Automatic generation of CityGML LoD3 building models from IFC models

MSc thesis in Geomatics
by Sjors Donkers

December 2013

TUDelft
Department of GIS Technology
OTB Research Institute for the Built Environment

On cover:

IfcOpenHouse IFC model

http://blog.ifcopenshell.org/2012/11/say-hi-to-ifcopenhouse.html

Colophon:

This document is created using the LaTeX Thesis Template

by Steven Gunn and Sunil Patel

http://users.ecs.soton.ac.uk/srg/softwaretools/document/templates/

http://www.sunilpatel.co.uk/thesis-template/

Delft University of Technology
Master of Science in Geomatics

Automatic generation of CityGML LoD3 building models from IFC models

by

Sjors Donkers

December 2013

Supervisors	Dr. Hugo Ledoux
	Dr. Junqiao Zhao
Graduation professor	Prof.dr. Jantien E. Stoter
Co-reader	Ir. Paul de Ruiter
External examiner	Prof.dr. Massimo Menenti

Delft University of Technology
MSc Geomatics
The Netherlands

Technische Universiteit Delft

Abstract

CityGML is a standardized data format used to store the semantic information and geometries of buildings and other object classes of 3D city models. The Level of Detail of current state of the art city models (LoD2) is not sufficient for accurate environmental simulations like noise, the solar potential of windows and other types of analyses. An LoD3 building model represents the full architectural exterior of a building with balconies, windows and so forth. The generation of these models needs to be automated as it is otherwise infeasible due to the required high amount of manual labour. In the architectural world, detailed building models are created in IFC format. This thesis shows that it is possible to automatically generate valid and semantically rich CityGML LoD3 building models directly from IFC models. Also an initial investigation is done on the possibilities for the conversion of IFC models to CityGML LoD4.

For the conversion the semantic and geometric validity requirements are determined for CityGML. A methodology for the conversion is developed and a prototype implementation is made to prove the effectiveness of the conversion. The conversion consists of three parts: 1) The extraction and mapping of IFC semantics to CityGML semantics; 2) A geometric generalization which extracts the exterior shell using a transformation based on Boolean and morphological operations; 3) Semantic and geometric refinements which optimize the model for analyses. The developed prototype is able to successfully convert IFC models to CityGML LoD3. All the resulting models were geometrically validated according to the ISO19107 standard, and semantics were checked manually. Few improper semantics occur in the output due to missing semantics in IFC. For example, there are no semantics for balconies or dormers in IFC. Recommendations are given to improve the alignment between the two formats. For IFC additional semantics are recommended whereas it is important for CityGML to specify how certain aspects are to be modelled.

The research presented in this thesis can be used as the foundation for future work on the interoperability between Architecture and Geomatics. The software package is open source and freely available at https://github.com/tudelft-gist/ifc2citygml.

Acknowledgements

More than a year ago, my supervisor Hugo Ledoux approached me beaming with enthusiasm: “I have found the perfect topic for you!” he said. How could I refuse? So I didn’t refuse and now at the end of my thesis I’m still glad I didn’t. It took a while to complete, but it was worth the effort and I have learned a lot.

I would like to thank both my supervisors, Hugo Ledoux and Junqiao Zhao, for their support and guidance and for pushing me to keep improving my work. In particular I would like to thank Junqiao Zhao for all the great discussions related to this thesis. Also, I’m very grateful for the feedback I received from Jantien Stoter and Paul de Ruiter. I found it truly valuable.

I am thankful to all people at OTB and all that are in any way related to the project. Especially Karl-Heinz Häfele and Roeland Boeters have been very helpful for providing me feedback throughout the project and for kick starting my project with their knowledge and experience. Furthermore, I would like to thank the following people for pointing me in the right direction: Jakob Beetz, Joran Jessurun, Wouter Goedhart, Lars de Vries, Thomas Krijnen, Léon van Berlo, Ruben de Laat and Liu Liu.

Also I would like to thank my friends for everything and whatever they did to contribute to the project: Jiebo Qiu, Lie Ping Huang, Moonhee Lee, Vera Liem, Yang Yue and Jelte van Oostveen. All your contributions are greatly appreciated.

Lastly I would like to say hi to my mom and dad: “Hi!”. Thank you for supporting me all the way through my education.

Contents

List of Figures

List of Tables

Acronyms

AABB Axis-Aligned Bounding Boxes.

ADE Application Domain Extension.

AEC Architecture, Engineering and Construction.

AHN Actueel Hoogtebestand Nederland.

B-rep Boundary-representation.

BGT Basisregistratie Grootschalige Topografie.

BIM Building Information Modeling.

BIMserver Building Information Modelserver.

CAD Computer-Aided Design.

CGAL Computational Geometry Algorithms Library.

CityGML City Geography Markup Language.

CSG Constructive Solid Geometry.

FME Feature Manipulation Engine.

GIS Geographic Information System.

GML Geography Markup Language.

GUI Graphical User Interface.

HalfedgeDS Halfedge Data Structure.

IDM Information Delivery Manual.

IFC Industry Foundation Classes.

IFD International Framework for Dictionaries.

IFG IFC for GIS.

IMGeo InformatieModel GEOgrafie.

ISO International Organization for Standardization.

KIT Karlsruhe Institute of Technology.

LOD Level-Of-Development.

LoD Level-of-Detail.

OFF Object File Format.

OGC Open Geospatial Consortium.

SEdge Spherical Edge.

SFace Spherical Face.

SIG3D Special Interest Group 3D.

SNC Selective Nef Complex.

STEP STandard for the Exchange of Product model data.

SVertex Spherical Vertex.

TNO (Nederlandse centrale organisatie voor) Toegepast Natuurwetenschappelijk Onderzoek.

UBM Unified Building Model.

UML Unified Modeling Language.

XML Extensible Markup Language.

Chapter 1

Introduction

This thesis is within the field of Geomatics, a field which is specialized in, but not limited to, the usage, analysis and distribution of (geo) data and information (hereafter referred to as geodata). The value of the geodata depends on a range of factors such as the availability, accuracy, amount of detail, whether the data is up-to-date and the consistency with which it is stored relative to similar geodatasets. Not only do these factors contribute to the ease and ability to analyse the geodata, but also to the interoperability between applications.

Architecture is a field in which a vast amount of high value data is created manually. By bridging the gap between the fields of Geomatics and Architecture a new data source can become available for geo-applications with an unprecedented amount of detail. In this thesis a methodology is developed for an automated conversion of standardized data formats of the respective fields. While there are other ways to get geodata like laser scanning, surveying, photogrammetric measurements or manual modelling [1], a conversion has the benefit of allowing for the reuse of already captured information without being as time consuming as rebuilding the models manually. Reusing data also assures consistency of the data used in both fields. Unsurprisingly the interest in such a conversion is increasing [2–4].

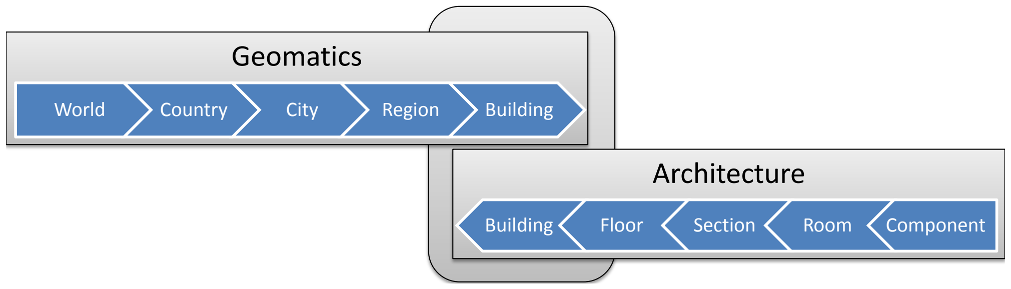


FIGURE 1.1: Overlap between the fields of Geomatics and Architecture (Adapted from: [5])

The fields share a common interest (see Figure 1.1) in 3D building models which are becoming increasingly more important in many applications [6]. Building models can amongst others be used in cadastral and environmental analysis [7] or for queries like the solar potential of a roof [8] or the number and distribution of windows [9, 10]. Two standardised data formats, City Geography Markup Language (CityGML) [11] used by the field of Geomatics and Industry Foundation Classes (IFC) [12], an open data model for Building Information Modeling (BIM), by Architecture form the two most prominent semantic 3D modelling formats for buildings [1].

IFC has been accepted as an open standard by the Dutch ‘Forum en College Standaardisatie’ [13]. This means that governmental organisations are required to use the IFC data format when working with BIM. As a result, a significant rise in the availability can be expected for IFC models. CityGML on the other hand is the standard used for modelling the 2D Basisregistratie Grootschalige Topografie (BGT) as to facilitate a future transition to 3D [14].

IFC is more detailed. It models everything between the individual components up to buildings, while in CityGML, five Levels-of-Detail (LoDs) are defined for building models where each level describes what geometric and semantic representations are expected (see Figure 1.2). LoD0 is a 2.5D digital terrain model (DTM). The first level having volumetric buildings is LoD1, in which they are represented by extruded blocks. In LoD2 roof structures and larger building installations like balconies are present, but they are still generalised geometries. LoD3 contains the full exterior of an architectural model with detailed wall and roof structures, doors and windows. LoD3 is extended in LoD4 by adding interior structures like rooms, stairs and furniture [15].

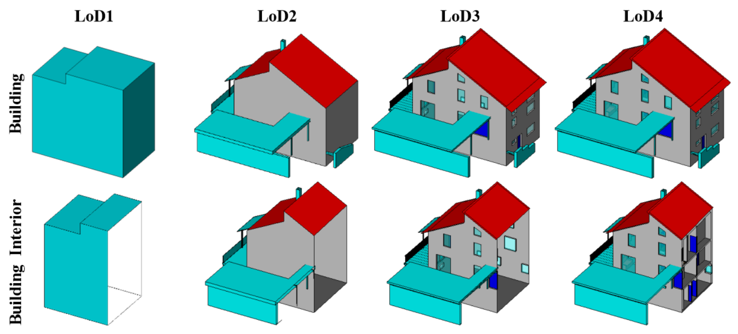


FIGURE 1.2: Building model in LoD1 to LoD4 (Source: [16])

According to Blaauboer et al. [17], the generation of lower LoD building models can be automated, but LoD3 as well as LoD4 models are usually generated by the conversion from IFC . However, all current developments on this conversion have certain defects as is elaborated in Chapter 2. In general, the converters do not transform the geometry and the semantic mapping is limited. Furthermore, other projects focus on the conversion of IFC to lower CityGML LoDs. Thus the contribution made by defining a valid and openly accessible conversion method for LoD3 and higher would be significantly more valuable. As a result, the goal of this research project is to generate valid and semantically rich CityGML geometry at LoD3 from IFC building models and exploring the possibilities for generating LoD4.

The conversion to CityGML involves geometric calculations as well as mapping of semantics, as is depicted in Figure 1.3. IFC models are built using mostly primitives and swept solids in combination with Constructive Solid Geometry (CSG) [18] while CityGML only uses Boundary-representation (B-rep). LoD3 building models represent the full architectural exterior, thereby the geometric accuracy concerning the exterior of a building is equivalent between the two formats. This is especially so since details like the bricks in a wall and other materials are stored as properties in IFC instead of geometry. However, installations like beams and columns are generalised in CityGML LoD3. There are several requirements on IFC input models for the conversion, these are discussed in Section 4.4.

Aside from how objects are modelled, also what is modelled is different. For generating an LoD3 building model, extracting the exterior envelope from the IFC geometry is an essential geometric calculation. A valid exterior shell defines a clear separation between out- and inside. The validity is necessary to guarantee the correct output for processing or manipulation operations, like the calculation of the volume, creation of a buffer or operations such as intersection, touch and contain [19].

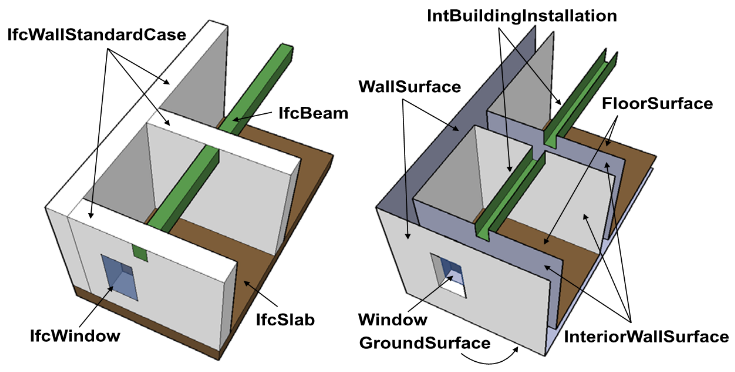


Figure 1.3: Differences between IFC (left) and CityGML (right). In IFC solid objects are represented, while CityGML represents only visible surfaces. The faces of an IFC object can have multiple different semantics in CityGML (Source: [20])

Contributions of this thesis through the implementation of algorithms for the automatic conversion of IFC to CityGML are:

1. A method for constructing LoD3 CityGML building models from detailed IFC building models becomes available.
    - The new source for CityGML models allows for an increase of the availability of building models and the ability to keep them up-to date.
    - At LoD3, models with a higher accuracy and amount of detail are provided than the state of the art LoD2 models.
    - The models adhere to the CityGML standard and are to a large extent geometrically consistent with the IFC models.
    - It will improve the interoperability between Geomatics and Architecture and the reuse of information, thereby reducing the high costs for the generation of 3D city models [21].
2. Suggestions for improvement, derived from the conversion practise, are given on the modelling methodology and standards of IFC and CityGML. Also advice for conversions to other LoDs is provided, especially for LoD4.

1.1 Research Questions

This thesis aims to develop a methodology for the conversion of architectural building models modelled in IFC to CityGML LoD3. Three important aspects are the semantics, the geometry and the validity of the output models. It is possible to model a wide and diverse range of building models in IFC. As such the methods should be flexible enough to deal with complex models.
The main research question is:

> **Is it possible to generate valid and semantically rich CityGML building models at LoD3 from IFC models, and can this method be extended to LoD4?**

Sub questions have been defined to help answer the main research question:

1. What is a valid CityGML LoD3 geometry and which semantics are needed?
2. How can IFC models be converted to CityGML LoD3?
    a) What geometric operations are required for both the geometric and the semantic conversion?
    b) What semantic information is required for both the geometric and the semantic conversion?

3. How can the conversion be extended to LoD4?

A conversion method, presented in Chapter 3, is defined and implemented into a prototype. The prototype is tested and its output is validated with the goal of answering the research questions. The implementation and validation of the prototype are described in Chapter 4.

The conversion methodology and the prototype are built upon three pillars, semantic mapping, geometric transformation and the geometric and semantic refinement (see Figure 1.4). In the first stage the useful geometries and semantics from the IFC file are filtered from the others and the CityGML semantics are determined. The second stage uses the filtered geometries and transforms them into CityGML compliant geometries representing the exterior shell. The last stage optimizes the geometry for analytical computations and resolves semantic issues before the model is written to the output CityGML file. Although LoD3 is the main target, LoD4 rooms are researched as an initial investigation of the possibilities and problems of the conversion to LoD4.

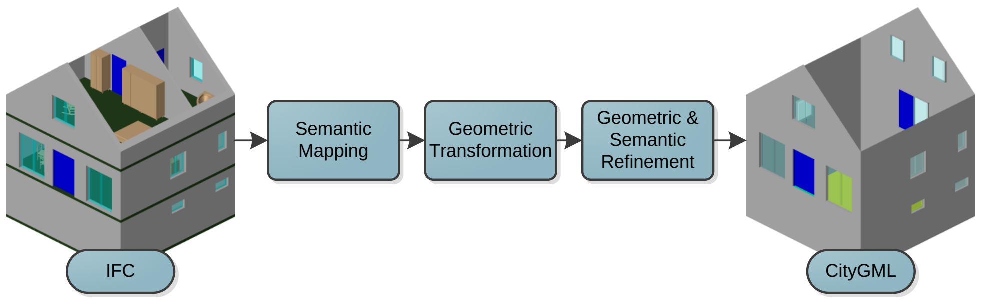


FIGURE 1.4: General workflow diagram of the prototype

Models using the IFC2x3 release version are used as input since at the start of this project it was the latest final release and IFC4 support is still very limited. The output of the proposed methods fully adheres to the standards imposed by CityGML 2.0.0 and International Organization for Standardization (ISO)19107. The evaluation of results is made with these four factors:

1. The syntax must be correct.

2. The geometry must be in line with the CityGML standard and valid according to definition of the ISO/Open Geospatial Consortium (OGC).

3. The geometry should represent the input model with minimal differences. Differences are considered significant when the shortest distance from geometry in the output to the input is larger than 0.01 m.

4. The semantics should be applicable and correctly mapped.

1.2 Scope

The methods which are developed during this project are designed to facilitate the use of IFC data in 3D Geographic Information System (GIS) applications. The focus is not input-driven by converting IFC, thus attempting to store all its information into a CityGML model, but output-driven by constructing a proper CityGML model using only the information that is needed for a successful conversion that keeps as much as possible of the input information relevant for the CityGML output. An implementation of a both geometrically and semantically input-driven conversion is presented by Laat and Berlo [3].

El-Mekawy et al. [22] state that for the semantic mapping, CityGML needs to be overloaded with additional information to be matched and integrated with IFC. An example is CityGML not supporting classes for storeys or openings that do not contain doors or windows [1]. However the conversions envisioned in those cases are input-driven and aim at a full bidirectional integration of the two formats. No proper geometric conversion nor a complete semantic mapping previously been published for LoD3 with an output-driven focus.

LoD3 requires the exterior shell geometry and the CityGML semantics related to the objects which are part of that exterior. That is, geometric relations like 'Roof', 'Window' and 'BuildingInstallation' will be linked to the geometry, but textures and properties like for example the thermal transmittance of an object will not be considered. By making no more use of semantic IFC properties than strictly necessary for extracting information that is relevant for CityGML LoD3, the conversion will be applicable to a larger variety of IFC models. For example, the model may contain an 'IfcSpace' which has external space boundaries that cover the complete exterior shell making the geometric conversion trivial. However, the model is not required to have such information and thus no assumptions are made about its presence.

Several aspects are outside the scope of this thesis. The conversion of the terrain or site around the building is not covered in this thesis. Therefore also no terrain intersection curves are generated. Furthermore, this research does not intend to enhance the model by detecting features. If certain semantic properties are not stored in the IFC file, either the model or the IFC standard should be updated. For example, chimneys in CityGML should be marked as 'BuildingInstallation', but the semantic property 'IfcChimney' is newly introduced in IFC4, thus not available during the project. Developing feature recognition software is time consuming and will likely be limited with respect to accuracy and possible ambiguities. An update of the IFC standard which increases the semantic capabilities of IFC, will provide better results.

The conversion methodology is limited to building models for which there should be only one exterior shell (if geometrically possible). If there are multiple buildings in the input, these will not be separated unless they form separate exterior shells, for example when they are disjoint. An IFC file with multiple buildings can still be converted by splitting it beforehand and merging it with an appropriate method afterwards. By doing so the decision is left up to the user to decide whether he wants the buildings to have either shared or separate geometries. Both options are equally valid according to the CityGML standard.

Finally, how to construct the explicit geometry from an IFC file is not covered by this report since this is part of the IFC specification and can be found in the IFC2x3 Model Implementation Guide [23]. It is also assumed that the input is valid according to the IFC format. Thus no healing is required.

1.3 Outline

Background information on CityGML and IFC is given in Chapter 2 together with definitions of proper CityGML LoD3 and an evaluation of the current state of development. Chapter 3 covers the conversion methodology for semantics, geometry, and for creating and ensuring the validity of the output model. The implementation of the prototype is described in Chapter 4, while also the evaluation of the results can be found there. Finally in Chapter 5 provides the conclusions, recommendations for both CityGML and IFC and future work from this research.

Chapter 2

Background and Related Work

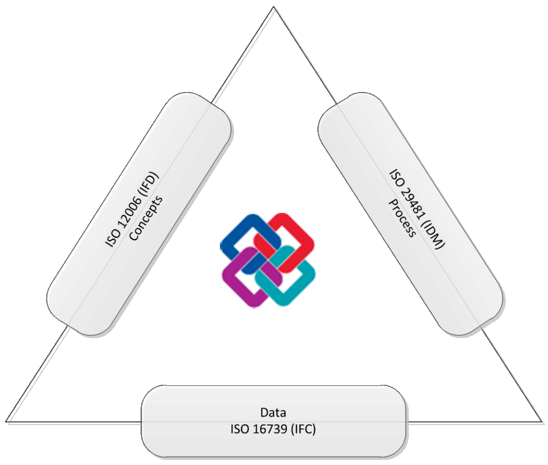
Connecting the BIM with the GIS world is not a new topic and progresses have been made. For the integration of two data formats there are four conversion strategies: mapping, alignment, transformation and fusion [24]. Since BIM and GIS are primarily designed for different purposes, an implementation of the fusion strategy at the system level is still not feasible according to Wu and Hsieh [25]. An implementation at the data level is however proposed by El-Mekawy et al. [26], a Unified Building Model (UBM). IFC semantics are mapped to generalized UBM semantics, making it an incomplete fusion which is believed to be unusable by the BIM world. Efforts of aligning CityGML and IFC have been made in the form of extensions. For IFC the IFC for GIS (IFG) extensions was created, for CityGML the GeoBIM extension [3]. The GeoBIM Application Domain Extension (ADE) provides for the ability to store IFC type geometries and semantics in a CityGML file format. Thereby no link is made with the format expected in the core of CityGML. There are several converters available which do convert IFC into CityGML without extending the standard. These are: a BIMserver plugin by A.J. Jessurun, IFCExplorer by Karlsruhe Institute of Technology (KIT) and Feature Manipulation Engine (FME) from Safe Software. As is elaborated in Section 2.4, none of the converters is able to generate semantics or geometry which fully adheres to the CityGML standard. In this thesis project different strategies are applied for the conversion of IFC to CityGML LoD3. In addition, methods for the transformation of the geometry and the mapping of semantics are developed.

Nagel [21] has implemented and documented the procedures for converting IFC properly to CityGML LoD1. When doing such a conversion between formats it is prudent to know their purpose and characteristics. Knowledge of the standards is necessary during research, but also to be able to verify the validity of output models.

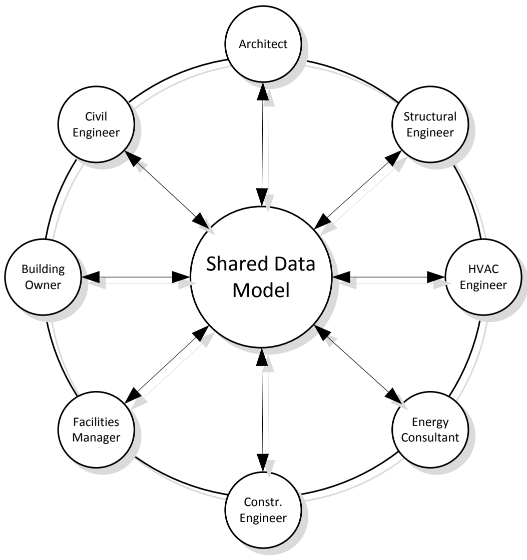
In Section 2.1 and Section 2.2 relevant information on the standards of IFC and CityGML is provided. The definitions for what is considered valid in CityGML are given in Section 2.3. These definitions are then used in Section 2.4 to evaluate converters which are currently available.

2.1 IFC

IFC is a standardized open data model developed for BIM by the international organization buildingSMART. IFC depends on two other buildingSMART standards for BIM, the data dictionary International Framework for Dictionaries (IFD) and the process definition Information Delivery Manual (IDM) (see Figure 2.1a). BIM is used in multidisciplinary building projects (Architecture, Engineering and Construction (AEC) amongst others, see Figure 2.1b) for managing complex communication and information sharing processes throughout the life cycle of the building in a multi-dimensional model [27]. The IFC data format is based on the EXPRESS language as a part of the STandard for the Exchange of Product model data (STEP) standard (ISO 103030) for product data exchange [28]. There is no universally accepted building model for IFC [29]. As such, the focus of this research is on the IFC core described in the IFC standard documentation and ISO 16739 standard (see Appendix A).



(A) buildingSMART standards for BIM



(B) IFC, a shared data model

FIGURE 2.1: Information on IFC (Source: [5])

At the start of this project IFC4 had not yet been released. Barely any of the IFC viewers, creation tools and development libraries support IFC4, therefore the IFC2x3 release is used during this project. IFC2x3 can also be encoded in the Extensible Markup Language (XML) format, ifcXML2x3, however the STEP physical file encoding is the preferred file structure [23].

A Unified Modeling Language (UML) diagram with some of the more relevant IFC semantic properties and relations is shown in Figure 2.2. It is important to note that an

'IfcObject' and its subclasses can recursively be decomposed by other 'IfcObjects'. There are many other relations between objects possible, but there are only two other relations relevant for CityGML. The 'IfcRelContainedInSpatialStructure' is used to determine whether an object is part of the building or the surrounding site. The 'IfcRelDefines-ByType' relation is used to check whether there is an 'IfcTypeObject' which contains more information on the object.

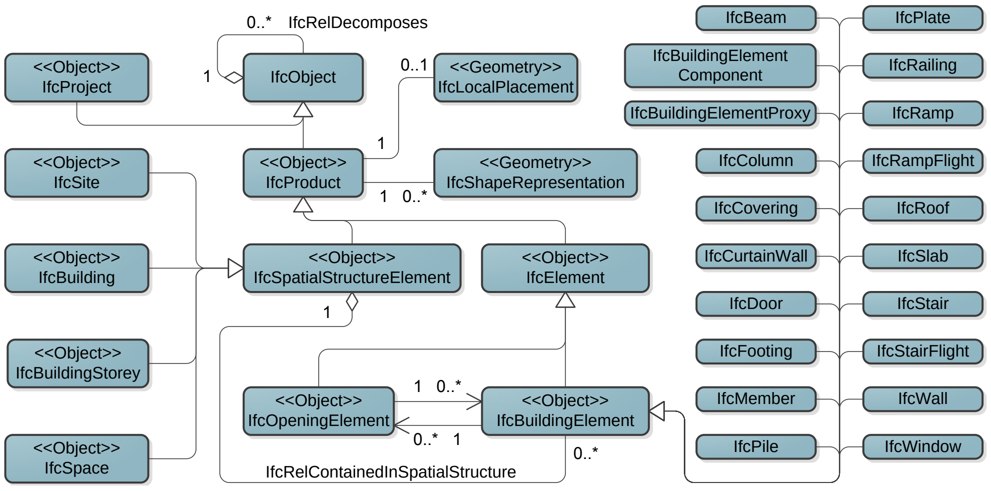


FIGURE 2.2: UML diagram for IFC entities, only the most relevant objects and relations are represented (Sources: [12, 22, 30])

There are many types of modelling representations defined in IFC2x3. The representations for solid models can be divided into three main categories (see Figure 2.3).

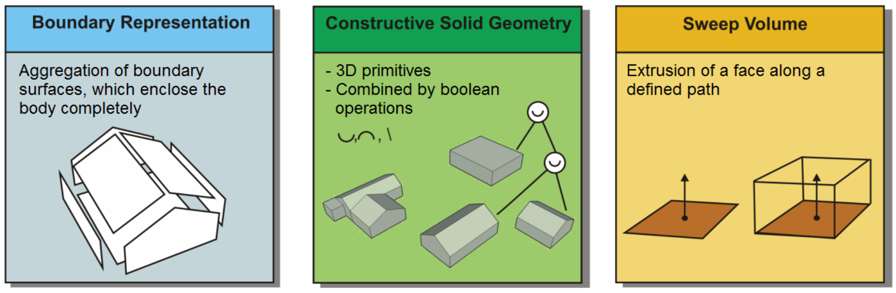
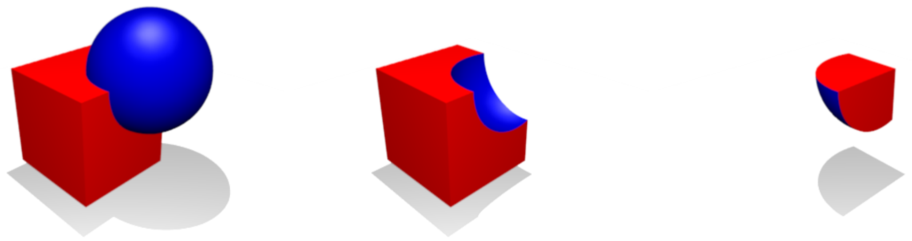

Boundary Representation
Aggregation of boundary surfaces, which enclose the body completely
Constructive Solid Geometry
- 3D primitives
- Combined by boolean operations
Sweep Volume
Extrusion of a face along a defined path

FIGURE 2.3: The three possible approaches for representing 3D objects in IFC (Adapted from: [31])

- **B-rep** - In a B-rep a solid body is represented by planar faces. The faces are located only at the boundary of the body and enclose the body completely [25]. Every face marks the border between what is inside and outside the body. B-rep is used for complex geometry objects such as 'IfcDoor' and 'IfcWindow'.

- **Sweep volumes** - Sweep volumes define a solid body by a 2D profile and a path [32]. The explicit geometry of the body can be computed by moving the profile along the path. The 2D profile may be a primitive shape such as a disk or a rectangle, or a polygon which may contain holes. The sweep can either be a linear extrusion or a rotational sweep, where the path is defined by an axis and an angle [21].

- **CSG** - CSG is used to create solid bodies by one or many Boolean operations on base solids. A Boolean operation between two geometries generates a new geometry which can for example be the union difference or intersection (see Figure 2.4). The set of operations needed to create the final solid can be represented in a tree structure. The leafs of the tree are the base solids. The base solids can be defined by one of the former defined modelling types or by 3D primitives like a sphere, cube or half-spaces [21].



(A) Boolean union (∪) (B) Boolean difference (−) (C) Boolean intersection (∩)

FIGURE 2.4: Boolean operations between a cube and a sphere (Source: [33])

The geometry of the later two types are stored implicitly, meaning only the parameters are given to generate the geometry. A IFC viewer/reader has to apply the sweeping and CSG computations before being able to visualise or use the objects. In Figure 2.5 an example is given for implicit CSG geometry, the figure also depicts how at the same time the relations between a door and a wall are stored.

Calculating the explicit B-rep geometry does not yield a unique solution. For example a disk should be converted to a regular polygon, however the number of sides of the polygon depends on the converter. For the purpose of creating valid CityGML geometries, it is sufficient to have the geometry represent the original model correctly. Although, there are three geometric models, in practice most IFC models are built using sweep volumes and CSG [18].

There are two possibly confusing differences between IFC and CityGML to note. The first is that in IFC implicit geometry refers to the geometry that is implied by the parameters stored in the IFC file and the definitions in the IFC Object Model specification. In contrast, according to the CityGML standard, implicit geometry is geometry which is stored once as a template and can be reused multiple times by referring to it.

The second one is that BIM uses the acronym Level-Of-Development (LOD) (capital 'o') which indicates the state of development from conceptual (level 100) to as-build

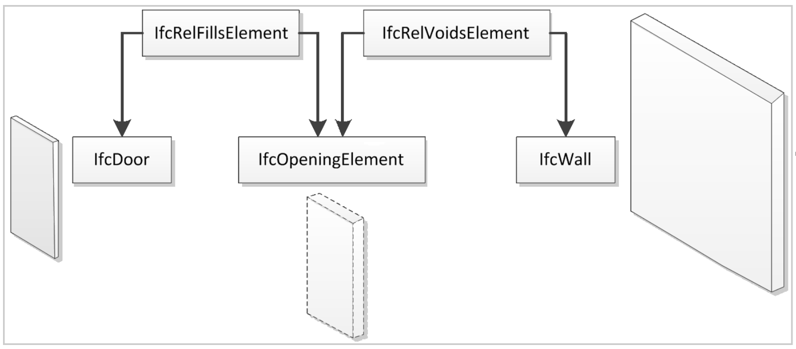


FIGURE 2.5: An example of implicit geometry: the wall is cut by the opening element using the Boolean difference. The door is then placed within the gap in the wall (Adapted from: [5])

(level 500), which is not to be confused with Level-of-Detail in CityGML [34]. To make matters worse, Level-Of-Development used to be called Level-of-Detail. The LOD has no influence on whether it is possible to convert the model to CityGML, though models with higher LODs are recommended as they contain more accurate information.

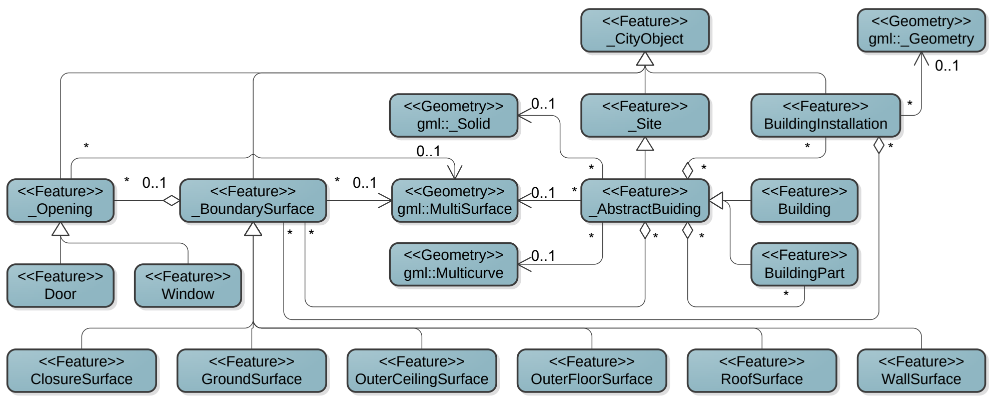
2.2 CityGML

From the GIS world the CityGML format was created as the standard for 3D modelling of cities, to be used for analytical purposes. It has since also extended beyond the scope of cities. As the view of a real city is mostly determined by buildings, building objects are often the focus of 3D city models [21].

The latest version is CityGML 2.0.0. It is based on a number of standards from the ISO 191xx family and is implemented as an application schema for Geography Markup Language (GML 3.1.1) from OGC. CityGML has been developed by the Special Interest Group 3D (SIG 3D) of the initiative Geodata Infrastructure North-Rhine Westphalia, Germany [15].

