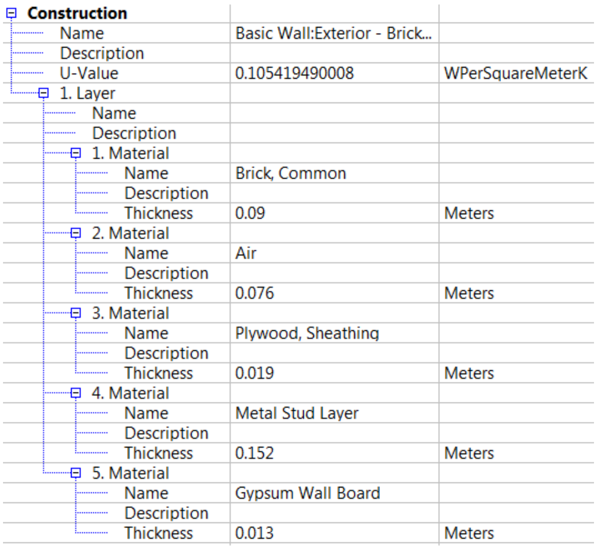
(167 KB)

gbXML campus tree structure



```
⊟ campus0LofJkI2P8HqKqqvDeJ8mR
  ⊟ building0LofJkI2P8HqKqqvDeJ8mO
    ⊞ Storey_4
    ⊟ Storey_3
      ⊟ GBXML_Space [3]
        ⊟ Space_11
            object1CItwG0HEDBoYOILrHfbX
            object3wLIGKqxbCfhDSSu28qR0d
          ⊟ object0Y6yJMN2T78P4_hmD7LCts
              objectM_Fixed0406x1830mm313475
          ⊟ object0kCI4rXKP9PB7cwqrba_s7
              objectM_Fixed0406x1830mm313519
          ⊟ object14CE22fIT2mv0Yqss6lbey
              objectM_Fixed0915x0610mm313411
            object1H2Ue8xEv8uhSQbBeMeE1d
            object1zOJSZcPXFVAiQSfVDDCxi
        ⊞ Space_10
        ⊞ Space_9
    ⊟ Storey_2
      ⊟ GBXML_Space [3]
        ⊞ Space_8
        ⊞ Space_7
        ⊞ Space_6
    ⊟ Storey_1
      ⊟ GBXML_Space [5]
        ⊞ Space_5
        ⊞ Space_4
        ⊞ Space_3
        ⊞ Space_2
        ⊞ Space_1
```

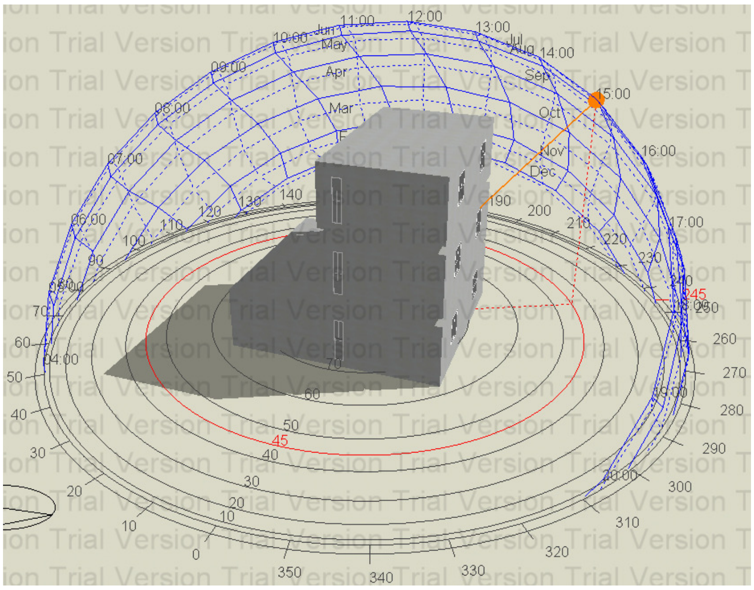
Property relationships (e.g. Brick Wall)



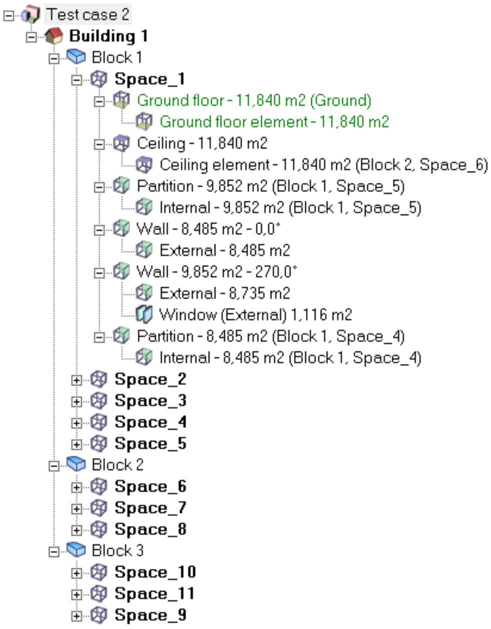
Construction		
Name	Basic Wall:Exterior - Brick...	
Description		
U-Value	0.105419490008	WPerSquareMeterK
1. Layer		
Name		
Description		
1. Material		
Name	Brick, Common	
Description		
Thickness	0.09	Meters
2. Material		
Name	Air	
Description		
Thickness	0.076	Meters
3. Material		
Name	Plywood, Sheathing	
Description		
Thickness	0.019	Meters
4. Material		
Name	Metal Stud Layer	
Description		
Thickness	0.152	Meters
5. Material		
Name	Gypsum Wall Board	
Description		
Thickness	0.013	Meters

(3) Simulation in DesignBuilder:

Visualization (July 15 at 15:00 hrs)

