

Daniel Ladenhauf

From Building Information Models to Simplified Geometries for Energy Performance Analysis

Master's Thesis

Graz University of Technology

Institute of Computer Graphics and Knowledge Visualisation

Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Dieter W. Fellner

Supervisor: Dipl.-Math. Dr.techn. Torsten Ullrich

Graz, September 2014

This document is set in Palatino, compiled with pdfL^AT_EX₂ ϵ and Biber.

The L^AT_EX template from Karl Voit is based on KOMA script and can be found online:
<https://github.com/novoid/LaTeX-KOMA-template>

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Eidesstattliche Erklärung¹

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am _____

Datum

Unterschrift

¹Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Abstract

Building energy analysis has become an increasingly “hot” topic in recent years, as awareness for energy efficiency is rising and governments are setting energy performance standards for buildings as a measure to reduce greenhouse gas emissions. At the same time, the *building information modeling* paradigm is aiming to develop comprehensive digital representations of building characteristics based on semantic 3D models. Most of the data required for energy performance calculation can be found in such models. However, extracting the relevant data is not a trivial problem. Currently, this is often done manually by re-creating models from scratch.

This thesis presents an algorithm to prepare input data for energy analysis based on Industry Foundation Classes, a standard format for building information models. This is the main goal of the research project GINGER, supported by the Austrian Research Promotion Agency (FFG). The crucial aspect is geometric simplification according to semantic constraints: the building element geometries are reduced to a set of surfaces representing the thermal shell as well as the internal boundaries between spaces. These boundary parts are then associated with material layers and other thermally relevant data. A prototypical implementation shows that the algorithm works even on models that are incomplete to some degree. The presented approach, previously discussed at the International Academic Conference on Places and Technologies, can effect significant time savings compared to manual re-modeling, enabling iterative energy analysis as part of the building design process.

Zusammenfassung

Energieanalyse von Gebäuden wurde in den letzten Jahren, angesichts des steigenden Bewusstseins für Energieeffizienz sowie der Einführung von Mindeststandards als Maßnahme zur Emissionsreduktion, zu einem zunehmend „heißen“ Thema. Ein weiterer Trend im Bausektor ist *Building Information Modeling*, das Entwickeln umfassender digitaler Repräsentationen von Gebäudemerkmalen auf Basis semantischer 3D-Modelle. Die meisten der für Energieberechnungen benötigten Informationen sind in solchen Modellen enthalten. Die Überführung der relevanten Daten in ein geeignetes Format ist jedoch nicht trivial und muss derzeit häufig manuell erfolgen.

Diese Masterarbeit stellt einen Algorithmus vor, der aus Modellen im *Industry Foundation Classes*-Format (einem Standardformat für *Building Information Models*) die Daten für Energieanalysen extrahiert. Dies ist das Hauptziel des Forschungsprojekts GINGER, das von der österreichischen Forschungsförderungsgesellschaft (FFG) unterstützt wird. Der entscheidende Aspekt dabei ist die Vereinfachung der Gebäudegeometrie anhand semantischer Gruppen. Die Bauteile werden zu Oberflächen reduziert, die die thermische Gebäudehülle sowie die inneren Begrenzungen zwischen Räumen darstellen. Diese „Boundary Parts“ werden anschließend mit den Materialschichten und anderen thermisch relevanten Daten verknüpft. Ein für diese Arbeit implementierter Prototyp zeigt, dass der Algorithmus auch für bis zu einem gewissen Grad unvollständige Modelle funktioniert. Der hier beschriebene Ansatz, der auch bereits an der International Academic Conference on Places and Technologies präsentiert wurde, kann eine deutliche Zeitersparnis im Vergleich zur manuellen Neumodellierung bewirken und somit iterative Energieanalyse als Teil des Entwurfsprozesses von Gebäuden ermöglichen.

Publications

Some parts of this thesis have appeared, or have been accepted to appear, in the following peer-reviewed publications:

- [Lad+14a] D. Ladenhauf, R. Berndt, E. Eggeling, T. Ullrich, K. Battisti, and M. Gratzl-Michlmair. “From Building Information Models to Simplified Geometries for Energy Performance Simulation.” In: *Proceedings of First International Academic Conference on Places and Technologies*. University of Belgrade – Faculty of Architecture. 2014, pp. 669–676.
- [Lad+14b] D. Ladenhauf, R. Berndt, U. Krispel, E. Eggeling, T. Ullrich, K. Battisti, and M. Gratzl-Michlmair. “Geometry Simplification According to Semantic Constraints: Enabling Energy Analysis Based on Building Information Models.” In: *Computer Science - Research and Development* (2014), accepted to appear.

Acknowledgements

I would like to express my gratitude to Torsten Ullrich for being such a supportive and helpful supervisor. I would also like to thank the staff at Fraunhofer Austria, especially Eva Eggeling and René Berndt from the GINGER project team, as well as Kurt Battisti and Markus Gratzl-Michlmair from our project partners, for their great cooperation. Furthermore, I owe thanks to my friends and my family, especially my parents, for all their support and encouragements throughout my studies. Kathi, thank you for being so awesome.

Contents

Abstract	iv
Zusammenfassung	v
Publications	vi
Acknowledgements	vii
1 Introduction	1
2 Theoretical Foundations	4
2.1 From Computer-Aided Design to Open Building Information Modeling	4
2.1.1 Building Information Modeling (BIM)	5
2.1.2 Industry Foundation Classes (IFC)	7
2.2 Building Energy Performance	15
2.2.1 Legal Requirements in the European Union and Austria	15
2.2.2 Calculation of Thermal Energy Performance	17
3 Related Work	19
3.1 Building Information vs. Energy Modeling	19
3.2 Berkeley Lab Approach	21
3.2.1 Space Boundaries	21
3.2.2 Semi-Automated BIM to BEM Process	23
3.3 Numerical Robustness in Computational Geometry	25
4 Algorithm	28
4.1 Reference Models	29
4.2 Input and Data Structures	31
4.2.1 Input Requirements	33
4.2.2 Geometric Representation	34
4.2.3 Boundary Parts	35

Contents

4.3	Geometric Simplification	36
4.3.1	Detecting the Boundary per Space	37
4.3.2	Boundary Parts between Spaces	39
4.3.3	Filling Gaps in the Boundary	41
4.3.4	Boundaries between Building Stories	41
4.4	Export	44
4.4.1	Processing Additional Data	44
4.4.2	ArchiPHYSIK File	45
5	Implementation and Results	47
5.1	Key Aspects of Implementation	47
5.1.1	BIMserver	47
5.1.2	JTS Topology Suite	49
5.1.3	Guava Library	50
5.1.4	3D Geometry Output	50
5.2	Results	51
5.2.1	Performance	54
5.2.2	Problem Cases	54
6	Discussion	58
	Bibliography	60

List of Figures

1.1	An IFC model side by side with the respective Sketchup model	3
2.1	Example of an IFC entity specification in EXPRESS	8
2.2	Example of an IFC inheritance graph in EXPRESS	9
2.3	Excerpt of an IFC STEP file	10
2.4	Screenshot of an IFC model	11
2.5	Screenshot from Solibri Model Checker	13
2.6	Poor 3D model example	14
2.7	Energy performance certificate	16
3.1	Space boundaries	22
3.2	Screenshot from Solibri Model Checker (Berkeley Lab rule set)	24
3.3	Building graph and corresponding geometry	26
3.4	Representable planes for with low coefficient resolution	27
4.1	Documentation of the output requirements	30
4.2	Reference model 1: simple bungalow	30
4.3	Reference model 2: simple two-story home	31
4.4	Reference model 3: twin home	32
4.5	Reference model 4: office building	32
4.6	Reference model 5: high-rise office building	33
4.7	Union of triangulated geometry	34
4.8	2D coordinates	35
4.9	Boundary part class diagram	36
4.10	Footprint after first step of finding boundary parts	38
4.11	The result of the first step as a 3D rendering	38
4.12	Footprint after merge interior boundary parts	40
4.13	The three steps of extracting the boundary	42
4.14	Footprint with final boundary parts	42
4.15	Story detection	43

List of Figures

4.16	Example of an ArchiPHYSIK XML file.	46
5.1	The system architecture	48
5.2	A JSON example	49
5.3	A Wavefront OBJ example	50
5.4	Comparison of input and output geometries for Model 2	51
5.5	The intermediate results of each of the three processing steps	52
5.6	Comparison of input and output geometries for Model 5	53
5.7	Spaces for elevator shafts	55
5.8	Problem B: exterior spaces marked as interior	56
5.9	Problem C: multiple building elements	56
5.10	Problem D: corner windows	56

1 Introduction

In view of human-induced climate change and increasing awareness for energy conservation and efficiency, demands regarding building energy performance are rising. The building sector accounts for 40% of the energy consumed in Europe [Con14]. Therefore, the European Union has set the *Directive on the energy performance of buildings*, which requires member states to establish minimum requirements for new and renovated buildings. Moreover, it demands that energy performance certificates are issued for all buildings and residential units on the market, to inform prospective buyers and tenants about the expected space heating demand, primary energy demand, and CO₂ emissions [ET10]. Energy performance analysis is applied to prepare such certificates, but also to ensure that architectural designs meet the requirements. The input data for these calculations include climate data for the location of the building, the thermal characteristics of the used materials, as well as the building geometry [PSM10].

Another trend in the architecture, engineering and construction industry is *building information modeling (BIM)*. This term refers to the idea of storing all kinds of physical and functional characteristics of a building in a shared digital model based on a 3D representation. The key aspect that differentiates BIM from *computer-aided design (CAD)* is the semantics: walls, windows, spaces, etc., can be distinguished from each other, and all have their own characteristics. This opens the door for a wide range of use cases for the model, across different disciplines, and throughout the life-cycle of a building. Thus, interoperability is a critical factor for this paradigm. The industry alliance *buildingSMART* promotes interoperability between different BIM systems with the *Industry Foundation Classes (IFC)*, an open data model that has been widely adopted by the industry in recent years.

It seems obvious to build a bridge between these two trends, and make it possible to take advantage of IFC models for energy performance analysis. This is the goal of the research project GINGER (Graphical Energy Efficiency Visualization in Architecture), which this thesis has emerged from. Supported by the Austrian Research Promotion

1 Introduction

Agency (FFG), it is undertaken by Fraunhofer Austria Research GmbH in cooperation with A-NULL Bauphysik GmbH and Ingenieurbüro Gratzl e.U.

Currently, a common approach is to start from scratch when preparing the input data for an energy calculation. In order to automate this time-consuming process, and to give relevant feedback during the design phase, a solution for extracting only the relevant data and a simplified geometry is needed. Creating a completely new file by hand when almost all the information is available in a comprehensive digital model is cumbersome. This sounds almost like printing a document and re-typing parts of it on the computer, when you actually have the document stored as a digital file on the same computer and might as well use *copy and paste*. To be fair, 3D models are a bit more complex than plain text, and even though they cannot simply be copied and pasted between different programs, import and export of common 3D file formats is often possible.

However, it does not end there: the geometry needed for energy performance calculation is different from the one in a building information model. On the one hand, it is much simpler, with single planar surfaces instead of volumetric building element geometries. On the other hand, it has a stronger emphasis on the relationships between spaces: each surface represents a boundary between exactly two spaces, or a space and the outside air. In Figure 1.1, an IFC model is compared with a *Sketchup* model of the same building, as it could be used for the import into an energy analysis tool. The IFC model contains the detailed, volumetric geometries of all building elements as well as a lot of non-geometric data. With a file size of twelve megabytes, it is about 80 times as large as the *Sketchup* file (150 KB). There, some parts are completely left out, while others are greatly simplified.

Nevertheless, aiming for an at least semi-automatic solution seems both necessary and feasible. Especially for larger buildings, the manual way is very time-consuming, which hinders iterative energy analysis embedded in the building design process. The semantic data stored in an IFC model should enable an algorithm to find the boundary surfaces and place them according to specific rules. These are based on norms and regulations, such as the guideline for energy savings and heat insulation in Austria [Aus11].

In the next chapter, some background about the two topics that this thesis aims to link is given. Building information modeling is explained in comparison with CAD, with a focus on semantic and parametric modeling as well as IFC. An overview about building energy performance includes relevant legal requirements in the

1 Introduction

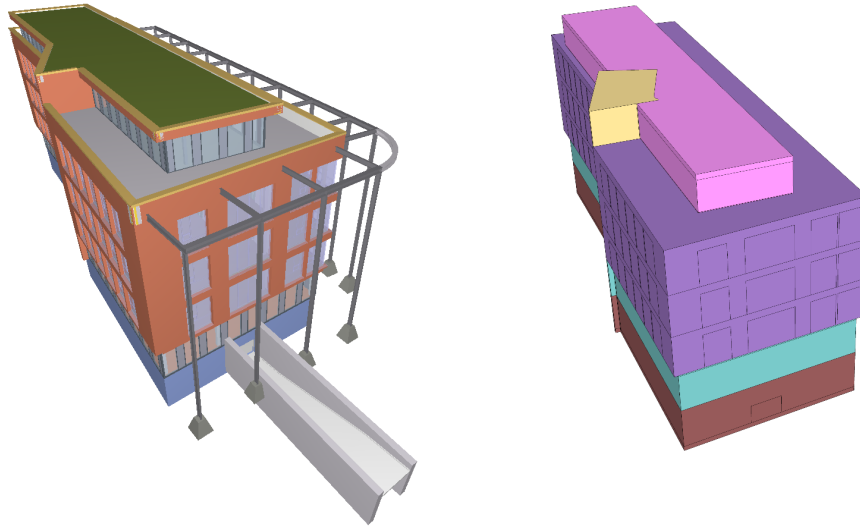


Figure 1.1: An IFC building information model (left) side by side with a *Sketchup* model of the same building (right) as used for the import into an energy analysis tool. The IFC model contains the detailed, volumetric geometries of all building elements. In the *Sketchup* model, some parts are completely left out, while others are greatly simplified.

European Union and Austria, as well as some basic information about energy calculation methods. Related work is discussed in Chapter 3. A closer look is taken at an approach by Berkeley Lab researchers, who create energy models based on building information models. Computational geometry, and the common numerical robustness issues related with it, are also addressed. Chapter 4 presents the algorithm for extracting simple building models for energy analysis from complex models. After explaining the data structure for the geometric representation and how it is associated with semantic data, the geometric simplification algorithm is broken down in detail. In Chapter 5, some key aspects of the prototype implementation, such as the used libraries, are described, followed by an overview about results and challenging cases. A concluding chapter reflects on the thesis and gives an outlook on future work.

2 Theoretical Foundations

2.1	From Computer-Aided Design to Open Building Information Modeling	4
2.1.1	Building Information Modeling (BIM)	5
2.1.2	Industry Foundation Classes (IFC)	7
2.2	Building Energy Performance	15
2.2.1	Legal Requirements in the European Union and Austria	15
2.2.2	Calculation of Thermal Energy Performance	17

The algorithm presented in this thesis is based on building information modeling (BIM), and extracts data for building energy performance analysis. To provide the necessary background for the following sections, these concepts are explained here.

2.1 From Computer-Aided Design to Open Building Information Modeling

Computer-aided design, or CAD, is “the use of computer programs and systems to design detailed two- or three-dimensional models of physical objects, such as mechanical parts, buildings, and molecules [Caro3].” Its origins go back to the 1960s, long before personal computers (PCs) and modern graphical user interfaces (GUIs) came up. The *Sketchpad* system by Ivan Sutherland from 1963 is considered the ancestor of modern CAD systems. It used a screen and a light pen connected to a computer. Aerospace and automotive companies were the first to adopt CAD systems. In the 1980s, the first CAD programs that ran on PCs were developed, most notably *AutoCAD* by *Autodesk*.

In essence, basic 2D CAD approaches are similar to drawing with pencil and paper. Floor plans, side views and cuts are drawn independently. 2½D CAD refers to the capability of drawing lines in the space, resulting in a wire-frame model, which

2 Theoretical Foundations

does not define volumes and faces. This can be achieved with 3D CAD systems, which can create virtual models of objects, describing their full geometric characteristics [Hen+10]. An advantage of 2D CAD is the better long-time archival capability: 2D drawings can be stored on microfilm, while 3D models need to be archived digitally. This makes them subject to digital obsolescence, and preservation strategies like ongoing format migration need to be employed to overcome it [Neu+10]. Another possibility is to extract and archive 2D views from 3D models.

For data exchange between different CAD systems, *Autodesk's Drawing Interchange File Format (DXF)* has emerged as an industry standard. It is important to note that this format is used for geometry data only, without support for the semantics of a building model. For example, a wall is exchanged as a generic building block, not as a wall object with additional attributes like materials, thermal transmittance, or even just the information that it is a wall [BLL04].

It is hard to get reliable information about the CAD market for free, but there is anecdotal evidence that 2D CAD is still common among architects today. AutoCAD remains the most popular CAD software among architects, but there is a trend towards full-blown 3D CAD and BIM [Arc11].

2.1.1 Building Information Modeling (BIM)

Building information modeling (BIM) is a commonly used buzzword in today's building industry. As with many buzzwords, it is not always clear what concept is associated with the term, as a review of literature and marketing material shows.

The American *National Building Information Model Standard (NIBS)* project committee defines BIM as "a digital representation of physical and functional characteristics of a facility. A BIM is a shared knowledge resource for information about a facility forming a reliable basis for decisions during its life-cycle; defined as existing from earliest conception to demolition [Nat14]."

Other definitions are summarized in a literature review by Abbasnejad and Moud, who conclude that a generally accepted comprehensive definition of BIM has not been established yet, and different stakeholders (architects, builders, owners, etc.) have mixed expectations towards BIM [AM13].

2 Theoretical Foundations

Most definitions agree with the cited one from *NIBS* in what the main difference to CAD, even 3D-CAD, is: a building information model does not just store the geometry of a building, but includes semantic data about the functions of the buildings and its elements. Furthermore, BIM is intended to be used throughout the building's life-cycle, containing information for planning, design, construction, operation, and maintenance. That is, a model is not only used by architects, contractors, and suppliers, but by all kinds other users, e.g., government agencies, owners, real estate agents, facility managers, etc.

In this context, sometimes the terms 4D-, 5D-, and 6D-BIM (and occasionally even higher numbers) are used:

- 4D adds the time dimension to the 3D space of a model, describing construction schedules for building components.
- 5D puts cost information on top of that, enabling estimations about material and labor cost (quantity take-off).
- 6D refers to facility management data added to a model, enabling life-cycle management based on a BIM, possibly supported by sensor-captured data [AM13].

Eastman et al. help to understand BIM by describing examples that are *not* BIM technology. As already mentioned, models without object attributes, but only 3D data, are not considered BIM. Also, models composed of multiple 2D drawings that have to be combined, or models that do not automatically reflect changes made in one view in other views, are not building information models. Moreover, Eastman et al. consider parametric object capabilities as essential for BIM. Parametric objects in BIM can include rules to automatically modify associated objects (e.g., a wall is changed when a door is placed in it), and for ensuring feasibility (e.g., regarding size and manufacturability) [Eas+11]. Such intelligent objects are similar to the idea of generative modeling, where a 3D object is described by the operations necessary to generate the object, rather than the result of these operations [KSU14].

