Student thesis series INES nr 559

Conforming building models derived in Esri's environment to the Swedish national specification for 3D data

Guðrún Jensdóttir

2021
Department of
Physical Geography and Ecosystem Science
Lund University
Sölvegatan 12
S-223 62 Lund
Sweden

Guðrún Jensdóttir (2021).

Working title: Conforming building models derived in Esri's environment to the Swedish national specification for 3D data. (English)

Master degree thesis, 30 credits in *Geomatics*

Department of Physical Geography and Ecosystem Science, Lund University

Level: Master of Science (MSc)

Course duration: *January* 2021 until *September* 2021

Conforming building models derived in Esri's environment to the Swedish national specification for 3D data.

Guðrún Jensdóttir

Master thesis, 30 credits, in *Geomatics*

Supervisors:

Helena Elvén Eriksson
Department of Physical Geography and Ecosystem Science

Per-Ola Olsson
Department of Physical Geography and Ecosystem Science

Exam committee:
Pengxiang Zhao, Department of Physical Geography and Ecosystem Science
Olive Niyomubyeyi, Department of Physical Geography and Ecosystem Science

Acknowledgments

My utmost thanks go to my supervisors, Per-Ola, for providing excellent guidance and support throughout the realization of this project, and Helena, for providing thoughtful guidance and advice along the way. My thanks also extend to Max Huber for reading through the thesis and providing valuable and constructive feedback. I would also like to thank Babacar Ndiaye and Ann-Sofie Silverskar from the municipality of Karlskrona for providing me with the necessary data and information on their 3D city modeling process.

Thank you, Johanna Asch, Max Huber, and Max Mangold, for the many enjoyable study sessions and lunch breaks. I am grateful for the companionship and the support, which sustained my motivation levels during these unprecedented times. At last, I want to thank Tandri Gauksson for providing valuable insights and feedback, and most importantly, moral support.

Abstract

Three-dimensional (3D) city models are increasingly utilized among stakeholders of different sectors, particularly those dealing with building information. 3D city models are a three-dimensional digital representation of urban structures, most prominently buildings. They offer diverse applications and have gained wide acceptance as a support tool in planning practices.

The push to achieve greater interoperability in the GIS sector has been a motivating factor for standardized practices for 3D city models. National frameworks for 3D data have been established in several countries to facilitate information sharing between different actors such as municipalities or other administrative bodies. A national specification pertaining to 3D data is in development in Sweden, along with a standard aimed explicitly at 3D building models, denoted as NS building. Municipalities will eventually have to adjust their modeling practices to adhere to the standard in the realization of their 3D city models. Proprietary software, such as Esri, has been common to establish 3D city models in the context of municipalities in Sweden. The interoperability of pre-existing 3D building models derived in an Esri environment with NS building has not yet been addressed.

Considering that the development of the Swedish standard has come a long way, it is essential to test its applicability in different scenarios. Ensuring interoperability and data homogenization remains an important facet in GIS research. Since many municipalities in Sweden are working within Esri's platform, it is highly relevant to examine how the standard can be applied in this context.

This thesis proposes a workflow that transforms Esri-derived building models and conforms them to the NS building standard. This is performed to test the Swedish standard's applicability for building models in Esri MultiPatch format. A spatial Extract, Transform, and Load (ETL) process facilitates the conversion, using Safe Software's Feature Manipulation Engine (FME). The conversion centers around making the geometries and attributes of a test dataset compatible with NS building models of different levels of detail (LoDs) through the process of de-aggregation, restructuring, and defining surfaces according to their types (bottom, roof, or wall surfaces). Subsequently, the output models are stored in the initial Esri file geodatabase (File GDB).

Keywords: Geomatics, GIS, 3D building models, 3D city models, conversion, Esri FileGDB, spatial ETL, FME, MultiPatch, national standard, NS building, transformation.

Contents

List of figures

List of tables

Abbreviations

2.5D	Two-and-a-half-dimensional
2D	Two-dimensional
3D	Three-dimensional
ADE	Application Domain Extension
CAD	Computer-Aided Design
COLLADA	Collaborative Design Activity
DEM	Digital Elevation Model
ETL	Extract, Transform, and Load
FileGDB	File Geodatabase
FK	Foreign Key
FME	Feature Manipulation Engine
GIS	Geographic Information System
GML	Geography Markup Language
IFC	Industry Foundation Classes
ISO	International Organization for Standardization
LiDAR	Light Detection and Ranging
LoD	Level of detail
NS	National specification/standard
OGC	Open Geospatial Consortium
PK	Primary Key
SHP	Shapefile
UML	Unified Modeling Language
XML	Extensible Markup Language
XSD	XML Schema Definition

1. Introduction

It has become a common practice among administrative bodies and municipalities around the world to add three-dimensional (3D) city models to their local data infrastructure for planning development. Their implementation allows for applications related to planning, visualization, decision-making, public-participation incentives, and many others to take place (Billen et al. 2014; Julin et al. 2018). 3D city models are increasingly used to support urban development and policymaking and have gained wide acceptance as a support tool in planning practices (Herbert and Chen 2015). They also have applications within architectural design, disaster risk management, and facility management (Solou and Dimopoulou 2016). In recent years, 3D city models have further become a significant component within the smart city and the digital twin paradigms (Roland et al. 2015).

3D city models provide a three-dimensional digital representation of urban areas at various levels of detail (LoDs), focusing on urban objects and structures, most prominently buildings (Zhu et al. 2009). LoDs regulate the amount of detail used to represent the virtual world, and its usage typically involves finding an appropriate trade-off between fidelity and speed of 3D graphics (Luebke et al. 2003). The implication is that more generalized geometry of buildings includes less data and, therefore, has shorter computation time (INSPIRE 2013). A building model developed by architects or building developers at a local level is likely to be more detailed in geometry if there is a focus on its design. However, for an urban planner or a municipality that deals with buildings at a larger scale, e.g., on a neighborhood or city-scale, a more generalized building model may be sufficient for some tasks (INSPIRE 2013).

Concerning building models, the CityGML LoD framework is generally referred to, which includes five consecutive and well-defined LoDs, ranging from LoD0 to LoD4 (see *Section 2.1.2.*) (INSPIRE 2013; Biljecki et al. 2016). The LoD required depends on the 3D object's supposed application. There might only be a need for extruded boxes (CityGML LoD1) to represent buildings for some tasks. In contrast, other applications call for the models to include simple roof structures (CityGML LoD2) or well-represented facades (CityGML LoD3) (Zlatanova et al. 2012).

Two-dimensional (2D) and two-and-a-half-dimensional (2.5D) data differs from 3D data in that 3D data can contain multiple points of different altitudes (z) with identical planar coordinates (xy). 2.5D data, in contrast, can only include one altitude value per planar coordinate pair. A typical 2.5D data is, e.g., a digital elevation model (DEM). With 2.5D data, a pseudo-3D simulation is possible, but it is not 'true' 3D (Ford 2004; Billen et al. 2014).

At the early stages of their development, 3D city models were predominantly graphical or geometrical, not providing well-defined semantic structures and descriptions (Kolbe 2009). Semantically enriched models typically require objects to be specifically defined based on their nature or what they represent. Roofs, walls, floor plans are described as such and are the parts

that together make up a building. In some cases, architectural features of greater detail such as stairs, windows, chimneys may also be featured. Moreover, the relation between objects, e.g., door to walls, walls to roof, is often expressed (Gröger and Plümer 2012).

Data for 3D city models is collected using various acquisition techniques, including photogrammetry, Light Detection and Ranging (LiDAR), or GPS surveying (Zhu et al. 2009). With recent technological advances in photogrammetry and LiDAR, the practice of automatically reconstructing 3D city models has become more accessible to practitioners across different fields (Stoter et al. 2020). Typically, they are built by combining several methods, such as 3D reconstruction and data integration, e.g., by merging photogrammetry or laser scanning data with Geographic Information Systems (GIS) data such as 2D building footprints (Biljecki et al. 2015; Julin et al. 2018).

Technological advancements have been rapid in the development of digital 3D city models with numerous different platforms and data formats available today. However, converting 3D city models between formats remains challenging, either because of incompatible semantics or from a geometric point of view (Stoter et al. 2020). From a geometric perspective, models could be of different scales and LoDs or include redundant or inconsistent information. Further, models may not be represented the same way between formats or have alignment issues (Billen et al. 2014). Thus, interoperability is an ongoing research topic, and several different open standards, common technical solutions, and policies have been introduced for the storage and exchange of 3D data (Julin et al. 2018).

Interoperability is the ability of different information systems to exchange data, information, or processing capabilities (Worboys and Duckham 2004). This definition of the term is supported by the International Organization for Standardization (ISO) (International Organization for Standardization 2015). Concerning GIS, interoperability is essential since users commonly need to integrate spatial data that are derived from various sources (Worboys and Duckham 2004). The usefulness of a 3D city model across different platforms depends on its interoperability. Thus, interoperability between different 3D formats is a prominent topic within the scope of 3D city model research (Julin et al. 2018), and there is a need for standardized practices in their realization if their accessibility is to be expanded.

With the standardization of 3D city models, more value is added to the practice of information sharing and semantic representation of urban structures (Zhu et al. 2009; Gröger and Plümer 2012). Standardization has also proved essential in ensuring consistency of the data's geometrical and semantic aspects (Stoter et al. 2020). CityGML is currently among the most prominent open-source formats used to model and represent 3D city models in a standardized manner (Liu et al. 2017). Furthermore, 3D national frameworks are widely being developed or are already established in some countries, e.g., in the Netherlands and Turkey, to move towards interoperability and data consistency on a nationwide scale. National standards for 3D

buildings or city models fulfill the need to obtain more uniform management of building information (Eriksson et al. 2020).

Several municipalities in Sweden have come a long way in building 3D replicas of their precincts and already utilize them in numerous tasks. Meanwhile, a Swedish national specification for 3D data, is in development, and preliminary 3D building modeling guidelines have been published, denoted as NS building. The specification's primary purpose is to facilitate information exchange between municipalities and different actors in the built environment and create a consistent workflow when creating 3D city models (Lantmäteriet 2019). The specification provides, for example, a proposed outline for the construction of building models and the digitization of urban planning processes. Eventually, municipalities in Sweden will have to adhere to the national specification in their 3D replicas' future development. Many of the municipalities work on their 3D platform within a closed-source environment, and interoperability with the national specification for building models remains unclear. Municipalities that use proprietary formats, such as Esri's File Geodatabase (FileGDB), for their 3D data may face challenges in conforming their existing building models to the standardized national framework.

The research covers the process of converting 3D data within Esri's environment to a building model in a format that is consistent with the national specification (NS building) and store the resulting output models within the initial geodatabase. The process is facilitated by a spatial Extract, Transform and Load (ETL) workflow and tested on buildings within the municipality of Karlskrona in southern Sweden, using Safe Software's Feature Manipulation Engine (FME).

This thesis aims to:

A. Develop and propose a method to transform 3D building models derived from an Esri environment to make them compatible with the Swedish national specification, NS building, and store the output models in an Esri geodatabase.
B. Apply the proposed method using buildings within the municipality of Karlskrona in southern Sweden as a case study.

2. Background

This section summarizes topics and research related to the case study, such as the continual standardization of 3D city models, the common formats used, and developments in national standards for 3D data, focusing on the Swedish NS Building. The application of spatial ETL in research and practice is also explained.

2.1. Common standards, formats, and platforms for 3D city models

There can be many incentives for data standardization. A major reason is that standardized data formats make the exchange and transition between applications more feasible by promoting interoperability (Hajji and Billen 2012). For example, the Open Geospatial Consortium's (OGC's) CityGML standard focuses on establishing a common definition for entities, attributes, and relations of 3D city models. This is significant in respect to sustainable maintenance of 3D infrastructure and promoting the reuse of data across different fields (Gröger et al. 2012). Further, one of the expected benefits to developing standardized building models, as stated in the Swedish NS, is that they help make better-educated decisions and create a more efficient basis for, e.g., digital planning processes such as building permit automation (Lantmäteriet 2021b).

Numerous software platforms enable 3D city modeling, popular ones being Esri's ArcGIS Pro and CityEngine, Autodesk's AutoCAD, Maya, and Revit, GRAPHISOFT's ARCHICAD, and Trimble's SketchUp, many of which define their own proprietary format for 3D data. GIS and CAD/BIM-based platforms target a broad audience. However, their 3D capabilities are more niche, catering to professionals, such as planners, architects, geo-information specialists, and surveyors, rather than the general public (Julin et al. 2018).

The data differs depending on its intended use or purpose. For example, 3D data generated within game engine applications are primarily focused on high-quality visualization abilities. Moreover, in contrast to GIS or CAD/BIM platforms, they often do not enable spatial referencing (Zlatanova et al. 2012; Julin et al. 2018). Proprietary 3D formats within GIS and CAD/BIM, i.e., those that are derived from a particular software platform, are in common use. In fact, with their vast user base and acceptance, these data formats have become so-called de facto standards. This describes, for example, Esri's FileFDB and Shapefile (SHP), SketchUp's COLLADA, and AutoCAD's DXF (Zlatanova et al. 2012).

There are also formats available that have been established as standards by international organizations, for example, VRML, X3D, Industry Foundation Classes (IFC), and CityGML (Zlatanova et al. 2012). KML, once Google's proprietary format, has also been adapted as a standard. The two most common standardized 3D formats, IFC and CityGML, generally cater to different industries. IFC namely targets the building information modeling (BIM) domain

and is developed as a reference model for building objects, while CityGML is rather intended for the 3D geographical information system (GIS) domain (El-Mekawy et al. 2012; Gröger et al. 2012).

2.1.1. Esri MultiPatch

Esri's line of software products are among the most widely used in the realm of commercial GIS. The way in which Esri handles 3D objects is by using boundary representation in the form of triangle strips, triangle fans, triangles, or rings (*Figure 1*). These geometry types are so-called patches, and their collection is called a MultiPatch. MultiPatch is an industry-standard data format and a geometry type developed by Esri in 1997 (Esri 2008). They can be stored and exchanged in either shapefiles (SHP), or FileGDB formats.

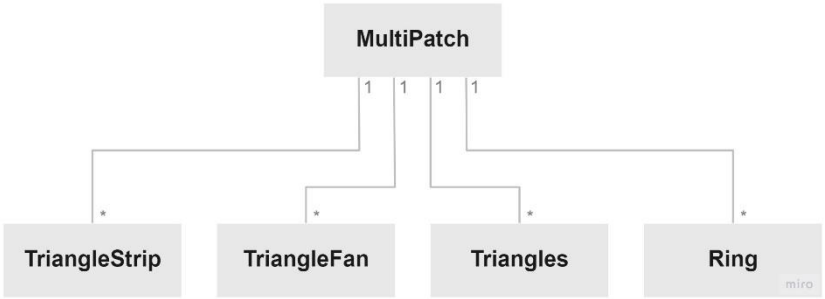


Figure 1: The structural relationship of MultiPatch. Figure is adapted from Esri (2008).

