OGC Indoor Mapping and Navigation Pilot Engineering Report

Table of Contents

Publication Date: 2019-10-23

Approval Date: 2019-09-13

Submission Date: 2019-08-16

Reference number of this document: OGC 18-089

Reference URL for this document: http://www.opengis.net/doc/PER/IndoorPilotER

Category: OGC Public Engineering Report

Editor: Charles Chen (Skymantics, LLC)

Title: OGC Indoor Mapping and Navigation Pilot Engineering Report

---

OGC Public Engineering Report

indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Chapter 1. Executive Summary

The OGC Indoor Mapping and Navigation Pilot Initiative was sponsored by the National Institute of Standards and Technology (NIST) Public Safety Communications Research (PSCR) Division. This initiative addressed key challenges related to indoor mapping and navigation for the purpose of supporting first responders in fields such as fire-fighting. The focus of this initiative was on developing the capabilities and workflows required for pre-planning operations. This included scanning each building to produce a point cloud dataset and converting this source data into various intermediate forms to support the generation of indoor navigation routes. This Engineering Report (ER) describes the work conducted in this initiative, the lessons learned captured by participants, and future recommendations to support the public safety efforts and interoperability of the standards. It is expected that future OGC initiatives will address the real-time, event-driven aspects of indoor mapping and navigation for first response situations.

First responders typically survey high-risk facilities in their jurisdiction at least once per year as part of a pre-planning process. Pre-planning outputs are often in the form of reports, and first responders may generate their own hand-drawn maps during the process or annotate available floor plans (e.g., from computer-aided design models). Pre-planning is time-consuming, inefficient, and inherently complex considering the information and level of detail that should or could be captured, the lack of automation, and the difficulty identifying notable changes to facilities and infrastructure during successive pre-planning surveys.

Mobile three-dimensional (3D) Light Detection and Ranging (LiDAR) has been identified as a potentially transformational technology for first responders. Using LiDAR and 360-degree camera imagery, coupled with advanced software processing, first responders could efficiently capture 3D point clouds and a wealth of other information, both observed and derived, while walking through buildings as part of routine pre-planning operations. The use of 3D LiDAR and imagery has many potential upsides beyond just creating point clouds for visualization and mapping (e.g., use in localization, object classification, integration with virtual/augmented reality solutions, change detection, etc.).

Requirements and Research Motivation

The primary motivation for addressing the topic of indoor-mapping is based on a real-world scenario where public-safety first-responders such as firefighters are attempting to navigate a building to rescue a civilian or fellow downed firefighter. The scenario is focused on operational pre-planning using LiDAR based point cloud scans of buildings which may be used to map navigational routes around hazards and risks. To accelerate research and development for this public-safety-driven scenario, the requirements of this initiative were to conduct the following prototyping and demonstration activities:

- Create and convert 3D indoor LiDAR point cloud models and associated imagery to functional building and navigation models.
- Generate, store, and serve point cloud, building, and navigation models for visualization and navigation.
- Derive dynamic turn-by-turn indoor navigation instructions based on the navigation model.
- View and annotate point cloud data, imagery, and building models, along with navigation routes

and instructions into, through, and out of buildings.

- Capture and annotation of Public Safety features through buildings based on scans or images and including the public safety features as part of a Public Safety Extension.

Further, this initiative supported the improved interoperability of location-based technologies for indoor mapping and navigation through the use of OGC web services and OGC standards such as City Geography Markup Language (CityGML) and IndoorGML. The benefits from this initiative include the following:

Derived Benefits

- Benefits to end users
  - First responders benefit from new tools and applications for improved awareness
  - Emergency evacuation processes can be expedited through better tools
  - Demonstration videos provide better community outreach for educational purposes
- Benefits to SWG/DWG
  - Working groups may learn about the nuances in the use of point cloud data and associated standard data models
  - New insights into existing gaps in IndoorGML and CityGML for interoperability and real-world applications
  - Development of new algorithms or applications for IndoorGML navigation applications
- Benefits to Developers
  - Ensure standards are truly interoperable for future standards and releases
  - Discovery of possible improvements to OGC standards and lessons learned
  - Lessons learned can assist with future research and reduce the learning curve

Key Findings

This pilot was successful in demonstrating 1) transformations of Point Clouds to CityGML with Public Safety Application Domain Extension (ADE), 2) transformations of CityGML into IndoorGML with Public Safety Extension, and 3) visualizations of the Public Safety data in a Pre-planning Tool Client using open standards and basic navigation capabilities. The participants met several challenges regarding data quality in the provided point cloud data, different interpretations of standards which caused errors and anomalies, and limited toolsets for manipulating CityGML and IndoorGML data. Many of these challenges are described within the individual ER sections and also in the final conclusions of the demonstration (see Chapter 11, *Public Safety Scenario (Demonstration)*).

Through the efforts of this pilot initiative, the following key findings have been determined:

- Scanning was just the first step. It is critical that point cloud data scans are conducted thoroughly. Every effort should be made to ensure every unlocked door is opened and every locked door is annotated, and the resulting point cloud should be generated with Red-Green-Blue (RGB) data, not just grayscale, to ensure the best results from automation tools. Additional data cleansing of the point cloud data will still be required due to inherent building physics such as reflective surfaces and transparent glass. Automated conversions from one data form to another rarely produced a 100% complete result. A substantial amount of manual effort was required to repair the resulting output and fill gaps such as missing walls and doors. Once data has been cleansed and transformed to CityGML, manual annotation of public safety features is necessary. It is anticipated that future improvements in tools and software algorithms can begin to reliably automate some of these tasks.

- CityGML Public Safety ADE was created as an ontology for first-responders. IndoorGML Core did not provide all requirements for Public Safety applications, so the IndoorGML Public Safety Extension was created to support the Public Safety contextual items.
- If possible, all data should be georeferenced. If not, the data needs to be checked and manually repositioned to ensure the resulting data models are overlaid correctly. Misalignment due to improper georeference can result in improper subspacing due to rotation, incorrect network grid modeling, and navigation routing anomalies (e.g., zigzag lines).
- Current editing tools for CityGML and IndoorGML are not sufficient enough to support editing functionality, and so use of other tools such as Revit (an IFC Editor) or TICA (an open source CityGML editor), to process point cloud data was necessary. It was found that conversion from point cloud to IFC first, then to CityGML simplified the process due to the better editing tools.
- It remains an open question regarding how to best present navigation options to the end user. Higher density indoor subspacing network models allow for more precise navigation but clutter up the user’s view. Two methods are documented in this ER regarding room space calculations using centroid of rooms or grid-based subspacing. The goal is to achieve turn-by-turn directions, but the best result depends on the building and some combination of methods.
- Some differences in GML 3.1.1 used by CityGML and GML 3.2.1 used by IndoorGML caused some issues. Future versions should consider harmonization of the standards.

Recommendations for Future Work

Future work items are documented in Section 11.4.2, “Recommendations for Future Work”. In summary, the recommendations for future work are as follows:

- In terms of public safety, consideration should be made for both Indoor and Outdoor mapping as well as utilizing road and water networks and other accessibility
- The National Alliance for Public Safety GIS Foundation (NAPSG) provides the public safety symbology, but CityGML and IndoorGML are semantic models that are not well-suited for symbology. The Styled Layer Descriptor (SLD) and Symbology Encoding (SE) standards could be improved beyond 2D into 3D to support this use case.
- A large extent of this initiative involved data conversions into data models. For navigation, modeling must be extremely clean, whereas validation checks and business rules should be created to ensure the data is properly converted and suitable for use in calculating navigable routes.
- Closer coordination between CityGML and IndoorGML is needed to ensure interoperability. This includes the GML difference, harmonized public safety ADE/Extension, and use of common building/space/room IDs.
- Development of web and mobile clients could support public safety user applications as well as consideration for JSON, GeoJSON, and OpenAPIs.

- Augmented Reality could provide significant benefits to public safety use cases, and use of 3D models used in this initiative can support these efforts.

1.1. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Contacts

Name	Organization	Role
Charles Chen	Skymantics	Editor
Dean Hintz	Safe Software	Contributor
Ki-Joune Li	Pusan National University	Contributor
Jason MacDonald	Compusult	Contributor
Ken Geange	Compusult	Contributor
Mohsen Kalantari	Faramoon	Contributor
Mike Cross	Skymantics	Contributor
Abdoulaye Diakite	University of New South Wales	Contributor
Chih-Wei Kuan (Will)	Feng Chia University	Contributor

1.2. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Chapter 2. References

The following normative documents are referenced in this document.

- OGC: OGC 12-019, OGC® City Geography Markup Language (CityGML) Encoding Standard, Version 2.0 [https://portal.opengeospatial.org/files/?artifact_id=47842]
- OGC: OGC 14-005r5, OGC® IndoorGML Implementation Specification, Version 1.0.3 [http://docs.opengeospatial.org/is/14-005r5/14-005r5.html]
- [OGC: OGC 19-032, CityGML Public Safety ADE Engineering Report]
- OGC: OGC 06-121r9, OGC® Web Services Common Standard [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2]
- OGC: OGC 14-065, OGC® WPS 2.0 Interface Standard [http://docs.opengeospatial.org/is/14-065/14-065.html]
- OGC: OGC 08-062r7, OGC® Reference Model, Version 2.1 [http://rap.opengeospatial.org/orm.php]
- OGC: Indoor Mapping and Navigation CFP, Version 1.2 [https://portal.opengeospatial.org/files/?artifact_id=79833/]
- OGC: 09-025r2, OGC® Web Feature Service 2.0 Interface Standard – With Corrigendum [http://docs.opengeospatial.org/is/09-025r2/09-025r2.html]
- Cloud Compare [https://www.danielgm.net/cc/]
- Safe Software: Scenario: Victoria Airport Esri Geodatabase to IMDF [https://knowledge.safe.com/articles/76176/scenario-victoria-airport-esri-geodatabase-to-imdf.html]
- wetransform: hale:studio software [https://www.wetransform.to/products/halestudio/]
- STEMLab: Spatio-Temporal Databases Laboratory (STEMLab) IndoorGML Tools [https://github.com/stemlab]
- National Alliance for Public Safety GIS (NAPSG) Public Safety library of features [https://www.napsgfoundation.org]
- NIST: NIST Public Safety Innovation Accelerator Program Point Cloud City Program [https://www.nist.gov/news-events/news/2018/10/pscr-awards-over-750k-psiap-point-cloud-city]
- Hancock County Point Cloud City [https://www.nist.gov/ctl/pscr/hancock-county-point-cloud-city]
- PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, April 10, 2017 [https://arxiv.org/pdf/1612.00593]
- StemLab: IndoorGML to CityGML with PSExtension [https://github.com/STEMLab/IndoorGML2CityGML_PSExtension]
- UNSW: IndoorGML as a Linear Cell Complex (LCC) class of CGAL, University of New South Wales [https://github.com/grid-unsw/IndoorGML2LCC]
- Safe Software: OGC Indoor GML Pilot [https://knowledge.safe.com/articles/96851/ogc-indoor-gml-pilot.html]
- 52North: JavaPS [https://github.com/52North/javaPS]

Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard OGC 06-121r9 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

- Light Detection and Ranging (LiDAR) is a surveying method that uses laser light to illuminate an object or space and measure reflected light with a sensor. The variations in laser reflection and wavelengths is then used to generate a 3-dimensional representation of the target.

3.1. Abbreviated terms

- 3D 3-Dimensional
- 3DPS Three-Dimensional Portrayal Service
- ADE Application Domain Extension
- AEC Architecture, Engineering, and Construction
- AR Augmented Reality
- BIM Building Information Modeling
- CFP Call for Participation
- CGAL Computational Geometry Algorithm Library
- CLI Command Line Interface
- CSW Catalog Service for Web
- CSW-T Transactional Catalog Service for Web
- DDIL Denied, Degraded, Intermittent, or Limited
- DWG Domain Working Group
- EPSG European Petroleum Survey Group
- ER Engineering Report
- ETL Extract Transform Load
- FME Feature Manipulation Engine (Safe Software)
- FTP File Transfer Protocol
- GML Geography Markup Language
- IFC Industry Foundation Class
- ISO International Organization for Standardization
- IMDF Indoor Mapping Data Format (Apple Inc.)
- LAZ LiDAR data file extension (.laz)
- LCC Linear Cell Complex
- LiDAR Light Detection and Ranging

- MBB Minimum Bounding Box
- NAPSG National Alliance for Public Safety GIS
- NIST National Institute of Standards and Technology
- NRG Node Relation Graph
- OGC Open Geospatial Consortium
- ORM OGC Reference Model
- OWS OGC Web Services
- PC Point Cloud
- PCD Point Cloud Data
- PS Public Safety
- PSCR (NIST) Public Safety Communications Research Division
- PSX Public Safety Extension
- RGB Red Green Blue
- RM-ODP Reference Model for Open Distributed Processing
- SWG Standards Working Group
- WFS-T Transactional Web Feature Service
- WGS84 World Geodetic System 1984
- WPS Web Processing Service
- WG Working Group (SWG or DWG)

Chapter 4. Overview

The Indoor Mapping and Navigation Pilot is an OGC Innovation Program initiative that addresses key challenges related to indoor mapping and navigation for first responders. The focus is on developing capabilities and workflows required to support pre-planning operations. These first responders periodically survey high-risk facilities as part of a pre-planning process and formulate reports which require floor plans and maps. Considerations are made to simplify this process to improve the efficiency, reduce complexity, and capture the necessary details through use of automation and tools.

This overview provides the following viewpoints according to the OGC Reference Model (ORM) which provides an architecture framework for the ongoing work of the OGC. The structure of the ORM is based on the Reference Model for Open Distributed Processing (RM-ODP).

4.1. Enterprise Viewpoint

The concept in this initiative is to take advantage of currently available LiDAR technology and camera imagery to capture a building as a set of 3D point cloud data during routing pre-planning operations, and then transform the data into usable formats for visualization and mapping. These tools already exist for the architecture, engineering, and construction (AEC) community, and it is expected that future investments will significantly lower the costs of tools such that it will become a cost-effective approach for public safety, building managers, and other industries.

In order to demonstrate this public-safety driven scenario, the following activities were conducted in this initiative:

- Create and convert 3D indoor LiDAR point cloud models and associated imagery into functional building and navigation models.
- Store and serve point cloud, building, and navigation models for visualization and navigation.
- Derive dynamic indoor routes instructions based on the navigation model.
- Enrich and annotate building models and navigation models, along with navigation routes with public safety features to help guide first responders in pre-planning activities.

4.2. Information Viewpoint

The Information Viewpoint considers the information models and encodings that will make up the content of the services and exchanges to be extended or developed to support this initiative. The following technical service components, data exchanges, and data model extension were developed by the initiative participants and demonstrated in this initiative:

- **Building Data** - The building data consists of captured 3D point cloud data of buildings
- **Public Safety Features CityGML Application Domain Extension (ADE)** - An XML extension of the CityGML standard to annotate features with public safety specific metadata and descriptions.
- **Building Modeler Service I (2 instances)** - A web processing service that converts point cloud

data into CityGML format.

- **Navigation Modeler Service I (3 instances)** - A web processing service (WPS) that converts CityGML format data into IndoorGML format.
- **Building Model Repositories (3 variations)** - A data storage and access service (Catalog) that provides the capability to store and retrieve data.
- **Indoor Navigation Service I (2 instances)** - A web processing service that calculated an indoor navigable route using the IndoorGML data.
- **Pre-planning Tool Client I (2 instances)** - A user interface client to interact with the various services and data components in a public safety scenario.

Figure 1 below shows a tiered technical viewpoint of the components in this pilot. Each tier is comprised of various components which access each other through open standards and interfaces to demonstrate interoperability. The **Access Tier** represents those components which include the data and the component services required to store and access the data. The **Business Process Tier** represents the components which provide data conversion into other data formats. The **Client Tier** represents the pre-planning clients which provide a user interface to interact with the components.

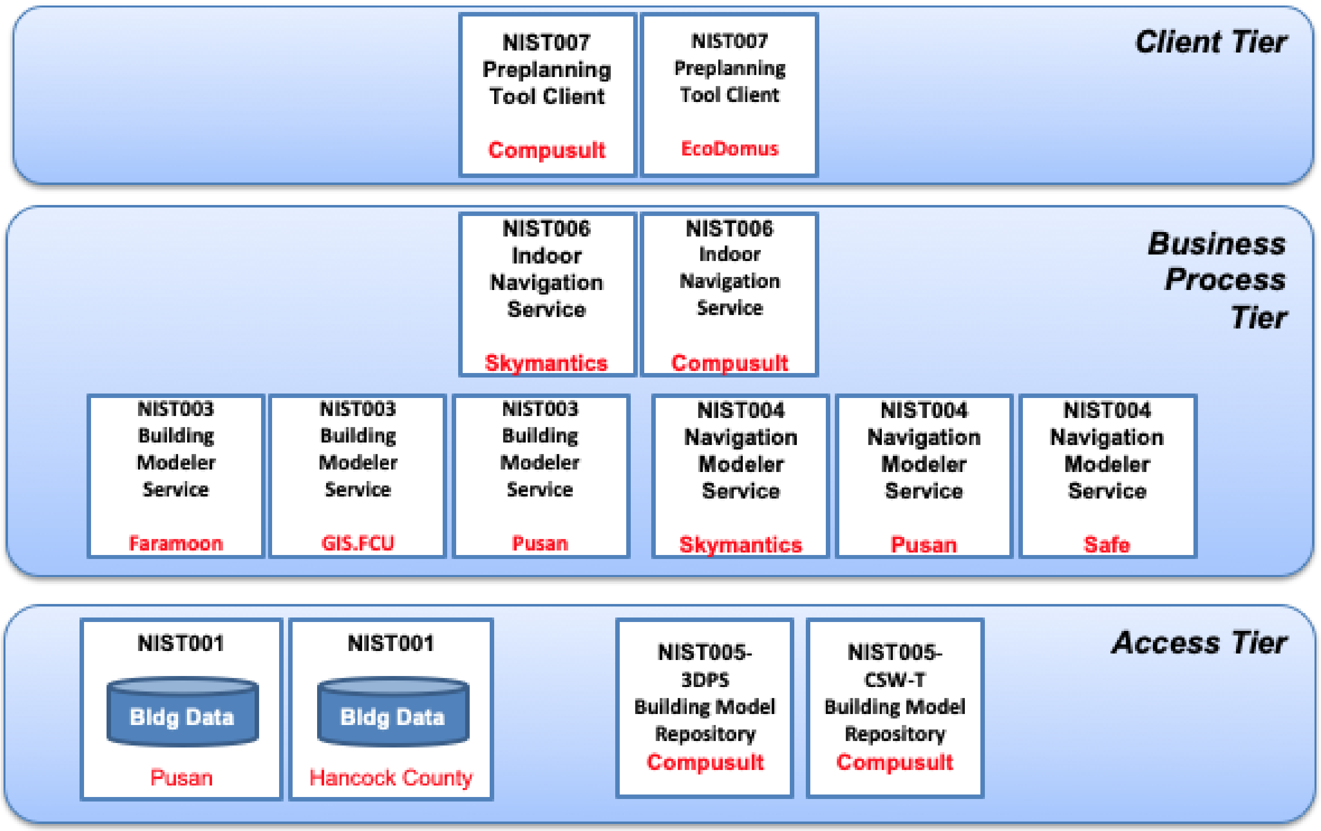

Client Tier
NIST007 Preplanning Tool Client Compusult
NIST007 Preplanning Tool Client EcoDomus
Business Process Tier
NIST006 Indoor Navigation Service Skymantics
NIST006 Indoor Navigation Service Compusult
NIST003 Building Modeler Service Faramoon
NIST003 Building Modeler Service GIS.FCU
NIST003 Building Modeler Service Pusan
NIST004 Navigation Modeler Service Skymantics
NIST004 Navigation Modeler Service Pusan
NIST004 Navigation Modeler Service Safe
Access Tier
NIST001 Bldg Data Pusan
NIST001 Bldg Data Hancock County
NIST005-3DPS Building Model Repository Compusult
NIST005-CSW-T Building Model Repository Compusult

Figure 1. Component Architecture

4.3. Computational Viewpoint

The Computational Viewpoint is concerned with the functional decomposition of the system into a set of objects that interact at interfaces – enabling system distribution. The interface architecture for the Indoor Mapping and Navigation Pilot initiative is derived from the functional architecture provided in the CFP with some key changes. The architecture, as described by the Indoor Mapping and Navigation CFP [https://portal.opengeospatial.org/files/?artifact_id=79833/], places a focus on the use of

OGC web services for implementation of the modeler services as seen in Figure 2.

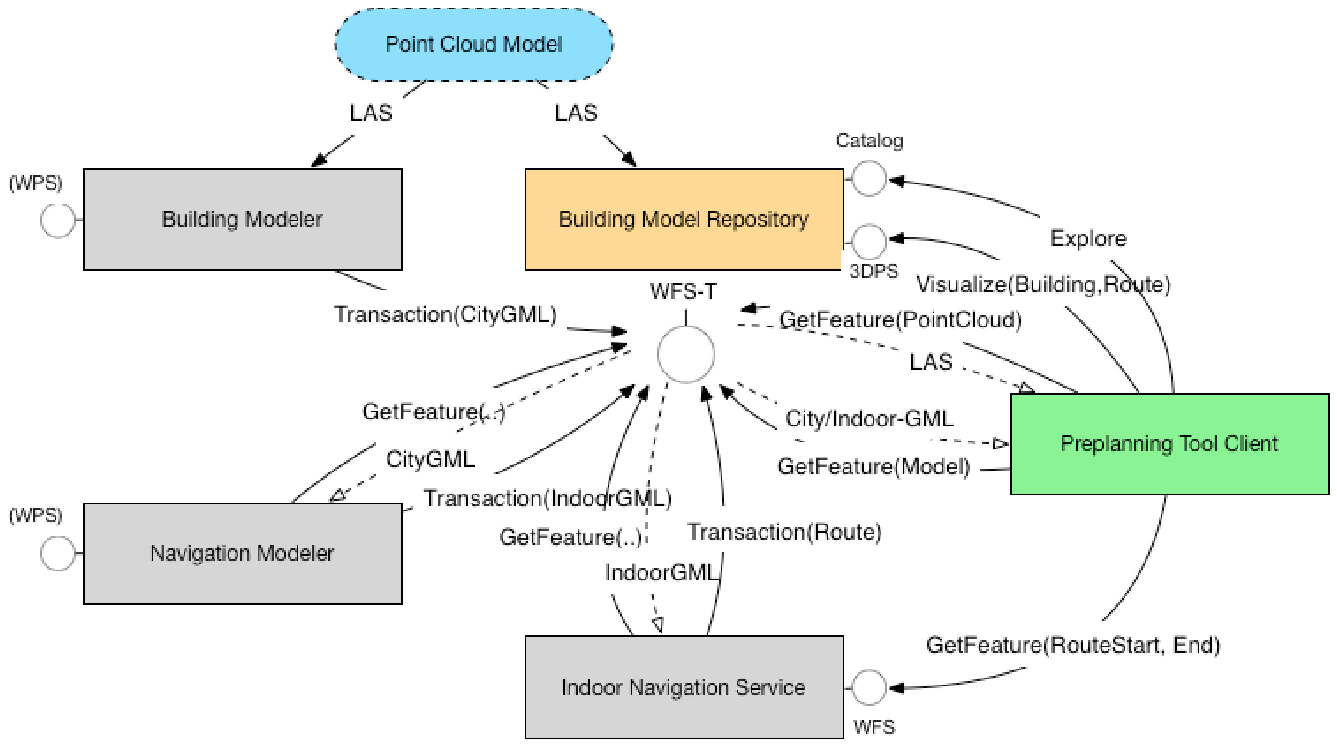


Figure 2. CFP Component Architecture

During the kick-off discussions, it was determined that the initial viewpoint of the Building Modeler and Navigation Modeler fit well in the paradigm for WPS 2.0. However, it was determined that the interactions between the Pre-planning Tool Client and the Indoor Navigation Service is is better implemented using WPS instead of WFS. Additionally, the WFS-T originally envisioned for the Building Model Repository was replaced with a CSW-T interface. All components interface directly with the Building Model Repository via the CSW-T interface, retrieving necessary data and storing the resulting transformed data. The clients access the CSW-T to retrieve each data set to display in the client user interface and generate indoor routes for navigation. Figure 3 shows the updated component architecture after discussion at the Indoor Pilot Kick-off meeting.

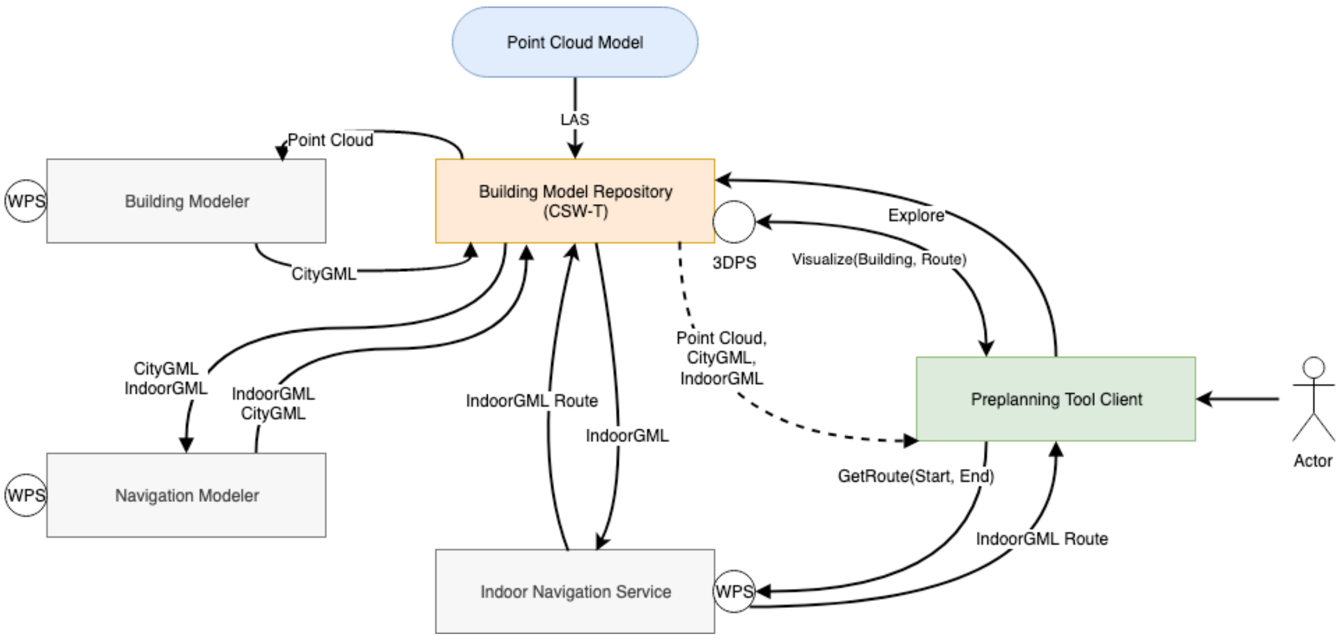


Figure 3. Updated Component Architecture

It should be noted that the Navigation Modeler service was implemented in two different ways: 1) converting the CityGML output of the Building Modeler into IndoorGML, and 2) reversely, converting point cloud data directly into IndoorGML and then back into CityGML. The reason for this is due to the fact that one participant focused on the indoor spatial mapping from point cloud data rather than from a top-down building geometry to indoor space workflow. More details can be found in Chapter 7, *Navigation Modeler Service*.

4.4. Engineering Viewpoint

The Engineering Viewpoint is concerned with the infrastructure required to support system distribution. It focuses on the mechanisms and functions required to support distributed interaction between objects in the system, and hides the complexities of those interactions. It exposes the distributed nature of the system, describing the infrastructure, mechanisms and functions for object distribution, distribution transparency and constraints, bindings and interactions.

The following sequence diagram describes the conceptual flow for the Pilot architecture.

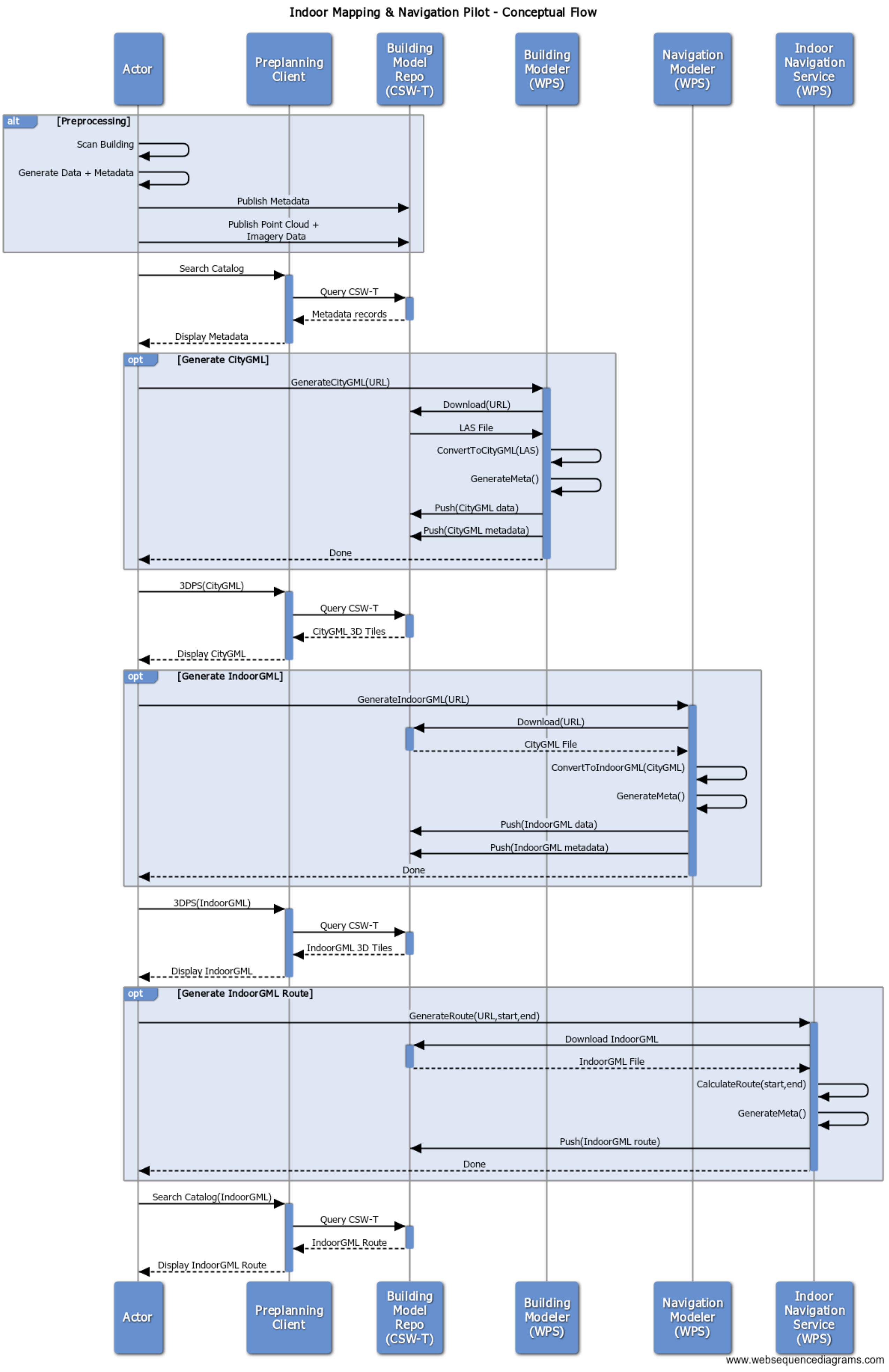


Figure 4. Indoor Mapping & Navigation Pilot - Conceptual Flow

4.5. Technology Viewpoint

Each of the following chapters in this ER describe the individual Technology Viewpoint of the various components, how they are developed, the software tools utilized, and the distribution of components needed to achieve the result of the Technology Integration Experiments (TIEs) conducted in this initiative.

Chapter 5, *Data Sources* describes the various data sets provided in this pilot. The datasets include point clouds, CityGML, and IndoorGML data. This chapter also catalogs the provided sample data, the official demonstration data, data derived and converted from the original formats, and details the data cleansing and enrichment. Some data is enriched with Public Safety Features CityGML ADE, which is documented in a separate ER. For more information, refer to the OGC 19-032 CityGML Public Safety ADE Engineering Report.

Chapter 6, *Building Modeler Service* describes the Building Modeler Service which converts point cloud data into CityGML data. This task was to create a building modeler application that can convert point cloud models and associated images (such as those generated by the Building Data) into semantic 3D building models compliant with the most recent or stable version of CityGML (2.0). Models generated by this component are notated with Public Safety Features from the CityGML ADE. The building modeler may be developed as a web processing service (WPS) compliant with the OGC WPS 2.0 Interface Standard (OGC14-065), although this was not a strict requirement.

Chapter 7, *Navigation Modeler Service* describes the Navigation Modeler Service which converts CityGML data into IndoorGML data. This task was to create a navigation modeler application that can convert output from the Building Modeler Service to IndoorGML usable for indoor navigation. The Navigation Modeler also generated navigation network information to support route calculations by the Indoor Navigation Service. The navigation modeler may be developed as a WPS compliant service, although this was not a strict requirement.

Chapter 8, *Building Model Repository* describes the Building Model Repository which stores the various datasets and makes them available to other services and end users. This task was to create a building model repository to store and serve point cloud, building, and navigation models (see Building Data, Building Modeler Service, and Navigation Modeler Service). The repository exposes a model catalog and provides authenticated access. The repository is interoperable with application clients.

Chapter 9, *Indoor Navigation Service* describes the Indoor Navigation Service which consumes the IndoorGML data to produce a navigation route based on parametric inputs. This task involved creating an indoor navigation service that can derive ‘turn-by-turn’ instructions between any two points in a building, including exits, based on network models created by the Navigation Modeler Service and stored in the IndoorGML. The service used navigation algorithms, not necessarily developed in this Initiative, which were optimized for specific criteria (e.g. routes optimized based on time, distance, or risk) requested by the Pre-planning Tool Client.

Chapter 10, *Pre-planning Tool Client* describes the Pre-planning Tool Client for end user access to the services and dataset renderings for mapping and navigation display. This task was to create a visualization client application for users to request and view point cloud data and building models from the Building Model Repository, as well as navigation instructions and routes from the Indoor Navigation Service. The client is a graphical user interface that enables public safety personnel to view and annotate models captured during pre-planning. The client should seamlessly transition between 2D and 3D views and should allow users to visualize hypothetical routes into, through, and out of buildings along with the appropriate metrics (e.g. estimated time, distance, risk, etc.) based on additional input parameters considered by the navigation service. The client should use the OGC WFS Interface Standard and OGC 3DPS to request and receive models, scenes, and services.