Leading BIM authoring tools are *ArchiCAD* by *Graphisoft* [Gra14a], *Revit* by *Autodesk* [Aut14b], and *Microstation* with its BIM extension by *Bentley Systems* [Ben14].

By the very nature of BIM as a collaborative process between many different parties, data exchange is a crucial factor.

2.1.2 Industry Foundation Classes (IFC)

To facilitate data exchange between different players in the building industry, which is one of the main ideas of BIM, it is necessary to make BIM systems by different vendors interoperable. The industry alliance *buildingSMART*, formerly *International Alliance for Interoperability*, is tackling this challenge. It is developing an open data model for building and construction industry data called Industry Foundation Classes (IFC), which has become an ISO standard (ISO 16739).

IFC is built upon the *Standard for the Exchange of Product model data (STEP)*, which is a comprehensive ISO standard for the exchange of both geometric and a broad range of other product-related data for many different product types (electronic, mechanical, automotive, ship, etc.). STEP has been adopted by the industry as a neutral format for exchanging data between different CAD systems or CAD and other applications. It includes an information modeling language called EXPRESS, which is used to define the entities and their relationships. The IFC schema is also specified with EXPRESS, as the example code in Figure 2.1 shows. The actual data exchange is decoupled from the information model. The most common way is to use plain text files as specified in the standard, so-called STEP files [Pra01]. IFC (.ifc) files follow the STEP syntax. Excerpts from an IFC file with its simple and concise syntax are shown in Figure 2.3. IFC also supports XML files (.ifcxml) and zipped STEP files (.ifczip). The currently most widely implemented version is IFC2x3 TC1; the specification for the newest version IFC4 has been released in 2013 [Lie+12].

Model Structure

IFC consists of entities that have attributes as well as relationships with each other. Entities are the types of objects, relations (i.e. objectified relationships), or properties. Object entities can be physically tangible items (e.g. wall, window), other physically existing items (e.g. space) or conceptual items (e.g. processes, work tasks). Relation entities are used to uncouple relationship semantics from the objects. Property entities allow the assignment of characteristics (that are not described by object attributes) via relationships. Entities are structured in an inheritance hierarchy (i.e. entities are subtypes/supertypes of other entities) [Lie+12]. The excerpt from the IFC specification in Figure 2.1 shows parts of the definition of the abstract object *IfcSpatialStructureElement* as well as the relation *IfcRelContainedInSpatialStructure*. The

2 Theoretical Foundations

```
ENTITY IfcSpatialStructureElement
  ABSTRACT SUPERTYPE OF (ONEOF(IfcBuilding, IfcBuildingStorey,
    IfcSpace, IfcSite))
  SUBTYPE OF (IfcProduct);
  LongName          : OPTIONAL IfcLabel;
  CompositionType  : IfcElementCompositionEnum;
  INVERSE
  ReferencesElements : SET OF IfcRelReferencedInSpatialStructure
    FOR RelatingStructure;
  ServicedBySystems : SET OF IfcRelServicesBuildings
    FOR RelatedBuildings;
  ContainsElements  : SET OF IfcRelContainedInSpatialStructure
    FOR RelatingStructure;
  ...
END_ENTITY;

ENTITY IfcRelContainedInSpatialStructure
  SUBTYPE OF (IfcRelConnects);
  RelatedElements   : SET [1:?] OF IfcProduct;
  RelatingStructure : IfcSpatialStructureElement;
  ...
END_ENTITY;
```

Figure 2.1: Example of an IFC entity specification in EXPRESS, showing parts of the definition of the abstract object *IfcSpatialStructureElement* as well as the relation *IfcRelContainedInSpatialStructure*. All blue underlined words are entities. Left of the colons are the attribute names. The keyword *inverse* marks relations that link to the defined entity, e.g., with *ContainsElements* one can access all *IfcRelContainedInSpatialStructure* occurrences that link back to this *IfcSpatialStructureElement*.

2 Theoretical Foundations

```
ENTITY IfcWindow;
ENTITY IfcRoot;
  GlobalId      : IfcGloballyUniqueId;
  OwnerHistory : OPTIONAL IfcOwnerHistory;
  Name          : OPTIONAL IfcLabel;
  Description   : OPTIONAL IfcText;
  ...
ENTITY IfcObject;
  ...
INVERSE
  ...
  IsDefinedBy : SET OF IfcRelDefinesByProperties
                FOR RelatedObjects;

ENTITY IfcProduct;
  ObjectPlacement : OPTIONAL IfcObjectPlacement;
  Representation  : OPTIONAL IfcProductRepresentation;
  ...
ENTITY IfcBuildingElement;
  ...
ENTITY IfcWindow;
  OverallHeight : OPTIONAL IfcPositiveLengthMeasure;
  OverallWidth  : OPTIONAL IfcPositiveLengthMeasure;
  PredefinedType : OPTIONAL IfcWindowTypeEnum;
  PartitioningType : OPTIONAL IfcWindowTypePartitioningEnum;
END ENTITY;
```

Figure 2.2: Example of an IFC inheritance graph in EXPRESS, showing how *IfcWindow* is derived from *IfcBuildingElement*, *IfcProduct*, etc.

keyword *inverse* marks relations that link to the defined entity, e.g., with *Contains-Elements* one can access all *IfcRelContainedInSpatialStructure* occurrences that link back to this *IfcSpatialStructureElement*. The inheritance graph in Figure 2.2 highlights how entities are defined with supertypes, showing how *IfcWindow* is derived from *IfcBuildingElement*, *IfcProduct*, etc.

Building elements can be associated with spatial structure elements (site, building, story, space) to define the logical makeup of a model. The screenshot in Figure 2.4 shows the model tree and a 3D visualization of the selected story in *Solibri Model Viewer*. For the geometric representation, various types can be used, e.g., constructive solid geometry (CSG), sweeping solids (e.g., extrusion), or boundary representation (B-rep). In Figure 2.3, one can see how the spatial structure for a model is built with *IfcRelAggregates* relations. An *IfcWallStandardCase* is linked to it with the *IfcRelContainedInSpatialStructure* relation. An *IfcPropertySet* is used to add a property to it. The shape representation for the wall is defined with an *IfcExtrudedAreaSolid* that has its profile defined with an *IfcPolyline* between four *IfcCartesianPoints*.

2 Theoretical Foundations

```

#56 = IFCPROJECT('344...jSU', #28, 'Büro+Verwaltungsgebäude', $,
$, $, 'Bauantrag', (#53, #12374), #42);
#81 = IFCRELAGGREGATES('0Du...SAT', #28, $, $, #56, (#75));
#75 = IFCSITE('20F...uIce', #28, 'Gelände', $, $, #72, $, $,
.ELEMENT., (48, 6, 50, 862729), (11, 27, 38, 146037),
433., $, #63);
#112 = IFCRELAGGREGATES('2b...mbt', #28, $, $, #75, (#110));
#110 = IFCBUILDING('00t...N2A', #28, 'Büro+Verwaltungsgebäude',
$, $, #108, $, $, .ELEMENT., $, $, #99);
#134 = IFCRELAGGREGATES('118...fU5', #28, $, $, #110, (#132,
#20711, #77684, #168715, #252095, #327327, #372753));
#77684 = IFCBUILDINGSTOREY('2u8...knJ', #28, '1. OG', $, $, #77683,
$, $, .ELEMENT., 4.);
#137732 = IFCRELAGGREGATES('0d$...$p6', #28, $, $, #77684, (#137729,
#137801, #137870, ...));
#137729 = IFCSPACE('3zk...d5R', #28, '8', $, $, #137698, #137726,
'Schacht', .ELEMENT., .INTERNAL., $);
#77771 = IFCRELCONTAINEDINSPATIALSTRUCTURE('2Zq...3xT', #28, $, $,
(..., #80284, #81886, #82882, ...), #77684);
#80284 = IFCWALLSTANDARDCASE('2vr...DdD', #28, 'Wand-002', $, $,
#80235, #80280, 'B9D...9CD');
#80906 = IFCRELDEFINESBYPROPERTIES('0eT...D4x', #28, $, $,
(#80284), #80904);
#80904 = IFCPROPERTYSET('3id...wlj', #28,
'AC_Pset_RenovationAndPhasing', $, (#80903));
#80903 = IFCPROPERTYSINGLEVALUE('Renovation Status', $,
IFCLABEL('Existing'), $);
#80280 = IFCPRODUCTDEFINITIONSHAPE($, $, (#80266, #80277));
#80266 = IFCSHAPE REPRESENTATION(#157, 'Body', 'SweptSolid',
(#80256));
#80256 = IFCEXTRUDEDAREASOLID(#80246, #80253, #80254, 3.8);
#80246 = IFCARBITRARYCLOSEDPROFILEDEF(.AREA., 'GK - 2-lagig
beplankt (150 x 42800)', #80244);
#80244 = IFCPOLYLINE((#80236, #80238, #80240, #80242, #80236));
#80236 = IFCCARTESIANPOINT((0., 0.));
#80238 = IFCCARTESIANPOINT((42.8, 0.));
#80240 = IFCCARTESIANPOINT((42.8, 0.15));
#80242 = IFCCARTESIANPOINT((0., 0.15));

```

Figure 2.3: Excerpt of an IFC STEP file, showing how the spatial structure is defined with *IfcSite*, *IfcBuilding*, *IfcBuildingStorey* and *IfcSpace* objects linked with *IfcRelAggregates* relations. An *IfcWallStandardCase* is linked to it with the *IfcRelContainedInSpatialStructure* relation. A *IfcPropertySet* is used to add a property to it. The shape representation for the wall is defined with an *IfcExtrudedAreaSolid* that has its profile defined with an *IfcPolyline* between four *IfcCartesianPoints*.

2 Theoretical Foundations

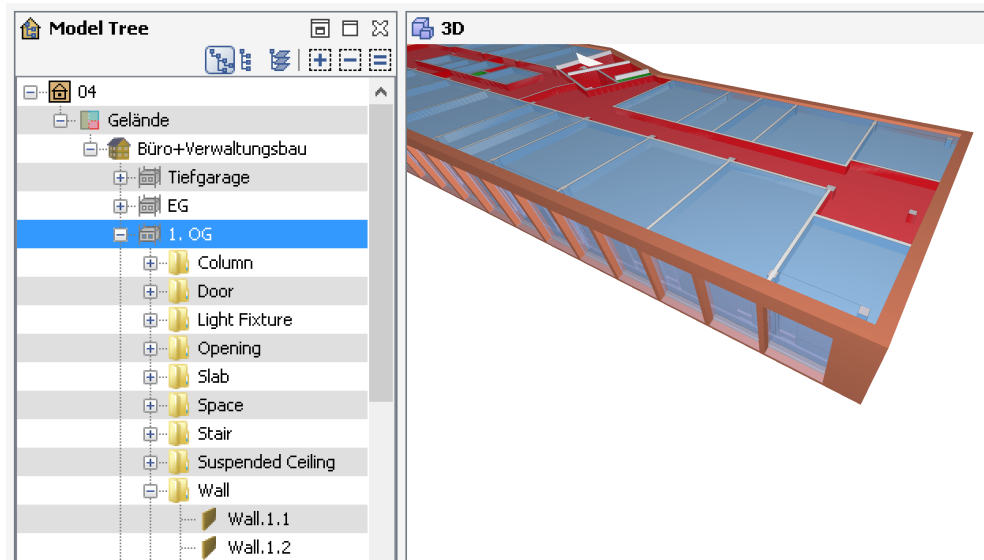


Figure 2.4: Screenshot of an IFC model in *Solibri Model Viewer* with the model tree (Project - Site - Building - Building Story - ...) and a 3D visualization of the selected building story.

Adoption and Data Quality

IFC has been widely adopted in the building industry in recent years. Most BIM authoring tools and architecture CAD tools are able to import and export IFC models. Some countries, like Norway, Denmark and Finland, promote the use of BIM and IFC with specific guidelines, some even requiring IFC models in the procurement of public projects [WVN09].

However, data quality is a big issue, as practical experience during the development of the algorithm for this thesis and a glance at the literature reveal. Poor data quality occurs because of different reasons, and has to be tackled accordingly [Shao4].

One reason can be incomplete or incorrect implementations of IFC by the exporting tools. This is not surprising when one considers the enormous scope of IFC with its 653 entities [Lie+12]. Model View Definitions (MVDs) are a response to the problem that one usually does not know what to expect from an IFC model: IFC, by its nature, is rich, but flexible (e.g., many optional attributes and relationships) and redundant, because it has to address various needs from architects, suppliers, engineers, and others. MVDs define subsets of the IFC schema for certain exchange

2 Theoretical Foundations

tasks, outlining what data are expected in a specific use case [Eas+11]. An MVD can be considered a software requirements specification for IFC-supporting tools. Official MVDs are being developed by buildingSMART, with the Coordination View being the first and most widely implemented view [bui14a]. BuildingSMART has set up a certification process to ensure that software vendors implement IFC according to the standard, based on MVDs. For the Coordination View, many software products have been certified for import and/or export already, including ArchiCAD and Revit [bui14b].

Another source of poor data quality, probably even harder to eliminate, is how BIM authoring tools are being used. Many users in architecture may not use BIM software in its intended way: they just use it for drafting and designing buildings, so that they *look* the way they should. An extreme example to illustrate this point is the 3D model shown in Figure 2.6, containing a 2D image of a tree, which looks fine from a specific angle. Without having the purpose of building information modeling and all the possible use cases in mind during creation, it is very likely to end up with models that do not *behave* the way they should. An obvious approach to improve such situations is to make users better understand BIM and IFC with tutorials and training. Moreover, there are quality assurance tools that can be applied, like the *Solibri Model Checker* [Sol14]. It can check whether a model follows sets of formally defined rules, e.g., that certain attributes have values assigned, every room has an opening, there are no clashes or gaps between building elements, the spatial structure of the model is well-defined and corresponds to the geometry, etc. Figure 2.5 shows an example.

2 Theoretical Foundations

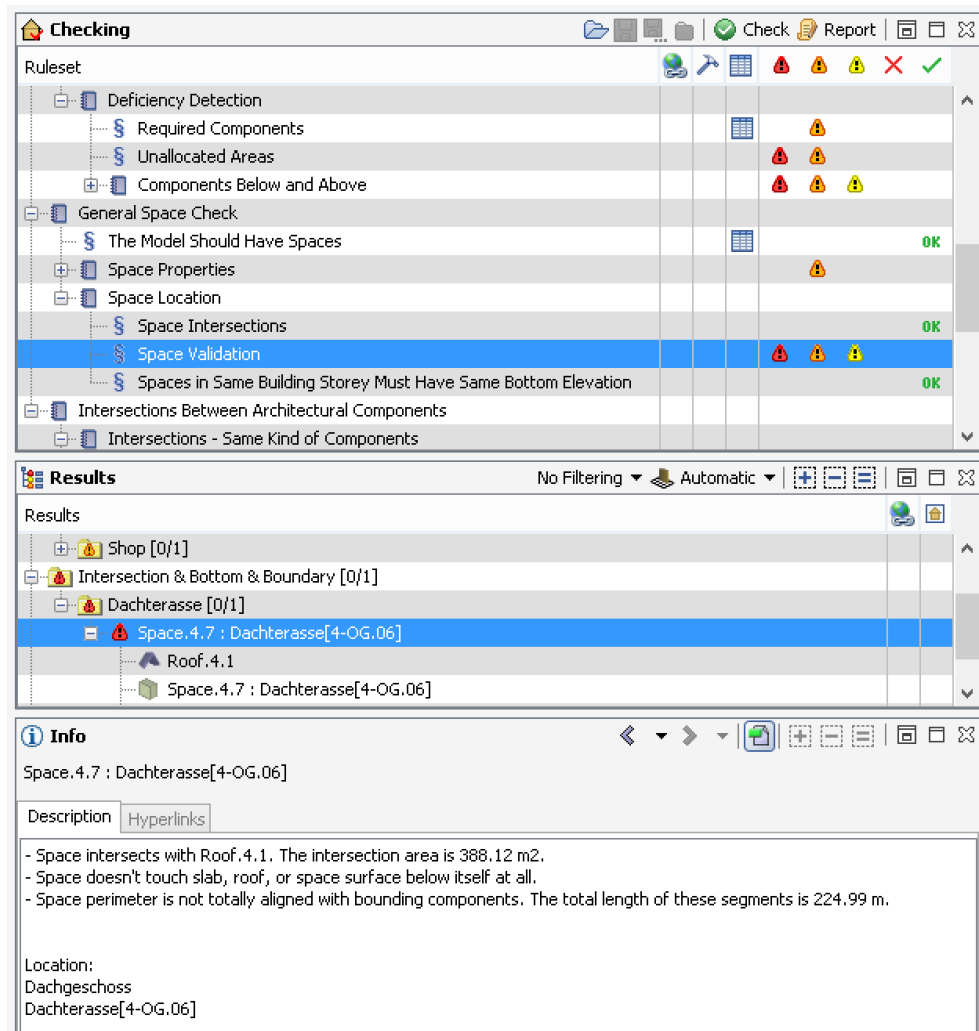


Figure 2.5: Screenshot from Solibri Model Checker showing some results of a check against standard rule sets. Selected is the result of a space check: it was detected that one of the spaces does not touch the slab below it, and intersects the roof above it.

2 Theoretical Foundations

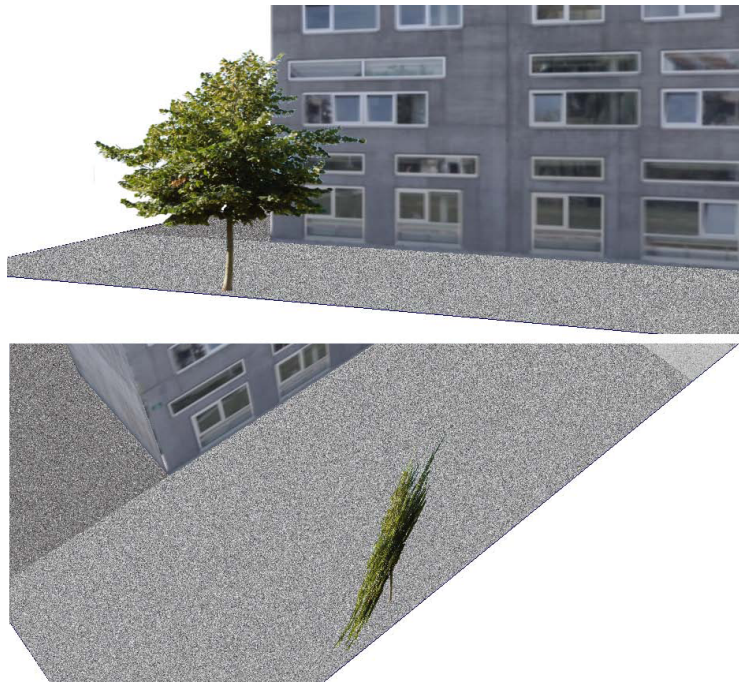


Figure 2.6: Example of a 3D model containing a 2D image of a tree, only looking good from a specific angle [Set13].