CityGML provides a standard for the meaning and definition of objects. As a result users can rely on a specific data quality and data creators, like municipalities, can focus on gathering useful data [35]. The metadata in CityGML file can be interpreted by both computers and humans, and there are extensive semantics which are directly linked to the geometries in a spatially aggregated hierarchy. Furthermore, for objects there can be five LoD representations. In Figure 2.6 the UML diagram for the relevant semantics of a CityGML LoD3 building module is shown. A complete UML diagram for all LoDs can be found in Appendix B.

One of the issues with the current state of CityGML is the lack of available creation and design tools, which would allow for a simplified generation process. Also common data sources, like Actueel Hoogtebestand Nederland (AHN)-1 and AHN-2, used to create models are often acquired only every few years, resulting in models which are already



<<Feature>> _CityObject
<<Geometry>> gml::_Geometry
<<Geometry>> gml::_Solid
<<Feature>> _Site
<<Feature>> BuildingInstallation
<<Feature>> _Opening
<<Feature>> _BoundarySurface
<<Geometry>> gml::MultiSurface
<<Feature>> _AbstractBuiding
<<Feature>> Building
<<Geometry>> gml::Multicurve
<<Feature>> BuildingPart
<<Feature>> Door
<<Feature>> Window
<<Feature>> ClosureSurface
<<Feature>> GroundSurface
<<Feature>> OuterCeilingSurface
<<Feature>> OuterFloorSurface
<<Feature>> RoofSurface
<<Feature>> WallSurface
0..1
*

FIGURE 2.6: CityGML UML for LoD3 (Adapted from: [11])

outdated when they are created. The room for interpretation that is given in the specification is an issue when performing analyses on CityGML models. 3D InformatieModel GEOgrafie (IMGeo) is a Dutch initiative which proposes a more precise specification on how CityGML files should be modelled. The set of allowed geometries in CityGML are limited to planar B-rep (see Section 2.1). Arcs, although not disallowed by CityGML, are not used in this project since planarity can only be achieved if the incident faces are coplanar. Furthermore, there validity cannot always be determined. A limited set of geometries is not necessarily bad since a limited set of geometries improves the ease at which a dataset can be analysed as the geometries are more predictable. Yet it also decreases the modelling possibilities. For example circular features, which are present in IFC, have to be approximated by a set of line segments.

CityGML is often called an exchange format, this gives the impression that it is meant as an intermediate step between two data formats. However, CityGML is defined as a common semantic information model for the representation of 3D urban objects that can be shared over different applications [11]. It is the final product and therefore creating valid and semantically rich CityGML to be used in GIS applications is of more importance than retaining all semantic and geometric information stored in IFC files.

2.3 Validity Criteria for CityGML LoD3

In this section the constraints and definitions of proper CityGML LoD3 semantics and geometry are established. These definitions are later used for defining the conversion methodology and to verify the validity of the output models generated by the prototype implementation.

2.3.1 Requirements and Geometric Constraints on Semantics

A building model in CityGML can contain three types of objects with volumetric geometries, namely the 'Building' itself, 'BuildingParts' and 'BuildingInstallations'. A 'Building' can recursively consist of several 'BuildingParts' and can have multiple 'BuildingInstallations'. The exterior shell of each of these objects can have boundary surfaces. For LoD3 the following types are defined: 'WallSurface', 'RoofSurface', 'ClosureSurface', 'GroundSurface', 'OuterCeilingSurface' and 'OuterFloorSurface'. These boundary surfaces can also have 'Openings' which are either 'Doors' or 'Windows'. In Figure 2.7 the boudary surfaces are shown and how they are generally applied. 'Openings' and 'ClosureSurfaces' imply a connection between the interior and exterior of the shell, however in contrast to a 'ClosureSurface' an 'Openings' may be closed, blocking the transit of people [11].

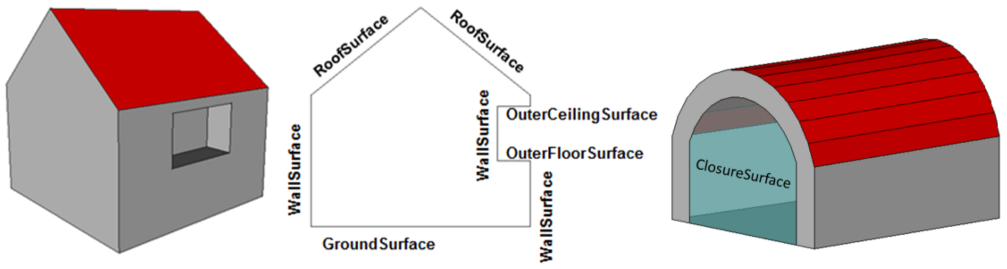


Figure 2.7: LoD3 boundary surfaces, no 'Openings' (Source: [36])

The CityGML standard states that "The polygon defining the ground plate is congruent with the buildings footprint" [11]. In this thesis this statement is interpreted as: the union of the vertical projections of all 'GroundSurfaces' should be equal to the building's footprint. There are also constraints on the normal of certain boundary surfaces. Not all combination of normal directions and boundary surfaces are allowed. For example, if a 'GroundSurface' face has an upward pointing normal it is invalid, the semantics of the face would be valid if it for instance were a 'WallSurface' instead. These are shown in Table 2.1, where down and up are interpreted as all direction aimed lower or higher than horizontal respectively.

Table 2.1: Normal vector limitations on semantic surfaces

Surface type	**Allowed direction(s)**
WallSurface	All
RoofSurface	All
ClosureSurface	All
GroundSurface	Only down
OuterCeilingSurface	Only down
OuterFloorSurface	Only up
Opening	All

Storing the boundary surfaces and the solid geometry separately in a CityGML file breaks the spatio-semantic coherence of CityGML. However it is necessary as the standard does not allow for a solid to be bounded by boundary surfaces (see Figure 2.6). This is done to maintain compatibility with Geography Markup Language (GML). The separation allows in LoD4 to model 'Rooms' with different surface orientation for the solids and boundary surfaces, normals pointing outward for solids and inward for interior boundary surfaces. However the inversion of the normals can also be unambiguous deduced from the object type to which the solid belongs. When separating the solids and boundary surfaces the geometries should then be stored in the boundary surfaces and linked via 'XLinks' as is shown in Figure 2.8.

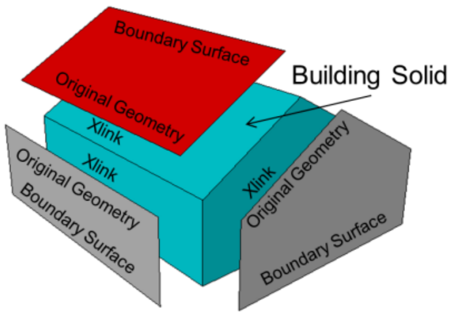


Figure 2.8: Linking surfaces and solids using 'XLinks' (Source: [36])

2.3.2 Geometric Validity Requirements

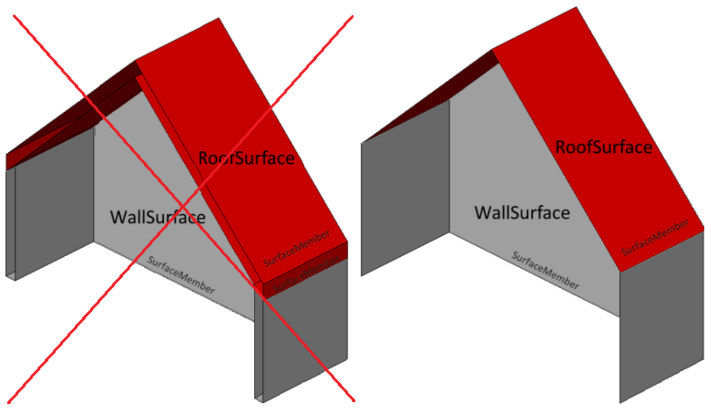


Figure 2.9: Only visible surfaces in CityGML (Adapted from: [36])

There is a significant difference between the geometric representation of IFC and CityGML. In CityGML only the visible (virtual) surfaces are modelled (see Figure 2.9). Extracting the exterior shell from the solids of IFC is a required operation for the output to adhere to the standard. For LoD1 to LoD4, the solid of an '_AbstractBuilding' is comprised of only one such shell. An inner shell, as shown in Figure 2.10, representing a cavity in the solid is thereby not allowed. The concepts of exterior shell or the solid of an '_AbstractBuilding' are almost interchangeable, however the shell refers only to the boundary surface of the solid.

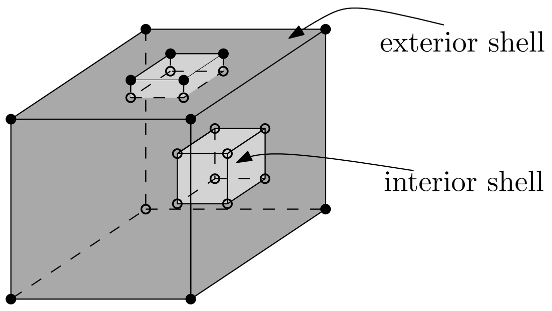


FIGURE 2.10: One solid with one exterior shell and one interior shell, a cavity (Source: [19])

Unlike 'Buildings', the geometry for '(Int)BuildingInstallations' is completely unrestricted, and may therefore also have inner shells. Inner shells at LoD3 are however believed to be undesirable. For LoD4, having no inner shells means that the 'Room' solids overlap the 'Building' solid. The solids in CityGML represent a virtual boundary or a division of space. A point which is within a 'Room', is also geometrically within the 'Building'. Overlapping solids at LoD4 are therefore a necessity. For example, when the exterior shell has a 'ClosureSurface', which is also shared by a 'Room'. If one would want to model this case using a 'Building' solid having inner shells which are filled by the 'Room' solid, the result would be an invalid solid. The invalidity is due to the requirement that the intersection between shells of a solid may only be 2D or higher. Conversely, at LoD3 the solids should not overlap, and in case two solids touch, the intersecting surfaces should be linked using 'XLinks'.

The ISO19107 standard defines the criteria for a solid to be valid. An important criteria is that the shells of a solid are 2-manifold which is beneficial for analytical computation. A manifold shell has the property that, around every location on the shell, the neighbourhood is homeomorphic to a disk [37]. For a ring to be 1-manifold this neighbourhood should be a line. A frequently occurring special case of a non-2-manifold situation is when there are more than two faces incident to an edge. Due to the constraints on the geometries in this project only four types of non-manifold cases need to be distinguished for solids/shells (see Figure 2.11). The interior of the two solids on the left are however disconnected and the solids can therefore both be stored as two separate manifold solids without any changes to the shape. When present in the solid of a 'Building', the smallest solid is stored as a 'BuildingPart'. The methodology for handling the other non-manifolds is given in Section 3.3.1.

Degeneracies are another form of unwanted geometries. An example of a degeneracy is when two sequential edges of a face (partially) lay on top of each other. When present in a triangle its area would be zero. In Figure 2.12 two cases of surface degeneracies are

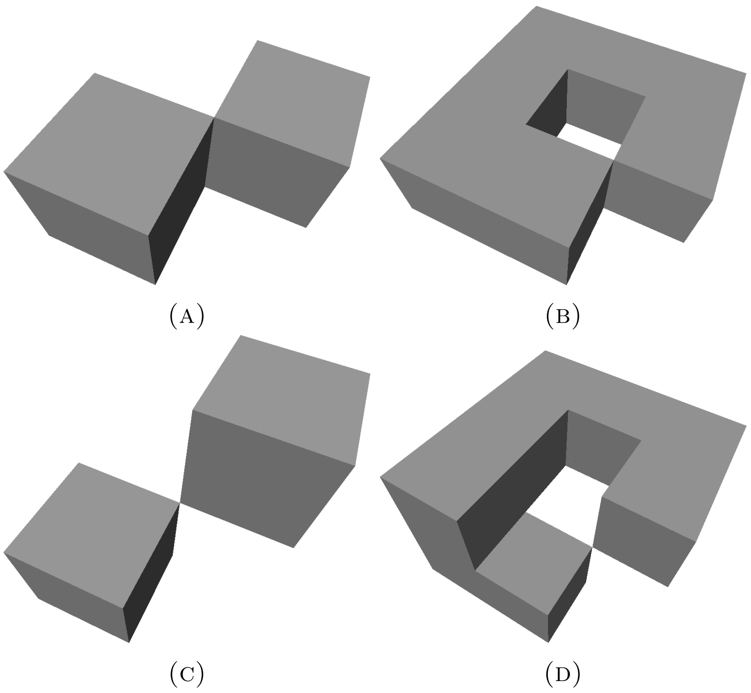


FIGURE 2.11: Four non-manifold cases: two volumes on the left, single volumes on the right, edge non-manifolds at the top, vertex non-manifolds below

distinguished. Based on their shapes in a triangular mesh they are called needles and caps [38]. In the case of a needle two vertices have the same coordinates, whereas those of the vertices of a cap are all distinct. Degeneracies according to ISO do not make a geometry invalid, however as indicated by van Oosterom et al. [39], when using floating-point arithmetic, parts of polygons can collapse and become invalid. Those situations can occur even when the distance 'd' between a vertex and an edge or another vertex is almost zero. To optimize the geometry for analysis and since a CityGML file should be prepared for the usage in many applications, degeneracies are avoided if possible. Fully degenerates can be fixed, however nearly degenerates cannot always be removed without changing the shape as is described in Section 3.3.1.

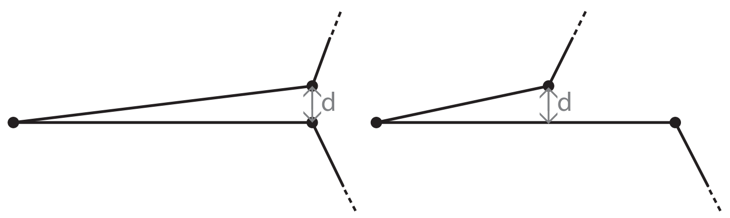


FIGURE 2.12: Two types of degeneracies if distance d=0, left a needle, right a cap

The validation of solids according to ISO19107 was researched by Ledoux [19]. Based on his work and some interpretations by Boeters [40] the following list of invalid geometries has been created.

- **Ring level**:

- *Not valid 1-manifold*
  - *Ring not closed* - The last vertex of a ring must be the same as the first.
  - *Dangling edges* - Vertices may not be connected to only one edge.
  - *Ring self intersect* - Edges should not intersect each other.
- *Free geometries* - All edges and vertices must be connected to the ring.
- *Inconsistent edge orientation* - An edge in a ring should start with the last vertex of the previous edge.

- **Face level**:
  - *Non planar face* - The vertices of a face must be in one plane.
  - *Inner ring wrong orientation* - Inner rings must have an opposite orientation with respect to the outer ring.
  - *Inner ring intersects at more than one point* - An inner ring may only touch other rings at one point each.
  - *Inner ring outside outer* - An inner ring must be inside the outer ring.
  - *Interior is not connected* - The interior of a face must be connected.
- **Shell level**:
  - *Not valid 2-manifold*
    - *Shell not closed* - The shell must be watertight and thus not contain holes.
    - *Dangling edges or faces* - Vertices may not be connected to only one edge and edges may not be connected to only one face.
    - *Shell self intersects* - Faces should not intersect each other.
  - *Free geometries* - All faces, edges and vertices must be connected to the shell.
  - *Inconsistent face orientation* - The normals of faces should either all point in- or outward.
- **Solid level**:
  - *Bad surface normal orientation* - The normals of all faces of the solid should point outwards relative to the interior.
  - *Shells intersection is 2D or higher* - Shells in the solid may only intersect at points and line segments.
  - *Inner shell outside outer* - Inner shells may not be outside the outer shell.
  - *Interior not connected* - The interior of a solid must be connected

2.4 Current Development

Conversion tools from IFC to CityGML, but also CityGML to IFC are freely available. Three of such tools have been tested in this research and are described below. They are evaluated at the end of this section.

- **BIMserver -** The Building Information Modelserver (BIMserver) is being developed to centralize the information of a construction project and make the collaboration between actors more efficient and effective. It therefore is also capable of exporting the IFC files to COBie, CityGML, Collada, KMZ and SceneJS. Their current implementation of a IFC to CityGML writer, developed by A.J. Jessurun, aims at producing a CityGML file which can be visualized by CityGML viewers. The closed source IFC Engine from (Nederlandse centrale organisatie voor) Toegepast Natuurwetenschappelijk Onderzoek (TNO) is used by the converter. A wire frame front view of a resulting conversion can be seen in Figure 2.13. The implementation by Laat and Berlo [3] is not part of the evaluation as their intentions were to extend CityGML to support IFC classes and geometries.

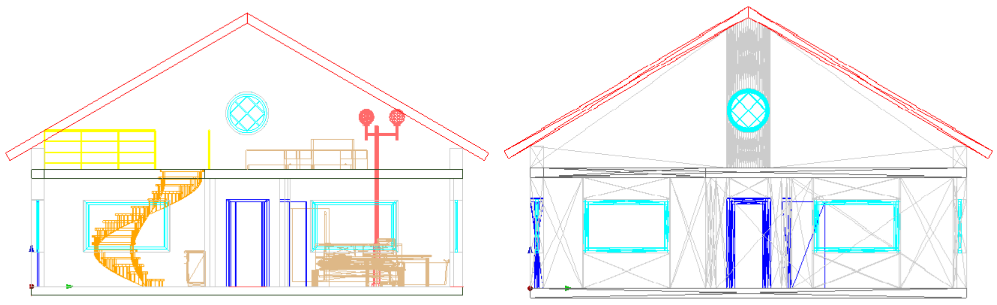


FIGURE 2.13: Conversion results by BIMserver, IFC on the left, CityGML on the right

- **KIT IFCExplorer -** At KIT a software environment is being developed called IFCExplorer. At the moment they are able to merge CityGML, CityGML ADEs, BoreholeML, gbXML, Alkis, XPlanGML and Map Services into one scene as native features. Their goal is not to transform IFC to CityGML, but to have all models natively in one environment. Integrating IFC will be a next step [41].

  Their software is however capable of converting between IFC and CityGML. The converter can create all LoDs to some extent, but there are no solids defined and the geometry does not yet adhere to the CityGML specification. LoD1 and LoD2 can also be converted to IFC, which results in models that are in compliance with the IFC specification, but does not comply to Coordination View 2.0, which is an initiative to define the correct implementation of IFC. Development by KIT continues on LoD1 and LoD2. On request, the conversion results are not shown since the results are not proper CityGML yet which could lead to misinterpretations.

- **Safe Software FME -** FME is a flexible application which does not use a one-to-one format architecture. There is support for both reading IFC as well as writing CityGML, however the required data model transformations need to be built in the Workbench. The conversion is thus not a native function of FME. There are currently no plans to create a converter for generating proper CityGML geometry from IFC. Results of a conversion using a pre-made FME workbench [42] can be seen in Figure 2.14.

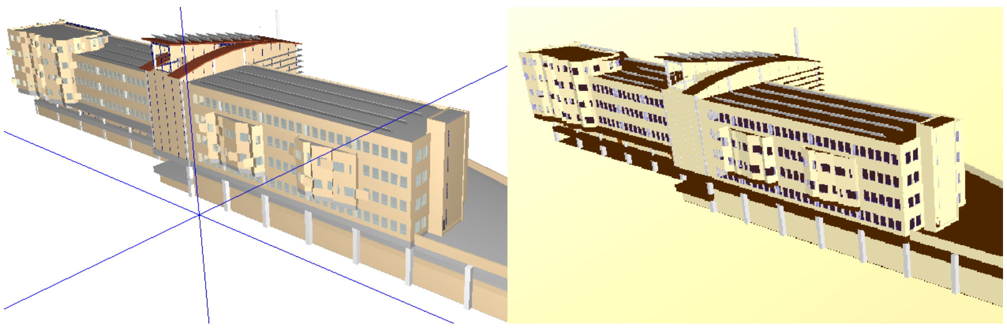


Figure 2.14: Conversion results by Safe Software FME, IFC on the left, CityGML on the right [42]

Although the output of the different conversion tools visually represents the input building to some extent, none of the converters is currently capable of creating valid LoD3 CityGML geometries, nor fully correct semantics (see Table 2.2). The calculation of the explicit geometries from IFC is properly handled by the Bimserver and IFCExplorer, but the FME workbench does not apply Boolean operation. In CityGML only (virtual) surfaces that are visible in reality should be modelled. The results of each of the converters still show the whole original geometry. The IFCExplorer however, filters out internal objects by relying on the 'external' attribute. The semantics are at a similar level. Due to the limited geometric conversion no distinction is made between interior and exterior wall surfaces and none of the converters handles the surface normal constraints on the semantics. In general one could say that the current converters are not capable of generating LoD3 CityGML building models from IFC which are optimized for the use in 3D GIS applications. In Chapter 3 a methodology is given for a converter which does just that.

Table 2.2: Evaluation of current IFC to CityGML LoD3 converters

	FME	BIMserver	IFCExplorer
Correct Explicit IFC Geometry	✗	✓	✓
Transformation of Geometry	✗	✗	✗
Correct Semantics	✗	✗	✗
ISO Validity of Geometries	Equal to IFC	Equal to IFC	Equal to IFC

Chapter 3

Methodology for the Conversion

In Chapter 2 the information on the two standards used in this thesis and the requirements on their formats is given. In this chapter the information is used to formulate the actual methodology for the automatic generation of CityGML LoD3 building models from IFC models. In Chapter 4 an implementation of a prototype for this methodology is described together with an evaluation of the results.

The methodology is based on the three pillars described in Section 1.1: semantic mapping, geometric transformations and refinements. Instructions on how to parse an IFC file and how to generate the explicit geometry can be found in the IFC2x3 Model Implementation Guide [23]. The flow diagram in Figure 3.1 provides an overview of the conversion methodology. Detailed flow diagrams for the whole conversion can be found in Appendix C. The methodology outlined in this chapter can be used by others as a guideline for the automatic generation of LoD3 building models.

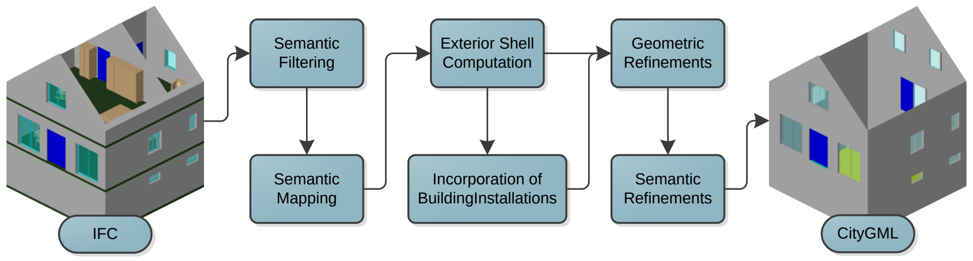


FIGURE 3.1: General workflow diagram of the conversion methodology

First the semantic filtering and mapping is described in Section 3.1. The methods produce explicit IFC geometries with CityGML semantics. Section 3.2 explains the methods which have been developed for the transformation of these IFC geometries into CityGML geometries. During the geometric transformation the semantics are maintained. At the end of this section alternative methods for the geometric transformation are evaluated and discussed. Lastly, in Section 3.3 the refinement processes are clarified which are needed for the creation of a CityGML and ISO19107 conform file.

3.1 Semantic Filtering and Mapping

A LoD3 buildings in CityGML can have semantic properties at three different levels of geometry: the solid level, the face level and the curve/line string level. Since terrain and thereby the terrain intersection curve is out of the scope of this thesis and 'BuildingInstallations' remain solid, edge level semantics are not discussed. Furthermore since a building can only have one solid ('BuildingParts' are only created when geometrically necessary), passing semantics at the solid level is trivial, thus leaving only the extraction of face level semantics from IFC.

IFC on the on the other hand has a completely different structure for storing semantics, as is explained in Section 2.1. In IFC an object is connected via a network of relations to other objects. For the extraction of CityGML semantics sometimes the type of object is sufficient, at other times the network needs to be traversed in search of the optimal semantics. To determine what the semantics are in CityGML for one particular face, a combination of multiple semantic values from IFC and certain geometric properties are required.

This section aims to provide an unambiguous methodology for the extraction of CityGML semantics. As such, it also covers the implementation to some extent. The filtering and mapping is done by analysing all the 'IfcObjects' in the input IFC file. This section is describes when an 'IfcObject' is relevant for the conversion and how CityGML semantics can be extracted from the IFC semantics. Important properties of an object which are used are:

- Whether it has geometry, IfcObjects do not always have geometry
- Whether it is contained in a building
- The entity class
- The 'PreDefinedType' attribute
- Whether it decomposes another object
- The normal vectors of the faces, a geometric property

A complete overview can be seen in Appendix C Figure C.1. These processes will yield a set of relevant geometries of which the faces contain the CityGML semantics. While in this methodology only the boundary surface types are attached to faces, other attributes may be attached as well.

3.1.1 Filtering of IFC objects based on semantics

There are around 900 classes defined in the IFC schema [3]. These classes are used to store information like geometric representations, properties, relations and topology.

Considering the output driven approach, which is explained in Section 1.2, very few of these classes are relevant for CityGML LoD3. Furniture for example are not relevant to the buildings exterior shell they are also movable and should therefore not be modelled as a 'BuildingInstallation'. Furthermore, objects that do not have geometry or are not part of the building can be ignored. Filtering these objects leaves only objects which have meaningful mappings in CityGML.

The relevant classes are 'IfcSpace' and all the subtypes of 'IfcBuildingElement' (see Figure 2.6). All other classes either represent movable objects or are abstract classes without geometry. For each 'IfcObject' present in the IFC file it is checked whether it has geometry and whether it is contained within a building. For the latter the 'IfcRel-ContainedInSpatialStructure' relation is used recursively. In Figure 3.2 a flow diagram is shown for the filtering process.

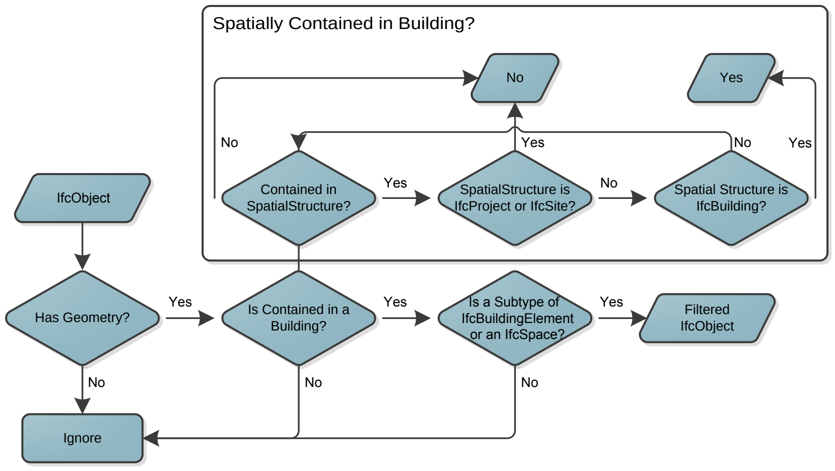

Spatially Contained in Building?
No
Yes
No
Yes
No
Yes
Contained in SpatialStructure?
Yes
SpatialStructure is IfcProject or IfcSite?
No
Spatial Structure is IfcBuilding?
IfcObject
Has Geometry?
Yes
Is Contained in a Building?
Yes
Is a Subtype of IfcBuildingElement or an IfcSpace?
Yes
Filtered IfcObject
No
No
No
Ignore

Figure 3.2: Workflow diagram for the filtering of IfcObjects (the boxed processes detail the process to which they are connected)

One may notice that the 'IsExternal' property is not used for the filtering. The value of 'IsExternal' should be true when the geometry of the object is connected to the exterior of the building. Using only objects with this property set to true could potentially result in a speed up during the geometric transformation. However, by doing so the converter would become dependent on the right application and consistent usage of the property [21]. This property is therefore ignored. For similar reasoning subtypes of the relation 'IfcRelConnects' are not used during the geometric transformation. If the geometries of two objects are connected this can also be deduced from the geometries themselves.

3.1.2 Semantic Mapping from IFC to CityGML

The mapping of semantics determines for every face of the filtered 'IfcOjects' what the CityGML boundary surface type is when the face is to be part of the exterior shell of the building. Since LoD3 does not have inner shells, thus does not distinguish between interior and exterior, the mapping can be performed before the geometric transformation. The mapping of a face depends on its normal and the types and relations of the 'IfcObject' to which it belongs.

The semantics mapping has been developed from scratch for this thesis using the documentations of the two standards [11, 12]. For every combination of an entity type, 'PredefinedType' and face normals type, a link is made to the CityGML boundary surface counterpart. To which boundary surface type the link is made is based on the intended purpose and use of the 'IfcObject' given the entity and 'PreDefinedType', and how CityGML expects the faces of such an object to be modelled.

For some combination the link is certain, for example when the entity type is 'IfcRoof'. These mappings are called 'final'. In other cases it is possible that more information results in a different mapping. These cases are called 'temporary'. For example, an 'IfcPlate' on its own could potentially be anything and is thus mapped to a 'WallSurface'. However, the 'IfcPlate' might decompose an 'IfcWindow', in which case the 'IfcPlate should be mapped to 'Window' in CityGML. As such, when a combination is marked as 'Temporary', the mapping of the parent object which the active object decomposes is used. This search is done recursively until a 'final' mapping is found, or the object does not decompose another parent object, or the parent object is no longer a subtype of 'IfcBuildingElement'. In case no 'final' mapping could be determined the last 'temporary' mapping is used instead. The whole process is depicted in Figure 3.3.

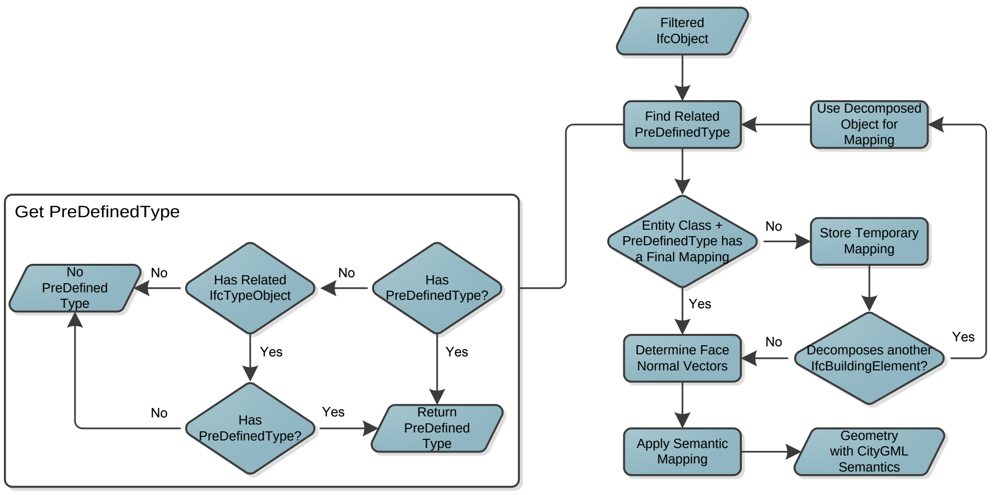

Filtered IfcObject
Find Related PreDefinedType
Use Decomposed Object for Mapping
Get PreDefinedType
No PreDefined Type
No
Has Related IfcTypeObject
No
Has PreDefinedType?
Yes
Yes
No
Has PreDefinedType?
Yes
Return PreDefined Type
Entity Class + PreDefinedType has a Final Mapping
No
Store Temporary Mapping
Yes
Determine Face Normal Vectors
No
Decomposes another IfcBuildingElement?
Yes
Apply Semantic Mapping
Geometry with CityGML Semantics

FIGURE 3.3: Workflow diagram for the mapping of semantics (the boxed processes detail the process to which they are connected)

The mapping is shown in Table 3.1. The definitions of up and down are given in Section 2.3.1. Note that there are no boundary surface properties determined for 'BuildingInstallations'. The reason for which is that the set of boundary surfaces present in CityGML is too limited. It is believed to be undesirable for example for a stairs to be covered by 'Floor-' and 'WallSurfaces. In case semantics for balconies and dormers are added to the IFC standard, then for those 'BuildingInstallations' the boundary surface properties using the exact same methodology as for 'Buildings'. The site/terrain is out of the scope of the this thesis, however if the geometry of the terrain is known, all faces bellow the terrain with a normal pointing down should be mapped to 'GroundSurface'.

'BuildingInstallations' are different from 'Buidings' and 'BuildingParts' as they can have any kind of geometry. Types of objects that are stored as 'BuildingInstallations' are for instance: stairs, columns, beams, dormers and chimneys. The latter would in IFC2x3 need to be modelled using 'IfcBuildingElementProxy' instead of 'IfcWall' in order for it to be marked as 'BuildingInstallation'. Dormers are not explicitly marked in IFC. The only indication that there might be a dormer is when an 'IfcRoof' is decomposed by other 'IfcRoofs', but this is not sufficient for extracting the geometry potential dormers.

A distinction can be made between columns and beams which are detached from the building and columns and beams which are part of a wall and thus potentially part of the exterior shell of the building. The semantic mapping is different for both kinds. Although possible, it is not common practise in IFC to model columns and beams as part of another object, such as an 'IfcWall' or 'IfcSlab'. In case this modelling practise is not applied, the kind of column or beam cannot be determined from the semantics and requires user input.

There is another issue which may require user input which is caused by the mapping of 'IfcSpace' to 'ClosureSurface'. An 'IfcSpace' represents a zone which can be constraint by physical and virtual boundaries. This mapping is very desirable when the entrance to a building is a large open space, see for instance the train station in Figure 2.7. However, an 'IfcSpace' can also be outside of de building while still being modelled as part of the building. This occurs for example when the zone of a balcony is modelled, as can be seen in Figure 3.4. Here the 'IfcSpace' should be ignored, but IFC provides no semantics to distinguish these cases.