A single multipatch feature class object can be made up of these geometrical primitives to represent 3D surfaces. It can thus be considered as a container to these surface geometry types (Esri 2008). The MultiPatch UML-conceptual model (*Figure 1*) specifies that *TriangleStrip, TriangleFan, Triangles,* and *Ring* have a many-to-one associative relationship to MultiPatch. MultiPatches may be composed of one or more of these geometries or a combination of them. Esri's 2008 white paper on MultiPatches describes the geometries of MultiPatch in the following manner (*Table 1):*

Table 1: A description of MultiPatch's geometries (Esri 2008).

Geometry type	Description
esriGeometryMultiPatch	A collection of surface patches
esriGeometryRing	An area bounded by one closed path
esriGeometryTriangleStrip	A surface patch of triangles defined by three consecutive points
esriGeometryTriangleFan	A surface patch of triangles defined by the first point and two consecutive points
esriGeometryTriangles	A surface patch of triangles defined by non-overlapping sets of three consecutive points each

What separates triangles from triangle strips or fans is that there are no shared vertices between triangles, even if they touch one another. Triangle strips share a single edge with a neighboring

triangle, and fans share a central vertex for each connected triangle (*Figure 2*). This ultimately leads to fewer vertices needing to be stored and a smaller file size.

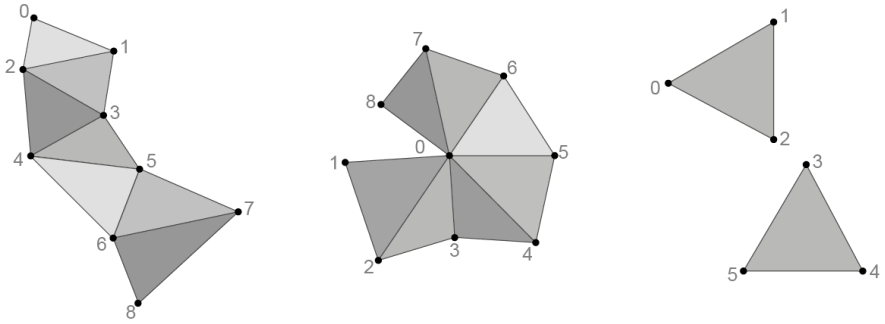


Figure 2: Multipatch geometries visualized. Figure is adapted from Esri (2008).

MultiPatches can represent simple objects such as spheres and cubes or complex ones like buildings and trees (Esri 2008). In an attribute table, 3D objects represented with MultiPatches have a single record per object, allowing for the storage of complex 3D structures composed of multiple geometric constants in a single attribute.

2.1.2. CityGML

CityGML was first developed as the general trend of 3D city models was moving more onto a semantic representation instead of the simpler geometric or graphic oriented representation, meaning that 3D objects and their properties are well-defined, with structures and topological relationships. In its development, it was considered to become a base information model for 3D city models (Jusuf et al. 2017).

The CityGML standard is an open data model and XML-based format developed by the OGC to store and exchange 3D city and landscape models. The standard is realized as an application schema of the Geography Markup Language version 3.1.1 (GML3) (Gröger et al. 2012). Schema is the skeletal structure of databases and is the outcome of data modeling activities (Jusuf et al. 2017). Each CityGML object can store information about semantic properties, geometry, topology, and appearance according to the standard. Topology refers to the relationships between geometries in the model. In contrast, appearance refers to the surface's observable properties, e.g., the ability of a 3D model to store external visual data such as realistic texture images (Kolbe 2009; Zlatanova et al. 2012).

Version 2.0 of CityGML was released in 2012 and remains the current version at the time of this writing. With this version, there are five different LoDs defined (*Figure 3*), with higher levels introducing more complex features. LoD 0 represents the building's footprint, while LoD

1 is a coarse block representing the extrusion of LoD 0 to the height of the building. The LoD 2 building model has a simple roof shape, and where parts of the building model can be semantically defined, e.g., as roof, wall. LoDs 3 and 4 are architecturally detailed, with LoD 4 including indoor features (Biljecki et al. 2016).

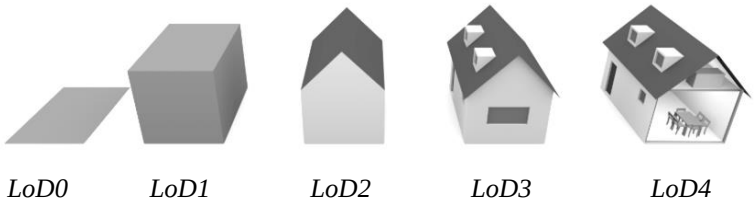


Figure 3: Different LoDs specified by CityGML 2. Figure is from Biljecki et al. (2016).

Version 3.0 of CityGML is well underway and introduces some improvements and additions to the standard, such as different definitions of the LoDs, including the inclusion of inside surfaces in all LoDs and removal of LoD4 (Kutzner et al. 2020). Moreover, its revision sets out to increase interoperability with other prevalent standards (Kutzner et al. 2020).

CityGML models can be enriched using Application Domain Extensions (ADEs), which provide the possibility of adding new feature classes and attributes while the semantic structures, geometry, appearance, and topological properties of CityGML remain intact (Zlatanova et al. 2012; Biljecki et al. 2018). This allows national standards to be compatible with an open standard format while also following their own specifications and requirements. ADEs are different from generic objects and attributes in CityGML in that ADEs are defined in an additional XML schema definition (XSD) file with its unique namespace. The XSD file then imports the XSD of the extended CityGML modules (Gröger et al. 2012).

CityGML has proved useful in some cases in the establishment of 3D standards in a national context via an ADE (Stoter et al. 2013; Gruber et al. 2014; Ates Aydar et al. 2016). This is how the Swedish national building model was initially planned to be implemented. Even with CityGML being a comprehensive standard, its current version cannot fulfill all requirements needed for the establishment of a national standard without an extension such as ADE (Eriksson et al. 2020).

CityGML has been used as a base in the realizations of several 3D national specifications. This may be reasoned from several standpoints. In a 2012 study, Zlatanova et al. did a comprehensive study comparing 3D standards, including de facto standards, based on several criteria. The comparison was brought forth in the process of establishing a national 3D standard for the Netherlands. The criteria that were included were for example the support for semantics, objects, attributes, georeferencing, and web-use, were all deemed important in establishing a common 3D spatial data infrastructure (SDI) using a generic standard. CityGML scored relatively well on all measures, compared to the other examined data formats, i.e., VRML, X3D, KML, COLLADA, IFC, GML3, DXF, SHP, and 3D PDF (Zlatanova et al. 2012).

2.2. National standards for 3D city models

The Netherlands is one of the countries that has extended its 2D standardization framework to include 3D geo-information. The establishment of the Dutch national standard for 3D geodata resulted from a pilot project initiated in 2010 (Stoter et al. 2013). This project aimed to define a common 3D approach for the whole country via a 3D spatial data infrastructure (SDI) (Stoter et al. 2011). The Dutch 3D national standard that has been realized aligns with OGC's CityGML standard and is formally established as a CityGML 2.0 ADE (Stoter et al. 2011; Stoter et al. 2013). Furthermore, the research conducted while forming the standard proposes a generic framework for extending CityGML for a specific context, in this case for national purposes. This process had not been well documented before in OGC's specifications (Van Den Brink et al. 2013).

Germany and Turkey followed a similar direction as the Netherlands, in terms of compatibility with the encoding standard of CityGML, to establish their 3D national geodata models in 2014 and 2016, respectively. Both national standards are prepared as an ADE, extending the existing thematic modules of CityGML according to the national needs (Gruber et al. 2014; Ates Aydar et al. 2016; Roschlaub and Batscheider 2018).

2.2.1. The Swedish national specification

The national specification for 3D data is published by Lantmäteriet, the Swedish mapping, cadastral, and land registration authority. It contains information on the construction of 3D building models and provides modeling guidelines for their establishment (Lantmäteriet 2021a). The modeling guidelines of NS building meet the requirements of ISO 19131 (Lantmäteriet 2021a), which is a standard regarding specification documents for geographic data products (International Organization for Standardization 2007).

NS building limits building models to the definition of the building's outer walls and excludes all indoor and apartment information, along with information on property assessment and construction. Moreover, it excludes non-building objects such as bridges and tunnels. According to the *Planning and Building act of 2010*, a building is defined as a permanent construction made up of a roof or roofs and walls designed for people to stay in it (Riksdagsförvaltningen 2010). NS building follows this definition as well.

The models of NS building should convey information regarding buildings' geometries, their declared purpose, area measurements, and height values (Lantmäteriet 2021a). Its conceptual model (*Figure 4*) describes that a building is composed of building parts. Building parts may have building installations. Building parts are represented with boundary surfaces; bottom, roof, outdoor roof, outdoor floor, wall, and closure surface (for objects that are not completely

closed by design), or a solid in the case of NS LoD 2.2 (Lantmäteriet 2021a). A solid is a volume-based object, such as buildings, represented by boundary surfaces (INSPIRE 2013).

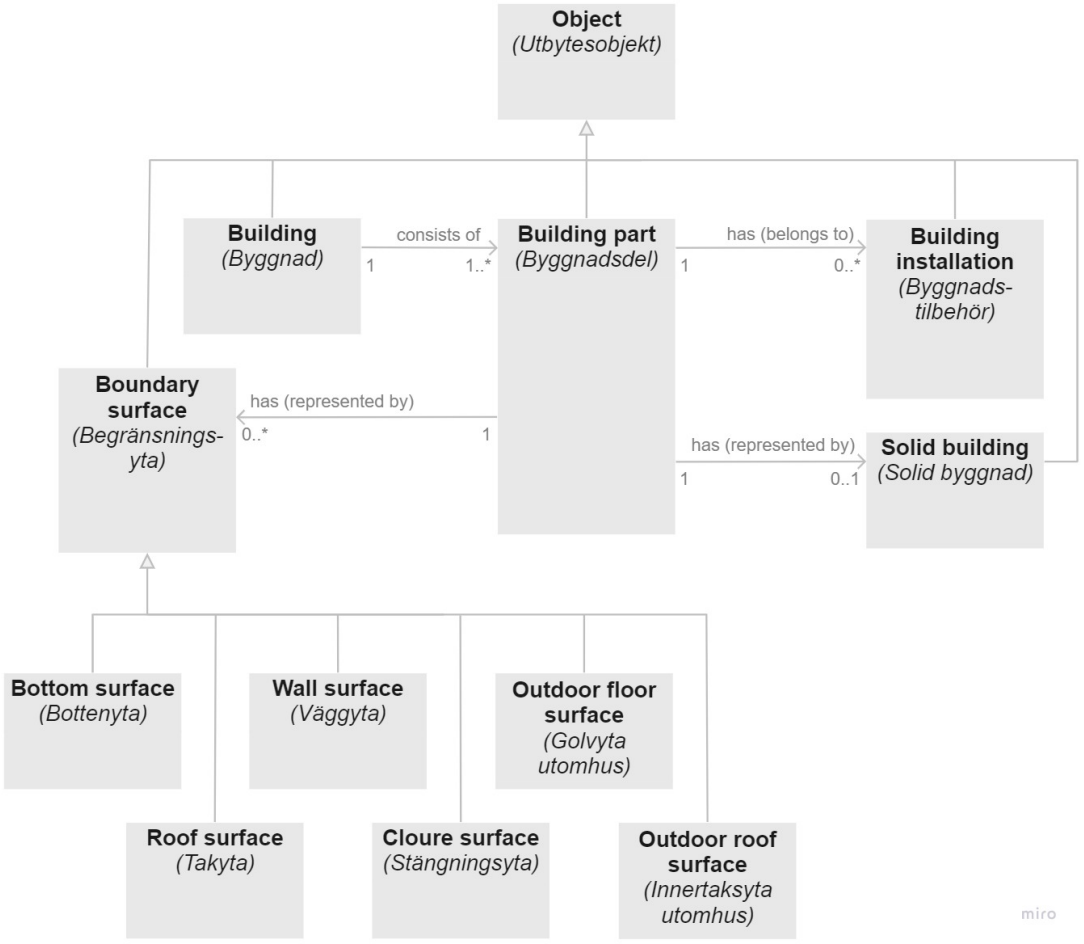


Figure 4: A simplified UML diagram of NS building. Figure is adapted and translated from Lantmäteriet (2021a).

The classes of NS building have various attributes associated with them, most of which belongs to building parts. *Table 2* exemplifies attributes that should belong to building parts. However, the attribute list is not exhaustive but simply enlists the most relevant attributes in the context of this thesis and can be generated using a building model's geometry.

Table 2: Attributes of NS building parts that are relevant to this thesis. Adapted from Lantmäteriet (2021b).

Name	Short description
absolut höjd tak	Absolute height value in reference system of a building object's roof parts (highest point).
absolut höjd botten	Absolute height value in reference system of a building object's bottom parts.
bruttoarea	Gross area, defined as the measurable parts of the floor plan, limited by the enclosing parts of the building such as walls, measured in m^2.
takvinkel	Roof pitch angle.
taktyp	Roof type.

Other attributes are, for example, quantifiable measures about the number of stories, stories above ground/underground, apartments within buildings, and various attributes depicting size measurements and the building's functions. The buildings of NS building can be declared as having any of the following primary (level 1) functions: residency (*s. bostad*), community function (*s. samhällsfunktion*), activities (*s. verksamhet*), industry (*s. industri*), agriculture (*s. lantbruk*), complementary building (*s. komplementbyggnad*) or other building (*s. övrig byggnad*). These building functions can then be expanded further, with function levels 2 and 3 (*Table 3*).

Table 3: Example of different levels of functions for NS building models (Lantmäteriet 2021b).

Function level 1	Function level 2	Function level 3
Community function	Culture	Library
Community function	Healthcare	Hospital
Residency	Small house	Townhouse

Several levels of detail (LoDs) are declared for the building models of NS building (*Figure 5*). The LoDs range from 0.1, 0.2, 2.1, and 2.2, each having three variations (a, b, c).