Chapter 5. Data Sources

This chapter describes the point cloud source data provided by the sponsor (Section 5.3, “Sponsor Provided Data”) and participants (Section 5.2, “Participant Provided Data”). It also describes the data derived from these sources into the CityGML, IndoorGML, and public safety extensions.

5.1. Types of Data

This Pilot uses several data sources for testing and experimentation including Point Cloud data, CityGML data, and IndoorGML data. CityGML is an open standardized data model and exchange format to store digital 3D models of cities and landscapes. The data files are hosted by the OGC in a secure File Transfer Protocol (FTP) server accessible to participants. Enriched data includes cleansed data which is then enriched with an Application Domain Extension (ADE), a built-in mechanism of CityGML to augment its data model with additional concepts required by particular use cases. The ADE is a mechanism for enriching the data model with new feature classes and attributes, while preserving the semantic structure of CityGML. IndoorGML is an OGC standard for an open data model and XML schema for indoor spatial information. IndoorGML does not officially have an extension mechanism, but one was needed to carry the Public Safety features that were forwarded from the CityGML Public Safety ADE.

5.1.1. Point Cloud Data

The point cloud data used in this pilot is generated through LiDAR scans of various buildings. Another concurrent project was developing LiDAR building scans, and as a result the data contribution (Section 5.3, “Sponsor Provided Data”) was delayed until later in the pilot. Therefore, sample data (Section 5.2, “Participant Provided Data”) provided by the participants was used for the beginning of the development phase. Test data provided by Pusan National University (PNU) was used for initial development and testing. Later, efforts transitioned to official data provided by the sponsor.

5.1.2. CityGML Data

CityGML 2.0 was used to represent the building geometries including walls, floors, ceilings, doors, windows, furniture, etc. In order to begin development and testing earlier in the pilot, CityGML was generated from Victoria Airport’s publicly available sample dataset in Apple IMDF using FME from Safe Software. The data was later enriched with Public Safety ADE. Pusan National University generated IndoorGML from point cloud data using a partial manual process with an open source software application, and then converted IndoorGML to CityGML. Other participants’ early attempts to generate CityGML datasets from point clouds resulted in CityGML with a few large mesh features that lacked structure and were not readily convertible to IndoorGML.

One workflow that proved productive was to do automated conversion of point cloud to IFC data, and then use an editing tool (Revit) to structure and enrich the data for use in IndoorGML. A key part of this process was room generation. FME was used to convert and simplify the IFC to CityGML in a mostly automated process with some minor configurations to account for dataset variations.

5.1.3. IndoorGML Data

IndoorGML data describes the indoor spaces of a building. Spaces are divided into navigable and non-navigable spaces. The navigable spaces are linked to generate a navigational network, which can then be traversed to generate navigable routes. IndoorGML does not officially have an extension mechanism, but the participants collaborated to develop an IndoorGML Public Safety Extension to carry the public safety features that were forwarded from the CityGML Public Safety ADE. This functionally is being proposed to the OGC IndoorGML SWG for consideration and is planned to be used for the new upcoming IndoorGML 2.0 work.

5.2. Participant Provided Data

The following sample data was provided by the participants for testing and evaluation as components were being developed. This was done to ensure parallel development to achieve the TIEs later in the project while participants were waiting on the demonstration data.

5.2.1. Point Cloud - Korea University Central Plaza

This data contains a LiDAR point cloud scan of Korea University's Central Plaza, an underground plaza at the Anam Science Field Campus. The author, TeeLabs Co., is a collaborative work partner with Pusan National University and supplied the following point cloud data for use in this pilot initiative. The data contains a large point cloud with more than 100 million points and contains images with camera pose data. Most of the noise in the data has been cleansed, and each object is individually separated. The data was cleansed using Cloud Compare [https://www.danielgm.net/cc/], an open source point cloud viewer.

1. Name: KU-Central-Plaza
2. Source: 3D LiDAR x 2EA, 360 Camera, IMU
3. Editing: Cleansing (PCD Noise Removal) using a semi-automatic process
4. Author: TeeLabs, Co. LTD.

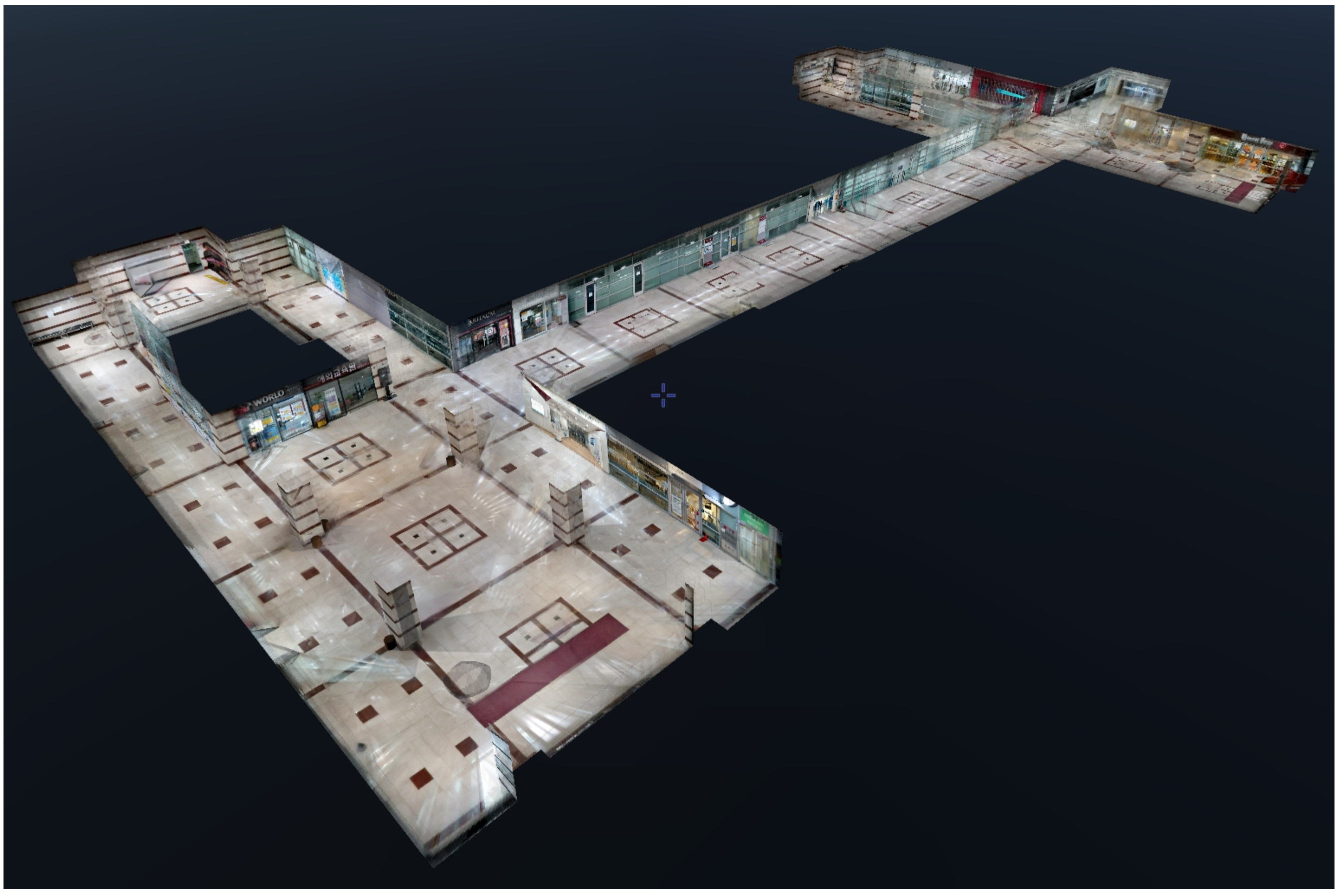


Figure 5. Korea University, Central Plaza Point Cloud

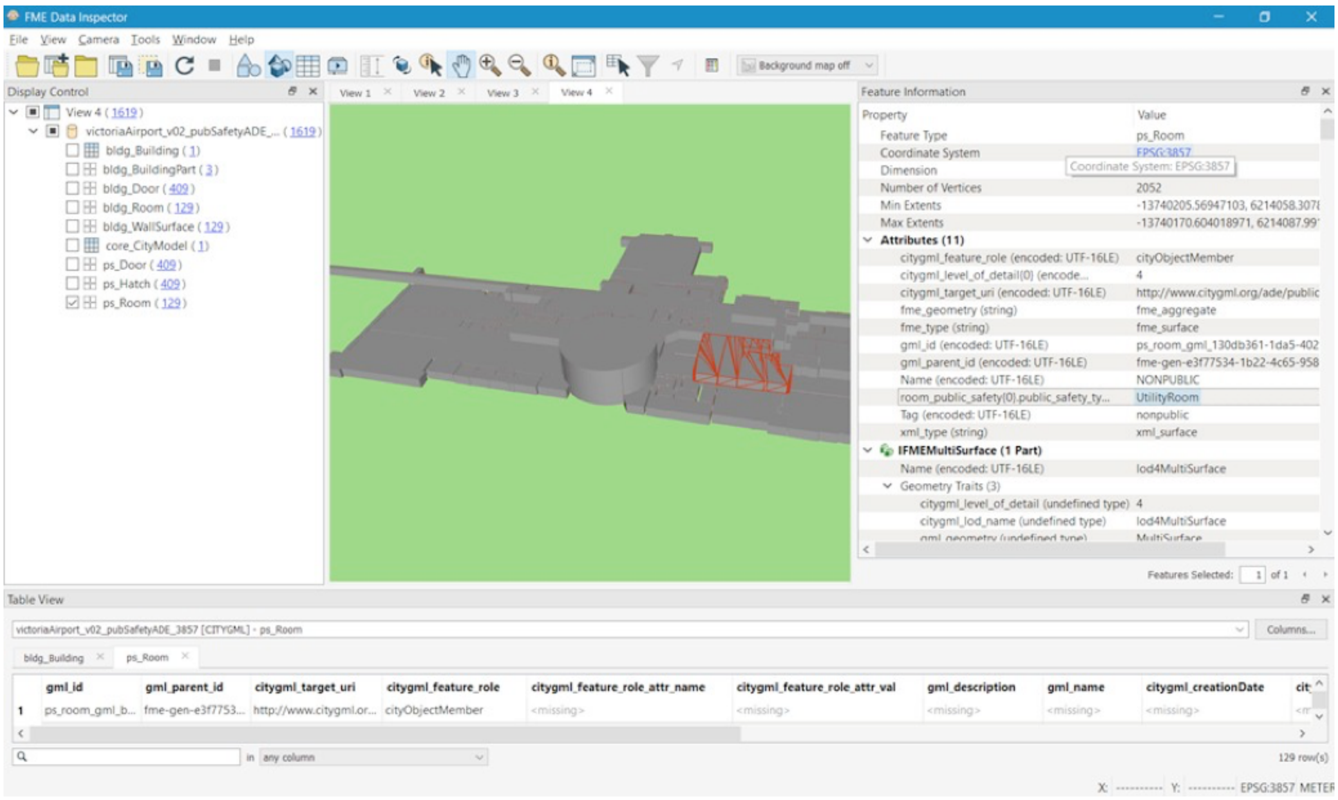
5.2.2. CityGML - Victoria Airport

Safe Software provided a sample data set for Victoria Airport in CityGML format derived from an Apple IMDF data set. This data was used for initial development and testing of participants' individual components prior to the TIEs. This data was modified by Safe Software to include the Public Safety ADE. It includes PublicSafetyRooms, PublicSafetyDoors and Hatch features.

The overall workflow for the Indoor Pilot depended on generating good quality CityGML data outputs from point cloud data. Because it took a couple of months to accomplish this, having an interim dataset in CityGML to work with effectively saved those teams developing downstream components - that depended on CityGML as input - from waiting a couple of months to start development. Having a dataset to apply the draft ADE also helped with PS ADE design and vetting.

Some features contained in Victoria Airport data resulted in interesting anomalies such as the circular room in the center with pedestrian walkways around the perimeter. This proved challenging to calculate routing because it was difficult to automate the navigable spaces of nonplanar geometries. Since there was no floor in the open center, it was also difficult to cut the room into spaces. This attempt demonstrates that more advanced work could be done to improve identification of interior sub-spacing which is not always separated by floors, walls, and ceilings.

For more information on the Victoria Airport dataset, or to access the source IMDF, see: https://knowledge.safe.com/articles/76176/scenario-victoria-airport-esri-geodatabase-to-imdf.html



Property	Value
Feature Type	ps_Room
Coordinate System	EPSG:3857
Dimension	
Number of Vertices	2052
Min Extents	-13740205.56947103, 6214058.3078
Max Extents	-13740170.604018971, 6214087.99
Attributes (11)	
citygml_feature_role (encoded: UTF-16LE)	cityObjectMember
citygml_level_of_detail(0) (encode...	4
citygml_target_uri (encoded: UTF-16LE)	http://www.citygml.org/ade/public
fme_geometry (string)	fme_aggregate
fme_type (string)	fme_surface
gml_id (encoded: UTF-16LE)	ps_room_gml_130db361-1da5-402
gml_parent_id (encoded: UTF-16LE)	fme-gen-e3f77534-1b22-4c65-958
Name (encoded: UTF-16LE)	NONPUBLIC
room_public_safety{0}.public_safety_ty...	UtilityRoom
Tag (encoded: UTF-16LE)	nonpublic
xml_type (string)	xml_surface
IFMEMultiSurface (1 Part)	
Name (encoded: UTF-16LE)	lod4MultiSurface
Geometry Traits (3)	
citygml_level_of_detail (undefined type)	4
citygml_lod_name (undefined type)	lod4MultiSurface
gml_geometry (undefined type)	MultiSurface

victoriaAirport_v02_pubSafetyADE_3857 [CITYGML] - ps_Room

	gml_id	gml_parent_id	citygml_target_uri	citygml_feature_role	citygml_feature_role_attr_name	citygml_feature_role_attr_val	gml_description	gml_name	citygml_creationDate	cit
1	ps_room_gml_b...	fme-gen-e3f7753...	http://www.citygml.or...	cityObjectMember	<missing>	<missing>	<missing>	<missing>	<missing>	<m

Figure 6. Victoria Airport CityGML with Public Safety ADE in FME Data Inspector (Safe Software)

5.2.3. CityGML - Korea University Central Plaza

This CityGML data was provided by Pusan National University through conversion of the original manually cleansed point cloud into IndoorGML and then into CityGML. While the data may be lossy compared to the original data set, it provided a basis for early attempts for participants to develop the components.

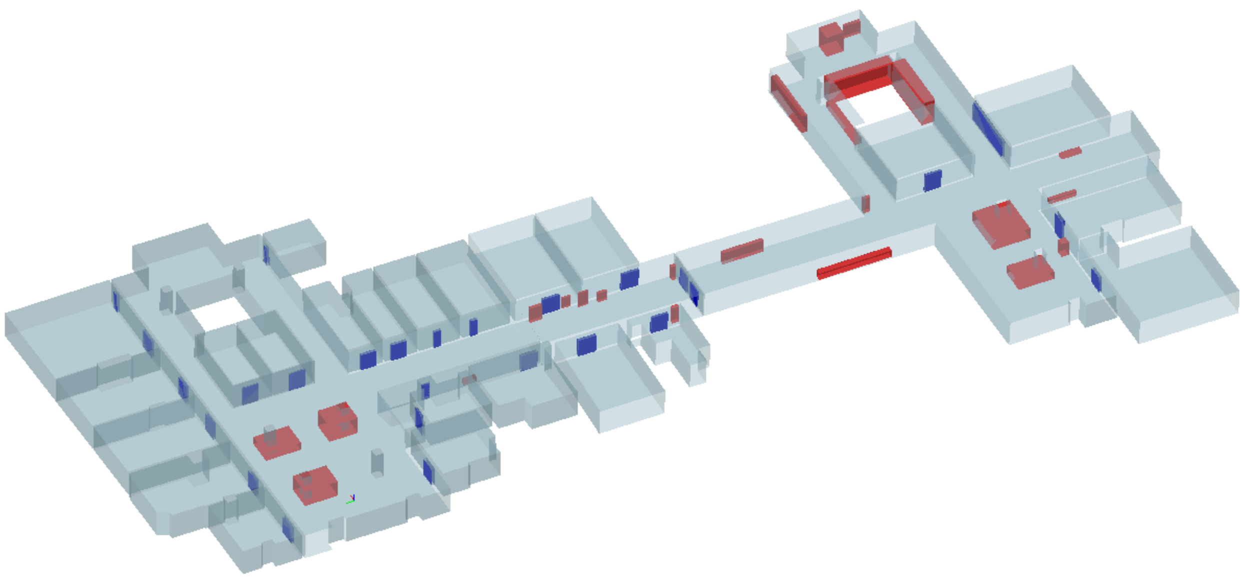


Figure 7. Korea University, Central Plaza CityGML

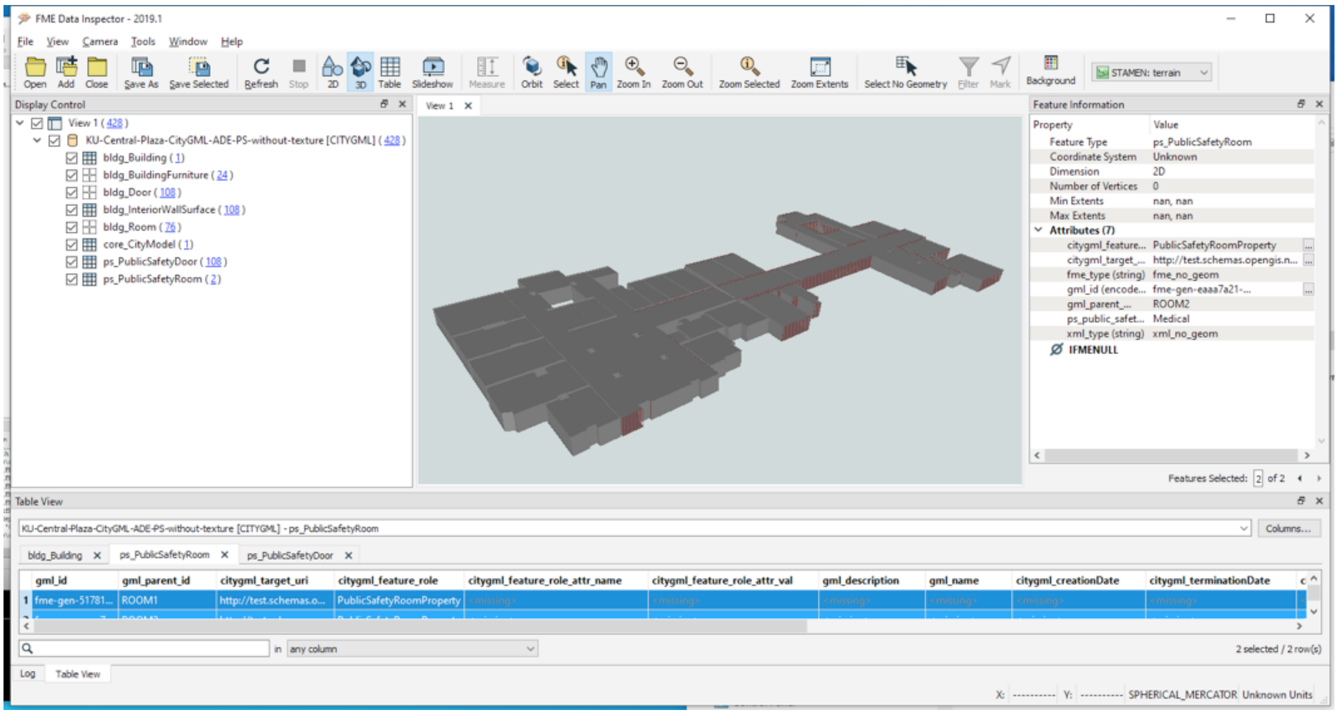
1. Name: KU-Central-Plaza (Central Plaza, Korea University)
2. Source: Converted from IndoorGML Public Safety Extension to CityGML by Hale, a GML based

open source conversion tool: https://www.wetransform.to/products/halestudio/)

3. Editing: None
4. Author: Pusan National University

5.2.4. CityGML - Korea University Central Plaza with Public Safety ADE

This data is the same as the previous CityGML with additional Public Safety ADE enriched in the data. The data is shown in Figure 8 displayed using FME. Note the 'ps_' namespace prefix for the public safety related properties and attributes. Also note the public safety feature types - in this case: PublicSafetyDoor and PublicSafetyRoom.


FME Data Inspector - 2019.1
File View Camera Tools Window Help
Open Add Close Save As Save Selected Refresh Stop 2D 3D Table Slideshow Measure Orbit Select Pan Zoom In Zoom Out Zoom Selected Zoom Extents Select No Geometry Filter Mark Background
STAMEN: terrain
Display Control
View 1
Feature Information
Table View
Log Table View

Display Control:

- View 1 (428)
  - KU-Central-Plaza-CityGML-ADE-PS-without-texture [CITYGML] (428)
    - bldg_Building (1)
    - bldg_BuildingFurniture (24)
    - bldg_Door (108)
    - bldg_InteriorWallSurface (108)
    - bldg_Room (76)
    - core_CityModel (1)
    - ps_PublicSafetyDoor (108)
    - ps_PublicSafetyRoom (2)

Feature Information:

Property	Value
Feature Type	ps_PublicSafetyRoom
Coordinate System	Unknown
Dimension	2D
Number of Vertices	0
Min Extents	nan, nan
Max Extents	nan, nan
Attributes (7)	
citygml_feature...	PublicSafetyRoomProperty
citygml_target_...	http://test.schemas.opengis.n...
fme_type (string)	fme_no_geom
gml_id (encode...	fme-gen-eaaa7a21-...
gml_parent_...	ROOM2
ps_public_safet...	Medical
xml_type (string)	xml_no_geom
IFMENULL	

Features Selected: 2 of 2

Table View: KU-Central-Plaza-CityGML-ADE-PS-without-texture [CITYGML] - ps_PublicSafetyRoom

Columns...

bldg_Building ps_PublicSafetyRoom ps_PublicSafetyDoor

	gml_id	gml_parent_id	citygml_target_uri	citygml_feature_role	citygml_feature_role_attr_name	citygml_feature_role_attr_val	gml_description	gml_name	citygml_creationDate	citygml_terminationDate
1	fme-gen-51781...	ROOM1	http://test.schemas.o...	PublicSafetyRoomProperty	<missing>	<missing>	<missing>	<missing>	<missing>	<missing>

in any column

2 selected / 2 row(s)

X: ---------- Y: ---------- SPHERICAL_MERCATOR Unknown Units

Figure 8. KU Central Plaza CityGML with PS ADE as displayed by FME showing public safety (ps_) feature types and attribution.

1. Name: KU-Central-Plaza-CityGML-ADE-PS.gml
2. Source: Pusan National University, Rendered by FME Data Inspector
3. Editing: None
4. Author: Pusan National University

5.2.5. IndoorGML - Korea University Central Plaza

This data is the intermediate data converted by Pusan National University from the point cloud data.

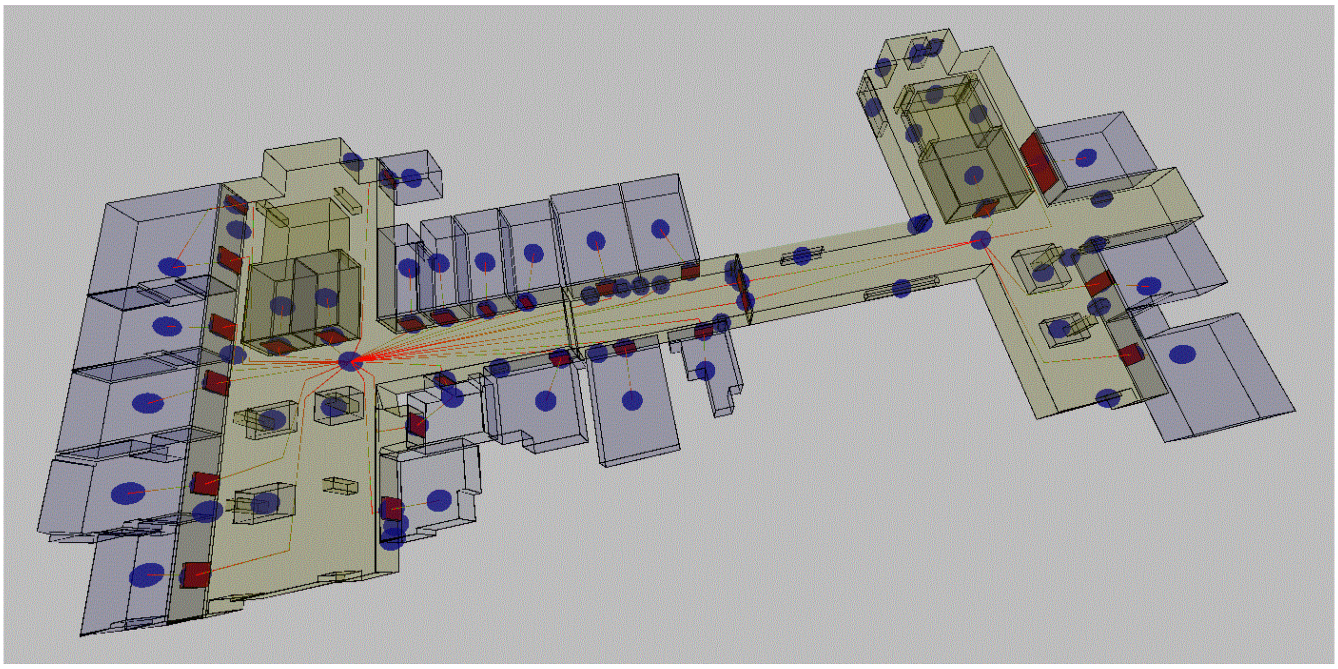


Figure 9. Korea University, Central Plaza IndoorGML

1. Name: KU-Central-Plaza (Central Square, Korea University)
2. Source:
    1. Converted to Mesh from Point Cloud
    2. Surface Extraction by TM2IN
    3. Cleaning and Editing for missing part by TICA and export to IndoorGML (TM2IN and TICA are developed by PNU and available at https://github.com/stemlab)
3. Editing: Editing missing sections (e.g., doors) and occluded rooms
4. Author: Pusan National University

5.2.6. Data Cleansing

During point cloud collection, it was discovered that many undesired artifacts appear in the point cloud data due to noise from LiDAR scans. LiDAR is a powerful tool for building scanning, but it has issues with transparent glass surfaces such as windows and doors. The laser beams pass through the glass surfaces, and the point cloud captures outdoor features through the glass such as trees. Reflective surfaces such as mirrors and marble flooring create additional artifacts in the point cloud. Also, during a LiDAR scan, moving features such as pedestrians and vehicles will create artifacts in the point cloud. These artifacts can often interfere with the calculation of indoor spaces and the location of walls and doors. Therefore, data cleansing is a necessary step to ensure reliable data for navigation.

The point cloud is composed of two categories; architectural components such as walls, ceilings, and floors, and non-architectural components such as chairs, tables, and other types of furniture in indoor space as shown by Figure 10.

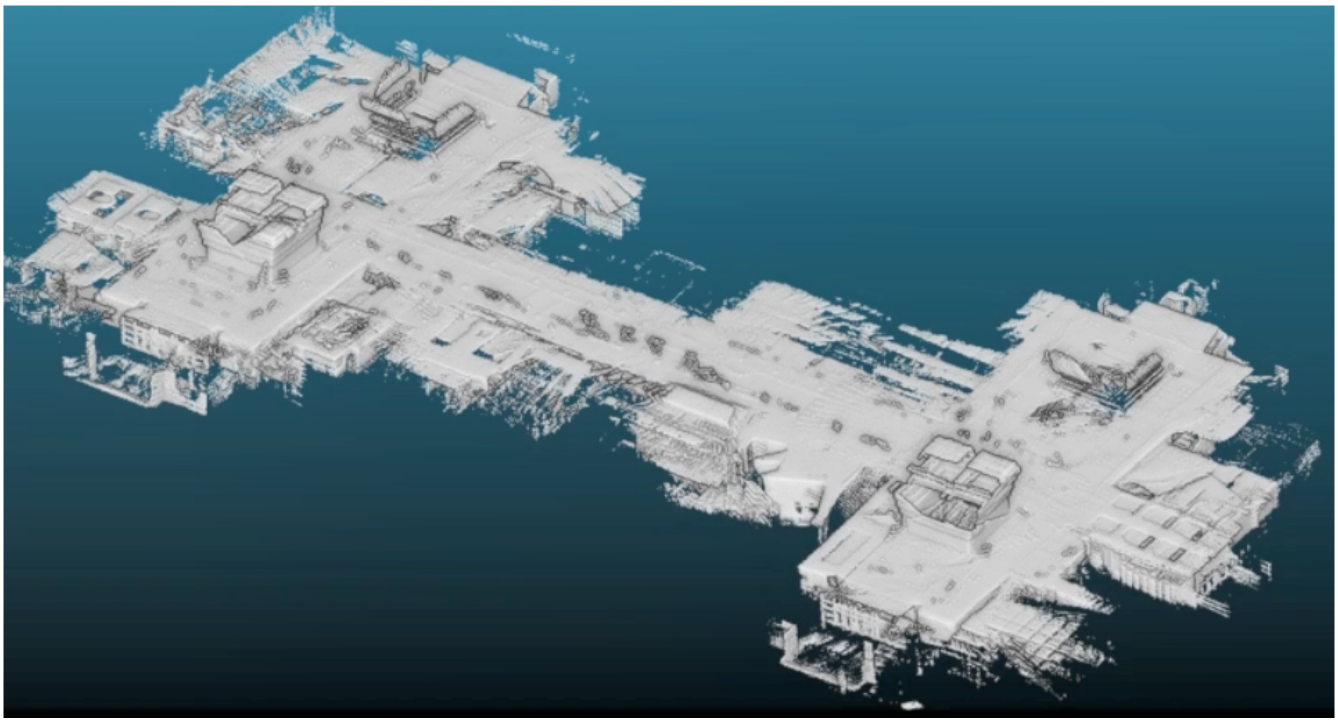


Figure 10. Overview of the point cloud data for Central Plaza at Korea University

The point cloud for architectural components is converted to CityGML and IndoorGML data at later steps, while the point cloud for non-architectural components is converted into non-navigable spaces of IndoorGML. Non-navigable spaces are practically obstacles to pedestrians and simplified into MBBs (Minimum Bounding Box). Together with navigable spaces, they are very useful in deriving indoor navigation networks. Examples of non-navigable spaces are shown in Figure 11 in red.

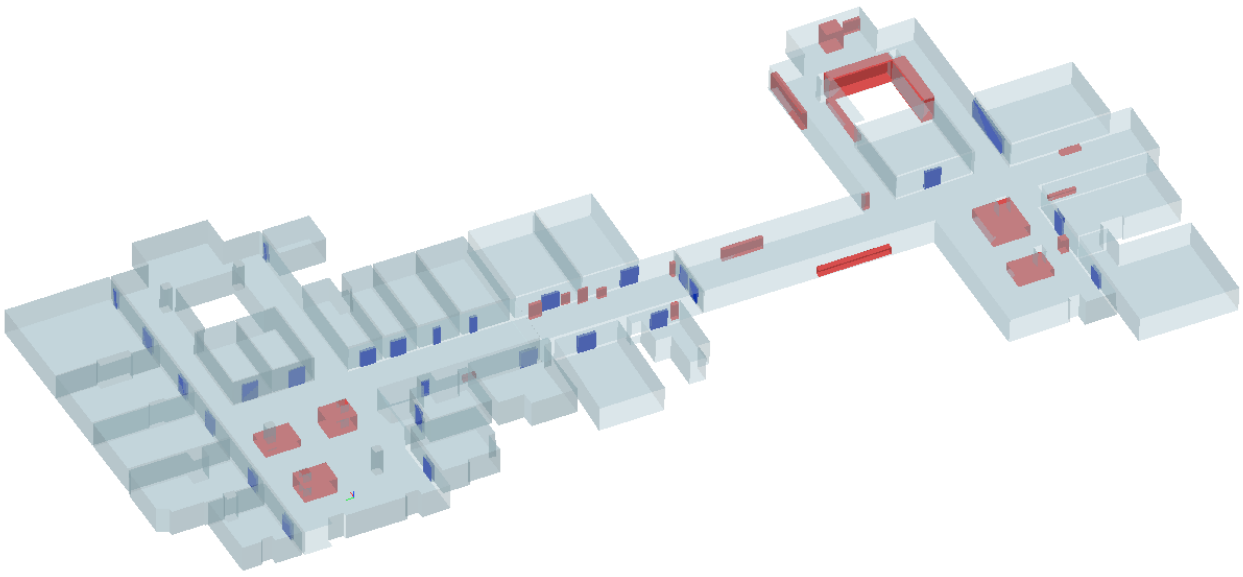


Figure 11. Non-Navigable Spaces of Central Plaza at Korea University

During the generation of point cloud data, the team encountered several issues. First, point clouds created where there is transparent glass leads to noisy data. For example, point clouds of outdoor spaces also contain objects detected through transparent windows (see Figure 12). There is a similar issue due to reflective material such as marble floors (see Figure 13).