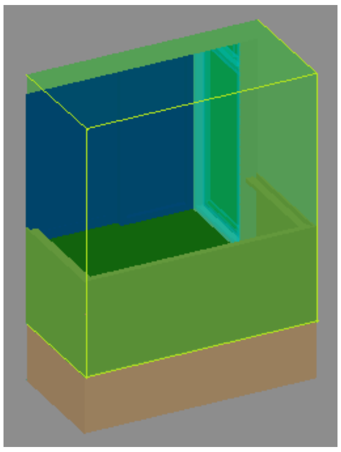


FIGURE 3.4: An IfcSpace can be used to model the zone of a balcony

Table 3.1: Semantic mapping from IFC to CityGML LoD3 based on semantics and the normal vector of each face. The temporary mappings can change based on the parent object. All faces bellow the terrain with a normal pointing down should be mapped to GroundSurface

IFC entity type	**PredefinedType**	**Temporary/ Final**	**CityGML LoD3 mapping based on face normal**		
			Up	Horizontal	Down
IfcBeam		Temporary		– BuildingInstallation –	
IfcBuildingElementComponent		Temporary		– BuildingInstallation –	
IfcBuildingElementProxy		Temporary		– BuildingInstallation –	
IfcColumn		Temporary		– BuildingInstallation –	
IfcCovering	CEILING	Final	OuterFloorSurface	WallSurface	OuterCeilingSurface
	FLOORING	Final	OuterFloorSurface	WallSurface	OuterCeilingSurface
	ROOFING	Final	RoofSurface	RoofSurface	OuterCeilingSurface
	Others	Temporary	WallSurface	WallSurface	WallSurface
IfcCurtainWall		Final	WallSurface	WallSurface	WallSurface
IfcDoor		Final	Door	Door	Door
IfcFooting		Final	OuterFloorSurface	WallSurface	GroundSurface
IfcMember		Temporary	WallSurface	WallSurface	WallSurface
IfcPile		Temporary		– BuildingInstallation –	
IfcPlate		Temporary	WallSurface	WallSurface	WallSurface
IfcRailing		Final		– BuildingInstallation –	
IfcRamp		Final		– BuildingInstallation –	
IfcRampFlight		Final		– BuildingInstallation –	
IfcRoof		Final	RoofSurface	RoofSurface	RoofSurface
IfcSlab	FLOOR	Final	OuterFloorSurface	WallSurface	OuterCeilingSurface
	ROOF	Final	RoofSurface	RoofSurface	OuterCeilingSurface
	LANDING	Temporary		– BuildingInstallation –	
	BASESLAB	Final	OuterFloorSurface	WallSurface	GroundSurface
	USERDEFINED	Temporary	OuterFloorSurface	WallSurface	OuterCeilingSurface
IfcStair		Final		– BuildingInstallation –	
IfcStairFlight		Final		– BuildingInstallation –	
WallSurface		Final	WallSurface	WallSurface	WallSurface
WallSurfaceStandardCase		Final	WallSurface	WallSurface	WallSurface
IfcWindow		Final	Window	Window	Window
IfcSpace		Final	ClosureSurface	ClosureSurface	ClosureSurface

Although the mapping is made from scratch it does correspond to the very basic mapping published by El-Mekawy and Östman [18] (except IfcCovering) and to that from Laat and Berlo [3] for as far as that is possible. Their mappings do not distinguish between LoDs. Both mappings map complete IFC solids to face level semantics, thereby disregarding the normal constraints and all faces of a solid are either interior or exterior. Also both mapping rely and misuse the 'IsExternal' property. When a wall is marked as exterior it only means that at least some part of the wall is part of the exterior. That is, at LoD4 faces of an exterior IfcWall need to be mapped to 'InteriorWallSurfaces' depending on whether the faces itself is internal or external. Lastly, whether the decomposition relations are taken into account for the mapping is unclear from the papers.

Using the mapping and the explicit geometry, a set of geometries can be created, each face of which is linked to its own CityGML semantic property.

3.2 Geometric Transformations

The next step in the conversion is to transform the explicit IFC geometries with CityGML semantics into CityGML geometries while maintaining the semantics and the converted CityGML semantics. In Section 3.2.1 an elaboration is given on the operations needed to extract the exterior shell geometry. The last transformation steps are given in Section 3.2.2. There the methods required for the incorporation of 'BuildingInstallations' into the scene are described. An overview can be found in Appendix C Figure C.2. In Section 3.2.3 other concepts for the geometric transformation from the conceptual design phase are evaluated.

The results which can be generated with the methods from this section are in the ideal case ready to be written to a CityGML file. In reality, however, some refinement is needed to assure the validity. These refinement processes are provided in Section 3.3.

3.2.1 Exterior Shell Computation using Boolean and Morphological Operations

The main concept for extracting the exterior shell is to make sure that all IFC geometries are topologically connected by unioning all of them into one and then removing all the geometries inside the outer boundary. In the most simple case the union results in just one solid geometry from which the interior can then be removed. This process is depicted in Figure 3.5.

During the unioning of the geometries it is important that the semantics and the converted CityGML semantics assigned to the faces are maintained, including the geometric boundaries of the faces. Ambiguities due to overlapping solids are assumed not to occur as overlapping solids are not modelling reality. It is however possible for the faces of a 3D solid and a 2D surface to overlap. This requires special treatment in the

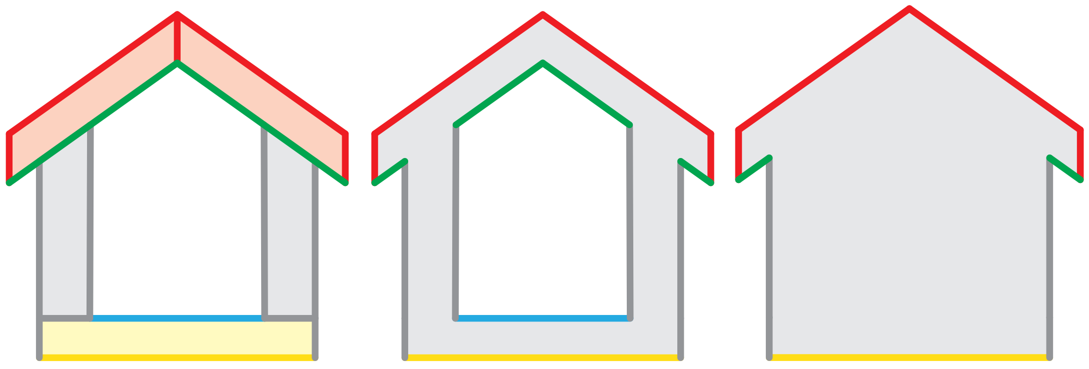

(A) Model before transformation, IFC geometries with CityGML semantics (B) Model after the Boolean union of all geometries (C) Model after removal of the interior geometry

FIGURE 3.5: Visualisation of the basic steps of the geometric transformation. The colors of the lines represent the boundary surface types (Roof- WallSurface etc.), but they may be further subdivided based on other attributes.

conversion. In Figure 3.6 an example is shown. There the 3D slab has a 2D covering to indicate that the upper side of the slab is 'ROOFING'. In those cases the semantics of the 2D geometry should always be prioritized during the conversion.

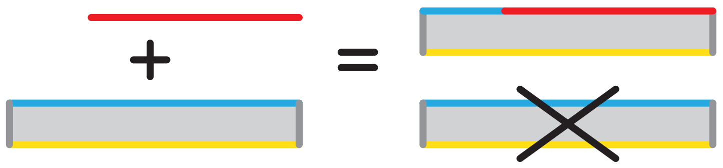


FIGURE 3.6: When two surfaces overlap, the semantics of a 2D surface object is used

It is possible that the result contains multiple solids of which the interiors are disconnected. In such cases the smaller solids are processed as CityGML 'BuildingParts' whereas the largest solid becomes the geometry of the main CityGML 'Building'. Furthermore, since interior geometry is unwanted it is removed at multiple stages during the transformation to avoid unnecessary computations. A flow diagram of the process is shown in Figure 3.7.

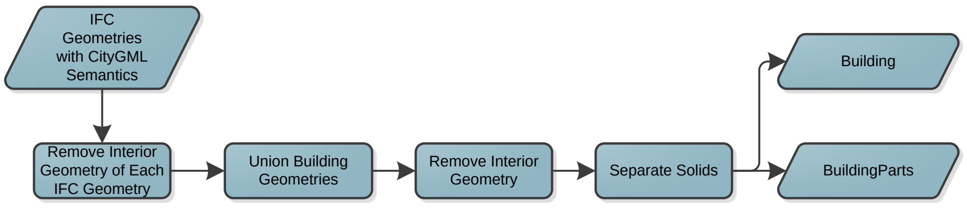

IFC Geometries with CityGML Semantics
Remove Interior Geometry of Each IFC Geometry
Union Building Geometries
Remove Interior Geometry
Separate Solids
Building
BuildingParts

FIGURE 3.7: Workflow diagram for extracting the exterior shell using union operation. This is a part of Figure C.2

3.2.1.1 Morphological Closing

Only unioning the geometries does not guarantee that the proper exterior shell geometry is extracted. Parts of the interior of the building can still be present. This occurs when there are holes in the building. Real buildings are never watertight: vents, chimneys and utilities all penetrate the exterior. Also during the modelling of the building small gaps may (unintentionally) be created. None of these causes makes the input IFC model invalid. They therefore need converted in such a way that they do yield valid CityGML data. This section explains how the gaps can be closed using the morphological dilation and erosion operations, while other methods are described in Section 3.2.3.

Dilation and erosion are two elementary operations of mathematical morphology. Dilation ($\oplus$) by a structuring element expands the input geometry, while erosion ($\ominus$) shrinks the input geometry (see Figure 3.8a and 3.8b) [43]. If the structuring element is a sphere, the operation is equivalent to the buffer operation known in GIS. The dilation of two input geometries A and B is defined by Equation 3.1. Erosion is the morphological dual to dilation, thus erosion is dilation of the background. Erosion is given by Equation 3.2, where X^c is the complement of X. Closing ($\bullet$) is an often irreversible morphological operation which is comprised of a dilation followed by an erosion operation by the same structuring element (see Equation 3.3 and Figure 3.8c).

$$A \oplus B = \bigcup_{b \in B} A_b = \{c | c = a + b, a \subseteq A \,\&\, b \subseteq B\} \tag{3.1}$$

$$A \ominus B = (A^c \oplus B)^c \tag{3.2}$$

$$A \bullet B = (A \oplus B) \ominus B = ((A \oplus B)^c \oplus B)^c \tag{3.3}$$

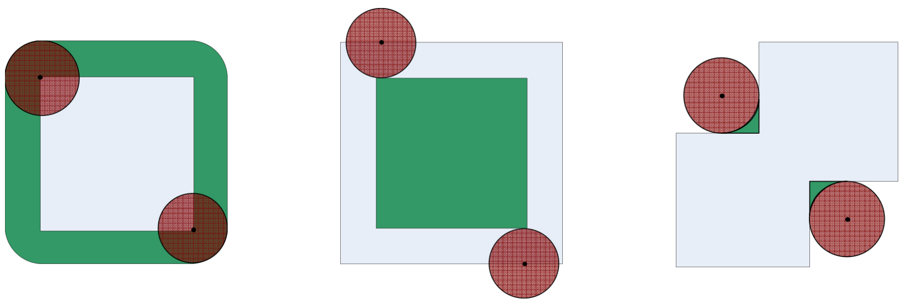
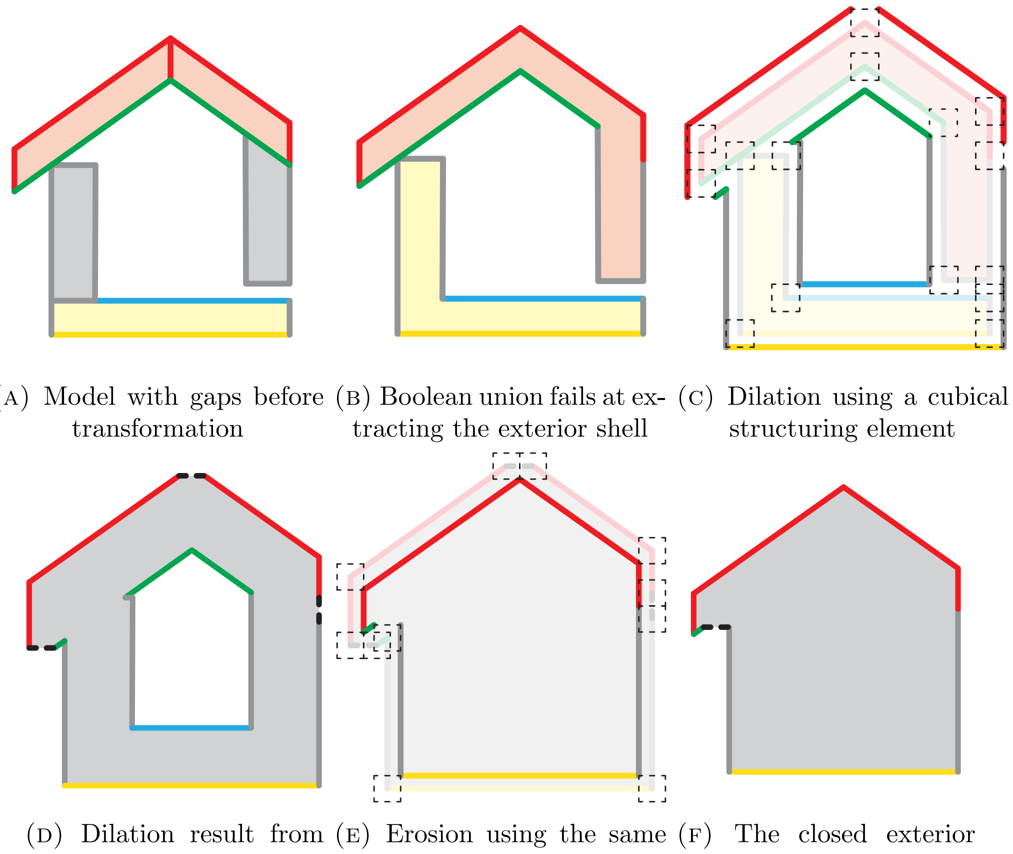

(A) Dilation (B) Erosion (C) Closing

FIGURE 3.8: Morphological operations using a disk as structuring element and the grey geometry as input (Source: [43])

The small gaps in the building are removed by applying a slightly altered version of morphological closing. In this version the interior geometries are removed after dilation and before erosion. By doing so the gaps do not reopen during the erosion operation. Figure 3.9 shows the process of extracting the exterior shell step-by-step. One may notice that a cubical structuring element is used. A spherical structuring element has a negative influence on computation time, due to large amount of linear geometries needed to approximate the sphere. Furthermore, a sphere closes concave parts of the geometry spherically (see Figure 3.8c). The spherical closure geometry would have to be approximated by many triangles, unnecesarily increasing the data size, while it is also believed to be an uncommon geometry for buildings. The cube was chosen as the assumption was made that its shape represents buildings in general the most, as all angles are 90°. There are however two more aspects of importance aside from the shape of the structuring element. These are the size and the orientation.



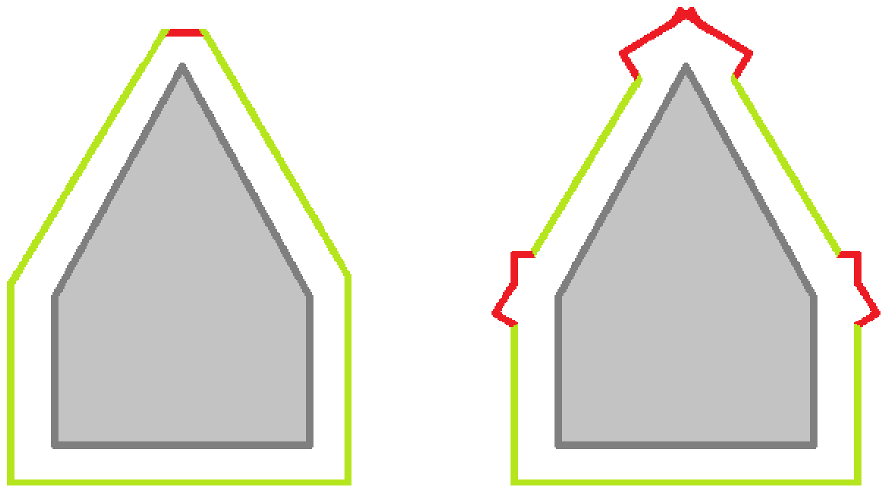
(A) Model with gaps before transformation (B) Boolean union fails at extracting the exterior shell (C) Dilation using a cubical structuring element

(D) Dilation result from which the interior geometry can be removed (E) Erosion using the same structuring element (F) The closed exterior shell with most semantics and geometries intact

FIGURE 3.9: Visualisation of the exterior shell extraction with closing

The size of the structuring element determines the maximum gap size which can still be fixed by the closing operation. From practical experience for the cases that require closing, a width of 100mm to 300mm is found to be reasonable. A larger width would soon close narrow windows, whereas a smaller radius might introduce very tiny or nearly degenerate geometries. Although 300mm can be used for all conversions, this would also introduce closing geometries for buildings that do not require closing. As such the size of the structuring element is a user setting in the prototype.

The orientation of the structuring element can be constant for the whole model, or for each face it can be rotated such that the it is aligned with the face [40]. The dilation of a simple geometry for both concepts is depicted in Figure 3.10. In this thesis it was found that, although the offset is constant for the face aligned concept, artefacts are created whenever there is a convex corner with an angle between 0° and 90°, or between 90° and 180°. Since these artefacts do not disappear during erosion, face aligned structuring elements are not used. Instead a constant orientation of the structuring element is used.


(A) Dilation with a structuring element having a constant orientation

(B) Dilation with face aligned structuring elements

FIGURE 3.10: Two concepts for orienting the structuring element when dilating a simple geometry

For the orientation the assumption is made that the vertical direction is always predominant for buildings, e.g., walls are vertical. This assumption fixes the cube with two faces perpendicular to the z-axis, leaving only room for rotation around the z-axis. The optimal orientation is then determined using the normals of the remaining faces from the IFC model. Faces with 0, 90 or 180 degrees rotation between the normals in the x-y plane are grouped. The orientation of the group of faces with the largest combined area is used. Least squares fitting of a plane though all faces is not used as it results in a sort of average of the input planes, while the most common orientation is always an orientation which is actually geometrically present in the IFC model. Figure 3.11 shows the flow diagram for the process of extracting the exterior shell including closing operation.

Just like with the Boolean set operations the semantics and the converted CityGML semantics of the faces need to be maintained. During dilation and erosion the semantics of a face are transferred only to the parallel face which is on the positive side of the original face. If there are two adjacent coplanar faces with different semantics, their dilation would overlap. This is resolved by removing the semantics of the overlapping

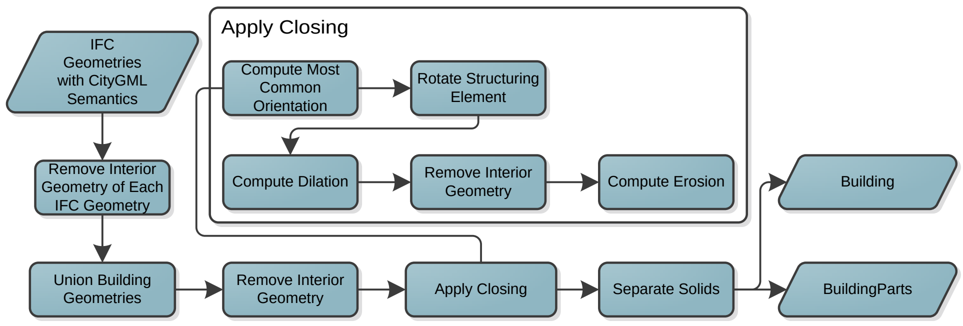

IFC Geometries with CityGML Semantics
Remove Interior Geometry of Each IFC Geometry
Union Building Geometries
Remove Interior Geometry
Apply Closing
Separate Solids
Building
BuildingParts
Apply Closing
Compute Most Common Orientation
Rotate Structuring Element
Compute Dilation
Remove Interior Geometry
Compute Erosion

FIGURE 3.11: Workflow diagram for extracting the exterior shell using union and closing operations. This is a part of Figure C.2

section, since during erosion the unassigned faces will disappear. An example of how the semantics are maintained can be found in Figure 3.9c.

To differentiate the semantics of a face, also attributes may be used other than the boundary surface type. For example the type or ID of the IFC object may be used so that the faces of an 'IfcSlab' are not merged with those of an 'IfcWall' even if their CityGML mapping is both 'WallSurface'. The newly created faces to close the gaps are often without semantics. The methodology for assigning semantics to those faces is described in Section 3.3.2.

3.2.2 Incorporation of BuildingInstallations

The CityGML standard states that the geometry of 'BuildingInstallations' in LoD3 should be generalized with respect to its shape in reality. This is in contrast with all the other geometries which are to represent the full architectural extent [11]. What the generalized shape should be is however not defined in the standard. According to Special Interest Group 3D (SIG3D) 'BuildingInstallations' should be constructed using line segments, faces and solids [36]. A fence for example should be constructed using only lines. These generalizations are found to be out of the scope of this thesis. The only operations that can be considered a generalization are that the interior geometry is removed and the geometries unioned. Solids in LoD3 should not overlap each other. Therefore the 'Buildings' and 'BuildingParts' generated using the methodology of Section 3.2.1 are cut from the 'BuildingInstallations' using the Boolean difference operation. The workflow for this process is shown in Figure 3.12.

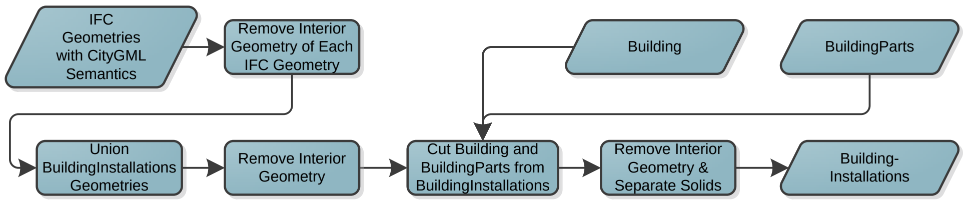

IFC Geometries with CityGML Semantics
Remove Interior Geometry of Each IFC Geometry
Building
BuildingParts
Union BuildingInstallations Geometries
Remove Interior Geometry
Cut Building and BuildingParts from BuildingInstallations
Remove Interior Geometry & Separate Solids
Building-Installations

FIGURE 3.12: Workflow diagram for processing BuildingInstallations. This is a part of Figure C.2

3.2.3 Concepts for Computing the Exterior Shell Geometry

Morphological closing is not a flawless method for closing the geometry. Although the process is automated, human evaluation is still required to determine whether the actual exterior shell has been extracted. Also local concave sections may be closed even when no closing was required, possibly creating faces without semantics. As such other concepts for computing the exterior shell have been generated and evaluated in this thesis at differing stages of development during this thesis. The concepts are divided into three groups: best matching model, convex hull based methods and closing methods. Techniques which require the rasterisation of the geometry, for example like the method published by Snyder et al. [44], are not considered due to the inherent loss of accuracy [45]. At the end of this section a trade-off is made between the concepts and explained why morphological closing is the best concept for the conversion of IFC data to valid CityGML data.

3.2.3.1 Template Matching

Template matching concept uses a large library with pre-made CityGML building models. During the conversion, the geometry and semantics of each building in the library are matched with the input's IFC explicit geometries with CityGML semantics. Rainsford and Mackaness [46] implemented template matching for the footprints of buildings. Each building model in the library should be scaled and rotated such that their match is optimal. Dufour et al. [47] showed how the size location and orientation can be determined using template matching.

This concept did not leave the early design phase and is not implemented due to the expectation that either the output would have a low amount of detail, or the library would have to be unrealistically large. An improvement to the concept is to match multiple smaller building sections and unioning them. However this is expected to result in the same need for closing which the Boolean union of all IFC geometries has. Kada [48] avoided the latter problem by decomposing the building only in horizontal directions, such that the sections do not overlap.

3.2.3.2 Convex Hull Based Methods

Convex hull based methods start from a geometry which is known to be a valid 2-manifold; the convex hull. The algorithms then try to approximate the exterior shell by decreasing the volume of the hull each in its own way.

Shrink Wrapping When an object is shrink wrapped, it is sealed within an air-tight membrane after which the air between the membrane and the object is removed. Kobbelt et al. [49] published a shrink wrapping method for remeshing polygonal surfaces. For the purpose of extracting the exterior shell, not all the air should be removed. Instead an equilibrium should be achieved between the surface tension of the membrane and the vacuum force.

In an initial implementation during this thesis, it was found that the required magnitude of the vacuum force differs depending on the location of the gap in the model. If a gap on one side is nicely closed another may need a larger or smaller vacuum force. Also determining rules for stopping the shrink wrapping process locally proved to be difficult. This together with the expected difficulties when handling semantics and when shrink wrapping geometries with genus-1 (homeomorphic to a doughnut) or higher, lead to abandoning of the shrink wrapping concept.

Heuristic Carving Heuristic carving is developed by Zhao et al. [50] to repair LoD2 building solids. The method uses a constrained Delaunay tetrahedralization of the input geometry and its convex hull. The carving process works by chipping away one tetrahedron at a time. A tetrahedron can only be removed when it meets the requirements. The removal must for example not result in an invalid geometry of the remaining solid and also the planar nature of buildings is taken into account. The concept is depicted in Figure 3.13.

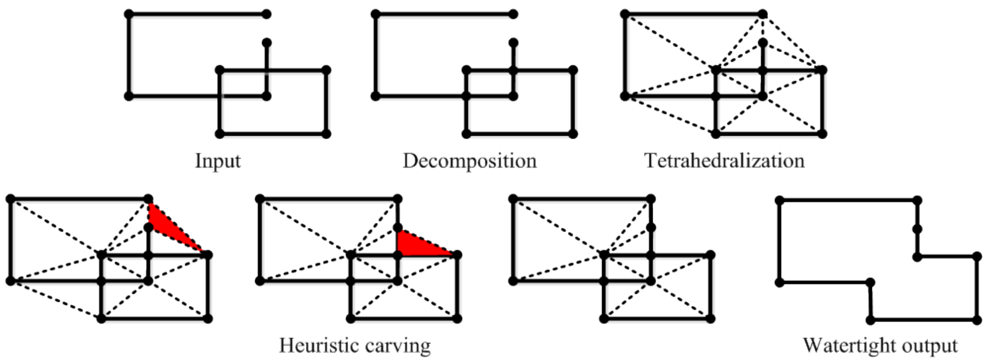


FIGURE 3.13: Heuristic carving a method for repairing geometry (Source: [50])

A key advantage of heuristic carving is that the concept is less influenced by thresholds like the size of the structuring element in morphological closing. Also since the

the concept aims at repairing solid geometries, it is less prone to errors in the input. Unlike morphological closing, this method can guarantee that the output is 2-manifold. However, this is not necessarily an advantage since the exterior shell could inherently be a non-manifold. When there is a non-manifold edge in the actual exterior shell, heuristic carving could leave the tetrahedron incident to that edge in order to keep the geometry 2-manifold. Yet placing a tetrahedron to repair a non-manifold is not an optimal method. Better methods are described in Section 3.3.

Although the range of gap sizes that can be closed is large, one fundamental aspect of the current implementation is that it keeps all the geometries from the input. Something which is exactly the opposite of what is needed, but the main principals of heuristic carving are still applicable. The method was not researched further as it is still very novel and only published at the end of this thesis project. However the method is a potential candidate to replace morphological closing in future work.

3.2.3.3 Closing Methods

Unlike convex hull methods, closing methods do not necessarily start from a valid geometry. Closing methods use all the input geometries and transformed them such that the exterior shell is created. Morphological closing as used in this project and described in Section 3.2.1 also falls into the category of closing methods.

Vertex normal closing The main downside of morphological closing is that sharp concave corners are closed by the structuring element regardless of whether closing is needed. In 2D these sharp corners can be maintained by dilating the polygon using the vertex normal. The vertex normal in 2D can be computed by averaging the two normals of the incident edges. For each edge an dilation area can be calculated by making a trapezoid using the edge and a parallel edge which endpoints lay on the vertex normals. Figure 3.14 shows how the four trapezoids from four edges dilate the polygon. The Boolean union of the polygon and all dilation trapezoids results in the dilated polygon with all corners intact.

The intersection between the parallel edge and the vertex normal can be computed using the coordinates of the original vertex, the dilation offset distance, and the scalar product between the unit vertex and edge normal vectors. The formula for calculating the new vertices is given in Equation 3.4. The geometry must not be degenerate as otherwise the new vertex would be at infinity.

$$\begin{gathered} v_2 = v_1 + \hat{n}_{vert} \cdot \frac{offset}{\cos\theta} \\ Where, \cos\theta = \frac{\bar{n}_{vert} \cdot \bar{n}_{edge}}{\|n_{vert}\| \cdot \|n_{edge}\|} \end{gathered} \tag{3.4}$$

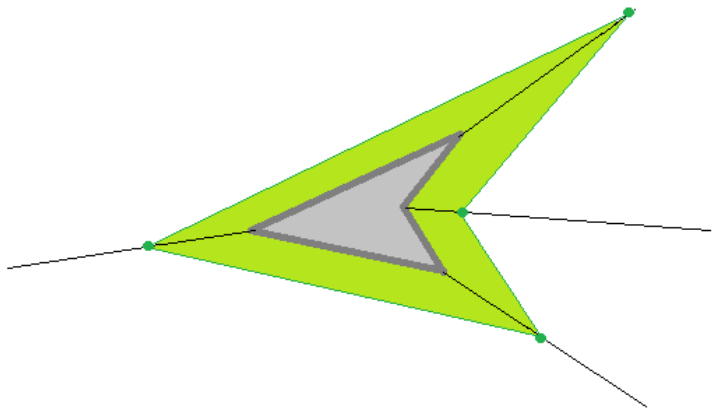
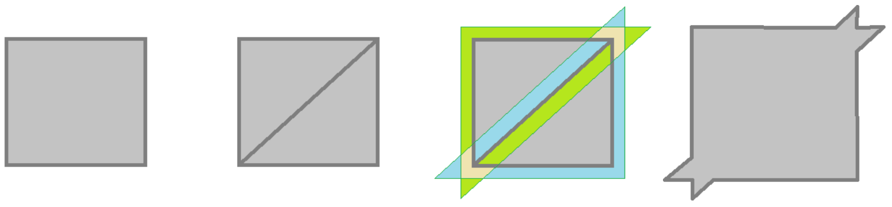


Figure 3.14: Vertex normal dilation in 2D maintains sharp corners, four trapezoids are created for the dilation

This concept works in 2D, but in 3D there is one major issue. A vertex normal does not necessarily exist, or how Jin et al. [51], who made a comparison of algorithms for vertex normal computation, stated it: "none of the available algorithms are particularly good." From research during this thesis the same conclusion was drawn when limiting the geometry to building shapes.

There is however a shape for which the vertex normal can always be determined, which is a tetrahedron. As such, after merging all the IFC geometries a tetrahedron decomposition could be applied to the resulting solids. Each tetrahedron can then be dilated safely, but artefacts arise when a vertex is dilated by multiple tetrahedra (see Figure 3.15). These artefacts remain present even after eroding the geometry. Though averaging all new positions for a vertex yields slightly better results, the artefacts cannot be prevented when there is a small gap. The artefacts which arise due to morphological closing are more predictable and less invasive making it the better method for closing.



(a) A rectangle to be dilated (b) Triangulation of the rectangle (c) Dilation of each individual triangle (d) The resulting dilated shell has artefacts due to the intersecting geometries

Figure 3.15: Artefacts occur when applying vertex normal dilation to individual triangles or tetrahedra in 3D

Scaling Geometries can be scaled about their individual centroids in order to closed the gaps between them. However, a solid never closes itself during a scaling operation.

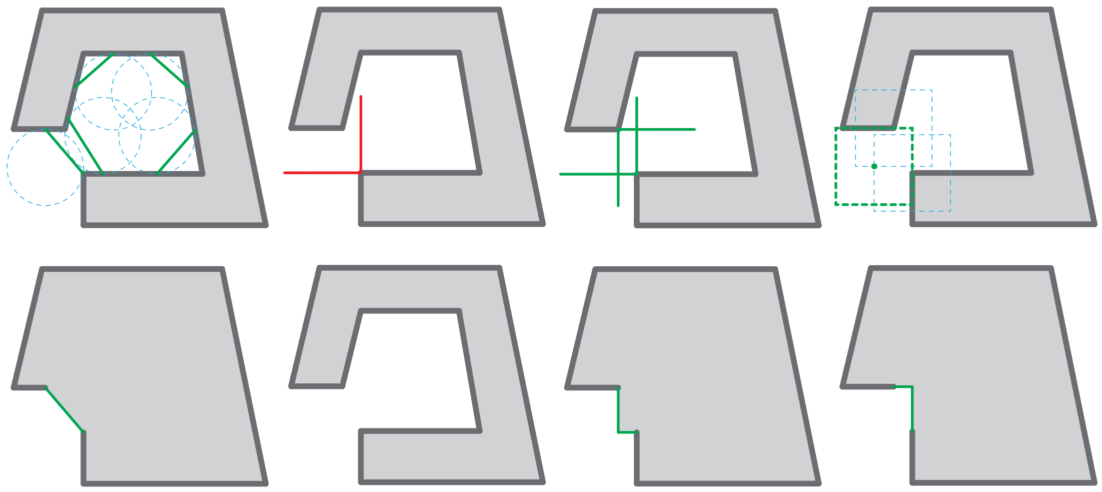
That is, if there is one solid with a gap, no matter how much the solid is scaled, the gap just scales with it. Therefore a convex decomposition is required which has the same downside as it does for vertex normal closing.

Patching Local operations can be applied instead of dilating and eroding the whole geometry. The technique is named patching as surface patches are to be applied between geometries when the distance between them is below a certain threshold. The concept is very similar to that of alpha-shapes [52] and a similar implementation is published by Bischoff et al. [53], but with the intent to repair geometries.

To explain the concept, imagine that a building is completely covered in snow and you are given a spherical snow shovel to remove as much of the snow as possible. Due to the size of the shovel, not all the snow can be removed. The remaining snow is where the linear patches are applied. Note that the implementation differs from the explanation which is why the concept does not belong to the convex hull methods.