2.2 Building Energy Performance

Buildings account for 40% of the total energy consumed in Europe [Con14]. In the light of climate change and the call for a reduction of greenhouse gas emissions, energy performance of buildings is an increasingly relevant topic.

2.2.1 Legal Requirements in the European Union and Austria

The European Union has undertaken many initiatives to reduce energy consumption in buildings. Most notably, it has set the *Directive on the energy performance of buildings (EPBD)* (first passed in 2002, recast in 2010) [ET10]. This directive requires all member states to put in place laws that implement certain measures to promote more energy-efficient buildings.

These measures include the establishment of minimum requirements for new buildings and ones that undergo major renovation, with regards to a cost-optimal level. This is “the energy performance level which leads to the lowest cost during the estimated economic lifecycle”.

To determine the energy performance of a building, the directive calls for calculation methodologies which take into account the thermal characteristics (such as thermal capacity and insulation), heating insulation and hot water supply, air-conditioning, lighting, as well as the indoor climate requirements. Based on the defined methodology, each member state has to have a system for energy performance certificates (EPCs), which shall be shown to prospective buyers and tenants, and also shall be included in respective advertisements. Together with financial incentives to promote improvements in the energy performance of buildings, this aims to decrease the total energy use in buildings. Figure 2.7 shows an example of an Austrian *Energieausweis* according to the current law. It includes a rating from A++ to G for space heating demand, primary energy demand, CO₂ emissions, and an overall energy performance factor [Aus11].

A report by the European Commission from 2013 about the impact of energy performance certificates on transaction prices and rents concludes that certification is effective and the market rewards energy efficiency improvements. For example, the report estimates that in Vienna and Lower Austria, a one-letter improvement has an effect of 8% for sales prices and 4.4% for rents. However, it also concludes that the EPCs have not reached their full potential yet [BLI13]. While the mentioned

2 Theoretical Foundations

Energieausweis für Wohngebäude

OIB-Richtlinie 6
Ausgabe: Oktober 2011

Logo

BEZEICHNUNG

Gebäude(-teil)	Baujahr
Nutzungsprofil	Letzte Veränderung
Straße	Katastralgemeinde
PLZ/Ort	KG-Nr.
Grundstücksnr.	Seehöhe

SPEZIFISCHER HEIZWÄRMEBEDARF, PRIMÄRENERGIEBEDARF, KOHLENDIOXIDEMISSIONEN UND GESAMTENERGIEEFFIZIENZ-FAKTOR (STANDORTKLIMA)

	HWB _{SK}	PEB _{SK}	CO ₂ SK	f _{GEE}
A ++				
A +				
A	A (Beispiel)		A+ (Beispiel)	A (Beispiel)
B		B (Beispiel)		
C				
D				
E				
F				
G				

HWB: Der Heizwärmebedarf beschreibt jene Wärmemenge, welche den Räumen rechnerisch zur Beheizung zugeführt werden muss.

WWB: Der Warmwasserwärmebedarf ist als flächenbezogener Defaultwert festgelegt. Er entspricht ca. einem Liter Wasser je Quadratmeter Brutto-Grundfläche, welcher um ca. 30 °C (also beispielsweise von 8 °C auf 38 °C) erwärmt wird.

HEB: Beim Heizenergiebedarf werden zusätzlich zum Nutzenergiebedarf die Verluste der Haustechnik im Gebäude berücksichtigt. Dazu zählen beispielsweise die Verluste des Heizkessels, der Energiebedarf von Umwälzpumpen etc.

HHSB: Der Haushaltsstrombedarf ist als flächenbezogener Defaultwert festgelegt. Er entspricht ca. dem durchschnittlichen flächenbezogenen Stromverbrauch in einem durchschnittlichen österreichischen Haushalt.

EEB: Beim Endenergiebedarf wird zusätzlich zum Heizenergiebedarf der Haushaltsstrombedarf berücksichtigt. Der Endenergiebedarf entspricht jener Energiemenge, die eingekauft werden muss.

PEB: Der Primärenergiebedarf schließt die gesamte Energie für den Bedarf im Gebäude einschließlich aller Vorketten mit ein. Dieser weist einen erneuerbaren und einen nicht erneuerbaren Anteil auf. Der Ermittlungszeitraum für die Konversionsfaktoren ist 2004 – 2008.

CO₂: Gesamte dem Endenergiebedarf zuzurechnenden Kohlendioxidemissionen, einschließlich jener für Transport und Erzeugung sowie aller Verluste. Zu deren Berechnung wurden übliche Allokationsregeln unterstellt.

f_{GEE}: Der Gesamtenergieeffizienz-Faktor ist der Quotient aus dem Endenergiebedarf und einem Referenz-Endenergiebedarf (Anforderung 2007).

Alle Werte gelten unter der Annahme eines normierten BenutzerInnenverhaltens. Sie geben den Jahresbedarf pro Quadratmeter beheizter Brutto-Grundfläche an.

Dieser Energieausweis entspricht den Vorgaben der Richtlinie 6 „Energieeinsparung und Wärmeschutz“ des Österreichischen Instituts für Bautechnik in Umsetzung der Richtlinie 2010/31/EU über die Gesamtenergieeffizienz von Gebäuden und des Energieausweis-Vorlage-Gesetzes (EAUG).

Figure 2.7: Exemplary Austrian energy performance certificate (*Energieausweis*) for residential buildings. It includes a rating from A++ to G for space heating demand, primary energy demand, CO₂ emissions, and an overall energy performance factor [Aus11].

report states that only 20% of property sales in Austria were accompanied with an EPC, a new revision of the Austrian law in force since December 2012 now makes it obligatory for property advertisements to feature EPCs, and introduces fines in case of infringement [Con13].

2.2.2 Calculation of Thermal Energy Performance

To calculate the thermal energy performance of buildings, two main methodologies can be used: stationary calculation and dynamic simulation. The first is a relatively simple technique that uses monthly or yearly climate averages, and requires little input data. The second is a more complex, time-resolved simulation with much higher input requirements, not only for climate data, but also related to the building geometry [pas13].

The Austrian calculation methodology for energy performance certificates, as defined in the *OIB-Richtlinie Energieeinsparung und Wärmeschutz* and the *ÖNORM B 8110*, uses a stationary model. The goal is to keep the effort for certificate issuance to a justifiable amount, yet provide “good enough” results, e.g. to enable meaningful comparisons of different buildings on the market. The input data necessary for such a calculation includes:

- building geometry data, namely the building elements’ area quantities, gross area, and gross volume
- the building elements’ thermal transmittance (U-values), or definition of their material layers (material, thickness)
- heating-degree-days (*Heizgradtage*) per year (determine the period in which the space heating is used)
- monthly climate data (average temperature, solar radiation for horizontal areas and all cardinal directions) [PSM10]

The German *Passive House Institute* argues that, even though simulation is the correct scientific approach, it is often not practical. For established building concepts, comparisons have shown that a simplified stationary approach gives very similar results. Furthermore, the simpler method has the advantage that it is less error-prone, as the best simulation is only as accurate as its input data. In practice, due to the high number of required values, inappropriate values could be entered for seemingly unimportant data [pas13].

2 Theoretical Foundations

Pont, Sommer and Mahdavi compare the results of the stationary model used for Austrian EPCs with the ones of a dynamic simulation for seven existing buildings in Vienna. Their study shows that the dynamic simulation gives significantly lower values than the EPC calculation, but the results of the two methods maintain a relatively constant relation [PSM10].

Gratzl-Michlmair, Graf and Goerth compare the Austrian and German EPC norms in terms of both results and effort. The basic approach in the two countries is very similar: a monthly balance is used, with usage profiles, local climate data, and building data as inputs. A notable difference is the zoning of a building, which is more detailed in Germany (generally space-wise, only spaces with identical usage and conditioning can be combined) than in Austria (broader usage profiles, often only one zone per building). Thus, the effort for issuing an EPC is much higher in Germany. However, the study finds that the differences in the results are negligible in many cases [GGG12].

A commonly used software tool for EPC calculations in Austria is *ArchiPHYSIK* by *A-NULL* [ANU14]. An example for the German market is *DIN V 18599*, developed by the *Fraunhofer Institute for Building Physics* [Erh+07].

Some examples for whole-building energy simulation software are *Tas* by *EDSL* [EDS14], *VE* by *IES* [IES14], and *EnergyPlus* by the *US Department of Energy* [US14]. Also the BIM software vendors *Graphisoft* and *Autodesk* offer energy simulation solutions built upon their BIM tools, namely *EcoDesigner STAR* [Gra14b] and *Green Building Studio* [Aut14a].

3 Related Work

3.1	Building Information vs. Energy Modeling	19
3.2	Berkeley Lab Approach	21
3.2.1	Space Boundaries	21
3.2.2	Semi-Automated BIM to BEM Process	23
3.3	Numerical Robustness in Computational Geometry	25

Building on the theoretical foundations provided in the previous chapter, an overview about recent research in the field of building energy analysis is given here.

3.1 Building Information vs. Energy Modeling

The simulation of thermal loads and energy use of buildings is also referred to as building energy modeling. Ryan and Sanquist state that building energy models exist since the 1980s, but have gained momentum only in recent years due to more computer power as well as increasing energy prices and awareness regarding global warming. Now that energy modeling is more common, stakeholders like architects, property developers and governments rely on its results for predictions about energy usage [RS12].

Mitchell observes that the adoption of BIM has increased in recent years, and that both the construction industry and governments have been a part of it. To make it more effective, he sees an "urgent need for cultural change" in the industry in terms of collaboration, as "there is little history of good process management". Regarding thermal performance analysis, he also notes that it has a long history of application, and is now becoming more important in the light of climate change. Besides the need for accurate geometric representation, he emphasizes the need for BIM compatible product libraries in this context, calling manufacturers of building components to

3 Related Work

provide accurate representations of their products with material and performance properties [Mit11].

Hitchcock and Wong describe the current practice of building energy modeling as “a combination of science of art”, with the science being in the simulation algorithms, while extracting the input data for these algorithms from different sources is the art. This is commonly performed manually, according to individual methods and rules-of-thumb that specialists develop based on their experience. The ideas of BIM as a resource for consistent building data to be used by all involved disciplines throughout the building life-cycle stands in opposition to the status quo, as the disciplines (e.g., architects, engineers and energy specialists) work rather separately, which results in inconsistencies between an “architectural view” and a “thermal view” of the same building. Hitchcock and Wong analyze different efforts regarding how to close the gap between BIM and energy simulation and conclude that automatic data exchange between the two is still an “elusive goal” [HW11].

Tupper, Franconi, et al. think along similar lines. They identify several prevalent issues regarding building energy modeling, such as a lack of consistency and reproducibility. The reasons for this include the poor support for data exchange in current building analysis tools, which requires extensive pre-processing of input data [Tup+11]. To improve the unsatisfactory situation, Franconi introduces the idea of a building energy modeling methods and processes framework, aiming to structure and organize different modeling tasks across applications, and the creation of guidelines [Fra11].

Specific research regarding the use of IFC for energy analysis is performed by Cemesova et al., who propose an extension of the IFC schema with an energy domain. They created a prototype using an extended IFC schema, borrowing concepts from a building performance simulation tool for passive houses [CHR13].

Manke et al., on the other hand, analyze how data for energy simulation can be imported from 2D CAD tools. They state that even though 2D tools are used by “a majority of the design professionals”, there is more effort put into developing interoperability with 3D and BIM data. They conclude that most tools have the ability to import standard 2D CAD file formats, but the geometry has to be redefined manually before the simulation [MGD13].

3.2 Berkeley Lab Approach

The work of a group at the *Lawrence Berkeley National Laboratory* that has studied how BIMs can be used as a basis for energy performance simulation, deserves particular attention. They have described *space boundaries* as the fundamental concept of *building energy simulation models (BEMs)*, and introduced a semi-automated process to convert IFC-based BIMs into BEMs, including an algorithm for the respective geometry simplification.

3.2.1 Space Boundaries

Bazjanac states that the current approach of building energy modeling is to manually remodel a given building geometry in a particular way that is suitable as input for energy simulation. These ad-hoc processes tend to be inaccurate and hardly reproducible. The resulting definitions of building elements (walls, slabs, windows, etc.) actually represent space boundaries, but modelers often do not realize their critical characteristics. Thus, Bazjanac systematizes five “levels” of space boundaries as a cornerstone for a standardized process of preparing BEMs [Baz10]:

- **1st level:** A 1st level space boundary is the surface of a building element that is continuously visible from within a space. For example, a wall may be modeled as one single object bounding several spaces on each side. In this case, the surface of this wall that is visible from a space (i.e., the part between two walls perpendicular to it) is a 1st level space boundary for this space. It is not relevant what is on the other side of the wall. The yellow surface in the left part of Figure 3.1 is a 1st level space boundary.
- **2nd level:** For energy simulation purposes, space boundaries must be modeled in a more detailed way that considers this “other side”, to account for different heat flows between spaces. If, for example, a wall separates one space on one side from two spaces on the other side, the 1st level boundary needs to be split up into two 2nd level boundaries, illustrated by the blue surfaces in the right part of Figure 3.1.
- **3rd level:** Not all parts of a 1st level boundary constitute 2nd level boundaries. Those parts where there is not a space, but a building element on the other side, are considered 3rd level space boundaries. In Figure 3.1, the narrow red part with another wall on the other side, is such a 3rd level boundary.

3 Related Work

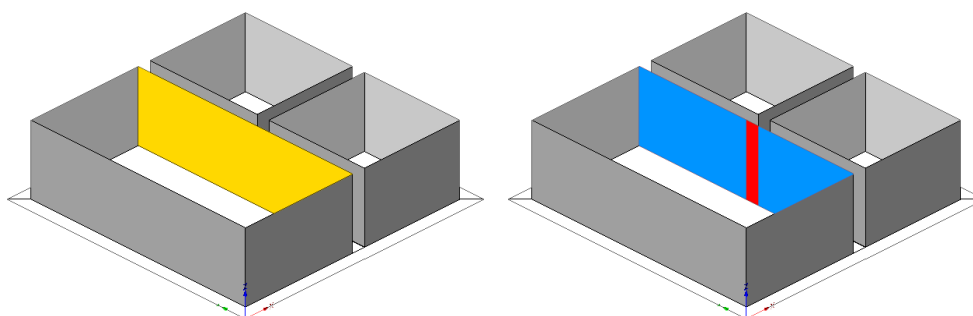


Figure 3.1: An example for space boundaries of a wall with one space on one side and two spaces on the other side. On the left, the yellow surface represents a 1st level boundary. On the right, the other side, with the perpendicular wall and the two spaces, is considered, constituting two 2nd level boundaries (blue) as well as a 3rd level boundary (red). Images based on [HL10].

- **4th level:** Depending on the placement of wall reference lines, some areas can be left unaccounted for by 3rd level boundaries, e.g., a small part of a slab's surface where two perpendicular walls meet on the other side. Such areas form 4th level boundaries.
- **5th level:** When building elements intersect at a non-right angle, another type of areas without perpendicular heat flow can occur, which are called 5th level space boundaries. As with the 4th level, there is no need to go into detail for the purpose of this thesis.

To summarize Bazjanac's classification of space boundaries, there are two main types of space boundaries: while a 1st level boundary simply corresponds to an entire surface of a space, the higher levels account for what is on the other side and thus can be used for thermal analysis. 2nd level boundaries represent areas of constant one-dimensional heat flow between spaces, 3rd to 5th level boundaries represent areas with no one-dimensional heat flow. Interior 2nd level boundaries always occur as pairs (one space boundary for each space). Bazjanac comments that "some CAD software developers" implement interior space boundaries on single planes along the center-lines of the building elements, instead of pairs. In his opinion, such simplifications, although claimed to be "good enough", should be avoided, as they can cause significant errors [Baz10].

In IFC, space boundaries can be modeled with the *IfcRelSpaceBoundary* entity, which has relationships with a space, an element, and an *IfcConnectionGeometry*. IFC also distinguishes the two basic types of space boundaries, while it only allows one of the

3 Related Work

two in a single model, as specified by the *Space Boundary Add-on* for the *Coordination View* MVD [HL10]. The space boundary classification of IFC can be summarized as follows:

- **1st level:** “Other side” has no influence. (cf. Bazjanac’s 1st level)
- **2nd level:** “Other side” has influence. (cf. Bazjanac’s 2nd to 5th level)
 - **2a** There is a space behind.
 - **2b** There is a building element behind.

3.2.2 Semi-Automated BIM to BEM Process

O’Donnell et al. describe the Berkeley Lab approach as a four-step process [ODo+13]:

1. Creating the BIM with a CAD tool
2. Validating the BIM with a model checking tool
3. Processing the geometry with a special tool
4. Validating the geometry for simulation

They list several points that need to be considered while creating the BIM, such as building location (latitude, longitude) and orientation (north axis), material definitions, zoning, etc. After exporting the model as IFC from the BIM authoring tool (they use *ArchiCAD* in their case study [Gra14a]), it should be checked with a model checking tool like Solibri against a rule set provided by the Berkeley Lab. The Solibri screenshot in Figure 3.2 shows this rule set. It ensures that the spatial structure is available, spaces are defined correctly, the geometries are complete, match the measurements in the object attributes, and do not intersect each other, etc. Any errors that may be detected should be resolved iteratively with the authoring tool, to make the model ready for the next step. This can be particularly difficult if the principle “develop BIM models with BEM in mind”, listed as one of the critical success criteria, is not followed [ODo+13].

Geometry Processing

For the conversion of the IFC model’s building geometry into such geometry that can be used as an input for the energy performance simulation engine *EnergyPlus*, the Berkeley Lab group has developed a piece of software called *Space Boundary Tool* (SBT). Rose and Bazjanac describe the underlying algorithm in an article [RB13].