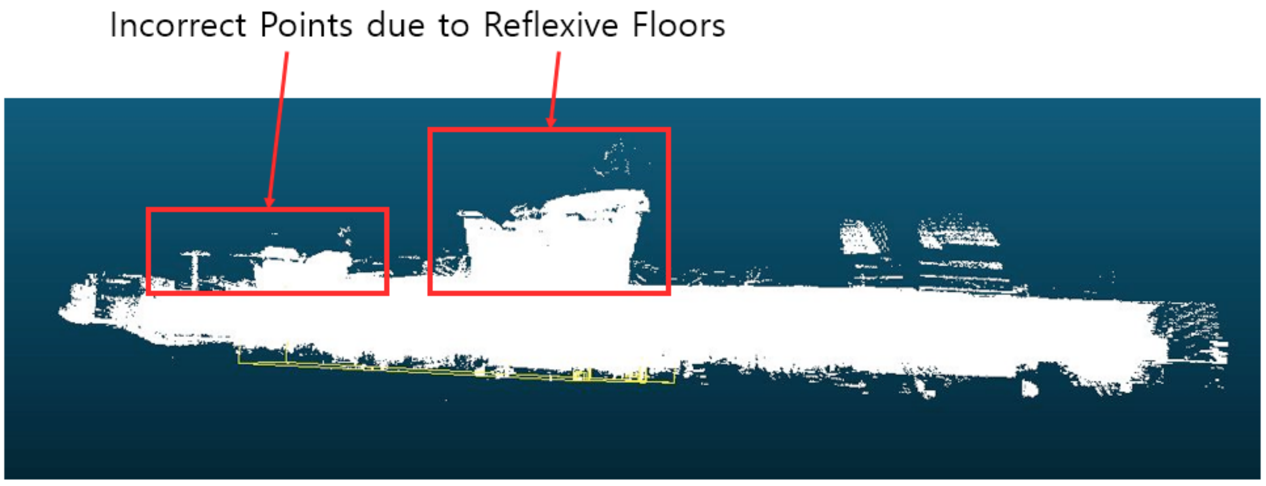


Figure 12. Example of Noisy Point Cloud Data due to Transparent Glasses

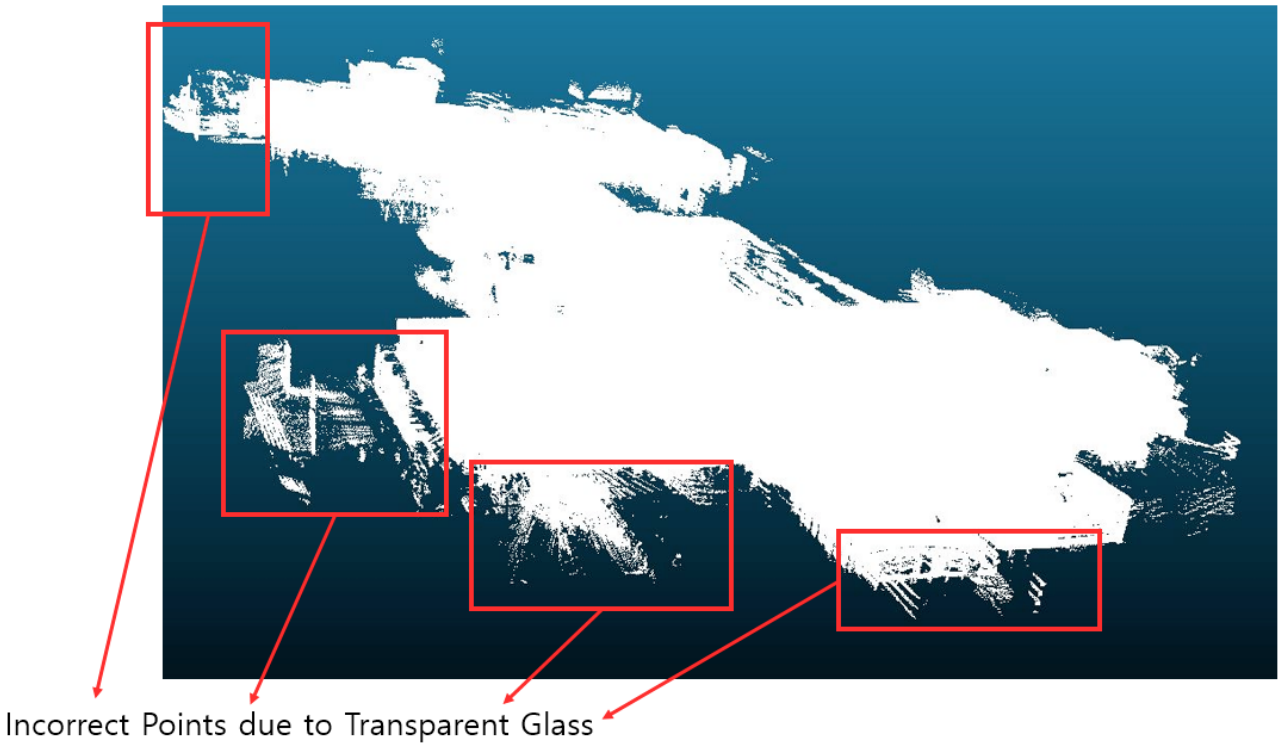


Figure 13. Example of Noisy Point Cloud Data due to Reflective Materials

Secondly, it is also difficult to produce accurately geo-referenced images of oblique objects such as oblique pillars. Thirdly, the coverage of point cloud data is rather incomplete due to occlusive areas. For example, we could not acquire the point cloud data in the rooms where the doors were locked.

Some automated processes can be used for partial data cleansing, but additional manual cleansing is still needed. For example, Cloud Compare is an automation tool that can be used to remove outlier points from a point cloud. Any remaining outlier points within the point cloud which were not caught by Cloud Compare must still be manually cleansed. Non-architectural objects such as furniture create obstacles within the indoor space that also interfere with indoor space calculations. One automation method is to calculate the vertical alignment of geometry such as walls to floors and ceilings, and then remove any objects or artifacts which are not part of the vertical or horizontal planes. This method may not be usable in some cases such as curved walls or dome-shaped ceilings, etc. In the sample data for Hana Square, there were some objects such as pillars which were not exactly vertical (see Figure 14). Since this architecture does not fit the assumptions of the automation method, the software was unable to distinguish the architectural from non-architectural components.

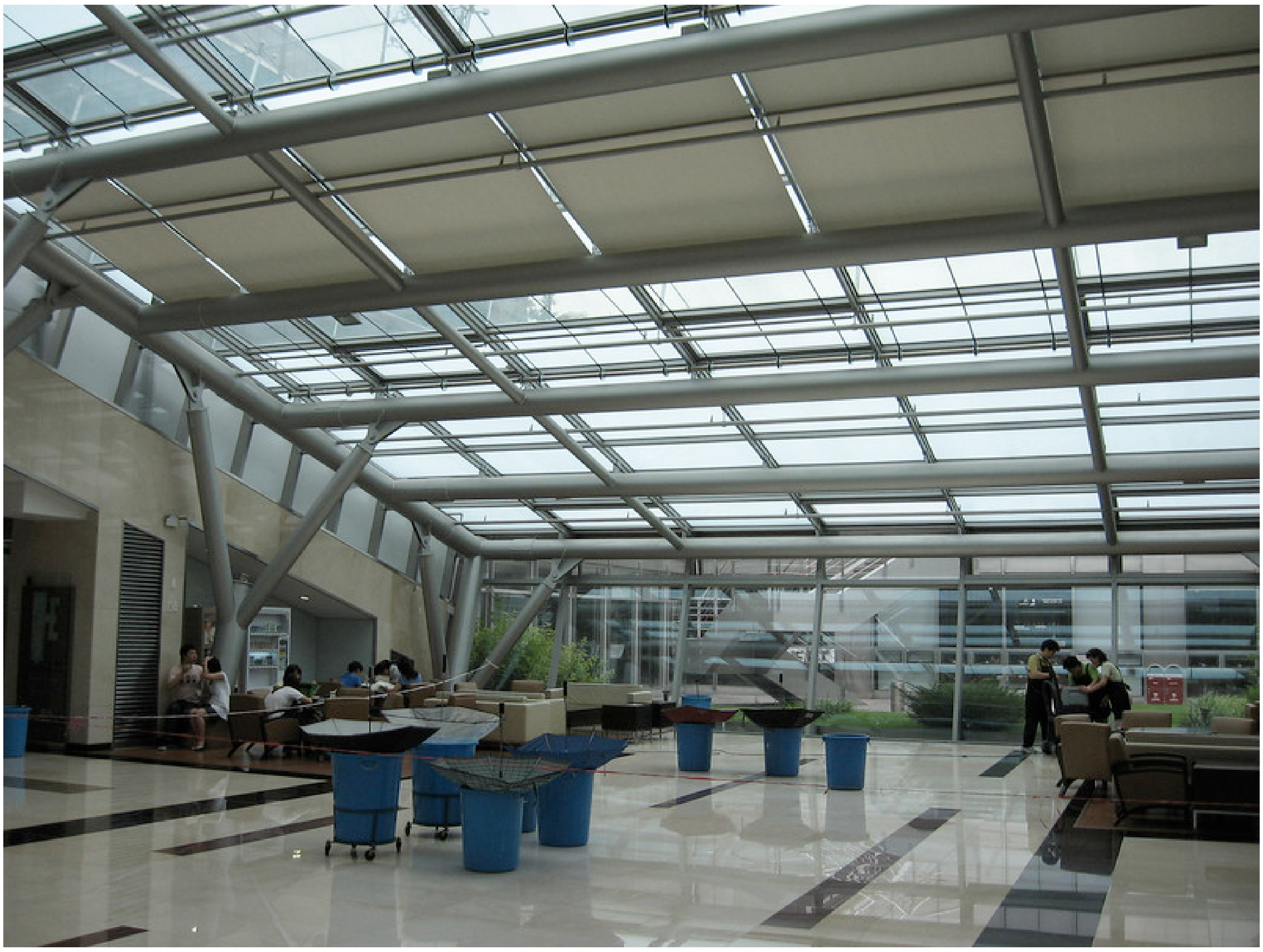


Figure 14. Korea University, Hana Square Architecture

An observation was made regarding how point cloud data is collected. In one data set, a Simultaneous Localization and Mapping (SLAM) robot was used to collect point clouds with a fixed height sensor. Upon inspection of the point cloud data, it was discovered that during the LiDAR scan, any data collected near the same level as the fixed position of the LiDAR sensor of the robot (i.e., the result is near the same linear level as the "eye" of the LiDAR sensor) results in a reduced output. This can result in poor data quality of certain features such as stairs near the same height of the sensor. One way to avoid this issue is to use a handheld SLAM sensor which can be varied in height during collection to ensure the best coverage of different angles of the environment.

5.2.7. Public Safety Data

The Public Safety enrichment process extends CityGML with the Application Domain Extension (ADE): Public Safety ADE schema version 3.1.2. At the time of this report this is published as ADE-PublicSafety.xsd. The development of the Public Safety ADE schema is the focus of the Public Safety ADE Engineering Report (OGC 19-032). Public Safety features are taken from the NAPSG Public Safety library of features. (https://www.napsgfoundation.org). The result was a new CityGML Public Safety ADE that adds five new feature types or elements to support public safety applications, namely: **PublicSafetyDoor**, **PublicSafetyWindow**, **PublicSafetyHatch**,

PublicSafetyIntBuildingInstallation, and **PublicSafetyRoom**. The ADE also defines a set of attributes or properties for each new feature type, along with supporting code lists to predefine lists of acceptable values for certain fields such as door type, opening type etc.

Considerable effort was required to test and validate the new application schemas as they evolved and ensure that the extension of CityGML was properly aligned with the CityGML specification, UML and schemas, and ensure that they supported the public safety applications as needed. One key part of this process was enriching actual CityGML datasets with new PS ADE elements and validating the results against the ADE schemas. As mentioned in Section 5.1.3, “IndoorGML Data”, the Public Safety Extension was created to support the transference of the Public Safety ADE from CityGML to IndoorGML.

5.3. Sponsor Provided Data

The data provided by the sponsor was from a concurrent project, which created an external dependency for this Pilot. The source of this point cloud data was the Hancock County, Mississippi Emergency Management Agency ("Hancock County"), which received an award under the NIST Public Safety Innovation Accelerator Program Point Cloud City program (https://www.nist.gov/news-events/news/2018/10/pscr-awards-over-750k-psiap-point-cloud-city). Data was provided by NVision Solutions Inc, which supplied geospatial software licenses and workstations needed to analyze data and create work products (https://www.nist.gov/ctl/pscr/hancock-county-point-cloud-city). The lessons-learned from the Section 5.2, “Participant Provided Data” were applied to the Hancock data set including the enrichment of Public Safety ADE to the CityGML data and conversion to Public Safety Extension in IndoorGML.

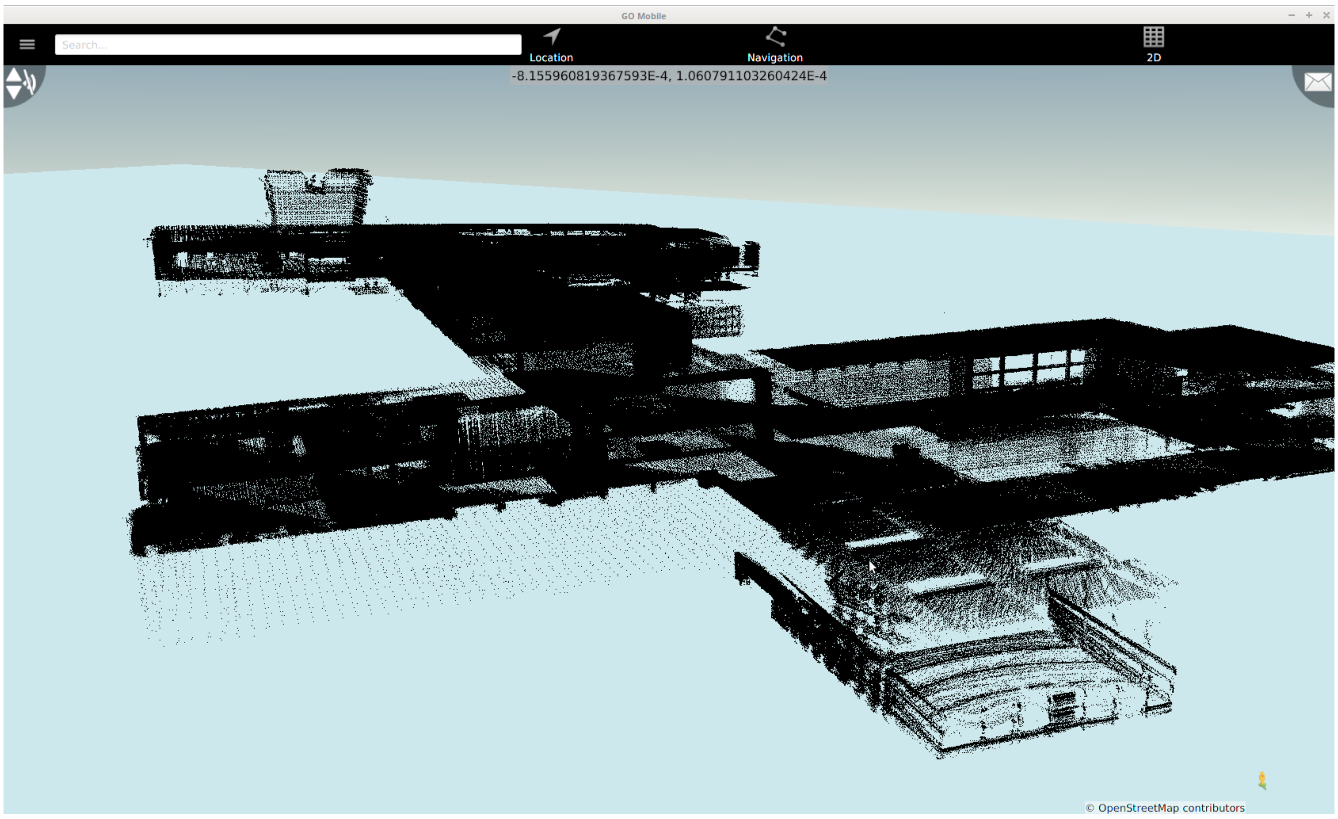

GO Mobile
Search...
Location
Navigation
2D
-8.155960819367593E-4, 1.060791103260424E-4
© OpenStreetMap contributors

Figure 15. Hancock County, Mississippi Emergency Management Agency

1. Name: Hancock County, Mississippi Emergency Management Agency

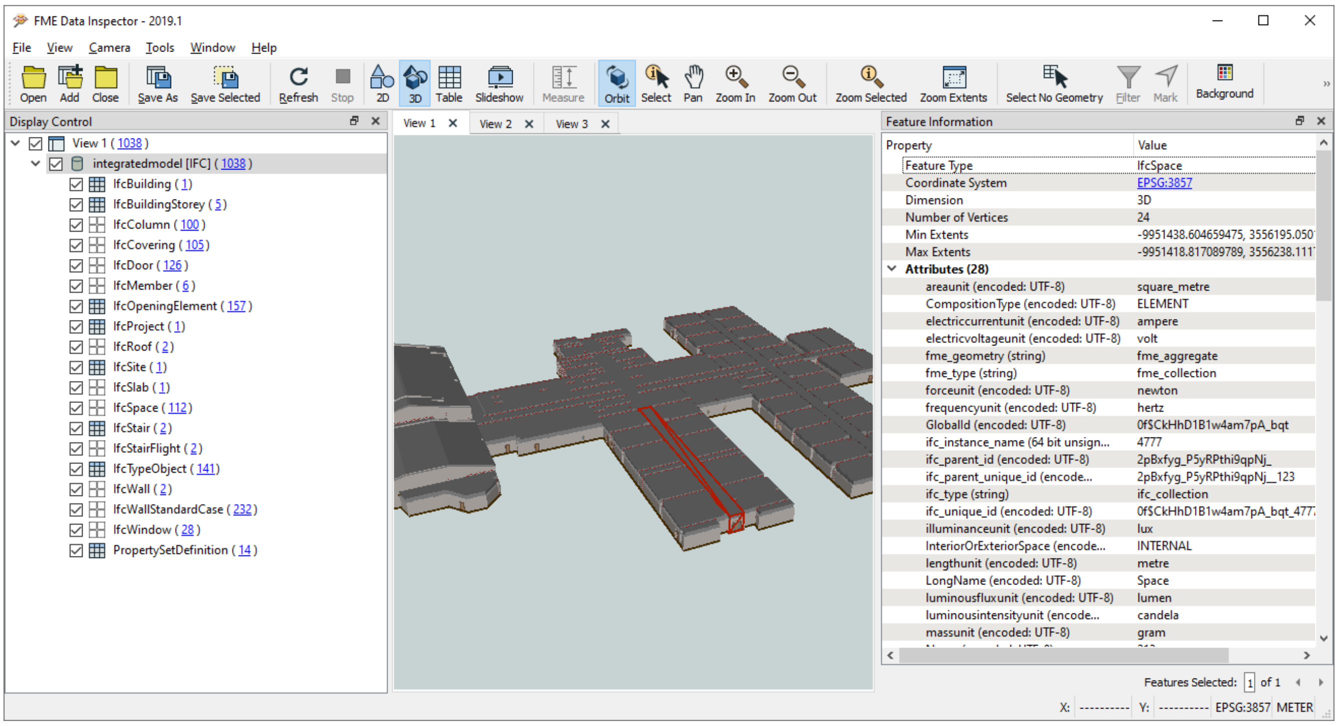
2. Source: GeoSlam Reb Zevo RT and Standard unit
3. Editing: Processed the data with GeoSlam processing software (GeoSlam Hub). No cleansing of the data.
4. Author: NVisions Solutions, Inc.

5.3.1. Conversion to IFC

Industry Foundation Class (IFC) data was generated during the conversion process from the Hancock point cloud data. IFC data generated from the Hancock County point cloud was first filtered and cleaned. A key part of the conversion process was computing surface normals and using them to filter out unwanted geometric complexity using the FME Spatial Extract-Transform-Load (ETL) tool. IFC complex solids were converted to CityGML and IndoorGML multi-surfaces and Boundary Representation (BRep) solids. One of the key challenges was geometry simplification. Walls, floors, and roofs in IFC are typically represented as volumes. For both CityGML and IndoorGML, it is generally preferable to have walls and ceilings as surfaces from which to build rooms. Typical manual approaches require an editor to identify which surfaces to keep (e.g., inner walls, ceilings, etc.) and which surfaces to eliminate (e.g., all edges, outer ceilings, etc.). In this case, Z surface normal values were used to filter surfaces such that $-0.1 < \text{surfaceZ} < 0.1$ produced vertical wall faces, $\text{surfaceZ} > 0.5$ produced the top of floors, and $\text{surfaceZ} < -0.5$ produced the bottom of the roof or ceiling.

Further experimentation is needed to refine these and other approaches. However, initial results suggest model-based approaches such as these hold promise for reducing the amount of manual effort required to take the raw surfaces produced from SLAM point clouds and refine them for use in generating structured building models in the form of IFC, IndoorGML or CityGML.

Figure 16 shows the Hancock County data converted from Point Cloud into IFC and displayed using FME.



View 1 (1038)
integratedmodel [IFC] (1038)
IfcBuilding (1)
IfcBuildingStorey (5)
IfcColumn (100)
IfcCovering (105)
IfcDoor (126)
IfcMember (6)
IfcOpeningElement (157)
IfcProject (1)
IfcRoof (2)
IfcSite (1)
IfcSlab (1)
IfcSpace (112)
IfcStair (2)
IfcStairFlight (2)
IfcTypeObject (141)
IfcWall (2)
IfcWallStandardCase (232)
IfcWindow (28)
PropertySetDefinition (14)

Property	Value
Feature Type	IfcSpace
Coordinate System	EPSG:3857
Dimension	3D
Number of Vertices	24
Min Extents	-9951438.604659475, 3556195.050
Max Extents	-9951418.817089789, 3556238.111
Attributes (28)	
areaunit (encoded: UTF-8)	square_metre
CompositionType (encoded: UTF-8)	ELEMENT
electriccurrentunit (encoded: UTF-8)	ampere
electricvoltageunit (encoded: UTF-8)	volt
fme_geometry (string)	fme_aggregate
fme_type (string)	fme_collection
forceunit (encoded: UTF-8)	newton
frequencyunit (encoded: UTF-8)	hertz
GlobalId (encoded: UTF-8)	0f$CkHhD1B1w4am7pA_bqt
ifc_instance_name (64 bit unsign...	4777
ifc_parent_id (encoded: UTF-8)	2pBxfyg_P5yRPthi9qpNj_
ifc_parent_unique_id (encode...	2pBxfyg_P5yRPthi9qpNj__123
ifc_type (string)	ifc_collection
ifc_unique_id (encoded: UTF-8)	0f$CkHhD1B1w4am7pA_bqt_477
illuminanceunit (encoded: UTF-8)	lux
InteriorOrExteriorSpace (encode...	INTERNAL
lengthunit (encoded: UTF-8)	metre
LongName (encoded: UTF-8)	Space
luminousfluxunit (encoded: UTF-8)	lumen
luminousintensityunit (encode...	candela
massunit (encoded: UTF-8)	gram

Features Selected: 1 of 1

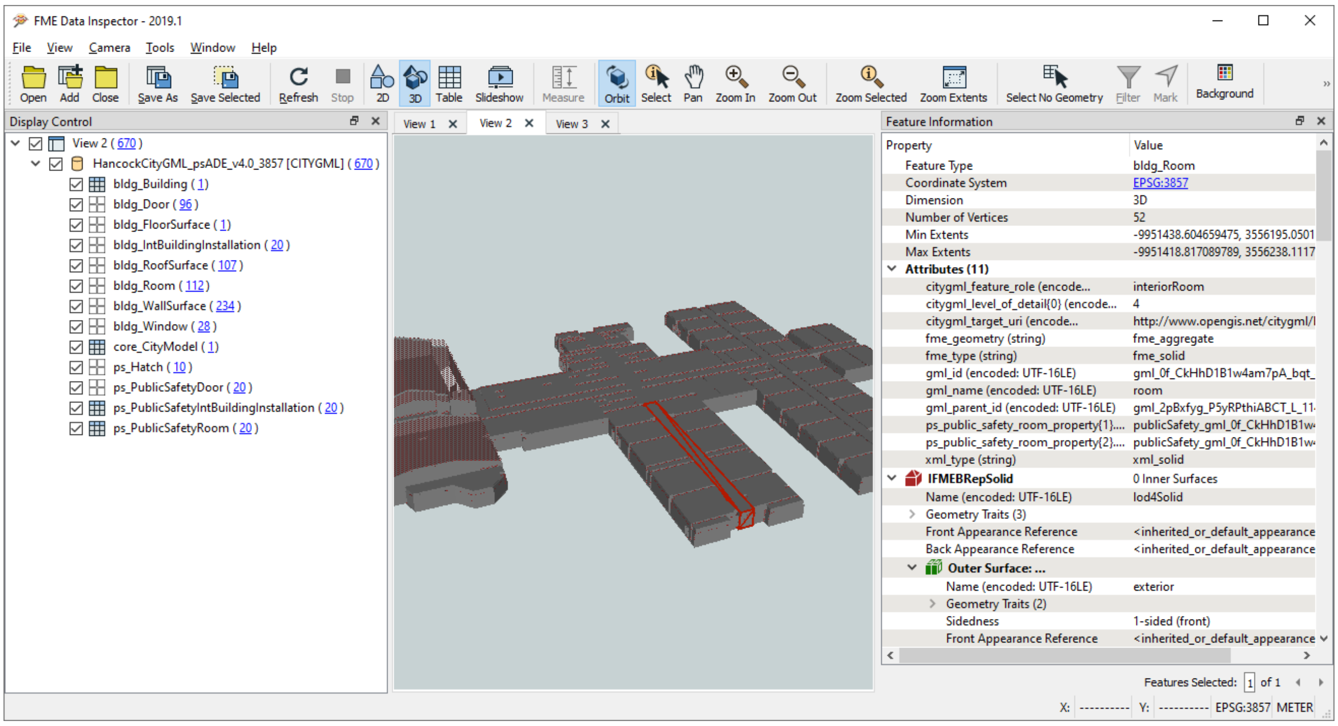
X: ---------- Y: ---------- EPSG:3857 METER

Figure 16. IFC source data from Building Modeler for Hancock County

1. Name: IFC Dataset Hancock County
2. Source: IFC source data from Building Modeler for Hancock County
3. Editing: Edited to clean data categorize features and create rooms
4. Author: Faramoon

5.3.2. IFC Conversion to CityGML with Public Safety ADE

Figure 17 shows the IFC data converted into CityGML and displayed in FME.


FME Data Inspector - 2019.1
File View Camera Tools Window Help
Open Add Close Save As Save Selected Refresh Stop 2D 3D Table Slideshow Measure Orbit Select Pan Zoom In Zoom Out Zoom Selected Zoom Extents Select No Geometry Filter Mark Background
Display Control
View 1 View 2 View 3
Feature Information

```
View 2 ( 670 )
  HancockCityGML_psADE_v4.0_3857 [CITYGML] ( 670 )
    bldg_Building ( 1 )
    bldg_Door ( 96 )
    bldg_FloorSurface ( 1 )
    bldg_IntBuildingInstallation ( 20 )
    bldg_RoofSurface ( 107 )
    bldg_Room ( 112 )
    bldg_WallSurface ( 234 )
    bldg_Window ( 28 )
    core_CityModel ( 1 )
    ps_Hatch ( 10 )
    ps_PublicSafetyDoor ( 20 )
    ps_PublicSafetyIntBuildingInstallation ( 20 )
    ps_PublicSafetyRoom ( 20 )
```

Property	Value
Feature Type	bldg_Room
Coordinate System	EPSG:3857
Dimension	3D
Number of Vertices	52
Min Extents	-9951438.604659475, 3556195.0501
Max Extents	-9951418.817089789, 3556238.1117
Attributes (11)	
citygml_feature_role (encode...	interiorRoom
citygml_level_of_detail{0} (encode...	4
citygml_target_uri (encode...	http://www.opengis.net/citygml/
fme_geometry (string)	fme_aggregate
fme_type (string)	fme_solid
gml_id (encoded: UTF-16LE)	gml_0f_CkHhD1B1w4am7pA_bqt_
gml_name (encoded: UTF-16LE)	room
gml_parent_id (encoded: UTF-16LE)	gml_2pBxfyg_P5yRPthiABCT_L_11
ps_public_safety_room_property{1}....	publicSafety_gml_0f_CkHhD1B1w
ps_public_safety_room_property{2}....	publicSafety_gml_0f_CkHhD1B1w
xml_type (string)	xml_solid
IFMEBRepSolid	0 Inner Surfaces
Name (encoded: UTF-16LE)	lod4Solid
Geometry Traits (3)	
Front Appearance Reference	<inherited_or_default_appearance
Back Appearance Reference	<inherited_or_default_appearance
Outer Surface: ...	
Name (encoded: UTF-16LE)	exterior
Geometry Traits (2)	
Sidedness	1-sided (front)
Front Appearance Reference	<inherited_or_default_appearance

Features Selected: 1 of 1

X: ---------- Y: ---------- EPSG:3857 METER

Figure 17. CityGML data converted from IFC data with FME

1. Name: Hancock County CityGML Public Safety ADE
2. Source: Converted from IFC to CityGML and then enriched with Public Safety ADE information using FME
3. Editing: Data Simplification, Geometry transformation, schema mapping, feature filtering, business rule application to autogenerate Public Safety ADE information
4. Author: Safe Software using FME converted from IFC provided by Faramoon

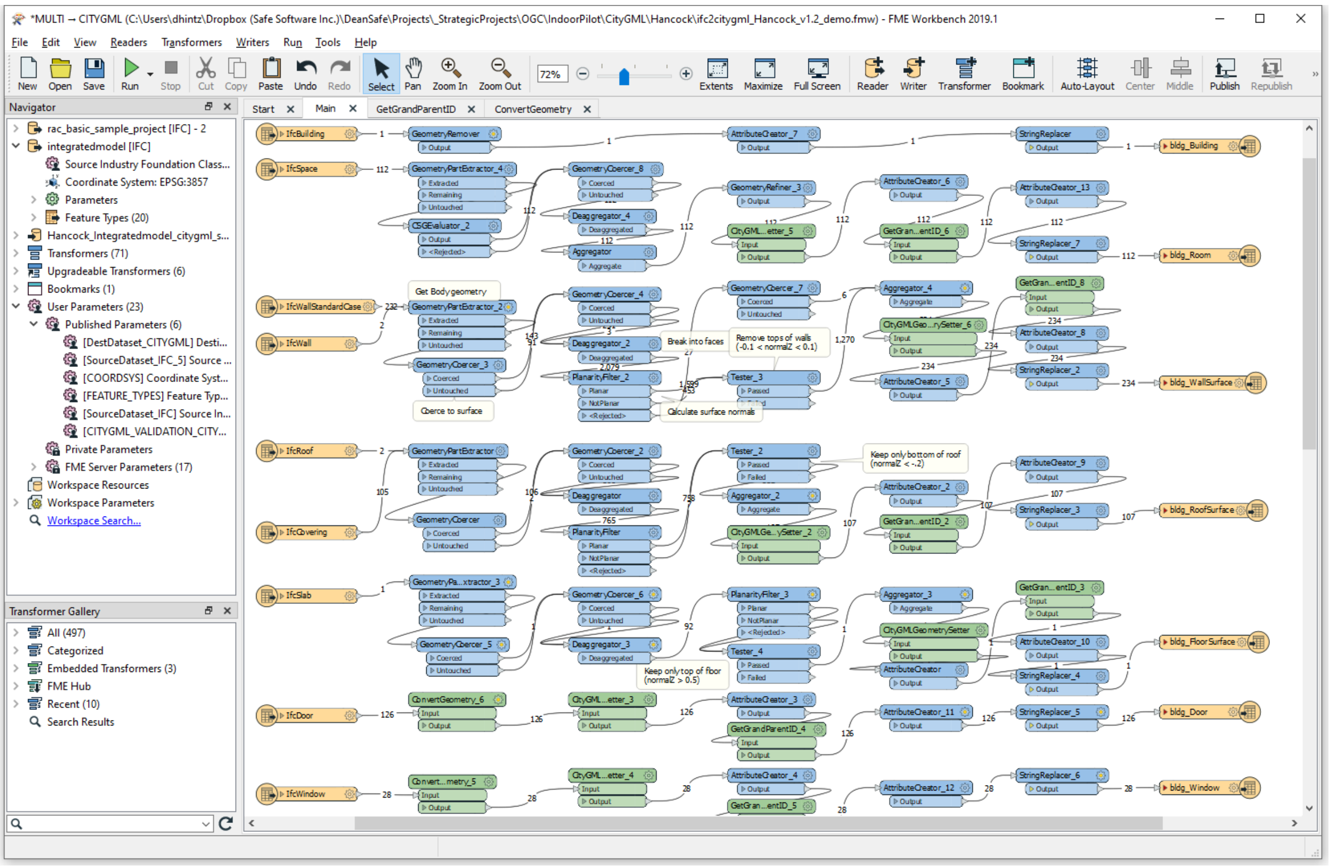
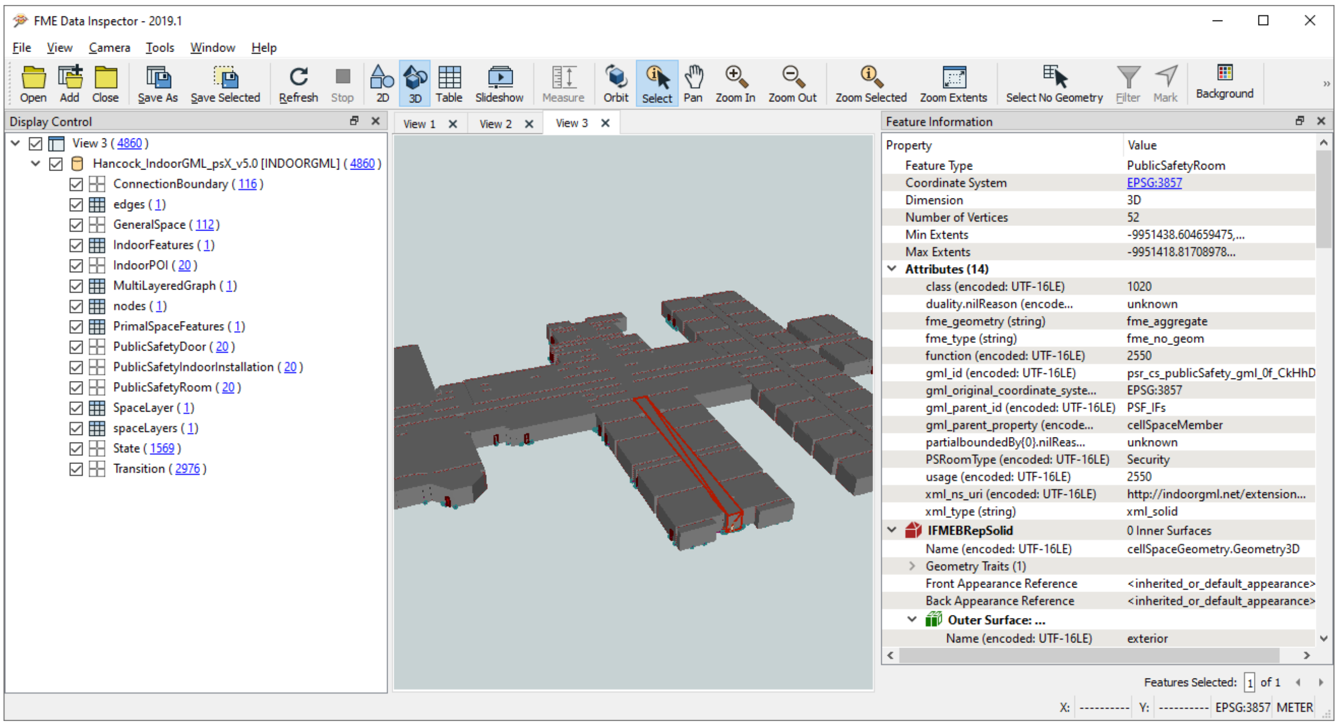