The advantage over morphological closing is that the input geometry is left untouched. Semantics would therefore not need to be maintained during Boolean operations. There are also disadvantages with this concept compared to morphological closing. Not all concave corners should be patched, for example the corners of a room or two Building Parts having each their own exterior shell should not be patched together when their geometries are close. Figure 3.16a shows how the patches are applied and what the resulting exterior shell would be. A proper exterior shell is extracted, however the shape of the closing geometry is not optimal.

The structuring element of morphological closing can be oriented in the most common orientation, similarly surface patches can be limited to the same orientations. By doing so recursively, the result in most cases is the same as morphological closing. However, as is shown in Figure 3.16b in some cases the correct exterior shell is not found. The difference occurs in morphological closing due to removal of the interior geometry while the geometry is dilated. Improvements to the concept may close the gap, but will never yield better results than morphological closing. An example of such an improvement is shown in Figure 3.16c.



(A) Freely oriented patches

(B) Axis aligned patching can fail to extract the exterior shell

(C) Axis aligned patching with additional patches at every vertex (only shown for the relevant corner)

(D) Exterior shell extracted using morphological closing

FIGURE 3.16: The results of patching compared to morphological closing

Procedural Modelling A remaining question may be: “Can the semantics not be used to extract the exterior?” The answer to which is yes. In fact semantics are used a lot when generating the explicit geometry. Subtypes of the relation ‘IfcRelConnects’ indicate for example whether a wall stands on top of a slab, or a roof is supported by a wall [12]. Given that there is no relation like ‘almost connects to’, only the object types, as they are used in the semantic mapping in Section 3.1.2, could potentially be useful.

Procedural modelling applies geometric operation in a defined sequence when the conditions for it, based on the shape and semantics, are met. A procedure may be that when the minimum vertical distance between a wall and a roof is lower than a certain threshold, the wall is extruded upwards till it hits the roof. However parts of the wall may as a result protrude through the floor of a different section. This requires additional rules. Procedural modelling was first implemented to model plants by Měch and Prusinkiewicz [54], but it is also used for modelling building architecture instantly by Wonka et al. [55]. Due to the complexity, estimated risk of uncaught cases and the overall feasibility, this concept was not evaluated in an implementation.

Morphological Closing Morphological closing, as described in Section 3.2.1, is selected as the most suitable concept for finding the correct exterior shell. Table 3.2 provides an overview of the described concepts and how they are evaluated relative to each other. Each of the concepts are rated from – to ++ for several criteria. A higher ranking means that the concept is expected to perform better on that criteria. The criteria are defined as follows:

- **Feasibility / Complexity -** The likelihood that the concept is capable of extracting the exterior shell and the complexity of the implementation.
- **Predictability -** How well the results from a concept can be predicted.
- **Number of Artefacts -** The number of places where geometry added while it was not needed (fewer places gives a higher score).
- **Shape of Artefacts -** How well the shape of the artefacts fits the building geometry.
- **Amount of Detail -** The capability of a concept to maintain or create high detail models.
- **Transferability of Semantics -** The ease at which semantics can be transferred from the input of the output.

Table 3.2: Qualitative trade-off of all concepts for extracting the exterior shell

Concept	Total Score	Feasibility / Complexity	Predictability	Number of Artefacts	Shape of Artefacts	Amount of Detail	Transferability of Semantics
Morphological closing	++ 28	++	++	+	++	++	+
Heuristic carving	+ 26	+	+	+	+	++	++
Patching	+ 26	+ -	++	+	+	++	++
Vertex normal closing	+ - 22	+	+ -	+ -	+ -	++	+
Procedural modeling	+ - 20	-	- -	+ -	+	++	++
Best matching model	- 18	-	- -	++	++	-	+
Shrink wrapping	- - 16	-	- -	+ -	+ -	+	+ -
Scaling	- - 16	- -	-	- -	- -	++	++

From Table 3.2 can be concluded that morphological closing offers the best methodology for obtaining an exterior shell without gaps. The advantages of morphological closing are summarized in the following list:

- The reliability is high as whenever there is a gap smaller than the width of the structuring element, the gap will be closed.
- Barely any semantics or geometric detail is lost due the operation. The only loss occurs due to artefacts.
- Only a small amount of artefacts is created at predictable locations, such as underneath roof overhang and round or triangular attic windows.
- The artefacts that are created are aligned with the building model.
- The resulting model is guaranteed to only contain solids, as Boolean operations cannot create geometries of a lower dimension than the input.
- It is possible to transfer the IFC semantics to the output.

3.3 Producing a CityGML and ISO Conform File

The methodology described in Section 3.1 and Section 3.2 covers the whole conversion process for the generation of CityGML LoD3 building models. However, the closing operation from Section 3.2.1 could potentially create faces without semantics and the validity of the geometry is not guaranteed during the transformation. As such the resulting model needs to be validated before it can be written to a CityGML file.

The refinement methodology is again divided into a semantic and a geometric part. Section 3.3.1 provides the geometric refinement for which a detailed flow diagram can be found in Appendix C Figure C.3. The semantic refinement is documented in Section 3.3.2 and its flow diagram in Appendix C Figure C.4.

3.3.1 Geometric Refinements

In order for the geometry in the output file to comply with the ISO19107 specification the geometry in the file needs to be 2-manifold. In this section the methodology is given for ensuring compliance with ISO19107 and how in future use of the data, issue due to floating-point arithmetic are limited.

The geometry was implicitly stored in the IFC file and is ultimately transformed into CityGML B-rep geometry. Due to these processes the geometry may have coordinates with a large number of decimals. These numbers will be rounded when written to a file. When doing so the geometry might change ever so slightly, possibly making the geometry invalid. To fix this problem all the coordinates are rounded and then the geometry is made valid again. The method for making geometries valid is described later in this section. This is an recursive process as making the geometry valid may introduce new vertices. Once all the coordinates are final and the geometry valid, the geometry is cleaned to limit the occurrence of degenerate faces and semantically refined before writing it to a file. Figure 3.17 shows a simplified overview of the geometric refinement.

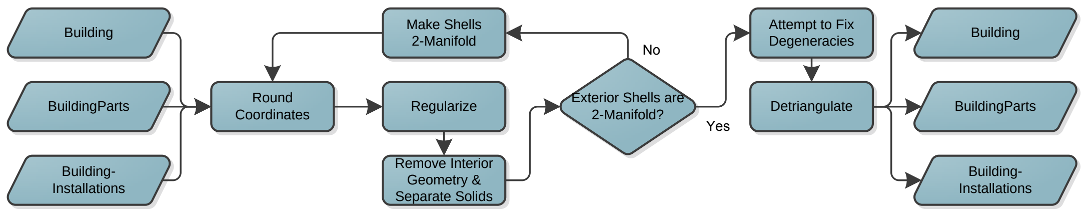


FIGURE 3.17: Simplified workflow diagram of the geometric refinement process

The process for ‘BuildingParts’ and ‘BuildingInstallations’ is slightly different to that of the ‘Building’, as rounding of the coordinates the geometries may be overlapping.

Therefore these overlapping volumes need to be removed using Boolean operations (see Figure C.3).

Regularization is a procedure which removes all dangling geometries, these are unwanted as they do not bound interior of the solid (see Figure 3.18). This is done by taking the closure of the interior. After rounding the coordinates and regularizing the geometry new inner shells may have been created or solids may have split in two (see Figure 3.19). They have to be removed and separated before it can be checked whether the shells are 2-manifold or not. The removed dangling geometries can be stored as 'BuildingInstallations'.

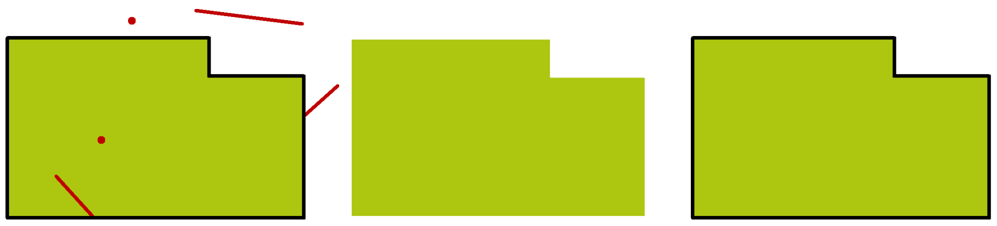

(A) Geometry before regularization (B) The interior is extracted from the geometry (C) The closure is applied to the interior where all geometries bound the interior

FIGURE 3.18: Regularization is applied to remove dangling geometries

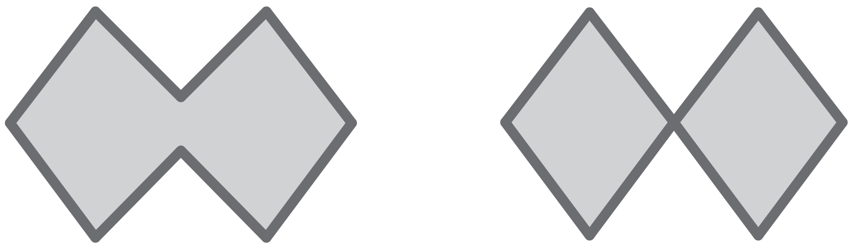
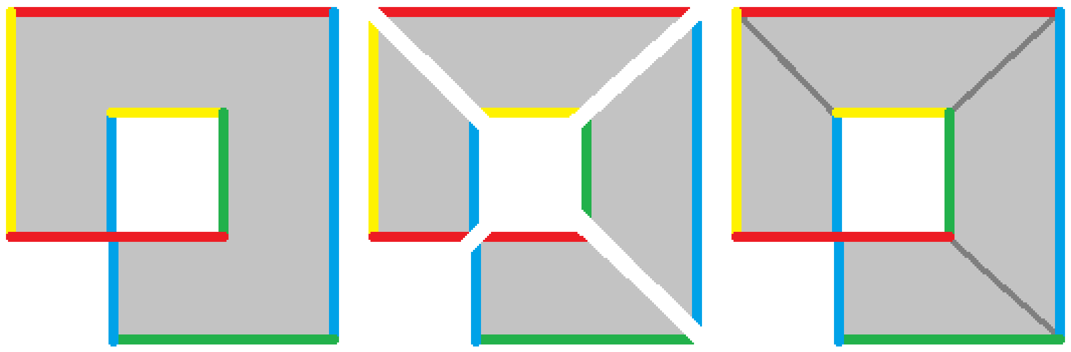

(A) Geometry before rounding coordinates consists of one volume (B) After rounding two vertices collapsed and split the volume in two

FIGURE 3.19: Rounding of coordinates may split volumes in two

3.3.1.1 Repairing Non-Manifold Vertices and Edges

The definition of a 2-manifold shell is given in Section 2.3.2. In this section only non-manifolds with a connected interior are discussed as the other manifold cases would have been split in previous processes.

To repair a non-manifold solid appropriately without changing the exterior geometry ISO19107 advices to split the solid into multiple parts. In the event of non-manifold vertices or edges, a convex decomposition can be applied to the geometry as is depicted in Figure 3.20. After the decomposition a set of solids is created where each face is either a semantic boundary surface or a shared boundary between two solids. The geometry of the boundary faces are not influenced by this operation and still represent the full (non-manifold) exterior shell of the building. However, for analytical purposes the solid geometries, which are stored separately as one 'gml::CompositeSolid', are now guaranteed to consist of only 2-manifold solids.



(A) A non-manifold geometry with semantic boundary faces

(B) A convex decomposition is applied ensuring that each solid is 2-manifold

(C) The boundary faces remain unchanged, yet the solid geometry is now valid

FIGURE 3.20: A non-manifold exterior can be decomposed in a CompositeSolid with multiple 2-manifold solids. The grey edges in Figure C represent shared faces of adjacent solids which do not have semantics

There are two other methods for repairing non-manifold geometry, both of which change the shape of the geometry locally and are thereby less optimal. The first developed by Mäntylä [56] who proposed a method for repairing non-manifolds in which the non-manifold vertex or edge is duplicated and moved slightly away from each other. The method is able to maintain the geometry as one solid. When a non-manifold edges belonging to a wall is repaired using this method, the wall become ever so slightly less vertical. This is believed to be an undesirable side-effect as the generated CityGML semantics depend on the surface orientation. Therefore, this method should not be used.

A second method is developed during this thesis. A non-manifold vertex can be fixed by cutting the solid with a small element surrounding the vertex. To create this element, first for every edge incident to the vertex an offset point on the edge is calculated at a minimal distance ε away from the vertex. The cutting element can then be constructed by taking the convex hull of all these points together with the original vertex as is depicted in Figure 3.21. In case ε is larger than the distance to the adjacent vertex, the vertex is used instead.

Since the procedure creates new vertices, it needs to be repeated until all vertices are manifolds. To speed up the process a search can be made for non-manifold edges. Fixing a non-manifold edge is similar to fixing a single vertex. The difference is in that cutting element is constructed using the offset points from both the vertices of the edge and the vertices themselves for the convex hull.

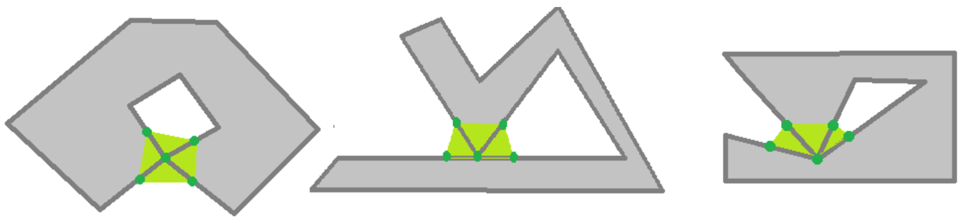


FIGURE 3.21: The (green) element to be removed to make a shell 2-manifold is constructed using the convex hull

The method for repairing non-manifolds by subtracting small elements from the geometry has the small advantage that the building geometry can be represented by only one solid. However, it does slightly change the shape of the building locally and may require multiple iterations before the geometry is 2-manifold. Therefore, doing a convex decomposition is the recommended method for repairing non-manifold solids.

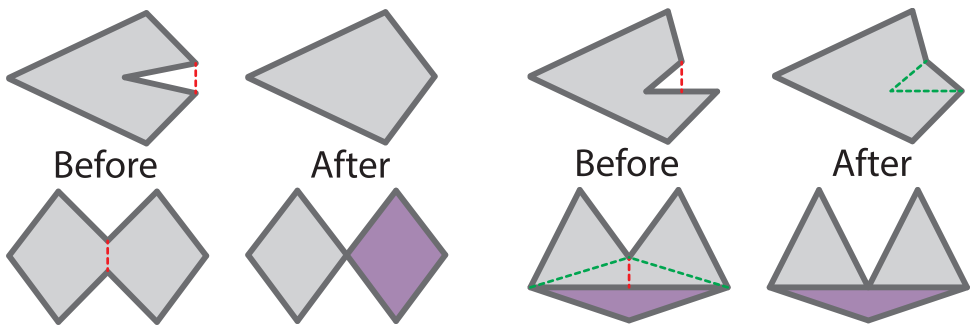
3.3.1.2 Finding and Repairing Degenerate Geometries

A robust procedure to eliminate degenerate faces from a triangular mesh was published by Botsch and Kobbelt [38]. However, it was decided not to use any operation that changes the shape of the geometry. The reason for which is that such kind of operation may have an cascading effect where a large number of small changes to the geometry may have a big impact on the shape.

As is explained in Section 2.3.2, there are two types of surface degeneracies; caps and needles. Another distinction can be made between perfect and nearly degenerate geometries. For perfect degenerates the distance 'd' between the vertex and the edge or other vertex is zero, while for nearly degenerates this distance is almost zero. Perfect degenerates are the only forms of degeneracies that can be removed in a robust way without changing the shape.

Perfect needles are basically two vertices on top of each other and can be removed by collapsing the (virtual) edge between them. By doing so the left and right side are split in separate faces and/or the triangles formed by the (virtual) edge are removed. Perfect caps can be removed by first creating a triangle using the related edge and vertex. Since this triangle is completely collinear it effectively does not have semantics and is always coplanar with the other face incident to the longest edge. By merging the triangle with the face on the opposite side of the longest edge the perfect cap is removed. Examples of both operations are shown in Figure 3.22.

In case one would want to maintain a triangular mesh, an attempt can be made to remove perfect caps by applying an edge flip to the longest edge of the degenerate triangle. This operation is also only allowed on coplanar faces with the same semantics. It is however possible that due to this operation new caps are created. When that happens, the cap cannot be removed by means of an edge flip. Attene [57] presented a



(A) Prefect needle fix (B) Prefect cap fix

FIGURE 3.22: Methods for removing perfect surface degeneracies. For perfect degeneracies the length of the dashed red line is zero

recursive method using edge flips on a growing region until all caps are solved. Although no coordinates are moved in this method, a smooth surface is assumed which is why the method is not applicable in this project.

It is important to note that even if all perfect degeneracies are removed there is still a possibility that when floating-point arithmetic is used, parts of polygons may collapse and become invalid. Therefore to reduce the likelihood of those situations, all the coplanar faces with the same semantics are unioned into larger faces. During this detriangulation no invalid faces must be created. Although this operation reduces the number of nearly degeneracies, it definitely does not guarantee that all of them are removed. Furthermore, besides degeneracies on the boundary of a solid also the interior of the solid may degenerate due to floating-point arithmetic. However, no perfect degeneracy occur in the interior due to the rounding of coordinates and subsequent fixing of the geometry.

3.3.2 Semantic Refinements

During the morphological closing operation and while making the exterior shells 2-manifolds, faces may have been created without semantics. Although generating semantics for these faces is not necessary for compliance with the CityGML standard, generating semantics might be required for certain applications. Since CityGML aims at the use in many applications a methodology was developed.

The semantics are assigned based on the normal direction of a face and whether it is directly or indirectly connected to certain boundary surfaces. By indirectly connected is meant that the face can be connected to a boundary surface through other unassigned faces having the same type of normal direction. The type of normal refers to whether it is up, horizontal or down. The decision process can be seen in Figure 3.23. ‘Opening’ semantics are only assigned when the unassigned faces are completely contained by ‘Openings’.

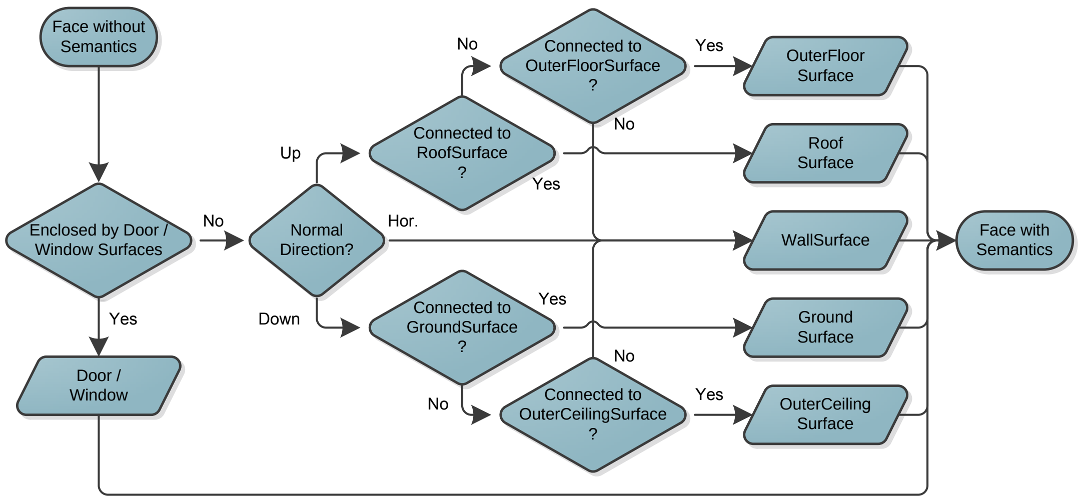


FIGURE 3.23: Workflow diagram for determining the semantics for faces without semantics

Alternatively the faces without semantics could all be assigned to 'ClosureSurfaces', however that would imply that there is connection between the interior and exterior of the building allowing for the transit of people.

Window and door 'Openings' in CityGML are required to be part of a boundary surface like for instance a 'WallSurface'. The boundary surface is not required to have geometry, which is useful in the case of an all glass facade. Therefore, in theory every 'Opening' face could be assigned to its own boundary surface, but this is believed to be very undesirable. Instead all faces that are connected to each other and have the same semantics are grouped and will be written to the file as one 'MultiSurface'. Here the semantics aside from the boundary surface type may also be for example the type or ID of the IFC object from which the face originates.

For an 'Opening MultiSurface' an adjacent 'BoundarySurface' is found for it to be part of. If there is none, a 'BoundarySurface' is found through other connected 'Openings'. In the unlikely case that the object is made entirely out of doors and windows, a 'WallSurface' without geometry is assigned.

After the geometric and semantic refinements the CityGML file can be written (see Figure 3.24). Due to the refinement processes the output fully adheres to both the CityGML and ISO standards, while also the chances of future complications due to floating-point arithmetic are reduced.

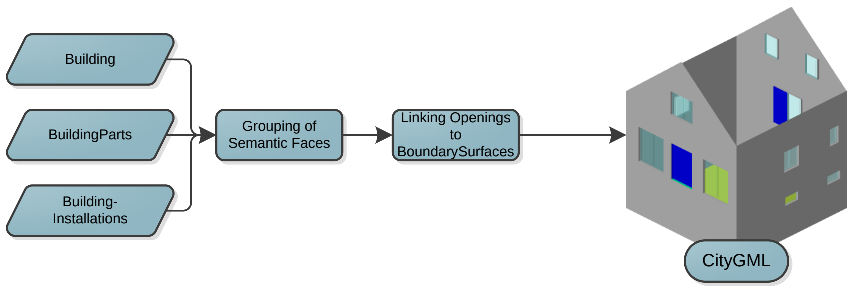


FIGURE 3.24: After the semantic refinement has been performed the model can be written to CityGML

Chapter 4

Implementation and Experimental Results

The main research question of this thesis is whether it is possible to do the conversion from IFC to valid CityGML data which keeps all relevant geometry and semantics. A complete conversion methodology has been developed which is proofed by a prototype implementation. The implementation of the prototype was also used to develop the conversion methodology. In this chapter the implementation of the prototype is described. The requirements for an IFC model to be eligible for conversion are given and experimental results are validated and the quality is evaluated.

Before the implementation is described, first the framework on top of which it is build is discussed in Section 4.1. The description of the development framework and the spatial data structures provide background information on why certain engineering decisions are made in the implementation. The implementation itself is described in Section 4.2. This is followed by the implementation of the conversion to LoD4 in Section 4.3. The description of the framework and the implementation is intended to aid others in the implementation of the methodology. With the implementation explained the focus is then put on the experimental results. To evaluate the performance of the conversion, first the requirements are described that make an input IFC file sufficiently clean to be converted. This is done in Section 4.4. Resulting models from the conversion are provided in Section 4.5. At the end of this chapter, in Section 4.6, the quality of the experimental results are discussed. Here also the validity of the models is given and the performance of the conversion prototype is evaluated.

4.1 Development Framework

The prototype is divided into two parts. The first part handles the preprocessing, while the second part does most of the conversion. Preprocessing involves reading the input

IFC file, generating the explicit geometry and parsing the semantics. Version 0.3.0 of the IfcOpenShell library is used for this purpose [58]. For the geometric computations IfcOpenShell relies on OpenCASCADE [59]. In this project OpenCASCADE version 6.5.4 is used.

The main part of the conversion is build using the Computational Geometry Algorithms Library (CGAL) version 4.2 [60]. CGAL is used as it provides a large range of geometric functionality which is well documented and guarantees robustness. The separation of the preprocessing and conversion parts is due to conflicts between the OpenCASCADE and CGAL libraries. The results from the preprocessing stages are fed into CGAL using a text file using the Object File Format (OFF) file format which is enhanced with semantics [61]. To avoid problems due to floating-point arithmetic, the 'Exact_predicates_exact_constructions_kernel' geometry kernel is used [62].

The prototype and all of the used libraries are open source. Also, although the prototype is developed on Windows it should also work on other operating systems. The prototype is written in the C++ programming language.

4.1.1 Spatial Data Structures

Two ways to store spatial data from the CGAL library are used in the implementation. These are the Halfedge Data Structure (HalfedgeDS) [63, 64] and the Nef polyhedra [65]. Both have advantages and disadvantages.

4.1.1.1 Halfedge Data Structure

The HalfedgeDS is an edge-centered data structure, where one halfedge is an oriented edge. The halfedge is linked to its previous, next and opposite halfedge and also has a link to the incident face and one related vertex (see Figure 4.1). The CGAL implementation of HalfedgeDS is restricted to 2-manifolds and the faces cannot contain holes [66]. Since the geometries in IFC are not necessarily 2-manifold the data cannot directly be stored in the HalfedgeDS. Also Boolean set operations are problematic as the resulting geometry from two 2-manifolds can be non-manifold (see Figure 4.2). Nevertheless, the HalfedgeDS can easily be extended to support semantics.

4.1.1.2 Nef Polyhedra

Since IFC allows any kind of geometry, the prototype requires a data structure which can handle all geometries. Nef polyhedra allow for the storage of non-manifold geometries and are also closed under regularized Boolean set operations, whereas a B-rep, like the HalfedgeDS, is not always closed under Boolean set operations [67]. For Nef polyhedra in CGAL, finite geometry is stored in sphere maps. Sphere maps were first introduced by Dobrindt et al. [68]. For every vertex the local neighbourhood is mapped onto a

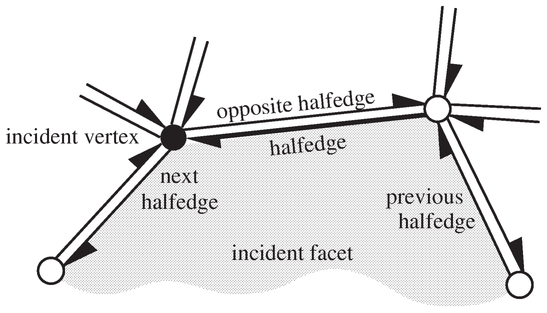


FIGURE 4.1: The connectivity of a halfedge in the HalfedgeDS (Source: [64])

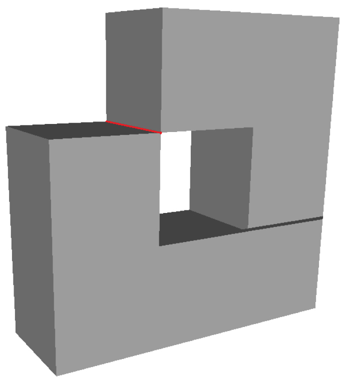


FIGURE 4.2: The union of two 2-manifold solids can result is a non-manifold solid

concentric sphere, which is the sphere map. Incident solids are represented by a Spherical Face (SFace) on the sphere map, incident faces by a Spherical Edge (SEdge) and incident edges by a Spherical Vertex (SVertex) (see Figure 4.3). To make the geometry of a Nef polyhedra more accessible, the more explicit Selective Nef Complex (SNC) representation is synthesised from the sphere maps. The SNC data structure provides a slightly altered, but similar interface to HalfedgeDS for edges, faces, shells and volumes. The SNC data is thereby read-only and cannot be used to alter the geometry.

The data structures of a Nef polyhedra allow for non-manifold geometries, holes and Boolean operations. A 2D example of a Boolean intersection operation is shown in Figure 4.4. Boolean operations can be performed in three steps [65].

1. All candidate vertices are found. Candidate vertices are the original vertices and all points of edge-edge and edge-face intersections.

2. For each vertex the local sphere map in each polyhedron is constructed. Given two sphere maps for a candidate vertex, the sphere maps are combined using spherical 2D Boolean operations.

3. The empty sphere maps are removed and the SNC structure is generated.

The current implementation of Nef polyhedra in CGAL does not support semantics. Enhancing the Nef polyhedra also proved to be difficult since the semantics would have

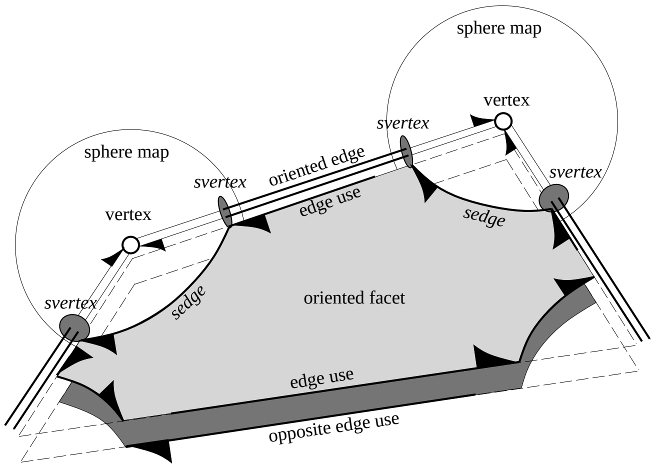
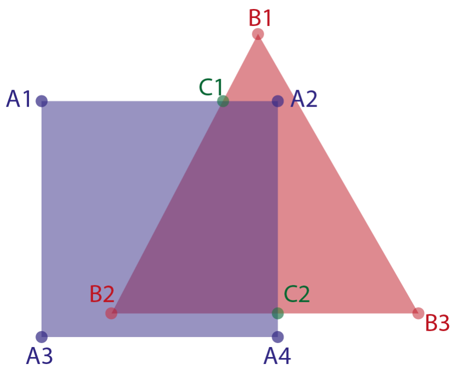


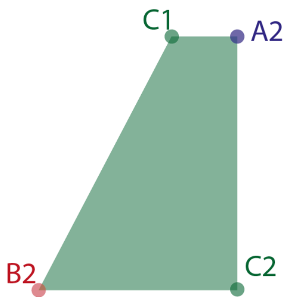
Figure 4.3: Two sphere maps, the SNC representation and connectivity of faces and edges for Nef polyhedra (Source: [65])

to be stored in the SEdges, as the SNC faces are recreated after a Boolean operation. Also 2D Boolean operations on the sphere map would need to be extended to support semantics.

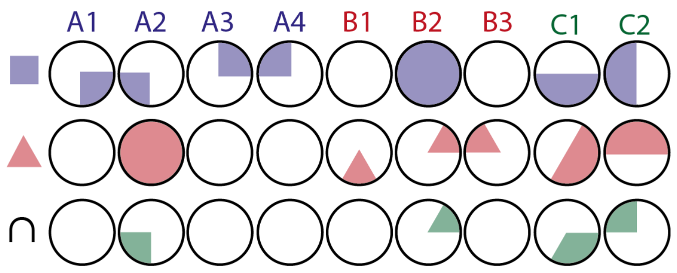
Making any changes to the geometry of a Nef polyhedron by methods other than Boolean operations is complicated. For example to move one vertex, the sphere maps of all connected vertices need to be updated and intersecting geometries need to be handled. Furthermore, when an object is turned into a Nef polyhedron the geometry is simplified; all coplanar faces are unioned. Given that semantics are not supported, also the borders between coplanar faces with different semantics are removed.



(A) Step 1: Candidate vertices are found



(B) Step 3: The remaining sphere maps are used to generate the SNC structure



(C) Step 2: 2D Boolean operations are applied to the sphere maps

FIGURE 4.4: The three step process of performing a Boolean intersection on two Nef polyhedra (Source: [40])

4.2 Prototype Implementation

The implementation differs in certain aspects from the methodology, due to limitations of the development framework. The differences are merely work-arounds and do not influence the validation of the methodology, but they do degrade the quality of the results. The influence on the results is discussed in Section 4.6. The goal of this section is to highlight and explain the differences between the methodology and the implementation and to provide an example of how the methodology can be implemented. The conversion process is briefly discussed again from start to finish while referring to the methodology when the implementation does not deviate from the methodology. The flow diagram in Figure 4.5 provides an overview of the implementation. In Section 4.3 the implementation for converting IFC to CityGML LoD4 is described.

The cause of the deviation between concept and realisation is the spatial data structures provided by CGAL that can only handle either semantics (HalfedgeDS), or 2-manifolds and boolean operations (Nef). The Nef polyhedra are required for the geometric transformation, therefore during that stage of the implementation the semantics and geometry are separated. Afterwards, the semantics and geometry are brought back together. Another complication with CGAL is that the function needed to separate

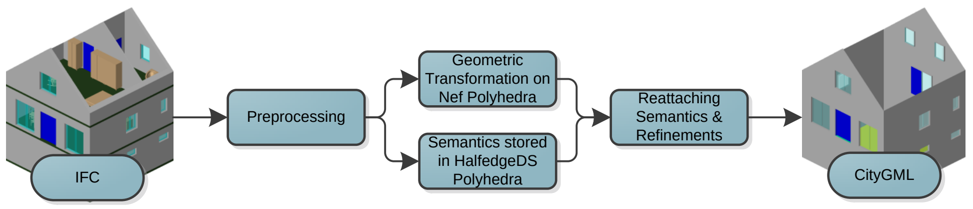

IFC
Preprocessing
Geometric Transformation on Nef Polyhedra
Semantics stored in HalfedgeDS Polyhedra
Reattaching Semantics & Refinements
CityGML

FIGURE 4.5: General workflow diagram for the implementation

volumes and remove inner shells from polyhedra is unreliable when the polyhedron is not a 2-manifold. Therefore every time the inner shells are removed, the geometry is made 2-manifold beforehand by subtracting local elements as is described in subsubsection 3.3.1.2.

In this section no distinction is made between 'Building', 'BuildingParts' and 'BuildingInstallations'. In general, for each of these the processing is as described in this section, with the exception that overlapping geometries should be prevented for 'BuildingParts' and 'BuildingInstallations' as is described in Section 3.3.1. Also, there are no boundary surfaces attached to 'BuildingInstallations'.