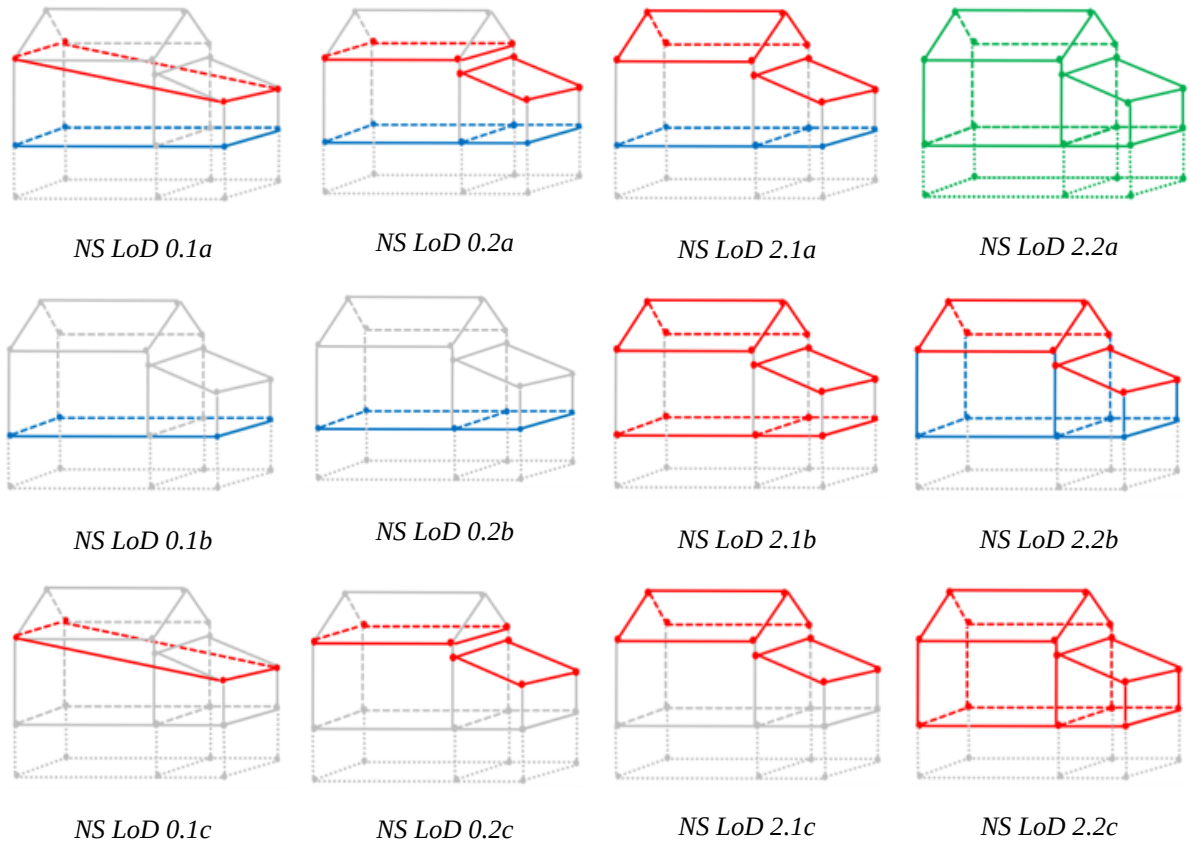


Figure 5: The different levels of details (LoDs) of the national specification's building models, version 1; test 1.1 Figure is from Lantmäteriet (2021b).

The colors in *Figure 5* refer to the measurement method used, i.e., red being surfaces measured with photogrammetry while blue represents surfaces derived from ground-based surveys. In the context of this thesis, these parts will be referred to as roofs and bottom surfaces for simplification, and output models will take on the same color scheme as displayed in the figure above to distinguish between the different parts.

2.3. Spatial ETL

Various translation efforts have been made using a process called spatial Extract, Transform, and Load (ETL). ETL processes establish the translation of data in a streamlined manner by extracting it from the source, applying the needed transformation on the data, and then loading it to the desired output (Sakr and Zomaya 2019). The translation takes place using repeatable workflows and various activities to facilitate the process. Spatial ETL applies this process to spatial data and is intended to overcome difficulties generally associated with traditional translation methods.

Safe Software's Feature Manipulation Engine (FME) was the first spatial ETL application established and is today an industry-leading spatial ETL tool (Safe Software n.d.-a). Within FME, the process of extracting, transforming, and loading is applied using a *reader*, a set of *transformers*, and a *writer* (Safe Software n.d.-a). A transformer is an FME's *Workbench* tool that can transform features from the source data to destination data and has the ability to read and/or write around 200 distinct formats. Over 500 different transformers facilitate data transformation within FME (Safe Software 2020), many of which can handle or are explicitly aimed towards 3D data.

Spatial ETL solutions in FME have been deemed successful in facilitating 3D building modeling, both in their creation from raw data and restructuring of existing models. They havebeen implemented in many studies regarding the transformation from one format to another. Selected research is presented below to exemplify different use cases and the advantages or challenges faced using spatial ETL.

Drešček et al. (2020) created a holistic spatial ETL approach within FME to generate 3D building models based on Unmanned Aerial Vehicle (UAV) data. They concluded that their efforts were effective, stating that their data process model allowed developers to easily control and adjust each process step.

In investigating integration possibilities between models derived from commercially available software (Esri CityEngine and Google's SketchUp) and CityGML, Dimopoulou et al. (2014) found FME to be the appropriate software to handle the transformation steps and to harmonize the initial 3D model with the CityGML standard. Kang and Hong (2015) proposed an ETL architecture that separates geometrical information of 3D models and their relevant properties

to move from a BIM to a GIS facility management structure. The results showed advantages of the ETL method in its reusability and data processing cost. In contrast, the disadvantages were in its limitations regarding flexibility and the extensibility of data schema. Hajji and Billen (2012) found that existing 3D models derived from different providers could be restructured and converted to CityGML. However, depending on their source, the output models were not all of the same quality despite being made interoperable. Further research may touch upon how to deal better with multi-source 3D data integration. They also conclude that restructuring 3D models is not a trivial process, especially regarding which transformer tool to use within FME, as there is a wide selection to choose from. In establishing the prerequisites for interoperating CityGML and IFC, El-Mekawy et al. (2008) conclude that challenges to the spatial ETL method for data exchange relate to semantic differences between systems and formats. Differences relate to issues regarding spatial data integration, including tabular data, e.g., schema matching problems and heterogeneity of sources, or spatial aspects, e.g., missing geometrical information. It was also observed that using the ETL method was a time-consuming process.

3. Methodology

The methodology section introduces the municipality of Karlskrona, the 3D city model received, and the selection of the study area for the case study. Then the tools used are described, followed by a general overview of the methodology applied in the case study. Finally, a detailed description of the conversion process from the initial dataset to an NS building model is given.

3.1. Study area

Karlskrona is the capital of Blekinge County in southern Sweden (*Figure 6*) (IntelligentCities n.d.)and one of Blekinge's five municipalities. Its architecture and town plan has gained the city an UNESCO World Heritage status, its well-preserved naval base dating from the 17th century being the main factor for the designation (VisitKarlskrona n.d.).

The municipality has taken steps to develop its robust digital platform, including the creation of a 3D city model. The model has the possibility of providing stakeholders and the public with information which the municipality may want to make accessible, such as noise levels, building permit applications, and new planning projects (Sjölin et al. 2019). Their strong digital presence got the municipality appointed as Sweden's second-best digitalization municipality in 2017. Moreover, it participated in The European Commission's Digital Cities Challenge in 2018 (IntelligentCities n.d.).

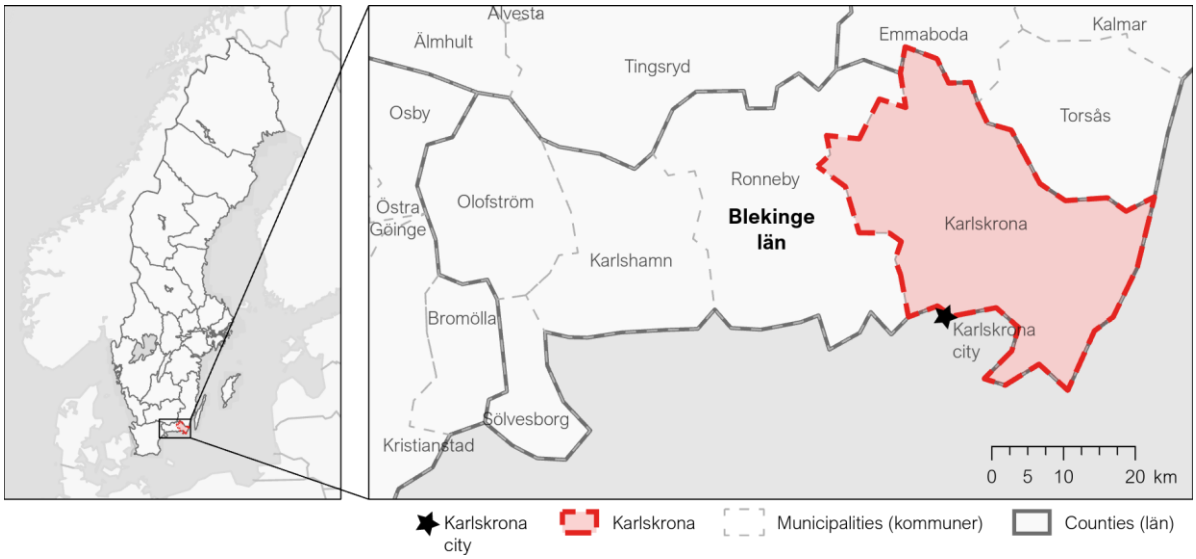


Figure 6: Karlskrona municipality in Sweden. Map data source: Lantmäteriet.

3.2. Data

A 3D city model was provided by the municipality of Karlskrona in Esri's FileGDB format. The 3D geodatabase included the entire city model that was constructed by the municipality.

The model was primarily created from 2D footprints of buildings and heights from Lantmäteriet's LiDAR data. The construction and maintenance of the 3D data was performed within an Esri environment using software such as ArcGIS Urban, CityEngine, and ArcGIS Pro to facilitate different steps of the process, making this dataset suitable to use as a case study. The FileGDB contained 3D objects stored as MultiPatch geometry types, representing buildings within the municipality. The building models were generally represented with a simple floor plan, facade, and roof, in a level of detail comparable to CityGML's LoD 2, as a coarse block with a simple roof structure (*Figure 8)*. Some buildings included more architectural elements and detail, namely important and noticeable buildings within the city, such as churches and administrative buildings. Those models were constructed within different software, i.e., SketchUp and Blender, before being imported to the city model in Esri's environment.

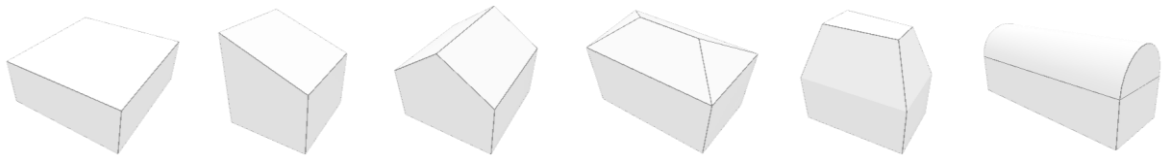
Table

trosso

	BuildingFID ▲	OBJECTID	BLDGHEIGHT	EAVEHEIGHT	ROOFFORM	BASEELEV
46	Building_62	30	2.96385	<null>	Flat	14.65552
47	Building_63	31	20	16	Gable	8.235957
48	Building_64	32	13.34362	8.943621	Gable	8.8002
49	Building_65	33	17.5	14.25	Gable	9.8049
50	Building_66	2153	10.59773	<null>	Flat	8.698219
51	Building_66	2152	10.59773	<null>	Flat	8.698219

Figure 7: Example attributes of the test data, viewed in FME. The dataset includes more attributes than are shown here.

The data contained various attributes to start with that included relevant information about the building models (*Figure 7*). Some attributes proved helpful at different steps of the transformation, such as the building's roof forms (*ROOFFORM*), identifiers of individual building objects/parts (*OBJECTID*), and IDs of buildings (*BuildingFID*), which may be composed of one or more building objects.



Flat *Shed* *Gable* *Hip* *Mansard* *Dome*

Figure 8: The different roof forms present in the test data.

The attributes depicting roof forms were obtained automatically by the municipality using Esri's software, along with information about the building's height (*BLDHEIGHT*), eave height (*EAVEHEIGHT*), and base elevation (*BASEELEV*). These attributes can all be derived from

Esri using the tool *Extract Roof Form*. The test data included the following roof forms: flat, shed, gable, hip, mansard, and dome (*Figure 8*).

A subset of the 3D city model received from Karlskrona was used for the case study. An area in the city center of Karlskrona on the island of Trossö was chosen (*Figure 9*), and the data within this area was extracted using ArcGIS Pro before being loaded onto FME. This was preferred to improve the processing speed of the tested transformation.

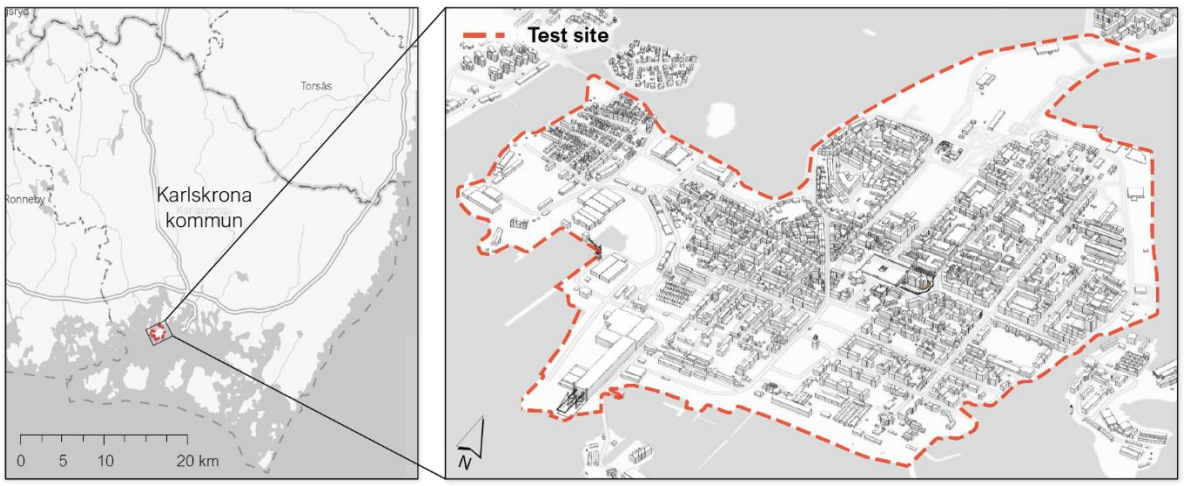


Figure 9: Data within the island of Trossö in the city of Karlskrona was extracted for the case study. Map data source: Lantmäteriet.