Figure 18. Spatial ETL (FME) process to convert Hancock County data from IFC to CityGML

5.3.3. CityGML Converted to IndoorGML with Public Safety Extension



Property	Value
Feature Type	PublicSafetyRoom
Coordinate System	EPSG:3857
Dimension	3D
Number of Vertices	52
Min Extents	-9951438.604659475,...
Max Extents	-9951418.81708978...
Attributes (14)	
class (encoded: UTF-16LE)	1020
duality.nilReason (encode...	unknown
fme_geometry (string)	fme_aggregate
fme_type (string)	fme_no_geom
function (encoded: UTF-16LE)	2550
gml_id (encoded: UTF-16LE)	psr_cs_publicSafety_gml_0f_CkHhD
gml_original_coordinate_syste...	EPSG:3857
gml_parent_id (encoded: UTF-16LE)	PSF_IFs
gml_parent_property (encode...	cellSpaceMember
partialboundedBy{0}.nilReas...	unknown
PSRoomType (encoded: UTF-16LE)	Security
usage (encoded: UTF-16LE)	2550
xml_ns_uri (encoded: UTF-16LE)	http://indoorgml.net/extension...
xml_type (string)	xml_solid
IFMEBRepSolid	0 Inner Surfaces
Name (encoded: UTF-16LE)	cellSpaceGeometry.Geometry3D
Geometry Traits (1)	
Front Appearance Reference	<inherited_or_default_appearance>
Back Appearance Reference	<inherited_or_default_appearance>
Outer Surface: ...	
Name (encoded: UTF-16LE)	exterior

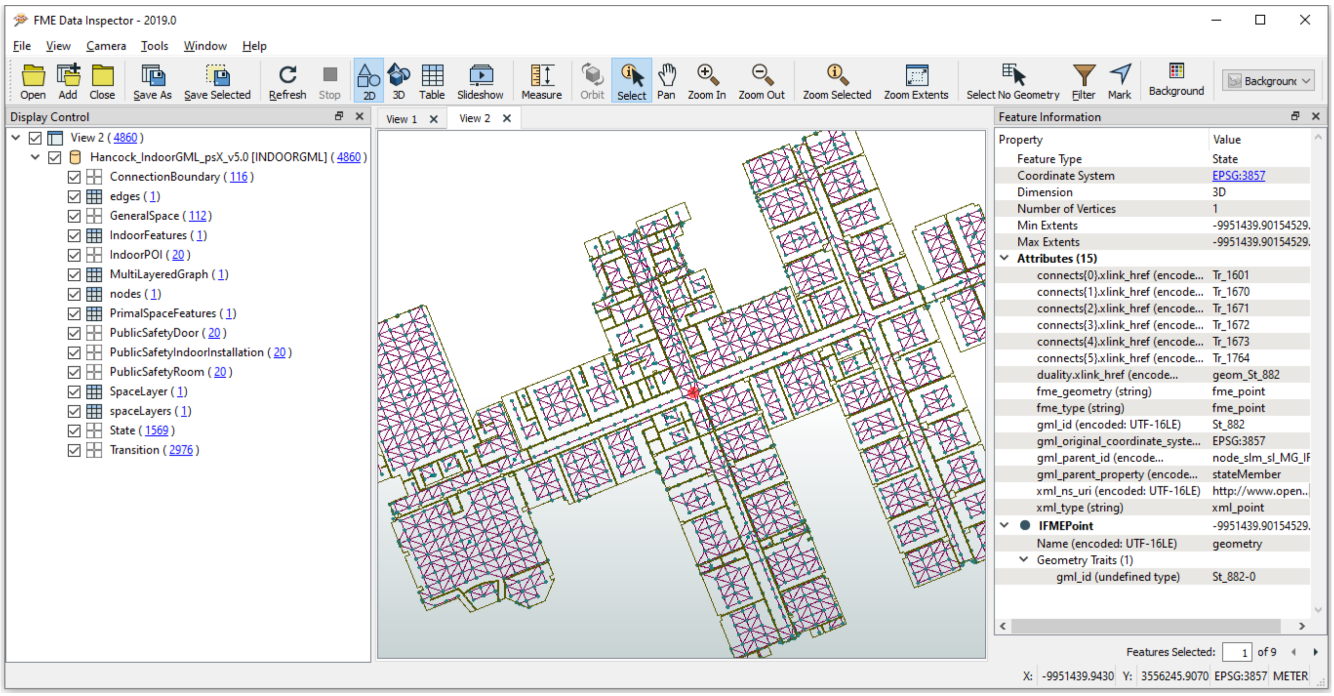
Figure 19. Hancock County IndoorGML

1. Name: Hancock County IndoorGML Public Safety Extension
2. Source: Hancock County CityGML Public Safety ADE
3. Editing: Schema mapping CityGML feature types and attributes to IndoorGML. ID and

relationship generation to associate feature types as per IndoorGML schema and UML requirements. Navigation network generation (States and Transitions).

4. Author: Safe Software using FME to convert CityGML Public Safety ADE to IndoorGML Public Safety Extension

5.3.4. IndoorGML Navigation Network with Public Safety Extension


FME Data Inspector - 2019.0
File View Camera Tools Window Help
Open Add Close Save As Save Selected Refresh Stop 2D 3D Table Slideshow Measure Orbit Select Pan Zoom In Zoom Out Zoom Selected Zoom Extents Select No Geometry Filter Mark Background
Display Control
View 1
View 2
Feature Information

- View 2 (4860)
  - Hancock_IndoorGML_psX_v5.0 [INDOORGML] (4860)
    - ConnectionBoundary (116)
    - edges (1)
    - GeneralSpace (112)
    - IndoorFeatures (1)
    - IndoorPOI (20)
    - MultiLayeredGraph (1)
    - nodes (1)
    - PrimalSpaceFeatures (1)
    - PublicSafetyDoor (20)
    - PublicSafetyIndoorInstallation (20)
    - PublicSafetyRoom (20)
    - SpaceLayer (1)
    - spaceLayers (1)
    - State (1569)
    - Transition (2976)

Property	Value
Feature Type	State
Coordinate System	EPSG:3857
Dimension	3D
Number of Vertices	1
Min Extents	-9951439.90154529.
Max Extents	-9951439.90154529.
Attributes (15)	
connects{0}.xlink_href (encode...	Tr_1601
connects{1}.xlink_href (encode...	Tr_1670
connects{2}.xlink_href (encode...	Tr_1671
connects{3}.xlink_href (encode...	Tr_1672
connects{4}.xlink_href (encode...	Tr_1673
connects{5}.xlink_href (encode...	Tr_1764
duality.xlink_href (encode...	geom_St_882
fme_geometry (string)	fme_point
fme_type (string)	fme_point
gml_id (encoded: UTF-16LE)	St_882
gml_original_coordinate_syste...	EPSG:3857
gml_parent_id (encode...	node_slm_sl_MG_IF
gml_parent_property (encode...	stateMember
xml_ns_uri (encoded: UTF-16LE)	http://www.open...
xml_type (string)	xml_point
IFMEPoint	-9951439.90154529.
Name (encoded: UTF-16LE)	geometry
Geometry Traits (1)	
gml_id (undefined type)	St_882-0

Features Selected: 1 of 9

X: -9951439.9430 Y: 3556245.9070 EPSG:3857 METER

Figure 20. IndoorGML for Hancock County with MultilayeredGraph, States, and Transitions representing Navigation Network - 2D View

1. Name: Hancock County IndoorGML Public Safety Extension

2. Source: Hancock County CityGML Public Safety ADE

3. Editing: Schema mapping CityGML feature types and attributes to IndoorGML. ID and relationship generation to associate feature types as per IndoorGML schema and UML requirements. Navigation network generation (States and Transitions). Note that early network generation was not aligned with Hancock corridors yielding a network that produced more irregular paths.

4. Author: Safe Software using FME to convert CityGML Public Safety ADE to IndoorGML Public Safety Extension

Chapter 6. Building Modeler Service

As described in Section 5.1.2, “CityGML Data”, the **Building Modeler Service** application converts point cloud models and associated images into semantic 3D building models compliant with the most recent or stable version of CityGML (2.0). Models generated by this component are annotated with Public Safety Features from the CityGML PS ADE. The building modeler was developed as web service that is compliant with the OGC WPS 2.0 Interface Standard (OGC14-065), although this was not a strict requirement.

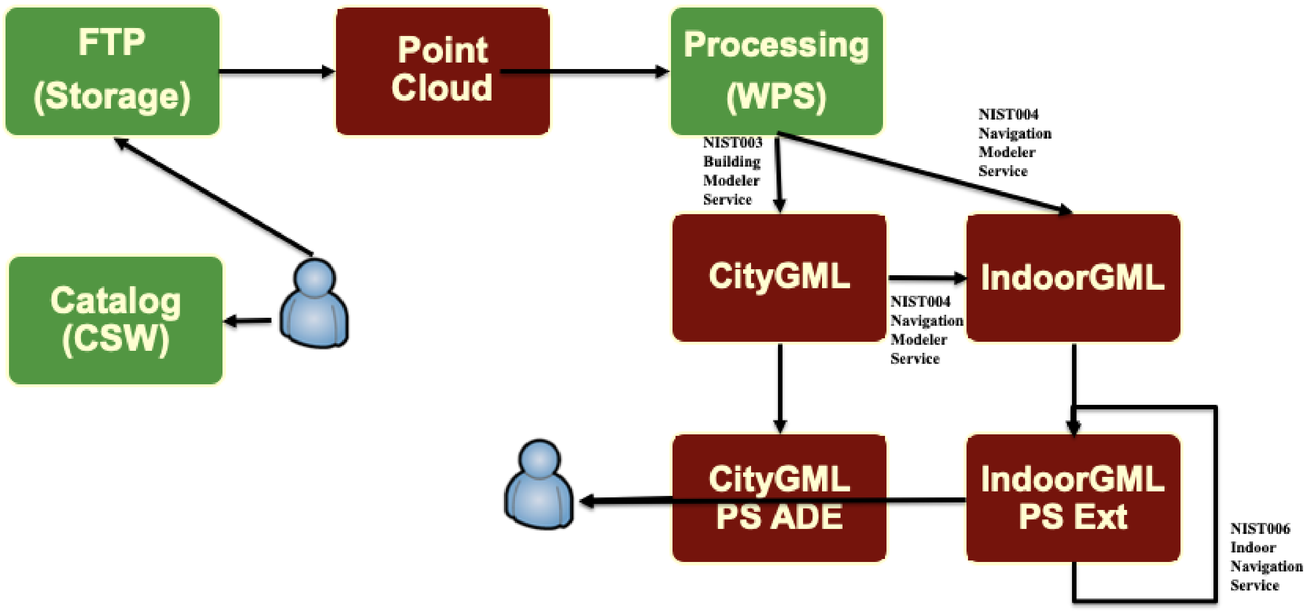

FTP (Storage)
Point Cloud
Processing (WPS)
NIST003 Building Modeler Service
NIST004 Navigation Modeler Service
Catalog (CSW)
CityGML
IndoorGML
NIST004 Navigation Modeler Service
CityGML PS ADE
IndoorGML PS Ext
NIST006 Indoor Navigation Service

Figure 21. Retrieving and Converting Point Cloud Data

6.1. Faramoon Building Modeler Service

Faramoon has designed and developed an automated technology that takes point clouds of building interiors and converts them into 3D models that conform to international standards without requiring any manual input. There are four stages including 1) structuring a point cloud, 2) filtering noises, 3) recognizing objects using deterministic and stochastic algorithms and methods, and 4) modeling indoor features in 3D.

Faramoon initially implemented a process using this methodology to maximize the value of the conversion process and increase productivity by removing the manual processes required to generate the CityGML. The process is purely based on the geometry of the point cloud without the use of the RGB data (i.e., images). The geometry is identified with attributes using semantics to describe ceilings, floors, walls, etc. These geometries are identified based on surfaces and perpendicularity to other surfaces to designate these spaces. Faramoon’s building modeler is independent of the data format provided. The initial LAZ data is first converted into PCD format. Following this, the PCD modeler executes the process flows described below.

6.1.1. Process Flow

Faramoon developed a WPS to convert point cloud files to CityGML files. A sequence diagram shown in Figure 22 describes each step in the process.

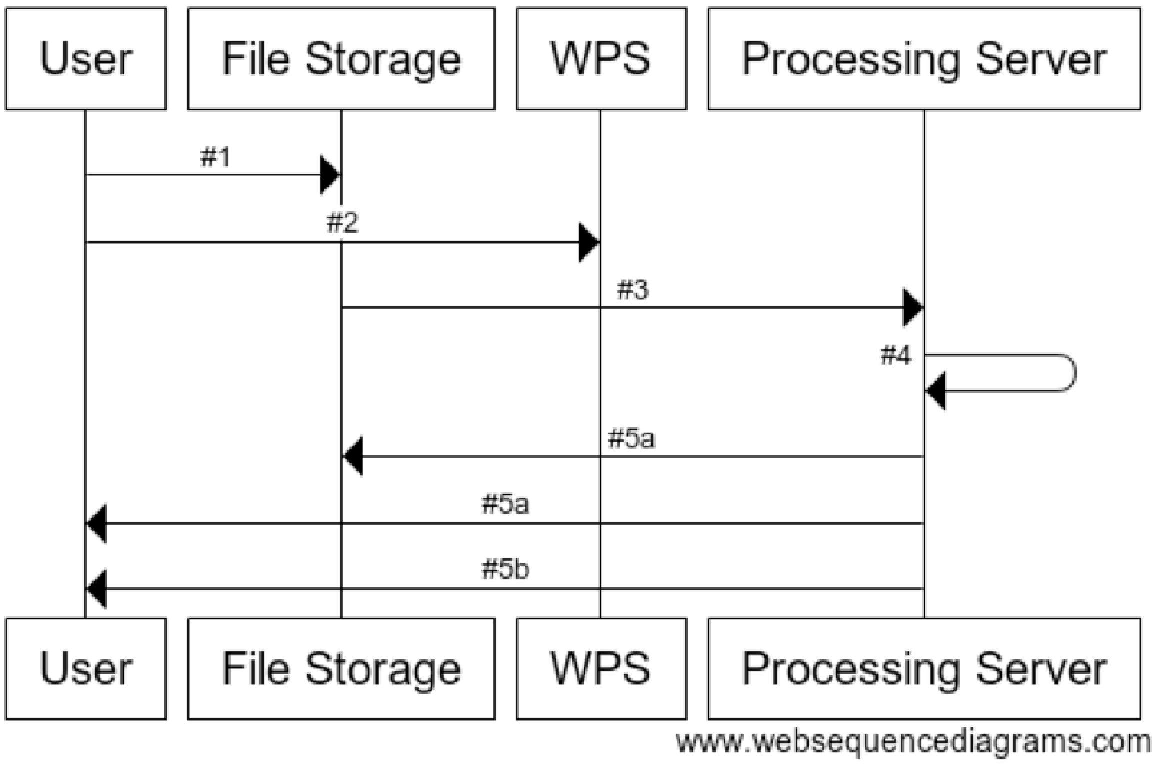


Figure 22. Faramoon Building Modeler Sequence Diagram

1. The user accesses the Faramoon File Storage server (SFTP) and uploads a point cloud in PCD file format.
2. The user makes an HTTP POST request to the WPS with the filename of the PCD file as the parameter.
3. The Processing Server downloads the PCD file from the File Storage server.
4. The Processing Server processes the file into a CityGML file with Faramoon's software.
   a. If the file was successfully processed into a CityGML file, the user receives a response from their HTTP POST request with a direct URL to the CityGML file. The processed CityGML file was uploaded back to the File Storage server.
   b. If the file was unsuccessfully processed into a CityGML file, the user receives a "Bad Request" response from their HTTP POST request.

Due to many challenges described in Faramoon's Section 6.1.2, "Lessons Learned", the WPS could not be fully utilized. Therefore, the resulting CityGML files were processed on a local computer. Figure 23 shows steps taken to model the point cloud datasets.

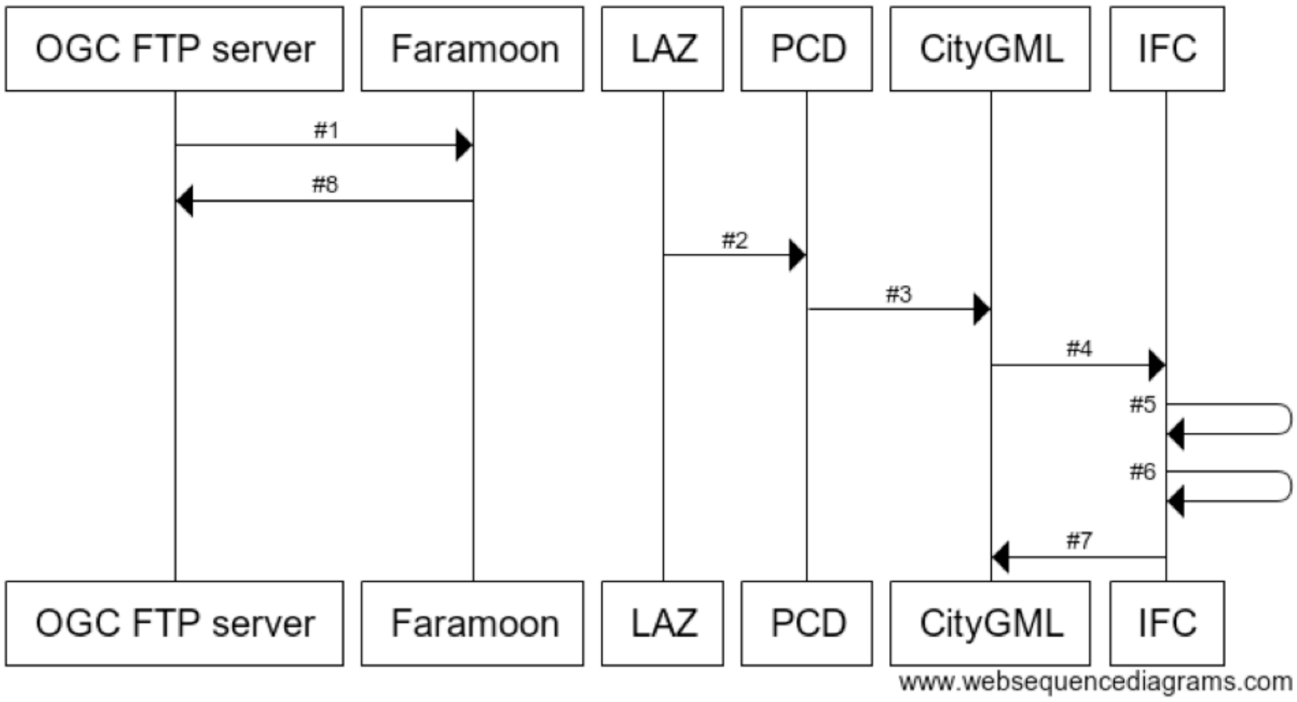


Figure 23. Faramoon Point Cloud Modeling

1. Faramoon downloads LAZ files from OGC FTP server.
2. Faramoon manually cleans and converts LAZ files into PCD file format.
3. PCD files are processed into CityGML files with Faramoon Software.
4. CityGML files are converted into IFC file format with Faramoon Software.
5. IFC files are manually modified with Autodesk Revit.
6. Modified IFC files are exported from Autodesk Revit.
7. Modified IFC files are converted to resulting CityGML files with a workbench with FME.
8. The resulting CityGML file was uploaded to the OGC FTP server.

6.1.2. Lessons Learned

In developing Faramoon's building modeler, the following challenges were identified:

- The provided point cloud data was in grayscale. This made it difficult to discern various entries and exits as well as other features in the point cloud. This also makes it more difficult to identify objects that could have been more easily distinguished using color (RGB) point clouds as opposed to only geometry (e.g., signage).
- While not part of the scope of this project, Faramoon attempted to identify some public safety features contained in point cloud data. Some of the public safety features (e.g., fire extinguishers, alarms, etc.) are difficult to identify due to the complexity of shape and size.
- Initially, Faramoon attempted to automate the conversion of Hancock County point cloud data. However, the point cloud data was very noisy, resulting in a data output with false walls, missing doors, and incorrect geometries, etc. Therefore, it was determined that the output data needed to be cleaned manually or enriched in order to be used for the demonstration. Initial attempts to use TICA proved to be difficult due to immature developmental status and lack of

existing documentation for the software. The automation tool from Faramoon can output either CityGML or IFC data (BIM standard). Since tools are limited for CityGML formats, Faramoon used Revit, a 3D BIM tool, to manually cleanse the output data in IFC format.

- The Hancock dataset was provided in six different point cloud files. The resulting CityGML file contained the processed results of all six point cloud files combined. This was challenging because the point clouds overlapped but did not accurately fit together. With trial and error, the modified IFC models were placed in several different combinations until an accurate representation of the building appeared.
- Georeferencing issues were discovered in the point cloud data because data was not geolocated. Therefore, to resolve this issue, a combination of Google Maps satellite imagery and QGIS was used. This was done by finding the location of the real building and positioning it along the corners of the Google Maps satellite imagery in QGIS, an open source GIS software, until the resulting CityGML model matched the satellite imagery.

6.2. GIS FCU Building Modeler Service

GIS FCU's building modeler service calls a WPS service to convert point cloud data into CityGML data. The service relied on the use of Machine Learning techniques through the use of the open source PointNet library. This software library uses recognition to identify and classify clusters of point clouds as objects and separate the objects into individual geometries. The process flow is shown in Figure 24.

6.2.1. Process Flow

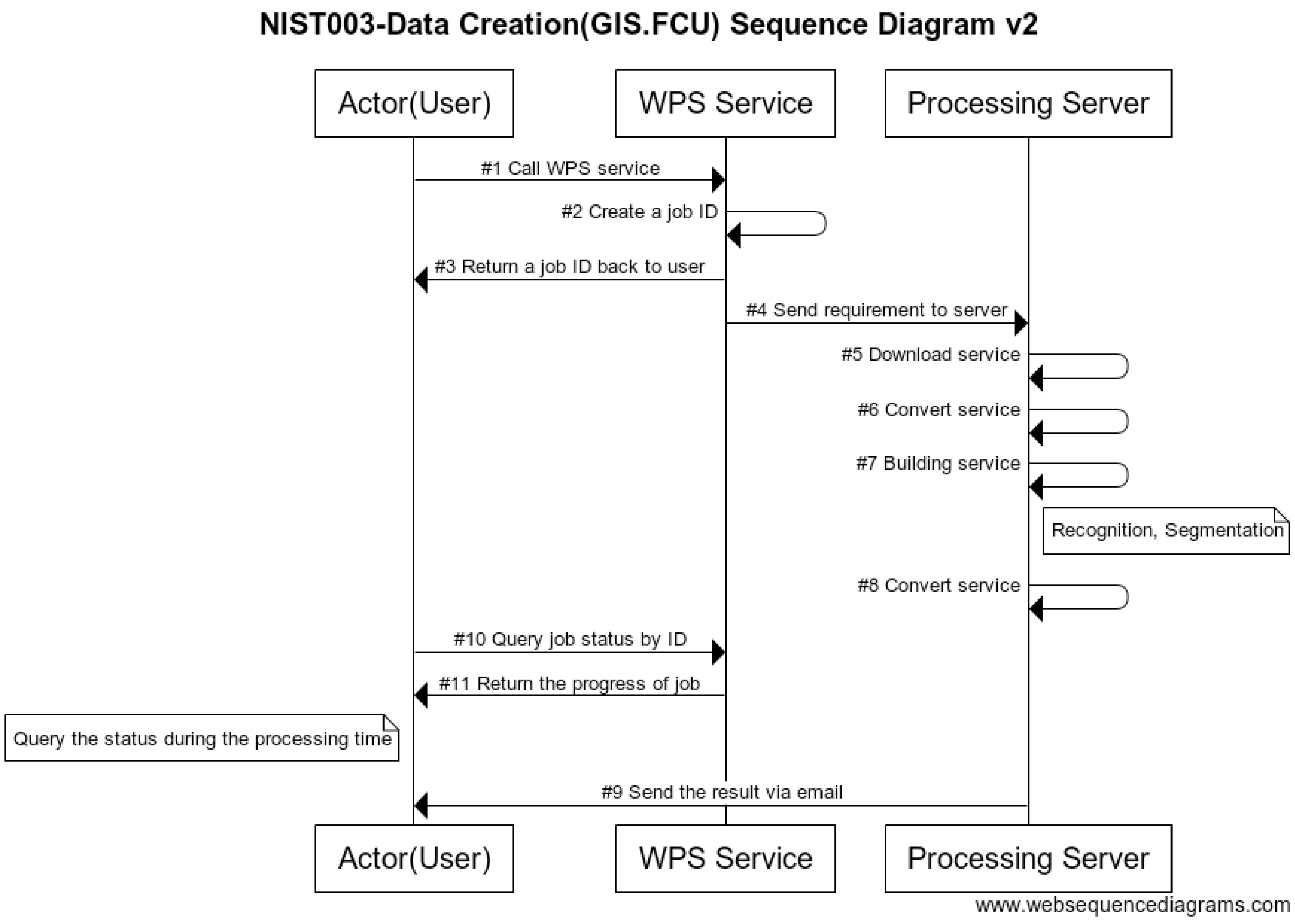

NIST003-Data Creation(GIS.FCU) Sequence Diagram v2
Actor(User)
WPS Service
Processing Server
#1 Call WPS service
#2 Create a job ID
#3 Return a job ID back to user
#4 Send requirement to server
#5 Download service
#6 Convert service
#7 Building service
Recognition, Segmentation
#8 Convert service
#10 Query job status by ID
#11 Return the progress of job
Query the status during the processing time
#9 Send the result via email
Actor(User)
WPS Service
Processing Server
www.websequencediagrams.com

Figure 24. GIS FCU Data Creation Sequence Diagram

1. The user calls the WPS service with a mail address and URL link to a LAS file
2. The WPS service creates a new job ID for this request
3. The WPS service returns the job ID to the user
4. The WPS service sends the requirements to a processing server with the mail address and URL
5. The download service downloads the LAS file using the associated URL
6. The convert service converts the LAS file to PCD format
7. The building service consists of 3 services, including a Recognition service, a Segmentation service, and a Conversion service.
    1. The Recognition service recognizes objects, walls, roof, floor and doors
    2. The Segmentation service segments objects into several parts points group
    3. The Conversion service converts the points group to a 3D surface which consists of objects of polygon type
8. The convert service converts the 3D surfaces into CityGML
9. The processing server sends the resulting URL of CityGML by email, or the user can use a job ID to query and retrieve the download URL
10. The user can query the job status by job ID during the processing time
11. The WPS service returns the status of this job. There are three types of statuses: Processing, Failed, and Finished.

The Point Cloud data is converted to CityGML with no manual intervention:

1. The WPS service downloads the point cloud dataset via the URL
2. It converts the LAS file to PCD format because PCD format is easier to manipulate. Then the file size is reduced by 100:1. This is a downsampling process which is considered lossy, but results in the reduction of the overall error rate.
3. The PointNet library (PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation https://arxiv.org/pdf/1612.00593) is used to reconstruct surfaces. PointNet is a C-based open source library which supports C++, Python, and Java with Deep Learning libraries to identify and recognize 3D objects within the PCD. GIS FCU encodes this in Python, and the libraries are listed below:
   a. The **Recognition Library** is used to identify and group objects based on clusters of point cloud points through a pattern recognition module. This includes 3D object recognition, hypothesis verification for 3D object recognition, and implicit shape model for calculating objects centers.
   b. The **Segmentation Library** segments point cloud data based on various algorithms.
   c. The **Surface Library** provides data smoothing and reconstructions using approximation algorithms.
4. Convert PCD file to SHP file to build 3D surface file.
5. Convert SHP file to GML file.
6. Build CityGML file from GML file.

6.2.2. Lessons Learned

Currently, PointNet is able to recognize some objects, but attempts to recognize boundaries and surfaces with straight polygons is very difficult.

Between the doors and the rooms, the spacing between the walls and the doors need to be highlighted. Attempts to use raw data resulted in a lot of noisy data. Downsampling the data by 10:1 showed some improvement with some noise. 100:1 was better, but 1000:1 was too lossy. The amount of reduction depends on the original raw data. Figure 25 shows the differences in the point clouds at various downsampling levels.

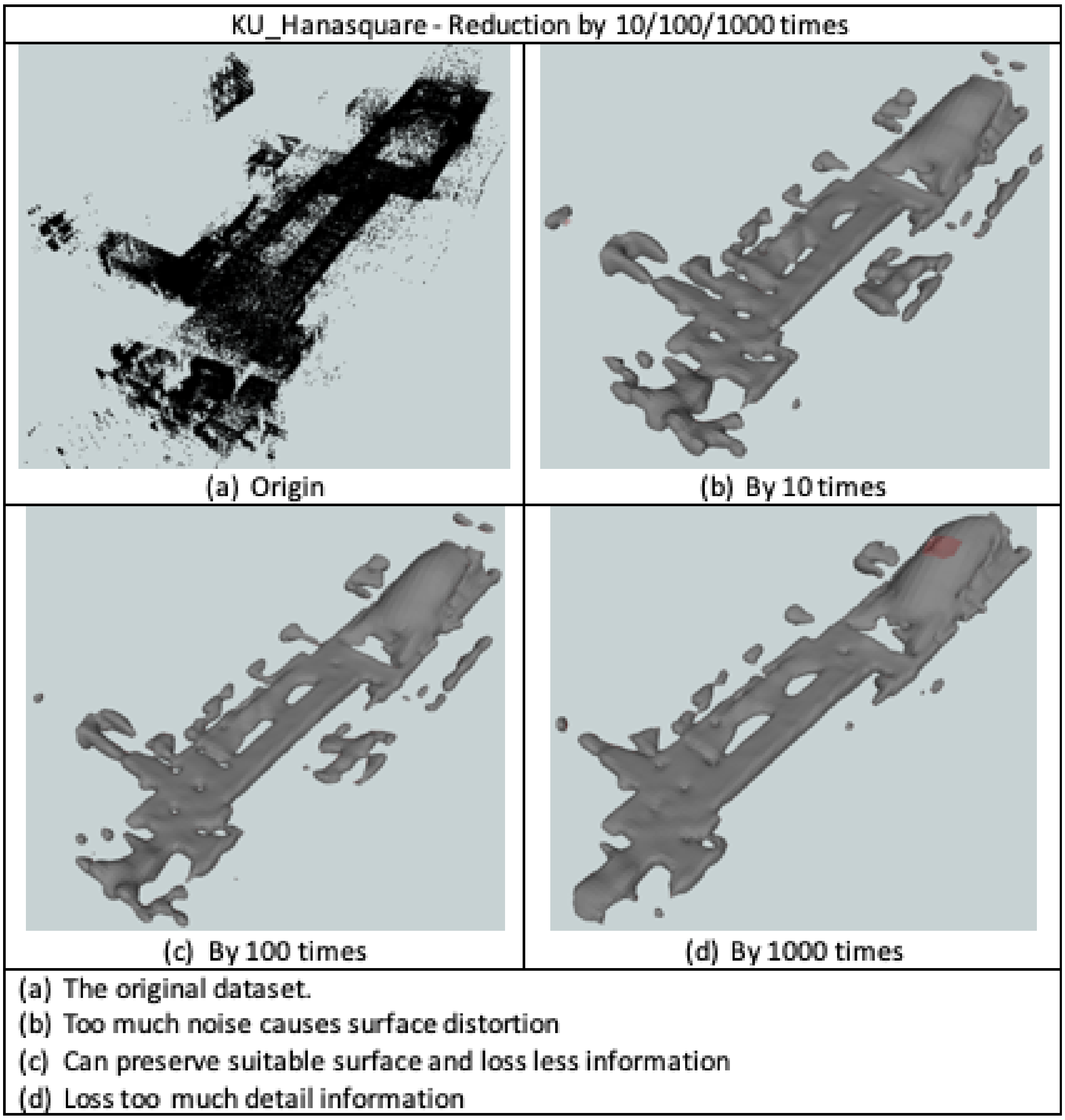

KU_Hanasquare - Reduction by 10/100/1000 times
(a) Origin
(b) By 10 times
(c) By 100 times
(d) By 1000 times
(a) The original dataset.
(b) Too much noise causes surface distortion
(c) Can preserve suitable surface and loss less information
(d) Loss too much detail information

Figure 25. GIS FCU Hana Square data reduction by 10X, 100X, and 1000X

PointNet uses a deep learning neural network algorithm, and it needs many more examples of polygon objects such as windows, doors, walls, etc. to teach the network algorithm to better recognize these objects in point cloud. For example, with enough point clouds of trees, deep learning can be used to filter out trees from indoor space recognition. Additional examples are needed to improve the accuracy of the algorithm.

Chapter 7. Navigation Modeler Service

As described in Section 5.1.3, “IndoorGML Data”, the purpose of the **Navigation Modeler Service** is to convert CityGML from the Building Modeling Service into IndoorGML for use by the **Indoor Navigation Service**. This service queries the CSW for a given building ID, retrieves the building data in CityGML via a URL, converts it to IndoorGML and then posts the results back to the CSW. The following describes the sub-components, process steps, data format, and service interfaces for this service component.

7.1. Pusan National University Navigation Modeler

The Pusan National University Navigation Modeler service operates differently than the prescribed data conversion flow. The workflow for PNU’s navigation modeler is to convert from point cloud data directly into IndoorGML, and then convert IndoorGML into CityGML. The conversion process conducted by Pusan National University is documented step-by-step as follows:

7.1.1. Conversion from Point Cloud data into IndoorGML

The point cloud data is captured. Figure 26 shows the undesired artifacts from unclean point cloud data.