4.2.1 Preprocessing

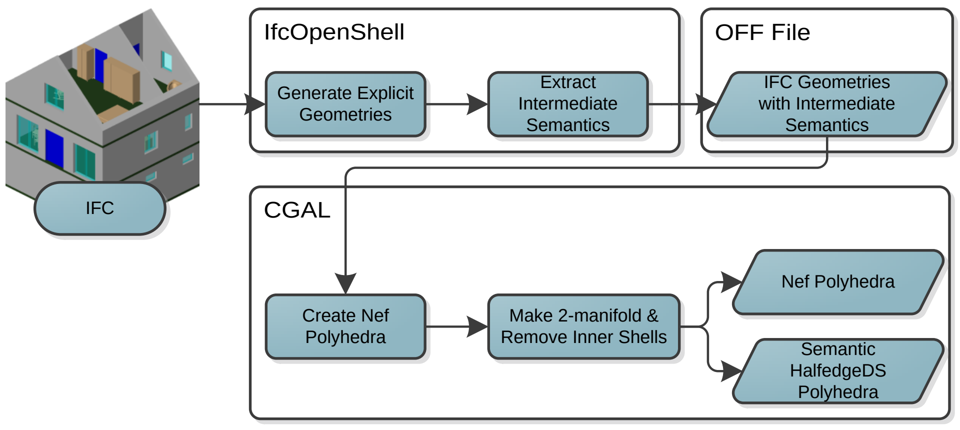
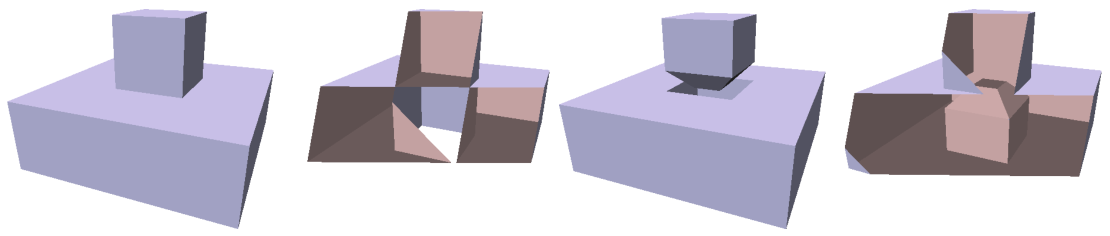

IFC
IfcOpenShell
Generate Explicit Geometries
Extract Intermediate Semantics
OFF File
IFC Geometries with Intermediate Semantics
CGAL
Create Nef Polyhedra
Make 2-manifold & Remove Inner Shells
Nef Polyhedra
Semantic HalfedgeDS Polyhedra

FIGURE 4.6: Flow diagram for the preprocessing stage of the implementation

A flow diagram of the preprocessing stage can be found in Figure 4.6. First the IFC file is parsed by IfcOpenShell, which also generates the explicit geometry. The semantics mapping at this stage does not yet use the normal direction of the faces. Instead intermediate semantic is mapped to each 'IfcObject'. These intermediate semantics are assigned such that they can be translated to the final CityGML semantics using only the normal direction of the faces after the geometric transformation. The triangulated geometries and their semantics then are written to a file in the OFF file format which is enhanced with semantics.

Since the geometry of an 'IfcObject' is not necessarily a 2-manifold and can even contain multiple volumes the object are imported into the CGAL environment as Nef polyhedra. Non solids geometries like surfaces are dilated using a structuring element with a width of 1 cm, such that all geometries are solid during the conversion. In order to be able to reattach the semantics at a later stage, every Nef polyhedron needs to be converted to a HalfedgeDS polyhedron. The semantics can then be stored in the faces of the HalfedgeDS polyhedron. To convert the Nef Polyhedron, it is first made 2-manifold using the methodology described in subsubsection 3.3.1.1. However, instead of subtracting elements from the geometry, they are unioned as not to create gaps between the geometries (see Figure 4.7). The inner shells are also removed and volumes separated before the Nef polyhedra are converted to a HalfedgeDS polyhedron. From the intermediate OFF file the semantics are then attached to the faces of the HalfedgeDS polyhedron.



(A) The input geometry consisting of two volumes: a rectangular doughnut and a cube (B) A cross-section view of the input to show the non-manifold edges (C) The edges are not repaired by subtracting elements, as it creates gaps (D) Instead, the edges are repaired by unioning elements, creating no gaps

FIGURE 4.7: Due to robustness issues the IFC input geometry is made 2-manifold before volumes can be separated or inner shells can be removed. Since subtracting creates gaps only at this stage the elements are unioned instead of subtracted

4.2.2 Geometric Transformation

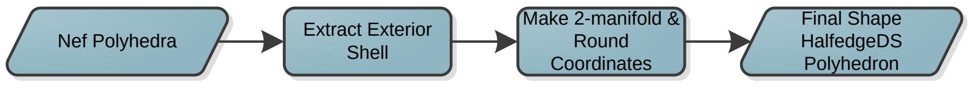

Nef Polyhedra
Extract Exterior Shell
Make 2-manifold & Round Coordinates
Final Shape HalfedgeDS Polyhedron

FIGURE 4.8: Flow diagram for the geometric transformation stage of the implementation

In the geometric transformation stage the Nef polyhedra are transformed into one final shape polyhedra of the exterior shell (see Figure 4.8). All the Nef polyhedra are processed according to the methodology defined in Section 3.2.1 for extracting the exterior shells of the 'Building' and possible 'BuildingParts'. The morphological dilation in CGAL is implemented using the Minkowski sum [69]. The Minkowski sum in CGAL is implemented as the convex hull of the vector sum of all points between two convex polyhedra [70] (see Figure 4.9). Since the Minkowski sum implementation requires convex polyhedra as input, the polyhedra are decomposed into convex parts and dilated using the oriented cubical structuring element individually. After the dilation the convex parts are recombined using the Boolean union [71]. The process is depicted in Figure 4.10.

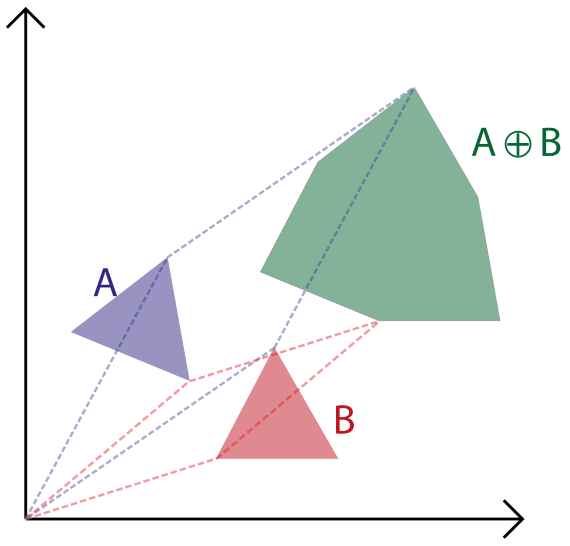
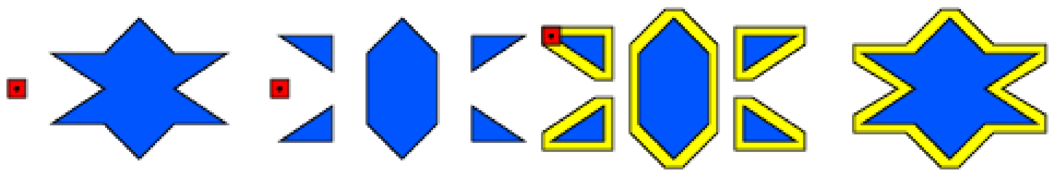


FIGURE 4.9: The Minkowski sum of two convex polyhedra A and B is the convex hull of the vector sum of all point between A and B (Source: [40])



(A) A concave star shape is dilated by a square structuring element

(B) The star shape is decomposed into convex parts

(C) Each of the parts is dilated separately

(D) The dilated parts are unioned to obtain the dilated star shape

FIGURE 4.10: A convex decomposition of both polyhedra into the pairwise Minkowski sums of the convex pieces, and the union of the pairwise sums (Source: [71])

The Minkowski sum cannot be used directly for erosion since the convex hull operation never decreases the volume. As such, the duality between erosion and dilation is used as it is presented in Equation 3.2. First a bounding box is constructed from which the Nef polyhedron is subtracted using the Boolean difference. This represents the complement of the Nef polyhedron. The complement is then dilated using the Minkowski sum. The erosion can now be found by subtracting the dilated complement from the original bounding box geometry using the Boolean difference. According to Hachenberger [69], in the worst case, the Minkowski sum executes in $O\left(n^3m^3\right)$ time, where n

and m are the complexities of the two input polyhedra (the complexity of a Nef polyhedron is the sum of its vertices, halfedges and spherical halfedges). The operation is thereby very time consuming.

After the exterior shell is extracted, the geometry is prepared for writing as is described in Section 3.3.1. Part of this preparation is to make the shells 2-manifold. In order to determine whether a shell with a point-set topology is 2-manifold the sphere map of each of its vertices can be inspected (see Figure 4.11). A vertex is 2-manifold when there is only one SFace on the sphere map, which is a valid polygon according to ISO19107, with the addition that the SFace may not have holes and instead of coplanar the polygon must be on the surface of the sphere. A non-manifold edge may be recognized by checking whether it has more than two incident faces. Alternatively one could check whether there are more than two SEdges incident to a SVertex.

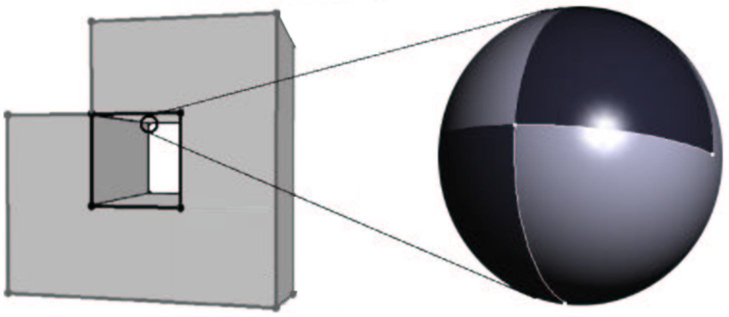


FIGURE 4.11: A sphere map of a non-manifold vertex, the blue SFaces represent the interior of the solid (Adapted from: [67])

Once the geometry is refined, the Nef polyhedron is converted into a triangulated HalfedgeDS polyhedron. At this point the shape is final, no vertices are added or moved in the following processes.

4.2.3 Reattaching and Assigning Semantics

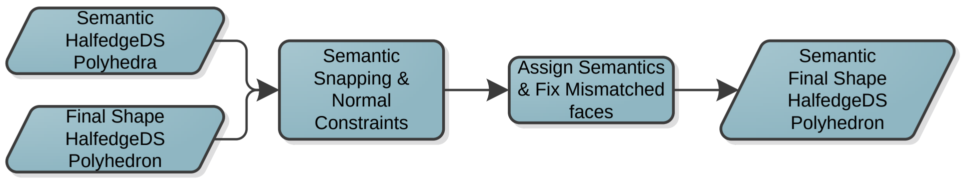

Semantic HalfedgeDS Polyhedra
Final Shape HalfedgeDS Polyhedron
Semantic Snapping & Normal Constraints
Assign Semantics & Fix Mismatched faces
Semantic Final Shape HalfedgeDS Polyhedron

FIGURE 4.12: Flow diagram for the process to get boundary surface semantics attached to each face

Since the shape is final, the semantics can be reattached to the geometry. At the end of this stage a boundary surface is assigned to every triangle (see Figure 4.12). There is now a triangulated HalfedgeDS polyhedron with the final shape of the exterior shell for CityGML and many semantic HalfedgeDS polyhedra representing the same building, but with IFC geometries. Many vertices in the exterior shell correspond to vertices in the IFC geometries. The semantics are to be transferred to the final shape polyhedron. This is done in a process named semantic snapping. For the centroid of every triangle in the final shape polyhedron the nearest semantic triangle is found. The semantics of the semantic triangle are assigned to the triangle from the final shape polyhedron (see Figure 4.13). If the least square distance between the centroid and the semantic triangle is larger than 1×10^{-2} m the triangle is considered to be too distant and semantics are not transferred.

Since this method requires many point to polyhedron calculations Axis-Aligned Bounding Boxes (AABB) trees are used to speed up the computations [72]. The triangles are the leafs of the thee. The leafs based on their bounding boxes are recursively grouped into branches making a tree. By building such a tree the computation time can be reduced for distance queries as whole branches can be ignored if it is more distant then another. As the CGAL implementation only allows for the least distant polyhedron to be returned, a separate AABB tree is constructed for each type of semantic value.

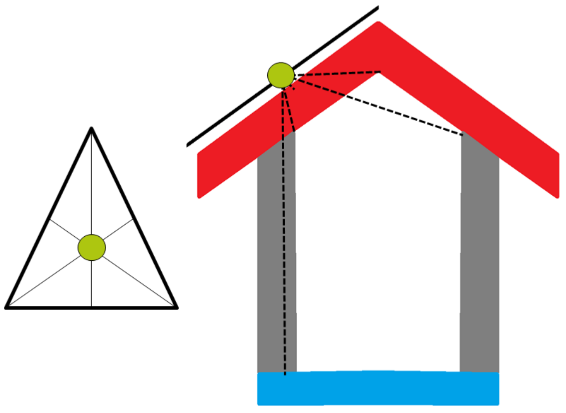


FIGURE 4.13: Semantics are transferred from the triangle (black) for which the least squared distance to the centroid (green) is minimum

It may occur that there are two semantic triangles from different semantic polyhedra for which the distance is minimum and approximately equal. If neither of them is too distant, the semantics are chosen based first on three other criteria. First the triangle is chosen where the angle between the normal vectors is minimum, then the triangle with the largest area and finally if all of the properties are approximately equal then simply the first occurring triangle is chosen. For each of the comparisons a threshold range is set for which the values are considered equal. 1×10^{-2} m is set as the range for which two distances are considered equidistant. The range for the normal direction is 0.175 rad (10°) and for the area to be considered equal a threshold of 1 m^2 is used. The method and thresholds have been determined by trial and error on many different models.

At this stage since the shape is final, the intermediate semantics can be translated into the final semantics. The normal direction constraints are applied to the semantics according to the mapping given in Section 3.1.2. Furthermore, for the faces which are considered too distant semantics are generated using the methodology described in Section 3.3.2. As can be seen in Figure 4.14, small triangles may have been snapped to the wrong semantics. Therefore, for small triangles the semantics are reassigned based on the largest total coplanar area of connected semantic faces.

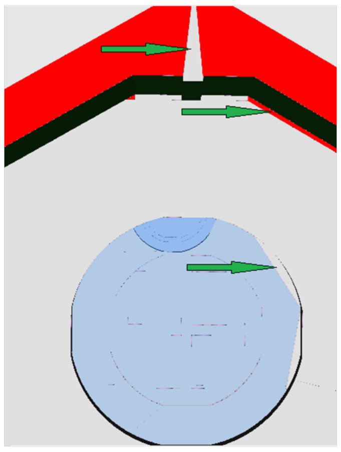


FIGURE 4.14: Mismatched semantics may occur due to the semantics snapping process

4.2.4 Processing Degenerate and Nearly Degenerate Faces

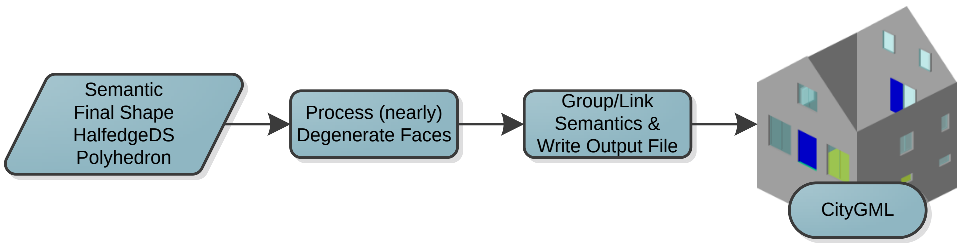
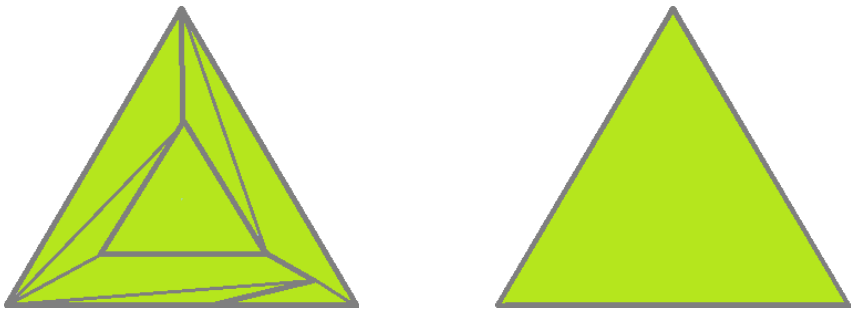


FIGURE 4.15: Flow diagram for the final process of the conversion

Given that the semantics values are now final and set for each triangle, the (near) degeneracies on the surface can be removed or limited respectively. Degeneracies are defined in Section 2.3.2. Afterwards the faces can be grouped and written to the output file (see Figure 4.15). Most of the degeneracies are created during the morphological operations on Nef polyhedra and the subsequent triangulation during the conversion to a HalfedgeDS polyhedron. They are thereby specific to the implementation. For repairing perfect degeneracies the methodology is followed as explained in Section 3.3.1. To prevent near degeneracies in the surface, the surface is detriangulated, by merging coplanar faces with the same semantics, and simplified, removing as many unnecessary vertices and edges as possible in the process. Figure 4.16 provides an example of a surface which is detriangulated.



(A) A coplanar triangulated surface

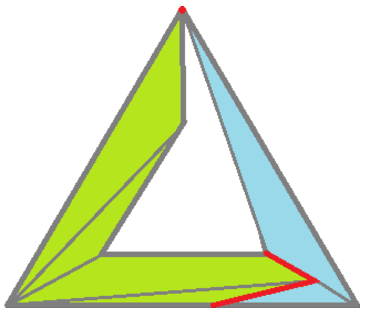
(B) The same surface, but detriangulated

FIGURE 4.16: A coplanar surface before and after detriangulation

Since the HalfedgeDS does not support faces with holes an algorithm was developed to union adjacent coplanar faces without creating holes or faces with non-1-manifold/self-intersecting rings. Non-1-manifold rings would break the validity requirements discussed in Section 2.3.2. The coplanarity of the faces can either be calculated using exact arithmetic or using a threshold. The downside of using exact coplanarity is that far fewer faces are unioned with each other, increasing the file size by roughly 50 % and often leaving the nearly degenerate triangles in the output. It is also believed not to be common practise to use exact arithmetic for this purpose. Instead, the coplanarity of the triangles is determined based on the angle between the normals using a threshold of 1×10^{-4} rad.

The triangles are sorted into groups where all triangles are coplanar to the base triangle of the group. The algorithm iterates over all the edges of the polyhedron. For each edge it is checked whether the incident faces are coplanar and have the same semantics. If they do, the faces can be unioned unless the border that divides them is not one connected set of edges, as that would break the validity rules as discussed in Section 2.3.2. The border between two faces is defined as all vertices and edges that are incident to both faces. A case for which the faces cannot be unioned is shown in Figure 4.17a. Note that the colours of the faces do not represent different semantics, but are only there to distinguish the two faces. The whole surfaces of the exterior shell polyhedron is a connected 2-manifold surface. The white space between the green and blue faces therefore represent either non-coplanar faces or faces with different semantics.

In case the border is not one connected set of edges, a subset of edges that is connected can potentially still be simplified (see Figure 4.17b). The implemented simplification is similar to that published by Meijers [73]. A connected subset of edges is simplified as follows. Between the end points of the first and second edge a new edge is constructed. If this edge does not intersect with the boundary of either of the incident faces, then the first and second edge are removed. If the new edge does intersect with the boundaries then the new edge is removed. The process continues by applying the same algorithm to


(A) The border between the green and blue face is represented in red; the faces cannot be unioned since the border is not one connected set of edges

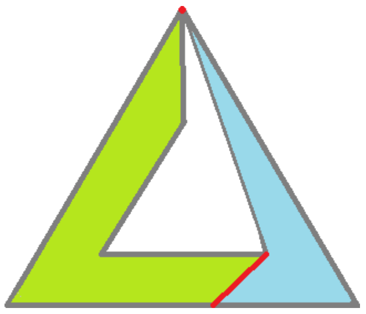
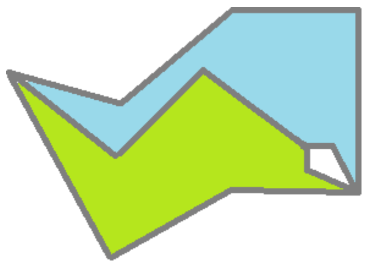
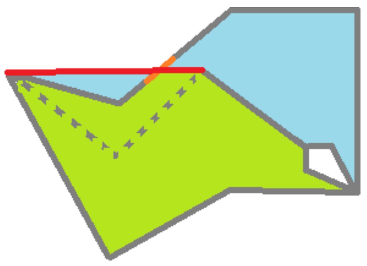

(B) The surface can be detriangulated and the border can be simplified, but the border cannot be removed

FIGURE 4.17: A coplanar surface before and after detriangulation (without creating holes) and simplification of the border

the third edge and depending on the outcome either the second or the new edge. This is repeated until all edges from the subset have been processed. Figure 4.18 shows the simplification of a border. After merging the coplanar faces, collinear edges may occur for which the shared vertex is connected to only the two collinear edges. These vertices can be removed from the geometry by merging the two edges.


(A) Two faces that cannot be unioned


(B) The first vertex cannot be removed as otherwise the geometry becomes invalid

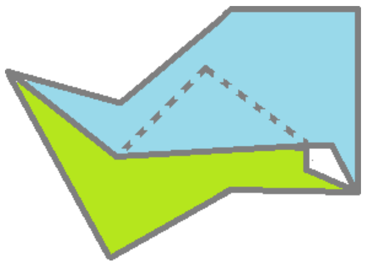

(C) The second vertex can be removed

FIGURE 4.18: The border between coplanar faces that cannot be unioned can be simplified, however the simplification should not result in overlapping geometries

4.2.5 Writing CityGML

The faces are grouped and openings are linked appropriately as defined in Section 3.3.2. When writing the CityGML file each group of faces is written as one 'BoundarySurface' or 'Opening' with the geometry stored in a 'gml::MultiSurface'. For every 'Building' and every 'BuildingPart', also one 'gml::Solid' is written using 'XLinks' to the geometry in the 'MultiSurfaces' as that is conform the CityGML standard [11]. Since 'BuildingInstallations' are not given 'BoundarySurfaces', the geometry is directly stored in the 'gml::Solid'. After writing the CityGML file it can be validated and its quality evaluated.

4.3 Simple Creation of LoD4 Rooms

Although the main focus of this thesis is on LoD3 models, a small side step is made to LoD4 as well. During the geometric transformation inner shells are removed from the geometry at several steps. Each inner shells is potentially the geometry of a room. However, it is found difficult to unambiguously determine whether an inner shell is a room, or just a (small) enclosed cavity. Instead an almost trivial conversion is made from the geometry of 'IfcSpaces'.

'IfcSpaces' are added to IFC to create space boundaries for the use in energy analyses [74, 75]. The 'IfcSpaces' for a storey of a building are shown in Figure 4.19. An 'IfcSpace' can be bounded by building elements and virtual surfaces. The faces of an 'IfcSpace' can be semantically linked to building elements. In Figure 4.20 an example of an 'IfcSpace' which is linked to building elements is shown.

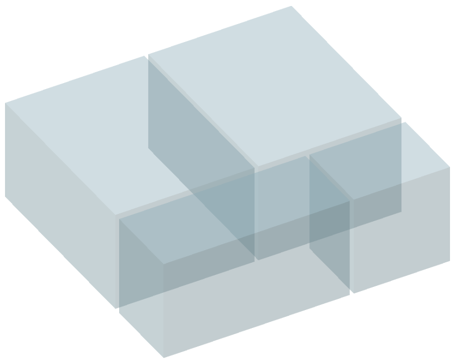


FIGURE 4.19: The IfcSpaces for one storey of a building

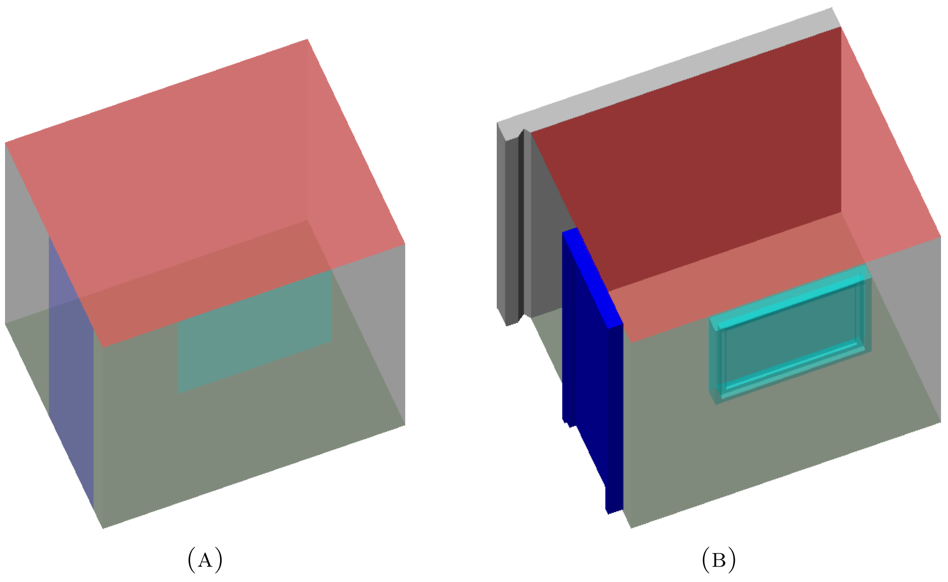
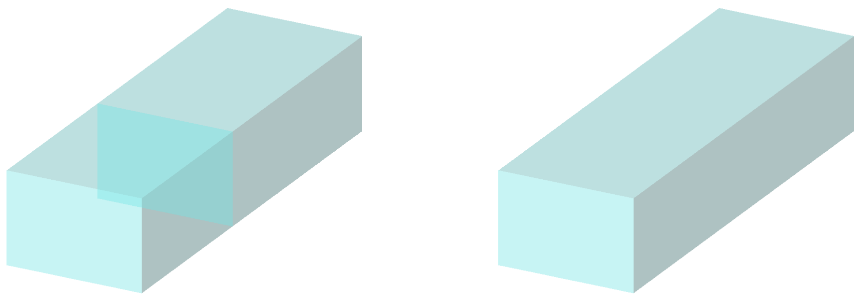


FIGURE 4.20: An IfcSpace can be linked to space boundaries which are IfcBuildingElements. Figure B shows several connected building elements, the other connected walls, roof and floor slabs are not shown

The geometry is generated by IfcOpenShell from the ‘IfcSpaces’ like any other object. Unlike building elements, spaces have always solid geometry as a space is inherently 3-dimensional. Some of the spaces may share a virtual space boundary. In this implementation the related spaces are union creating a larger space (see Figure 4.21). The virtual space boundary can instead be modelled by an ‘ClosureSurface’ which is referenced by both rooms. The geometric refinements are also applied to the rooms to ensure their validity.


(A) Two IfcSpace sharing a virtual space boundary

(B) The two IfcSpaces are union creating one large space

FIGURE 4.21: Connected spaces are union into larges spaces

As the implementation of the conversion to LoD4 is a mere initial exploration of the problems and possibilities, the semantics in these experiments are limited to ‘FloorSurface’, InteriorWallSurface and ‘CeilingSurface’ based on the normal directions, up, horizontal and down respectively. When writing the rooms to a file care is taken that the normals of the ‘BoundarySurfaces’ are opposite to those of the solid geometry. This is done by using ‘XLinks’ in combination with an ‘OrientableSurface’, the latter can be used to indicate that the surface normal is reversed.

The semantic mapping for LoD3 can trivially be converted to a semantic mapping for LoD4 by taking into account whether the geometry after the geometric transformation is interior or exterior. However, LoD4 requires more IFC classes to be mapped to CityGML semantics depending on the desired amount of detail of the conversion. For example, besides ‘IfcBuildingElements’ also ‘IfcFurnishingElements’ should be mapped.

4.4 Input Data Requirements

There are few conditions which prevent the conversion to CityGML from being successful. In general, for an IFC file to be suitable for conversion, it need to comply to the IFC2x3 specifications, the geometry needs to be clean and the semantics need to be applied appropriately and sufficiently. In this section a number of common modelling errors are provided which make the conversion impossible or at least less optimal. For example, CityGML semantics for a roof cannot be generated when it is not stored as such in IFC, thus the ‘IfcSlab’ does not have the ROOF ‘PredefinedType’ nor a ROOFING ‘IfcCovering’, nor does it decompose a roof object. The roof in the CityGML model is then given a ‘FloorSurface’ instead of a ‘RoofSurface’ during the conversion.

Another requirement is that the geometry and semantics need to be stored in the IFC file such that they can be interpreted unambiguously by an IFC-compliant reader. In order for a model to comply with the CityGML specification, the geometric accuracy should be better than 0.5 m for LoD3 and 0.2 m for LoD4 [11]. Since the exterior of LoD3 building can be reused in LoD4 it is recommended that the geometric accuracy of the IFC input model is better than 0.2 m. The geometry should represent the building, but does not need to be valid according to ISO19107. In theory, the conversion should work for as long as the the geometry can be extracted from the IFC file. Solid geometry is preferred, since non solid geometry will be dilated by the implementation such that it is a solid. For example, B-rep geometry should not have missing faces. Figure 4.22 shows a door where some of the faces were not properly stored or missing from the IFC file. In such an event the prototype tries to recover from the bad geometry by dilating it. Although it is not preferred, objects may contain multiple solids, non-manifolds or have an inconsistent face orientation. These cases can be recovered from during the geometric transformation.

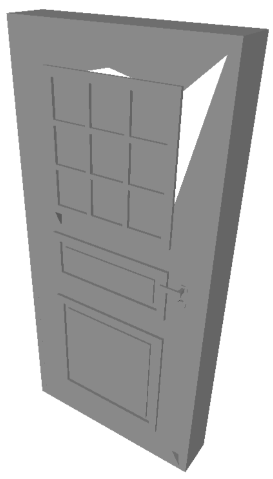
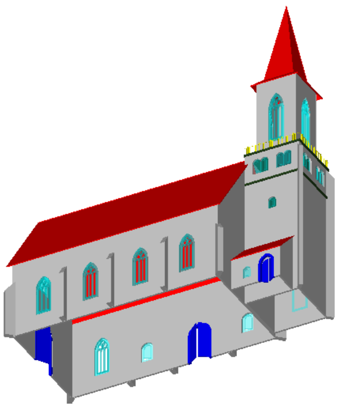


FIGURE 4.22: A solid should be completely enclosed by faces, there are several faces missing from this door

The explicit geometry must at least represent the complete exterior of the building, including the base slab. If there are major sections of the building missing, like in Figure 4.23, even the closing operations cannot recover without using an oversized structuring element causing a large loss of detail. Also the explicit geometry should represent doors and windows in their closed state. This also holds for revolving and sliding doors. Figure 4.24 depicts a revolving door wrongly modelled in its open state.


(A) A church missing a base slab

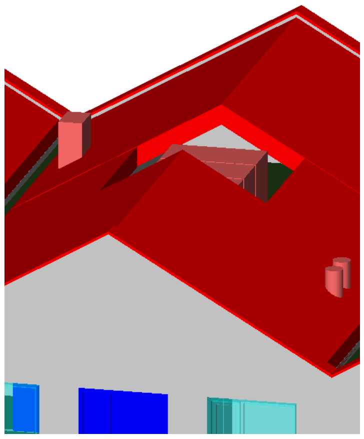

(B) Building where part of the roof is missing

Figure 4.23: Buildings where major sections are missing

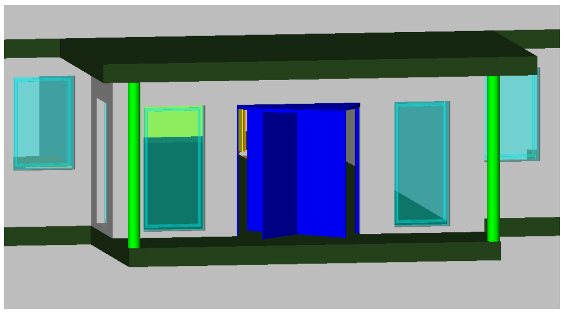


Figure 4.24: All doors and windows should be closed in their explicit geometry, even revolving and sliding doors. The revolving door shown here is not closed leaving a gap

The geometry in IFC should represent physical objects such that the objects can be produced individually and combined during the construction of the building. The IFC solids may therefore not overlap unless a Boolean difference operation is specified between them. If there are overlapping solids the semantic mapping may become ambiguous as there can be two different mappings for the same surface (see Figure 4.25).

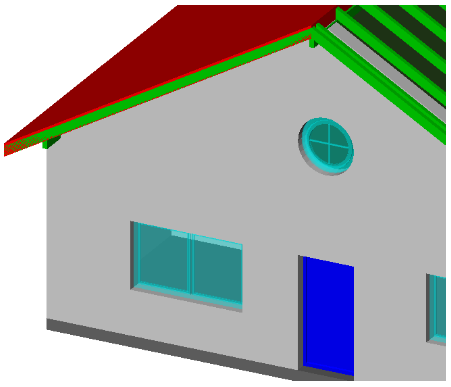


Figure 4.25: Solid geometry should not overlap as it is not physically possible. Here the roof slab and the beams intersect

For the semantics of the input can be said that more is generally better. For semantics there is only one requirement on the input IFC file which is that the objects that make up the exterior of the building should pass the filtering process from Section 3.1.1, thus they should be subtypes of 'IfcBuildingElement' and they should be (indirectly) contained in a 'IfcBuilding' spatial structure. To get optimal results the objects should be given semantics such that the mapping, given in Section 3.1.2, is optimal. This includes the issue related to chimneys, beams, columns and 'IfcSpaces' as discussed in the same section. If not, the result will still be valid, but semantics will be lacking. By making the IFC and modelling specification better defined this can be prevented.

There are two common problems in the way 'IfcSlabs' are modelled. First, when the lowest slab forms the foundation of the building it is sometimes given the 'PredefinedType' FLOOR instead of BASESLAB. The slab is actually both, but only one 'PredefinedType' can be assigned to one object. The solution would be to assign the BASESLAB type to the slab and attach an 'IfcCovering' FLOORING type to the upper side of the of the slab. By doing so both semantics can be stored.