3 Related Work

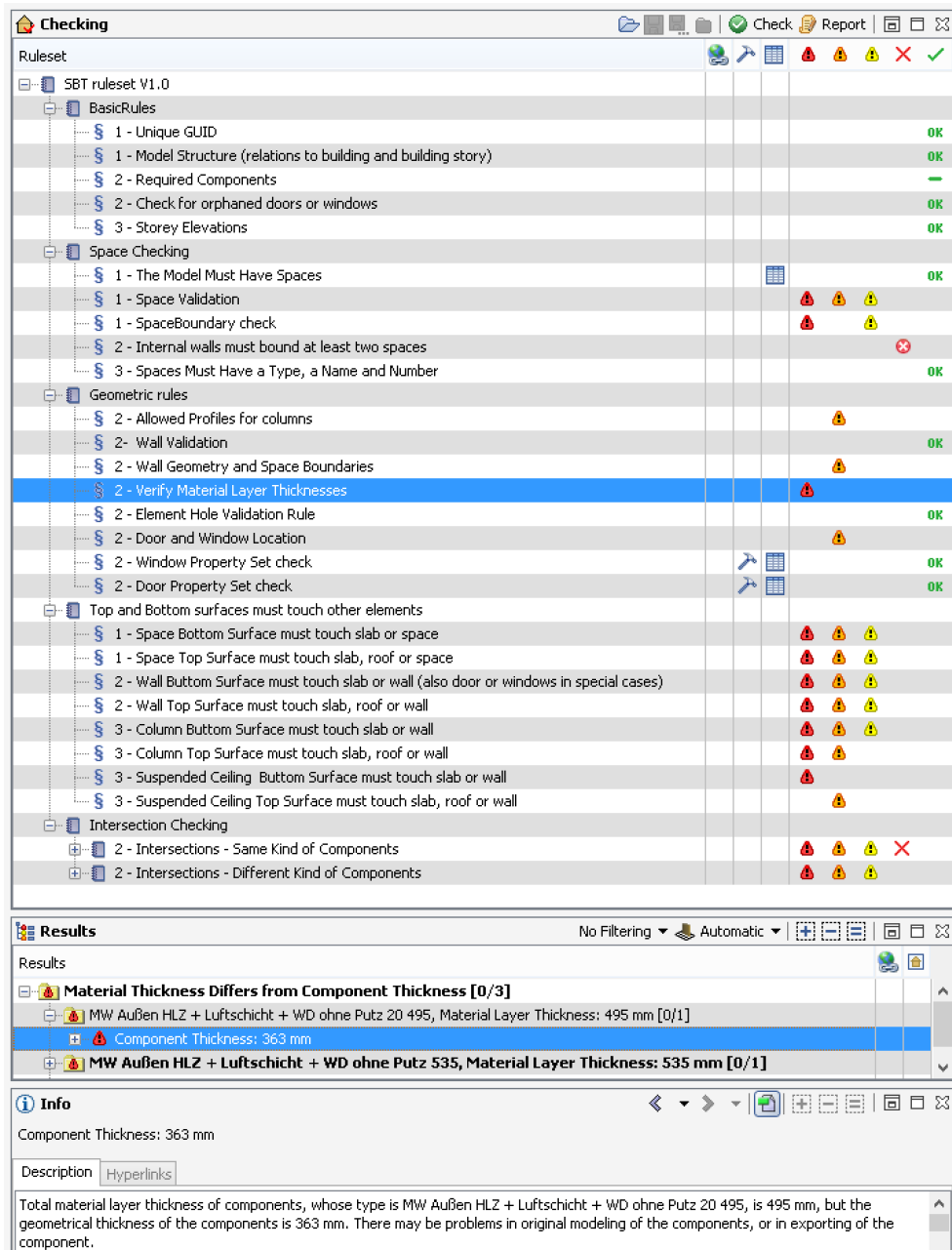


Figure 3.2: Screenshot from Solibri Model Checker showing the rule set provided by the Berkeley Lab. Selected is a negative result of a rule that checks whether the defined material layers match the geometry of a wall in thickness.

3 Related Work

Their algorithm starts with 1st level space boundaries (which are present as the faces of *IfcSpace* geometries) and splits them into higher level boundaries based on the detection of possible one-dimensional heat flows. For this purpose, a “building graph” is built, where paths (named “transmission paths”) represent such heat flows.

Vertices are either 1st level space boundaries or parts of building elements between them.¹ Edges are physical connections between them. Each vertex holds either one polygon, given by the 1st level space boundary, or a pair of two polygons, derived from the building element geometry. An edge between two vertices is created if any two of their polygons are coplanar and intersect each other.

Figure 3.3 is taken from the original paper and shows a part of a building graph as well as the corresponding building geometry from above. The vertices s_0 to s_5 are derived from 1st level space boundaries, c_0 to c_2 and w are derived from building elements. One of the transmission paths is from s_1 via w to s_3 , because one of the polygons of w is coplanar and intersects the polygon of s_1 , and the one other is coplanar and intersects the polygon of s_3 . The other transmission paths are (s_0, c_0, w, s_3) , (s_1, w, c_1, s_4) , (s_1, w, s_5) , and (s_2, c_2, w, s_5) . Note that the end-points of each of these (internal) transmission paths are 1st level space boundaries (the exterior facade is a special case), whereas multiple building elements in between are possible. No paths are built where the end-points would be too far away from each other to allow heat flow between them.² Each path in the building graph constitutes a pair of thermal space boundaries, which essentially make up the BEM geometry for *EnergyPlus*.

3.3 Numerical Robustness in Computational Geometry

Since this thesis involves the application of computational geometry algorithms, some thought has to go into aspects of numerical robustness.

A quote by Mulmuley poetically summarizes the root of many problems in this area: “Dealing with the finite nature of actual computers is an art that requires infinite patience” [Mul94]. As Hoffmann observes, implementations of geometric

¹More exactly, Rose and Bazjanac describe such a vertex as “an effect of a building element on potential one-dimensional heat transfer [RB13].”

²Rose and Bazjanac use a value of 0.5m

3 Related Work

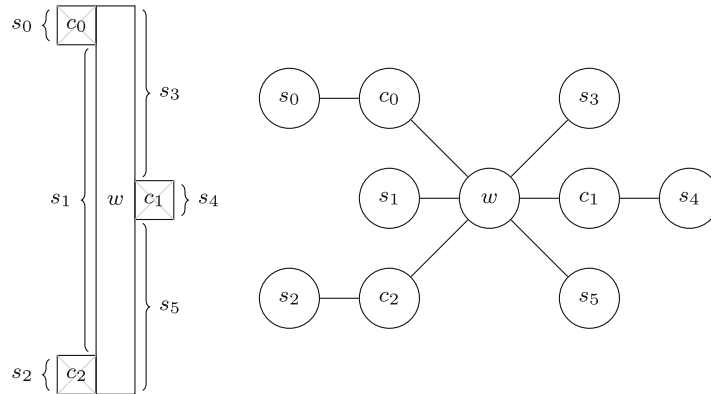


Figure 3.3: Exemplary building geometry from, viewed from above, with corresponding building graph. Vertices are either 1st level space boundaries or parts of building elements between them. Edges are physical connections between them. Paths represent one-dimensional heat flows between spaces. Image taken from [RB13].

operations are error-prone, and many errors are rooted in the fact that floating-point representations are used for algorithms that assume real representations: “objects conceptually belonging to a continuous domain are analyzed by algorithms doing discrete computation.” This is a serious problem particularly for geometric objects, which consist of a numerical part (vertex coordinates, plane equations) and a combinatorial part (edge and face boundaries, adjacencies, incidences). Imprecise numbers can lead to contradictory information about an object, e.g., four faces might be supposed to meet in a common vertex, but their plane equations might represent planes that intersect in four different points [Hof89]. In geometric computations, imprecise numbers can quickly lead to errors in the combinatorial part [Sch97].

To tackle this issue, two basic approaches can be applied: either adapt the algorithm to deal with imprecise computation, or employ exact computation [Sch97].

How to deal with imprecise computation is strongly dependent on the application. Several techniques to reach robustness despite imprecision have been presented for specific geometric algorithms, but a general solution does not exist. The problem is to avoid inconsistencies in the evaluation of predicates (e.g., does a point lie on a line) based on the results of earlier evaluations [Sch97].

The other approach is to make exact computation possible. As Yap notes, the notion of geometric exactness is weaker than numeric exactness, as numeric exactness is

3 Related Work

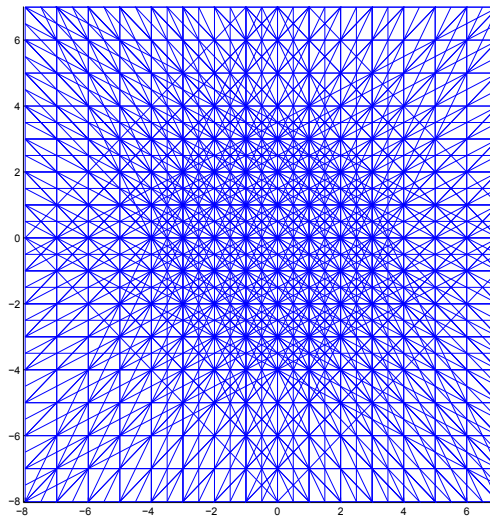


Figure 3.4: This grid gives an idea about the high number of representable planes even with a low resolution for the coefficients a, b, c, d in the plane equation $ax + by + cz + d = 0$. In this case, a, b, c have 2 bits each, and d has 4 bits. Image taken from [KUF14].

not always necessary, as long as the combinatorial decisions are errorless. Obviously, a prerequisite for exact geometric computation is exact input, which is not always available. In some cases, such as a set of points without any constraints as input, even approximate inputs can be assumed as “exact”. In most cases though, one has to deal with inexact input, either by reformulating the algorithm so that the input can, again, be treated as exact (e.g., inexact points become exact centers of small spheres), or by transforming the input to exact values, which is not trivial [Yap97].

Krispel et al. describe a set of exact geometric predicates based on plane-based representations for polygonal meshes. That is, vertices are expressed as intersection of three planes, which are represented by the coefficients of the plane equation as integers. As Figure 3.4 shows, even low precision for the coefficients yields a high number of possible plane configurations. However, as the input has to be present in this limited precision, it is more suitable for certain applications, like procedurally constructed meshes. The effects of importing arbitrary polygon meshes into the plane-based representation and the necessary quantization has yet to be investigated [KUF14].

4 Algorithm

4.1	Reference Models	29
4.2	Input and Data Structures	31
4.2.1	Input Requirements	33
4.2.2	Geometric Representation	34
4.2.3	Boundary Parts	35
4.3	Geometric Simplification	36
4.3.1	Detecting the Boundary per Space	37
4.3.2	Boundary Parts between Spaces	39
4.3.3	Filling Gaps in the Boundary	41
4.3.4	Boundaries between Building Stories	41
4.4	Export	44
4.4.1	Processing Additional Data	44
4.4.2	ArchiPHYSIK File	45

The goal of the algorithm is to create simple building models for energy analysis based on complex building information models. Currently, this task is done manually, i.e., by re-creating models from scratch. A widely used tool to create simple 3D models is *Sketchup*, which has a freeware version [Tri14]. An add-on for *Sketchup* can be used to export models to energy analysis tool *ArchiPHYSIK*. For an effective export, certain guidelines have to be followed, for example the use of colors to distinguish different types of building elements [ANU14]. Even with the necessary modeling knowledge, this approach is feasible primarily for small buildings, but very time-consuming for larger models.

The ideal algorithm would do the same as a human expert who is modeling the data for energy analysis by looking at and using the data from the IFC model. Faced with a complex, yet incomplete representation of a real-world building, he or she decides which data are relevant and which are not based on his semantic understanding.

4 Algorithm

As discussed in Section 2.1.2, a problem with IFC, or BIM in general, is that the semantics are often not integrated in the way it would be possible. One cannot rely on all necessary data to be available as a lot of attributes and relationships are optional, and MVDs are not yet implemented on a broad basis. Naturally, the data that is present is not necessarily correct. Imperfections in the geometry, e.g., geometries of different building elements intersecting each other, are common. Perfect BIMs that do not have such flaws and contain the complete spatial structure (*IfcSite*, *IfcBuilding*, *IfcBuildingstory*, *IfcSpace*), including space boundaries to determine which building elements bound which spaces and where, would make a different, simpler approach for the algorithm possible.

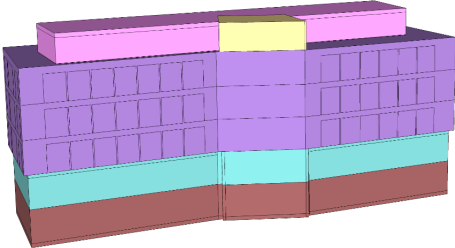
4.1 Reference Models

To guide the development, a set of reference IFC models provided by our project partners was used. The collection includes very simple models as well as large and complex real-world buildings. For each IFC model, a *SketchUp* model was created to document the desired output, along with a spreadsheet that lists how certain ambiguous aspects should be handled. Figure 4.1 shows an example of this documentation.

Model 1: Simple Bungalow The first model (see Figure 4.2) is a simple, fictitious bungalow with only one space, one door, and two windows. Around its flat roof are parapet walls, which are not part of the thermal boundary. The wall intersections at corners are “staircase-shaped”. This building can be used to test the basic functionality of the algorithm without interior boundaries between spaces.

Model 2: Simple Two-Story Home This building (see Figure 4.3) has the same footprint as the bungalow, but with a 2nd story on top. The top story has three spaces, one of which is L-shaped (that is, concave). One of the interior walls has two spaces on one side. Other than in the first model, the walls are chamfered in the corners. This model can be used to test the algorithm’s capabilities to detect interior boundaries.

4 Algorithm



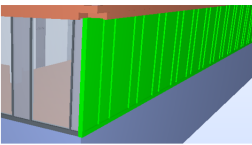
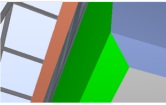
	A	B	C	D	E
12					
13	Eingabe EG: "Fassade"	Ausgabe ebene Wand mit durchgehendem Verglasungsband	Begründung Rahmenerhebung nicht von Bedeutung, Rahmenanteil wird später in APS eingegeben.	Anforderung an IFC/Zeichner/Progr. Problem hier ist, dass die Fassade mit allen Rahmenerhebungen detailliert gezeichnet, aber als EIN Bauteil "Fassade" eingegeben wurde. Hier muss das Programm automatisch erkennen, dass es die Rahmen nicht braucht und stattdessen EINE Wandebene ausgeben. Wo soll es diese platzieren? Auf Höhe der Rahmen oder des Glases? Es muss auch erkennen können, wenn Verschattungslamellen o. Ä. gezeichnet wurden, dass diese nicht maßgebend sind für die Lage des vereinfachten Fensters.	Bilddoku 
14	Grundriss	Grundriss	Maße aus Wandbauteilen; wurden in diesem Fall mit Gehung an den Gebäudeecken gezeichnet, daher einfach die length als Außenmaß genommen	fraglich, ob es für jeden Zeichner praktikabel/möglich ist, die Wände mit Gehung zu zeichnen. In ArchiCAD ja, und ob das Maximalmaß immer der Gebäudeaußenkante entspricht? Bei rückspringenden Loggien ist das nicht der Fall! (Außerdem Zusatzinfo: Maximallänge der Dämmschicht ist die Außenkante der therm. Hülle) Dämmung bei diesem Gebäude mit Dicke unter "Material" eingegeben, aber wie soll das Tool dann aus dieser Info die Lage der Dämmschicht	
15					

Figure 4.1: The output requirements for each reference model are documented by a *Sketchup* model as well as a spreadsheet explaining ambiguities and how they should be handled.

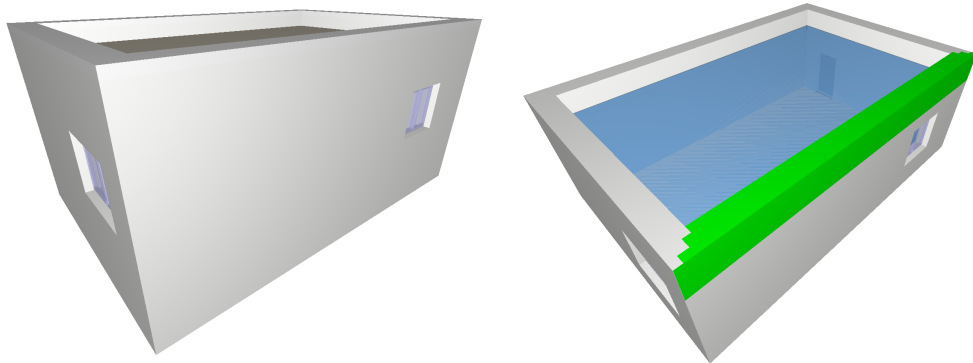


Figure 4.2: Reference model 1: simple bungalow. The left part shows the whole building with its parapet roof. On the right, one of the walls is highlighted to make the "staircase-shaped" corners visible.

4 Algorithm

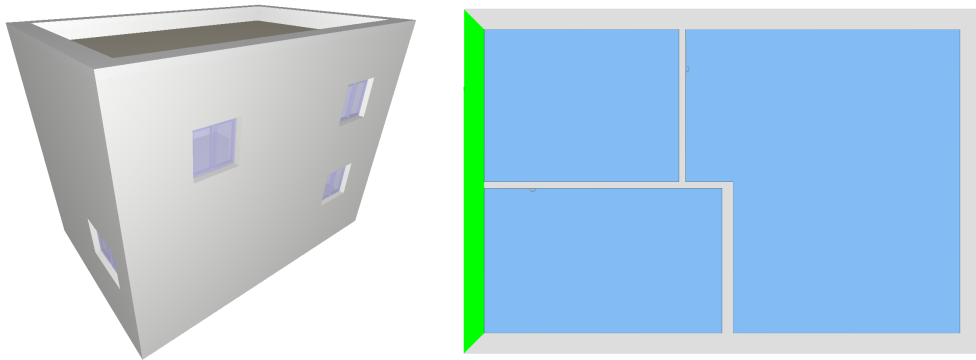


Figure 4.3: Reference model 2: simple two-story home. The left part shows the whole building. The right part shows the floor plan of the upper floor with its three spaces. One wall is highlighted to show the chamfered corners.

Model 3: Twin Home This two-story twin home (see Figure 4.4) is a real-world example with a more complex geometry. The model also contains some objects that are irrelevant for energy analysis, such as a swimming pool, trees, and carports. Also exterior parts of the site, such as sections of the lawn, are modeled as *IfcSpace*.

Model 4: Office Building This extensive model (see Figure 4.5) has six stories: a basement garage, large shop spaces in the first floor, many small office spaces in the upper floors, and a roof terrace. The building geometry is quite complex, with many columns, large windows, and a composite curtain wall.

Model 5: High-Rise Office Building This office building (see Figure 4.6) has 21 stories. In the middle of the tower is a block with elevators, staircase, and lavatories, around it one open-plan office space per story. The algorithm has to deal with these spaces with “holes” as well as the rounded corners of the tower.

4.2 Input and Data Structures

Without going into implementation-specific details, the input requirements as well as some essential data structures are described here, to make the explanation of the actual geometric simplification algorithm more comprehensible.