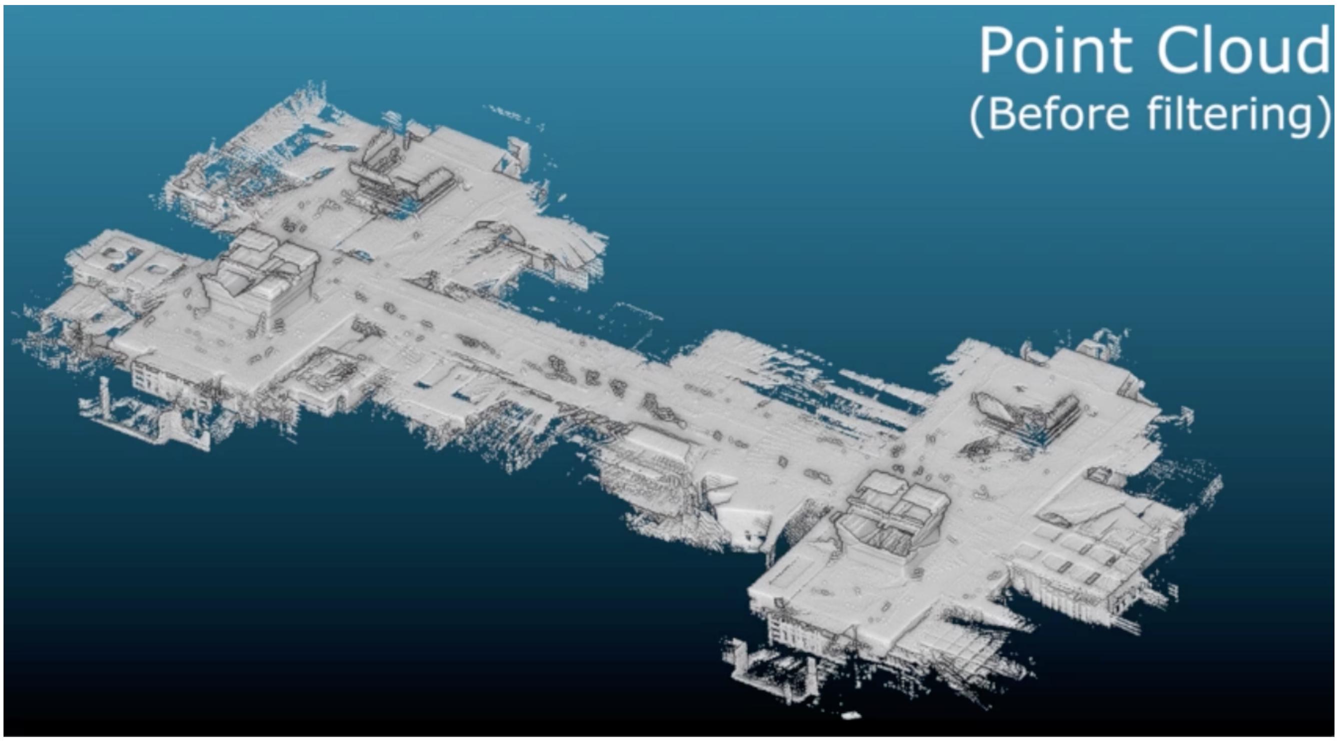


Figure 26. Point Cloud Data before data cleansing

The data is cleansed following a data cleansing process.

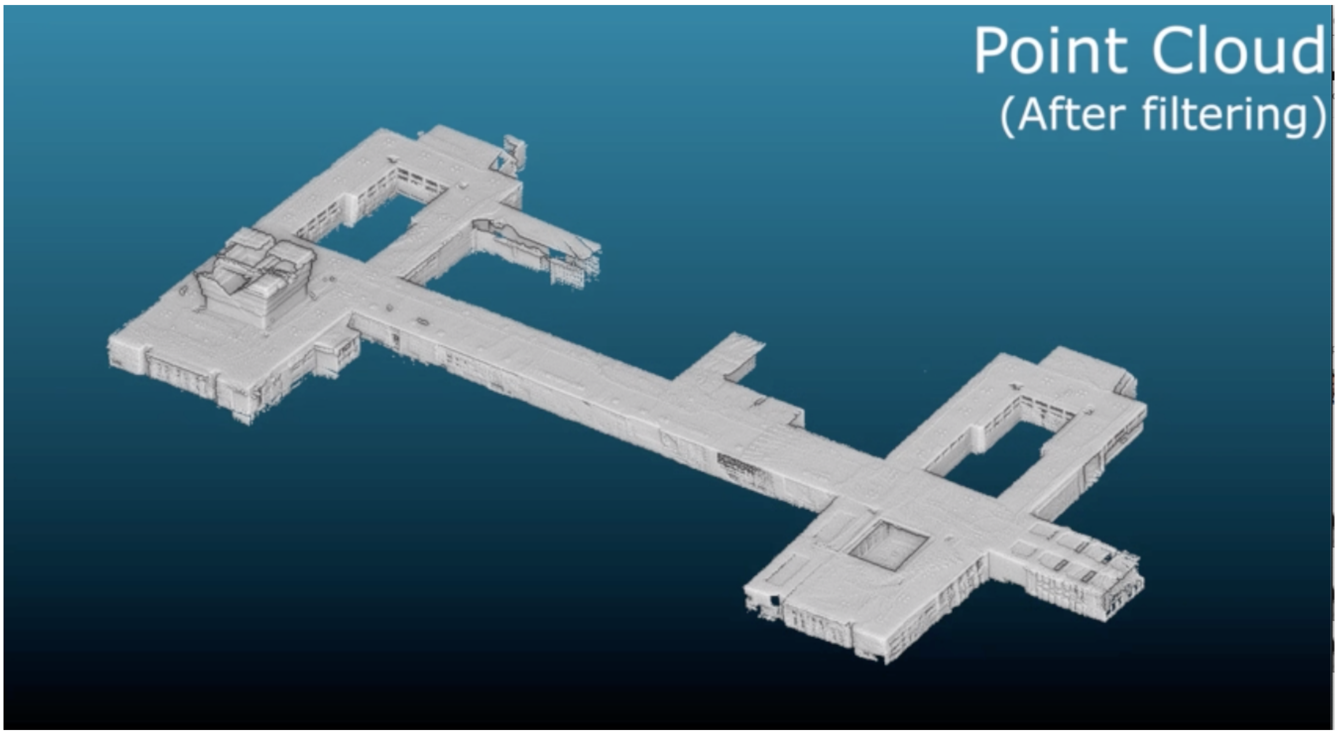


Figure 27. Point Cloud Data after data cleansing

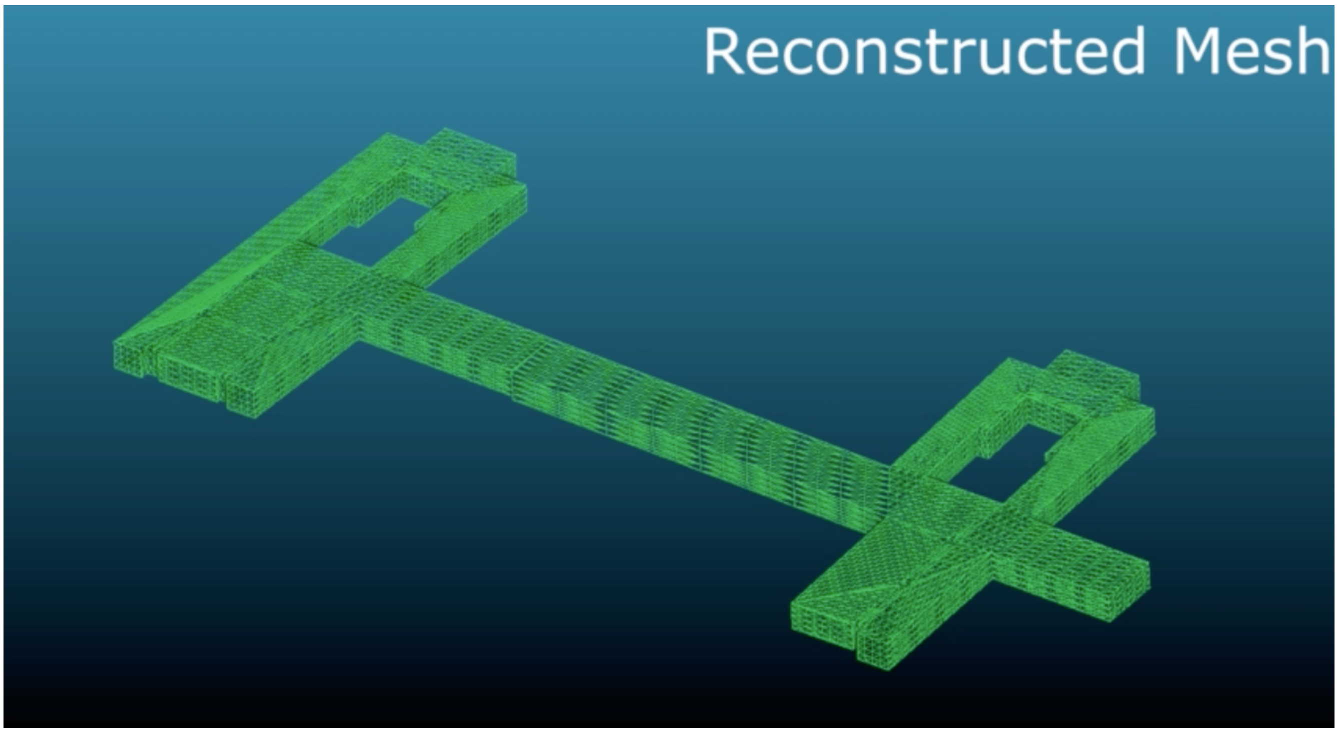


Figure 28. Reconstructed Mesh from Cleansed Point Cloud data

A reconstructed mesh is created from the cleansed point cloud data.

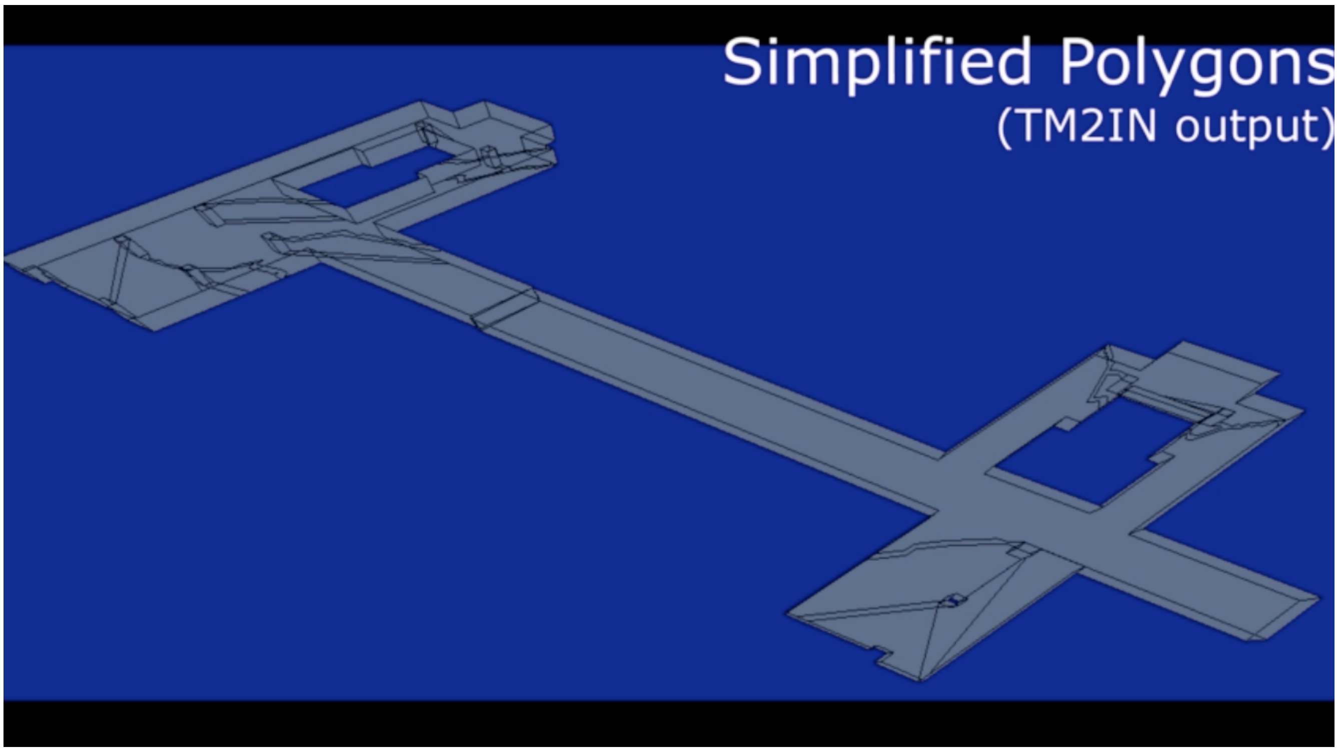
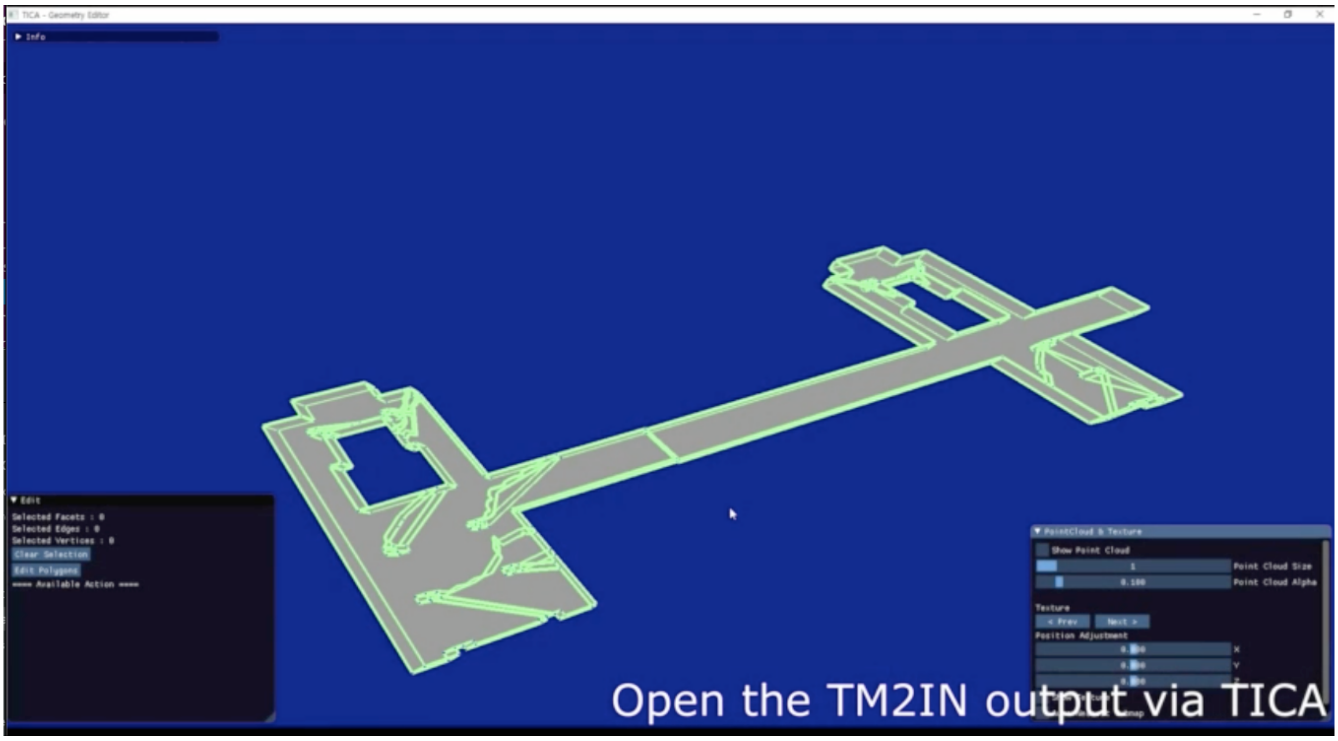


Figure 29. Simplified Polygon representation

A set of simplified polygons are defined based on the mesh



▼ Edit
Selected Facets : 0
Selected Edges : 0
Selected Vertices : 0
Clear Selection
Edit Polygons
==== Available Action ====

▼ PointCloud & Texture
Show Point Cloud
1 Point Cloud Size
0.100 Point Cloud Alpha
Texture
< Prev Next >
Position Adjustment
0.000 X
0.000 Y
0.000 Z

Figure 30. Conversion using TICA tool

The simplified polygons are converted using a tool called TM-based IndoorGML Cleaning & Authoring (TICA) tool.

```
https://github.com/STEMLab/TICA
```

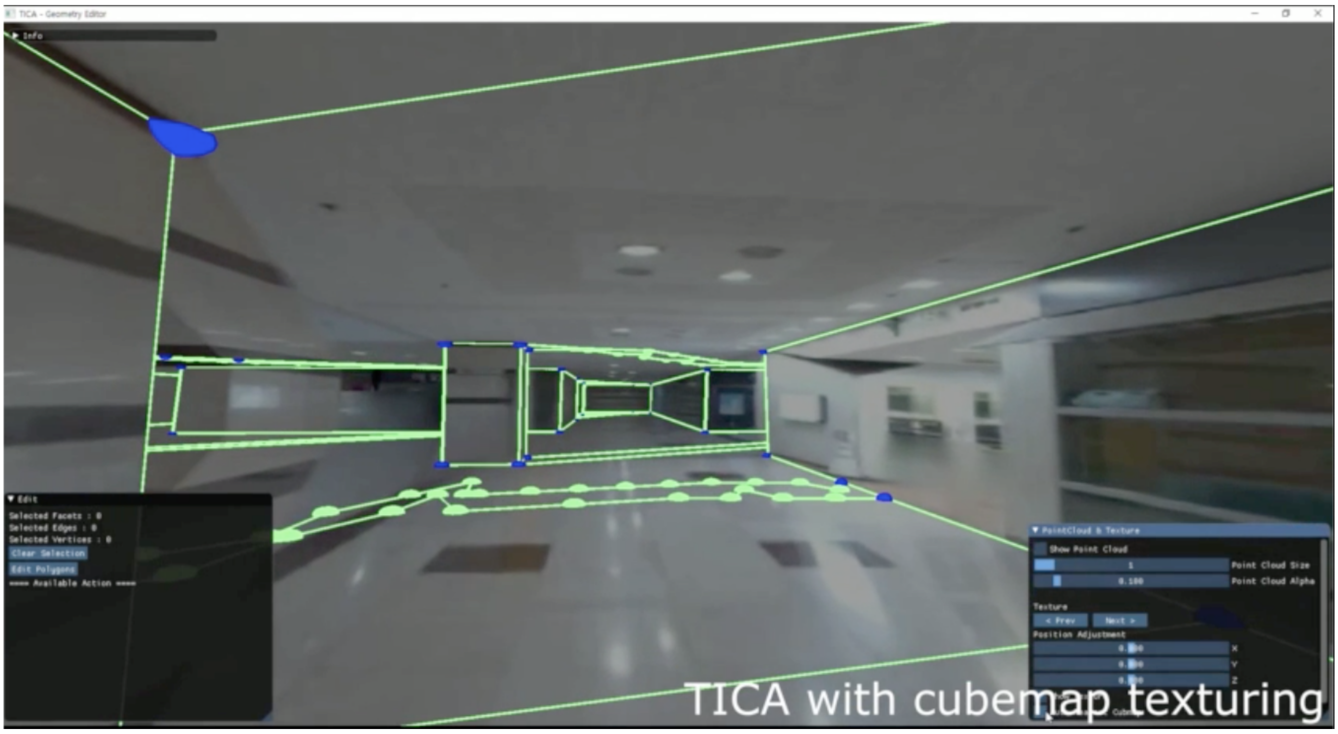


Figure 31. Cubemap Texturing applied using TICA

TICA is used to apply cubemap texturing.

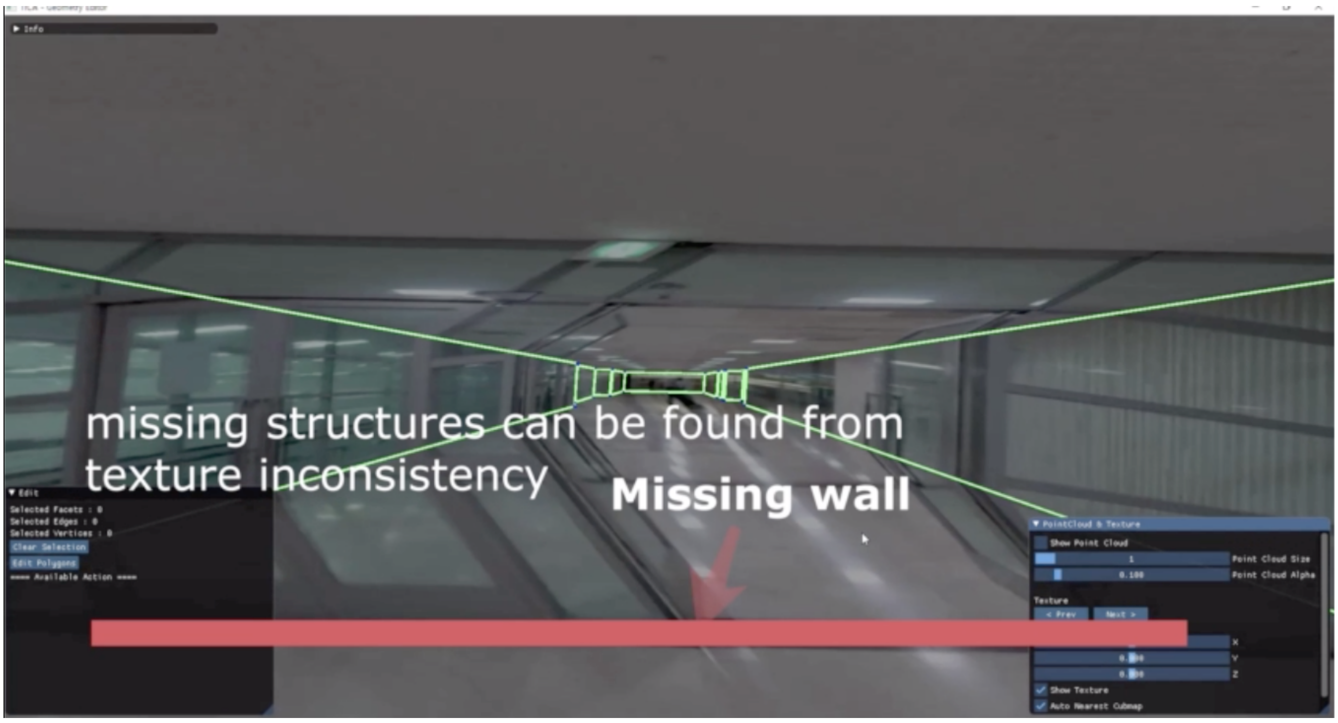


Figure 32. Addition of missing walls

Missing walls are manually added.

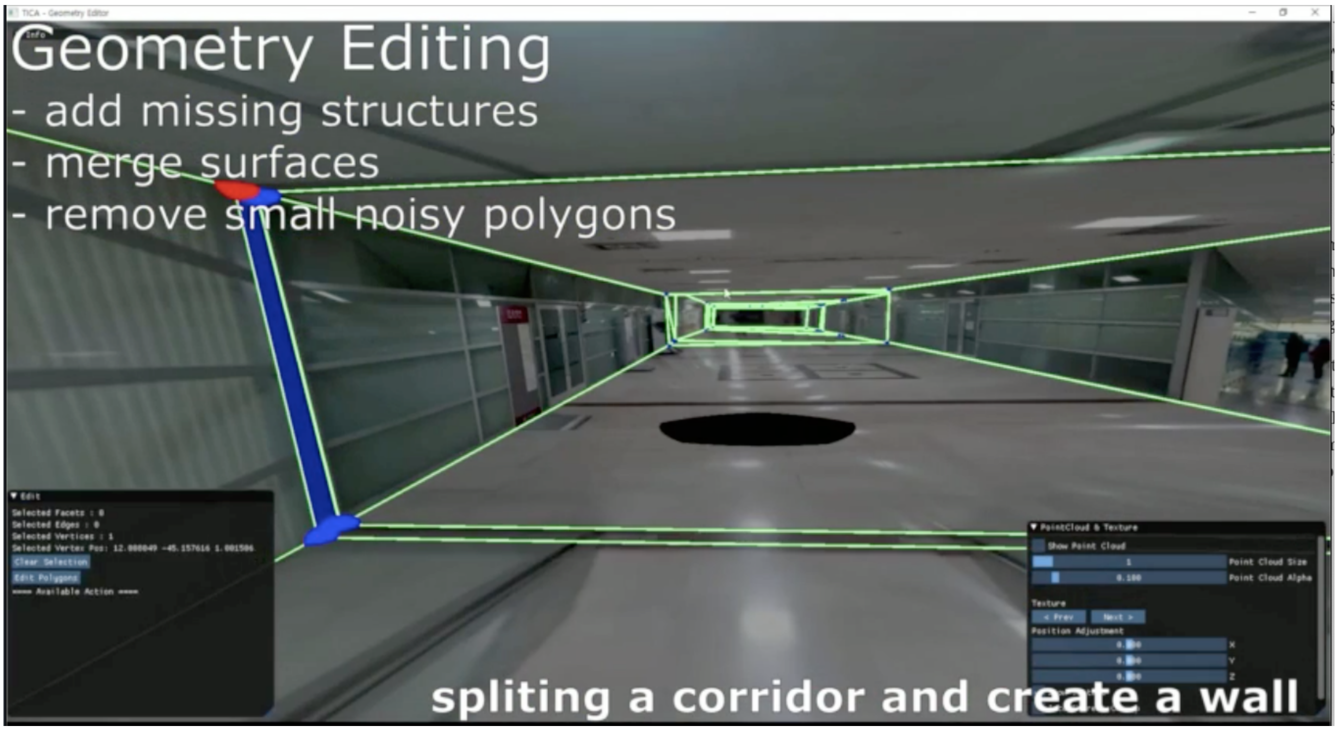

Geometry Editing
- add missing structures
- merge surfaces
- remove small noisy polygons
spliting a corridor and create a wall

Figure 33. Splitting a corridor and creating a wall

Large single spaces are not very useful for navigation. Splitting a large single space into several smaller spaces is called sub-spacing. Partition of sub-spacing can be done using TICA for TM-based pipelines to construct indoor space models (see Figure 33).

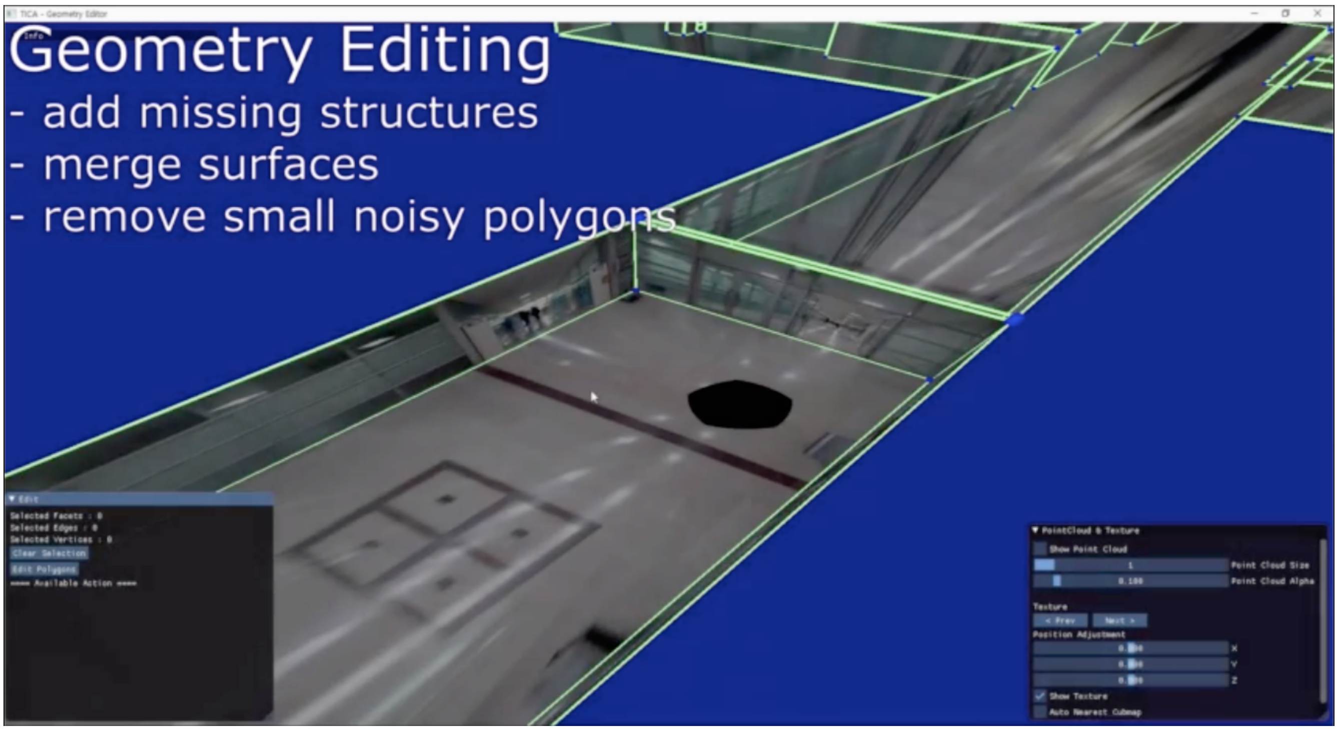
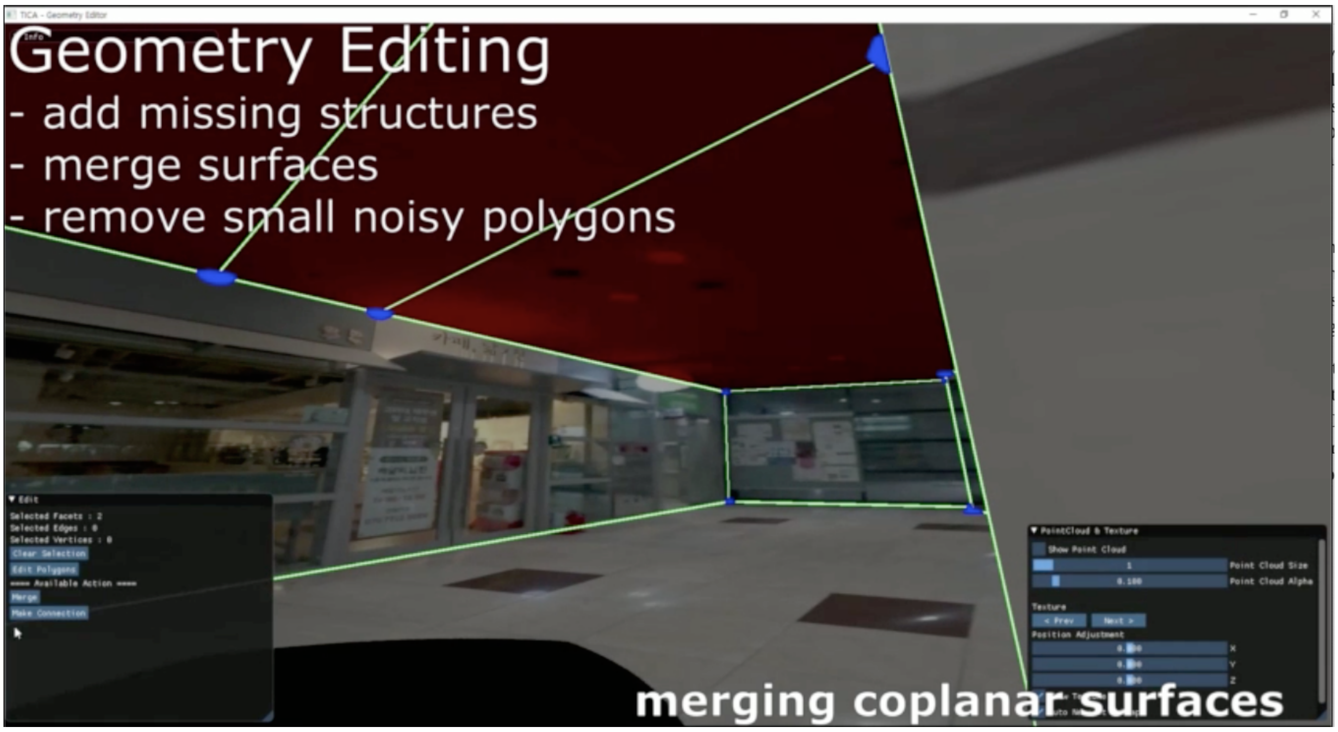


Figure 34. Polygon cleaning

The geometry is edited to add missing structures, merge surfaces, and remove "noisy" polygons.