The selection included a wide variety of building types that could be used to account for alternative workflow scenarios caused by, e.g., different geometry types, roof forms, or missing attributes. For example, buildings composed of meshes were interpreted differently in FME and had to be subjected to a different workflow than those building models made up of individual faces in a multi-surface aggregate. This is expanded on in the methods. The test data contained 2320 individual building parts/objects, which each have a unique identifier, called *OBJECTID*. The test data included both buildings that had been created solely within Esri's software using building information and a few buildings constructed using different software and imported to the database. The workflow presented attempts to accommodate all building models within the test-site, both Esri derived and those sourced elsewhere.

3.3. Context of thesis work

Municipalities in Sweden are responsible for collecting building data locally and delivering it to the national geodata platform/geoportal, where consumers may retrieve the building models. This can either be done by directly delivering data to the geoportal or providing access to their local storage via the geoportal. The data made available should be in accordance with the NS building standard before being delivered to the user. Karlskrona and other municipalities have already started working on their 3D city models. Those that use Esri's software work with the

MultiPatch format for their 3D buildings, which would need to be processed to be compatible with the national specification before being retrieved by the consumer (*Figure 10*).

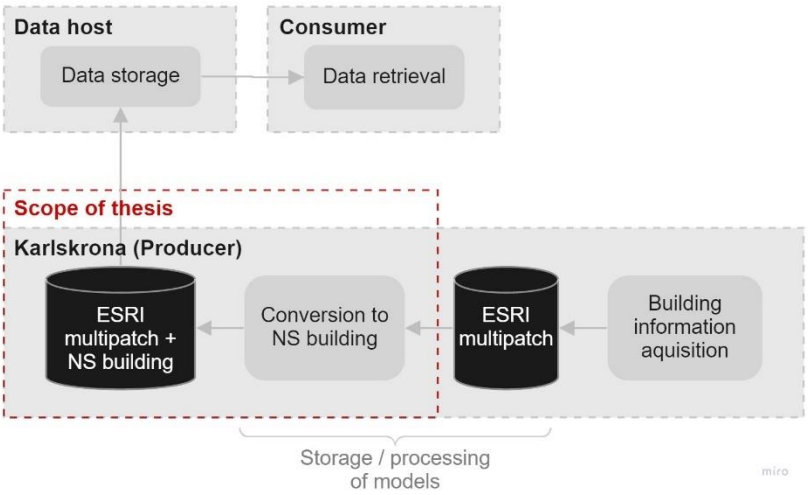


Figure 10: Scope of thesis work in the larger context of 3D building data collection, storage, and exchange between different stakeholders. Figure is adapted and translated from Lantmäteriet (2021a). See original information flow for NS building in Appendix A.

As presented in the information flow of NS building (Figure 10), producers are the public or private entities that collect and decide on the deliverance of building information to consumers. They are, e.g., the municipalities, and in the context of this thesis, the municipality of Karlskrona. Municipalities in Sweden are ultimately responsible for collecting data, creating NS building comparable models, and storing them.

3.4. Tools

FME by Safe Software was used in this research to perform the transformation of the data to make it compatible with NS building as proposed in *version 1, test 1.1* of the modeling guidelines of the national specification, published by Lantmäteriet. FME supports numerous data formats both as a *reader* and *writer*, including Esri's FileGDB and Shapefiles, IFC, and CityGML, and has various capabilities and built-in tools to facilitate the transformation process (Safe Software n.d.-a). The term *reader* in FME's context refers to the source data that is read into the program initially, and the *writer* is the destination data, i.e., the resulting output models.

The workflow described in the following sections includes the usage of FME tools, denoted as *transformers*. Transformers that proved essential to the conversion were, for example, *Aggregator, BoundsExtractor, GeometryPartExtractor, GeometryPropertyExtractor, GeometryValidator, Orientor, SurfaceNormalExtractor,* and *Tester*. The functionality and implementation of these tools is explained along with the workflow in subsequent sections.

3.5. FME's interpretation of MultiPatch

FME supports the extraction of FileGDB data formats and MultiPatch's geometry types. However, once the data is read in FME, MultiPatch will be interpreted as FMEs own geometry type, for example, as *IFMEMultiSurface*. Geometry structures within FME are hierarchical. Each MultiSurface, which represents a single building object, may be composed of multiple face parts (*IFMEFace*). Faces in FME can be defined as 3D planar areas containing either a raster, polygon, or a donut. Generally, each face part is composed of a single polygon (*IFMEPolygon*), which is made up of a line (*IFMELine*), having four coordinates if the underlying geometry is a triangle, as opposed to when the geometry type is a triangle fan or strip. An example of this composition can be seen in *Figure 11* below, which shows FMEs interpretation of the structure of a simple building object composed of triangles. Each triangle has four *xyz* coordinates, with the first and last being identical, thus representing a closed plane in a three-dimensional space.

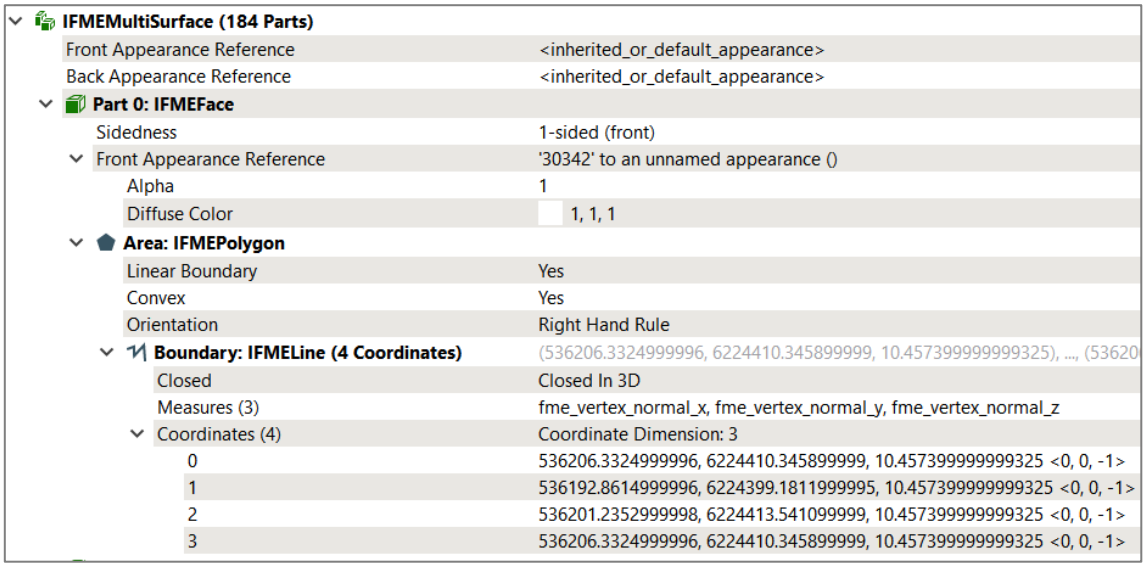


Figure 11: A single building object viewed in FMEs Feature Information window. This figure only exemplifies a single face part of 184 in total.

In Karlskrona's dataset, some building models were different from what is exemplified above. In addition to being composed of individual triangles/faces, some were also composed entirely of triangle meshes (*IFMEMesh*), i.e., triangle strips or fans. Of the 2320 extracted building models, 111 (5%) were made up of meshes.

For building models consisting of triangles, the proposed workflow for the transformation of the geometries was relatively straightforward, but additional steps needed to be taken to facilitate the transition of all 3D buildings. The workflow for building models made up of both geometry types is mentioned in the following sections.

3.6. ETL process

The first step of the transformation was to import the subset data as a reader in a FileGDB format into FME to be processed and worked on. Following the import, the transformation steps were implemented to make the building models compatible with NS building. The transformation process centered around making the data consistent with the geometry of NS building while also providing possible attributes that are required of the NS building models. After the transformation process, output models for each of the required LoDs were added as a feature class within the initial geodatabase (*Figure 12*).

Figure 12: The ETL process. The initial database contains building models in Esri MultiPatch format, which is worked on and transformed within FME. Building models are transformed so that they are according to NS building. Output models are stored in the initial database.

3.7. Conversion

Throughout the next chapters, each step of the conversion is explained alongside simple figures of the FME workflow. Note that workflow figures are generally placed following the overview of each of the steps that they portray.

3.7.1. Attribute retrieval, de-aggregation of surfaces & computation of measures

The steps taken in this section can be summarized as follows (*Figure 13*): Attributes for roof types (*taktyp*) and absolute heights for each object's bottom and roof parts (absHojdBotten and absHojdTak) were created. IFMEFace geometries were then extracted from the building models to de-aggregate them, and the same was done for Mesh Parts, and their vertex normals recalculated. The next step was to extract the surface normals and store them as attributes.

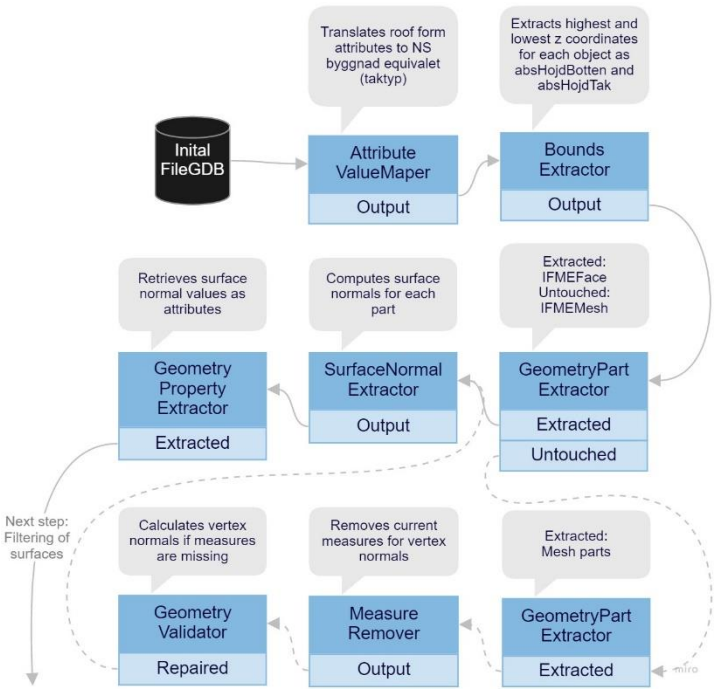


Figure 13: Workflow steps for sections 3.7.1.1. to 3.7.1.5.

3.7.1.1. Attribute: taktyp

The NS building attribute *taktyp*, or roof type, could be acquired by pointing the Esri-derived roof forms to equivalent NS building roof type values. This was done using *AttributeValueMapper*, where the translated roof form (destination value) was pointed to an equivalent source value of the NS building roof type attributes. Subsequently, the tool created a new column in the attribute table, including the matched destination values (*takyp*) as records. *Table 4* shows the matched attributes. NS building roof types were considerably more than the test dataset included, as displayed below. The destination values with empty source values were not mapped.

Table 4: Translation of roof form attributes to the relevant Swedish records that conform to NS building.

Source value	Destination value
Flat	lågt lutande tak
Shed	pulpettak
Gable	sadeltak
Hip	valmat tak
Mansard	mansardtak
Dome	kupoltak
-	halvvalmat tak
-	kägeltak
-	pyramidtak
-	bågtak
-	penthustak
-	sågtandtak
-	tälttak
-	dubbelkrökt tak

3.7.1.2. Attributes: absolut höjd botten & absolut höjd tak

Deriving the absolute heights of building objects is done using *BoundsExtractor*, which retrieves the minimum and maximum values of a feature's coordinates as new attributes (Safe Software 2020). With the tool, you can choose which coordinates to extract: the minimum or maximum of a feature's x, y, or z coordinates. The *z*-min and *z*-max coordinates were retrieved for each building object. They are representative of the absolute height of the building's floor plans and the roof height. *Z*-min values were denoted as *absHojdBotten* and *z*-max values as *absHojdTak.*

3.7.1.3. De-aggregating surfaces

The test data comprises two different composite geometries used to represent solid 3D building models. They are either made up of triangle composites or composed of meshes. The 3D objects needed to be de-aggregated, i.e., split up, into their geometrical parts. Buildings consisting of meshes had to be subjected to a different workflow, as opposed to the models composed of triangles, to be fully de-aggregated.

The extraction of geometries was performed using FME's *GeometryPartExtractor*. *GeometryPartExtractor* extracts or removes specific geometry parts using a test clause (Safe Software 2020). The test clause that was used (*Table 5*) separated faces (*IFMEFace*), which are made up of triangles, from their composite geometry (*IFMEMultiSurface*) and made it possible to work with each individual geometry part (*Figure 14*).

Table 5: Tool parameters of GeometryPartExtractor.

Part to test	Left value	Operation	Right value
This part	Geometry type	=	IFMEFace

Using *GeometryPartExtractor* also separated IFMEFace parts from those made up of meshes (*IFMEMesh*) since only the triangled surfaces were extracted, while the meshes remained untouched. The untouched geometry parts (meshes) were automatically filtered from the extracted ones using the extractor, making it possible to work on them separately without adding additional steps.

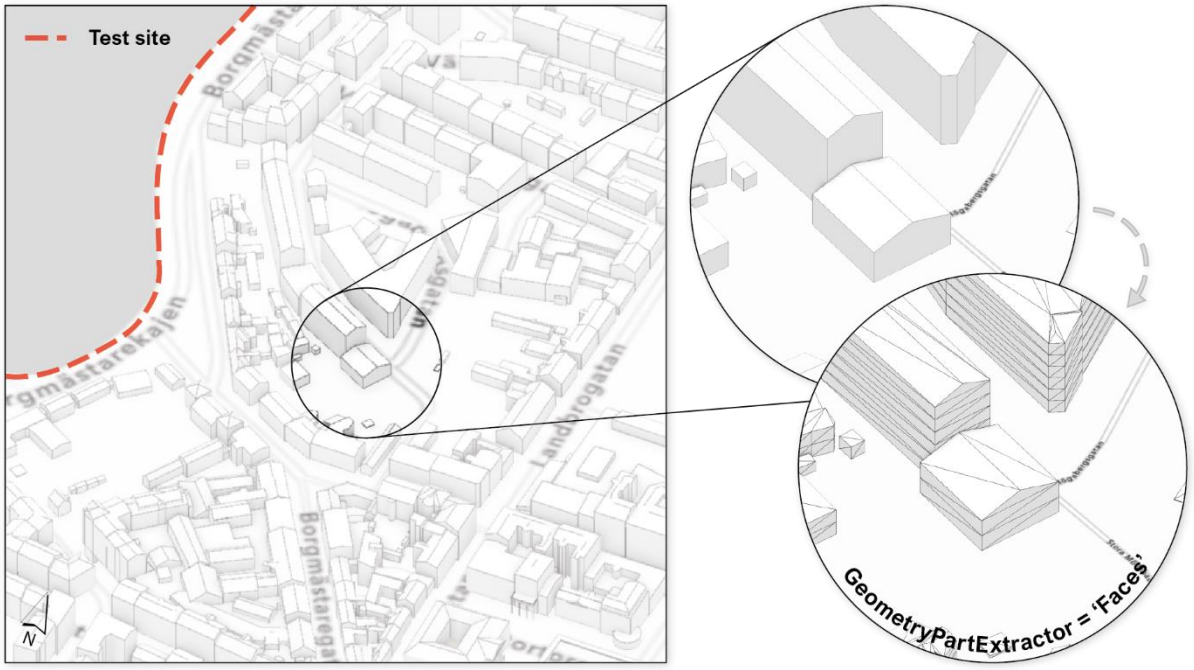


Figure 14: The process of extracting faces from multi-surfaces was applied on the test data, resulting in separate records per each face part of the building models.