4 Algorithm

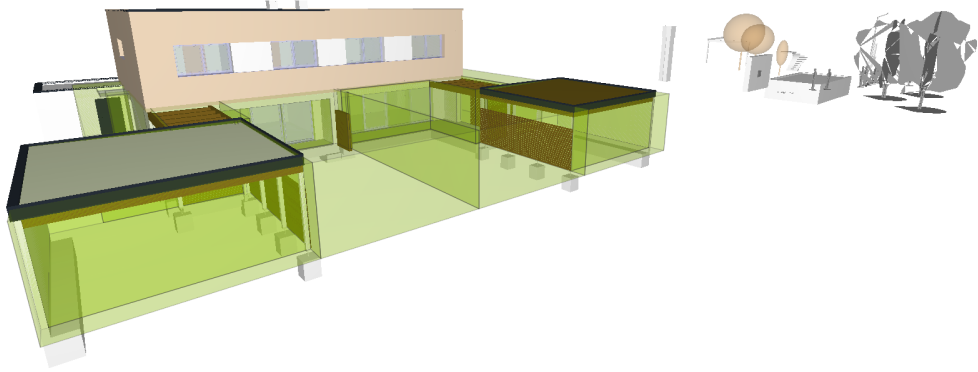


Figure 4.4: Reference model 3: twin home. In the foreground, one can see the carports and how exterior parts of the site are modeled as spaces. The model contains other irrelevant objects such as a swimming pool and trees.

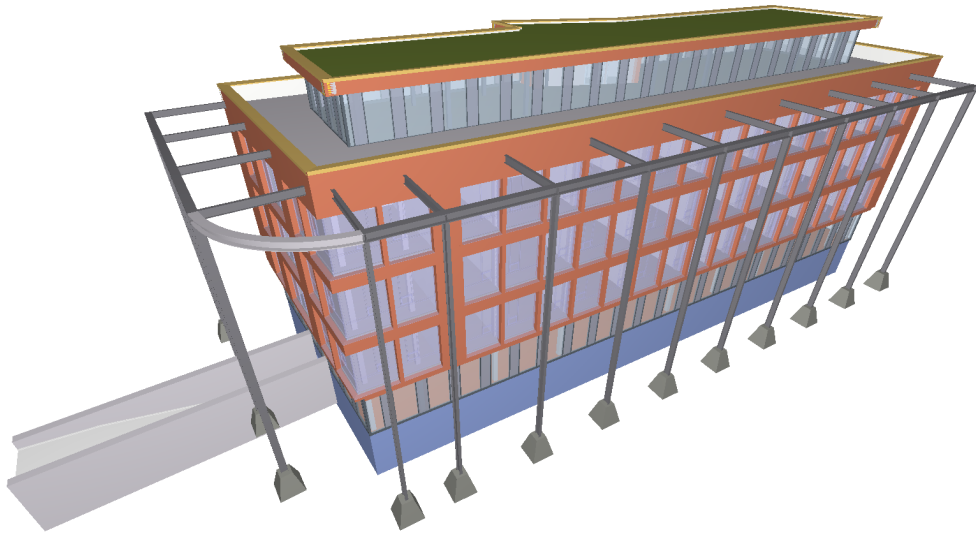


Figure 4.5: Reference model 4: office building with six stories, including a basement garage and a roof terrace.

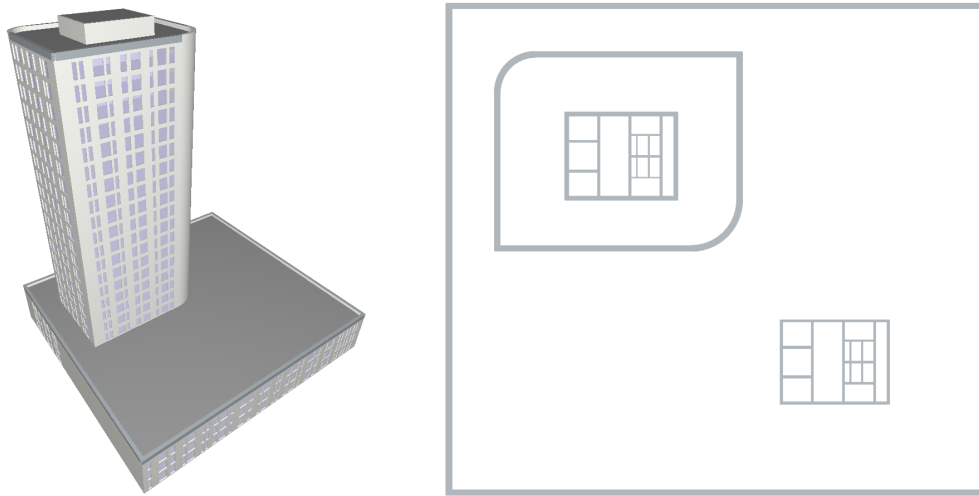


Figure 4.6: Reference model 5: high-rise office building with 21 stories. The right part shows the footprint of the building with its large lower floors, the tower with its two rounded corners, and the facility blocks with staircase and elevators.

4.2.1 Input Requirements

In response to the data quality issues in many IFC models that were discussed above, the input requirements are kept to a reasonable minimum and many data are determined through the semantically grouped geometries. The input data required for the algorithm are:

- Building elements (walls, slabs, windows, doors)
- Spaces (there should be “no air” in the building that is not inside a space)
- Full 3D geometry associated with all building elements and spaces, represented as geometry type that can be triangulated
- Thermally relevant data (material layers, window properties, etc.) associated with the building elements

As described in Section 2.1.2, IFC allows for geometry to be stored in various types of representation, e.g., constructive solid geometry objects, extrusions, or boundary representation, and the different types can be mixed within a model. Moreover, IFC objects have their own local coordinate system with a placement relative to another object or the absolute coordinate system.

4 Algorithm

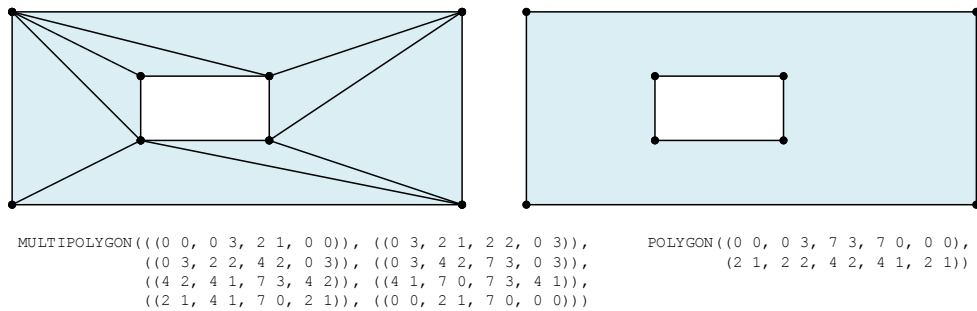


Figure 4.7: A wall with a window opening in its triangulated form (left) and after unioning the triangles (right). The vertices are the same, but instead of eight triangles, there is one polygon with a hole.

4.2.2 Geometric Representation

In a preprocessing step, all geometries are triangulated, so that the algorithm does not have to deal with the various geometry representations or different local coordinate systems. Furthermore, the triangulation eliminates curved surfaces, which are not supported by energy analysis tools and thus need to be approximated by planar surfaces. Subsequently, the triangulated geometries are simplified by unioning the triangles that belong to the same object and lie on the same plane. For example, a wall with a window opening represented as eight triangles becomes one rectangle with a hole, defined by the exact same vertices, as shown in Figure 4.7. Similar operations are performed for (intermediate) boundary geometries.

After this geometry preprocessing step, all geometries, both input (building elements and spaces) as well as output (boundary parts, described below), are represented as polygon soups. Each polygon is defined by the list of points in global 3D coordinates, the normal vector (in its normalized form), and the distance of its host plane from the origin. It also holds a reference to the building element, as it is important for the algorithm that each polygon “knows” the building element it belongs to, i.e., the semantic information is maintained. For the collection, a map with the normal vector as key and the polygons sorted by the distance as values is used for easy access during boundary detection. As slight imprecision within the normal vectors cannot be ruled out, a tolerance for the keys is applied within the implementation of the map.

4 Algorithm

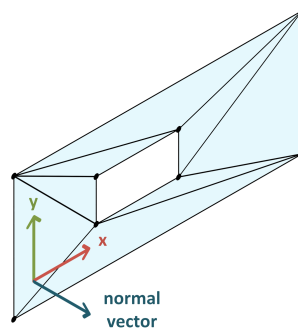


Figure 4.8: Polygons are converted from 3D to 2D by calculating an orthonormal vector to the normal vector for the x -axis, and the cross product of this vector and the normal vector as y -axis.

Significant parts of the geometric simplification are performed in 2D space. Polygons are converted from 3D to 2D by assuming their host plane as xy -plane. That is, the x -axis is an orthonormal vector to the normal vector, and the y -axis is the cross product of this vector and the normal vector. Figure 4.8 illustrates this setting. When I refer to 2D coordinates in the description of the algorithm, I imply that 3D coordinates are converted in this manner and the result is converted back.

4.2.3 Boundary Parts

A central object for the algorithm is what I call a “boundary part”. In BIMs, there are tangible building elements (walls, slabs, doors, windows, etc.), and spaces. Even though relationships between instances of these two basic types can be modeled in IFC, they remain separated from each other. A wall, for example, might have several spaces on either side. For energy analysis, this must be modeled in a different way: the spaces on both sides of a building element constitute “parts”, i.e., building elements are split up if necessary. As shown in Figure 4.9, each boundary part is associated with exactly one building element and one or two spaces (one for exterior boundary parts, two for interior ones). The geometry is homogeneous in orientation, i.e., all polygons share the same normal vector (in most cases, there is only one polygon). In cases where building elements and spaces have shapes that would imply boundary parts with heterogeneously oriented polygons (e.g., approximated rounded walls), they are simply split up.

4 Algorithm

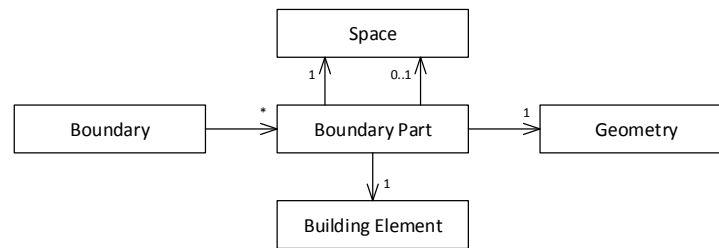


Figure 4.9: The boundary part is a central object for the algorithm. Each boundary part belongs to one (for exterior parts) or two (for interior parts) spaces and one building element.

The essence of the algorithm is finding these boundary parts, including their geometries, which will be described in the next section. From the collection of boundary parts, the output (that is, the input for energy analysis) can be derived easily. Note that the concept of boundary parts is similar to the (higher level) space boundaries in the Berkeley Lab’s algorithm described in Section 3.2. In both approaches, surfaces are split up so that each surface has only one space on either side. The obvious difference is that in the GINGER approach, there are single boundary parts between two spaces, where Berkeley Lab has pairs of two space boundaries (one for each space). The position of the boundary part surfaces is on the centerline between two spaces, or on the outer edge of exterior elements, following the norm for Austrian energy performance certificates [Aus11]. The thickness and material information comes straight from the IFC building element, which is associated with the boundary part, while in Berkeley Lab’s algorithm, layered building elements are split up into single-material elements, and the thickness is represented in the output geometry.

4.3 Geometric Simplification

The geometric simplification algorithm finds the boundary parts based on the building element and space geometries in three steps:

1. Detecting the boundary per space
2. Merging boundary parts between spaces
3. Filling gaps in the boundary

4.3.1 Detecting the Boundary per Space

In the first step, boundary parts are detected for each space as if this space was the only one. That is, it is assumed that all boundary parts of a space are exterior. Thus, at this point, each boundary part has only one space, and geometrically it corresponds to the part of the outer face of its building element vis-à-vis the space face.

Each space's boundary geometries are detected as described in Algorithm 4.1. The polygon map described in Section 4.2 makes finding the right polygons to consider easy: starting from a space face, only those polygons with the same normal vector and a greater distance to the origin have to be checked, starting with the polygon closest to the space face. The actual geometric operations are performed in 2D space: intersections of building element polygons with the space face polygon become part of the boundary. By comparing each checked polygon to the boundary geometry that has been detected so far (p_b in the pseudo code), it is ensured that polygons "further out" the building geometry are not wrongly detected. Additionally, an upper bound (e.g., a distance of 1 m to the space face) is used to avoid unnecessary checks.

Algorithm 4.1 Find Boundary for Space

```

1: polygons  $p$  have a normal vector  $n$  and a distance from origin  $d$ 
2: let  $P$  be the set of all polygons of any building element
3: let  $P_s$  be the set of all space face polygons
4: for all  $p_i \in P_s$  do
5:    $P_c \leftarrow \{p \in P: n = n_i \wedge d > d_i\}$ 
6:    $p_b \leftarrow \emptyset$  (an "empty" polygonal geometry)
7:   for all  $p_j \in P_c$  do
8:     if  $p_j \cap p_i \neq \emptyset$  then
9:       add  $p_j \cap p_i \setminus p_b$  to the collection of boundary polygons
10:       $p_b \leftarrow p_b \cup p_j \cap p_i$ 
11:     end if
12:   end for
13: end for

```

The resulting boundary geometries are grouped per building element and stored in boundary parts. The result of this step performed for all spaces of an exemplary building story is shown schematically in Figure 4.10.

4 Algorithm

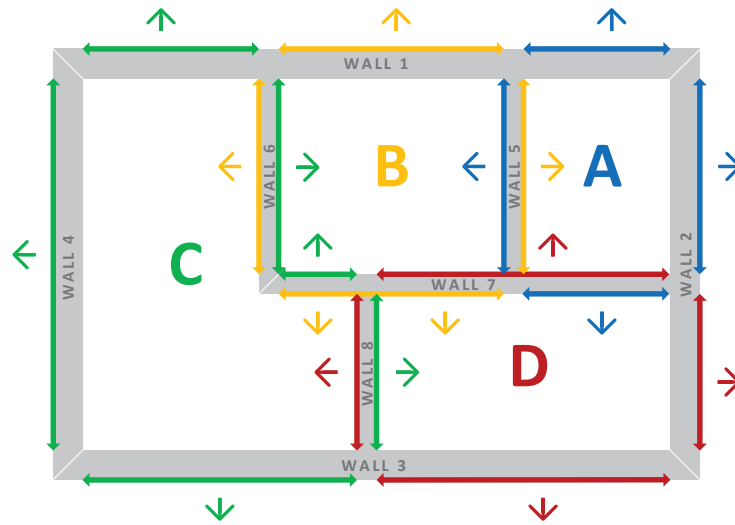


Figure 4.10: Footprint showing the state after the first step of finding boundary parts. For each space, those parts of building elements that are parallel to and within the limits of a space face are detected as boundary parts of this space.

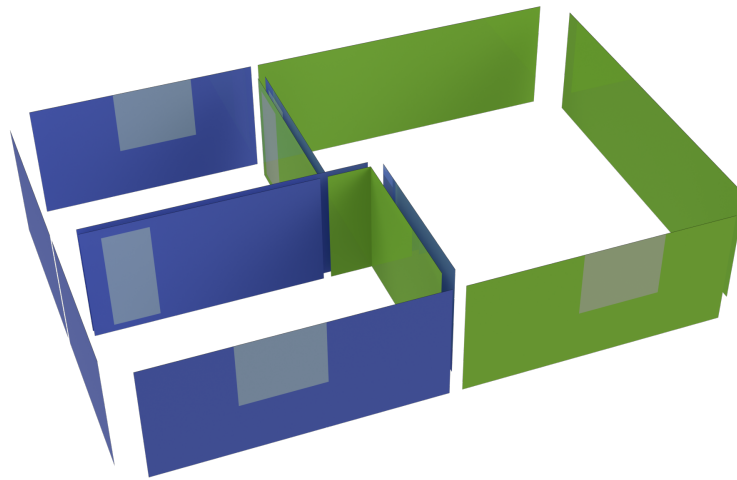


Figure 4.11: The result of the first step as a 3D rendering. Each space has been treated as if it was the only space: there is no distinction between interior and exterior boundaries yet.

4 Algorithm

Windows and doors form a special case. Their geometries typically include many details that are not relevant for energy modeling, e.g., detailed framing, window handles, door knobs, etc. On the other hand, quite relevant data for windows are not included in IFC: there is no information about what part of the geometry is glass and what part is framing. This would be useful for energy analysis as it allows calculating the length of the glass edge bond. Because this information cannot be modeled in IFC, the framing geometry is not relevant for the algorithm's output. Thus, window and door geometries are replaced by their oriented bounding boxes beforehand. Moreover, after the boundary detection, it is ensured that window/door boundary parts lie on the same plane as the respective wall boundary parts, as Figure 4.11 shows.

4.3.2 Boundary Parts between Spaces

During the first step, all boundary parts have been detected. However, the neighbor relationships between spaces, whether next to each other or on top of each other, have not been considered yet. Thus, in the following step, those pairs of boundary parts that lie between two spaces are detected and merged into one boundary part that is associated to both spaces.

Algorithm 4.2 Merging Interior Boundary Parts

```
1: boundary parts  $b$  have a building element  $e$ , a space  $s$ , a space  $t$ , and a geometry  $g$  with normal vector  $n$ 
2: let  $B$  be the set of all  $b$  found for any space (all having  $t = null$ )
3: for all pairs  $(b_i, b_j) \in B$  do
4:   if  $e_i = e_j \wedge s_i \neq s_j \wedge n_i = -n_j$  then
5:      $g_{intersection} \leftarrow g_i \cap g_j$ 
6:     if  $g_{intersection} \neq \emptyset$  then
7:       create new  $b_{i,j}$ 
8:        $e_{i,j} \leftarrow e_i$ 
9:        $s_{i,j} \leftarrow s_i, t_{i,j} \leftarrow s_j$ 
10:       $g_{i,j} \leftarrow g_{intersection}$  (with  $n_{i,j} \leftarrow n_i$ )
11:     end if
12:   end if
13: end for
```

Algorithm 4.2 describes this step. To qualify as a “merging candidate” pair, two

4 Algorithm

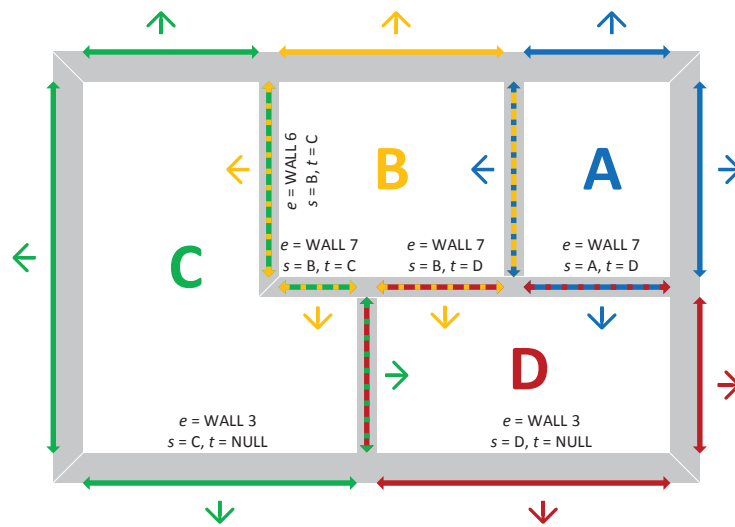


Figure 4.12: Footprint showing the state after the second step, merging interior boundary parts. Pairs of boundary parts that lie between two spaces are merged into one. The arrows indicate the normal vectors.

boundary parts have to belong to the same building element, bound two different spaces, and have opposite normal vectors, like the boundary parts between belonging to wall 6 between the spaces A and B in Figure 4.10. If these conditions are met, the two geometries are intersected in 2D space. An empty intersection means that the pair does not constitute an interior boundary part (the two spaces are not neighbors here), thus no action is required. If there is an intersection, a new boundary part, associated to both spaces, is created. Its geometry corresponds to the (2D) intersection and is placed in the center of the two original geometries. It inherits the normal vector of the “first” original boundary part, that is, the normal vector points from space s to space t . This is simply a convention to be able to tell which space is on which side of the boundary part during export. The result of the merging step is shown in Figure 4.12.