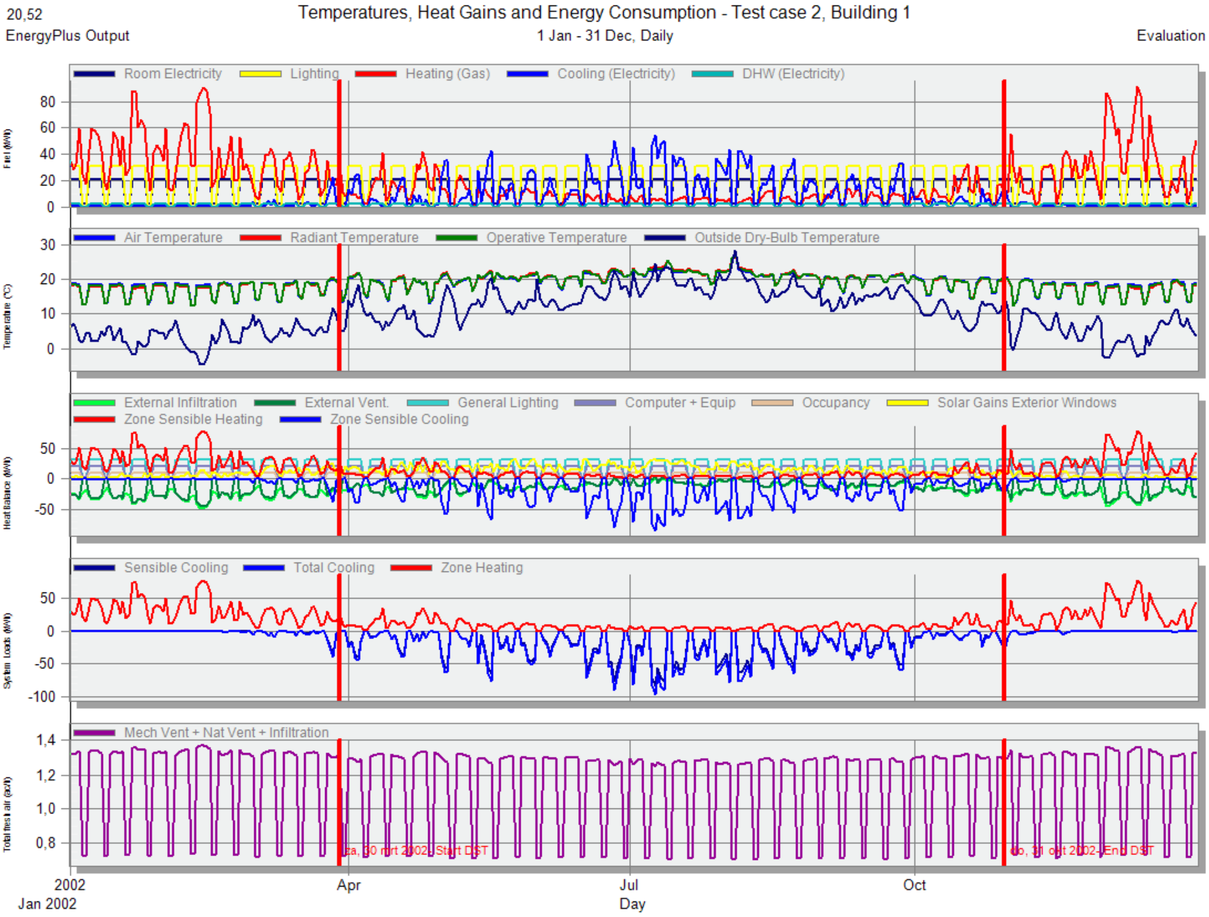


Tree structure

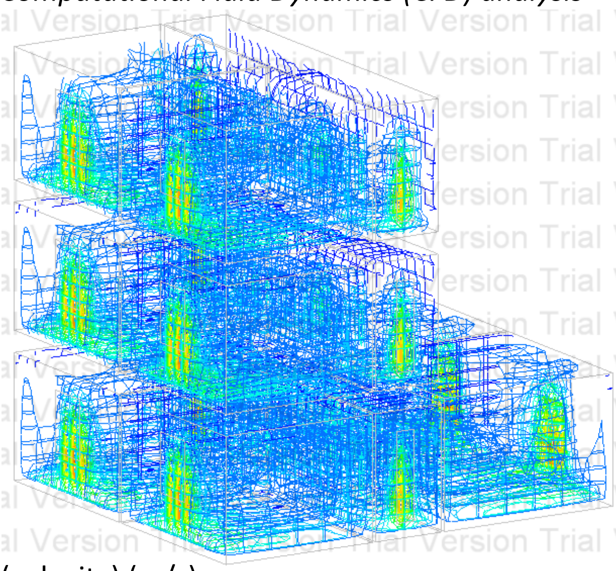


- Test case 2
  - **Building 1**
    - Block 1
      - **Space_1**
        - Ground floor - 11,840 m2 (Ground)
          - Ground floor element - 11,840 m2
        - Ceiling - 11,840 m2
          - Ceiling element - 11,840 m2 (Block 2, Space_6)
        - Partition - 9,852 m2 (Block 1, Space_5)
          - Internal - 9,852 m2 (Block 1, Space_5)
        - Wall - 8,485 m2 - 0,0°
          - External - 8,485 m2
        - Wall - 9,852 m2 - 270,0°
          - External - 8,735 m2
          - Window (External) 1,116 m2
        - Partition - 8,485 m2 (Block 1, Space_4)
          - Internal - 8,485 m2 (Block 1, Space_4)
      - **Space_2**
      - **Space_3**
      - **Space_4**
      - **Space_5**
    - Block 2
      - **Space_6**
      - **Space_7**
      - **Space_8**
    - Block 3
      - **Space_10**
      - **Space_11**
      - **Space_9**

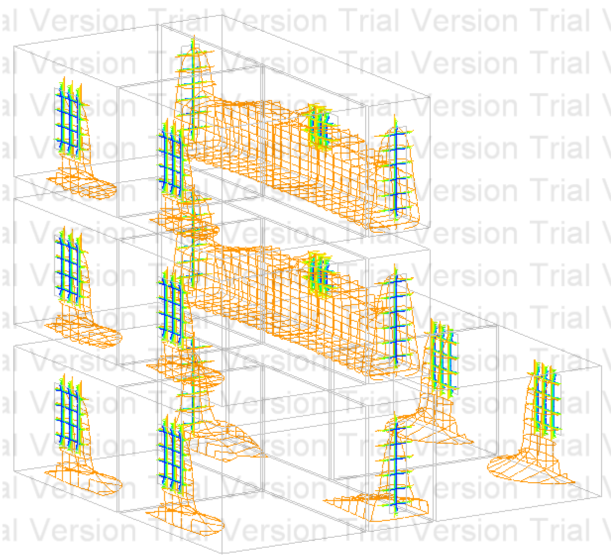
Simulation with EnergyPlus



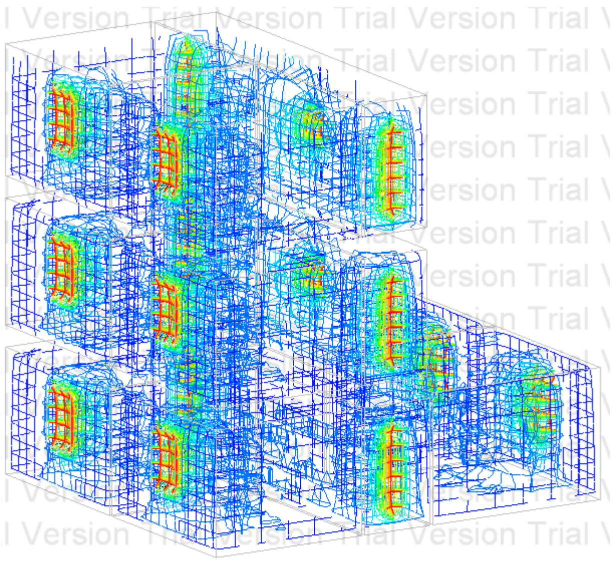
Computational Fluid Dynamics (CFD) analysis



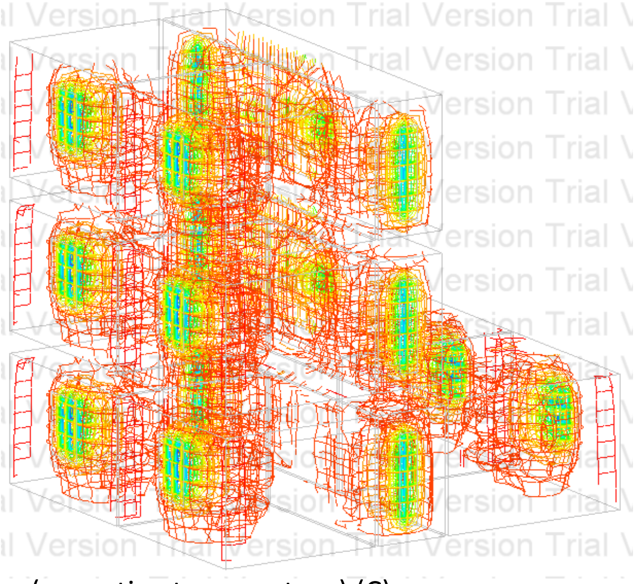
(velocity) (m/s)



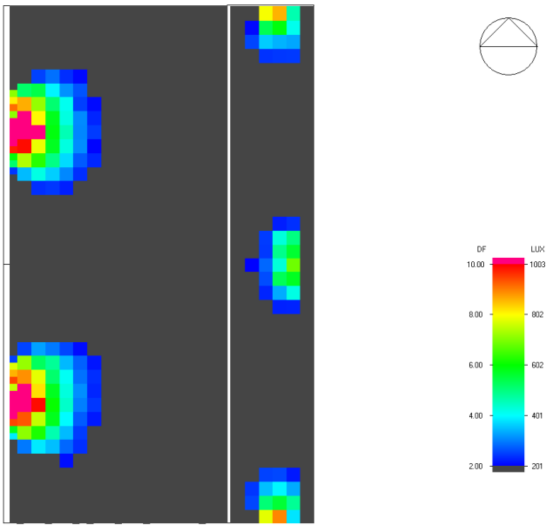
(temperature) (C)



(predicted Percentage of Dissatisfied) (%)



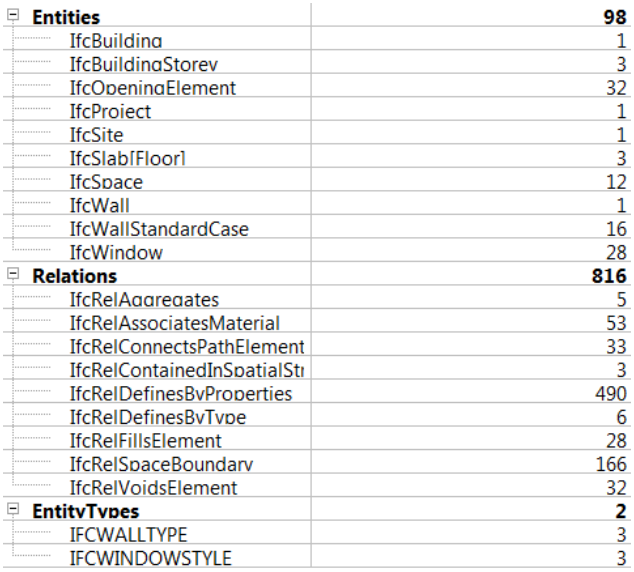
(operative temperature) (C)