The second problem occurs when a part of the slab should have different semantics, but is not subdivided since it is one physical slab. This occurs for example when the first storey above ground level has a floor slab which protrudes out of the building to form a roof above the entrance (see the top slab in Figure 4.24). It is unlikely that the protrusion is meant as a floor and should either have a ROOFING covering or be separated from the rest of the slab. The latter is preferred as the adjacency between the slabs can be stored by having the slabs aggregate an 'IfcElementAssembly' with the 'PredefinedType' SLAB_FIELD.

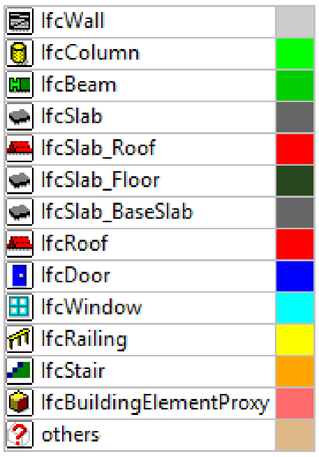
4.5 Experimental Results

The implemented prototype is able to successfully convert IFC models to valid CityGML LoD3 models and can also generate simple room shapes for LoD4. In Section 4.5.1 the input and the LoD3 results are shown and discussed for ten models. Section 4.5.2 covers the results for LoD4. For these models the validity and quality is assessed and the performance of the conversion processes as a whole is evaluated in Section 4.6.

4.5.1 Input and LoD3 Model Results

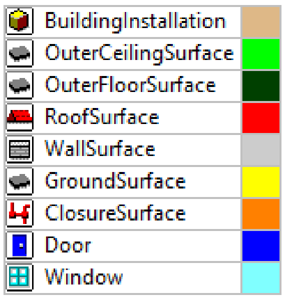
The images of both the IFC and the CityGML models are made using FZK Viewer v4.0 by KIT. Some of the images have been manually altered to remove rendering issues as not to confuse the reader. For each of the models an image is shown for both the input IFC model and the generated CityGML model, also for each a rendering without the roof is presented to show the interiors (or lack thereof) of the building models. Figure 4.26 provides legends for the meaning of the colours in both the IFC and CityGML models.

Furthermore, for each of the models the statistics and a description of the characteristics are given. The protoype is tested on an Intel Sandy Bridge CPU at 4.5 GHz with sufficient ram. The geometry and semantics of the models are evaluated with automated and manual methods which are discussed in Section 4.6.



IfcWall	⬜
IfcColumn	🟩
IfcBeam	🟩
IfcSlab	⬛
IfcSlab_Roof	🟥
IfcSlab_Floor	🟩
IfcSlab_BaseSlab	⬛
IfcRoof	🟥
IfcDoor	🟦
IfcWindow	🟦
IfcRailing	🟨
IfcStair	🟧
IfcBuildingElementProxy	🟥
others	🟫

(A) Legend for the most common objects in IFC



BuildingInstallation	🟫
OuterCeilingSurface	🟩
OuterFloorSurface	🟩
RoofSurface	🟥
WallSurface	⬜
GroundSurface	🟨
ClosureSurface	🟧
Door	🟦
Window	🟦

(B) Legend for the CityGML models

FIGURE 4.26: Legends for both the IFC and the CityGML models

IfcOpenHouse

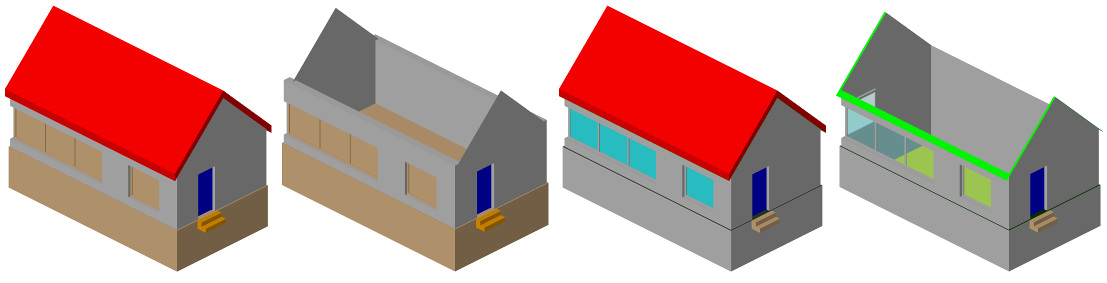

(A) Input IFC (B) IFC without roof (C) Output CityGML (D) CityGML without roof

FIGURE 4.27: IfcOpenHouse input and conversion result

IfcOpenHouse is a simple building generated with IfcOpenShell. The window opposite to the door is not rendered by FZK Viewer due to a minor compliance issue in the model concerning ‘OpeningElements’. This however did not affect the conversion. The models does not require a closing operation for the extraction of the proper exterior shell. The only improvement that can be added to the CityGML model is an ‘XLink’ relation for the faces where the stairs touch the building. Also for the windows can be argued whether or not the frame should be part of the window. The conversion takes only six seconds due to the simplicity of the model.

TABLE 4.1: Information and statistics for IfcOpenHouse (Source: blog.ifcopenshell.org/2012/11/say-hi-to-ifcopenhouse.html)

Name	IfcOpenHouse
File Name	IfcOpenHouse.ifc
Number of IFC Entities	46
Input File Size (MB)	0.102
Output File Size (MB)	0.045
Structuring Element Width (mm)	0
BuildingParts/Installations	0/1
Conversion Time (s)	6

FZK-House

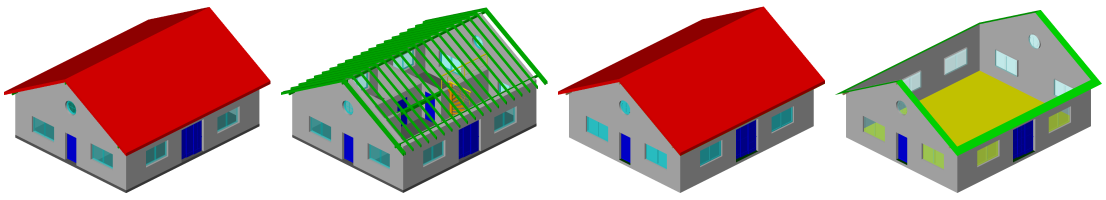

(A) Input IFC (B) IFC without roof (C) Output CityGML (D) CityGML without roof

FIGURE 4.28: FZK-House input and conversion result

FZK-House is a fictional building with furniture and rooms. It is the first of many models exported by ArchiCAD. The IFC model has a roof support structure which overlaps with the roof geometry itself. The support structure should have been subtracted from the roof geometry for it to be a realistic model. The support structure does not cause any problems as it is composed out of 'IfcBeams' which are mapped as 'BuildingInstallation'. As such, the subtraction is performed during the conversion. The CityGML model does not have any geometric or semantic flaws.

TABLE 4.2: Information and statistics for FZK-House (Source: www.iai.fzk.de/www-extern/index.php?id=1174&L=0)

Name	FZK-House
File Name	AC13-FZK-Haus-CV.ifc
Number of IFC Entities	111
Input File Size (MB)	1.49
Output File Size (MB)	0.681
Structuring Element Width (mm)	0
BuildingParts/Installations	0/6
Conversion Time (s)	272

FJK-House

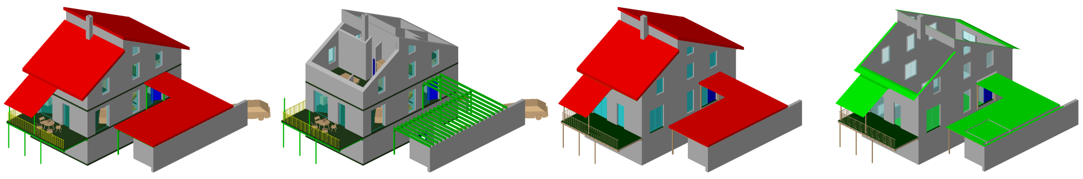

(A) Input IFC (B) IFC without roof (C) Output CityGML (D) CityGML without roof

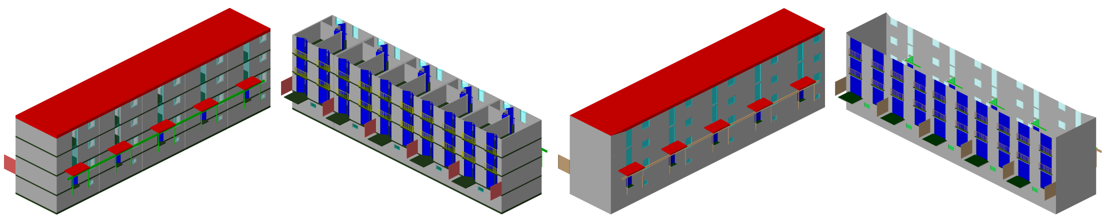
FIGURE 4.29: FJK-House input and conversion result

FJK-House is a building model with more external features compared to the previous models. It is the first model to require morphological closing for the proper exterior shell to be extracted. Without closing several of the rooms of the building are not removed from the geometry. The closing operation does however create several artefacts. The carport, balcony and chimney are part of the exterior shell in the CityGML model. With more semantics in IFC in the future, these might be converted to 'BuildingInstallations' instead. Even with all the furniture and vehicles filtered out, the 'BuildingInstallations' (beam, columns and stairs) take up a large amount of file space. This is amongst others caused by the large amount of beams that make up the support structure for the carport. The CityGML model does not have a 'GroundSurface' which is a flaw caused by the semantics of the related slab in the IFC model. The slab is a FLOOR while it for a proper mapping it should have been a BASESLAB.

TABLE 4.3: Information and statistics for FJK-House (Source: www.iai.fzk.de/www-extern/index.php?id=1167&L=0)

Name	FJK-House
File Name	FJK-Project-Final.ifc
Number of IFC Entities	274
Input File Size (MB)	13.9
Output File Size (MB)	1.12
Structuring Element Width (mm)	300
BuildingParts/Installations	12
Conversion Time (s)	217

Smiley West



(A) Input IFC (B) IFC without roof (C) Output CityGML (D) CityGML without roof

FIGURE 4.30: Smiley West input and conversion result

The Smiley West model consists of five terraced houses. Each house can be processed individually, however the scope of the thesis requires that one exterior shell is generated for the combination of buildings. Since the buildings are disjoint, without morphological closing four of the five buildings are processed as 'BuildingParts' of the largest model. However, by applying closing the exterior shell for the combination of models is extracted. The closing operation does not create any artefacts as the building is completely rectangular. The semantics of some areas of the surface are not mapped correctly. This is caused by missing border edges in the geometry. Also like the FJK-House and most others, the 'GroundSurface' is not generated.

Table 4.4: Information and statistics for Smiley West (Source: www.iai.fzk.de/www-extern/index.php?id=1168&L=0)

Name	Smiley West
File Name	AC-11-Smiley-West-04-07-2007.ifc
Number of IFC Entities	627
Input File Size (MB)	7.98
Output File Size (MB)	4.47
Structuring Element Width (mm)	300
BuildingParts/Installations	0/37
Conversion Time (s)	888

Niedriha

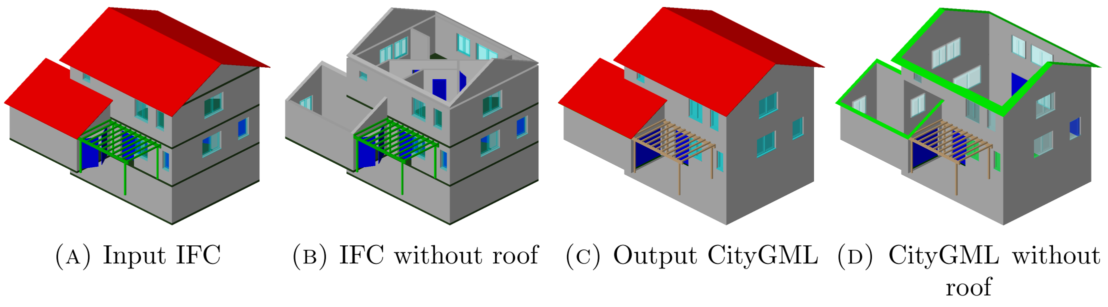

(a) Input IFC (b) IFC without roof (c) Output CityGML (d) CityGML without roof

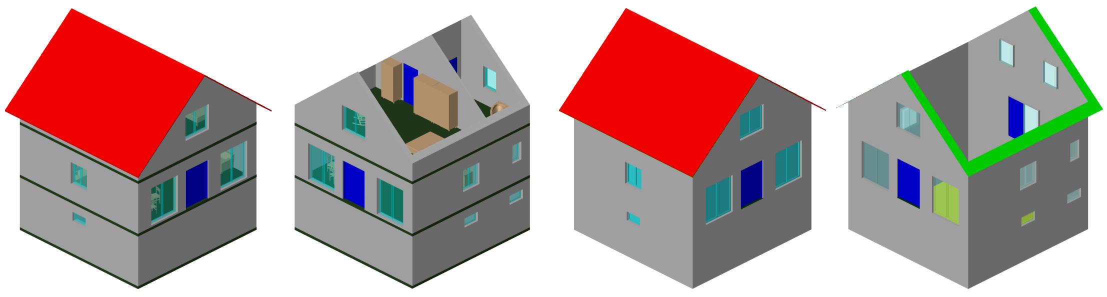
Figure 4.31: Niedriha input and conversion result

The Niedriha model features a building with a garage and a carport. The model is converted without closing, the result of which does lack a 'GroundSurface'. The garage is a potential candidate for a 'BuildingPart' in CityGML. To do so the garage would need to be modelled as a 'IfcBuilding' with PARTIAL as the 'CompositionType'. During the conversion, the exterior shell of the main building can then be subtracted from the exterior shell of the garage, while the surfaces where the new shells touch should be 'ClosureSurfaces'. However, since the model does not have such semantics, no 'BuildingParts' are generated.

Table 4.5: Information and statistics for Niedriha (Source: code.google.com/p/bimserver/source/browse/trunk#trunk%2FTestData%2Fdata)

Name	Niedriha
File Name	AC17-niedriha-V4.ifc
Number of IFC Entities	301
Input File Size (MB)	1.99
Output File Size (MB)	0.526
Structuring Element Width (mm)	0
BuildingParts/Installations	0/1
Conversion Time (s)	92

BIEN-ZENKER Jasmin-Sun



(A) Input IFC (B) IFC without roof (C) Output CityGML (D) CityGML without roof

Figure 4.32: BIEN-ZENKER Jasmin-Sun input and conversion result

BIEN-ZENKER Jasmin-Sun is the model used in the flow diagram throughout this thesis. The model does not require closing and there are no flaws despite overlapping geometries in the input model.

Table 4.6: Information and statistics for BIEN-ZENKER Jasmin-Sun (Source: www.iai.fzk.de/www-extern/index.php?id=2208&L=0)

Name	BIEN-ZENKER Jasmin-Sun
File Name	Bien-Zenker_Jasmin-Sun-AC14-V2.ifc
Number of IFC Entities	122
Input File Size (MB)	4.58
Output File Size (MB)	0.203
Structuring Element Width (mm)	0
BuildingParts/Installations	0/0
Conversion Time (s)	72

Office Building

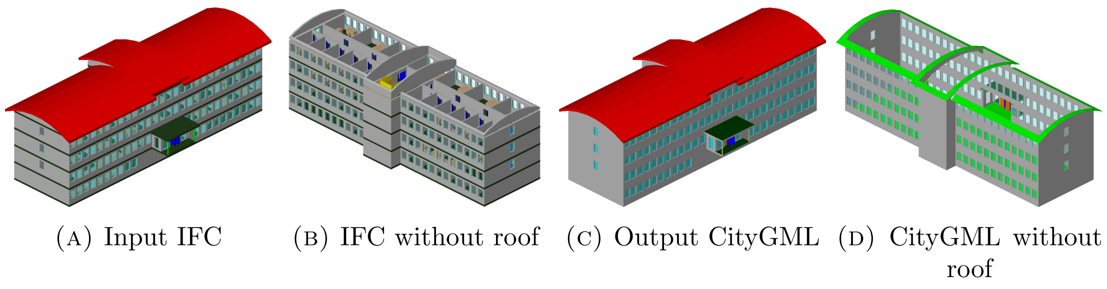

(A) Input IFC (B) IFC without roof (C) Output CityGML (D) CityGML without roof

FIGURE 4.33: Office Building input and conversion result

Office Building is the model where the entrance's revolving door is not closed in the IFC file (see Figure 4.24). Nevertheless, no closing is required for the generation of a proper exterior shell. The gap left open by the door is closed by an 'IfcSpace' modelling the entrance hall. Although this is not the preferred way to model the entrance, it is allowed. The lacking semantics of the slab above the entrance still result in it being mapped to a 'FloorSurface' while it should be a 'RoofSurface'.

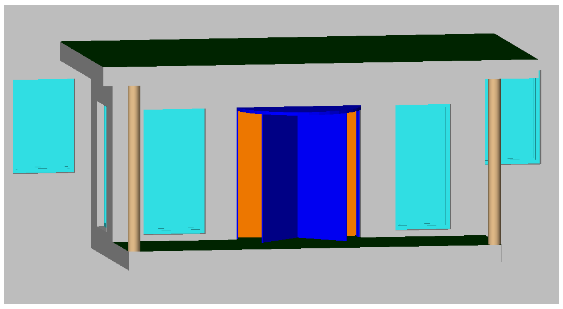


FIGURE 4.34: The gap left open by the door (Figure 4.24) is closed by a ClosureSurface due to an IfcSpace modelling the entrance hall

TABLE 4.7: Information and statistics for Office Building (Source: www.iai.fzk.de/www-extern/index.php?id=1184&L=0)

Name	Office Building
File Name	AC11-Institute-Var-2-IFC.ifc
Number of IFC Entities	1201
Input File Size (MB)	2.7
Output File Size (MB)	2.25
Structuring Element Width (mm)	0
BuildingParts/Installations	0/2
Conversion Time (s)	612

Haus-G-H

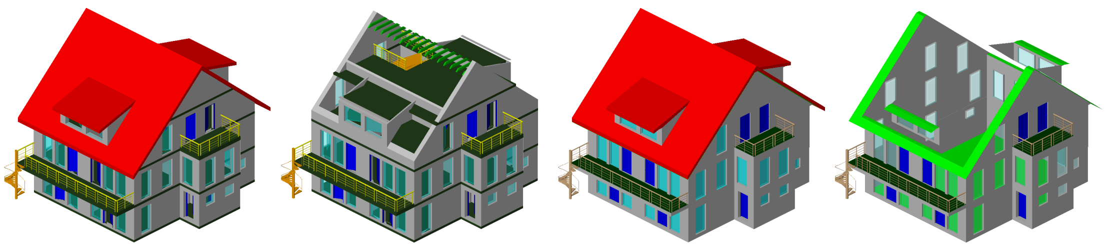

(A) Input IFC (B) IFC without roof (C) Output CityGML (D) CityGML without roof

FIGURE 4.35: Haus-G-H input and conversion result

Haus-G-H is another example of a model where closing is required as otherwise not all rooms are removed. Due to the closing operation artefacts appear underneath the roof overhang. Aside from the missing 'GroundSurface' the semantics could be better if also the balcony and dormer are extracted as 'BuildingInstallations'. There are also several faces with mismatched semantics.

TABLE 4.8: Information and statistics for Haus-G-H (Source: code.google.com/p/bimserver/source/browse/trunk#trunk%2FTestData%2Fdata)

Name	Haus-G-H
File Name	AC9R1-Haus-G-H-Ver2-2x3.ifc
Number of IFC Entities	301
Input File Size (MB)	4.34
Output File Size (MB)	5.28
Structuring Element Width (mm)	300
BuildingParts/Installations	0/4
Conversion Time (s)	1672

Model 4351

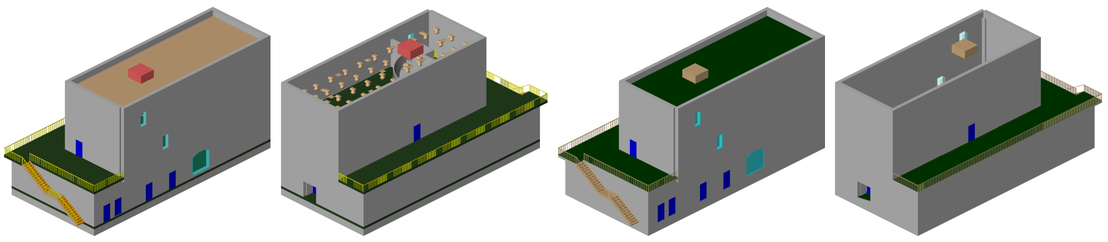

(A) Input IFC (B) IFC without roof (C) Output CityGML (D) CityGML without roof

FIGURE 4.36: Model 4351 input and conversion result

Model 4351 is unlike most others exported by Autodesk Revit. No closing is required for the conversion. The model does not have any objects relating to a roof. Since the highest slab is surrounded by a balustrade it is assumed that it is meant as a walkable surface. The CityGML standard does not specify how this should be modelled. Therefore, the 'FloorSurface' applied by the conversion is believed to be an appropriate semantic. In the resulting CityGML model the 'BuildingInstallations' again take up a large amount of space. The result also has one mismatched face.

TABLE 4.9: Information and statistics for Model 4351 (Source: code.google.com/p/bimserver/source/browse/trunk#trunk%2FTestData%2Fdata)

Name	Model 4351
File Name	4351.ifc
Number of IFC Entities	442
Input File Size (MB)	6.95
Output File Size (MB)	2.81
Structuring Element Width (mm)	0
BuildingParts/Installations	0/7
Conversion Time (s)	1041

1407 Opheusden WoZoCo

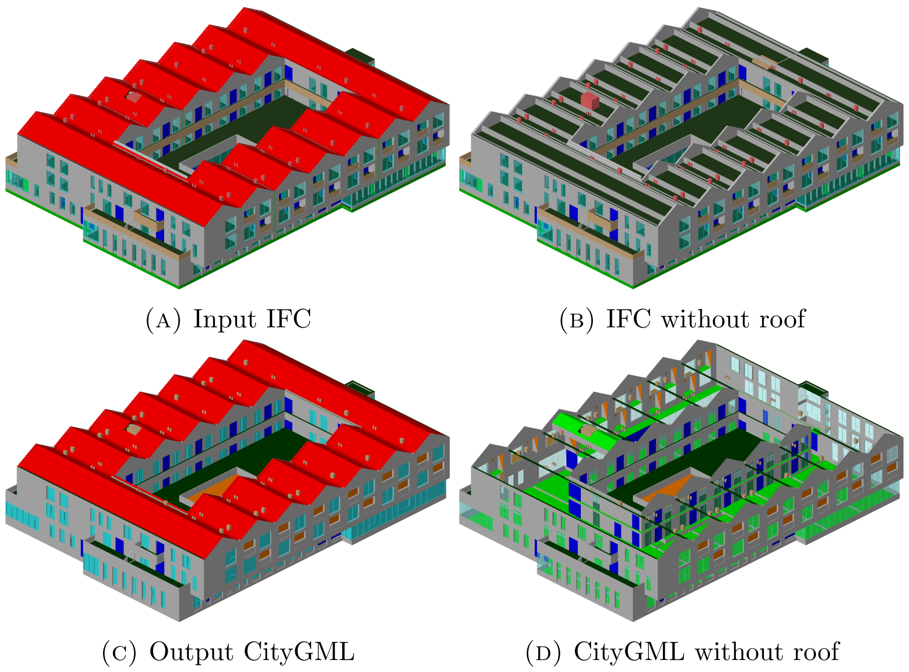

(A) Input IFC (B) IFC without roof
(C) Output CityGML (D) CityGML without roof

FIGURE 4.37: 1407 Opheusden WoZoCo input and conversion result

The 1407 Opheusden WoZoCo model is by far the largest model in size spatially. It contains 31 apartments and is exported from ArchiCAD. The missing roof section cannot be recovered by closing without also removing most doors and windows in the process. The roof section may be intentionally missing in which case it is converted appropriately. The model required closing for the extraction of a proper exterior shell.

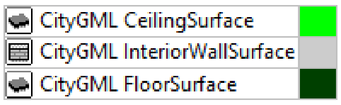
The model contains many ‘IfcSpaces’ which are outside the actual exterior shell. This is not a modelling error in the IFC file, but does degrade the quality of the conversion. The result has mismatched faces and no ‘GroundSurface’. Furthermore, 94% of the CityGML file is allotted to ‘BuildingInstallations’.

Table 4.10: Information and statistics for 1407 Opheusden WoZoCo (Source: code.google.com/p/bimserver/source/browse/trunk#trunk%2FTestData%2Fdata)

Name	1407 Opheusden WoZoCo
File Name	1407_BE_WOZOCO.ifc
Number of IFC Entities	3521
Input File Size (MB)	7.81
Output File Size (MB)	70.1
Structuring Element Width (mm)	300
BuildingParts/Installations	0/64
Conversion Time (s)	5495

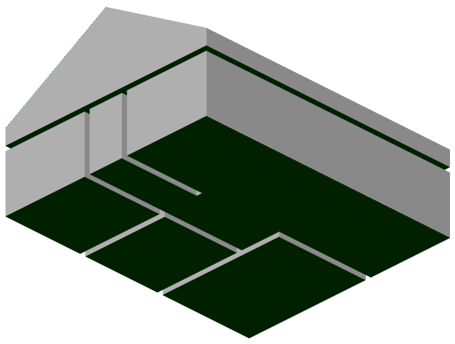
4.5.2 Generated Rooms for LoD4

Not all of the models presented in Section 4.5.1 have ‘IfcSpaces’ such that rooms can be generated for ‘LoD4. The meaning for the colours in the LoD4 room models can be found in Figure 4.38. In Figure 4.39 the rooms themselves are shown from four of the models. The only flaws that can be noted are the lack of ‘Openings’ and also the outside ‘IfcSpaces’ from the 1407 Opheusden WoZoCo model should not have been converted to rooms.

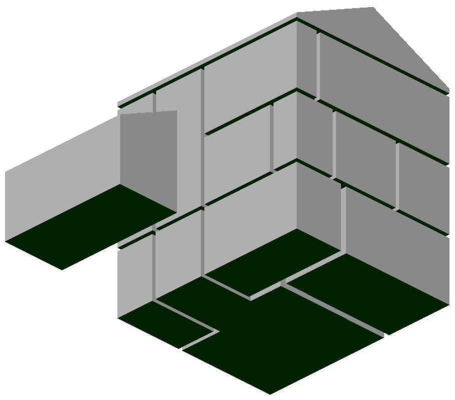


CityGML CeilingSurface	🟩
CityGML InteriorWallSurface	⬜
CityGML FloorSurface	⬛

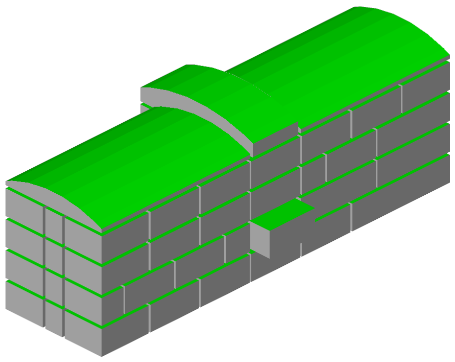
Figure 4.38: Legend for the CityGML LoD4 room models



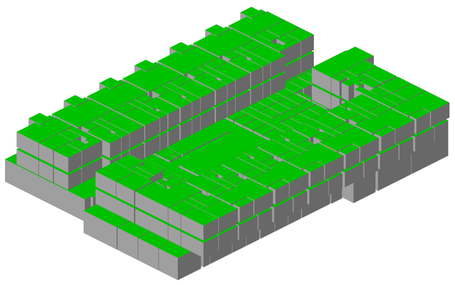
(A) Bottom-up view of the CityGML Rooms in the FZK-House model



(B) Bottom-up view of the CityGML Rooms in the Niedriha model



(C) Top-down view of the CityGML Rooms in the Office Building model



(D) Top-down view of the CityGML Rooms in the 1407 Opheusden WoZoCo model

FIGURE 4.39: Automatically generated LoD4 rooms

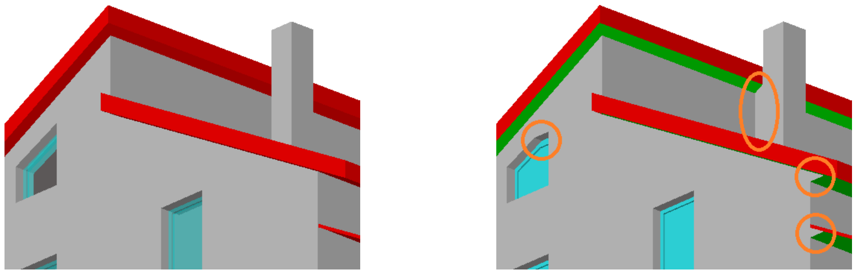
4.6 Validation and Evaluation of the Experimental Results

The models generated by the prototype and presented in Section 4.5 are examined in this section. The validity of the geometry and semantics are evaluated in Section 4.6.1 to provide insight on where the prototype performs well and where there is room for improvement. Aside from just the output, the performance of conversion process as a whole is discussed in Section 4.6.2. The statistics for all models can be found in Table 4.11. The evaluation of the conversion to LoD4 and the possibilities are discussed separately in Section 4.6.3.

4.6.1 Validation and Quality of the Generated LoD3 models

During the creation of the solid geometry, the ISO19107 validity standard is taken into account. The output models are validated using the 3D validator by Ledoux [19]. All the models are found valid according to the ISO19107 definitions. As expected the validator does indicate that there are nearly degenerate faces remaining, moreover in Section 4.6.2.

The exterior shell was successfully extracted for each of the building models. For four of the models closing is required. The closing is needed as otherwise rooms and other parts of the interior are not removed due to a gap in the exterior. Examples of artefacts can be seen in Figure 4.40.


(A) Input IFC model

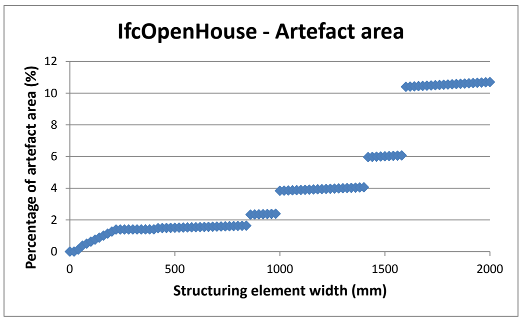
(B) Output CityGML model, converted with a 300 mm structuring element

FIGURE 4.40: Artefacts occur at concave parts of the geometry that are not aligned with the structuring element

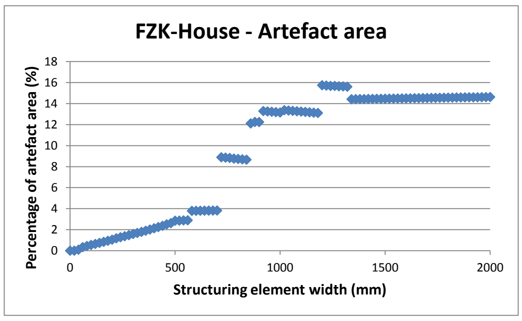
To quantify the influence of the artefacts on the geometry, for each of the models the percentage of surface area added by the artefacts is approximated. This is done by calculating the area for all triangles that are found too distant during the semantic snapping process explained in Section 4.2.3. The average artefact area is 0.8 % of the total area for all models which required morphological closing. As expected, when no closing is applied the average artefact area is 0 %.

To visualise the influence of the size of the structuring element the graphs in Figure 4.41 are made. The graphs show how the percentage of artefact area increases with an increasing size of the structuring element. Two models are used for which the proper exterior is already extracted without closing. At first the percentage grows gradually as only artefacts occur at places like underneath roof overhang. The discontinuous jumps in the graphs occur when doors and windows collapse. Collapsing doors and windows must not occur during the conversion as doors and windows are valuable semantics to an LoD3 CityGML model. As such, a maximum structuring element width of 300 mm is recommended. In Figure 4.42 the output of the FZK-House is shown for multiple sizes of the structuring element.

For future work, a methodology can be developed for removing the artefacts. In Figure 4.43 a Boolean difference operation is shown between the exterior geometry and all the input solids. This approach potentially makes it possible for a user to select artefacts from the remaining solids such that they can be removed from the exterior geometry. Heuristic carving, as explained in Section 3.2.3, can possibly be adapted to automate this process.

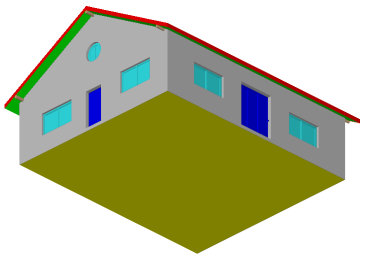


(A)

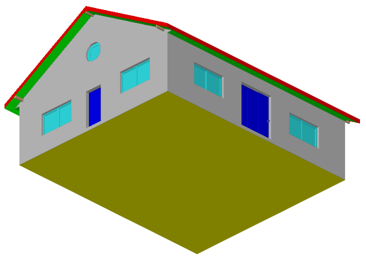


(B)

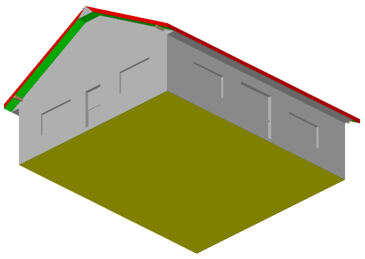
FIGURE 4.41: Percentage of artefact area as a function of the width of the structuring element



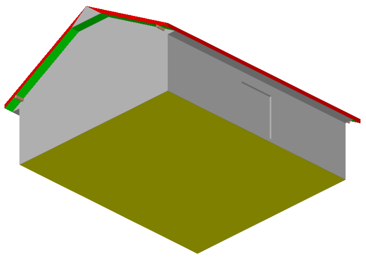
(A) Element width 0 mm



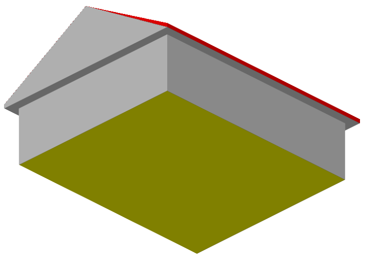
(B) Element width 300 mm



(C) Element width 1000 mm



(D) Element width 2000 mm



(E) Element width INFINITE

FIGURE 4.42: The influence of the width of the structuring element on the output of FZK-House. Too large elements make doors and windows collapse(C & D). An element with an near infinite size can be used to generalize the building (E)

For the semantics it can be argued that they are valid since the normal constraints are explicitly applied. Also, visual inspection revealed no errors concerning the normal constraints. This does not mean that faces cannot have been given the wrong semantics. During the creation of a Nef polyhedron the edges between coplanar faces are removed. These edges are not always recovered during the triangulation. As such, parts of facets are given the wrong semantics, an example is given in Figure 4.44. This issue is again specific to the implementation.