The original boundary parts are not discarded immediately, because they may be merged into more than one interior boundary part. For example, consider a segment of a wall with one space on one side, and two spaces on the other side. In such a case, three original boundary parts are merged into two interior boundary parts, while the original part belonging to the single space on one side is split up and merged into both of those.

4.3.3 Filling Gaps in the Boundary

After all interior boundary parts have been merged, the number of parts is down to the final number, and every part is at its final position. The last step to be performed on the geometry is the elimination of gaps between parts which occur depending on the simplification scheme. Since the spaces represent net volumes (i.e., the space geometries do not include building elements), gaps at the corners are unavoidable as the boundary parts are placed on the outside of building elements.

Adjacent non-horizontal boundary parts can be determined by checking the IFC relationship *IfcRelConnectsElements* for their building elements. If this relationship is not set in the model, adjacency can be heuristically determined by checking if the smallest distance between any pair of points of two boundary parts is within a threshold.

For each pair of adjacent boundary parts, gaps are filled by intersecting their host planes, as described in Algorithm 4.3.

Algorithm 4.3 Fill Gap between Adjacent Boundary Parts

```

1: for all  $edge_i \in polygon_{part1}$  do
2:    $point_{intersection} \leftarrow$  host line of  $edge_i \cap$  host plane of  $polygon_{part2}$ 
3:   if  $point_{intersection} \neq \emptyset$  then
4:     mark  $point_i \in edge_i$  closer to plane to be replaced by  $point_{intersection}$ 
5:   end if
6: end for
7: replace marked points
8: repeat with swapped roles for  $polygon_{part1}$  and  $polygon_{part2}$ 

```

Figure 4.13 shows a step-by-step example of boundary detection, merging, and gap filling. In Figure 4.14, the result after the gap filling is depicted.

4.3.4 Boundaries between Building Stories

In principle, the described algorithm works for any orientation of boundaries, whether vertical, sloping, or horizontal. However, the filling algorithm would have trouble with filling gaps between horizontal boundary parts and certain compositions of vertical boundary parts, for example those of a round wall, or a continuous boundary part in one storey and multiple boundary parts directly above it, with

4 Algorithm

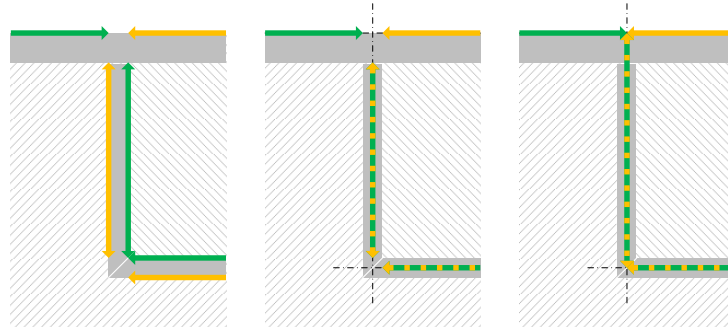


Figure 4.13: The three steps of extracting the boundary from a full building geometry. Left: boundary part detection per space. Center: pairs of interior boundary parts are identified and merged, i.e. belonging to two spaces now, and centered. Right: gaps between boundary parts caused by building elements are filled by intersecting them with their neighbors.

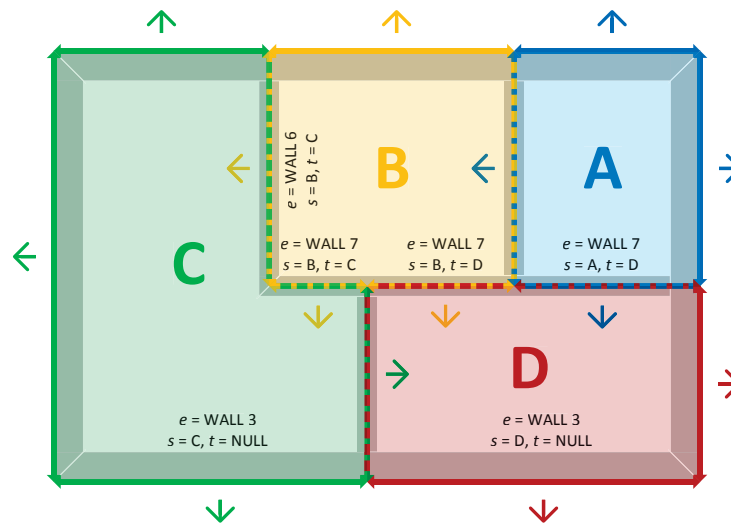


Figure 4.14: Footprint with final boundary parts. Each part now belongs to exactly one building element and one (exterior boundary) or two (interior boundary) spaces.

4 Algorithm

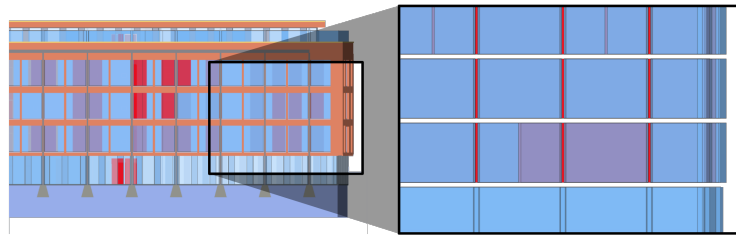


Figure 4.15: Profile view of IFC space geometries in the office building reference model. Building stories can easily be identified based on the spaces' z-coordinates.

horizontal boundary parts at different elevation (because of different slab thickness) in between. Moreover, staircase openings in slabs are to be ignored in the energy model (the horizontal boundary part continues through the opening).

For these reasons, horizontal boundaries are treated as a special case: the presented algorithm steps are performed story by story, and the horizontal boundaries are added afterwards.

Hence, the information about which spaces belong to which story is needed first. This can be retrieved from the spatial structure relationships in IFC. However, as already mentioned in the beginning of this chapter, these relationships are sometimes not available, and sometimes even wrong. Therefore, GINGER is able to optionally detect the stories based on the elevations of the space geometries. As shown in Figure 4.15, the storeys can be detected by gaps between the space geometries throughout the building, which can be identified by the spaces' minimal and maximal z-coordinates. These gaps also determine the exact elevation of the horizontal boundary parts beforehand: they lie on the centerlines between stories.

The geometries for the horizontal boundary parts are created by forming polygons out of the horizontal edges of the non-horizontal boundary parts (i.e., the polygons are already there, they just have to be made explicit). The building elements (slabs) belonging to these new boundary parts can be found easily based on their geometries, similar to the non-horizontal case, except that the boundary geometries are already determined. The merging (including splitting) of horizontal boundary parts works exactly the same as for the others.

4 Algorithm

Putting the pieces together, the overall process for the geometric simplification looks like this:

- Detect building stories
- Determine elevation of horizontal boundaries
- Detect (non-horizontal) boundaries per space
- Merge interior boundaries
- Fill gaps between non-horizontal boundaries
- Fill gaps between non-horizontal and horizontal boundaries (based on their elevation)
- Detect horizontal boundaries based on the horizontal edges of the non-horizontal boundaries
- Merge interior horizontal boundaries

The result of this process is a set of interior and exterior boundary parts.

4.4 Export

After the geometric simplification, additional data needs to be prepared for the export along with the boundary geometry.

4.4.1 Processing Additional Data

The boundary parts now get associated with additional relevant data from the depths of the IFC model which are retrieved via their building element associations. Also general model data like location and azimuth are exported.

Most important for energy calculations are of course the materials that the building elements are composed of. *IfcMaterial* objects are associated to building elements via *IfcMaterialLayerSetUsage* (holds a direction sense, positive or negative), *IfcMaterialLayerSet*, and *IfcMaterialLayer* (holds the thickness). The thermal transmittance values may be available in an *IfcThermalMaterialProperties* object. If no value is stored, only the name of each material is exported, and the thermal transmittance (U-value) is calculated in ArchiPHYSIK. Material layers are exported in the direction from the first space of the boundary part to the second space (or the outside, for exterior parts), which corresponds to the normal vector of the part. To ensure correct order,

4 Algorithm

the normal vector, the element's *IfcLocalPlacement* structure (nested coordinate systems), and the direction sense from the *IfcMaterialLayerSetUsage* object have to be considered.

For windows, relevant information is stored in several objects. For example, the thermal transmittance coefficient can be found in the property set *Pset.WindowCommon*. Another relevant value there is the glazing area fraction. In *Pset.WindowGlazingType*, the number of glass layers and their thicknesses can be stored.

With the *IfcZone* entity, spaces can be grouped into zones (e.g., by usage: office, storage, commercial, residential, etc.). As this might be used for all kinds of groupings, it is an option in GINGER to either export these zones, or every single space as a separate zone. Other than that, zoning is outside the scope of this thesis and has currently to be done in *ArchiPHYSIK*.

4.4.2 ArchiPHYSIK File

All this data is exported as an *ArchiPHYSIK* file. The *ArchiPHYSIK* (.aps) format is an XML¹ format according to a schema [ANU05]. The file includes, among others, definitions of building elements including their geometries (that is, the boundary geometries) and material layers, zones, and the relationships between them. The geometries are represented as SVG² paths, that is, as 2D coordinates, whereas 3D information is added by defining the three axes and the origin. Figure 4.16 shows an excerpt from an *ArchiPHYSIK* XML file exported by GINGER. It includes the zone definitions and a wall with its geometry (the first line describes the outer ring, the latter a hole for a window) and material layers, as well as references to two zones.

¹Extensible Markup Language [W3Co8]

²Scalable Vector Graphics [W3C11]

4 Algorithm

```
<?xml version="1.0" encoding="UTF-8" ?>
<archiphysik xmlns="http://www.a-null.com/archiphysik/2005"
  xmlns:svg="http://www.w3.org/2000/svg">
  <building-structure>
    <header>
      <created-date-time>2014-08-04T12:03:23</created-date-time>
      <cad-host>GINGER BIMserver Client</cad-host> ...
    </header>
    <dictionary>
      <thermalzones>
        <zone idx="-1" name="Exterior air" condition="EXT_AIR" />
        <zone idx="1" name="02.001" condition="INT_COND" />
        <zone idx="2" name="02.002" condition="INT_COND" />
        <zone idx="3" name="01.001" condition="INT_COND" />
        <zone idx="4" name="02.003" condition="INT_COND" />
      </thermalzones> ...
    </dictionary>
    <walls>
      <wall name="Wand-002" unique_guid="925..."
        orientation="4.712..." declination="0.0" thermalZoneIdxA=
          "3" thermalZoneIdxB="-1" glazingFactor="1.0">
        <geometry origin="(12.0, 0.0, 0.0)" axis_x="(-0.0, 0.707...,
          0.707...)" axis_y="(0.0, -0.707..., 0.707...)"
          axis_z="(1.0, 0.0, 0.0)">
          <svg:path fill-opacity="1.0" stroke-width="0.01"
            d="M -0.1060 -0.1060 L 2.3511 2.3511 L 8.0079 -3.3057
              L 5.5507 -5.7629 L -0.1060 -0.1060" />
          <svg:path fill-opacity="1.0" stroke-width="0.01"
            d="M 4.2156 -2.9428 L 5.6298 -4.3571 L 6.6905 -3.2964
              L 5.2763 -1.8822 L 4.2156 -2.9428" />
        </geometry>
        <composite>
          <compLayer name="Verputz, Gips" thickness="0.015" />
          <compLayer name="Beton, Stahlbeton 25" thickness="0.25" />
          <compLayer name="Dämmung, hart EPS" thickness="0.2" />
          <compLayer name="Verputz, Kunstharz" thickness="0.005" />
        </composite>
      </wall> ...
    </walls> ...
  </building-structure>
</archiphysik>
```

Figure 4.16: Example of an *ArchiPHYSIK* XML file (excerpt). Two of the defined thermal zones are referenced by the wall. The wall geometry is defined by the origin and the three axes as well as 2D paths. The material layers are listed with names and thicknesses, ordered from zone A to zone B.

5 Implementation and Results

5.1	Key Aspects of Implementation	47
5.1.1	BIMserver	47
5.1.2	JTS Topology Suite	49
5.1.3	Guava Library	50
5.1.4	3D Geometry Output	50
5.2	Results	51
5.2.1	Performance	54
5.2.2	Problem Cases	54

A prototypic implementation of the algorithm, the results it is able to produce, and problems that occurred, are presented in this chapter.

5.1 Key Aspects of Implementation

The prototype for the algorithm is implemented in Java. This decision was mainly influenced by the low threshold regarding build management, platform independence, experience among the GINGER project partners, and the availability of libraries that enabled an effective implementation.

5.1.1 BIMserver

The presented implementation is a client for *BIMserver*, which is an open source model server for IFC models [Bee+11]. It is developed by researchers from *The Netherlands Organisation for Applied Scientific Research (TNO)* and *Eindhoven University of Technology*. The community is active and the software is constantly improved. Bugs

5 Implementation and Results

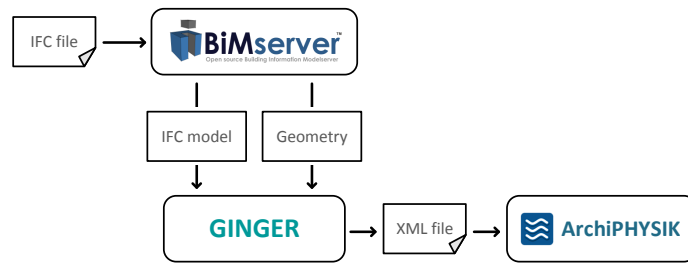


Figure 5.1: The system architecture shows that the IFC files are checked into the *BIMserver*, where the model structure is stored in a database and triangulated geometry is generated. *GINGER* downloads both from there and creates an XML file as input for *ArchiPHYSIK*.

that I reported were quickly resolved. Room for improvement is left regarding the documentation.

BIMserver does not store IFC files. Instead, they are “checked in”, parsed and interpreted, and the model data is stored in a relational database. This enables merging, filtering and querying, and object-level versioning, while keeping the IFC data schema. However, *BIMserver* itself is “only” a platform, without a user interface, intended for other software to communicate with it via web service interfaces. *BIMserver* has interfaces for JSON¹ as well as SOAP [W3Co7] calls. Related projects are built on top of it and provide web interfaces for model management (e.g., *bimvie.ws*) or 3D viewers (e.g., *BIMsurfer* [BIM14]). The company *Catenda* provides a SaaS (Software as a Service) solution based on *BIMserver* called *bimsync* [Cat14].

A key feature is the built-in IFC render engine to generate triangulated geometry upon check-in, and store it along with the model data. As discussed in Section 4.2, the *GINGER* algorithm uses polygon soups as its internal geometric representation. Thus, the triangulated geometries from *BIMserver* are an ideal starting point.

The client uses *BIMserver*’s Java client library to retrieve both the model data and the geometry, as illustrated in Figure 5.1. Thus, the model structure of IFC is available as an object hierarchy in Java. The geometry is serialized into a JSON string by *BIMserver* and deserialized into Java objects in the client with the help of the *Gson* library [Goo14a]. The building elements and spaces are mapped with their geometries via the unique object identifiers (“Oid”) which *BIMserver* assigns to all IFC objects, and which is also contained in the JSON geometry strings, as shown in Figure 5.2

¹JavaScript Object Notation [ECM13]

5 Implementation and Results

```
"geometry": [ {  
  "material": "2425518",  
  "type": "geometry",  
  "coreId": "66064",  
  "primitive": "triangles",  
  "positions": [  
    27.204308, 15.840237, 3.0, 27.204308, 15.840237, 3.99,  
    27.204308, 15.850237, 3.99, 27.204308, 15.850237, 3.0,  
    27.204308, 15.840237, 3.0, 27.204308, 15.850237, 3.99, ...  
  ]  
}
```

Figure 5.2: A part of a JSON string defining triangulated geometry as it is received from *BIMserver*. The compact syntax allows to define objects with key-value pairs, and arrays.

5.1.2 JTS Topology Suite

The algorithm uses Boolean operations (intersection, union, difference) in 2D space. For these operations, the *JTS Topology Suite* is used, a native Java library for 2D geometric operations [Dav07]. JTS claims to provide robust algorithms, that is, their design includes handling round-off errors. However, since it does not apply the exact computation paradigm as described in Section 3.3, caution is required regarding the input data for predicates and operations to get the desired results.

JTS provides several measures that help to avoid robustness issues like dimensional collapse, nearly coincident lines, or other problems due to round-off errors. Firstly, it allows developers to define a “precision model” for each geometry. With a “fixed” precision model, one can define a discrete grid of arbitrary size, so that all coordinates lie on the intersections of this grid. Input coordinates have to be manually reduced to the chosen precision with the *GeometryPrecisionReducer*. I had good results with a grid size of 0.1 mm, which is generally enough to represent the accuracy of architectural models.

Reducing the precision alone does not necessarily provide better robustness. Another useful technique is geometry snapping. JTS provides methods to snap vertices and segments of one geometry to another with a given snap distance tolerance. Snapping can eliminate nearly-coincident edges and thus avoid problems with geometric operations. After experiments I chose a tolerance of 0.2 mm. When a robustness-related error (*TopologyException*) occurs, another suggested step is to perturb geometries by translating or rotating them slightly [Dav12]. With fixed precision and snapping, I was able to avoid such errors.

5 Implementation and Results

```
# IfcWallStandardCase Basiswand:Ziegel 12:1157002
g BOUNDARYPART_ELEMENT_11403888_SPACE_21824118_SPACE_22020726
v 0.0 8.0 -0.15
v 12.0 8.0 -0.15
v 12.0 0.0 -0.15
v 0.0 0.0 -0.15
v 0.0 8.0 -0.15
vn 0.0 0.0 -1.0
f -1//-1 -2//-1 -3//-1 -4//-1 -5//-1
```

Figure 5.3: Definition of a simple, rectangular boundary part in *Wavefront OBJ* format. This is a group (*g*) with one face (*f*), consisting of five vertices (*v*, first and last are equal to close the polygon) and a normalvector (*vn*). Lines starting with # are comments.