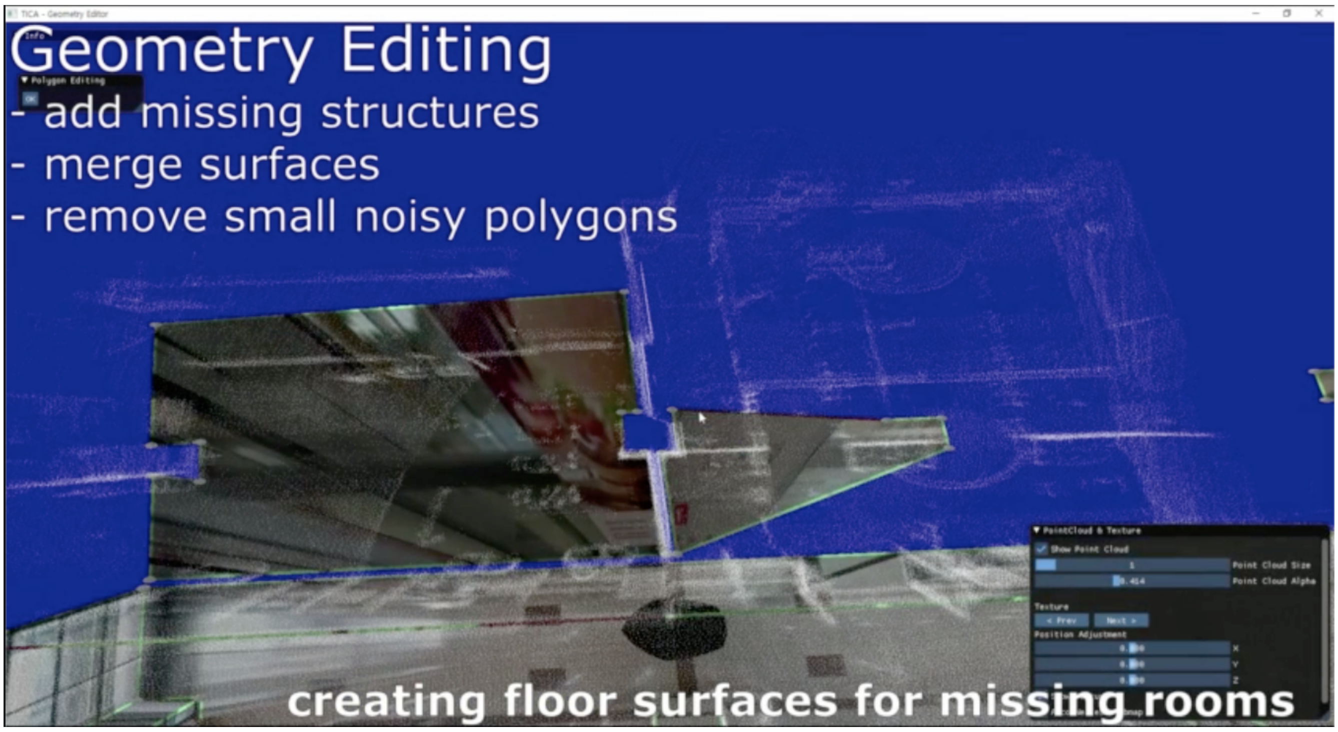


Selected Facets : 2
Selected Edges : 0
Selected Vertices : 0

Show Point Cloud
1 Point Cloud Size
0.100 Point Cloud Alpha

Figure 35. Merging of coplanar surfaces

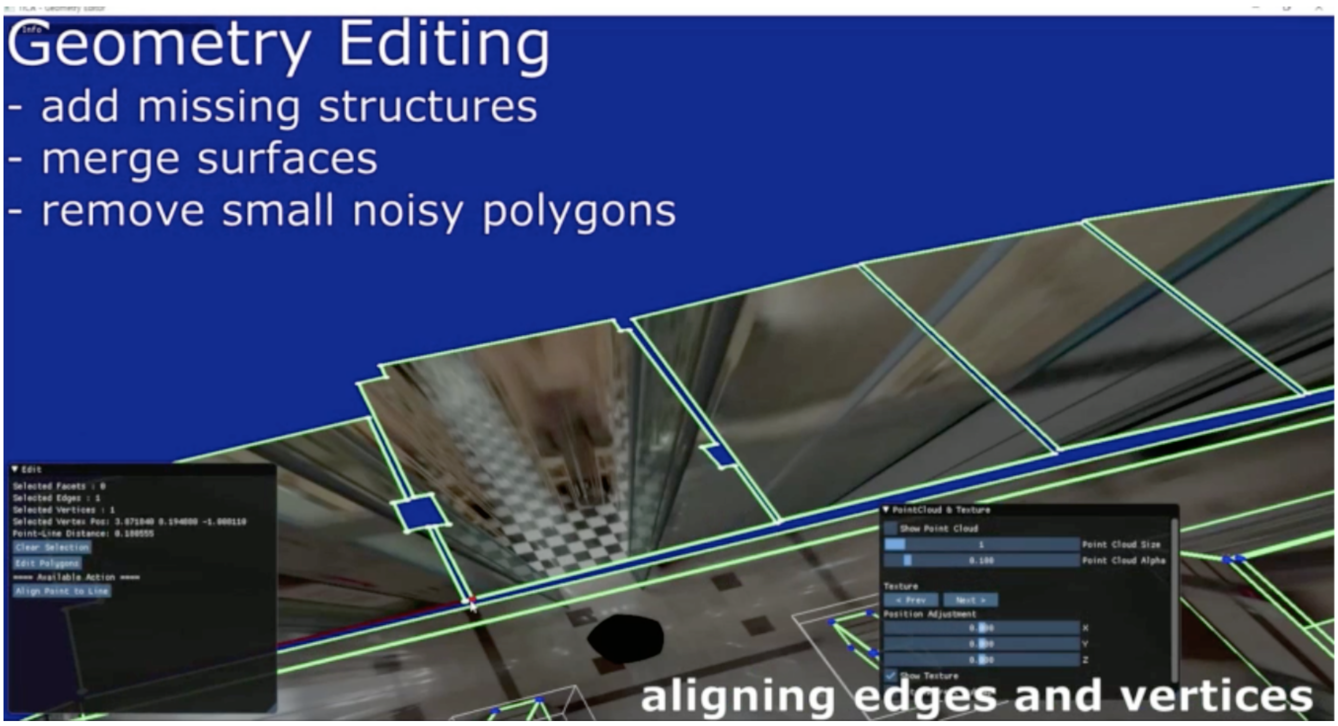
Coplanar surfaces that have splits or stitches are merged into a single planar surface.



[✓ Show Point Cloud]
1 Point Cloud Size
0.414 Point Cloud Alpha

Figure 36. Drawing floors manually for missing point cloud data

Missing floors are drawn for areas where the SLAM sensor could not capture point clouds.


Geometry Editing
- add missing structures
- merge surfaces
- remove small noisy polygons

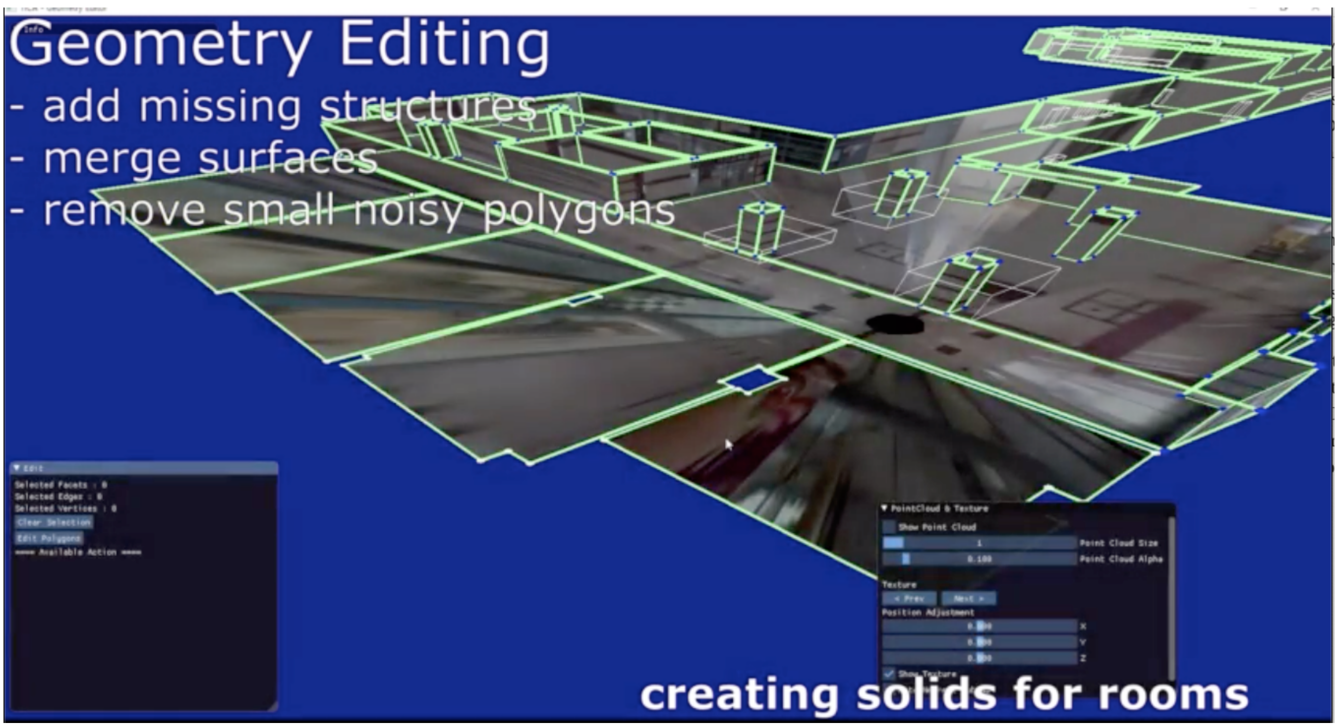
```
▼ Edit
Selected Facets : 0
Selected Edges : 1
Selected Vertices : 1
Selected Vertex Pos: 3.071040 0.194000 -1.000110
Point-Line Distance: 0.100555
[Clear Selection]
[Edit Polygons]
==== Available Action ====
[Align Point to Line]

▼ PointCloud & Texture
[ ] Show Point Cloud
1        Point Cloud Size
0.100    Point Cloud Alpha
Texture
[< Prev] [Next >]
Position Adjustment
0.000    X
0.000    Y
0.000    Z
[✓] Show Texture
```

aligning edges and vertices

Figure 37. Manual processing of aligning vertices and edges

Vertices and edges are aligned.


Geometry Editing
- add missing structures
- merge surfaces
- remove small noisy polygons

```
▼ Edit
Selected Facets : 0
Selected Edges : 0
Selected Vertices : 0
[Clear Selection]
[Edit Polygons]
==== Available Action ====

▼ PointCloud & Texture
[ ] Show Point Cloud
1        Point Cloud Size
0.100    Point Cloud Alpha
Texture
[< Prev] [Next >]
Position Adjustment
0.000    X
0.000    Y
0.000    Z
[✓] Show Texture
```

creating solids for rooms

Figure 38. Creating room solids

Solids are created for each room.

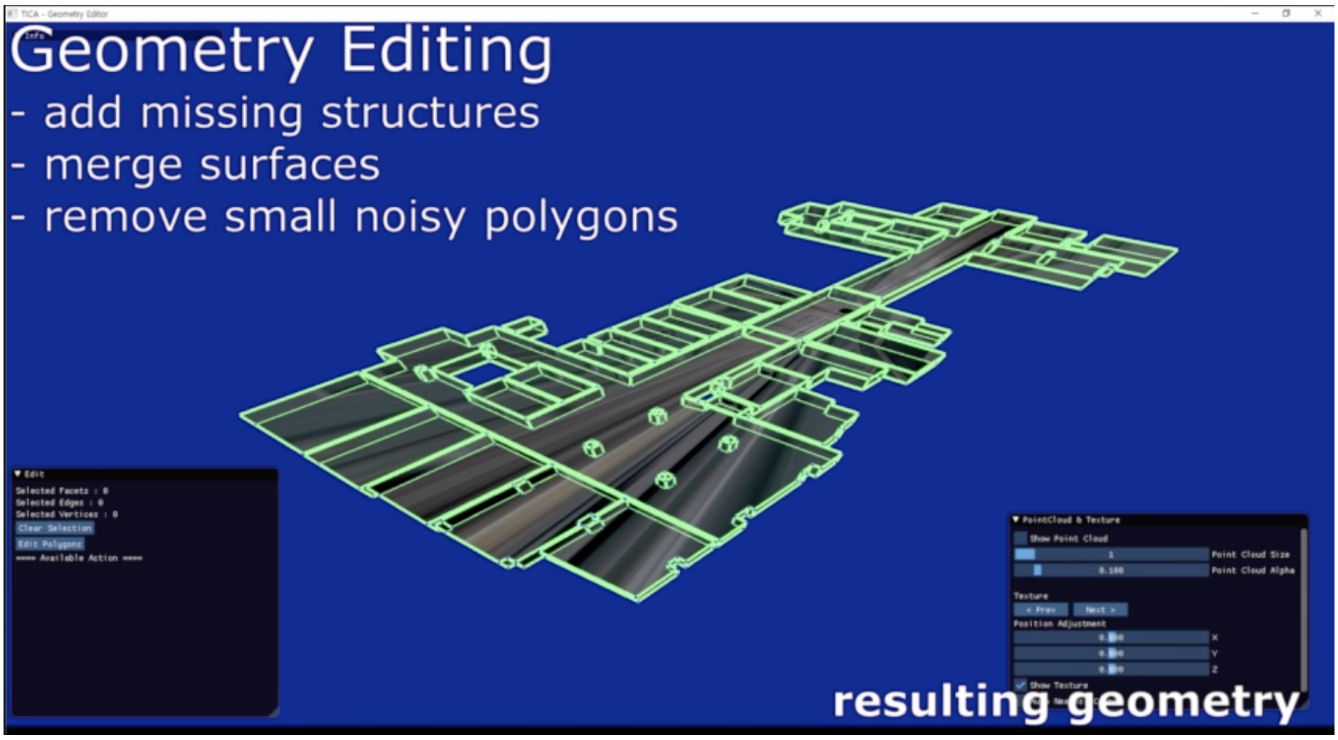


Figure 39. Resulting geometry

The result is a clean 3D geometry of the indoor space.

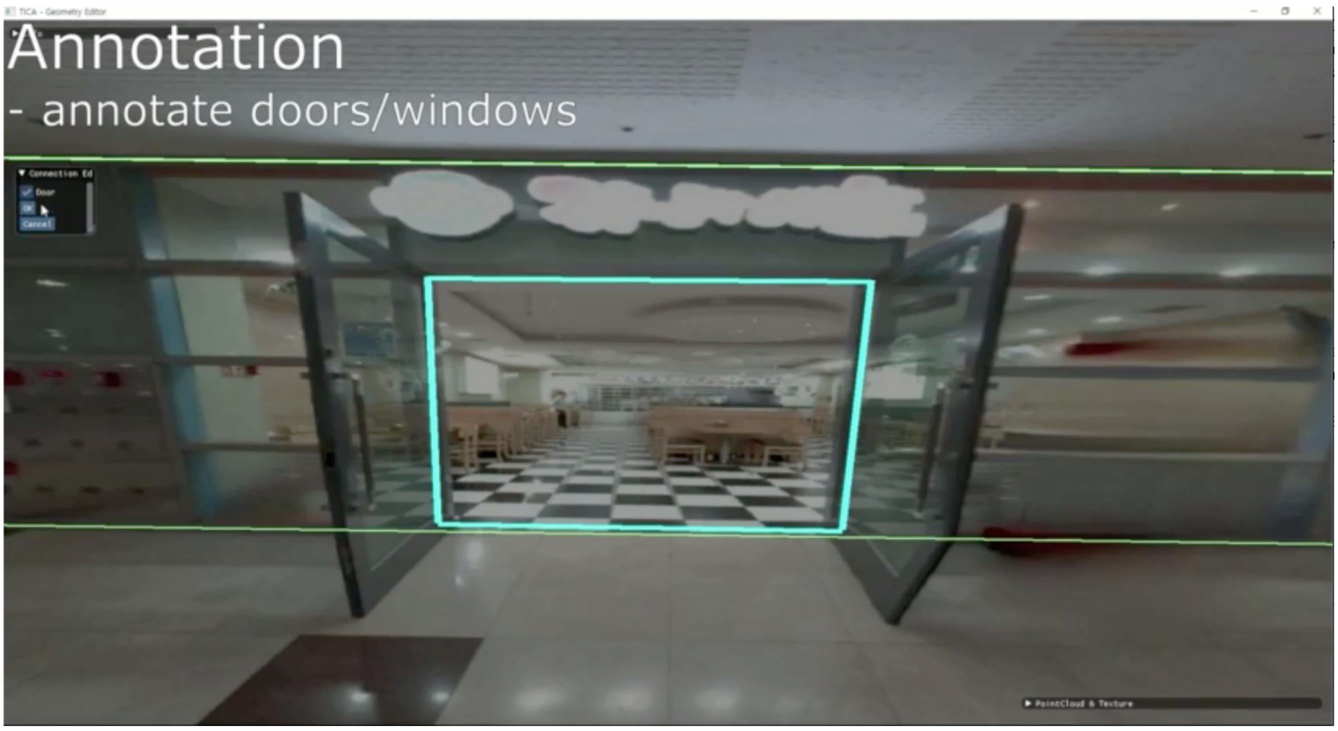


Figure 40. Annotating Doors and Windows

Doors and windows must be annotated.

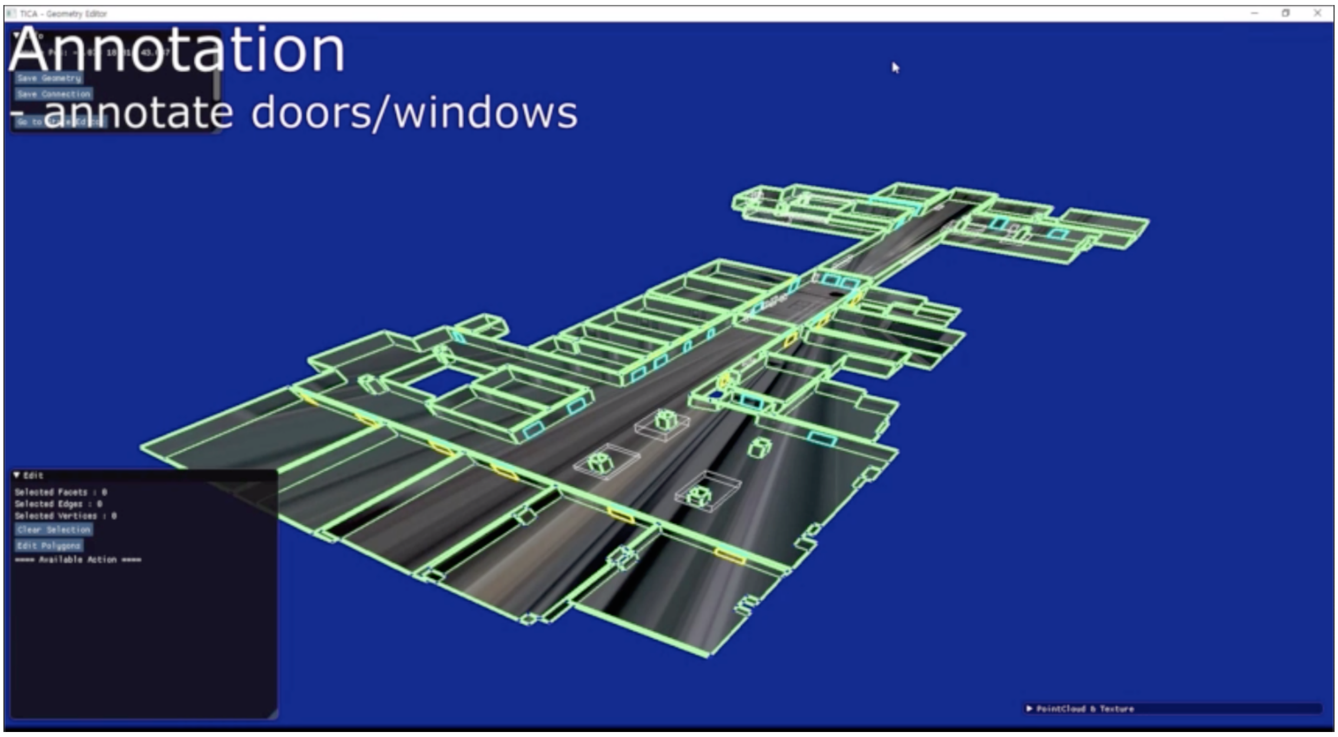


Figure 41. Annotated doors and windows

Figure 41 shows the result after doors and windows are annotated.

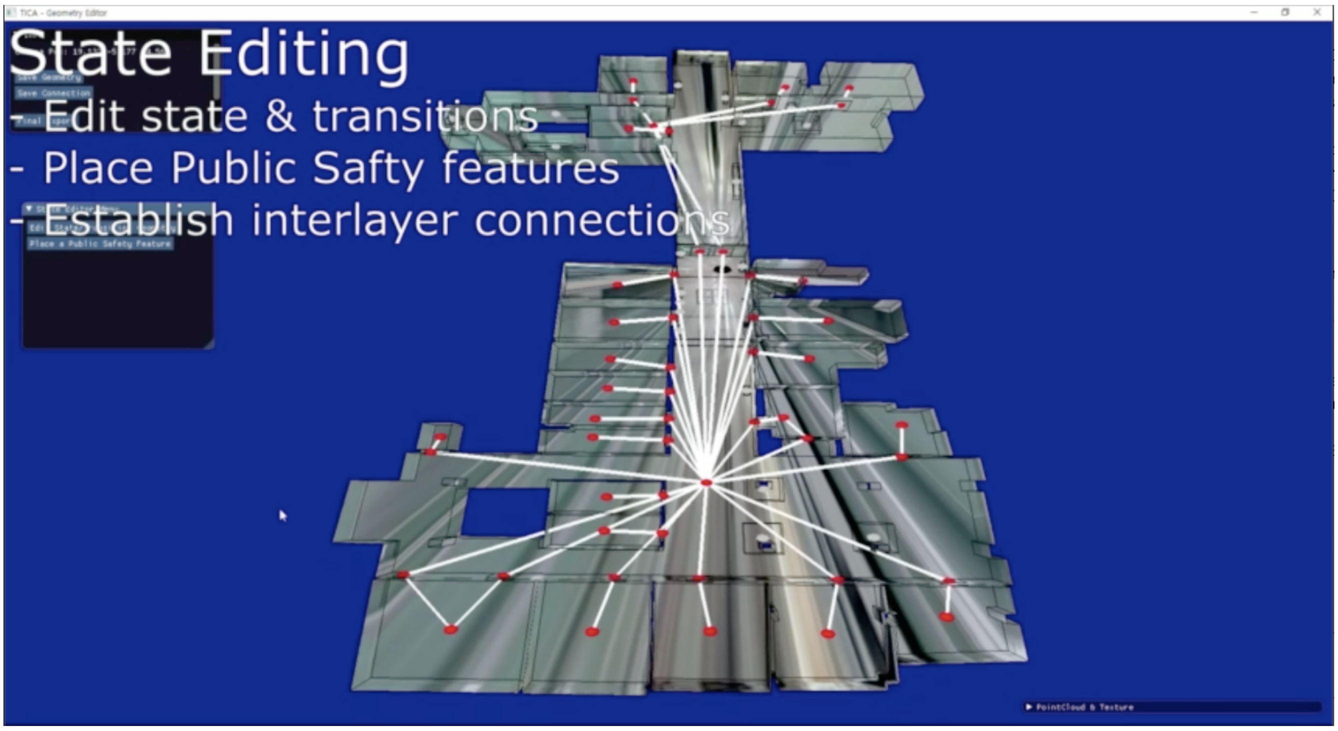

State Editing
- Edit state & transitions
- Place Public Safty features
- Establish interlayer connections

Figure 42. Multilayer graph of indoor navigable spaces

States and transitions are edited and interlayer connections are made.

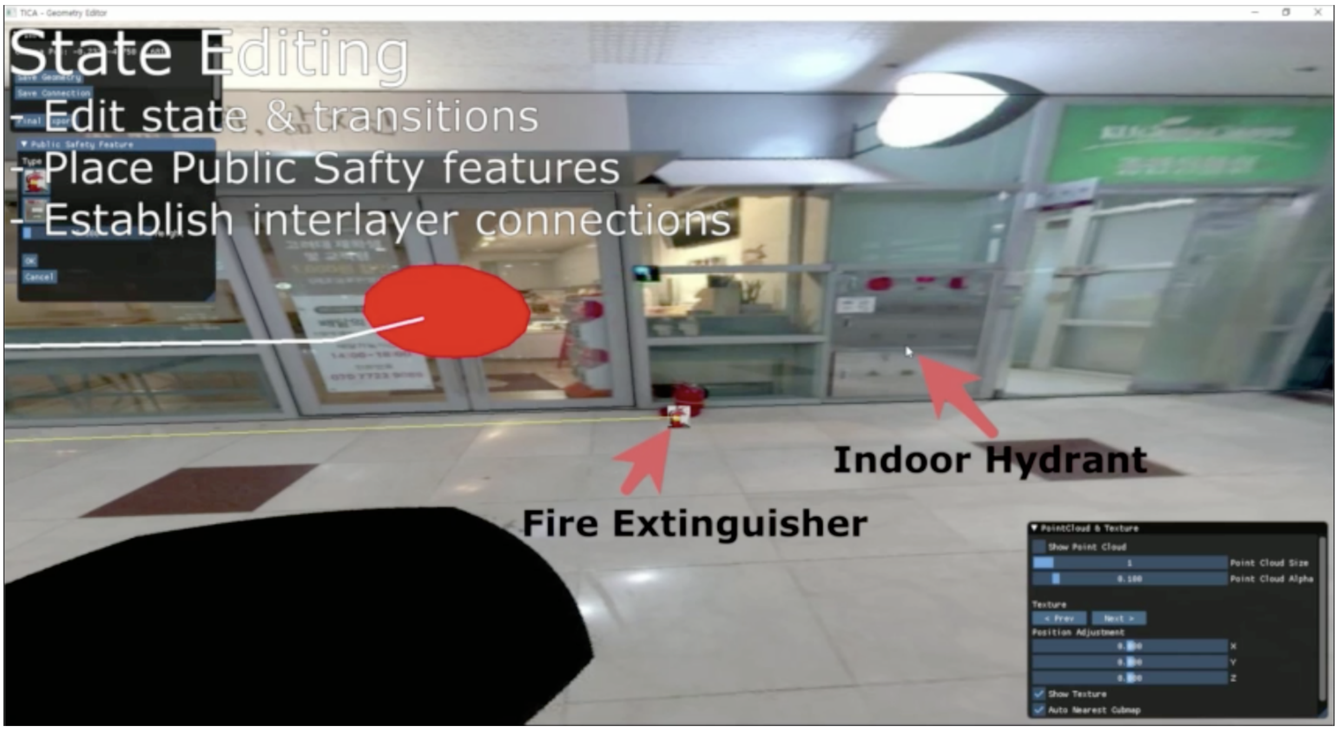

State Editing
- Edit state & transitions
- Place Public Safty features
- Establish interlayer connections
Indoor Hydrant
Fire Extinguisher

Figure 43. Public Safety Features

Public Safety features are added.

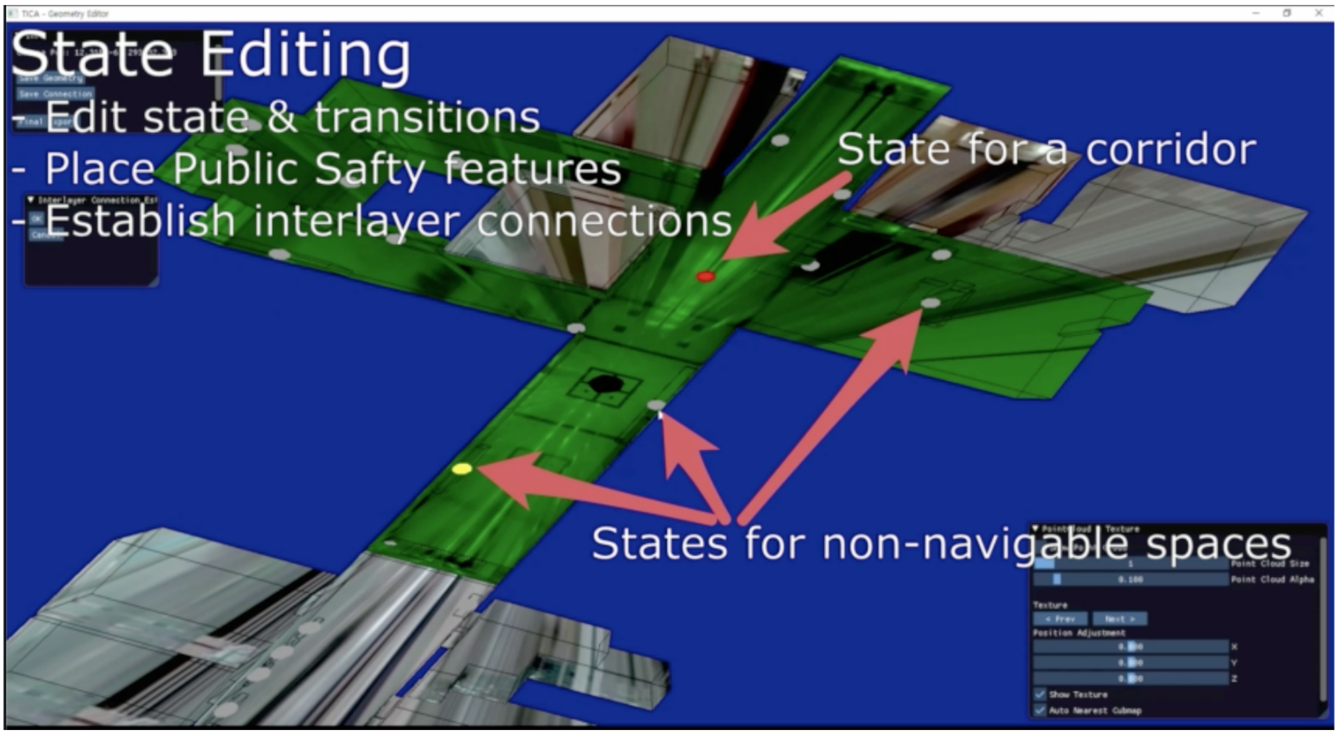
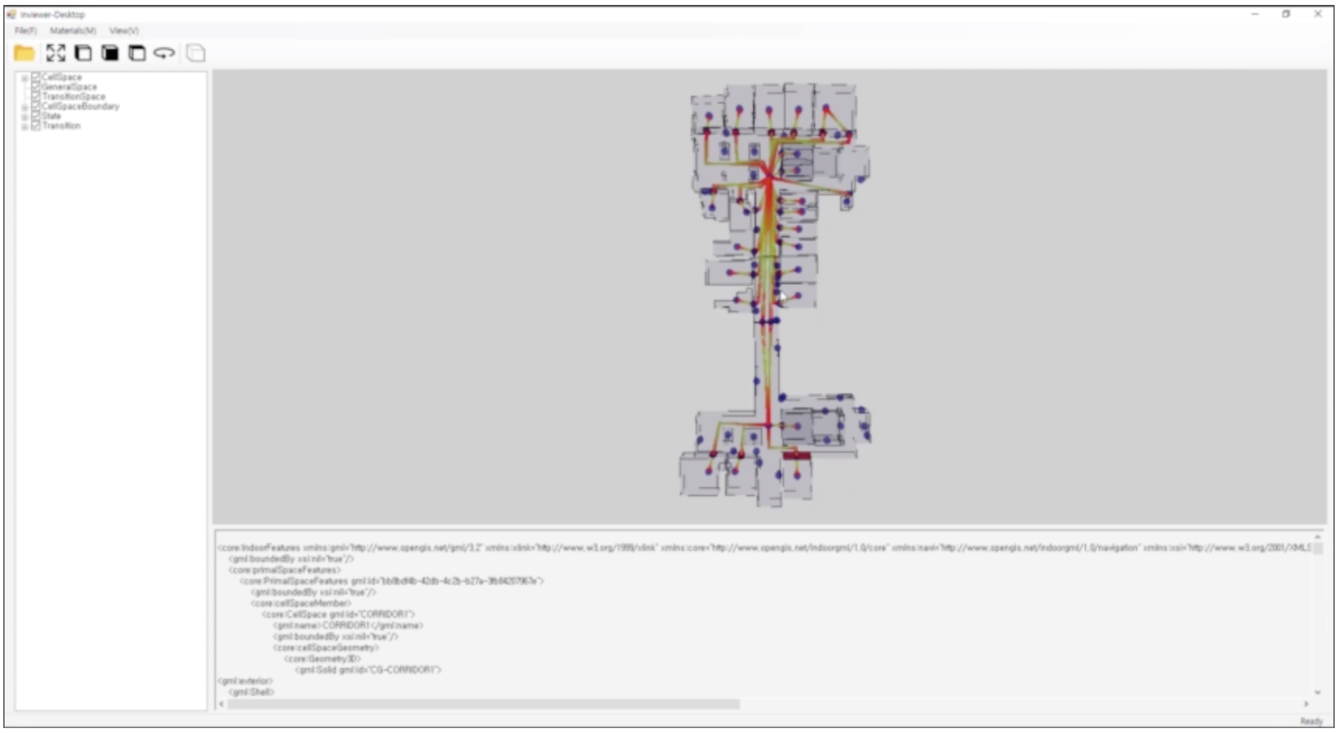

State Editing
- Edit state & transitions
- Place Public Safty features
- Establish interlayer connections
State for a corridor
States for non-navigable spaces

Figure 44. Navigable and Non-navigable spaces

The states for corridor and states for non-navigable spaces are finalized.



```
<core:IndoorFeatures xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:core="http://www.opengis.net/indoorgml/1.0/core" xmlns:navi="http://www.opengis.net/indoorgml/1.0/navigation" xmlns:xsi="http://www.w3.org/2001/XMLS
  <gml:boundedBy xsi:nil="true"/>
  <core:primalSpaceFeatures>
    <core:PrimalSpaceFeatures gml:id="bb8bd4b-42db-4c2b-b27a-3fb8420796 7e">
      <gml:boundedBy xsi:nil="true"/>
      <core:cellSpaceMember>
        <core:CellSpace gml:id="CORRIDOR1">
          <gml:name>CORRIDOR1</gml:name>
          <gml:boundedBy xsi:nil="true"/>
          <core:cellSpaceGeometry>
            <core:Geometry3D>
              <gml:Solid gml:id="CG-CORRIDOR1">
<gml:exterior>
  <gml:Shell>
```

Figure 45. InViewer Application display of final result

The final result can be depicted in the InViewer application.

7.1.2. Conversion from IndoorGML into CityGML

Figure 46 shows the use of point cloud data with cuboid images to create texture mappings in a 3-dimensional representation using TICA.