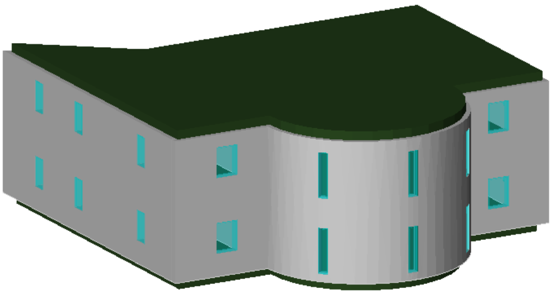
(daylighting) (LUX)

TEST CASE 3

(1) Optical IFC test with the FZKViewer:

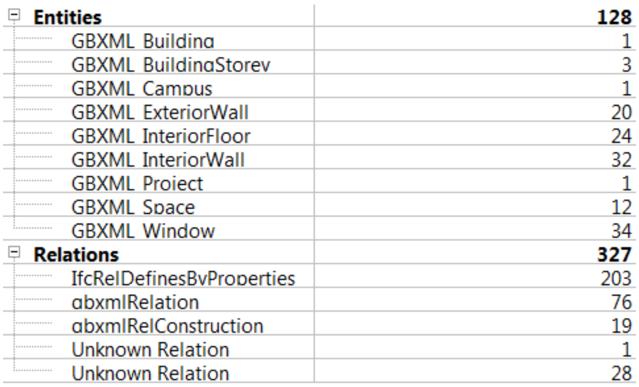


Entities	**98**
IfcBuilding	1
IfcBuildingStorey	3
IfcOpeningElement	32
IfcProject	1
IfcSite	1
IfcSlab[Floor]	3
IfcSpace	12
IfcWall	1
IfcWallStandardCase	16
IfcWindow	28
Relations	**816**
IfcRelAggregates	5
IfcRelAssociatesMaterial	53
IfcRelConnectsPathElement	33
IfcRelContainedInSpatialSt	3
IfcRelDefinesByProperties	490
IfcRelDefinesByType	6
IfcRelFillsElement	28
IfcRelSpaceBoundary	166
IfcRelVoidsElement	32
EntityTypes	**2**
IFCWALLTYPE	3
IFCWINDOWSTYLE	3

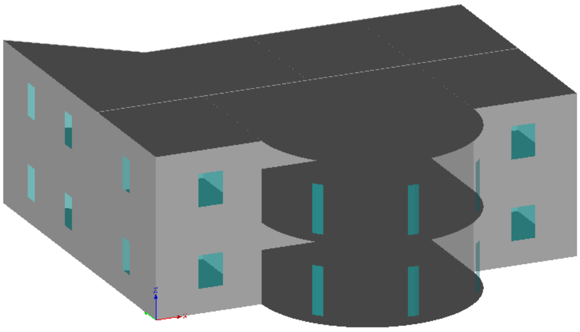


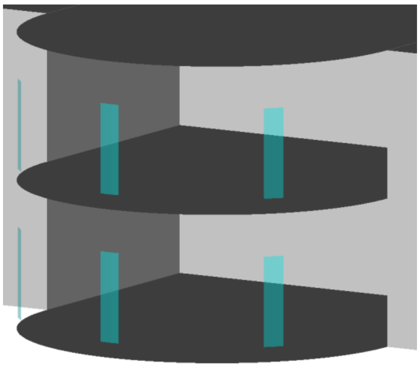
(543 KB)

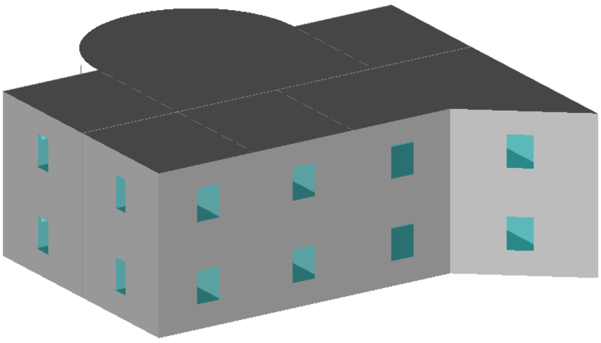
(2) Optical gbXML test with the FZKViewer:



Entities	**128**
GBXML Building	1
GBXML BuildingStorey	3
GBXML Campus	1
GBXML ExteriorWall	20
GBXML InteriorFloor	24
GBXML InteriorWall	32
GBXML Project	1
GBXML Space	12
GBXML Window	34
Relations	**327**
IfcRelDefinesByProperties	203
gbxmlRelation	76
gbxmlRelConstruction	19
Unknown Relation	1
Unknown Relation	28

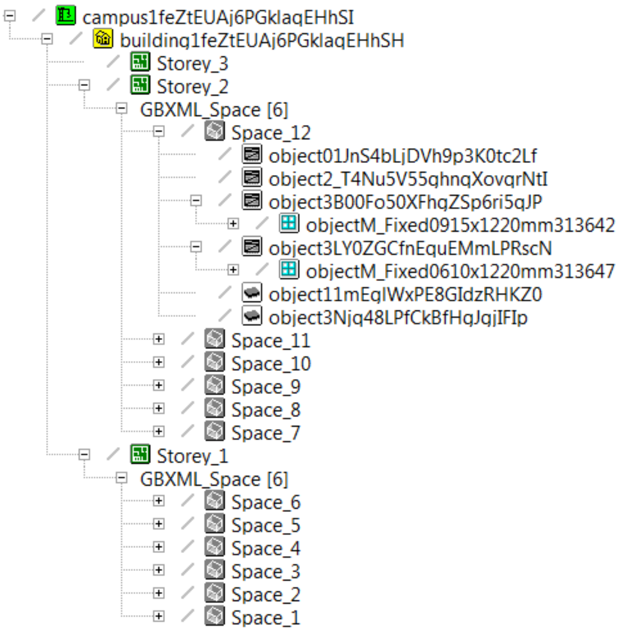






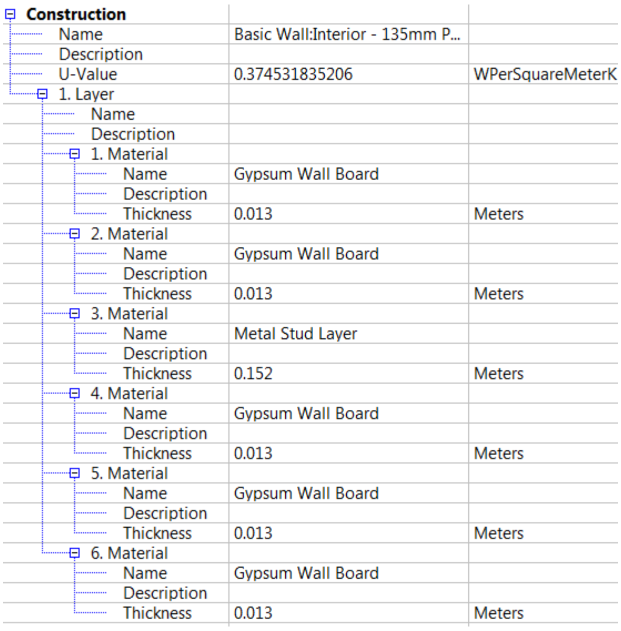
(274 KB)

(Angled walls are not supported, but windows positioned in angled walls are divided into two faces) *gbXML campus tree structure*



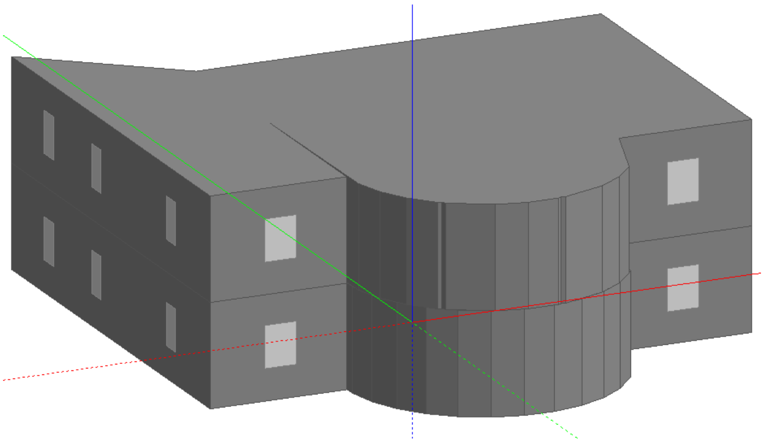
```
campus1feZtEUAj6PGklaqEHhSI
  building1feZtEUAj6PGklaqEHhSH
    Storey_3
    Storey_2
      GBXML_Space [6]
        Space_12
          object01JnS4bLjDVh9p3K0tc2Lf
          object2_T4Nu5V55qhnqXovqrNtI
          object3B00Fo50XFhqZSp6ri5qJP
            objectM_Fixed0915x1220mm313642
          object3LY0ZGCfnEquEMmLPRscN
            objectM_Fixed0610x1220mm313647
          object11mEqlWxPE8GIdzRHKZ0
          object3Njq48LPfCkBfHqJqjIFIp
        Space_11
        Space_10
        Space_9
        Space_8
        Space_7
    Storey_1
      GBXML_Space [6]
        Space_6
        Space_5
        Space_4
        Space_3
        Space_2
        Space_1
```