5.1.3 Guava Library

Guava is an open source Java library developed by *Google*. It includes many classes that improve the ones of the standard *Java Class Library* and utilities that make a programmer's job easier. I use it for preconditions, comparators, iterators, etc. Most mentionable, I use the *SortedSetMultimap* [Goo14b] for the geometry map with the normal vectors as keys, polygons sorted by their distance from the origin as values. A standard hash map with lists as values would work as well, but the *Guava* class has a cleaner interface as it allows one and the same key to have multiple values, which improves code readability.

5.1.4 3D Geometry Output

To enable constant testing during the development of the algorithm, it was important to integrate some kind of 3D output from the very beginning. This could easily be accomplished with the *Wavefront OBJ* format, a plain-text file format to store geometric objects [MR96]. Geometry objects with their polygons are written in an OBJ file so that intermediate results can instantly be checked in a 3D viewer. Every boundary part is defined as a group in OBJ, as shown in Figure 5.3, to allow single parts to be selected in the viewer. The syntax is very simple; the group, vertex, normal vector and face definitions are all that is needed for this purpose. Unfortunately, OBJ does not support holes in polygons, although, depending on the viewer, they can be *faked* by adding the holes as faces after the first face.

5 Implementation and Results

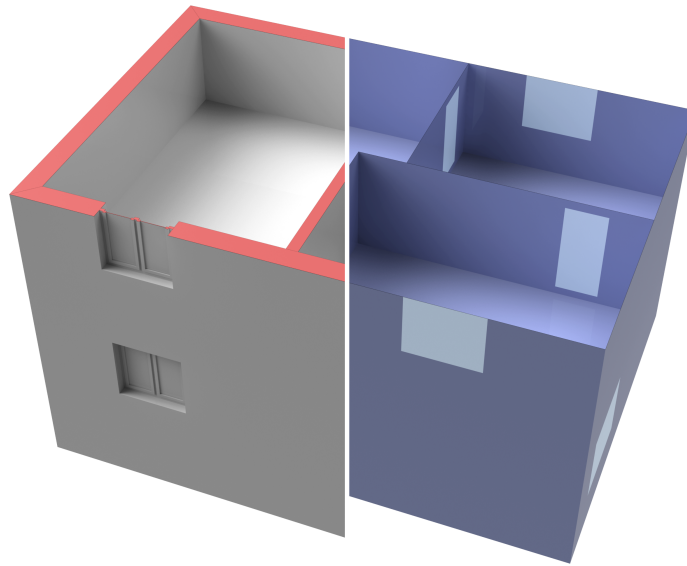


Figure 5.4: Comparison of input (left) and output (right) geometries for Model 2. The boundary parts lie on the outer edges of exterior building elements, and on the centerline of interior ones. Windows are shifted to align with their host elements. A cross-section is cut near the top to make the inside visible.

5.2 Results

The prototype has been tested with the five reference models presented in Section 4.1, and the results have been compared to the respective *Sketchup* models. Apart from some problem cases that are not handled at this stage of the project, and thus not part of this thesis, the implementation works robustly, and the results are satisfactory. The rendering in Figure 5.4 compares the input and output geometries of Model 2. One can see how the boundary parts lie on the outer edges of exterior building elements, and on the centerline of interior ones. Figure 5.5 illustrates how this end result emerges in the three processing steps of the geometric simplification algorithm: the first steps treats each space as if it was the only one, in the second steps the interior boundary parts are merged and split up, and in the final steps the gaps in the corners are filled. An input/output comparison for a larger building, Model 5, is shown in Figure 5.6.

5 Implementation and Results

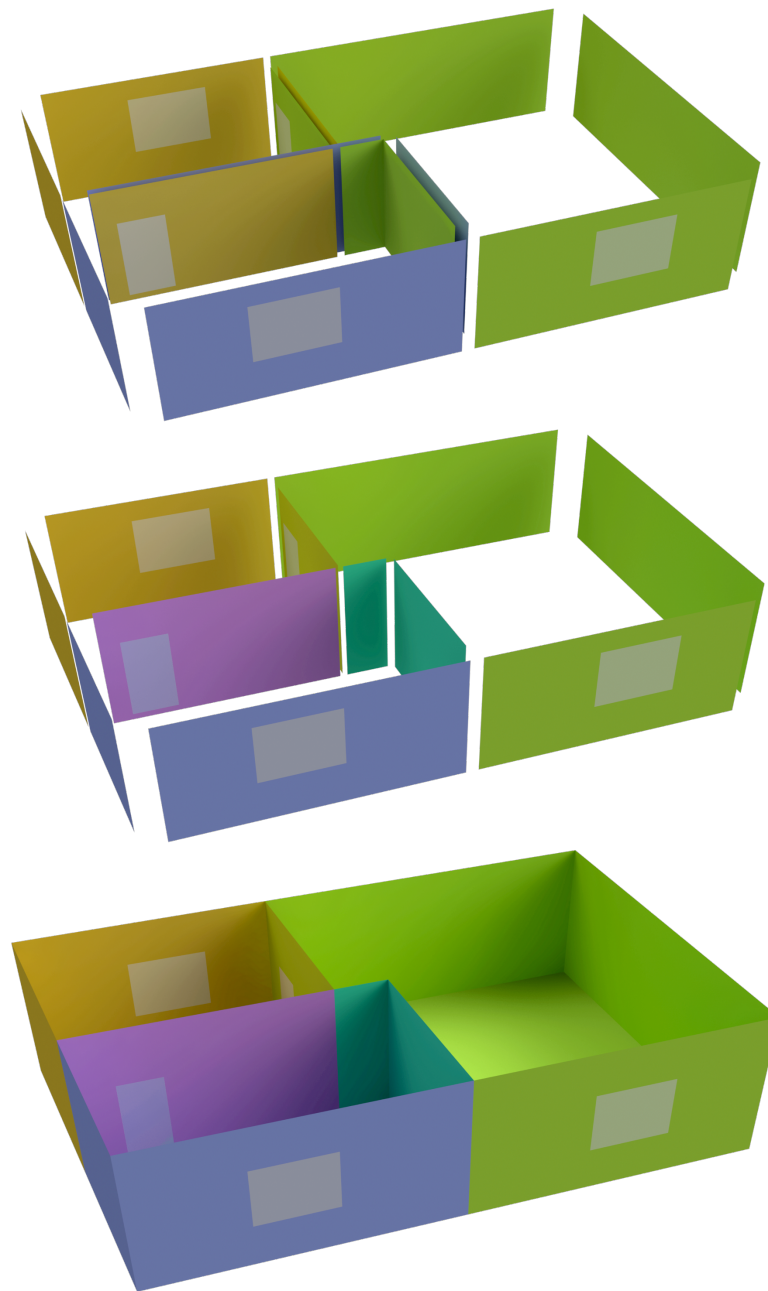


Figure 5.5: The intermediate results of each of the three processing steps by example of the upper building story of Model 2. Top: each space is treated as if it was the only one; hence, all boundary parts are “exterior” and belong to only one space. Middle: the original boundary parts are merged and split up depending on the neighboring spaces. Bottom: gaps in the corners are filled.

5 Implementation and Results

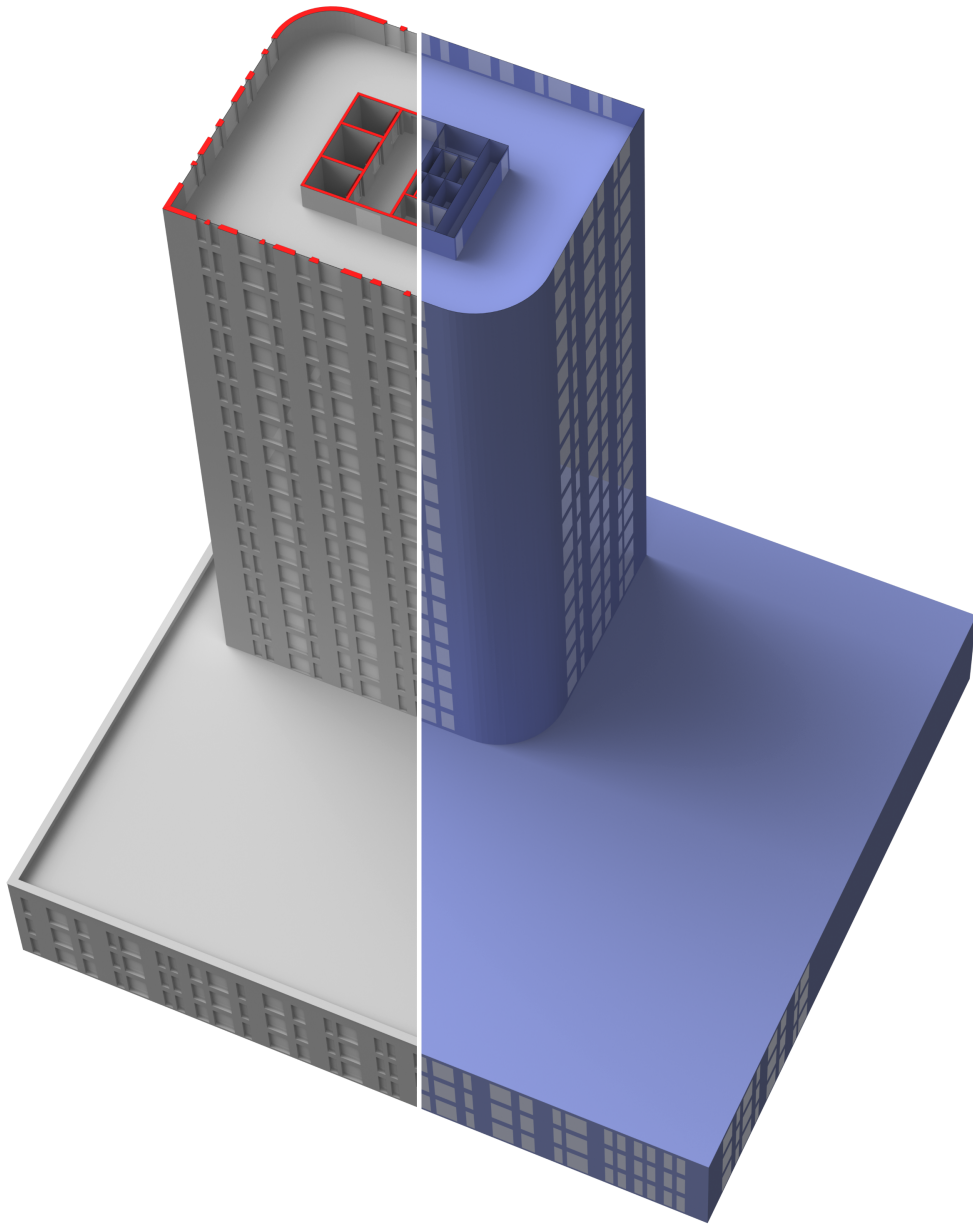


Figure 5.6: Comparison of input (left) and output (right) geometries for Model 2. The boundary parts lie on the outer edges of exterior building elements, and on the centerline of interior ones. Windows are shifted to align with their host elements. A cross-section is cut near the top to make the inside visible.

5.2.1 Performance

The running times are acceptable even for larger models. The first processing steps (finding the initial boundaries, as well as merging and splitting them), including the respective 2D operations, already have a very good performance, while the assembling of the data structure and the geometric representation, as well as the filling algorithm, could benefit from optimizations. The following table shows the total running times (including retrieving the models from *BIMserver*) on an Intel Core i7 at 3.6 GHz.

Model	Spaces	IFC File Size	Running Time
Model 1: Simple Bungalow	1	210 KB	4 s
Model 2: Simple Two-Story Home	4	460 KB	6 s
Model 3: Twin Home	34	7 MB	20 s
Model 4: Office Building	128	12 MB	35 s
Model 5: High-Rise Office Building	335	20 MB	104 s

5.2.2 Problem Cases

Although the prototype shows that the algorithm works fine for most models, even if they are incomplete to some degree, a few unresolved issues remain. Here, these problem are described, example cases from the reference models are given, and possible solutions are suggested. Some of the problems directly are caused by incorrect models, others expose limitations of the current state of the algorithm.

Problem A: Incorrect Spatial Structure and Multi-Story Spaces Many buildings have spaces that span over multiple building stories, e.g., atriums or, more common, elevator shafts. The correct way to model such spaces is to split them up into one “sub-space” per story [Theo7]. IFC supports this with the *IfcSpatialStructureElement CompositionType* [Lie+12]. When multi-story spaces are modeled as ordinary single spaces, as in Model 5 (see Figure 5.7), the optional story detection, which groups spaces into stories based on their geometries, fails. In that case, the stories have to be retrieved from the IFC spatial structure. Unfortunately, this data is often incomplete and/or incorrect (e.g., building elements and spaces arbitrarily associated with different *IfcBuildingStorey* objects). To be able to proceed in such a situation, the algorithm would have to detect multi-story spaces heuristically, e.g., on the basis of its height relative to the other spaces. A different case of incorrect spatial structure

5 Implementation and Results

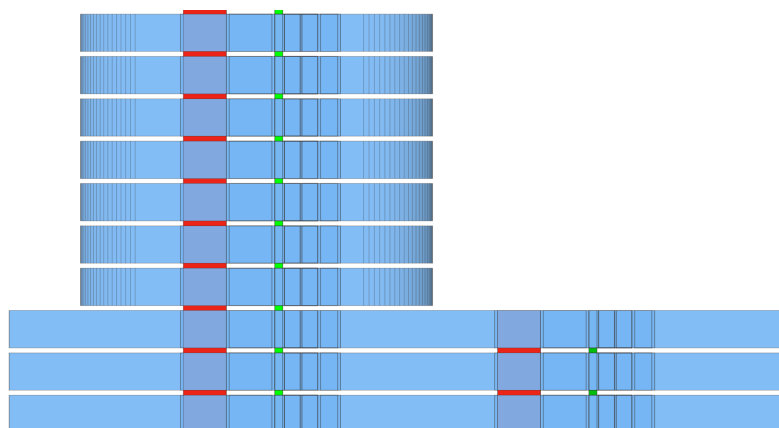


Figure 5.7: Problem A occurring in Model 4. The image shows the spaces viewed from the side. The continuous red and green spaces are elevator and utility shafts. They make story detection based on the space geometries more challenging.

was found in another real-world model, which is not part of the reference model set. There, three buildings of the same project are modeled as one single *IfcBuilding* object. The stories are poorly structured as well. To make things worse, the three buildings differ in elevation due to a sloping location, so that the space geometries overlap and do not leave constant gaps across the model. To solve this, the single buildings would have to be detected based on the geometries.

Problem B: Exterior Spaces Another space-related problem case occurs in Model 3, where exterior sections of the site (e.g., carports, lawn areas) are modeled as *IfcSpace* (see Figure 5.8). This would not be problematic, if these spaces were correctly marked as *External* in the *IfcInternalOrExternalEnum*. Since they are modeled as internal, the algorithm cannot easily exclude them. Again, some heuristic might help, e.g. excluding spaces that are not enclosed by building elements. However, since it is only a matter of changing an attribute value for some objects, such models are easy to fix manually, even with a text editor.

Problem C: Multiple Building Elements In its current version, the algorithm assumes only one building element between a space face and the outer boundary. That is, if there are multiple consecutive building elements between two spaces, or between a space and the outside air, the algorithm wrongly places the boundary

5 Implementation and Results

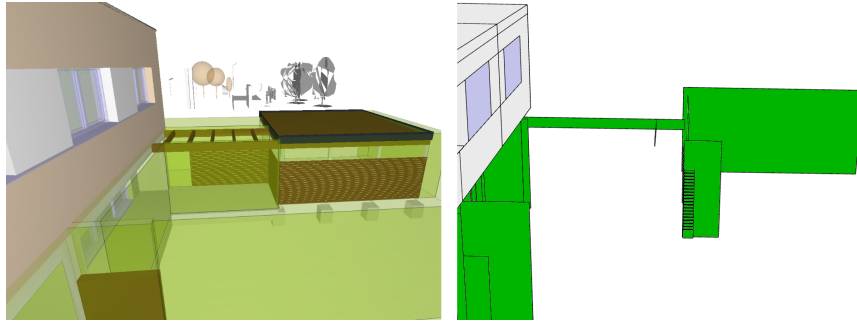


Figure 5.8: Input (left) and output (right) showing Problem B occurring in Model 3: outer areas of the site are modeled as spaces which are incorrectly marked as interior. The carport and outer walls should be ignored.

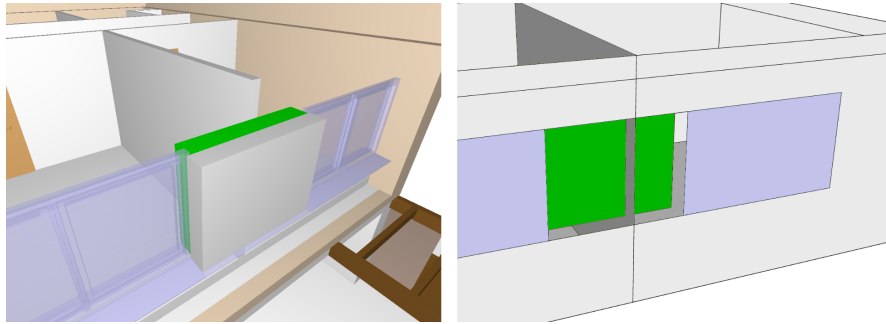


Figure 5.9: Input (left) and output (right) showing Problem C occurring in Model 3: two separate walls are behind each other. Only the innermost element is considered.

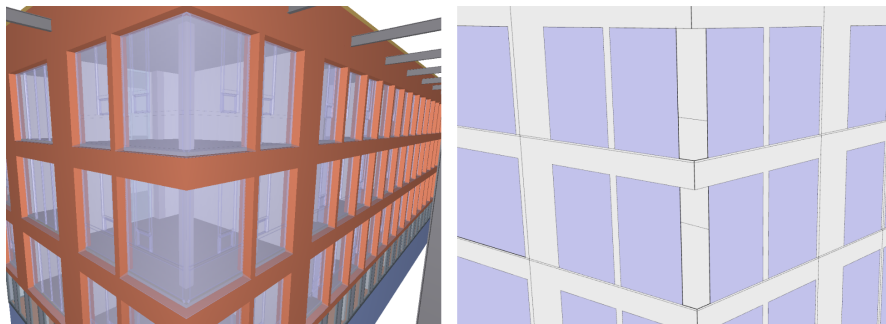


Figure 5.10: Input (left) and output (right) showing Problem D occurring in Model 4: gaps caused by windows that meet in corners are not filled.

5 Implementation and Results

part on the outer edge of the innermost element. Such situations as in Model 3 (see Figure 5.9) are not very common, as usually, walls and slabs are modeled as single objects with multiple material layers. Columns that touch walls, which would lead to the same problem, are probably more prevalent. In any case, this is a limitation that has to be fixed. The algorithm has to keep on searching for building element polygons “behind” already detected polygons. Also, it will be necessary to add support for multiple building elements in boundary parts, and/or to further split up boundary parts depending on how these building elements are arranged. What exactly the output should be needs to be discussed in the course of the project.