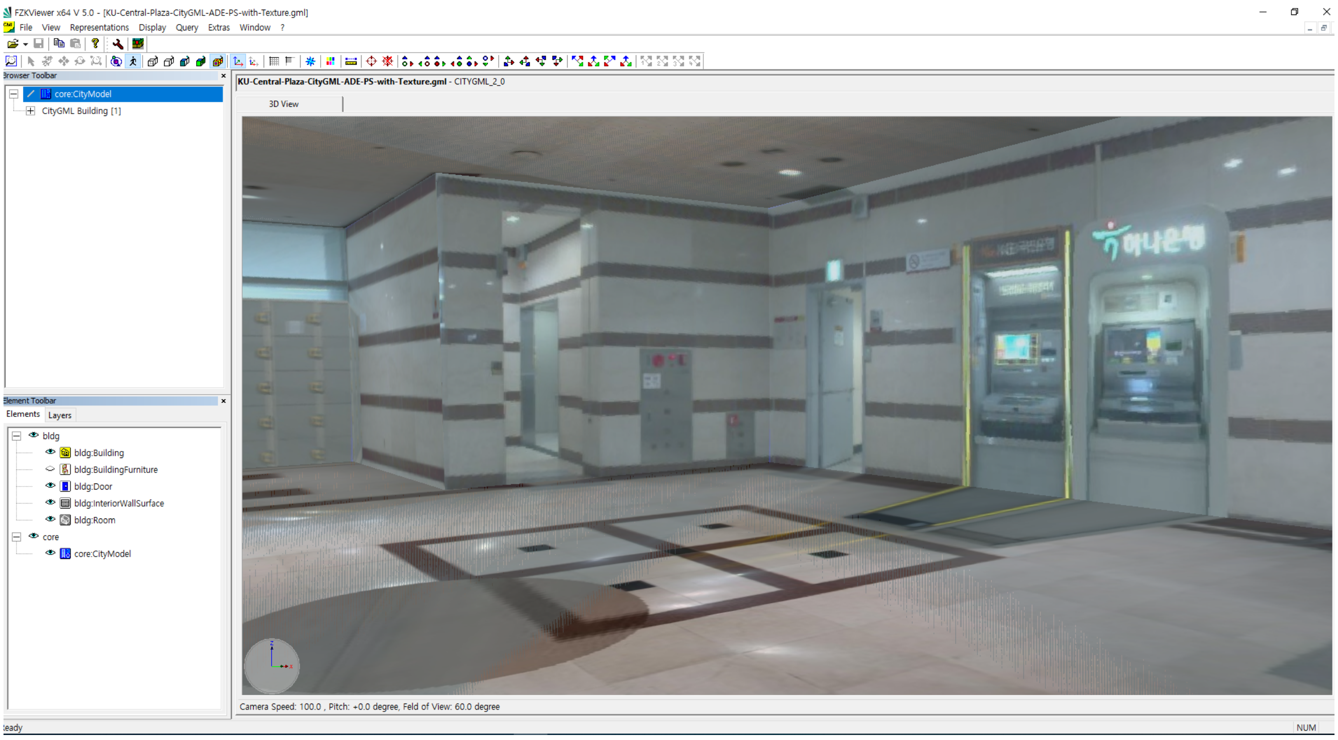

FZKViewer x64 V 5.0 - [KU-Central-Plaza-CityGML-ADE-PS-with-Texture.gml]
File View Representations Display Query Extras Window ?
Browser Toolbar
core:CityModel
CityGML Building [1]
KU-Central-Plaza-CityGML-ADE-PS-with-Texture.gml - CITYGML_2_0
3D View
Element Toolbar
Elements Layers
bldg
bldg:Building
bldg:BuildingFurniture
bldg:Door
bldg:InteriorWallSurface
bldg:Room
core
core:CityModel
Camera Speed: 100.0 , Pitch: +0.0 degree, Field of View: 60.0 degree
Ready
NUM

Figure 46. KU, Central Plaza - CityGML Interior with Public Safety ADE and Texture Applied

Once the IndoorGML data set with public safety features and textures are prepared, it is converted to CityGML using Hale, an open-source conversion tool.

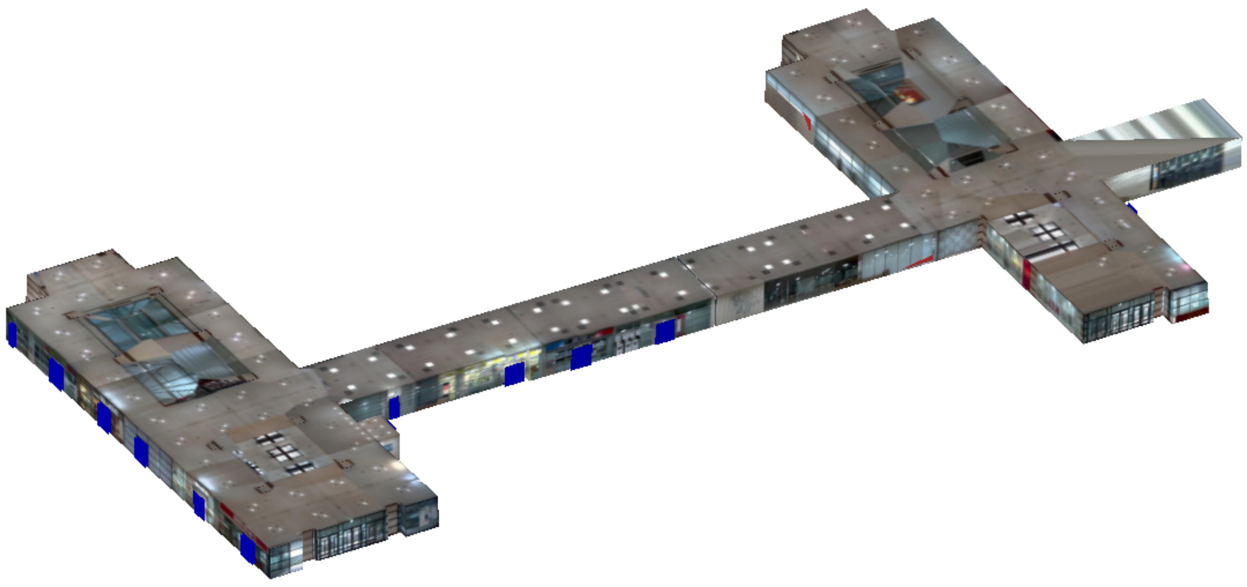


Figure 47. KU,Central Plaza - CityGML with Public Safety ADE and Texture Applied

This process uses the Hale transformation rule. The Hale Mappings are provided at the following URL:

```
https://github.com/STEMLab/IndoorGML2CityGML_PSExtension
```

Since IndoorGML and CityGML are modeled as complex schemas, we have to define the mapping rules from the top level to lower ones.

- The mapping rules in the attached file aim at converting IndoorGML 3D objects to CityGML Building features.
- For the mapping rules between IndoorGML and CityGML, start with PrimalSpaceFeatures as the top level.

Hale provides several approaches, among which two are applied:

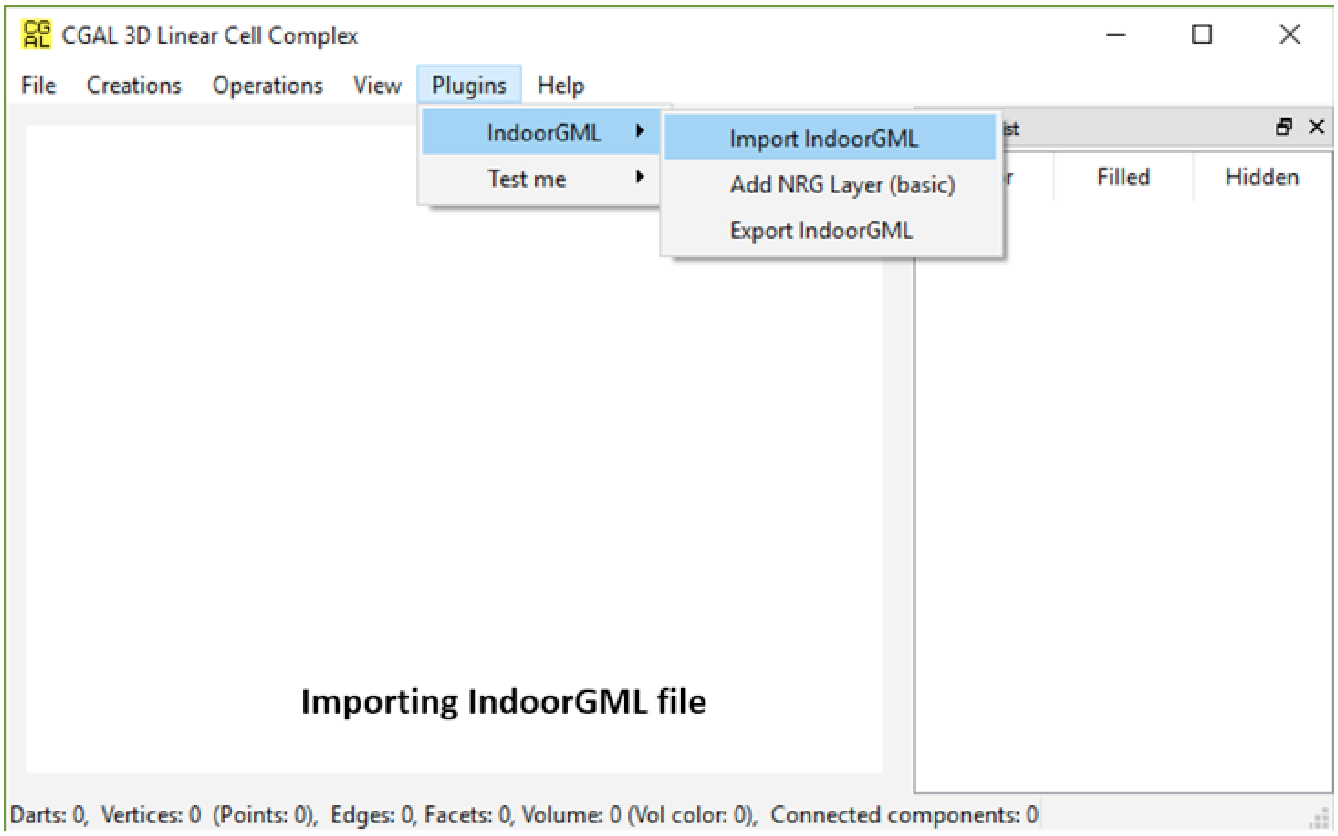
- ReType:toplevel
- Rename: mapping rules of lower levels of ReType (1:1, 1:n, n:1)
- Retype PrimalSpaceFeatures in IndoorGML to CityModel in CityGML.
- Define the mapping rules between feature types of the lower levels
  - IndoorGML.cellSpaceMember–CityGML.FeatureMember
  - IndoorGML.cellSpace–CityGML.Room
- Define the mapping rules for the attributes such as
  - id
  - geometry

When converting from CityGML to IndoorGML, GML.ID is

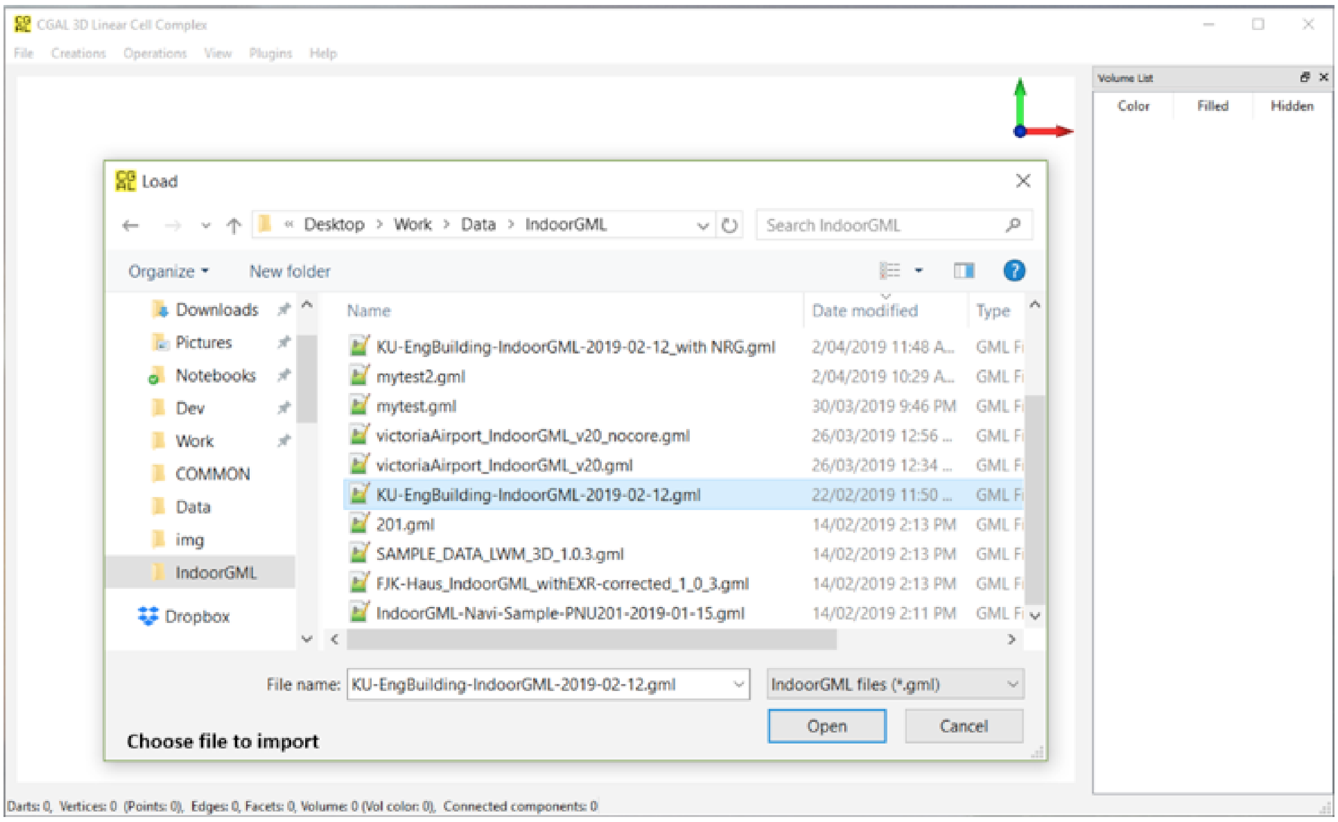
- not mandatory in GML3.1.1, which is the base standard for CityGML 2.0
- Mandatory in GML3.2.1, which is the base standard for IndoorGML
    - generated using ‘Generate Unique Id” function

7.1.3. IndoorGML Network Sub-Spacing

The GRID team from University of New South Wales (UNSW) provides a tool, called IndoorGML2LCC which allows to add a Node Relation Graph (NRG) to IndoorGML models. The tool, available online (https://github.com/grid-unsw/IndoorGML2LCC), uses the Linear Cell Complex (LCC) package from the Computational Geometry Algorithm Library (CGAL) as a data structure to handle the geometric and topological information described in IndoorGML. It is provided as a plugin that fits directly to the 3D LCC demo of CGAL. While the installation process of IndoorGML2LCC is described in the Github page, its use is described below.



Darts: 0, Vertices: 0 (Points: 0), Edges: 0, Facets: 0, Volume: 0 (Vol color: 0), Connected components: 0



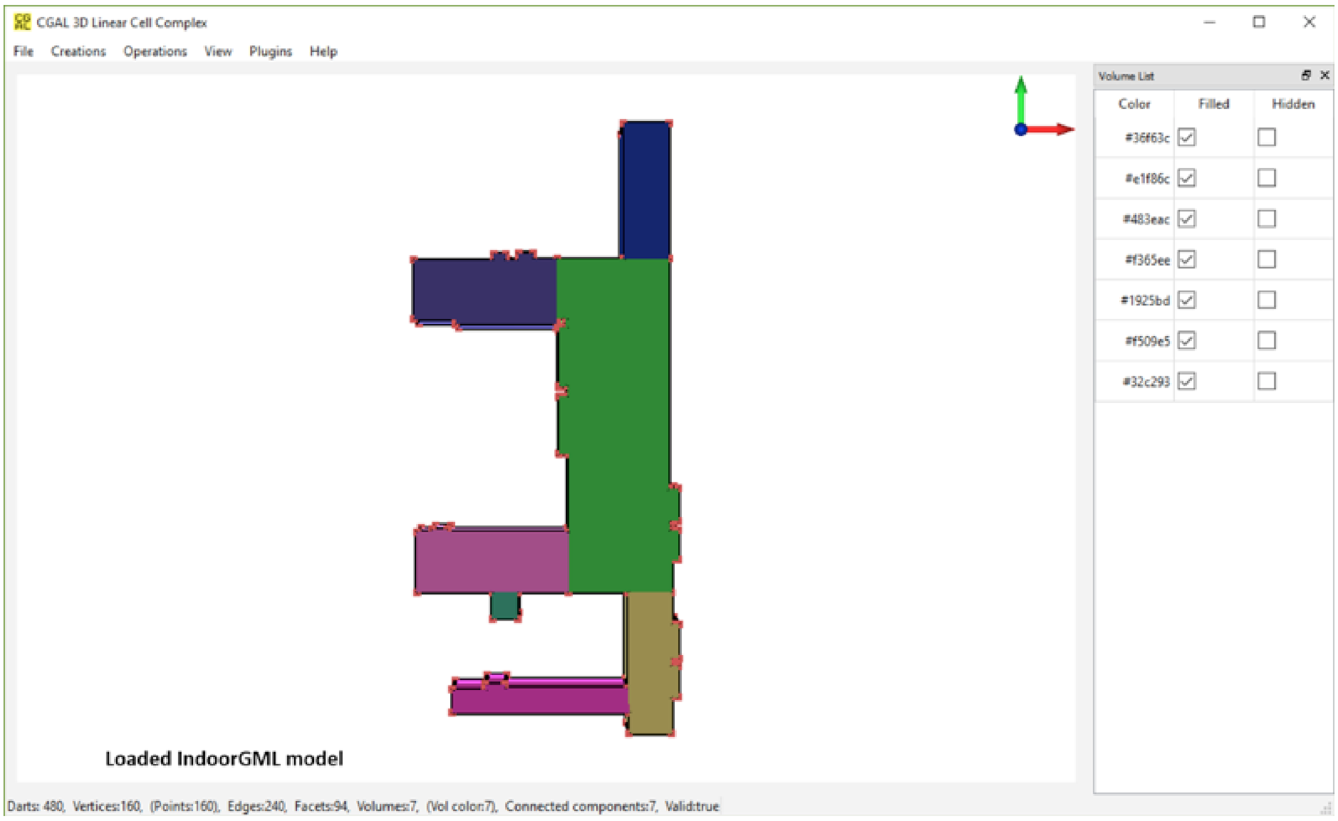
Name	Date modified	Type
KU-EngBuilding-IndoorGML-2019-02-12_with NRG.gml	2/04/2019 11:48 A...	GML Fi
mytest2.gml	2/04/2019 10:29 A...	GML Fi
mytest.gml	30/03/2019 9:46 PM	GML Fi
victoriaAirport_IndoorGML_v20_nocore.gml	26/03/2019 12:56 ...	GML Fi
victoriaAirport_IndoorGML_v20.gml	26/03/2019 12:34 ...	GML Fi
KU-EngBuilding-IndoorGML-2019-02-12.gml	22/02/2019 11:50 ...	GML Fi
201.gml	14/02/2019 2:13 PM	GML Fi
SAMPLE_DATA_LWM_3D_1.0.3.gml	14/02/2019 2:13 PM	GML Fi
FJK-Haus_IndoorGML_withEXR-corrected_1_0_3.gml	14/02/2019 2:13 PM	GML Fi
IndoorGML-Navi-Sample-PNU201-2019-01-15.gml	14/02/2019 2:11 PM	GML Fi

File name: KU-EngBuilding-IndoorGML-2019-02-12.gml
IndoorGML files (*.gml)

Darts: 0, Vertices: 0 (Points: 0), Edges: 0, Facets: 0, Volume: 0 (Vol color: 0), Connected components: 0

Figure 48. Importing files from the user interface of IndoorGML2LCC

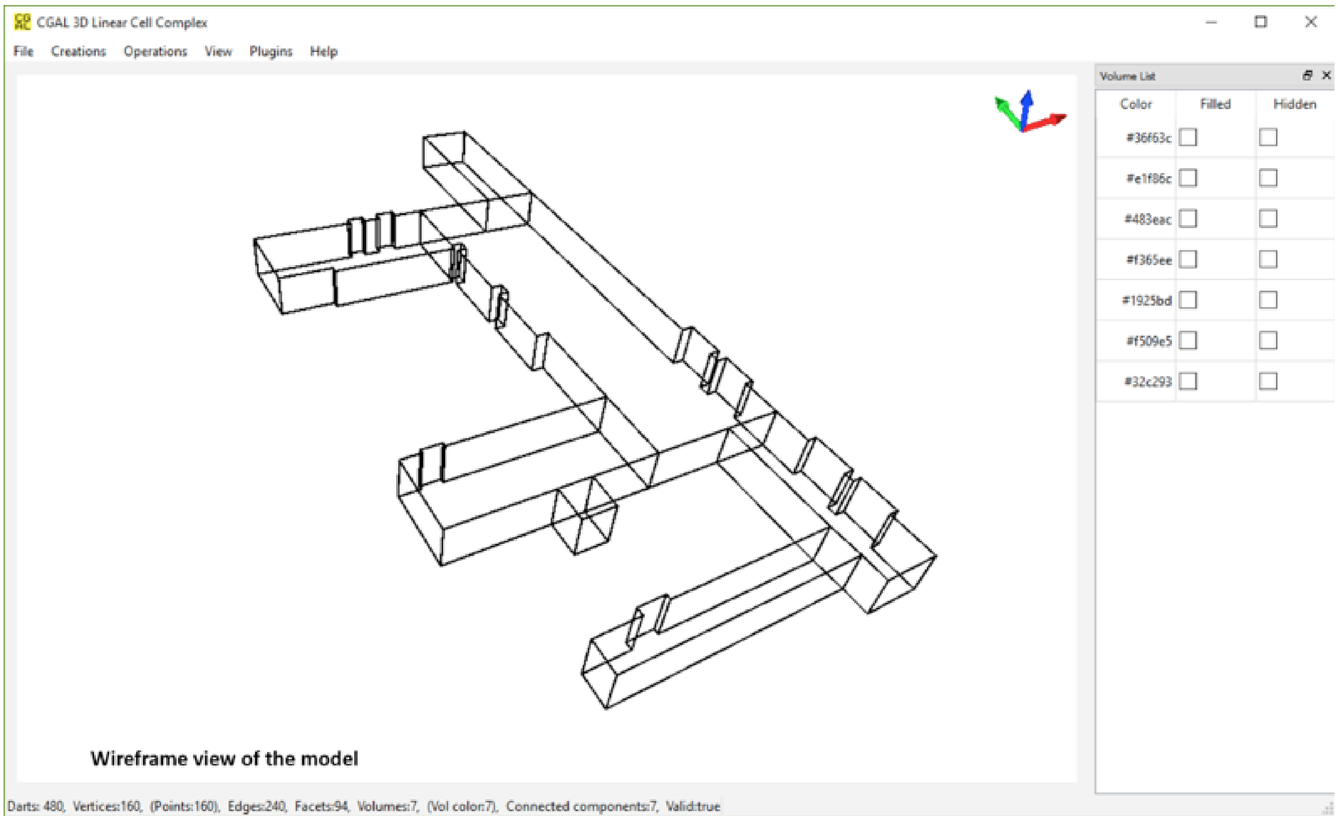
The IndoorGML files loaded in the tool were expected to be missing topological information and solely describe the spaces called CellSpace units. However, it is designed to preserve such data if already available and add another graph layer on top of it.



Color	Filled	Hidden
#36f63c	✓	☐
#e1f86c	✓	☐
#483eac	✓	☐
#f365ee	✓	☐
#1925bd	✓	☐
#f509e5	✓	☐
#32c293	✓	☐

Darts: 480, Vertices:160, (Points:160), Edges:240, Facets:94, Volumes:7, (Vol color:7), Connected components:7, Valid:true

Figure 49. Viewing the IndoorGML Model (Original View)

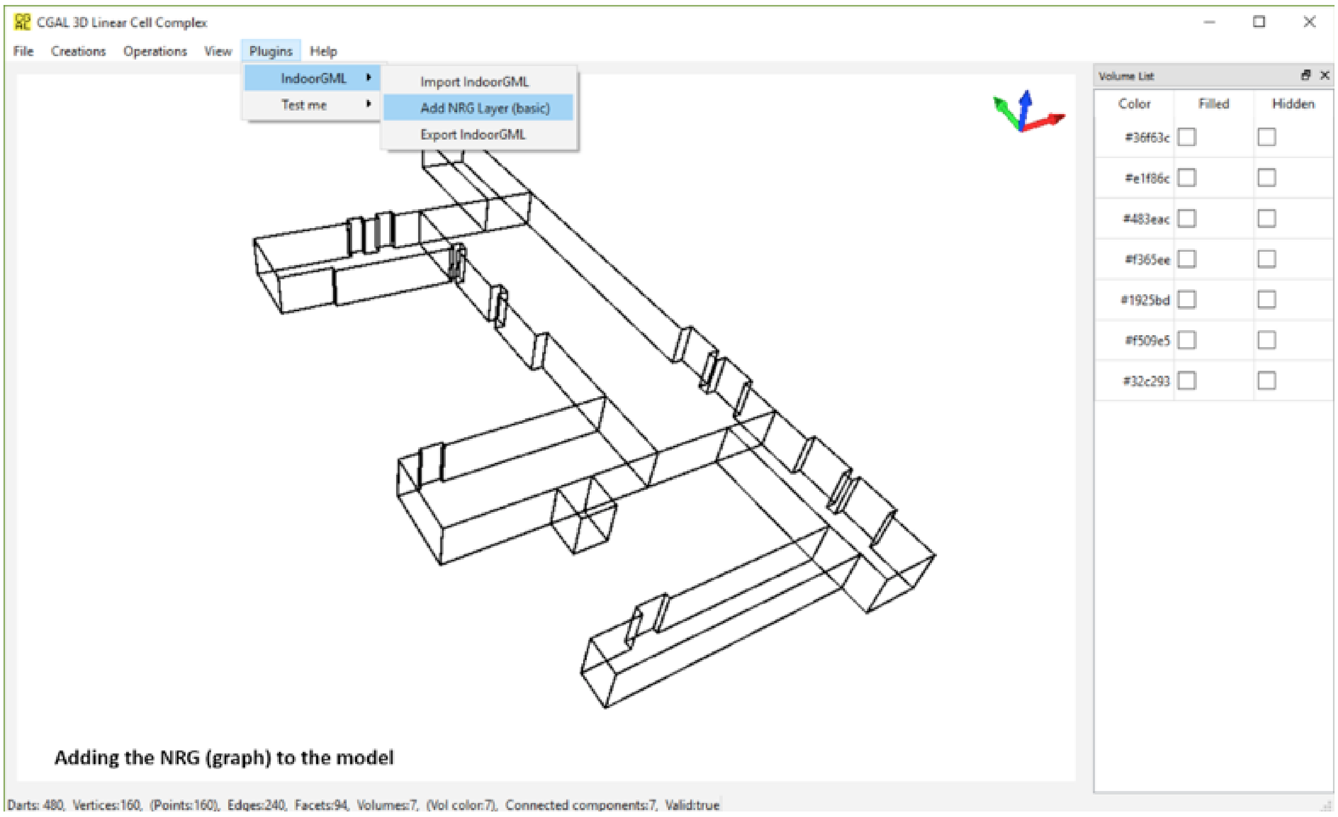


Color	Filled	Hidden
#36f63c	☐	☐
#e1f86c	☐	☐
#483eac	☐	☐
#f365ee	☐	☐
#1925bd	☐	☐
#f509e5	☐	☐
#32c293	☐	☐

Darts: 480, Vertices:160, (Points:160), Edges:240, Facets:94, Volumes:7, (Vol color:7), Connected components:7, Valid:true

Figure 50. Viewing the IndoorGML Model

Once the IndoorGML model is loaded, the user interface (UI) allows several views of the model and interaction options. Figure 49 illustrates the original view where every cell is represented with a unique color, which stands as a basic ID in the LCC. That view can be changed into a wireframe to see just the outline of the cells shown in Figure 50.

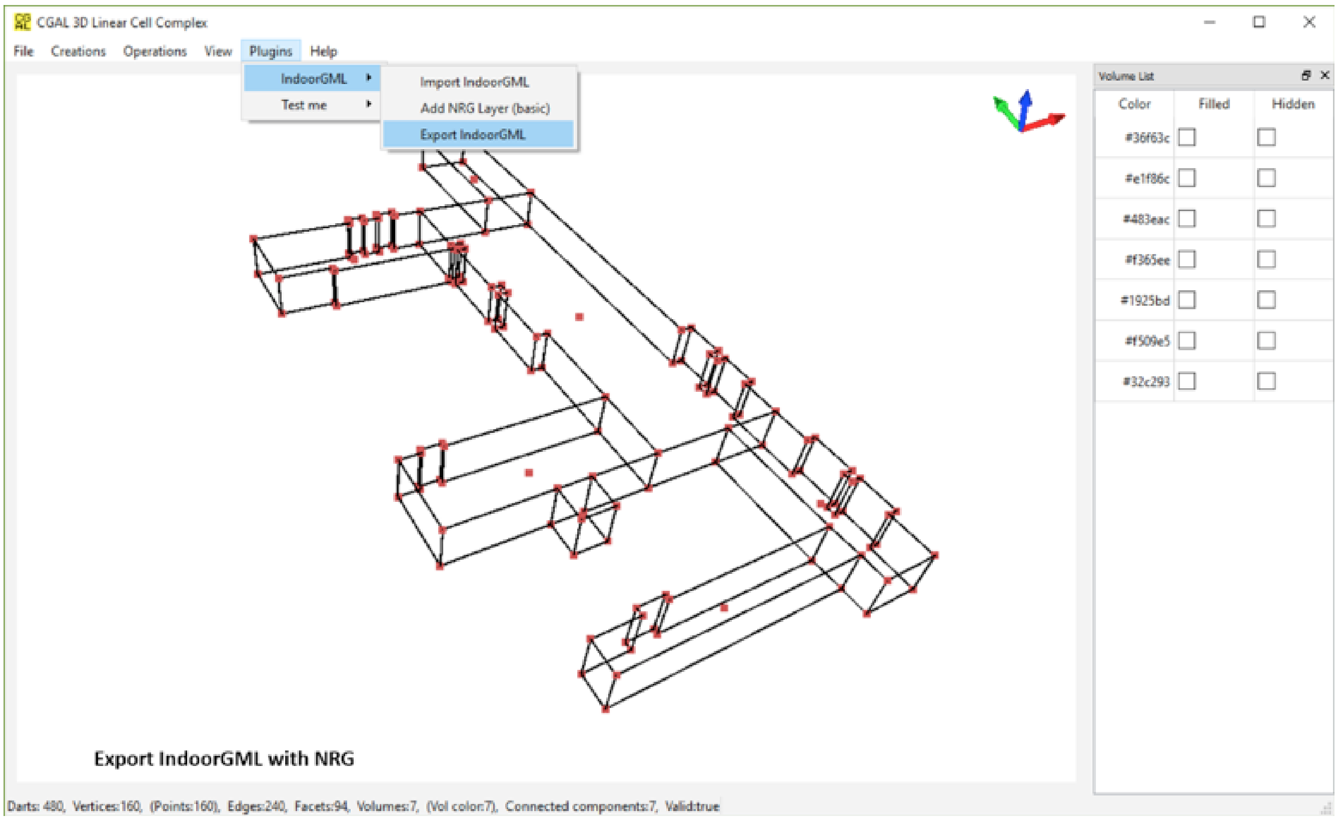


Color	Filled	Hidden
#36f63c	☐	☐
#e1f86c	☐	☐
#483eac	☐	☐
#f365ee	☐	☐
#1925bd	☐	☐
#f509e5	☐	☐
#32c293	☐	☐

Darts: 480, Vertices:160, (Points:160), Edges:240, Facets:94, Volumes:7, (Vol color:7), Connected components:7, Valid:true

Figure 51. NRG operation on the IndoorGML model

From the loaded CellSpace units, each of them is represented as a 3-Cell in the LCC. From there, the NRG can be automatically derived by running the “Add NRG Layer” operation (see Figure 51). It is called “basic” because it is implementing the original Poincare Duality theory which consists in abstracting every volume with a point (node) and every volume to volume adjacency with an edge, forming together the NRG. The nodes are obtained based on the centroids of the CellSpaces (see Figure 52) and their adjacency relationships are deduced from the faces that they share between them.

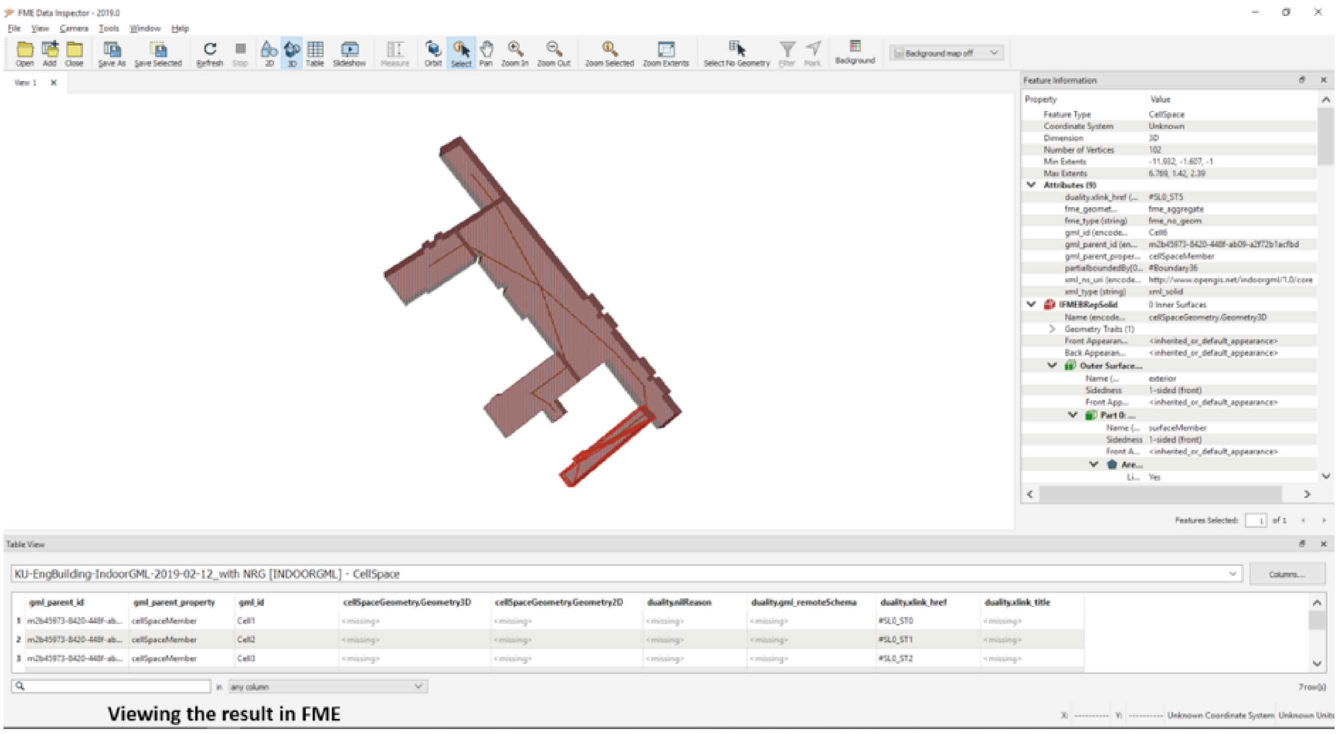


Color	Filled	Hidden
#36f63c	☐	☐
#e1f86c	☐	☐
#483eac	☐	☐
#f365ee	☐	☐
#1925bd	☐	☐
#f509e5	☐	☐
#32c293	☐	☐

```
Darts: 480,  Vertices:160,  (Points:160),  Edges:240,  Facets:94,  Volumes:7,  (Vol color:7),  Connected components:7,  Valid:true
```

Figure 52. Nodes of the CellSpaces and export process

While the graph visualization of IndoorGML2LCC is still under development, the exported model, enriched with an NRG layer can be visualized using FME (see Figure 53) or InViewer.