Property relationships (e.g. Interior Wall)



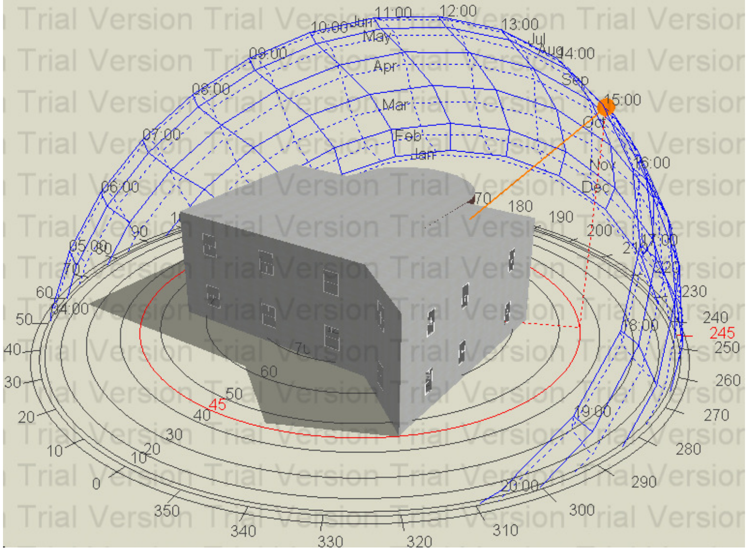
Construction		
Name	Basic Wall:Interior - 135mm P...	
Description		
U-Value	0.374531835206	WPerSquareMeterK
1. Layer		
Name		
Description		
1. Material		
Name	Gypsum Wall Board	
Description		
Thickness	0.013	Meters
2. Material		
Name	Gypsum Wall Board	
Description		
Thickness	0.013	Meters
3. Material		
Name	Metal Stud Layer	
Description		
Thickness	0.152	Meters
4. Material		
Name	Gypsum Wall Board	
Description		
Thickness	0.013	Meters
5. Material		
Name	Gypsum Wall Board	
Description		
Thickness	0.013	Meters
6. Material		
Name	Gypsum Wall Board	
Description		
Thickness	0.013	Meters

(3) Simulation in DesignBuilder:

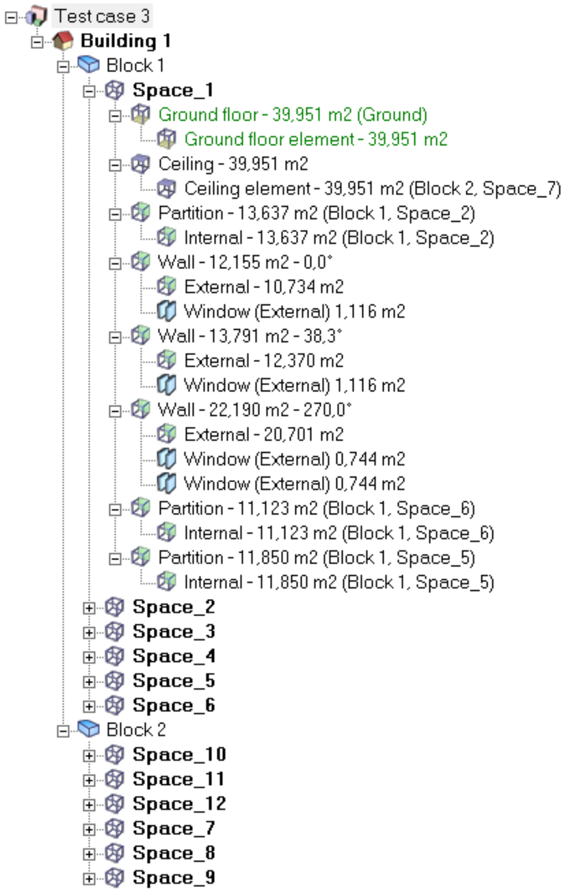


(DesignBuilder creates new faces to enclose spaces and run energy simulations)

Visualization (July 15 at 15:00 hrs)