GeometryPartExtractor was utilized again for extracting mesh parts to de-aggregate the meshes. The same test clause as is shown above was used, except using 'Mesh Part' instead of 'IFMEFace' as the right value, resulting in the extraction of individual faces/mesh parts for each mesh.

3.7.1.4. Vertex normals

The test data (explicitly the buildings composed of meshes) contained measures for vertex normals that were simply a copy of the vertex pool, i.e., the coordinates for each geometry part. This was not as it should be and would result in an incorrect calculation of surface normals within FME, an important measure for the transformation.

Vertex normals are the average of surface normal vectors of adjacent surface(s) to a vertex and are thus directional components associated with vertices of faces (Jin et al. 2005). FME's vertex normal algorithm normalizes these measures to range between -1 and 1 for each component (*xyz*) of the vertex normal (Safe Software n.d.-b). These vectors are not automatically computed if they come predetermined. Thus, for the vertex normals to be re-established correctly for the mesh parts, the predetermined measures needed to be removed first using *MeasureRemover* and calculated anew using *GeometryValidator*, to check for missing vertex normals and attempt a repair. *GeometryValidator* detects select issues in features and optionally attempts a repair if they are detected (Safe Software 2020).

The recalculated vertex normals contributed to the correct surface normals, which was the next step in the transformation. It also affected the appearance of the building models, as seen within FME's Inspector. Having wrong vertex normals affect the appearance of 3D objects, as they are often used to determine their shading when rendered (Safe Software n.d.-b). Before correcting these measures, the buildings with the wrong vertex normals could be easily distinguishable from the models that had the correct measures. The models composed of meshes could consequently be rendered correctly within FME (*Figure 15*).

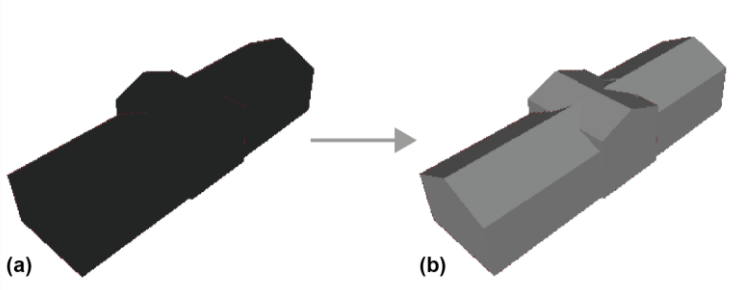


Figure 15: Re-establishing vertex normals for mesh parts resulted in building models composed of meshes (a) to have their appearance rendered correctly (b).

3.7.1.5. Surface normals

Once all buildings had been de-aggregated, the next step was to calculate the surface normals. The separation of the roof, bottom, and wall surfaces was performed, to an extent, using the surface normal of each geometry part of the building models. A surface normal is a normal vector perpendicular to a surface. Conventionally, a normal to a triangle surface is calculated by taking the vector cross product of two edges of the triangle. The order of vertices affects the normal's direction, i.e., whether it is inward-pointing or outward-pointing (OpenGL 2013).

Surface normals, given in values of *xyz*, indicate a surface's angle and orientation. They can thus be used to separate the parts of the building models based on if they are parts of a roof, wall, or floor plan if an assumption is made that these parts have a certain angle and orientation.

FME's tool *SurfaceNormalExtractor* was used to calculate surface normals. It computes this using the average of a face's existing vertex normals, and the *xyz* values extracted are automatically normalized by the tool to range between -1 and 1. The surface normal *z* value is valuable in determining how the faces are angled and their orientation. *Table 6* explains what these values depict.

Table 6: Surface normal z-values are indicative of the angle and orientation of a surface. Surfaces with z-values ranging between 1 and 0, or 0 and -1, are at an angle.

Z-value	Surface	Orientation
1	Horizontal	Outward
0	Vertical	-
-1	Horizontal	Inward

GeometryPropertyExtractor was used subsequently to create attributes for the extracted surface normals. *GeometryPropertyExtractor* can extract either geometry names or traits as new attributes (Safe Software 2020). The traits extracted were *surface_normal_x*, *surface_normal_y*, and *surface_normal_z*. Filtering the faces using surface normals is the next step of the transformation, and having these measures stored as attributes makes that easier.

3.7.2. Separation of surfaces

Following the de-aggregation of the surfaces, it was possible to estimate which geometry part belongs to a bottom surface, wall surface, and floor surface and aggregate those parts anew for distinction. The tool *Aggregator* can create homogenous collections by combining feature geometries with the option of aggregating them based on a shared attribute (Safe Software 2020). With each building model having a unique ID (*OBJECTID*), it was possible to group the geometry parts that share that identifier and aggregate them anew. *Figure 16* shows the workflow established in the separation of the surfaces for this purpose.

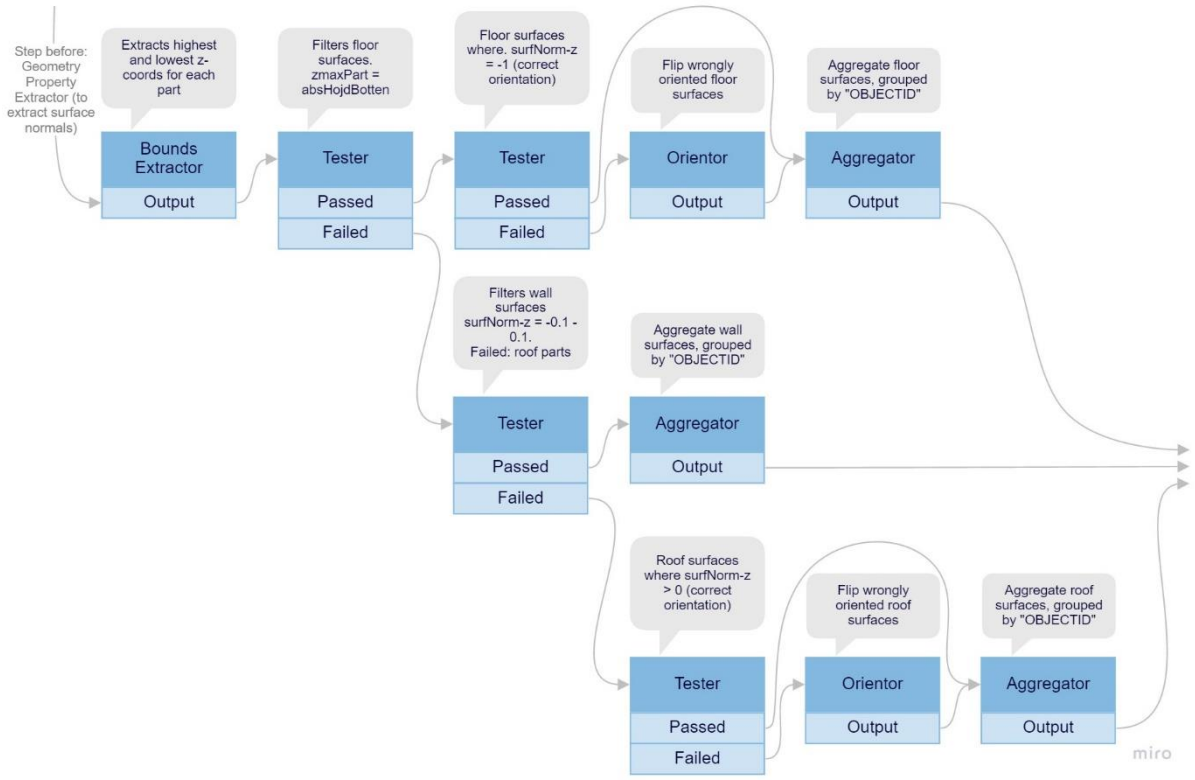


Figure 16: Separation of geometry parts (floor plans, walls, and roofs) based on elevation and surface normals.

To filter the faces, the surface normal *z*-values were used to a large extent. However, floors could be identified using elevation values of the individual buildings and their de-aggregated face parts. If a maximum elevation of a face part is equal to its corresponding building's minimum height, it was assumed that it belonged to a floor plan. This was preferred over using the surface normals, where $z = -1$, to deduct the floors because the test data displayed some orientation issues of some individual surfaces. This was observed in the externally sourced data, i.e., the meshed building objects. Ideally, the faces of the buildings should have an inward orientation (an inward-pointing normal) where they are enclosed within the buildings and vice versa. That is how Esri-derived building objects are modeled. If the orientation of the faces had been consistent, floors and flat roofs could easily be distinguished by using the surface normals *z*-values where -1 is equal to floors and 1 for flat roofs. Orientation inconsistencies created the problem that faces with surface normal *z*-values of -1 and 1 could both be floor plans, even though in most cases, i.e., for all Esri derived building models, the surface normal was as it should be. Using elevation values solely to deduct the floor plans from other faces eliminated that discrepancy error.

To separate the geometry parts according to the abovementioned queries, the tool *Tester* was used, which evaluates tests on features based on a given criterion and routes them according to the outcome, i.e., whether they pass or fail the criterion. (Safe Software 2020). An attribute was added to the isolated bottom, roof, and wall surfaces called *begränsningsyta*, to indicate what the surfaces represent; bottom (*bottenyta*), roof (*takyta*), or wall surface (*väggyta*).

With floor surfaces isolated from other parts, it was possible to correct the orientation errors that some of the data had displayed. This was done to establish better integrity for the externally sourced building models and for them to become more integrated with the Esri-derived models. *Tester* was used to filter wrongly oriented floor surfaces, i.e., those with positive surface normal *z*-values, and *Orientor* was used to flip them. *Orientor* adjusts the orientation of polygonal features (*Figure 17*), or in the case of linear features, their direction (Safe Software 2020).

Figure 17: The Orientor transformer in FME changes the orientation of a surface by changing the order of the vertices of polygonal features. Here the back is represented with a checkered pattern for distinction. If these faces were lying horizontally (surface normal), the outward-facing triangle would have a surface normal z-value of 1, while the inward-facing would have -1.

Surface normal *z*-values from -0.1 to 0.1 were used to separate wall parts from remaining geometry parts, i.e., all parts except those belonging to the bottom surfaces. Walls should have the *z*-value of 0 if they are truly vertical but including the range of -0.1 to 0.1 allows for a slight tilt. This range worked in identifying the wall parts present in the test data while simultaneously not capturing roof surfaces wrongly. A few of the externally sourced building models did have highly detailed roofs that contained vertical surfaces, which would be defined as a wall. This was a minor occurrence and not apparent with Esri derived building models

Having deducted the wall parts, the roofs should only remain (*Tester: Failed*). A workflow to orient all roof parts in the same direction was established the same way as was done with the floor plans, except all faces with negative surface normal *z*-values were filtered and then flipped. In aggregating the roof parts based on *OBJECTID*, a list was created of their surface normal *z*-values, which was important for the next step.

3.7.3. Roof surfaces

The steps pertaining to the aggregated roof surfaces and their attributes are explained in this section. This includes calculating the roof pitch angle (*takvinkel*), finding the absolute eave height, and creating roof surfaces for NS building LoD 0.2.

3.7.3.1. Attribute: takvinkel

Takvinkel is the pitch angle of the roof, i.e., the slope between the roof and the horizontal plane where the roof meets the wall. The pitch angle could be calculated for the aggregated roof surfaces in several steps (*Figure 18*), using the normal *z*-values derived before.

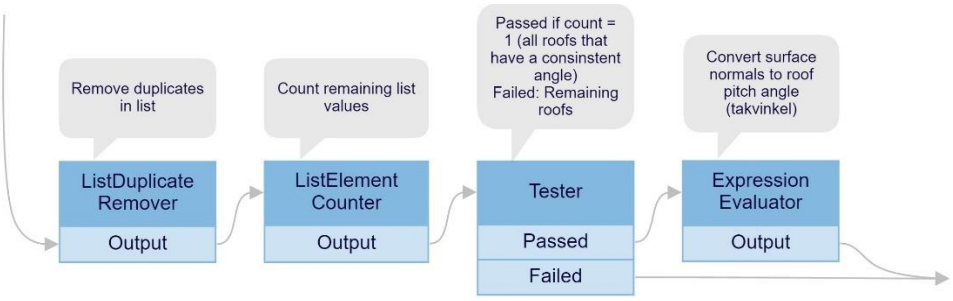


Figure 18: The steps taken to calculate the roof's pitch angle.

As mentioned, a list containing the *z*-values was created, so there's a single *z*-value per each geometry part that the roof consists of. To retrieve an angle that represents the whole roof, it should only have a single consistent angle throughout all of its parts, thus eliminating overly detailed roofs or those that display any sort of nuance in their form. To sort out roofs that have the same *z*-value for all their parts, *ListDuplacateRemover* was used, which removes duplicates of list attributes within an object. The remaining list values were subsequently counted using *ListElementCounter*, and the roofs with a single *z*-value remaining were filtered using *Tester* before performing the calculation.

It was possible to derive the pitch angle for the filtered roofs based on their surface normal *z* (*Figure 19*). Within FME, the normal of the surface is normalized so that $x^2+y^2+z^2=1$. The *z*-value represents the vertical line from the end of the normal to where the horizontal *xy*-plane meets the normal's base. Converting the surface normal *z*-values to the pitch angle was therefore performed using a basic trigonometric function; by calculating the inverse cosine of *z* (*Equation 1*).

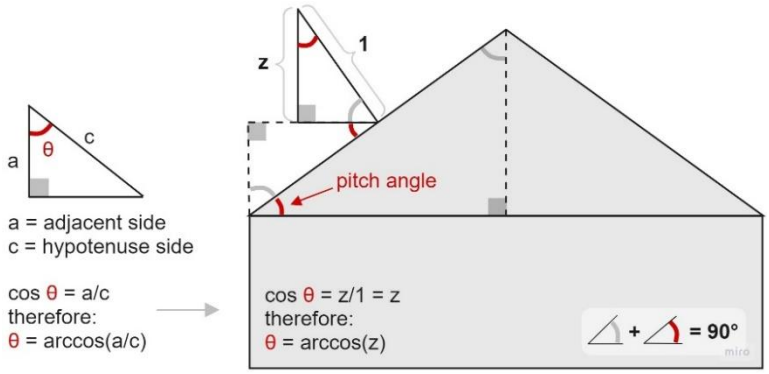


Figure 19: The pitch angle (takvinkel) was derived by finding the inverse cosine of the roof's normal z-value and then converting the resulting radians to degrees. The number one represents the normal of the roof's surface, while z is its z-value.