Problem D: Corner Windows In some buildings, windows are placed directly in corners, without another building element in between two windows. Figure 5.10 shows such a case in Model 4. Currently, the gaps in the corner between the two windows are not filled, because windows and doors are exempt from the filling process. In this case, it would be correct to increase the widths of the windows so that the gap is filled, as it is done with walls. However, there might be cases where there is a column in the corner that would need to be considered. Remember that the algorithm does not consider building elements that are adjacent to the space faces, only those that are directly in front of it. When filling the gaps, it is assumed that they are occupied by the adjacent building elements, not a third one. Whether or not this should be changed needs to be discussed within the project consortium.

Problem E: Clashing Geometries A surprisingly frequent problem in many models is geometry clashes, i.e., areas that are occupied by multiple building elements. This can lead to non-deterministic behavior of the algorithm, as clashing polygons of different building elements might be randomly detected as boundaries. As many clashes occur in corner areas (e.g., clashes between walls and a slab, or between two orthogonal walls), the gap filling approach actually helps to avoid resulting issues. Other than that, there is not much that the algorithm can do, because it cannot decide which of two clashing polygons to prefer. Detecting clashing polygons and show a warning is an option.

6 Discussion

After examining the theoretical foundations and related scientific work, I proposed a solution that aims to build a bridge between building information modeling and energy performance analysis, which have both gained momentum in recent years, but do not yet benefit from each other on a broad basis. The presented algorithm makes it possible to use models present in the open standard IFC as input for energy analysis. It encodes the knowledge of an expert as well as norms and guidelines in order to simplify the geometry and extract other relevant data.

The tests of the implemented prototype against the reference model set confirmed what had been suggested by the literature: data quality in BIM and IFC is a major issue, and one cannot expect perfect models. Concepts that sound nice in the IFC specification sometimes turn out to be poorly implemented in real-world models. Incomplete or incorrect spatial structures (breaking down the logical makeup of a project with sites, buildings, stories, and spaces) and space boundaries (defining where spaces and building elements meet), inconsistent modeling of layered elements, clashing geometries, as well as missing attributes are typical weak points. Such flaws exist for two main reasons. The first, poor implementation of IFC export by BIM authoring software due to its wide and deep scope, is addressed by *buildingSMART* with Model View Definition specifications and corresponding software certification. The second reason is probably harder to overcome: BIMs are often created without future use cases other than generation of plans and visualizations in mind, which, strictly speaking, makes them “non-BIMs”.

In the previous chapter, several problem cases have been documented. Even though some of them are directly related to modeling mistakes, it is possible to find workarounds. This would make the solution suitable for a much larger range of models. Certain degrees of incompleteness are already tolerated by the algorithm, for example missing building stories. However, the experience with the test models suggests that a fully automatic solution, that is, extracting the energy analysis data from IFC without any user interference, is not generally feasible. Instead, a

6 Discussion

semi-automatic approach is very promising. The GINGER core can be made a server component coupled to a *BIMserver* instance as well as a web-based user interface. The user interface could consist of a model tree and a 3D viewer. It should be possible to view both input and output geometry in the same viewer to enable direct comparisons. This could be achieved with the aforementioned *BIMsurfer*, a WebGL viewer component for *BIMserver*. Ideally, the user should be given the possibility to select certain areas in the 3D viewer and thereby give hints to the algorithm about semantic groups that are missing in the IFC model, or exempt certain parts. Additionally, heuristic analysis of certain aspects could be triggered via this interface. Such user interaction will not only make solving some of the problems easier, but will also increase the user's confidence in the results due to the instant feedback and comparison.

The contribution of this thesis is that building elements and spaces can be transformed into a novel representation that focuses on the boundaries constituting the thermal building envelope. Both exterior and interior boundary parts are represented as single surfaces that have no more than one space on each side, and are associated with a homogeneous set of material layers. This representation complies with the requirements for energy performance analysis: it is suitable for zoning as well as for calculating heat flows, and corresponds to the gross floor space of the building. The algorithm can be integrated into a semi-automatic solution for preparing input data for such calculations, while the common practice today is to manually remodel buildings just for the purpose of a single energy performance calculation. As a benefit, significant time and cost savings can be achieved, not only for the issuance of energy performance certificates, but also to enable iterative energy analysis as part of the building design process.

Bibliography

- [AM13] B. Abbasnejad and H. I. Moud. "BIM and Basic Challenges Associated with its Definitions, Interpretations and Expectations." In: *International Journal of Engineering Research and Applications* 3 (2013) (cit. on pp. 5, 6).
- [ANU05] A-NULL Bauphysik. *APS XML Schema Definition*. 2005. URL: <http://www.archiphysik.at/schema/2005/docs/aps.xsd.html> (cit. on p. 45).
- [ANU14] A-NULL Bauphysik. *ArchiPHYSIK*. <http://www.archiphysik.at>. Apr. 2014 (cit. on pp. 18, 28).
- [Arc11] Arch-Vision. *Q4 2011 European Architectural Barometer*. 2011 (cit. on p. 5).
- [Aus11] Austrian Standards Institute. *ÖNORM B 8110-6: Thermal insulation in building construction - Part 6: Principles and verification methods - Heating demand and cooling demand*. 2011 (cit. on pp. 2, 15, 16, 36).
- [Aut14a] Autodesk. *Green Building Studio*. 2014. URL: <http://www.autodesk.com/products/green-building-studio/overview> (cit. on p. 18).
- [Aut14b] Autodesk. *Revit*. 2014. URL: <http://www.autodesk.com/products/revit-family/overview> (cit. on p. 6).
- [Baz10] V. Bazjanac. *Space boundary requirements for modeling of building geometry for energy and other performance simulation*. Tech. rep. Lawrence Berkeley National Laboratory, 2010 (cit. on pp. 21, 22).
- [Bee+11] J. Beetz, L. van Berlo, R. de Laat, and P. Bonsma. "Advances in the development and application of an open source model server for building information." In: *Proceedings of the CIP W78 conference*. 2011 (cit. on p. 47).
- [Ben14] Bentley Systems. *MicroStation*. 2014. URL: <http://www.bentley.com/en-US/Products/MicroStation/> (cit. on p. 6).
- [BIM14] BIMsurfer. *BIMsurfer: Open source WebGL viewer for IFC models*. 2014. URL: <http://bimsurfer.org/> (cit. on p. 48).

Bibliography

- [BLI13] Bio Intelligence Service, R. Lyons, and IEEP. *Energy performance certificates in building and their impact on transaction prices and rents in selected EU countries*. Final report prepared for European Commission (DG Energy). 2013 (cit. on p. 15).
- [BLL04] J. Benner, K. Leinemann, and A. Ludwig. "Übertragung von Geometrie und Semantik aus IFC-Gebäudemodellen in 3D-Stadtmodelle." In: *Proc. CORP* (2004), pp. 573–578 (cit. on p. 5).
- [bui14a] buildingSMART. *Model View Definition Summary*. 2014. URL: <http://www.buildingsmart-tech.org/specifications/ifc-view-definition> (cit. on p. 12).
- [bui14b] buildingSMART. *Participants of the official buildingSMART IFC2x3 Coordination View V2.0 certification process*. 2014. URL: <http://www.buildingsmart-tech.org/certification/ifc-certification-2.0/ifc2x3-cv-v2.0-certification/participants> (cit. on p. 12).
- [Caro3] W. Carlson. *A Critical History of Computer Graphics and Animation*. Ohio State University. 2003. URL: <http://design.osu.edu/carlson/history/lesson10.html> (cit. on p. 4).
- [Cat14] Catenda. *bimsync*. 2014. URL: <https://bimsync.com/> (cit. on p. 48).
- [CHR13] A. Cemesova, C. Hopfe, and Y. Rezgui. "An approach to facilitating data exchange between BIM environments and a low energy design tool." In: *Proceedings of BS2013: 13th Conference of International Building Performance Simulation Association*. 2013, p. 8 (cit. on p. 20).
- [Con13] Concerted Action EPBD. *Implementing the Energy Performance of Buildings Directive - Featuring Country Reports 2012*. Ed. by Portuguese Energy Agency (ADENE). 2013 (cit. on p. 17).
- [Con14] Concerted Action EPBD. *Concerted Action Energy Performance of Buildings*. 2014. URL: <http://www.epbd-ca.eu/> (cit. on pp. 1, 15).
- [Dav07] M. Davis. "Secrets of the JTS Topology Suite." In: (2007) (cit. on p. 49).
- [Dav12] M. Davis. *JTS Frequently Asked Questions*. 2012. URL: <http://tsusiatsoftware.net/jts/jts-faq/jts-faq.html> (cit. on p. 49).
- [Eas+11] C. Eastman, P. Teicholz, R. Sacks, and K. Liston. *BIM Handbook*. 2nd ed. Wiley, 2011 (cit. on pp. 6, 12).
- [ECM13] ECMA International. *The JSON Data Interchange Format*. 2013. URL: <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf> (cit. on p. 48).

Bibliography

- [EDS14] EDSL. *Tas - a complete solution for the thermal simulation of new or existing buildings*. 2014 (cit. on p. 18).
- [Erh+07] H. Erhorn, J. de Boer, S. Wössner, K. Höttges, and H. Erhorn-Kluttig. "DIN V 18599: The German holistic energy performance calculation method for the implementation of the EPBD." In: *2nd PALENC Conference of the 28th AIVC Conference of Building Low Energy Cooling and Advanced Ventilation Technology in the 21st Century, Crete Island, Greece*. 2007, pp. 319–321 (cit. on p. 18).
- [ET10] European Parliament and The Council of the European Union. *Directive 2010/31/EU of the European Parliament and of the Council of 19 May 2010 on the energy performance of buildings*. 2010 (cit. on pp. 1, 15).
- [Fra11] E. Franconi. "Introducing a framework for advancing building energy modelling methods & processes." In: *Proceedings of Building Simulation 2011: 12th Conference of the International Building Performance Simulation Association*. 2011, K8–K15 (cit. on p. 20).
- [GGG12] M. Gratzl-Michlmair, C. Graf, and A. Goerth. "Vergleichsberechnung eines Energieausweises nach deutschen und österreichischen Algorithmen." In: *Bauphysik* 34.6 (2012), pp. 286–291 (cit. on p. 18).
- [Goo14a] Google. *Gson – a Java library to convert JSON to Java objects and vice-versa*. 2014. URL: <https://code.google.com/p/google-gson/> (cit. on p. 48).
- [Goo14b] Google. *Guava Multimap*. 2014. URL: <https://code.google.com/p/guava-libraries/wiki/NewCollectionTypesExplained#Multimap> (cit. on p. 50).
- [Gra14a] Graphisoft. *ArchiCAD*. 2014. URL: <http://www.graphisoft.com/archicad/> (cit. on pp. 6, 23).
- [Gra14b] Graphisoft. *EcoDesigner STAR*. 2014. URL: http://www.graphisoft.com/archicad/ecodesigner_star/ (cit. on p. 18).
- [Hen+10] D. Henckel, K. von Kuczkowski, P. Lau, E. Pahl-Weber, and F. Stellmacher. *Planen – Bauen – Umwelt*. VS Verlag für Sozialwissenschaften GmbH, 2010. ISBN: 9783531922881. URL: <http://books.google.at/books?id=NdGMeounOBUC> (cit. on p. 5).
- [HL10] K.-H. Häfele and T. Liebich. *IFC Implementation Agreement Space Boundary*. Tech. rep. buildingSMART, 2010 (cit. on pp. 22, 23).

Bibliography

- [Hof89] C. Hoffmann. "The problems of accuracy and robustness in geometric computation." In: *Computer* 22.3 (Mar. 1989), pp. 31–39. ISSN: 0018-9162. DOI: [10.1109/2.16223](https://doi.org/10.1109/2.16223) (cit. on p. 26).
- [HW11] R. J. Hitchcock and J. Wong. "Transforming IFC Architectural View BIMs for Energy Simulation: 2011." In: *Proceedings of Building Simulation*. 2011 (cit. on p. 20).
- [IES14] IES. VE. 2014. URL: <http://www.iesve.com/software> (cit. on p. 18).
- [KSU14] U. Krispel, C. Schinko, and T. Ullrich. "The Rules Behind – Tutorial on Generative Modeling." In: *Proceedings of Symposium on Geometry Processing / Graduate School* 12 (2014), 2:1–2:49 (cit. on p. 6).
- [KUF14] U. Krispel, T. Ullrich, and D. W. Fellner. "Fast and Exact Plane-Based Representation for Polygonal Meshes." In: *Proceeding of the International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing* 8 (2014), pp. 189–196 (cit. on p. 27).
- [Lie+12] T. Liebich, Y. Adachi, J. Forester, J. Hyvarinen, K. Karstila, K. Reed, S. Richter, and J. Wix. *Industry Foundation Classes IFC2x Edition 3 Technical Corrigendum 1*. 2012. URL: <http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/> (cit. on pp. 7, 11, 54).
- [MGD13] P. Manke, Y. K. Garg, and V. M. Das. "Energy simulation tools and CAD interoperability: A critical review." In: *Energy Efficient Technologies for Sustainability (ICEETS), 2013 International Conference on*. IEEE. 2013, pp. 191–196 (cit. on p. 20).
- [Mit11] J. Mitchell. "BIM & building simulation." In: *Proceedings of Building Simulation 2011: 12th Conference of the International Building Performance Simulation Association*. 2011, K1–K7 (cit. on p. 20).
- [MR96] J. D. Murray and W. van Ryper. *Encyclopedia of graphics file formats*. Vol. 1. 1996 (cit. on p. 50).
- [Mul94] K. Mulmuley. *Computational geometry: An introduction through randomized algorithms*. Vol. 54. Prentice-Hall Englewood Cliffs, NJ, 1994 (cit. on p. 25).
- [Nat14] National Institute of Building Sciences. *Frequently Asked Questions About the National BIM Standard*. 2014. URL: <http://www.nationalbimstandard.org/faq.php#faq1> (cit. on p. 5).

Bibliography

- [Neu+10] H. Neuroth, A. Oßwald, R. Scheffel, S. Strathmann, and K. Huth. *nestor Handbuch: Eine kleine Enzyklopädie der digitalen Langzeitarchivierung*. 2010 (cit. on p. 5).
- [ODo+13] J. T. O’Donnell, T. Maile, N. Cody Rose, E. M. Mrazović, C. Regnier, K. Parrish, and V. Bazjanac. “Transforming BIM to BEM: Generation of Building Geometry for the NASA Ames Sustainability Base BIM.” In: (2013) (cit. on p. 23).
- [pas13] passipedia.org. *Calculating energy efficiency*. [Online; accessed 7-August-2014]. 2013. URL: http://passipedia.passiv.de/passipedia_en/doku.php?id=planning:calculating_energy_efficiency&rev=1382708040 (cit. on p. 17).
- [Prao1] M. J. Pratt. “Introduction to ISO 10303—the STEP standard for product data exchange.” In: *Journal of Computing and Information Science in Engineering* 1.1 (2001), pp. 102–103 (cit. on p. 7).
- [PSM10] U. Pont, B. Sommer, and A. Mahdavi. “Ein Vergleich der Ergebnisse von stationärer und instationärer Berechnung thermischer Energiekennzahlen anhand bestehender Objekte in Wien.” In: *Building Performance Simulation in a Changing Environment*. 2010, pp. 515–522 (cit. on pp. 1, 17, 18).
- [RB13] C. M. Rose and V. Bazjanac. “An algorithm to generate space boundaries for building energy simulation.” In: *Engineering with Computers* (2013), pp. 1–10 (cit. on pp. 23, 25, 26).
- [RS12] E. M. Ryan and T. F. Sanquist. “Validation of building energy modeling tools under idealized and realistic conditions.” In: *Energy and buildings* 47 (2012), pp. 375–382 (cit. on p. 19).
- [Sch97] S. Schirra. “Precision and robustness in geometric computations.” English. In: *Algorithmic Foundations of Geographic Information Systems*. Ed. by M. van Kreveld, J. Nievergelt, T. Roos, and P. Widmayer. Vol. 1340. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1997, pp. 255–287. ISBN: 978-3-540-63818-6. DOI: 10.1007/3-540-63818-0_9. URL: http://dx.doi.org/10.1007/3-540-63818-0_9 (cit. on p. 26).
- [Set13] V. Settgast. “Processing Semantically Enriched Content for Interactive 3D Visualizations.” PhD thesis. Graz University of Technology, 2013 (cit. on p. 14).

Bibliography

- [Shao4] N. Shaikh. *Impact of code checking usage on IFC model, CAD, and end users: the challenge of imperfect data*. 2004. URL: http://www.buildingsmart.org/publications/viu-presentations/singapore-viu-workshop/t_20of_20the_20demand_20of_20code_20checking.pdf (cit. on p. 11).
- [Sol14] Solibri. *Solibri Model Checker*. 2014. URL: <http://www.solibri.com/products/solibri-model-checker/> (cit. on p. 12).
- [Theo7] The National 3D-4D-BIM Program. *GSA BIM Guide for Spatial Program Validation*. Tech. rep. U.S. General Services Administration, 2007 (cit. on p. 54).
- [Tri14] Trimble Navigation. *SketchUp*. 2014. URL: <http://www.sketchup.com/> (cit. on p. 28).
- [Tup+11] K. Tupper, E. Franconi, C. Chan, S. Hodgins, A. Buys, and M. Jenkins. "Building energy modeling: industry-wide issues and potential solutions." In: *Proceedings of the 12th Conference of International Building Performance Simulation Association*. 2011 (cit. on p. 20).
- [US 14] US Department of Energy. *EnergyPlus Energy Simulation Software*. 2014. URL: http://apps1.eere.energy.gov/buildings/energyplus/energyplus_about.cfm (cit. on p. 18).
- [W3Co7] W3C. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. 2007. URL: <http://www.w3.org/TR/soap12-part1/> (cit. on p. 48).
- [W3Co8] W3C. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. 2008. URL: <http://www.w3.org/TR/xml/> (cit. on p. 45).
- [W3C11] W3C. *Scalable Vector Graphics (SVG) 1.1 (Second Edition)*. 2011. URL: <http://www.w3.org/TR/SVG/> (cit. on p. 45).
- [WWNo9] A. Wong, F. K. Wong, and A. Nadeem. "Comparative roles of major stakeholders for the implementation of BIM in various countries." In: *Proceedings of the International Conference on Changing Roles: New Roles, New Challenges, Noordwijk Aan Zee, The Netherlands, 5-9 October*. Development Bureau, Government of the Hong Kong Special Administrative Region. 2009 (cit. on p. 11).
- [Yap97] C. K. Yap. "Robust geometric computation." In: *Handbook of discrete and computational geometry*. CRC Press, Inc. 1997, pp. 653–668 (cit. on p. 27).