There are also some problems due to limitations in the methodology. In many IFC models the foundation of the house is a floor slab. If the normal of a floor slab points down, instead of 'GroundSurface', by default floor slab is mapped to 'CeilingSurfaces' (see Figure 4.45a). If the slab is stored as a base slab in IFC semantic mapping would

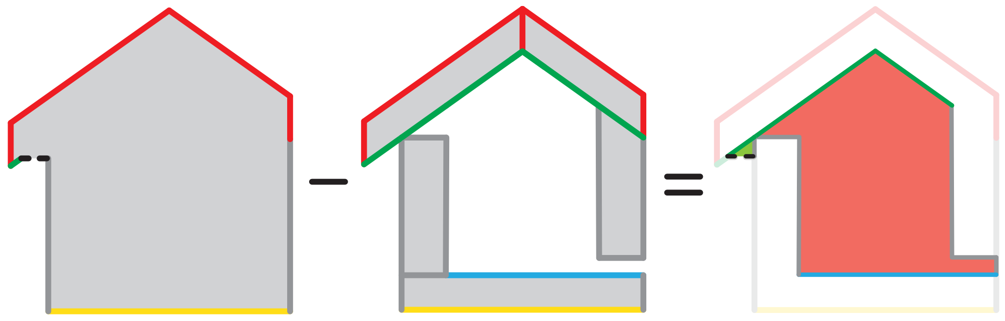
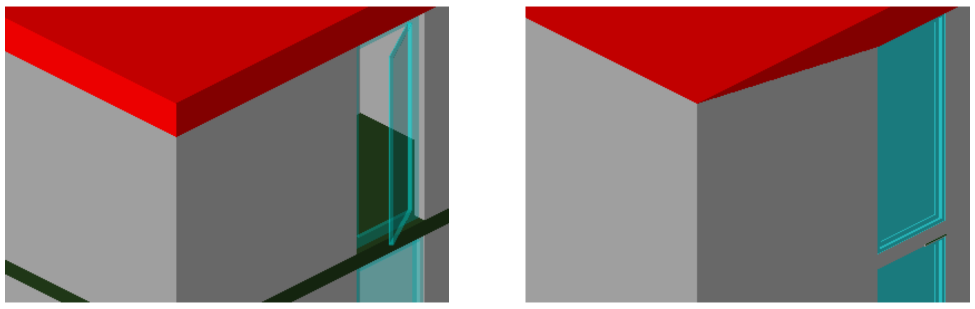


Figure 4.43: The artefacts, the interior geometry and the geometries that fill the gaps can be extracted by subtracting all the input geometries from the exterior solid. The artefact (light green area in right figure) can then be distinguished

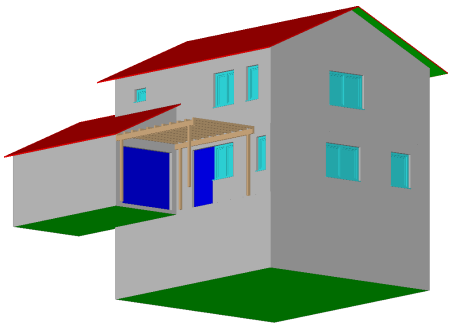


(a) Input IFC model

(b) Output CityGML model, the border edges between the vertical roof and wall faces are removed

Figure 4.44: Due to edge removed by Nef polyhedra, parts of faces may be assigned to the wrong semantics

have resulted in correct semantics. The terrain is required to determine the 'GroundSurface' accurately for every situation, but that is out of the scope of the thesis. Also, extracting CityGML semantics for which information cannot be stored in IFC is not possible. For instance, dormers and balconies (Figure 4.45b) have no special semantics in IFC and can therefore not be extracted as 'BuildingInstalations'. As such, in those cases the semantics are valid, but slightly less than optimal semantics. These semantics can easily be added to IFC by adding the 'PredefinedTypes' balcony and dormer to 'IfcElementAssembly'.


(A) The CeilingSurface (green) below the building and the garage should have been a GroundSurface

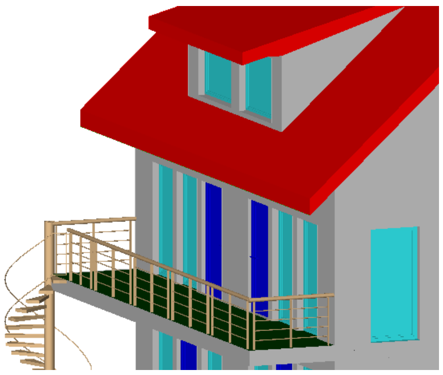
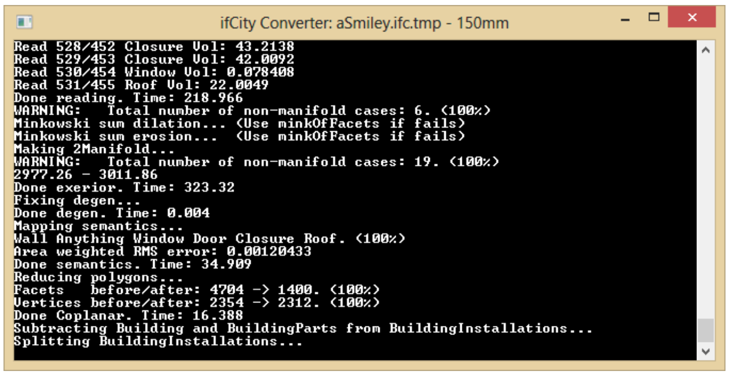

(B) A dormer and a balcony not recognized as BuildingInstallation

FIGURE 4.45: Several semantics are not optimal or wrong in the output due to missing semantics in IFC

4.6.2 Performance of the Conversion to LoD3

The prototype application consists of two executables, one for the preprocessing and the other for the conversion. A batch file enables the two executables to automatically run in sequence. Neither part of the program has a Graphical User Interface (GUI), but use the console to provide feedback on the progress. Figure 4.46 shows a screenshot of the console during the conversion of one of the models. The focus for the remainder of this section is only on the conversion part.



```
Read 528/452 Closure Vol: 43.2138
Read 529/453 Closure Vol: 42.0092
Read 530/454 Window Vol: 0.078408
Read 531/455 Roof Vol: 22.0049
Done reading. Time: 218.966
WARNING:   Total number of non-manifold cases: 6. (100%)
Minkowski sum dilation... (Use minkOfFacets if fails)
Minkowski sum erosion...  (Use minkOfFacets if fails)
Making 2Manifold...
WARNING:   Total number of non-manifold cases: 19. (100%)
2977.26 - 3011.86
Done exerior. Time: 323.32
Fixing degen...
Done degen. Time: 0.004
Mapping semantics...
Wall Anything Window Door Closure Roof. (100%)
Area weighted RMS error: 0.00120433
Done semantics. Time: 34.909
Reducing polygons...
Facets   before/after: 4704 -> 1400. (100%)
Vertices before/after: 2354 -> 2312. (100%)
Done Coplanar. Time: 16.388
Subtracting Building and BuildingParts from BuildingInstallations...
Splitting BuildingInstallations...
```

FIGURE 4.46: Screenshot of the console during the conversion process

In Figure 4.47 a graph is shown to indicate the influence of the input file size on the computation time of the conversion. A graph showing the computation time as a function of the number of objects in the input file gives a similar image. The graph is a good depiction of how unpredictable the required computation time is. Model '1407 Opheusden WoZoCo' is a special case which contains approximately ten times the number of objects compared to the other models. Its conversion took 95 minutes. In general, more complex models take longer to compute, but many factors are hard to predict like the complexity of the exterior shell and the number of 'BuildingInstallations'.

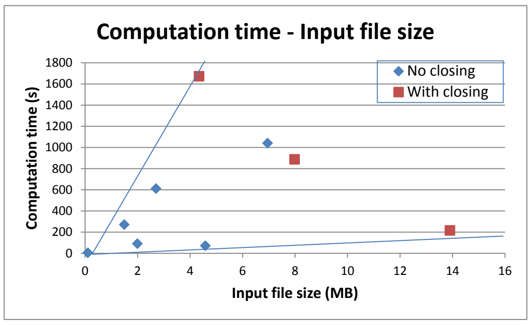


FIGURE 4.47: The computation time as a function of the input file size

To evaluate the influence of the closing operation on the computation time, it is plotted against the size of the structuring element in Figure 4.48. The graphs show how the computation time significantly increases as a result of the morphological operations. Furthermore, the graphs show how the computation time gradually decreases as the geometry becomes more simplified due to the morphological operations.

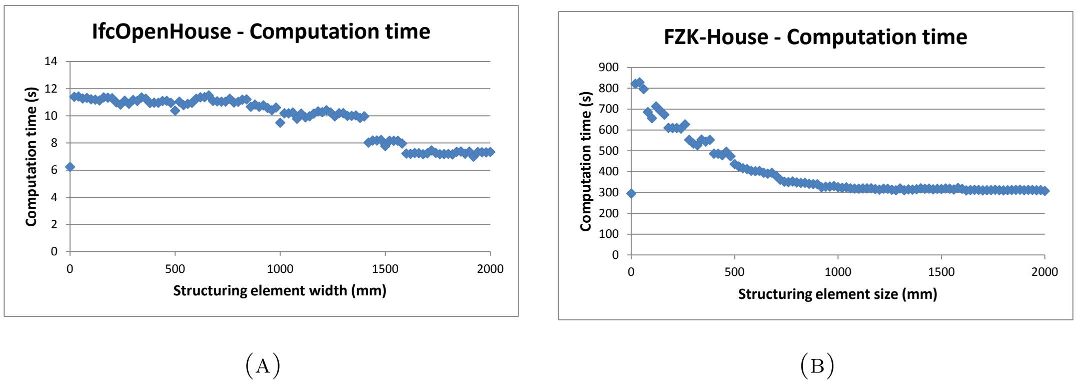


FIGURE 4.48: The computation time as a function of the width of the structuring element

In addition to the computation time, also the output file size is plotted against the input file size in Figure 4.49. Since the conversion can be seen as a generalisation, the output file size is expected to be lower than the that of the input. For the models in the graph the average file reduction is 60 %. The data from model '1407 Opheusden WoZoCo' is not plotted as it is at a completely different scale. Its input file size is 7.81 MB, the output 70.1 MB, that is nine times as much. By inspecting the file it is discovered that most of the file space is occupied by 'BuildingInstallations'. This is due to the high amount of faces required to model circular geometries in fences and complex geometries like stairs. For all the models with 'BuildingInstallations', the average file size dedicated to 'BuildingInstallations' is 50 %, with a minimum of 3 % and a maximum of 95 %. The latter percentage is from the '1407 Opheusden WoZoCo' model. While the 'BuildingInstallations' represent objects of a lower importance than the building geometry, it takes up a significant amount of space. Further generalisation to bring balance to the distribution of file space can therefore be considered valuable.

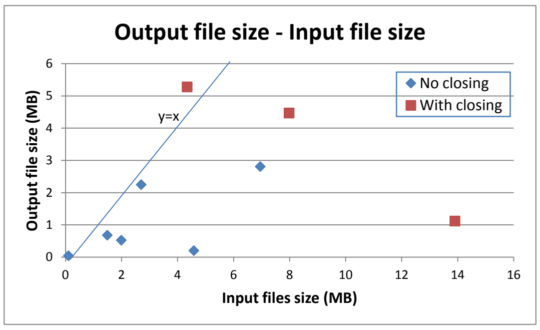


FIGURE 4.49: The output file size as functions of the input file size

The surface is detriangulated mainly for the purpose of limiting the chances of nearly degeneracies on the surface by removing as many unnecessary vertices and edges as possible. On average, 3.1 % of the vertices are removed for models that require closing. Only an average of 0.1 % of the vertices are removed in the other models. The difference is due to the additional vertices created during the morphological operations. For all of the models an average of 46 % of the edges are removed. The reduced geometries may prevent future problems due to degeneracies in other applications. The method could be slightly more effective if the detriangulation can also handle holes. A very welcome and not unexpected side-effect of the detriangulation is that it reduces the file size. The reduction is significant; on average a 66 % reduction is achieved, with a minimum of 62 % and a maximum of 73 %. The reduced files size due to the detriangulation, but also generalized 'BuildingInstallations' becomes beneficial especially when a large amount of building models are combined in one city model. Table 4.11 shows statistics for all models.

TABLE 4.11: Statistics for all models

	Structuring Element Width (mm)	IFC Entities	IFC Relations	Input File Size (MB)	Output File Size (MB)	BuildingParts/-Intallations	Conversion Time (s)	Artefact Area (%)	BuildingInstallation File Space (%)	Reduced Vertices (%)	Reduced Edges (%)	Detriangulation File Space Reduction (%)
IfcOpenHouse	0	46	28	0.10	0.05	0/1	6	0.00	11.11	0.00	44.87	65.22
FZK-House	0	111	249	1.49	0.68	0/6	272	0.00	3.38	0.14	48.23	73.03
FJK-House	300	274	450	13.90	1.12	0/12	217	0.95	76.16	5.01	47.87	62.13
Smiley West	300	627	2565	7.98	4.47	0/37	888	0.51	79.93	1.78	47.42	63.83
Niedriha	0	301	950	1.99	0.53	0/1	92	0.00	10.65	0.73	45.70	66.43
BIEN-ZENKER Jasmin-Sun	0	122	406	4.58	0.20	0/0	72	0.00	-	0.00	42.80	64.26
Office Building	0	1201	3611	2.70	2.25	0/2	612	0.00	3.56	0.00	42.99	62.78
Haus-G-H	300	301	1286	4.34	5.28	0/4	1672	1.24	81.44	4.31	43.76	64.49
Model 4351	0	442	2730	6.95	2.81	0/17	1041	0.00	85.12	0.00	48.63	69.49
1407 Opheusden WoZoCo	300	3521	4965	7.81	70.10	0/64	5495	0.54	94.56	1.36	45.56	61.90

4.6.3 Evaluation of the Possibilities for the Conversion to LoD4 models

The method that is developed for the creation of LoD4 Rooms for CityGML is different than the method for LoD3. Instead of generating the shell by transforming all relevant IFC geometries, only a single 'IfcSpace' object is used for to generate a room geometry. The created models are valid both geometrically and semantically. However, the implementation in the prototype is limited to 'Floor-', 'Ceiling-' and 'InteriorWallSurface'. It can be extended with 'Openings' by making use of the 'IfcBuildingElements' that are linked as 'space boundaries' to the 'IfcSpace'. 'Openings' (and 'ClosureSurfaces') between room in CityGML need to be linked either using XLinked surfaces or at the semantics level (see Figure 4.50). This needs to be taken into account when the methodology for LoD4 rooms is extended.

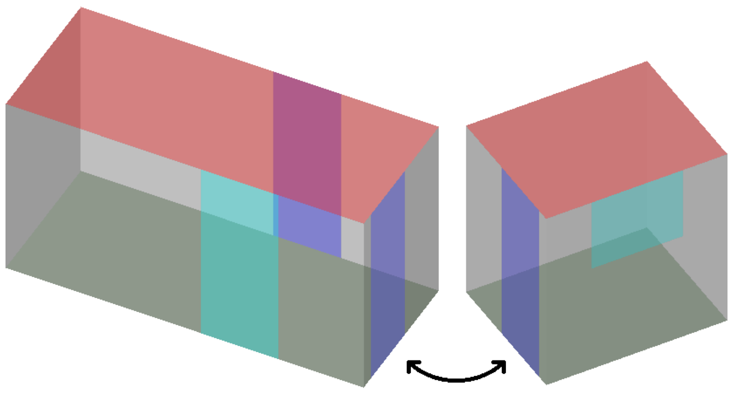


FIGURE 4.50: The Opening surfaces between room should either be linked using XLinks or linked at the semantic level

The alignment between the two standards is high with respect to the rooms and spaces in CityGML and IFC respectively. A required addition to the IFC specification is a property that indicates whether an 'IfcSpace' is inside or outside of the building. This would resolve the issue with balcony spaces as is first introduced in Figure 3.4. Furthermore, for the conversion of IFC to a full LoD4 model the mapping presented in Section 3.1.2 needs to be extended such that it also includes objects like utilities and furniture.

Chapter 5

Conclusions, Recommendations and Future Work

The methodology and its prototype implementation, presented in this thesis report allow for the automatic generation of CityGML LoD3 building models for use in 3D GIS applications. For optimal compatibility, the conversion is based on, and adheres to three standards: IFC, CityGML and ISO19107. To prove the effectiveness of the methodology a prototype application has been implemented and the output models created by the prototype validated. Conclusions drawn from this thesis are given in Section 5.1.

During the research several limitations of the IFC and CityGML standard found. Recommendations on both standards are given in Section 5.2 and Section 5.3 which can improve the interoperability between the two formats. Possibilities for future research on this topic are given in Section 5.4.

5.1 Conclusions

In this thesis it is shown that it is possible to generate valid and semantically rich CityGML LoD3 building models from IFC models. The validity requirements for CityGML LoD3 have been defined as well as the semantic information and geometric operations needed for the conversion. The effectiveness of the methodology has been successfully verified by the prototype implementation.

Also an initial investigation into the conversion from IFC to CityGML LoD4 is made. The geometric transformation cannot easily be extended to LoD4, due to the uncertainty of whether inner shells are rooms or not. A different approach is required and proposed which is based on the use of 'IfcSpaces'. 'IfcSpaces' are very similar in geometry compared to the rooms in CityGML. The geometric transformation is therefore much simpler. For a complete mapping of semantics more classes are required to be mapped, but the conversion is definitely possible.

This research provides the first automated method for generating valid CityGML building models with high detail. Previously, these models needed to be modelled manually. Other exiting methods for the conversion do not transform the IFC geometries and the semantics mapping is limited. Therefore, the results from those methods do not conform to the CityGML standard.

IFC has been accepted as an open standard by the Dutch 'Forum en College Standaardisatie' [13]. This will increase the adoption of IFC by Dutch governmental organisations and thereby the availability of IFC models. The automated conversion to CityGML makes it feasible to create high detail city models that are optimized for analyses. Such city model can also be kept up-to-date easily by merging changes whenever an IFC building model is updated. Architects may use the results to evaluate the interaction, like noise and heat transfer, between the neighbouring buildings and the environment, thereby making better informed design choices. Real estate agents can query the city models for houses with specific features, like large south facing windows.

The semantic part of the methodology is focussed on 'BoundarySurface' type semantics, but the conversion is not limited to transferring semantics at the face level. The methodology can easily be extended to also support attributes for solids like a 'Building'. The building's address in IFC can for instance trivially be mapped to CityGML, and every face can be linked to the originating 'IfcObject'. The implementation can therefore be extended to fit specific needs.

The semantic mapping between IFC and CityGML LoD3 requires changes to the standards for better alignment. Limitations in the conversion occur mostly due to information missing in the semantics of IFC. By making relatively small adjustments to the IFC standard, which are provided in Section 5.2, IFC and CityGML could be aligned such that for every model that complies to the IFC standard it can be guaranteed that the conversion to CityGML will be flawless.

The use of the methods from this thesis is not limited to the conversion from IFC to CityGML. The mapping is useful in other research trying to bridge the gap between the two formats. For example the reverse conversion or the creation of a UBM, which incorporate the needs of BIM and GIS users in one format [26]. The geometric transformation can even be applied to a wider field. The geometric transformation presented in this thesis is able to extract the exterior shell and also generalise further as is shown in Figure 4.42e. In general, any Computer-Aided Design (CAD) model can be simplified using the presented methodology. Furthermore, the geometric refinements can be used to optimize geometry for analysis by ensuring 2-manifold shells and removing degeneracies.

5.2 Recommendations for IFC

During this thesis project a new version of IFC (IFC4) was released. The new version brings georeferencing of building models and adds more classes. This is a good development and should be continued in the future. In this section recommendations are given for changes to the standard such that it can become better aligned with the CityGML standard.

A major step for the alignment of the two formats can be made by creating 'IfcSpaces' not only for rooms, but also for the exterior shell. It is already possible to define such a space, but there is no standardized way of storing the information that a particular space is an exterior shell. The alignment can be achieved by defining in the standard what parts of the building should be part of the exterior shell and how it can be stored as a special 'IfcSpace'. Software packages for the creation of IFC models could implement the geometric transformation methods defined in this thesis for the automatic generation of the exterior shell.

An exterior shell space in IFC would make the most difference, but there are other changes which can help the alignment as well. For 'IfcSpaces' in general it is useful to know not only whether a space touches the exterior, but also whether it is contained within the building is beneficial. For example the space for a balcony can then be ignored, whereas its geometry would otherwise be used as 'ClosureSurface'.

For balconies and dormers it is currently impossible to extract them as 'BuildingInstallations' without using complex feature recognition techniques, because IFC does not provide semantics for them. In IFC a new type of 'IfcSpatialStructureElement' could be created such that they can be recognized. At the same time this would also indicate that 'IfcSpace', which is part of the balcony spatial structure, should be ignored during the extraction of the exterior shell.

Although it is already possible in IFC to model columns and beams as part of a wall, it is not yet common practice. Depending on whether a column is part of a wall or not, it is modelled in CityGML by a 'WallSurface' or a 'BuilldingInstallation' respectively after the conversion. By defining in the IFC standard that columns and beams have to be part of a wall if it is applicable, the conversion no longer requires user input during the conversion.

5.3 Recommendations for CityGML

The CityGML leaves a lot of the modelling decisions open to the implementers. By providing a more detailed specification it becomes easier for modellers to know how to model, and for users to know what to expect. Several associations, like SIG3D [36] and Stoter et al. [6], provide their own detailed specifications. The CityGML standard should include a more detailed specification before fragmentation occurs.

The standard should clearly define when and how a building should be subdivided into 'BuildingParts' and 'BuildingInstallations'. Especially how 'BuildingInstallations' should be generalized. Due to the complex geometry of 'BuildingInstallations', like stairs and fences, the amount of file size is disproportional to that of the building geometry. However, CityGML does not provide any information on how the generalization should be applied.

Additionally, for the boundary surface types it should be made clearer when something belongs to a specific type or not. For example, it should be specified whether only glass should be modelled as window or also its frame. Furthermore, due to the inherent generalization required by the use of B-rep a specification is needed on how non-planar geometry should be modelled. For instance, how many planar faces are required to model an arch.

5.4 Future Work

The methodology leaves little room for improvement. The semantic mapping requires additional semantics in order for it to be improved and the geometric transformation works. On the short term support for the new IFC4 classes can be added, though due to its novelty very few IFC4 models are available yet. Also the more trivial attributes like the address can be mapped.

Given some more time, the methodology can be developed for extracting the terrain intersection curve. The terrain intersection curve not only provides where the terrain surface touches the building, but also enables the 'GroundSurface' to be generated properly. The implementation would benefit most from an internal spatial data structure to be used during the conversion which supports semantics, non-manifold solids and Boolean operations.

The long term goal is to achieve a high degree of interoperability between IFC and CityGML. This not only requires better alignment of the standards, but also methodologies for the conversion to other city objects like, tunnels and bridges, and for the creation of building models with multiple LoDs from IFC. The lower LoD models can be generated from the LoD3 model. By doing so, instead of making converters for each LoD separately, the different LoD models can be made in a geometrically consistent and semantically connected way (see Figure 5.1). The conversion methodology presented in this report can be used as the foundation for all.

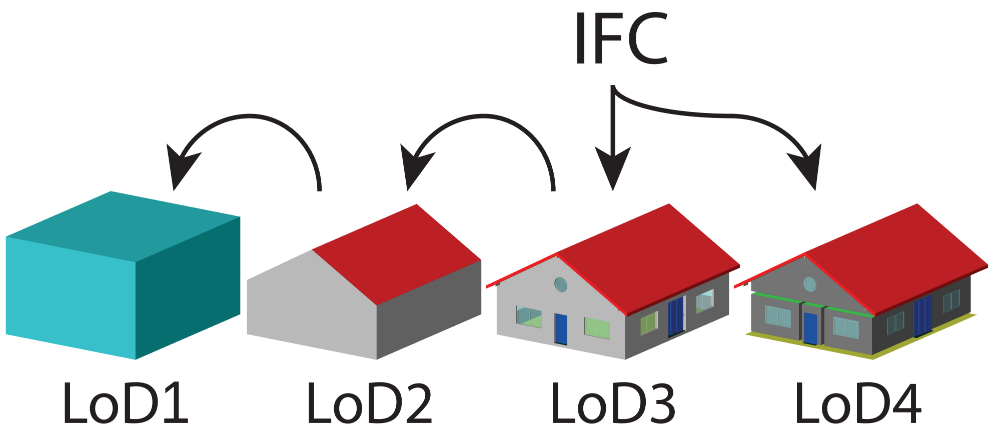
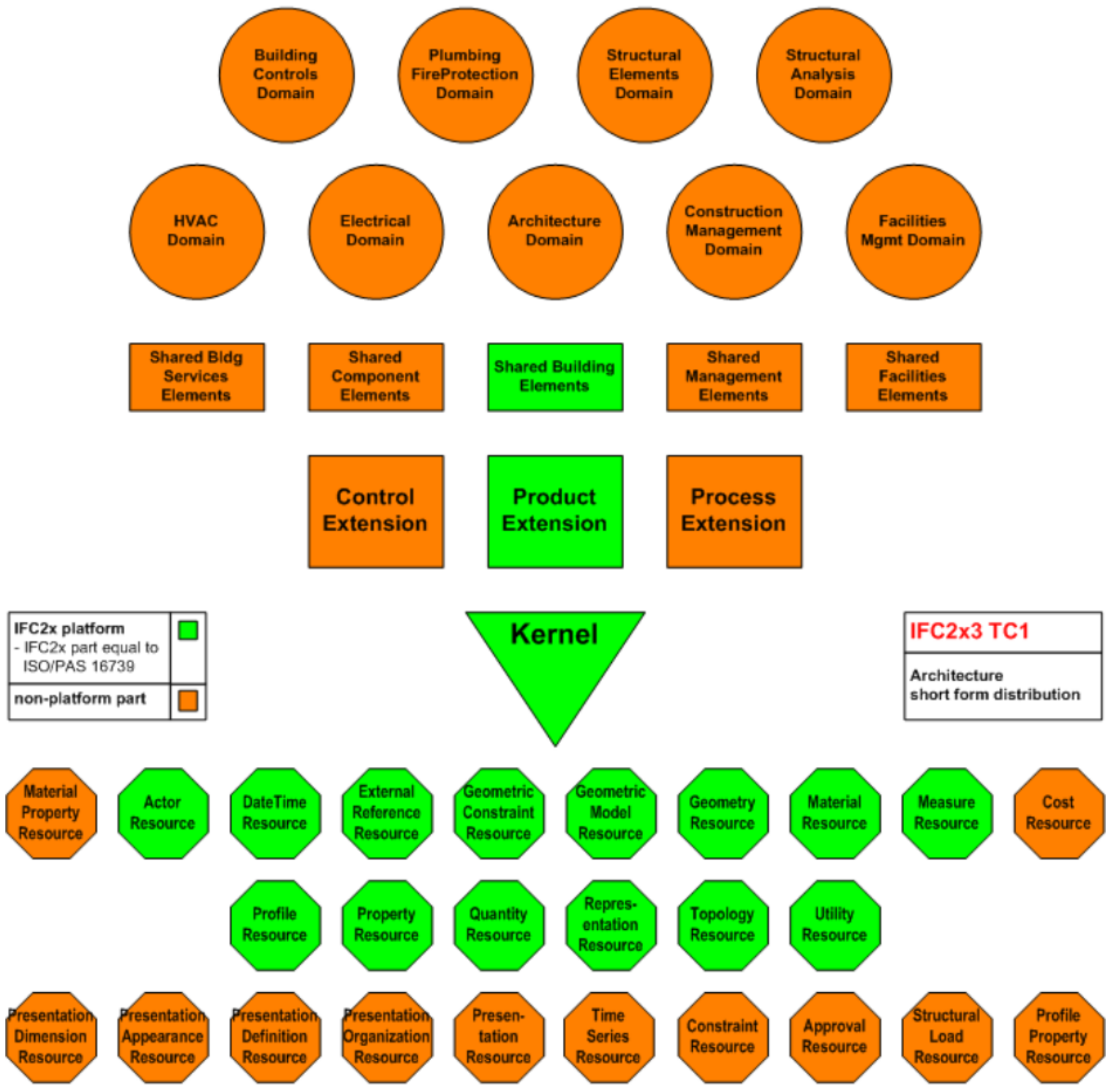


FIGURE 5.1: For consistency the lower LoDs can and should be generated from the LoD3 model

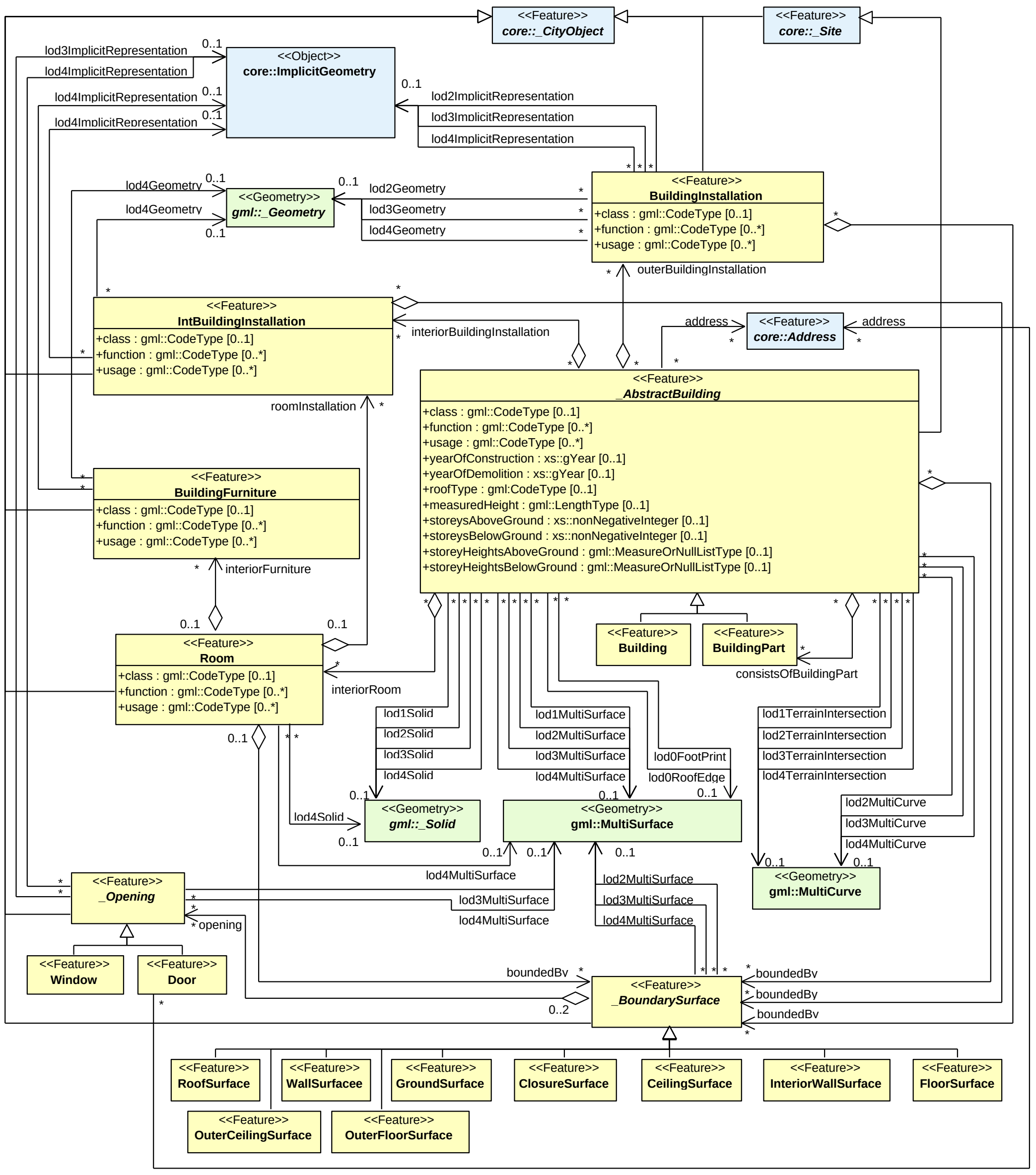
Appendix A

IFC 2x Platform Architecture


Building Controls Domain
Plumbing FireProtection Domain
Structural Elements Domain
Structural Analysis Domain
HVAC Domain
Electrical Domain
Architecture Domain
Construction Management Domain
Facilities Mgmt Domain
Shared Bldg Services Elements
Shared Component Elements
Shared Building Elements
Shared Management Elements
Shared Facilities Elements
Control Extension
Product Extension
Process Extension
IFC2x platform - IFC2x part equal to ISO/PAS 16739
non-platform part
Kernel
IFC2x3 TC1
Architecture short form distribution
Material Property Resource
Actor Resource
DateTime Resource
External Reference Resource
Geometric Constraint Resource
Geometric Model Resource
Geometry Resource
Material Resource
Measure Resource
Cost Resource
Profile Resource
Property Resource
Quantity Resource
Repres-entation Resource
Topology Resource
Utility Resource
Presentation Dimension Resource
Presentation Appearance Resource
Presentation Definition Resource
Presentation Organization Resource
Presen-tation Resource
Time Series Resource
Constraint Resource
Approval Resource
Structural Load Resource
Profile Property Resource