In FME, the inverse cosine function (*acos*) delivers the angle in radians, but the value of *takvinkel* should be in degrees. The conversion was made by multiplying the radians with $180°/\pi$, which is the degree equivalent to a single radian. The combined expression within *ExpressionEvaluator* to derive the pitch angle and convert to degrees was:

$$@acos(@abs(@Value(surface\_normal\_z))) * 180/@pi() \qquad \text{Equation 1}$$

The surface normal *z*-values needed to be expressed in absolutes because the attribute table still contained negative records where the orientation was initially wrong.

3.7.3.2. Computing absolute eave height

Absolute eave height is the height value, in the reference system, where roof and walls of buildings intersect. In contrast, relative eave height is the distance between base elevation and the eave of the building. This measure is needed for raising roof levels for NS LoD 0.2.

Relative eave height (*EAVEHEIGHT*) was already an attribute for some of the 3D objects on the test dataset, along with base elevation (*BASEELEV*). Using *ExpressionEvaluator*, the attribute *absEAVE* was created with the sum of eave height and base elevation for features where those values pre-existed. For the buildings (roof parts) that did not have these values, *absEAVE* was based on the minimum *z*-coordinates of roofs. The value was extracted from roofs using *BoundsExtractor*, and stored as *absEAVE* (*Figure 20*).

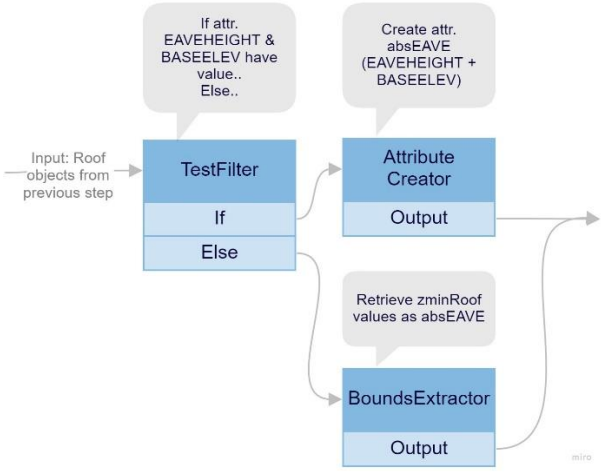


Figure 20: The steps taken to compute absolute eave height (absEAVE).

3.7.3.3. Roof parts for NS LoD 0.2

NS building LoD 0.2 roofs are represented with a flat surface where wall and roof intersect, on the eave of the building, while also preserving already planar roof forms, such as those of the type "shed", i.e., where the roofs are sloped towards a taller wall on one side. To create the roof parts of NS LoD 0.2, the roofs from the earlier steps were processed further (*Figure 21).*

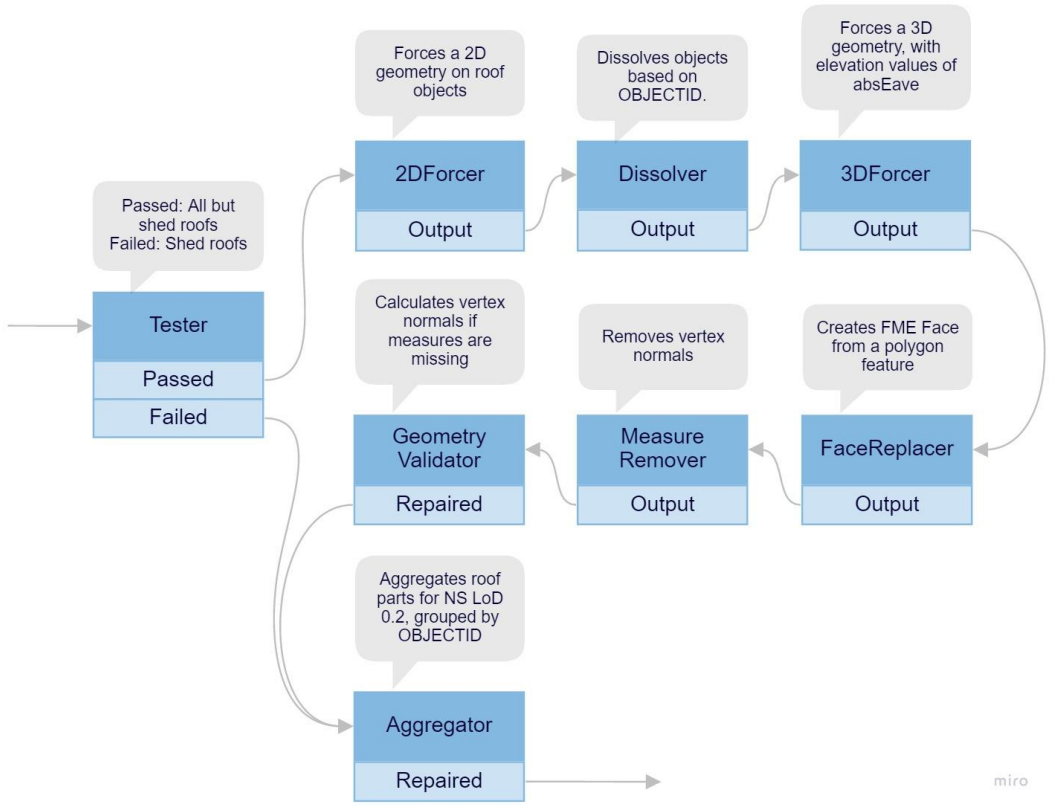


Figure 21: Steps taken to construct flat rooftops for NS LoD 0.2 building models.

A *Tester* was used to separate shed roofs from the others since only other roof forms needed to be processed. *2DForcer* was utilized to flatten the rooftops, which were subsequently dissolved based on their *OBJECTID*, to simplify the geometry of the resulting polygons. Then they were forced to have three dimensions anew and raised to the level of absolute eave (*absEAVE*) using *3DForcer*. *FaceReplacer* was then used to coerce a face geometry on the polygon feature, returning a surface compatible with the pre-existing ones.

The flattened rooftops still had vertex normals that assumed their previous structure and would therefore be rendered as if they were non-planar. Thus, the same method used in an earlier section was applied to the new roof surfaces, removing vertex normals and then using *GeometryValidator* to reapply them (*Figure 22*). The flattened surfaces were then joined with the prior isolated shed roofs and aggregated based on their *OBJECTID*.

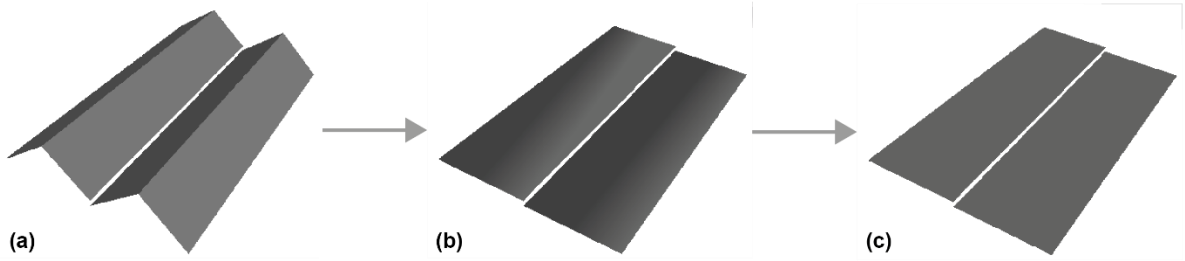


Figure 22: Roofs (a) were flattened (b) and their vertex normals reapplied (c).

3.7.4. Final steps

The remaining workflow before the data could be assembled into the relevant *writers* to store the output models is explained in this section (*Figure 23*).

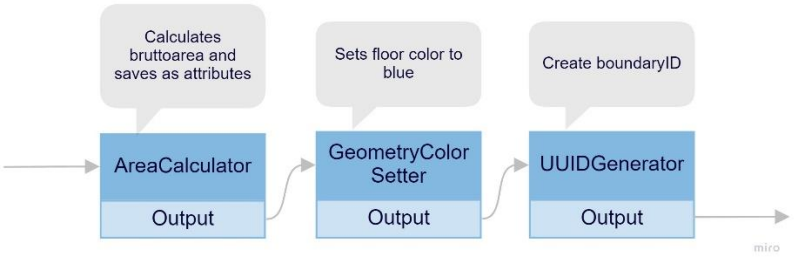


Figure 23: Final steps before the creation of NS LoDs. The attribute boundaryID was also applied to roof and wall surfaces.

The steps pertaining only to bottom surfaces were the calculation of the attribute *bruttoarea* and aggregating NS LoD 0.1 specific bottom parts, based on their *BuildingFID.*

Calculating the area (m^2) of the aggregated bottom surfaces was needed to create the attribute of *bruttoarea*. *Bruttoarea* depicts the area of the building object's floor plan. To calculate *bruttoarea*, *AreaCalculator* was used on the isolated floor parts. The tool calculates areas of polygonal objects in square meters and stores the results as new attributes (Safe Software 2020).

The wall surfaces did not need any additional steps. However, the aggregated surfaces (roofs, bottom, and walls) were assigned a unique identifier (*boundaryID*), using *UUIDGenerator*, which creates a universally unique identifier for features and stores them as attributes. The surfaces were color-coded for their visual distinction. Bottom surfaces were set to a blue color, whilst roof surfaces to red. This was done using *GeometryColorSetter* and was only for visualization purposes.

3.8. NS building models

The creation of NS building models of different LoDs within FME was a matter of assembling the geometries and attributes created in the transformation (*Figure 24*). This was different for each LoD. The schema of NS building models could be modeled to be Esri compatible using attributes, namely primary keys (PK) and foreign keys (FK). These keys, or attributes, exist to eliminate redundancy in records and to relate information between features or attribute tables based on a common identifier.

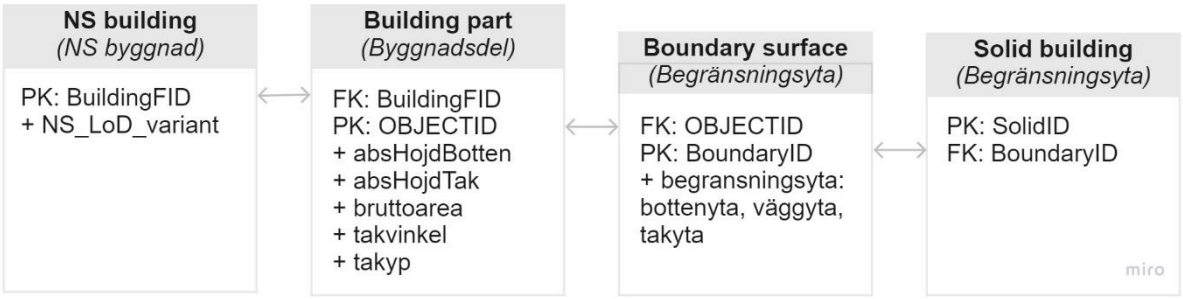


Figure 24: NS building classes, with corresponding attributes.

The isolated surfaces belonging to the bottom, roof, and wall parts were used to create the different LoD variants according to the specification. The surfaces that served as inputs for the construction of the final NS building models of different LoDs can be described as:

- NS LoD 0.1: includes bottom surfaces aggregated based on *BuildingFID*. It should also include roof parts, but they could not be modeled in an automatic manner in a way that would fit all buildings. *Section 5.3.* discusses the reasons behind this.
- NS LoD 0.2: includes the restructured roof surfaces (from S*ection 3.7.3.3.*), along with bottom surfaces. Aggregated (separately) based on their *OBJECTID*.
- NS LoD 2.1: pairs together the roof and bottom surfaces of the building object.
- NS LoD 2.2a: is a solid building and should thus include the surfaces needed to create an enclosed building object, consisting of roof, bottom, and wall surfaces.

The boundary surfaces served as a basis for the different LoDs and are representative of the building parts. To preserve the surfaces as distinct from one another, each of the building parts (identifiable with *OBJECTID*) has separate records per each aggregated surface (bottom, roof, or wall) relevant to the different LoDs. The output models thus have the following attributes: *OBJECTID*, *BoundaryID*, and *begransningsyta*, which specifies if the surface belongs to a bottom, roof, or wall. A separate attribute table stores the other attributes, which can be joined to the attribute table of the boundary surfaces based on a common identifier. Only the attributes of the building parts (*Byggnadsdel*) were stored in an external table, so the common identifier between boundary surfaces and building parts is *OBJECTID*. This was done to reduce redundant information, as each building part would be representative of one up to three surfaces per object, which all share the same attributes. Solid buildings share an identifier with the boundary surfaces, which can be used to relate to the attribute table of the building parts.

4. Results

The workflow established a process streamlining the conversion of a 3D city model, primarily consisting of building models derived in Esri and stored in a FileGDB in Esri MultiPatch format, to an NS building compatible model. The FME script arrived at to facilitate the process managed to capture all purely Esri-derived models and model them according to the Swedish standard, with separated surfaces of the bottom, roof, and wall parts. The broad method applied can be observed in *Figure 25* below.

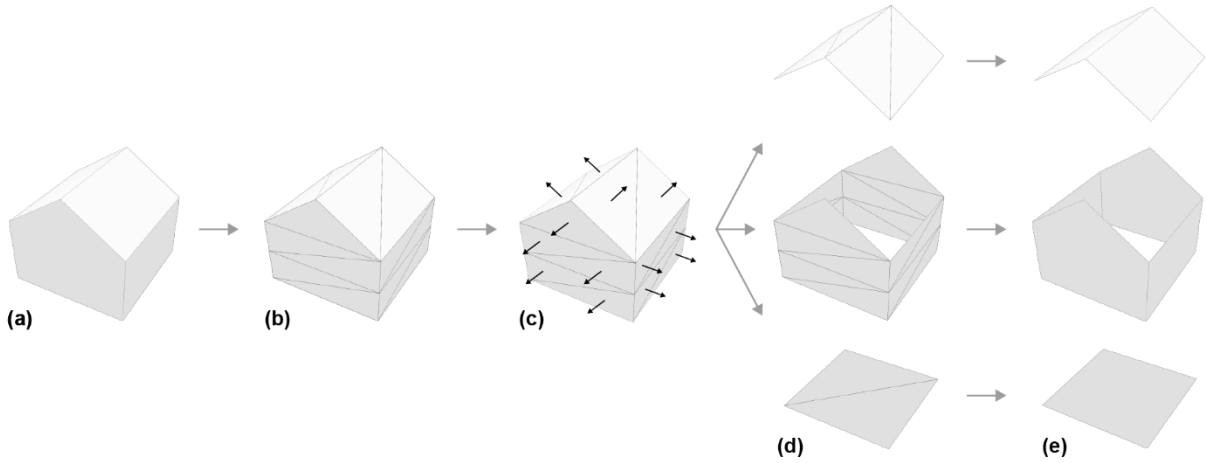


Figure 25:The FME script reconstructs Esri Multipatch building models (a) and reconstructs them to conform to NS building through the process of de-aggregation (b), extracting surface normas (c) and other relevant attributes to have distinct surfaces (d) of bottom, roof, and wall parts. The separate parts could subsequently be aggregated anew (e).