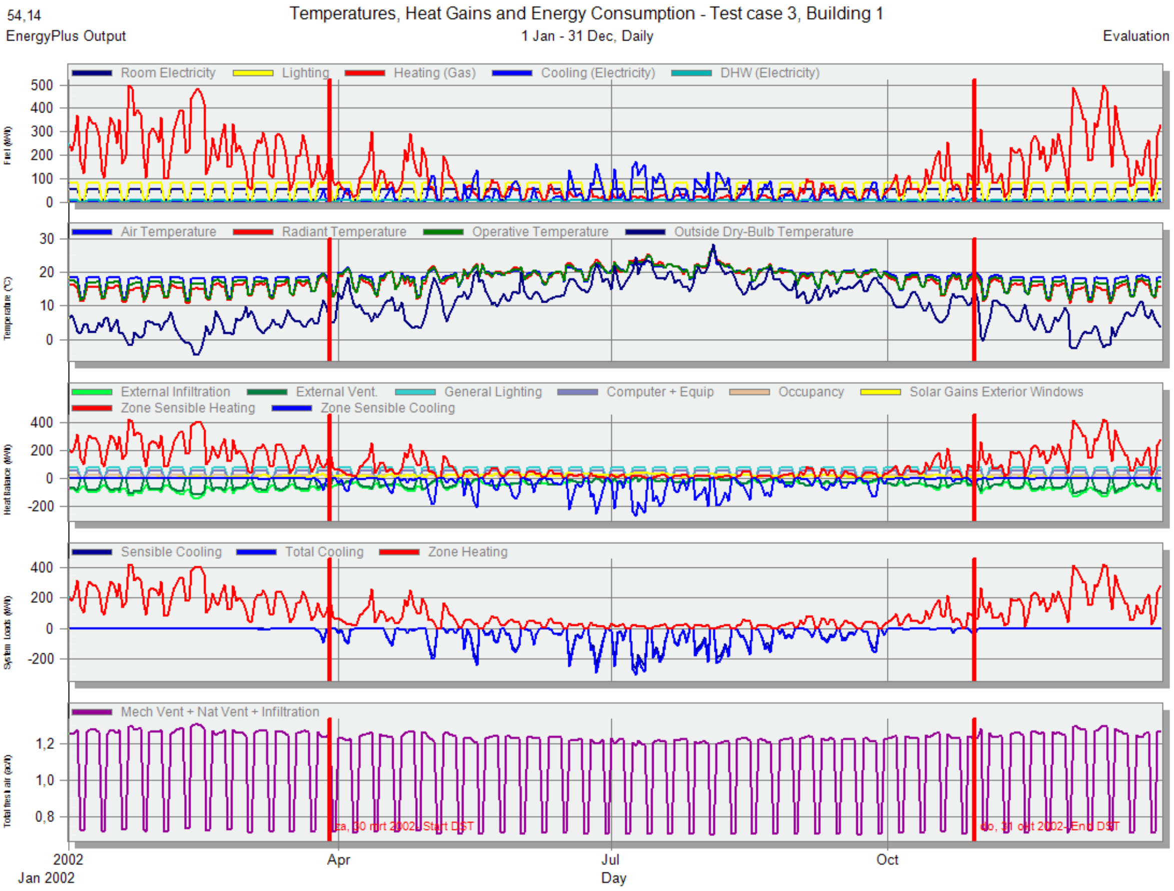


Tree structure

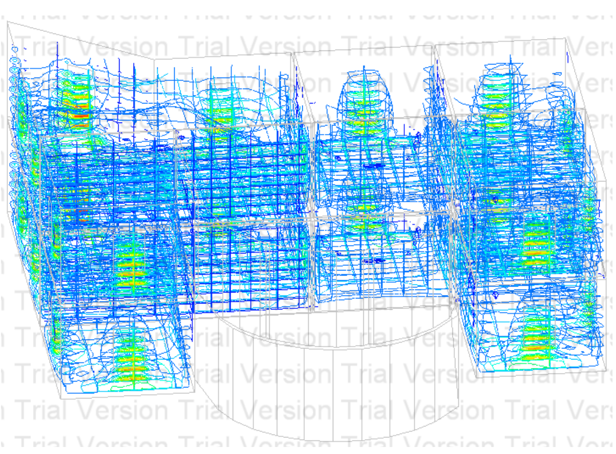


- Test case 3
  - Building 1
    - Block 1
      - Space_1
        - Ground floor - 39,951 m2 (Ground)
          - Ground floor element - 39,951 m2
        - Ceiling - 39,951 m2
          - Ceiling element - 39,951 m2 (Block 2, Space_7)
        - Partition - 13,637 m2 (Block 1, Space_2)
          - Internal - 13,637 m2 (Block 1, Space_2)
        - Wall - 12,155 m2 - 0,0°
          - External - 10,734 m2
          - Window (External) 1,116 m2
        - Wall - 13,791 m2 - 38,3°
          - External - 12,370 m2
          - Window (External) 1,116 m2
        - Wall - 22,190 m2 - 270,0°
          - External - 20,701 m2
          - Window (External) 0,744 m2
          - Window (External) 0,744 m2
        - Partition - 11,123 m2 (Block 1, Space_6)
          - Internal - 11,123 m2 (Block 1, Space_6)
        - Partition - 11,850 m2 (Block 1, Space_5)
          - Internal - 11,850 m2 (Block 1, Space_5)
      - Space_2
      - Space_3
      - Space_4
      - Space_5
      - Space_6
    - Block 2
      - Space_10
      - Space_11
      - Space_12
      - Space_7
      - Space_8
      - Space_9

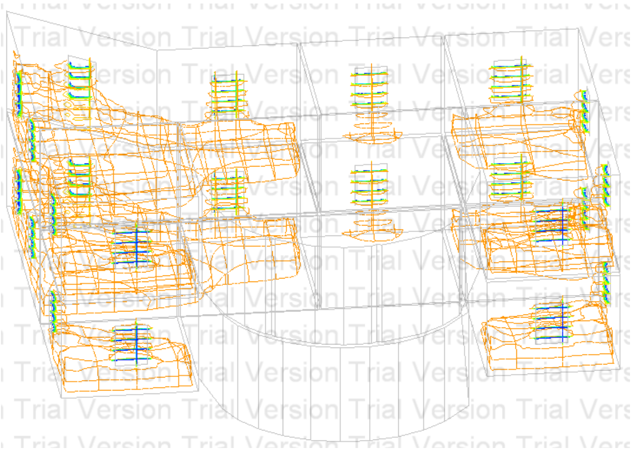
Simulation with EnergyPlus



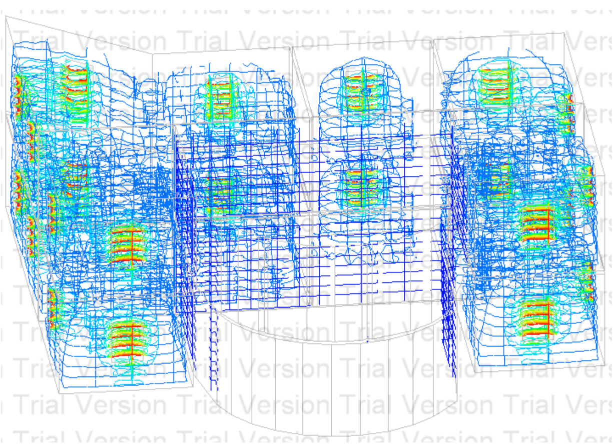
Computational Fluid Dynamics (CFD) analysis



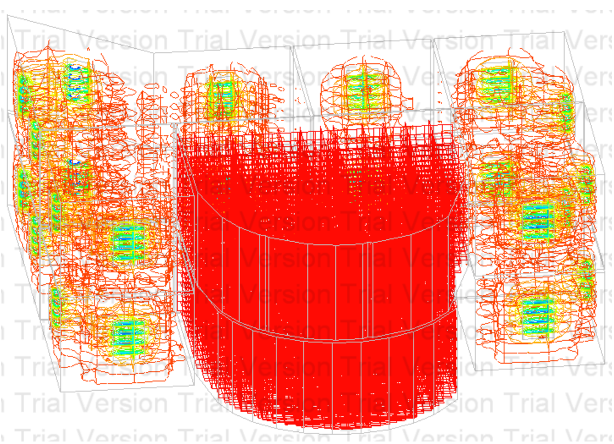
(velocity) (m/s)



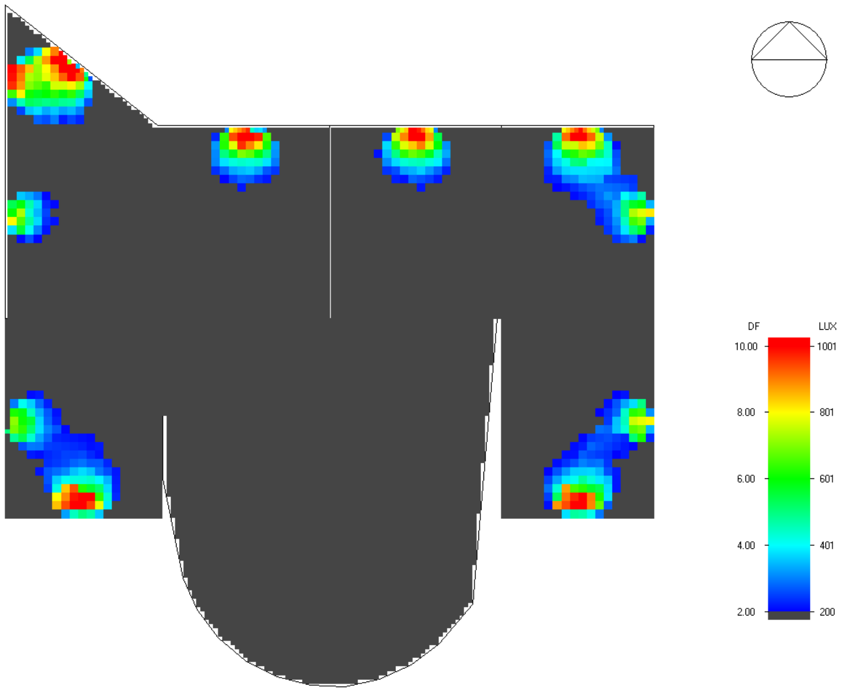
(temperature) (C)



(predicted Percentage of Dissatisfied) (%)



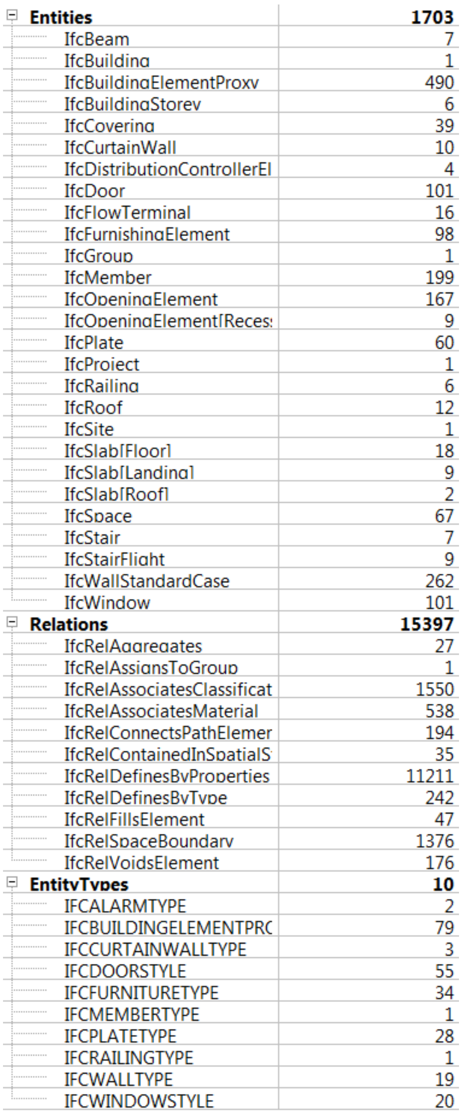
(operative temperature) (C)