Appendix B

CityGML 2.0.0 Building Module



<<Feature>> core::_CityObject
<<Feature>> core::_Site
<<Object>> core::ImplicitGeometry
<<Geometry>> gml::_Geometry
<<Feature>> BuildingInstallation
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
<<Feature>> IntBuildingInstallation
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
<<Feature>> core::Address
<<Feature>> _AbstractBuilding
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
+yearOfConstruction : xs::gYear [0..1]
+yearOfDemolition : xs::gYear [0..1]
+roofType : gml:CodeType [0..1]
+measuredHeight : gml::LengthType [0..1]
+storeysAboveGround : xs::nonNegativeInteger [0..1]
+storeysBelowGround : xs::nonNegativeInteger [0..1]
+storeyHeightsAboveGround : gml::MeasureOrNullListType [0..1]
+storeyHeightsBelowGround : gml::MeasureOrNullListType [0..1]
<<Feature>> BuildingFurniture
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
<<Feature>> Room
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
<<Feature>> Building
<<Feature>> BuildingPart
<<Geometry>> gml::_Solid
<<Geometry>> gml::MultiSurface
<<Geometry>> gml::MultiCurve
<<Feature>> _Opening
<<Feature>> Window
<<Feature>> Door
<<Feature>> _BoundarySurface
<<Feature>> RoofSurface
<<Feature>> WallSurfacee
<<Feature>> GroundSurface
<<Feature>> ClosureSurface
<<Feature>> CeilingSurface
<<Feature>> InteriorWallSurface
<<Feature>> FloorSurface
<<Feature>> OuterCeilingSurface
<<Feature>> OuterFloorSurface
lod3ImplicitRepresentation
lod4ImplicitRepresentation
lod4ImplicitRepresentation
lod4ImplicitRepresentation
lod2ImplicitRepresentation
lod3ImplicitRepresentation
lod4ImplicitRepresentation
lod4Geometry
lod4Geometry
lod2Geometry
lod3Geometry
lod4Geometry
outerBuildingInstallation
interiorBuildingInstallation
address
address
roomInstallation
interiorFurniture
interiorRoom
consistsOfBuildingPart
lod1Solid
lod2Solid
lod3Solid
lod4Solid
lod4Solid
lod1MultiSurface
lod2MultiSurface
lod3MultiSurface
lod4MultiSurface
lod0FootPrint
lod0RoofEdge
lod1TerrainIntersection
lod2TerrainIntersection
lod3TerrainIntersection
lod4TerrainIntersection
lod2MultiCurve
lod3MultiCurve
lod4MultiCurve
lod4MultiSurface
lod3MultiSurface
lod4MultiSurface
lod2MultiSurface
lod3MultiSurface
lod4MultiSurface
opening
boundedBy
boundedBy
boundedBy
boundedBy
0..1
0..2
*

Appendix C

Conversion Workflow Diagrams

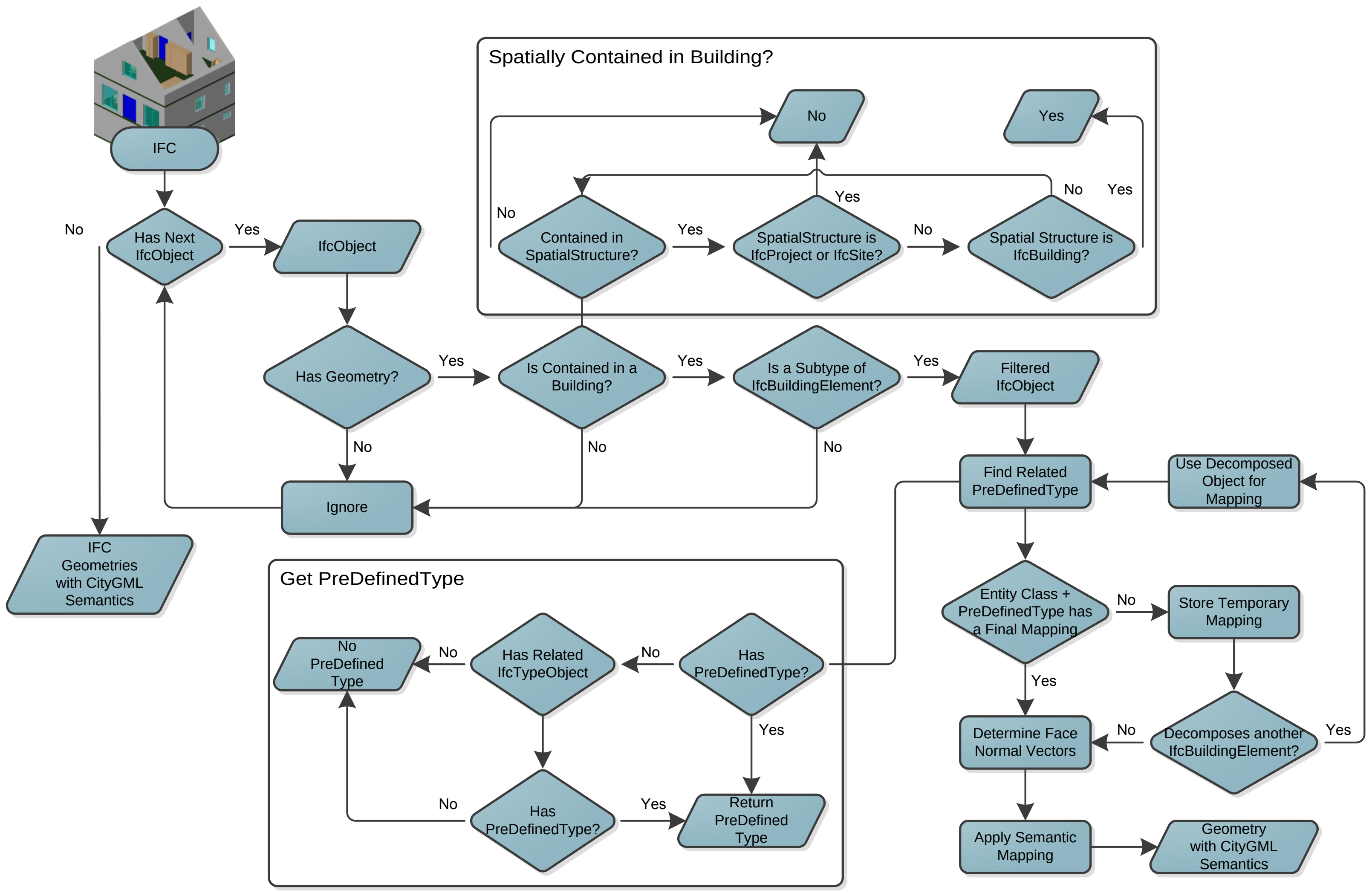

IFC
Has Next IfcObject
No
Yes
IfcObject
Has Geometry?
Yes
No
Ignore
IFC Geometries with CityGML Semantics
Spatially Contained in Building?
No
Yes
No
Contained in SpatialStructure?
Yes
SpatialStructure is IfcProject or IfcSite?
Yes
No
Spatial Structure is IfcBuilding?
No
Yes
Is Contained in a Building?
Yes
No
Is a Subtype of IfcBuildingElement?
Yes
No
Filtered IfcObject
Find Related PreDefinedType
Use Decomposed Object for Mapping
Entity Class + PreDefinedType has a Final Mapping
No
Store Temporary Mapping
Yes
Determine Face Normal Vectors
No
Decomposes another IfcBuildingElement?
Yes
Apply Semantic Mapping
Geometry with CityGML Semantics
Get PreDefinedType
No PreDefined Type
No
Has Related IfcTypeObject
No
Has PreDefinedType?
Yes
No
Has PreDefinedType?
Yes
Return PreDefined Type

FIGURE C.1: Workflow diagram for the filtering and mapping of semantics

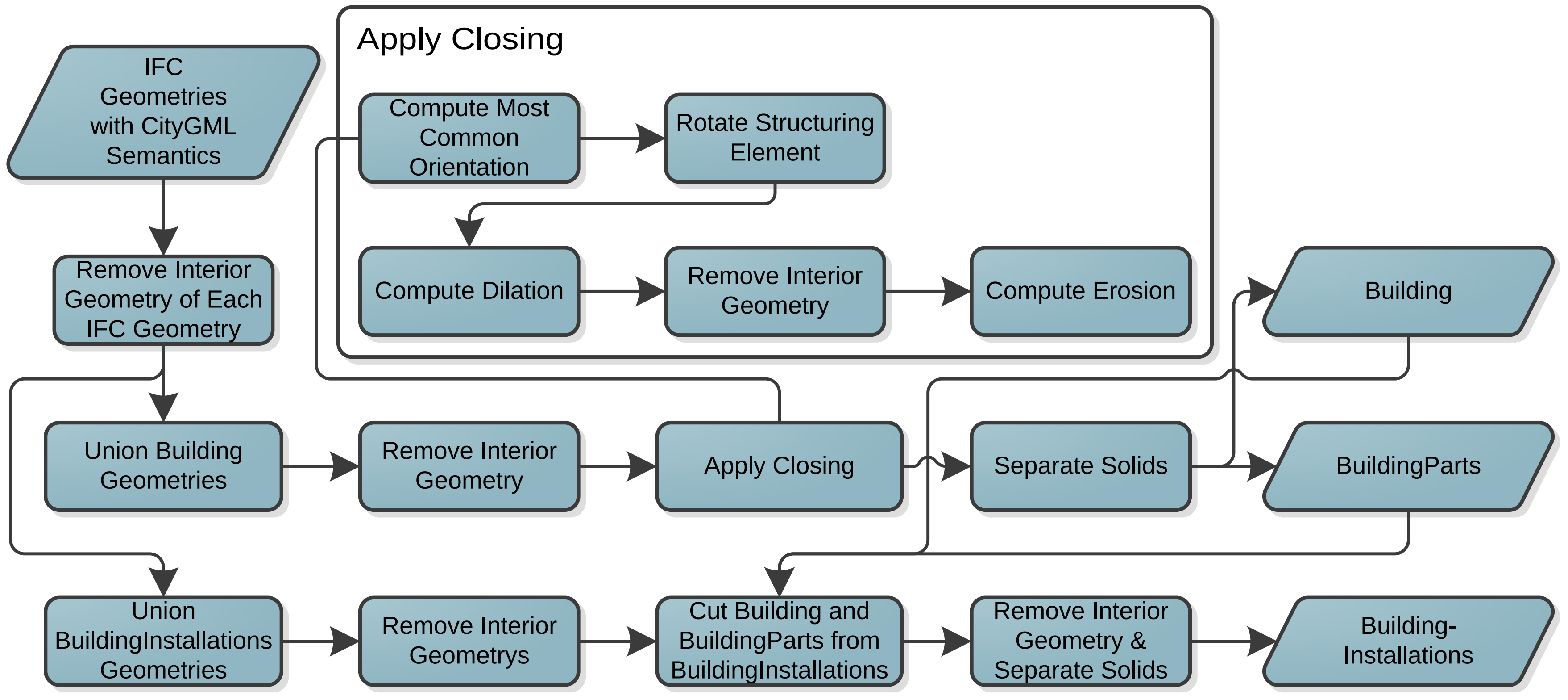

IFC Geometries with CityGML Semantics
Apply Closing
Compute Most Common Orientation
Rotate Structuring Element
Compute Dilation
Remove Interior Geometry
Compute Erosion
Remove Interior Geometry of Each IFC Geometry
Building
Union Building Geometries
Remove Interior Geometry
Apply Closing
Separate Solids
BuildingParts
Union BuildingInstallations Geometries
Remove Interior Geometrys
Cut Building and BuildingParts from BuildingInstallations
Remove Interior Geometry & Separate Solids
Building-Installations

FIGURE C.2: Workflow diagram for the geometric transformation

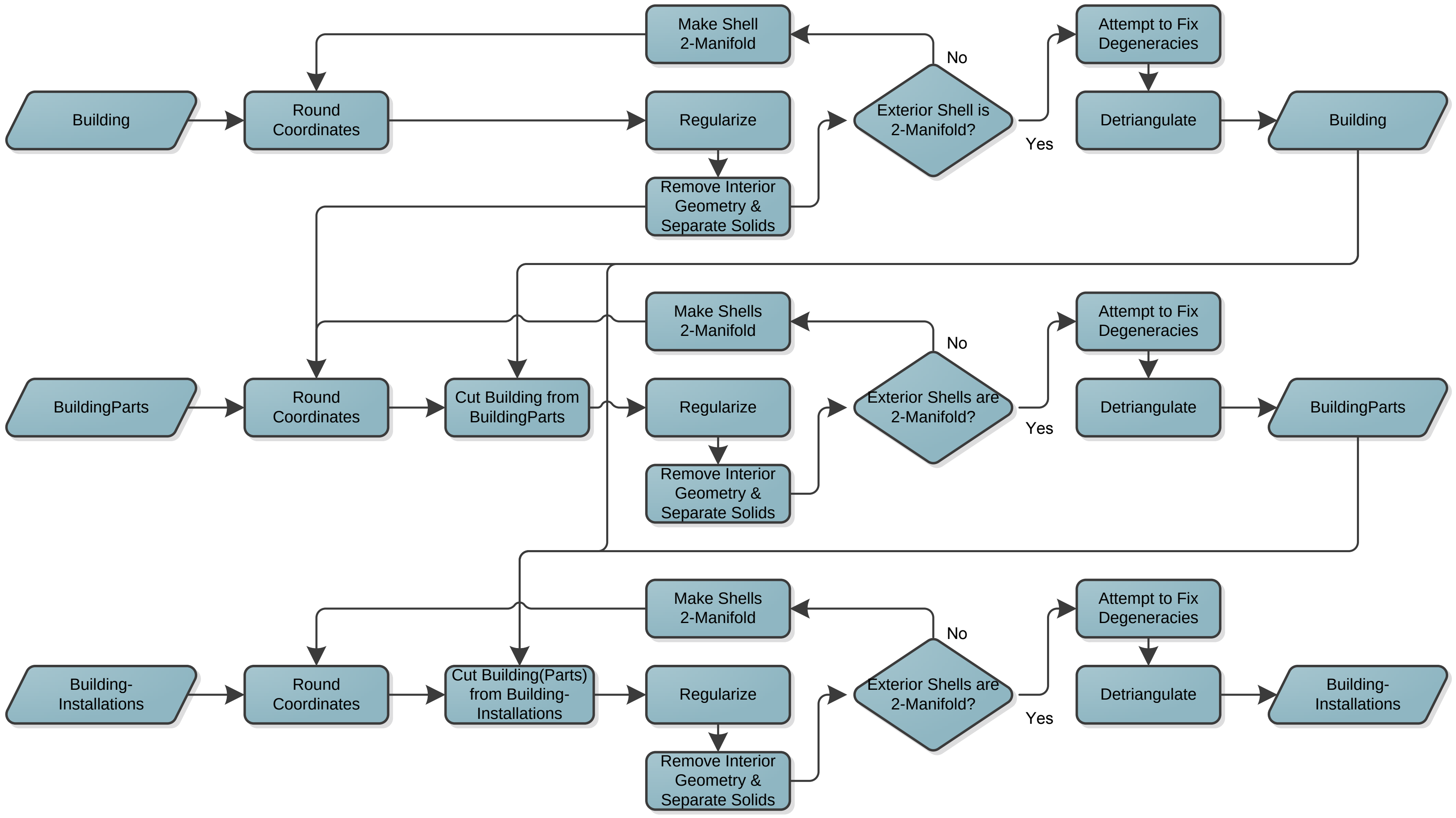

Building
Round Coordinates
Regularize
Remove Interior Geometry & Separate Solids
Exterior Shell is 2-Manifold?
No
Make Shell 2-Manifold
Yes
Attempt to Fix Degeneracies
Detriangulate
Building
BuildingParts
Round Coordinates
Cut Building from BuildingParts
Regularize
Remove Interior Geometry & Separate Solids
Exterior Shells are 2-Manifold?
No
Make Shells 2-Manifold
Yes
Attempt to Fix Degeneracies
Detriangulate
BuildingParts
Building-Installations
Round Coordinates
Cut Building(Parts) from Building-Installations
Regularize
Remove Interior Geometry & Separate Solids
Exterior Shells are 2-Manifold?
No
Make Shells 2-Manifold
Yes
Attempt to Fix Degeneracies
Detriangulate
Building-Installations

FIGURE C.3: Workflow diagram for the refinement of the geometry

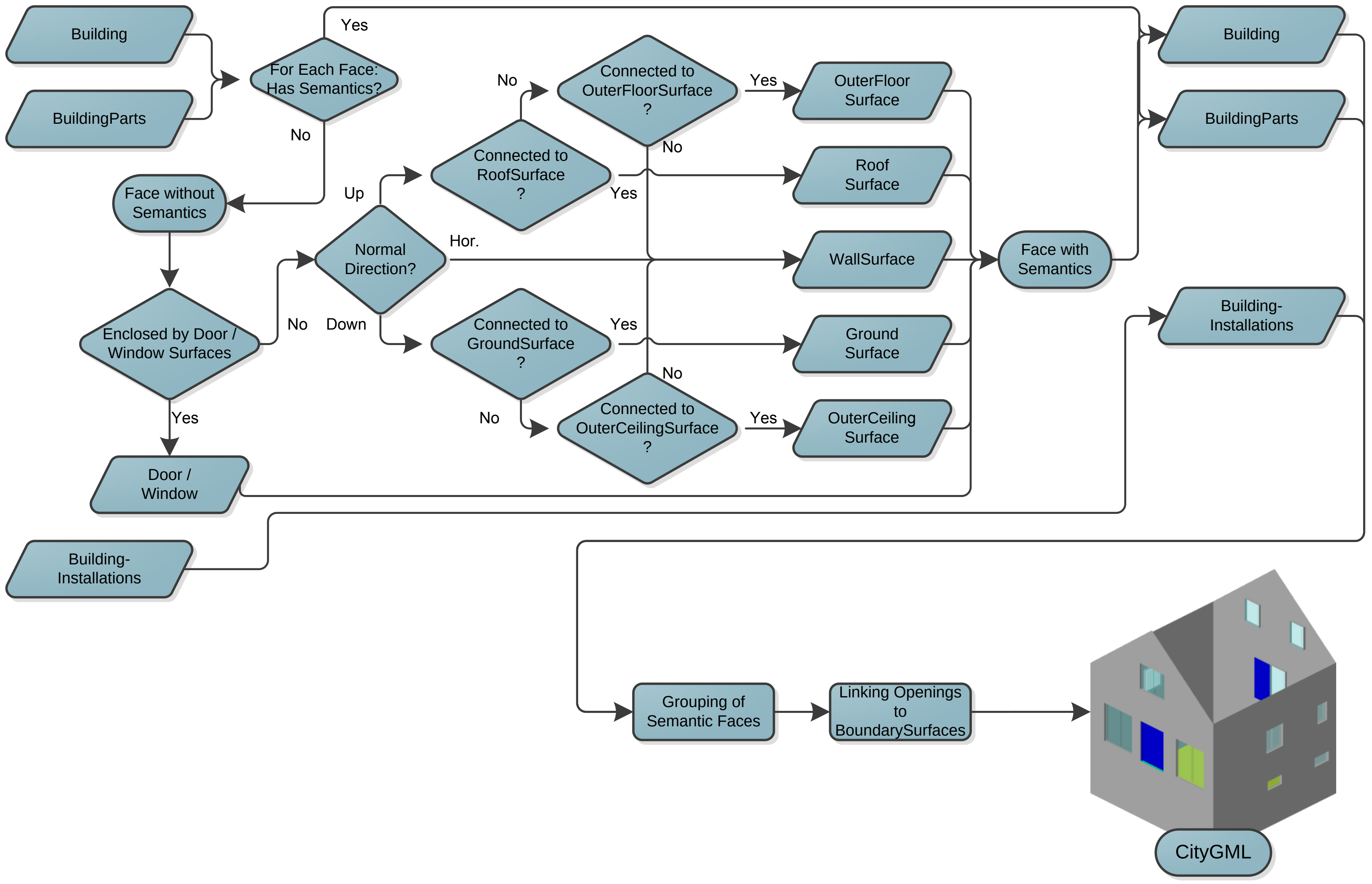


FIGURE C.4: Workflow diagram for the refinement of semantics after which the conversion is completed

Bibliography

[1] Umit Isikdag and Sisi Zlatanova. Towards defining a framework for automatic generation of buildings in CityGML using building information models. *3D Geo-Information Sciences*, pages 79–96, 2009.

[2] Umit Isikdag, Jason Underwood, and Ghassan Aouad. An investigation into the applicability of building information models in geospatial environment in support of site selection and fire response management processes. *Advanced engineering informatics*, 22(4):504–519, 2008.

[3] Ruben Laat and Léon Berlo. Integration of BIM and GIS: The development of the CityGML GeoBIM extension. *Advances in 3D Geo-Information Sciences*, pages 211–225, 2011.

[4] Sisi Zlatanova and Jakob Beetz. 3D spatial information infrastructure: The case of port rotterdam. In *Usage, Usability, and Utility of 3D City Models-European COST Action TU0801, Proceedings of the conference held 29-31 October, 2012 in Nantes, France. Edited by Th. Leduc, G. Moreau, and R. Billens, id. 03010, 8 pp.*, volume 1, page 03010, 2012.

[5] Jacob Beetz. Gebruik van 3D gebouw-modellen in 3D GIS: Building information modelling (BIM/IFC) voor beginners. Geonovum 3D Pilot Fase II Slotdag, November 2012.

[6] Jantien Stoter, Jacob Beetz, Hugo Ledoux, Marcel Reuvers, Rick Klooster, Paul Janssen, Friso Penninga, Sisi Zlatanova, and Linda den Brink. Implementation of a national 3D standard: Case of the Netherlands. *Progress and New Trends in 3D Geoinformation Sciences*, pages 277–298, 2013.

[7] Claudia Schulte and Volker Coors. Development of a CityGML ADE for dynamic 3D flood information. In *Joint ISCRAM-CHINA and GI4DM Conference on Information Systems for Crisis Management*, 2008.

[8] Aneta Strzalka, Nazmul Alam, Eric Duminil, Volker Coors, and Ursula Eicker. Large scale integration of photovoltaics in cities. *Applied Energy*, 93:413–421, 2012.

[9] Shinya Yasumoto, Andrew Jones, Keiji Yano, and Tomoki Nakaya. Virtual city models for assessing environmental equity of access to sunlight: a case study of Kyoto, Japan. *International Journal of Geographical Information Science*, 26(1): 1–13, 2012.

[10] Roberto Rodrigues, António Coelho, and Luís Paulo Reis. Data model for procedural modelling from textual descriptions. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[11] Open Geospatial Consortium. OGC City Geography Markup Language CityGML encoding standard version 2.0.0. 2012.

[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/index.htm`.

[13] Forum en College Standaardisatie. Lijst met open standaarden - ifc, November 2011. URL `https://lijsten.forumstandaardisatie.nl/open-standaard/ifc`.

[14] Linda van den Brink, Dick Krijtenburg, Hans van Eekelen, and Bart Maessen. Basisregistratie grootschalige topografie - Gegevenscatalogus BGT 1.1.1, July 2013.

[15] Thomas H Kolbe. Representing and exchanging 3D city models with CityGML. *3D geo-information sciences*, pages 15–31, 2009.

[16] Karl-Heinz Häfele. CityGML model of the FJK-Haus. Karlsruher Institut für Technologie, March 2011. URL `http://www.iai.fzk.de/www-extern/index.php?id=2196&L=1`.

[17] Jan Blaauboer, Joris Goos, Hugo Ledoux, Friso Penninga, Marcel Reuvers, Jantien Stoter, and George Vosselman. 3D pilot eindrapport werkgroep technische specificaties voor de opbouw van 3D IMGeo-CityGML. 2012.

[18] Mohamed El-Mekawy and Anders Östman. Semantic mapping: an ontology engineering method for integrating building models in IFC and CityGML. *Proceedings of the 3rd ISDE Digital Earth Summit*, pages 12–14, 2010.

[19] Hugo Ledoux. On the validation of solids represented with the international standards for geographic information. *Computer-Aided Civil and Infrastructure Engineering*, 28(9):693–706, October 2013.

[20] Claus Nagel, Alexandra Stadler, and Thomas H Kolbe. Conceptual requirements for the automatic reconstruction of building information models from uninterpreted 3D models. In *Academic Track of Geoweb 2009 Conference, Vancouver*, 2009.

[21] Claus Nagel. Ableitung verschiedener detaillierunsstufen von IFC gebaudemodellen. Master's thesis, 2006.

[22] Mohamed El-Mekawy, Anders Östman, and Ihab Hijazi. An evaluation of IFC-CityGML unidirectional conversion. *International Journal of Advanced Computer Science and Applications*, 3(5):159–171, 2012.

[23] Thomas Lieblich. IFC2x3 model implementation guide. buildingSMART International, 2009.

[24] Saïd Izza. Integration of industrial information systems: from syntactic to semantic integration approaches. *Enterprise Information Systems*, 3(1):1–57, 2009.

[25] I-Chen Wu and Shang-Hsien Hsieh. Transformation from IFC data model to GML data model: methodology and tool development. volume 30, pages 1085–1090. Taylor & Francis, 2007.

[26] Mohamed El-Mekawy, Anders Östman, and Khurram Shahzad. Towards interoperating CityGML and IFC building models: A unified model based approach. *Advances in 3D Geo-Information Sciences*, pages 73–93, 2011.

[27] Rizal Sebastian and Léon van Berlo. Tool for benchmarking BIM performance of design, engineering and construction firms in the Netherlands. *Architectural Engineering and Design Management*, 6(4):254–263, 2010.

[28] buildingSMART, 2013. URL `http://www.buildingsmart-tech.org/`.

[29] Thomas H Kolbe, Claus Nagel, and Alexandra Stadler. CityGML–a framework for the representation of 3D city models from geometry acquisition to full semantic qualification. In *Proc. of ISPRS Congress*, 2008.

[30] Claus Nagel and T. H. Kolbe. Conversion of IFC to CityGML. Technische Universität Berlin, OGC 3DIM WG, Paris, July 2007.

[31] TH Kolbe and L Plumer. Bridging the gap between gis and caad geometry, referencing, representations, standards and semantic modelling. *GIM international*, 18: 12–38, 2004.

[32] WP Wang and KK Wang. Geometric modeling for swept volume of moving solids. *Computer Graphics and Applications, IEEE*, 6(12):8–17, 1986.

[33] 2013. URL `http://commons.wikimedia.org/wiki/Main_Page`.

[34] André Monteiro and João Pedro Poças Martins. SIGABIM: a framework for BIM application. In *XXXVIII IAHS World Congress*, 2012.

[35] T. H. Kolbe. 100% CityGML - Introduction to CityGML, March 2011. URL http://collegerama.tudelft.nl/Mediasite/Play/7b440617cd1342b0b5b006fc0f6563ef1d.

[36] Special Interest Group 3D (SIG3D). Handbuch für die modellierung von 3D objekten - Teil 2: Modellierung gebäude (LoD1, LoD2 und LoD3), 2013. URL http://wiki.quality.sig3d.org/index.php/Handbuch_f%C3%BCr_die_Modellierung_von_3D_Objekten_-_Teil_2:_Modellierung_Geb%C3%A4ude_%28LOD1,_LOD2_und_LOD3%29.

[37] Christoph M Hoffmann. *Geometric and solid modeling: an introduction.* Morgan Kaufmann Publishers Inc., 1989.

[38] Mario Botsch and Leif Kobbelt. A robust procedure to eliminate degenerate faces from triangle meshes. In *VMV*, pages 283–290. Citeseer, 2001.

[39] Peter van Oosterom, Wilko Quak, and Theo Tijssen. About invalid, valid and clean polygons. In *Developments In Spatial Data Handling*, pages 1–16. Springer, 2005.

[40] Roeland Boeters. MSc thesis in Geomatics: Automatic enhancement of CityGML LoD2 models with interiors and its usability for net internal area determination. June 2013.

[41] Karl-Heinz Häfele. Personal communication via email, 2013.

[42] Safe Software. Converting BIM IFC data to CityGML, July 2012. URL http://fmepedia.safe.com/articles/Samples_and_Demos/Converting-BIM-IFC-data-to-CityGML.

[43] Xiangqian Jiang, Shan Lou, and Paul J Scott. Morphological method for surface metrology and dimensional metrology based on the alpha shape. *Measurement Science and Technology*, 23(1):015003, 2012.

[44] Wesley E Snyder, R Groshong, M Hsiao, KL Boone, and T Hudacko. Closing gaps in edges and surfaces. *Image and Vision Computing*, 10(8):523–531, 1992.

[45] Michael Schwarz and Hans-Peter Seidel. Fast parallel surface and solid voxelization on gpus. In *ACM Transactions on Graphics (TOG)*, volume 29, page 179. ACM, 2010.

[46] Desmond Rainsford and William Mackaness. Template matching in support of generalisation of rural buildings. In *Advances in Spatial Data Handling*, pages 137–151. Springer, 2002.

[47] Roger M Dufour, Eric L Miller, and Nikolas P Galatsanos. Template matching based object recognition with unknown geometric parameters. *Image Processing, IEEE Transactions on*, 11(12):1385–1396, 2002.

[48] Martin Kada. Generalization of 3d building models for map-like presentations. *T he International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences: XXXVII.[S. l.]: ISPRS*, pages 399–404, 2008.

[49] Leif P Kobbelt, Jens Vorsatz, and Ulf Labsik. A shrink wrapping approach to remeshing polygonal surfaces. In *Computer Graphics Forum*, volume 18, pages 119–130. Wiley Online Library, 1999.

[50] Z. Zhao, H. Ledoux, and J. Stoter. Automatic repair of CityGML LoD2 buildings using shrink-wrapping. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-2/W1:309–317, 2013. doi: 10.5194/isprsannals-II-2-W1-309-2013. URL `http://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/II-2-W1/309/2013/`.

[51] Shuangshuang Jin, Robert R Lewis, and David West. A comparison of algorithms for vertex normal computation. *The Visual Computer*, 21(1-2):71–82, 2005.

[52] Herbert Edelsbrunner and Ernst P Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics (TOG)*, 13(1):43–72, 1994.

[53] Stephan Bischoff, Darko Pavic, and Leif Kobbelt. Automatic restoration of polygon models. *ACM Transactions on Graphics (TOG)*, 24(4):1332–1352, 2005.

[54] Radomír Měch and Przemyslaw Prusinkiewicz. Visual models of plants interacting with their environment. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 397–410. ACM, 1996.

[55] Peter Wonka, Michael Wimmer, François Sillion, and William Ribarsky. *Instant architecture*, volume 22. ACM, 2003.

[56] Martti Mäntylä. *An introduction to solid modeling.* Computer Science Press, New York, USA, 1988.

[57] Marco Attene. A lightweight approach to repairing digitized polygon meshes. *The Visual Computer*, 26(11):1393–1406, 2010.

[58] IfcOpenShell - Open source IFC geometry engine, February 2013. URL `http://ifcopenshell.org/`.

[59] OPEN CASCADE S.A.S. Open CASCADE Technology - 3D modeling & numerical simulation, 2013. URL `http://www.opencascade.org/`.

[60] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[61] Geomview. OFF files - Polyhedra: polygons with shared vertices, 2007. URL `http://www.geomview.org/docs/html/OFF.html`.

[62] Hervé Brönnimann, Andreas Fabri, Geert-Jan Giezeman, Susan Hert, Michael Hoffmann, Lutz Kettner, Sylvain Pion, and Stefan Schirra. 2D and 3D geometry kernel. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.3 edition, 2013. URL `http://doc.cgal.org/4.3/Manual/packages.html#PkgKernel23Summary`.

[63] Kevin Weiler. Edge-based data structures for solid modeling in curved-surface environments. *Computer Graphics and Applications, IEEE*, 5(1):21–40, 1985.

[64] Lutz Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry*, 13(1):65–90, 1999.

[65] Miguel Granados, Peter Hachenberger, Susan Hert, Lutz Kettner, Kurt Mehlhorn, and Michael Seel. Boolean operations on 3d selective nef complexes: Data structure, algorithms, and implementation. In *Algorithms-ESA 2003*, pages 654–666. Springer, 2003.

[66] Lutz Kettner. Halfedge data structures. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.3 edition, 2013. URL `http://doc.cgal.org/4.3/Manual/packages.html#PkgHDSSummary`.

[67] Peter Hachenberger and Lutz Kettner. 3D Boolean operations on Nef polyhedra. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.3 edition, 2013. URL `http://doc.cgal.org/4.3/Manual/packages.html#PkgNef3Summary`.

[68] Katrin Dobrindt, Kurt Mehlhorn, and Mariette Yvinec. *A complete and efficient algorithm for the intersection of a general and a convex polyhedron*. Springer, 1993.

[69] Peter Hachenberger. 3D Minkowski sum of polyhedra. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.3 edition, 2013. URL `http://doc.cgal.org/4.3/Manual/packages.html#PkgMinkowskiSum3Summary`.

[70] Efi Fogel and Dan Halperin. Exact and efficient construction of Minkowski sums of convex polyhedra with applications. *Computer-Aided Design*, 39(11):929–940, 2007.

[71] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. Computational geometry: Algorithms and applications. *Springer-Verlag*, 1997.

[72] Pierre Alliez, Stéphane Tayeb, and Camille Wormser. 3D fast intersection and distance computation (AABB tree). In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.3 edition, 2013. URL `http://doc.cgal.org/4.3/Manual/packages.html#PkgAABB_treeSummary`.

[73] Martijn Meijers. Simultaneous & topologically-safe line simplification for a variable-scale planar partition. In *Advancing Geoinformation Science for a Changing World*, pages 337–358. Springer, 2011.

[74] Matthias Weise, Thomas Liebich, Richard See, Vladimir Bazjanac, Tuomas Laine, and Benjamin Welle. Implementation guide: Space boundaries for energy analysis, 2011.

[75] Vladimir Bazjanac. Space boundary requirements for modeling of building geometry for energy and other performance simulation. In *CIB W78: 27th International Conference*, 2010.