The separation of roofs, floors, and walls was essential to making the geometries of the building models compatible with NS building. After making that distinction, geometry parts belonging to bottom, floor, and wall surfaces were made into separate aggregate features and stored as a feature within each of the output models in different LoD (*Figure 26*).

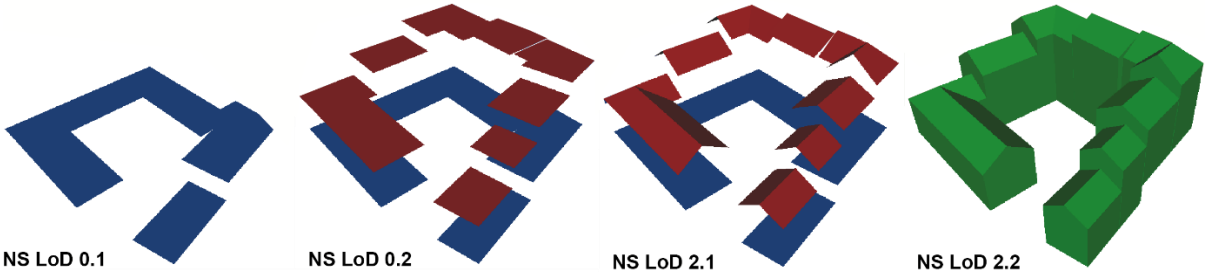


Figure 26: Example of the output models.

The number of surfaces that could be separated (bottom: 2300, roof: 2313, wall: 2315) did not match entirely with the number of initial building models (2320). The missing surfaces were only observed with the externally sourced models. The conversion of the externally sourced building models within the test dataset was successful to a large extent, with a few exceptions, partly due to some of the models being more detailed in structure and thus introducing more

nuances. The method established to separate the different surfaces works only if the input models are relatively simple, which was the case for most building objects. Some of the externally sourced models also had missing parts, e.g., floors, or inconsistent IDs, i.e., different *OBJECTIDs* for surfaces belonging to the same building object.

The output models (*Figure 27*) established with the workflow correlate with the geometries of NS building. Some issues arose with buildings that were not initially derived within Esri but combined with the municipality's database from SketchUp and Blender. The reasons behind those issues are discussed in *Section 5.4*. The conversion was successful in that the surfaces of the different LoDs could be modeled, apart from roof surfaces of NS LoD 0.1 (*Section 5.3*).

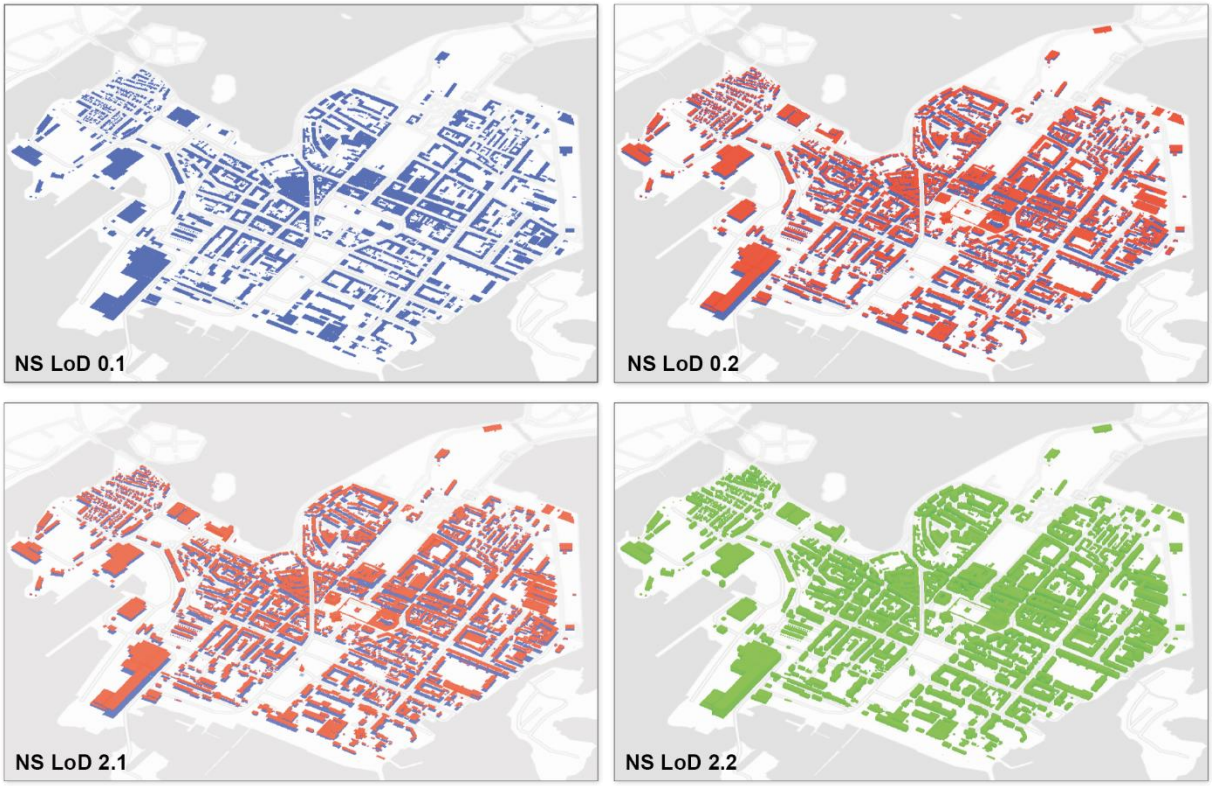


Figure 27: The entire test data was transformed into NS building models of different LoDs.

The attributes of the output models could be derived based on either the geometries or attributes of the initial dataset, such as roof pitch angle, roof form, their absolute heights, and the area (m^2) of the floor surfaces (*Figure 28*).

BuildingFID	OBJECTID▲	absHojdBotten	absHojdTak	bruttoarea	taktyp	takvinkel
Building_69	37	7.3726	28.1455	199.4243	sadeltak	30.7535
Building_70	38	2.4457	6.8728	129.7196	lågt lutande tak	0
Building_71	39	6.5956	24.7528	552.6943	sadeltak	<null>
Building_72	40	7.7607	30.4609	425.7311	sadeltak	26.0466
Building_73	41	2.6386	18.4112	166.467	sadeltak	19.0362
Building_74	42	1.4768	13.7461	302.3836	sadeltak	25.0078

Figure 28: Attribute table of NS building, with example records.

The output models (each consecutive LoD) could be successfully stored within the same geodatabase as the initial models, along with a stand-alone attribute table that may be paired with the models to gather the information of each object, as displayed in *Figure 27*.

Table 7: Size of output models compared to the initial test dataset.

Dataset	Size (MB)
Test data (Trossö)	5.130
NS LoD 0.1	0.515
NS LoD 0.2	0.976
NS LoD 2.1	1.970
NS LoD 2.2	5.750
Attribute table	0.133

The size of the individual LoDs within the database was compared to the initial dataset (*Table 7*). Lower LoDs resonate with having a smaller file size, as they are less detailed, whereas LoD 2.2 is larger than the initial data. The size of NS LoD 2.2 can be attributed to the models having all surfaces (bottom, roof, and walls) and a record for each per building part.

5. Discussion

5.1. Research relevancy and applicability of FME script

This study demonstrated how buildings in a 3D city model stored in an Esri environment could be converted to building models according to NS building and stored in the original geodatabase along with the original 3D models. The FME script developed for these purposes may be used as a starting point to further test the applicability of the Swedish standard within Esri and its compatibility with the format Esri MultiPatch. The script may be modified to fit the current version of it at a given time. It can be accessed at https://github.com/gu6863je-s/MultiPatchToNSbuilding.

In this study, the data were converted to NS building version 1.1, but since the standard was still under development at the time of writing, there have been some updates in the current version of NS building. It is also likely it will be subjected to further change in the future. However, these updates will not influence the general workflow developed in this study, even though some modifications of the FME scripts will be required to conform to the latest version of the standard. The processes of de-aggregating composite geometries of building models in MultiPatch format and the segmentation of surfaces according to bottom, roof and wall will be applicable regardless of possible modifications to the different levels of detail of the standard.

Considering that the development of the Swedish standard has come a long way, it is essential to test its applicability in different scenarios. Ensuring interoperability and data homogenization remains an important facet in GIS research. Since many municipalities in Sweden are working within Esri's platform, it is highly relevant to examine how the standard can be applied in this context. The method developed in this thesis is an attempt to specifically answer the testability of the Swedish standard in an Esri environment.

5.2. Spatial ETL within FME

It should be mentioned that the workflow arrived at is not the only way to transform the building models, and various paths could have been taken to get the same or similar results. For example, the calculation of the eave height can be established following a workflow that extracts the value from wall surfaces instead of roofs. The extraction using roofs was selected since it included fewer steps and because the data did not have building models with overhanging roofs. Thus, the eave height based on the minimum roof height should be identical to the eave height based on maximum wall height, excluding gable wall parts.

Because FME contains many different tools that may be used to restructure data, the challenge lay in selecting the most fitting transformers that did the job as desired. There is an extensive range of possibilities with the number of transformers FME makes available to its users. Further, the chance of developing plug-ins or introducing scripts to reconstruct data presents a

variety of options to get the intended results. It is now possible to access FME's workbench via Esri's ArcGIS platform with the Data Interoperability license, providing the possibility of creating a spatial ETL tool when working with data directly from ArcGIS. For example, a tool streamlining the process of converting MultiPatch to NS building may be established. As the focus of this study was on the method itself, this was not explored. However, it is mentioned to underline that FME and Esri are now in partnership and moving from one software to the other is made out to be efficient.

A downside to working with FME is that it is only possible to get a general sense of how their tools work based on documentation, forums, and via trials/errors, as their actual algorithms are in a black box and not open for inspection. This is understandable, as FME is proprietary software, but it may have been helpful to understand the applied transformers better. It is difficult to say how efficient the workflow established is, but there are obvious advantages to using FME to integrate Esri-derived 3D building models into NS building. The workflow established provides flexibility and can easily be adjusted to accommodate specific user cases if the source data differs somehow or if the version of NS building gets updated. Each branch of the workflow is testable, and since it branches into different outputs depending on the LoD, it is not necessary to run the whole process if only a specific LoD is desired. The script was not optimized for efficiency, but steps were simplified where they could observably be. However, it was not a time-consuming process to run the entire script for the test data. Even though the case study focused on a small dataset, the script should work well with a larger amount of data, such as a city model on a municipality level. This would be especially true for a 3D city model composed of building models derived solely within Esri, provided that the dataset is well harmonized. This pertains to the data's attributes and geometry. The script was developed for building models that have a similar level of detail as the test data, represented as a solid with a simple roof structure and roof orientation.

5.3. Modeling NS LoD 0.1 roofs

The difficulties of modeling NS LoD 0.1 pertain to the roof parts. They create a challenge in finding an automated method to construct them to fit the wide variety of buildings included in the test data. Firstly, it is unclear how the roof part of NS LoD 0.1 should be modeled, as the definitions of the LoDs are to a large extent undefined in the specification at the time of this writing. That could allow room for interpretation; however, not being an authority on the national specification, it would be inappropriate to define it arbitrarily.

Its modeling was tested based solely on how LoD 0.1 was visualized in the NS building guidelines. From its visualization, it was determined that the desired roof parts would be a 3D plane dissolved, or aggregated, based on the building ID (*BuildingFID*), which can include the main building and an additional building(s). The plane should be sloped to give an idea of the

building's eave height. This was tested by creating a process that selects extreme points of the aggregated roof surfaces and creating a convex hull polygon out of the points that have the three highest elevation (*z*) values. The elevation values were the *absEAVE* attribute created for each building object. The surface normal values could be retrieved and applied to the original plane using the convex hull and subsequently determine its slope. However, this only worked for specific buildings, while undesirable planes were computed for some of the roofs as the selection of extreme points was arbitrary. Depending on the points selected of the building plane, the resulting surface normals differed significantly, and some would produce roof planes that could descend below the eave height of the building. It was thus concluded that creating a planar surface extending across building objects with identical building IDs would be impossible for complex building structures, even though it works for buildings such as the one visualized in the guideline. Coupled with the uncertainty of how the roof parts should truly be modeled, it was determined not to include the process of creating NS LoD 0.1 roofs in the final FME script.

5.4. Meshes vs. triangle composites

Issues initially arose in establishing an automatic workflow for buildings composed of triangle meshes, which are 58% of the buildings featured in Karlskrona's entire 3D city model database but 5% of the test data.

The buildings composed of mesh geometry had the incorrect vertex normals assigned to them. Upon realizing that issue and resolving it, it allowed for the separation of faces as was done with the triangle composited building models. Some building models also displayed orientation errors, which could be remedied for floor plans and roofs because their orientation can be determined using the surface normals. The wrongly oriented wall parts were not fixed since they could not be filtered from the correctly oriented ones based on their surface normals. Thus, these parts had to remain as they were initially. These errors are attributable to the vertex order, and the rule (left hand or right hand) used to fit the geometry. The inconsistency with the orientation on meshed buildings was only apparent with building models imported from different software, SketchUp and Blender, and merged with the Esri geodatabase. They are likely constructed differently, which registers in the orientation of some of the surfaces. As mentioned, these errors were able to be fixed within the workflow, excluding wall parts, so that the faces have better consistency and harmony.

The meshed buildings introduced more errors, mainly because of their architectural detail, which presented difficulties in separating roof and floor parts from the walls as the surface normals could not be used stand-alone, as was with the Esri-derived models. As a result, not all output models give a true representation of the building's faces.