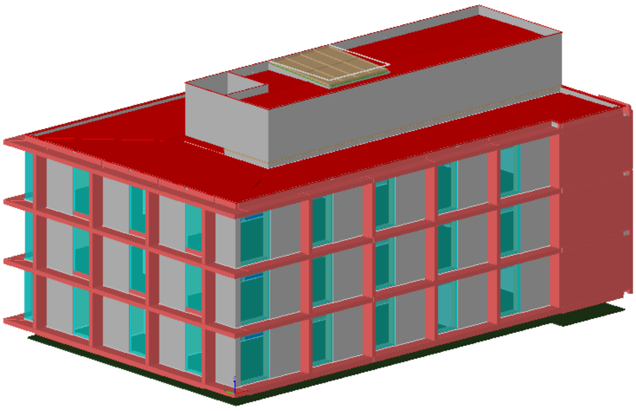
(daylighting) (LUX)

TEST CASE 4

(1) Optical IFC test with the FZKViewer:

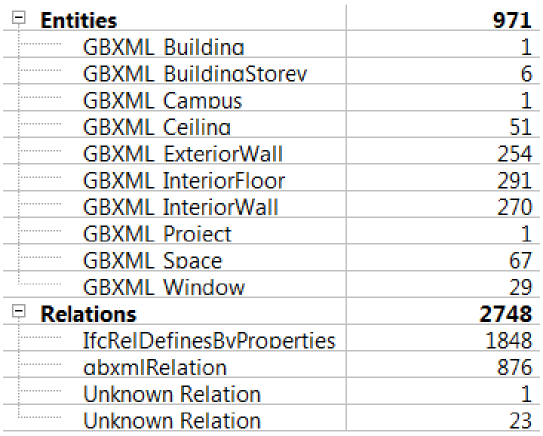


Entities	**1703**
IfcBeam	7
IfcBuilding	1
IfcBuildingElementProxy	490
IfcBuildingStorey	6
IfcCovering	39
IfcCurtainWall	10
IfcDistributionControllerEl	4
IfcDoor	101
IfcFlowTerminal	16
IfcFurnishingElement	98
IfcGroup	1
IfcMember	199
IfcOpeningElement	167
IfcOpeningElement[Reces:	9
IfcPlate	60
IfcProject	1
IfcRailing	6
IfcRoof	12
IfcSite	1
IfcSlab[Floor]	18
IfcSlab[Landing]	9
IfcSlab[Roof]	2
IfcSpace	67
IfcStair	7
IfcStairFlight	9
IfcWallStandardCase	262
IfcWindow	101
Relations	**15397**
IfcRelAggregates	27
IfcRelAssignsToGroup	1
IfcRelAssociatesClassificat	1550
IfcRelAssociatesMaterial	538
IfcRelConnectsPathElemer	194
IfcRelContainedInSpatialS	35
IfcRelDefinesByProperties	11211
IfcRelDefinesByType	242
IfcRelFillsElement	47
IfcRelSpaceBoundary	1376
IfcRelVoidsElement	176
EntityTypes	**10**
IFCALARMTYPE	2
IFCBUILDINGELEMENTPRC	79
IFCCURTAINWALLTYPE	3
IFCDOORSTYLE	55
IFCFURNITURETYPE	34
IFCMEMBERTYPE	1
IFCPLATETYPE	28
IFCRAILINGTYPE	1
IFCWALLTYPE	19
IFCWINDOWSTYLE	20

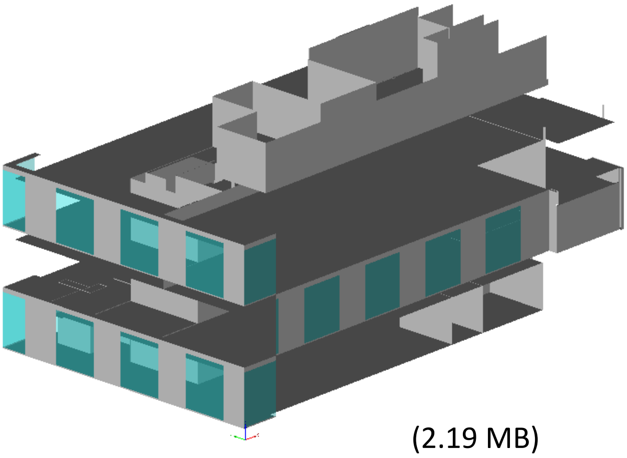


(13.0 MB)

(2) Optical gbXML test with the FZKViewer:

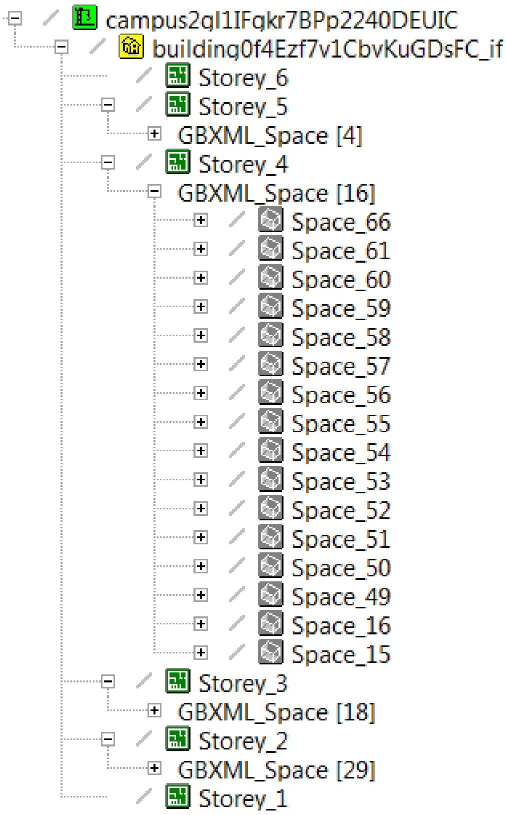


Entities	**971**
GBXML Building	1
GBXML BuildingStorey	6
GBXML Campus	1
GBXML Ceiling	51
GBXML ExteriorWall	254
GBXML InteriorFloor	291
GBXML InteriorWall	270
GBXML Project	1
GBXML Space	67
GBXML Window	29
Relations	**2748**
IfcRelDefinesByProperties	1848
gbxmlRelation	876
Unknown Relation	1
Unknown Relation	23



(2.19 MB)

gbXML campus tree structure (building storey 4)



```
campus2ql1IFqkr7BPp2240DEUIC
  building0f4Ezf7v1CbvKuGDsFC_if
    Storey_6
    Storey_5
      GBXML_Space [4]
    Storey_4
      GBXML_Space [16]
        Space_66
        Space_61
        Space_60
        Space_59
        Space_58
        Space_57
        Space_56
        Space_55
        Space_54
        Space_53
        Space_52
        Space_51
        Space_50
        Space_49
        Space_16
        Space_15
    Storey_3
      GBXML_Space [18]
    Storey_2
      GBXML_Space [29]
    Storey_1
```

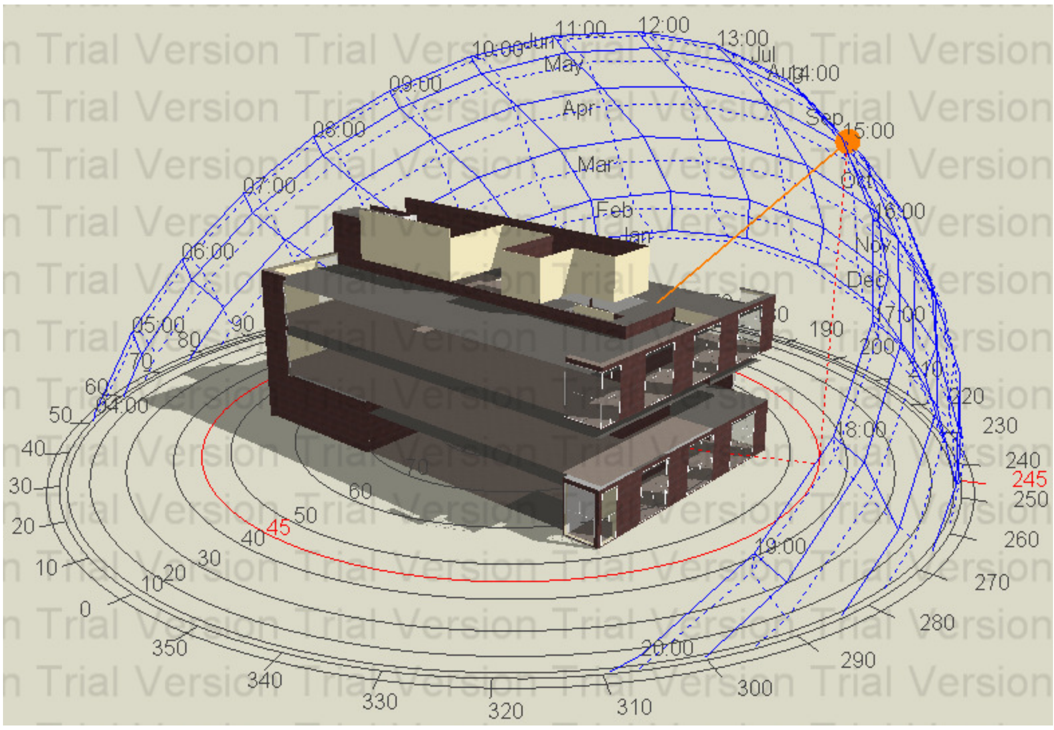
(3) Simulation in DesignBuilder:

Geometry related error message after import

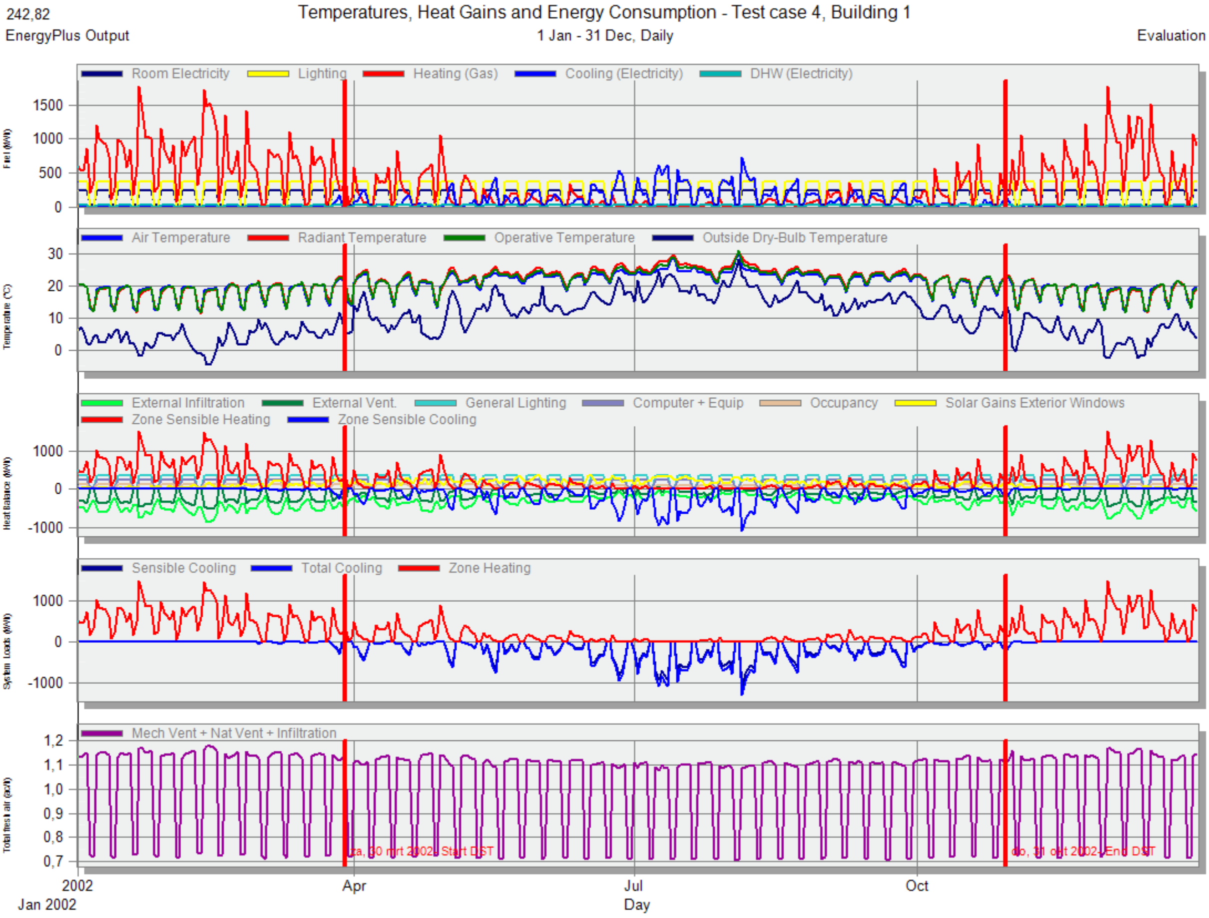
List of blocks not fully enclosed with one or more gap in surfaces:

- Space_3
- Space_9
- Space_12
- Space_14
- Space_17
- Space_19
- Space_23
- Space_25
- Space_26
- Space_27
- Space_28
- Space_29
- Space_30
- Space_35
- Space_37
- Space_39
- Space_40
- Space_41
- Space_42
- Space_43
- Space_49
- Space_51
- Space_53
- Space_54
- Space_55
- Space_57
- Space_62

Visualization (July 15 at 15:00 hrs)

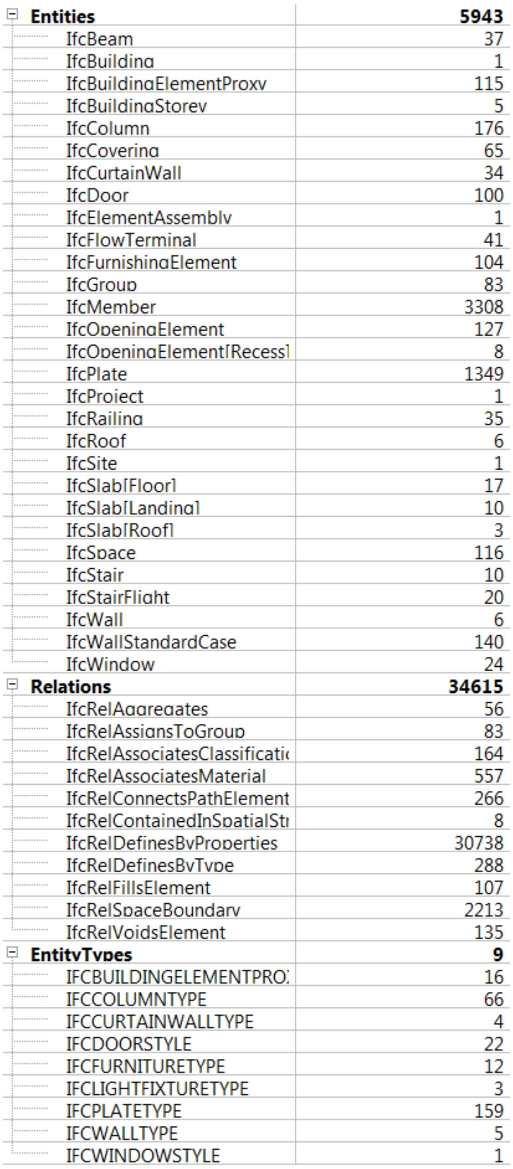


Simulation with EnergyPlus

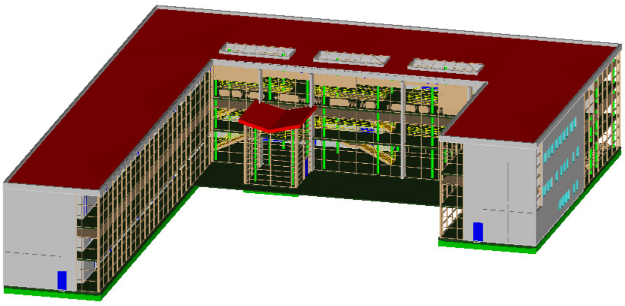


TEST CASE 5

(1) Optical IFC test with the FZKViewer:

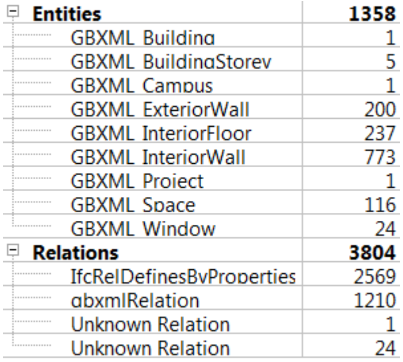


Entities	**5943**
IfcBeam	37
IfcBuilding	1
IfcBuildingElementProxy	115
IfcBuildingStorey	5
IfcColumn	176
IfcCovering	65
IfcCurtainWall	34
IfcDoor	100
IfcElementAssembly	1
IfcFlowTerminal	41
IfcFurnishingElement	104
IfcGroup	83
IfcMember	3308
IfcOpeningElement	127
IfcOpeningElement[Recess]	8
IfcPlate	1349
IfcProject	1
IfcRailing	35
IfcRoof	6
IfcSite	1
IfcSlab[Floor]	17
IfcSlab[Landing]	10
IfcSlab[Roof]	3
IfcSpace	116
IfcStair	10
IfcStairFlight	20
IfcWall	6
IfcWallStandardCase	140
IfcWindow	24
Relations	**34615**
IfcRelAggregates	56
IfcRelAssignsToGroup	83
IfcRelAssociatesClassificati	164
IfcRelAssociatesMaterial	557
IfcRelConnectsPathElement	266
IfcRelContainedInSpatialSt	8
IfcRelDefinesByProperties	30738
IfcRelDefinesByType	288
IfcRelFillsElement	107
IfcRelSpaceBoundary	2213
IfcRelVoidsElement	135
EntityTypes	**9**
IFCBUILDINGELEMENTPRO	16
IFCCOLUMNTYPE	66
IFCCURTAINWALLTYPE	4
IFCDOORSTYLE	22
IFCFURNITURETYPE	12
IFCLIGHTFIXTURETYPE	3
IFCPLATETYPE	159
IFCWALLTYPE	5
IFCWINDOWSTYLE	1

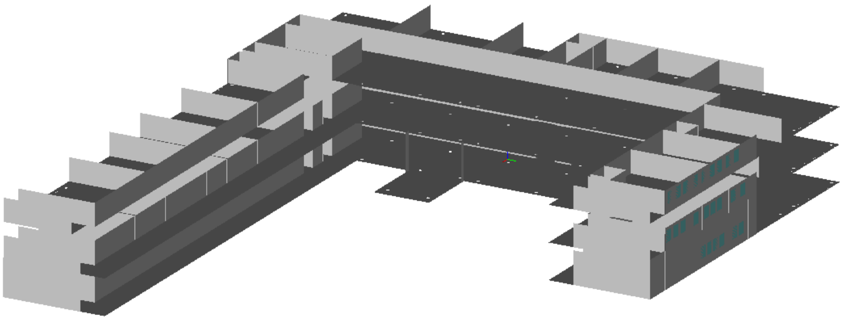


(60.8 MB)

(2) Optical gbXML test with the FZKViewer:

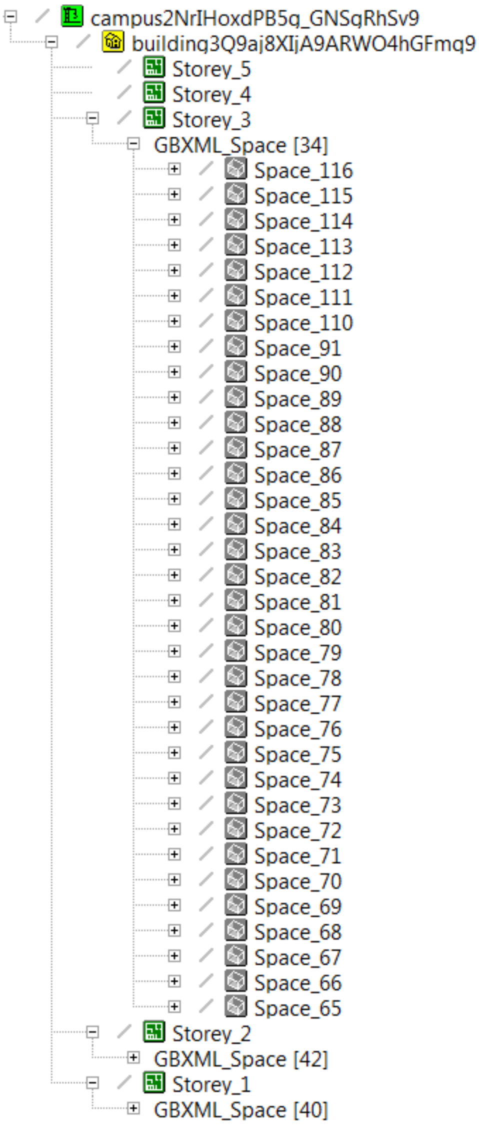


Entities	**1358**
GBXML_Building	1
GBXML_BuildingStorey	5
GBXML_Campus	1
GBXML_ExteriorWall	200
GBXML_InteriorFloor	237
GBXML_InteriorWall	773
GBXML_Project	1
GBXML_Space	116
GBXML_Window	24
Relations	**3804**
IfcRelDefinesByProperties	2569
gbxmlRelation	1210
Unknown Relation	1
Unknown Relation	24



(4.18 MB)

gbXML campus tree structure (building storey 3)



```
⊟ campus2NrIHoxdPB5q_GNSqRhSv9
  ⊟ building3Q9aj8XIjA9ARWO4hGFmq9
      Storey_5
      Storey_4
    ⊟ Storey_3
      ⊟ GBXML_Space [34]
        ⊞ Space_116
        ⊞ Space_115
        ⊞ Space_114
        ⊞ Space_113
        ⊞ Space_112
        ⊞ Space_111
        ⊞ Space_110
        ⊞ Space_91
        ⊞ Space_90
        ⊞ Space_89
        ⊞ Space_88
        ⊞ Space_87
        ⊞ Space_86
        ⊞ Space_85
        ⊞ Space_84
        ⊞ Space_83
        ⊞ Space_82
        ⊞ Space_81
        ⊞ Space_80
        ⊞ Space_79
        ⊞ Space_78
        ⊞ Space_77
        ⊞ Space_76
        ⊞ Space_75
        ⊞ Space_74
        ⊞ Space_73
        ⊞ Space_72
        ⊞ Space_71
        ⊞ Space_70
        ⊞ Space_69
        ⊞ Space_68
        ⊞ Space_67
        ⊞ Space_66
        ⊞ Space_65
    ⊟ Storey_2
      ⊞ GBXML_Space [42]
    ⊟ Storey_1
      ⊞ GBXML_Space [40]
```

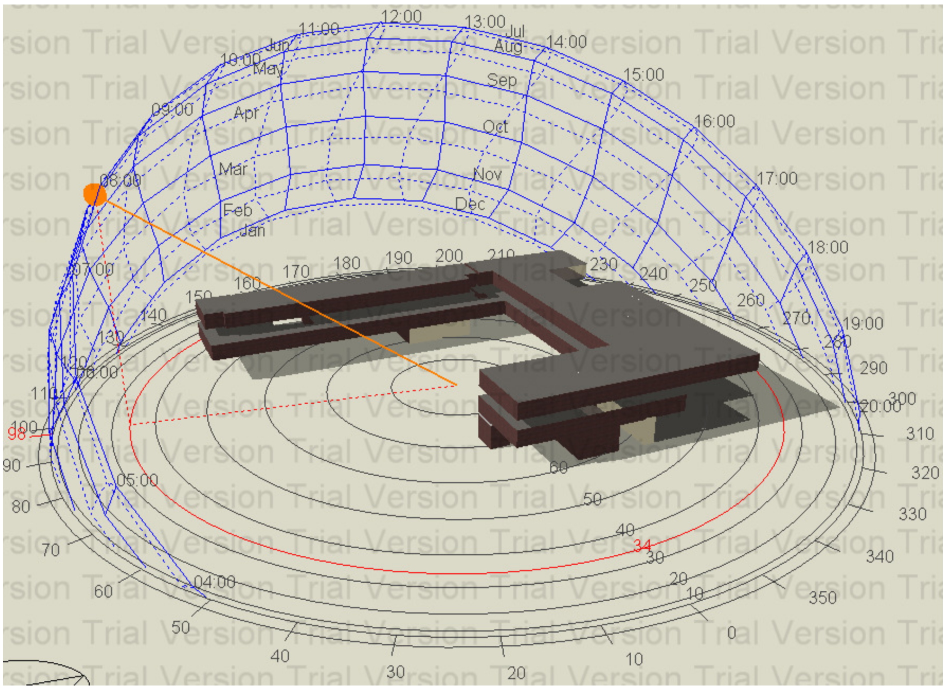
(3) Simulation in DesignBuilder:

Geometry related error message after import

List of blocks not fully enclosed with one or more gap in surfaces:

- Space_8
- Space_20
- Space_22
- Space_35
- Space_52
- Space_58

Visualization (July 15 at 15:00 hrs)



Simulation with EnergyPlus

The model cannot be modelled with EnergyPlus Evaluation licenses because it has more than 50 zones.