The externally sourced models introduced more nuance, which was difficult to account for on all fronts. However, it is highly realistic that a database would be composed of models from different sources. Presenting a workflow that addresses these building models and those solely constructed within Esri was important to make the workflow more inclusive of different kinds of scenarios. This results in an FME script that better accommodates a realistic 3D city FileGDB, composed of multi-source building models combined into one database. A future challenge could be to integrate multi-source 3D data better within the workflow, as the co-produced data was not of the same quality as the data created and handled solely within Esri. This was, for example, due to differences in their level of detail, orientation errors, geometry type. Having a larger dataset of multi-sourced models could introduce challenges that were not made apparent with the test data.

6. Conclusion

Conforming 3D building models to the Swedish national standard was tested using pre-existing models derived in Esri and externally sourced building models that had been pre-emptively joined with the tested Esri geodatabase of Karlskrona municipality. The conversion between the two formats, Esri MultiPatch and NS building, was performed using a spatial ETL process. It was facilitated within FME, which proved useful in the testability of each step implemented and had a wide variety of tools, denoted as transformers, to restructure the 3D city model of Karlskrona, specifically building models on the island of Trossö. Models were extracted from the initial FileGDB, transformation steps implemented, and output models loaded onto the initial database. Each LoD was stored as a single feature class and the attribute table stored in a stand-alone manner. The output models share a common identifier with the attribute table so that it may be joined with the building features. A workflow script was presented that streamlines the process of converting commercially derived 3D building models in Esri MultiPatch format to conform to the NS building standard.

The conclusions of the case study that is presented in this thesis can be summarized as follows:

- Building models derived in an Esri environment could be converted to NS building using a spatial ETL process and stored in a FileGDB.
- The automated workflow that was established was able to create the distinct surfaces needed for the different levels of detail of the Swedish standard for all Esri-derived models to be transformed. This excludes, however, roof surfaces of NS LoD 0.1.
- Issues regarding the modeling process pointed to the externally sourced building models, as they were, for example, differently constructed and sometimes of greater detail, and thus more nuanced than the Esri-derived models. Therefore, some of the surfaces of the externally sourced buildings could not be separated in the desired manner.

References

Ates Aydar, S., J. Stoter, H. Ledoux, E. Demir Ozbek, and T. Yomralioglu. 2016. Establishing a national 3D geo-data model for building data compliant to CityGML: Case of Turkey. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* XLI-B2**:** 79-86. DOI: 10.5194/isprsarchives-xli-b2-79-2016

Biljecki, F., K. Kumar, and C. Nagel. 2018. CityGML Application Domain Extension (ADE): overview of developments. *Open Geospatial Data, Software and Standards,* 3. DOI: 10.1186/s40965-018-0055-6

Biljecki, F., H. Ledoux, and J. Stoter. 2016. An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems,* 59**:** 25-37. DOI: 10.1016/j.compenvurbsys.2016.04.005

Biljecki, F., J. Stoter, H. Ledoux, S. Zlatanova, and A. Çöltekin. 2015. Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information,* 4**:** 2842-2889. DOI: 10.3390/ijgi4042842

Billen, R., A.-F. Cutting-Decelle, O. Marina, J.-P. de Almeida, M. Caglioni, G. Falquet, T. Leduc, C. Métral, et al. 2014. *3D City Models and urban information: Current issues and perspectives - European COST Action TU0801*. EDP Sciences.

Dimopoulou, E., E. Tsiliakou, V. Kosti, G. Floros, and T. Labropoulos. 2014. *Investigating integration possibilities between 3D modeling techniques*.

Drešček, U., M. Kosmatin Fras, J. Tekavec, and A. Lisec. 2020. Spatial ETL for 3D Building Modelling Based on Unmanned Aerial Vehicle Data in Semi-Urban Areas. *Remote Sensing,* 12**:** 1972. DOI: 10.3390/rs12121972

El-Mekawy, M., A. Östman, and I. Hijazi. 2012. A Unified Building Model for 3D Urban GIS. *ISPRS International Journal of Geo-Information,* 1**:** 120-145. DOI: 10.3390/ijgi1020120

El-Mekawy, M., A. Östman, and K. Shahzad. 2008. Geospatial Interoperability for IFC and CityGML : Challenges of Existing Building Information Databases. In *Innovations 08th, IEEE Conference, Dubai, December 16-18*.

Eriksson, H., T. Johansson, P.-O. Olsson, M. Andersson, J. Engvall, I. Hast, and L. Harrie. 2020. Requirements, Development, and Evaluation of A National Building Standard—A Swedish Case Study. *ISPRS International Journal of Geo-Information,* 9**:** 78. DOI: 10.3390/ijgi9020078

Esri. 2008. The Multipatch Geometry Type: An Esri White Paper. California, USA.

Ford, A. 2004. The visualisation of integrated 3D petroleum datasets in ArcGIS. In *Proceedings of 24th ESRI user conference*, 1-11.

Gröger, G., T. H. Kolbe, C. Nagel, and K.-H. Häfele. 2012. OGC city geography markup language (CityGML) encoding standard.

Gröger, G., and L. Plümer. 2012. CityGML – Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 71**:** 12-33. DOI: 10.1016/j.isprsjprs.2012.04.004

Gruber, U., R. Jens, and M. Seifert. 2014. Germany on the way to 3D-Cadastre. In *FIG Congress*, 16-21. Kuala Lumpur, Malaysia

Hajji, R., and R. Billen. 2012. Towards a collaborative and interoperable 3D Building database – A case study in Walloon region. EDP Sciences.

Herbert, G., and X. Chen. 2015. A comparison of usefulness of 2D and 3D representations of urban planning. *Cartography and Geographic Information Science*, 42**:** 22-32. DOI: 10.1080/15230406.2014.987694

INSPIRE. 2013. D2.8.III.2 Data Specification on Buildings – Technical Guidelines. European Commission Joint Research Centre.

IntelligentCities. n.d. Karlskrona: Intelligent Cities Challenge. Retrieved 21.02 2021, from https://www.digitallytransformyourregion.eu/cities/karlskrona.

International Organization for Standardization. 2007. ISO 19131:2007: Geographic information - Data product specifications. 40. Geneva, Switzerland.

International Organization for Standardization. 2015. ISO/IEC 2382:2015: Information technology - Vocabulary. Geneva, Switzerland.

Jin, S., R. R. Lewis, and D. West. 2005. A comparison of algorithms for vertex normal computation. *The Visual Computer*, 21**:** 71-82. DOI: 10.1007/s00371-004-0271-1

Julin, A., K. Jaalama, J.-P. Virtanen, M. Pouke, J. Ylipulli, M. Vaaja, J. Hyyppä, and H. Hyyppä. 2018. Characterizing 3D City Modeling Projects: Towards a Harmonized Interoperable System. *ISPRS International Journal of Geo-Information*, 7**:** 55. DOI: 10.3390/ijgi7020055

Jusuf, S. K., B. Mousseau, G. Godfroid, and J. H. V. Soh. 2017. Path to an integrated modelling between IFC and CityGML for neighborhood scale modelling. *Urban Science*, 1**:** 25.

Kang, T. W., and C. H. Hong. 2015. A study on software architecture for effective BIM/GIS-based facility management data integration. *Automation in Construction*, 54**:** 25-38. DOI: https://doi.org/10.1016/j.autcon.2015.03.019

Kolbe, T. H. 2009. Representing and Exchanging 3D City Models with CityGML. 15-31. Springer Berlin Heidelberg.

Kutzner, T., K. Chaturvedi, and T. H. Kolbe. 2020. CityGML 3.0: New Functions Open Up New Applications. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science,* 88**:** 43-61. DOI: 10.1007/s41064-020-00095-z

Lantmäteriet. 2019. Nationella specifikationer. Retrieved 22.01 2020, from https://www.lantmateriet.se/sv/webb/smartare-samhallsbyggnadsprocess/nationella-specifikationer.

Lantmäteriet. 2021a. Dataproduktspecifikation Byggnad: Version 1.0 test 3.

Lantmäteriet. 2021b. Informationsspecifikation Byggnad: Version 1.0 test 1.

Liu, X., X. Wang, G. Wright, J. Cheng, X. Li, and R. Liu. 2017. A State-of-the-Art Review on the Integration of Building Information Modeling (BIM) and Geographic Information System (GIS). *ISPRS International Journal of Geo-Information,* 6**:** 53. DOI: 10.3390/ijgi6020053

Luebke, D., M. Reddy, J. D. Cohen, A. Varshney, B. Watson, and R. Huebner. 2003. *Level of detail for 3D graphics*. Morgan Kaufmann.

OpenGL. 2013. Calculating a Surface Normal - Retrieved 25.02 2021, from https://www.khronos.org/opengl/wiki/Calculating_a_Surface_Normal.

Riksdagsförvaltningen. 2010. Plan- och bygglag (2010:900). ed. Riksdagen.

Roland, B., C.-D. Anne-Françoise, M. Claudine, F. Gilles, Z. Sisi, and M. Ognen. 2015. Challenges of Semantic 3D City Models: A Contribution of the COST Research Action TU0801. *International Journal of 3-D Information Modeling (IJ3DIM),* 4**:** 68-76. DOI: 10.4018/IJ3DIM.2015040106

Roschlaub, R., and J. Batscheider. 2018. Transformation of a Cadastre-Compliant 3D Building Model of Bavaria to INSPIRE. *International Journal of Spatial Data Infrastructures Research,* 13**:** 62-77. DOI: 10.2902/ijsdir.v13i0.449

Safe Software. 2020. FME: Transformer Reference Guide. British Columbia, Canada.

Safe Software. n.d.-a. Spatial ETL | FME. Retrieved 28. Mar. 2021, from https://www.safe.com/fme/key-capabilities/spatial-etl.

Safe Software. n.d.-b. Vertex Normals. Retrieved 27. Apr. 2021, from https://docs.safe.com/fme/html/FME_Desktop_Documentation/FME_ReadersWriters/!FME_Geometry/Vertex_Normals.htm.

Sakr, S., and A. Y. Zomaya. 2019. Encyclopedia of Big Data Technologies. Springer.

Sjölin, F., P. Weström, C.-M. Lanér, T. Johansson, P. Holgersson, M. Andersson, A. Larsson, P. Johansson, et al. 2019. Digital cities challenge: Digital transformation strategy for the city of Karlskrona. European Commission.

Solou, D., and E. Dimopoulou. 2016. Investigating the capabilities of semantic enrichment of 3D CityEngine data. SPIE.

Stoter, J., G. Arroyo Ohori, B. Dukai, A. Labetski, K. Kavisha, S. Vitalis, and H. Ledoux. 2020. State of the Art in 3D City Modelling: Six Challenges Facing 3D Data as a Platform. *GIM International: the worldwide magazine for geomatics*, 34.

Stoter, J., J. Beetz, H. Ledoux, M. Reuvers, R. Klooster, P. Janssen, F. Penninga, S. Zlatanova, et al. 2013. Implementation of a National 3D Standard: Case of the Netherlands. 277-298. Springer Berlin Heidelberg.

Stoter, J. E., G. Vosselman, J. Goos, S. Zlatanova, E. Verbree, R. Klooster, and M. Reuvers. 2011. Towards a national 3D spatial data infrastructure: case of the Netherlands. Photogrammetrie-Fernerkundung-Geoinformation.

Van Den Brink, L., J. Stoter, and S. Zlatanova. 2013. Establishing a national standard for 3D topographic data compliant to CityGML. *International Journal of Geographical Information Science*, 27**:** 92-113. DOI: 10.1080/13658816.2012.667105

VisitKarlskrona. n.d. The World Heritage City Karlskrona: Visit Karlskrona. Retrieved 21.02 2021, from http://www.visitkarlskrona.se/en/experience/the-world-heritage-city-karlskrona.

Worboys, M. F., and M. Duckham. 2004. *GIS: a computing perspective*. Boca Raton, Fla.: CRC Press.

Zhu, Q., M. Hu, Y. Zhang, and Z. Du. 2009. Research and practice in three-dimensional city modeling. *Geo-spatial Information Science*, 12**:** 18-24. DOI: 10.1007/s11806-009-0195-z

Zlatanova, S., J. Stoter, and U. Isikdag. 2012. *Standards for Exchange and Storage of 3D Information: Challenges and Opportunities for Emergency Response*.

Appendix A: Information flow for NS building

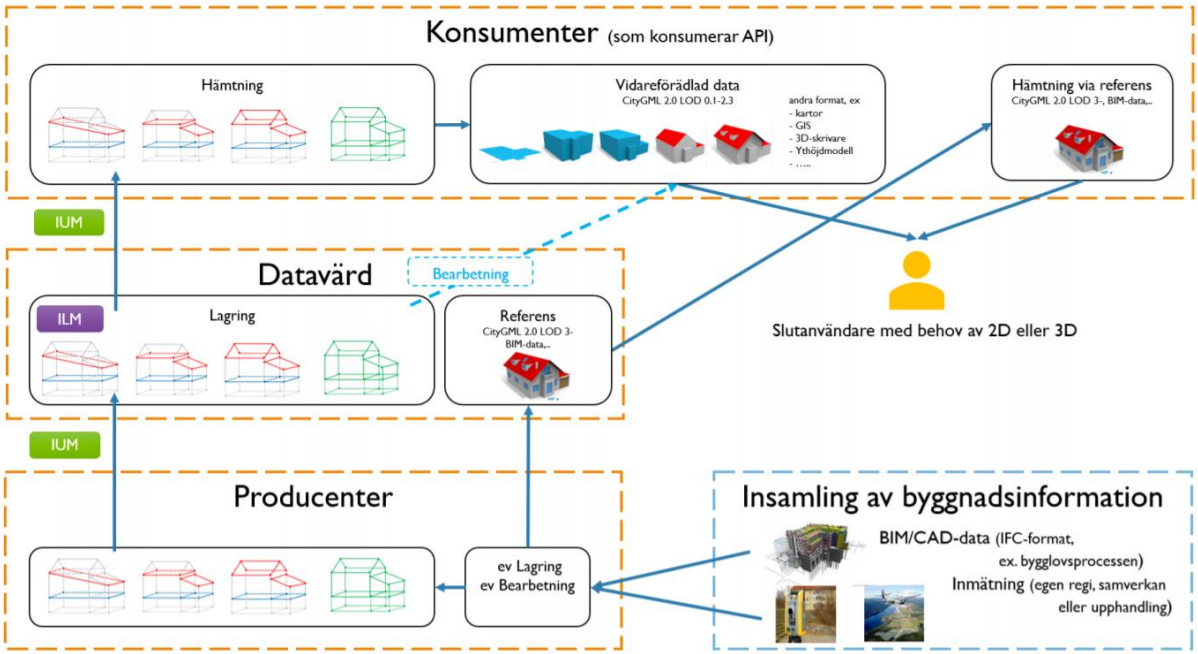


Figure 29: Information flow for NS building as represented in the modeling guidelines of the standard. Figure is from Lantmäteriet (2021a).