FME Data Inspector - 2019.0
File View Camera Tools Window Help
Open Add Close Save As Save Selected Refresh Stop 2D 3D Table Slideshow Measure Orbit Select Pan Zoom In Zoom Out Zoom Selected Zoom Extents Select No Geometry Filter Mark Background
Background map off
View 1
Feature Information
Table View
Columns...
any column
Viewing the result in FME

Property	Value
Feature Type	CellSpace
Coordinate System	Unknown
Dimension	3D
Number of Vertices	102
Min Extents	-11.932, -1.607, -1
Max Extents	6.769, 1.42, 2.39
Attributes (9)	
duality.xlink_href (...	#SL0_ST5
fme_geomet...	fme_aggregate
fme_type (string)	fme_no_geom
gml_id (encode...	Cell6
gml_parent_id (en...	m2b45973-8420-448f-ab09-a2f72b1acfbd
gml_parent_proper...	cellSpaceMember
partialboundedBy{0...	#Boundary36
xml_ns_uri (encode...	http://www.opengis.net/indoorgml/1.0/core
xml_type (string)	xml_solid
IFMEBRepSolid	0 Inner Surfaces
Name (encode...	cellSpaceGeometry.Geometry3D
Geometry Traits (1)	
Front Appearan...	<inherited_or_default_appearance>
Back Appearan...	<inherited_or_default_appearance>
Outer Surface...	
Name (...	exterior
Sidedness	1-sided (front)
Front App...	<inherited_or_default_appearance>
Part 0: ...	
Name (...	surfaceMember
Sidedness	1-sided (front)
Front A...	<inherited_or_default_appearance>
Are...	
Li...	Yes

Features Selected: 1 of 1

KU-EngBuilding-IndoorGML-2019-02-12_with NRG [INDOORGML] - CellSpace

	gml_parent_id	gml_parent_property	gml_id	cellSpaceGeometry.Geometry3D	cellSpaceGeometry.Geometry2D	duality.nilReason	duality.gml_remoteSchema	duality.xlink_href	duality.xlink_title
1	m2b45973-8420-448f-ab...	cellSpaceMember	Cell1	<missing>	<missing>	<missing>	<missing>	#SL0_ST0	<missing>
2	m2b45973-8420-448f-ab...	cellSpaceMember	Cell2	<missing>	<missing>	<missing>	<missing>	#SL0_ST1	<missing>
3	m2b45973-8420-448f-ab...	cellSpaceMember	Cell3	<missing>	<missing>	<missing>	<missing>	#SL0_ST2	<missing>

7 row(s)

X: ---------- Y: ---------- Unknown Coordinate System Unknown Units

Figure 53. Visualizing the generated NRG

7.2. Safe Software Navigation Modeler

Safe Software's approach used the FME spatial ETL tool to read the CityGML 2.0 Public Safety ADE data and transform it to IndoorGML v1.1 with the Public Safety Extension. Because FME uses a model-based approach, the transformation rules are tied to the underlying CityGML and IndoorGML schemas and are not dependent on particular datasets. This means that once the transformation was defined for one dataset, it was possible to use the same CityGML-to-IndoorGML model to transform other datasets. Some minor configurations were required, but these are mostly limited to mitigating deficiencies in individual datasets, or introducing custom business rules to auto-generate public safety content that was missing in a particular dataset.

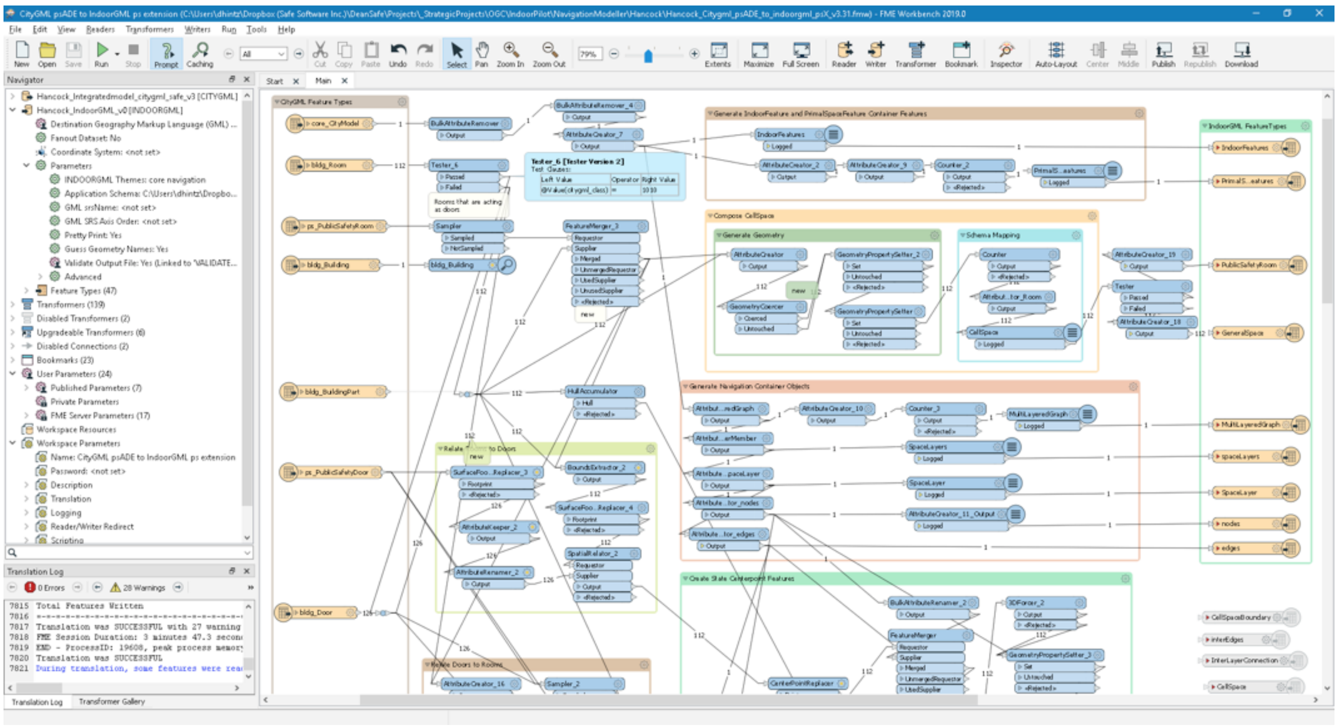
The ETL tool provides a number of functions mapped out in a feature flow diagram so that no coding is required. The primary transformation steps involved were: map schema from CityGML to IndoorGML, generate IDs, parent IDs, and parent property types to satisfy IndoorGML schema / UML requirements, and establish required nesting and parent child feature relationships. There were also business rules added to generate default values required for each feature type. Some of these are constants and some are based on feature context including public safety information.

Finally, a navigation network topology was generated and decomposed into nodes and linkages to support navigation. The navigation network was auto-generated within the cell spaces of the Indoor GML model using the state and transition feature types. The basic approach was to generate a Triangulated Irregular Network (TIN) and then remove all the edges that crossed wall boundaries. Doors were used to cut holes in the walls to allow the network to traverse the building. In some datasets a few key doors were missing so a few transition features were added to better support network connectivity. Finally, the output was validated against the IndoorGML Public Safety Extension schemas to make sure that the results were consistent with the requirements of the IndoorGML standard.

The main process steps for the Navigation Modeler (NM) Service are as follows:

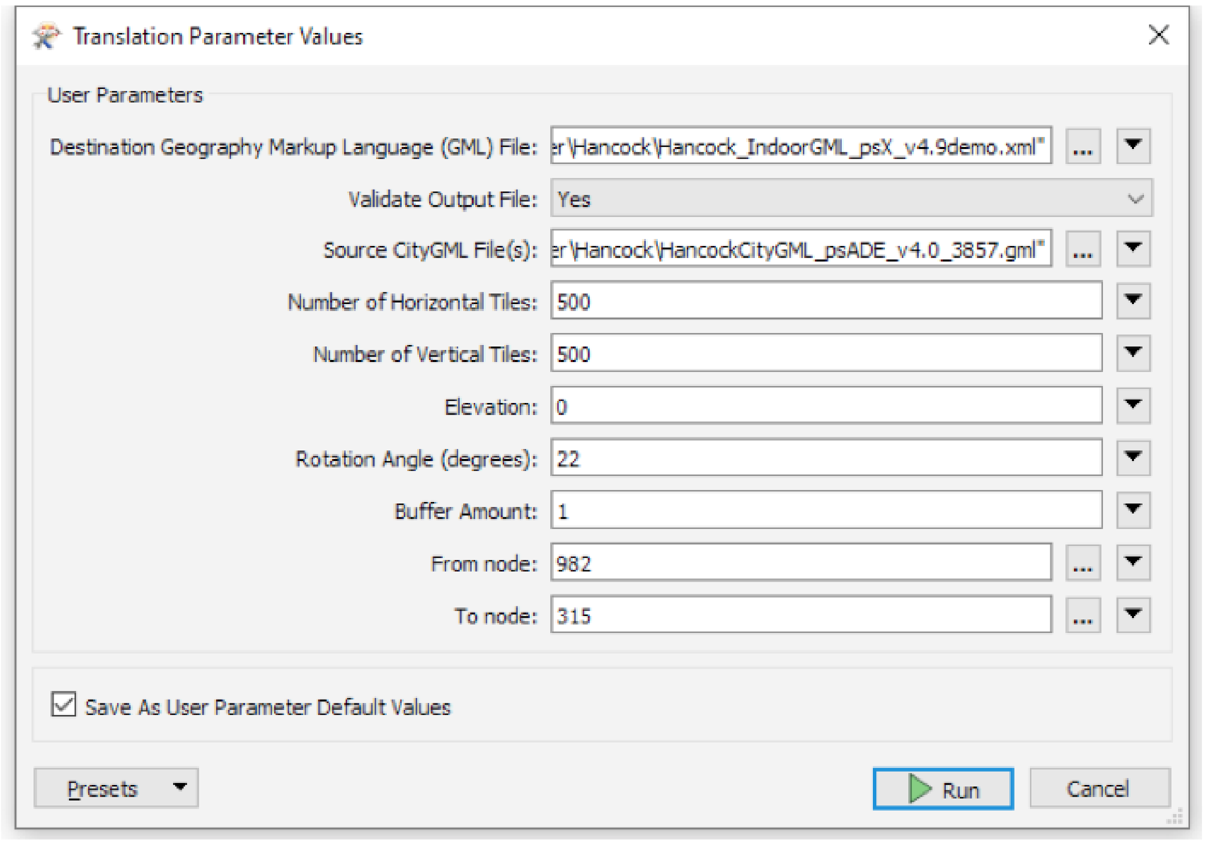
1. Query the CSW for a given building (CSW GET GetRecordById).
2. Parse the CSW response metadata and retrieve the associated CityGML via resolving dataset URL.
3. Convert CityGML 2.0 Public Safety ADE to OGC Indoor GML using data transformation model developed in FME:
   a. Map schema from CityGML to Indoor GML.
   b. Set default values where none are available in source.
   c. Merge indoor context information such as public safety data.
   d. Generate nodes and linkages to support navigation.
   e. Generate OGC IndoorGML v1.0. data in LLWGS84 (EPSG:4326) or Web Mercator (EPSG:3857) using an IndoorGML Public Safety Extension to IndoorGML in order to capture information related to the CityGML public safety ADE.
4. Perform validation and publish validated IndoorGML to publicly available file service end point.
5. Generate metadata and publish metadata record with building ID and dataset link to CSW (CSW POST Transaction Insert)

For more information including the FME workspace model used to transform from CityGML to IndoorGML see: https://knowledge.safe.com/articles/96851/ogc-indoor-gml-pilot.html



```
7815 Total Features Written
7816 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
7817 Translation was SUCCESSFUL with 27 warning
7818 FME Session Duration: 3 minutes 47.3 secon
7819 END - ProcessID: 19608, peak process memor
7820 Translation was SUCCESSFUL
7821 During translation, some features were rea
```

Figure 54. FME Data Transformation Workspace / ETL model to Translate from CityGML Public Safety ADE to IndoorGML Public Safety Extension for Hancock County Data


Translation Parameter Values

User Parameters

Destination Geography Markup Language (GML) File:	er\Hancock\Hancock_IndoorGML_psX_v4.9demo.xml"
Validate Output File:	Yes
Source CityGML File(s):	er\Hancock\HancockCityGML_psADE_v4.0_3857.gml"
Number of Horizontal Tiles:	500
Number of Vertical Tiles:	500
Elevation:	0
Rotation Angle (degrees):	22
Buffer Amount:	1
From node:	982
To node:	315

☑ Save As User Parameter Default Values

Figure 55. CityGML-to-IndoorGML Runtime Parameters

Runtime parameters control specification of CityGML input path, IndoorGML output path, navigation network grid size (vertical and horizontal tiles), rotation angle, and validation.

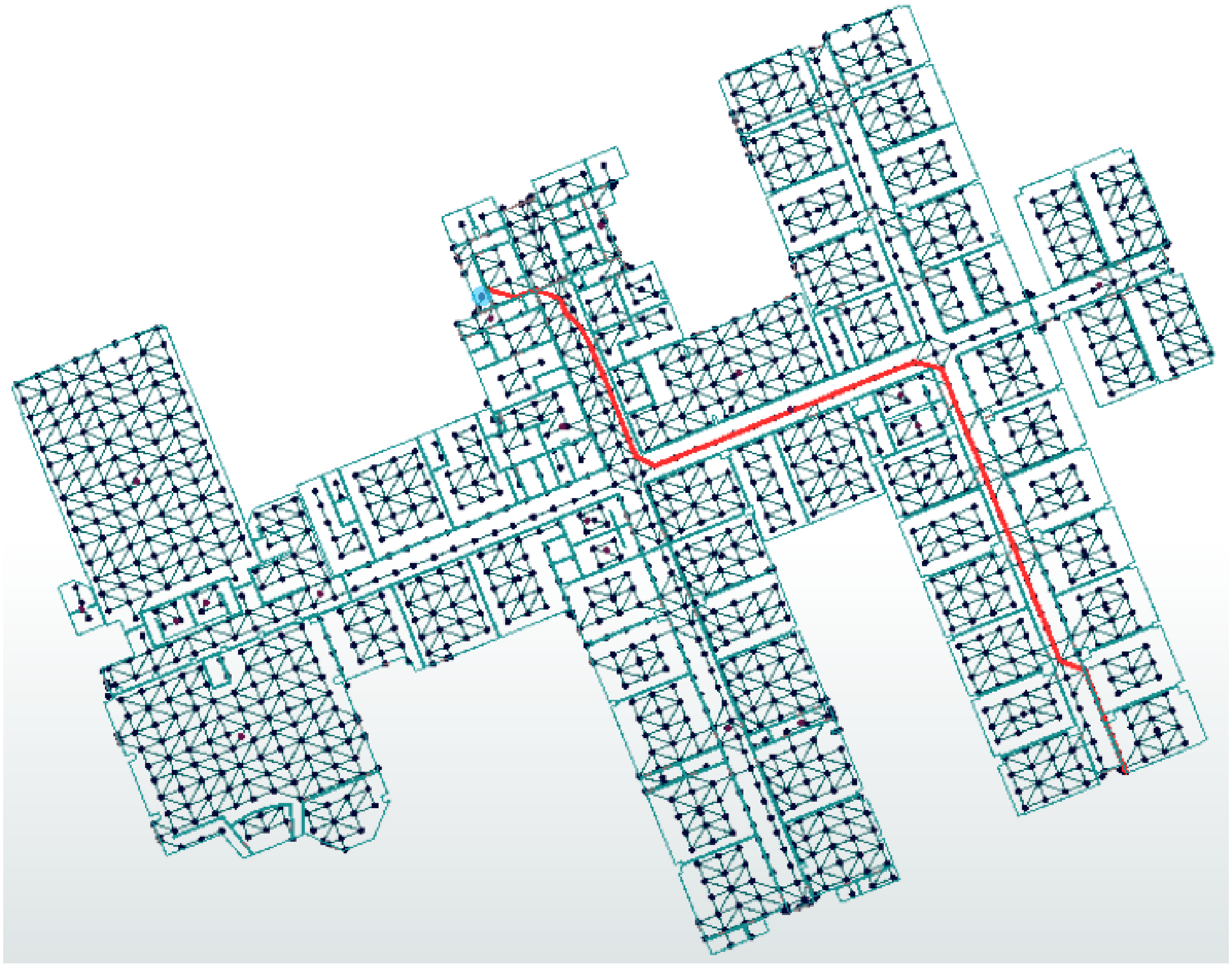
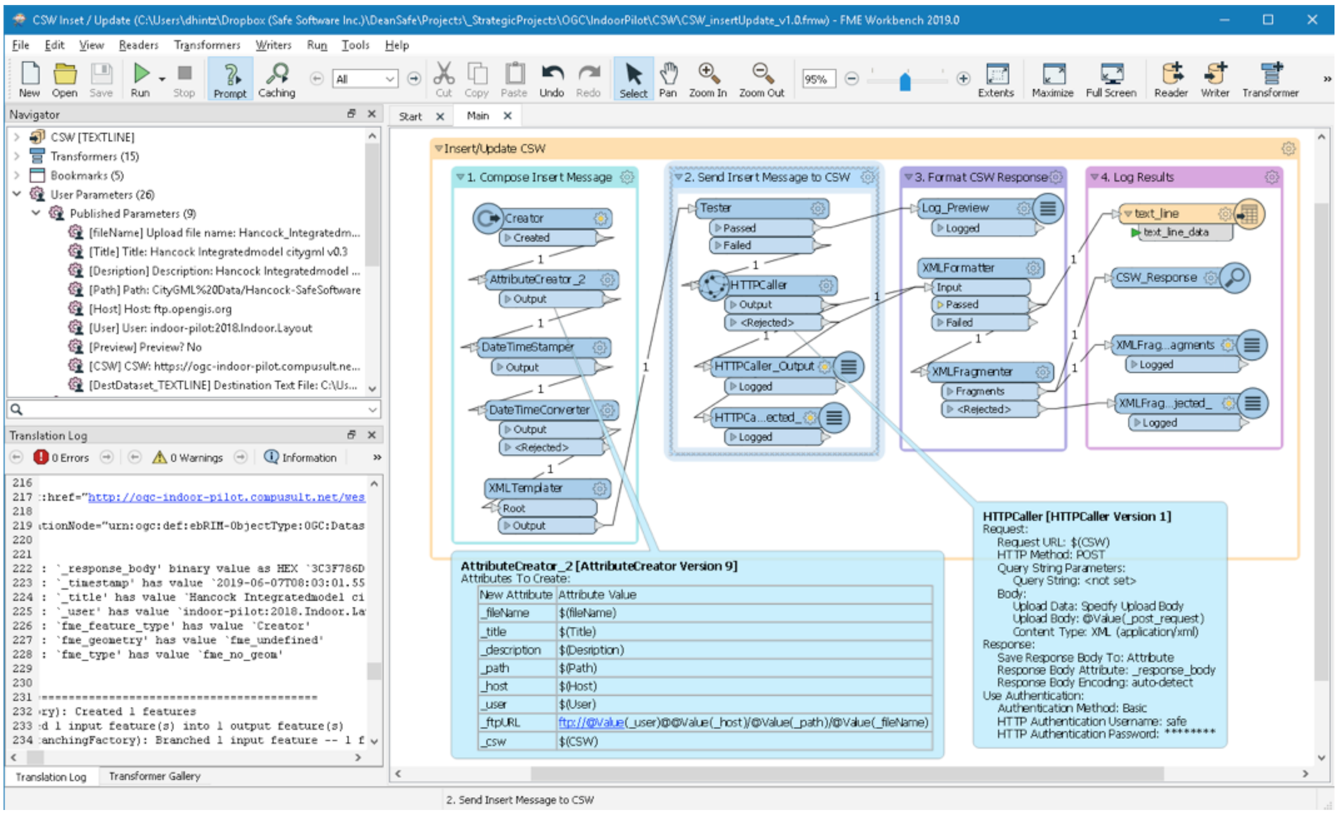


Figure 56. Navigation Modeler Service Network Output with Shortest Path Calculation

The core requirement for the Navigation Modeling Service is the ability to convert from CityGML Public Safety ADE to IndoorGML. During the kickoff meeting it was agreed that the central reference for all datasets would be an Indoor Pilot CSW. Each service that generates a new dataset publishes it to the CSW. Any service that requires a dataset can request it from the CSW based on its ID. The CSW also lists all available buildings by name and ID.


CSW Inset / Update (C:\Users\dhintz\Dropbox (Safe Software Inc.)\DeanSafe\Projects_StrategicProjects\OGC\IndoorPilot\CSW\CSW_insertUpdate_v1.0.fmw) - FME Workbench 2019.0
File Edit View Readers Transformers Writers Run Tools Help
New Open Save Run Stop Prompt Caching Cut Copy Paste Undo Redo Select Pan Zoom In Zoom Out Extents Maximize Full Screen Reader Writer Transformer
Navigator
Start Main
Insert/Update CSW
1. Compose Insert Message
Creator
Created
AttributeCreator_2
Output
DateTimeStamper
Output
DateTimeConverter
Output
<Rejected>
XMLTemplater
Root
Output
2. Send Insert Message to CSW
Tester
Passed
Failed
HTTPCaller
Output
<Rejected>
HTTPCaller_Output
Logged
HTTPCa...ected_
Logged
3. Format CSW Response
Log_Preview
Logged
XMLFormatter
Input
Passed
Failed
XMLFragmenter
Fragments
<Rejected>
4. Log Results
text_line
text_line_data
CSW_Response
XMLFrag...agments
Logged
XMLFrag...jected_
Logged
Translation Log Transformer Gallery
2. Send Insert Message to CSW

CSW [TEXTLINE]
Transformers (15)
Bookmarks (5)
User Parameters (26)
Published Parameters (9)
[fileName] Upload file name: Hancock_Integratedm...
[Title] Title: Hancock Integratedmodel citygml v0.3
[Desription] Description: Hancock Integratedmodel ...
[Path] Path: CityGML%20Data/Hancock-SafeSoftware
[Host] Host: ftp.opengis.org
[User] User: indoor-pilot:2018.Indoor.Layout
[Preview] Preview? No
[CSW] CSW: https://ogc-indoor-pilot.compusult.ne...
[DestDataset_TEXTLINE] Destination Text File: C:\Us...

Translation Log: 0 Errors, 0 Warnings, Information

```
:href="http://ogc-indoor-pilot.compusult.net/wes

tionNode="urn:ogc:def:ebRIM-ObjectType:OGC:Datas


 : `_response_body' binary value as HEX `3C3F786D
 : `_timestamp' has value `2019-06-07T08:03:01.55
 : `_title' has value `Hancock Integratedmodel ci
 : `_user' has value `indoor-pilot:2018.Indoor.La
 : `fme_feature_type' has value `Creator'
 : `fme_geometry' has value `fme_undefined'
 : `fme_type' has value `fme_no_geom'


===========================================
ry): Created 1 features
d 1 input feature(s) into 1 output feature(s)
anchingFactory): Branched 1 input feature -- 1 f
```

AttributeCreator_2 [AttributeCreator Version 9]
Attributes To Create:

New Attribute	Attribute Value
_fileName	$(fileName)
_title	$(Title)
_description	$(Desription)
_path	$(Path)
_host	$(Host)
_user	$(User)
_ftpURL	ftp://@Value(_user)@@Value(_host)/@Value(_path)/@Value(_fileName)
_csw	$(CSW)

HTTPCaller [HTTPCaller Version 1]
Request:
  Request URL: $(CSW)
  HTTP Method: POST
  Query String Parameters:
    Query String: <not set>
  Body:
    Upload Data: Specify Upload Body
    Upload Body: @Value(_post_request)
    Content Type: XML (application/xml)
Response:
  Save Response Body To: Attribute
  Response Body Attribute: _response_body
  Response Body Encoding: auto-detect
Use Authentication:
  Authentication Method: Basic
  HTTP Authentication Username: safe
  HTTP Authentication Password: ********

Figure 57. FME Workspace for interfacing with the CSW

Figure 57 shows the FME Workspace which was used to automate the posting of IndoorGML results generated by NIST004 to the CSW.

The primary risk factors to implementing this service were:

- Development of the CityGML Public Safety ADE and IndoorGML Public Safety Extension. This took a significant amount of collaborative effort and, naturally, changes to these schemas did require significant adjustments and retooling of transformation algorithms.
- Quality and type of incoming CityGML was not known ahead of time. Some of the earlier **point cloud-to-CityGML** sample datasets were composed of a few very large mesh features representing large building elements. These were not relatable to individual rooms, walls, or doors and, as such, were not usable for navigation.
- Availability of format and schema of data required by IndoorGML but not present in CityGML not yet known (e.g., occupancy, room use, and public safety information were often not available from the source CityGML, etc.)
- Basic validation for the output IndoorGML was conducted by validating the GML against the IndoorGML application schemas and Public Safety Extension schemas. Beyond this, there were few business rules provided to help qualify the quality of indoor data or assure accuracy of the output relative to the input IFC or CityGML.
- Navigation service example requests and use cases that need to be answered by IndoorGML

7.2.1. Considerations

CityGML is a useful input format for the Navigation Modeler because it is a mature and widely adopted standard, with many datasets available in CityGML format. CityGML is a good focal point for building data because it can be created from IFC, IMDF, BIM, and other GIS formats. Safe

Software generated CityGML data from Apple IMDF in order to create test data to begin early development and testing of the Indoor Navigation Modeler service.

IndoorGML is primarily an exchange format, not an application format. Examples of typical application formats include Apple IMDF and Google KML. IndoorGML’s core purpose is to provide information exchange to support indoor navigation. Domain application specific content should remain in the extensions. For example, the public safety features developed for IndoorGML belong as features defined in the schema of the Public Safety Extension, not in the core IndoorGML. While CityGML data is a good source of building data since it is widely available, IndoorGML is ideally suited for indoor mapping and navigation – the application for which it is designed.

The Navigation Modeler component converts from CityGML to IndoorGML from which navigation routes are calculated. CityGML is potentially richer in 3D physical representation and also contains more information than is typically needed for indoor applications. CityGML models are comprised of 3D surface structures. The intent of IndoorGML is to model interior spaces and the relationship between those spaces for the purposes of navigation. For example, interior furniture or installations are multi-surface solids in CityGML, whereas in IndoorGML they are modeled as 3D points. Point Clouds are converted into CityGML multi-surface objects to provide cleaner, discrete physical representations of the spaces and objects being modeled. This is then converted into IndoorGML, which is a simpler, logical representation of indoor spaces, connectivity, and points of interest.

7.2.2. Lessons Learned

- CityGML doors: Correct door definition is critical to any indoor mapping model, but often presents challenges when it comes to interpreting the type and state of the doors. Some doors were miscategorized as rooms which broke the network. The workaround to this was to filter these features out by their ids and correct their categorization.
- Depending on the coordinate transformations, some datasets had a north up layout while others such as Hancock County are oriented at a significant angle away from north up. In the latter case, we used a calibration factor to rotate the navigation network so that it aligned with the building corridors to produce a navigation network in line with building corridors. In the future, logic could be added to detect corridor orientation so that this rotation could be applied automatically.
- There were a variety of approaches that proved useful in geometry simplification. In the case of the Hancock County CityGML, the doors were complex multisurfaces. These were simplified into vertical planar surfaces using a combination of FME and ETL functions including a PlanarityFilter followed by a Tester set to filter out all faces except those whose Z surface normal = 0. This proved to be a useful approach to filter out everything except vertical surfaces.
- CenterPointReplacer was used to convert building installation objects represented as solids or multisurfaces in CityGML, into points of interest that were represented as a simple 3D point in IndoorGML.
- In practical situations, first responders need to transition between buildings or across outdoor spaces and in / out of buildings. To this end, focusing only on indoor spaces is not sufficient. There needs to be connection points which link buildings with external / outside spaces so that indoor / outdoor navigation can happen seamlessly.

- There is a need for a Quality Assurance process for data quality (e.g., Data Quality DWG). At the moment validation of output is primarily against the IndoorGML schemas including the Public Safety Extension. Other business rules need to be developed to validate the data content. Rooms need doors. Rooms should cover the level. Doors and room walls should be coincident. Rooms should be enclosed spaces. In FME, the Geometry Validator transformer was used for some of these checks.
- For the purpose of this pilot, a published parameter for navigation grid resolution was provided (horizontal and vertical tiles) to enable control of the granularity of navigation.
- The multi-layer graph generation from CityGML to IndoorGML may have varied results depending on the algorithm and resolution applied. These non-uniform results, when processed for navigation, could produce different navigation routes depending on the graph generated. Different applications may require different types of navigation routes. Basic instructions to get from room A to room B might only require room sequence information provided by a simple center-point network. Routing to support a first responder and provide the shortest path from point A to B across rooms of various sizes and with multiple entrances would likely require a higher resolution navigation network that supports multiple possible routes across a given room depending on obstacles, hazards, start and end points etc.
- Overall, once configured, spatial ETL transform models proved capable of automating most of the process for converting indoor building data from one format to another, particularly when the source and destination formats conformed to clearly defined open standards such as CityGML, IndoorGML or IFC. The initial FME transform model was developed to convert Victoria Airport CityGML to IndoorGML. Once this model was developed and tested, the same model was employed to convert the PNU Central Plaza CityGML to IndoorGML. Some modifications were required to manage anomalies in each dataset - such as misclassified doors. Also, edits were made to accommodate changes to the Public Safety ADE and enhancements to improve navigation network generation. However, by the time the Hancock dataset became available, only minimal calibration changes were required to perform the conversion (e.g. navigation network resolution and alignment calibration).

7.3. Skymantics Navigation Modeler

Skymantics was a relatively newer user of the CityGML and IndoorGML standards, enabling them to provide a fresher perspective on the tasks of implementing these standards. Skymantics developed the Navigation Modeler Service using WPS as the primary interface. The main process steps for the Navigation Modeler (NM) Service are as follows:

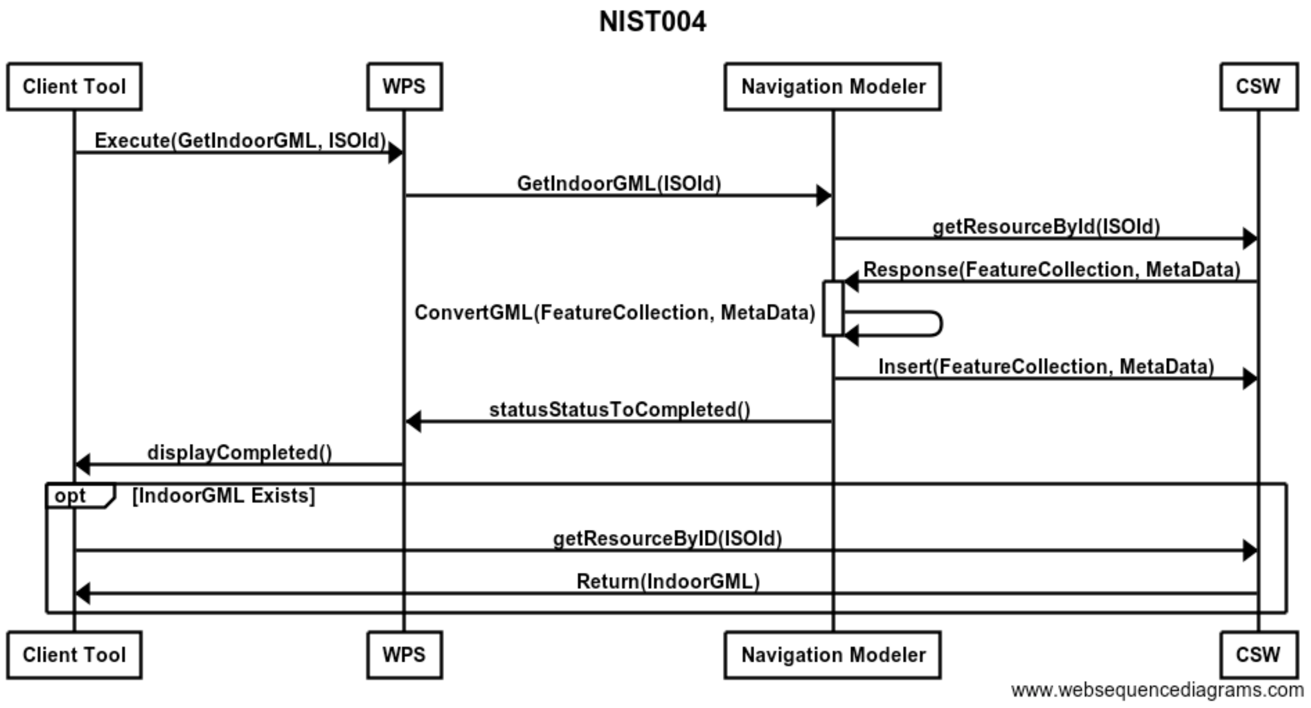

NIST004
Client Tool
WPS
Navigation Modeler
CSW
Execute(GetIndoorGML, ISOId)
GetIndoorGML(ISOId)
getResourceById(ISOId)
Response(FeatureCollection, MetaData)
ConvertGML(FeatureCollection, MetaData)
Insert(FeatureCollection, MetaData)
statusStatusToCompleted()
displayCompleted()
opt [IndoorGML Exists]
getResourceByID(ISOId)
Return(IndoorGML)
www.websequencediagrams.com

Figure 58. Updated Component Architecture

1. Client Tool makes a request to the WPS to retrieve IndoorGML data based on CityGML data as an input.
2. The WPS processes the request and submits a job to the Navigation Modeler service.
3. The Navigation modeler retrieves the information from the CSW based on the job inputs and processes the CityGML-to-IndoorGML conversion.
4. The Navigation modeler reinserts the new IndoorGML converted data back into the CSW.
5. The Navigation modeler responds to the WPS to notify completion of the job.
6. The WPS provides the client with the URL of the converted IndoorGML data.
7. The client is able to retrieve the completed IndoorGML from the CSW.

The Indoor Navigation Modeler is responsible for CityGML retrieval and IndoorGML insertion compliant with the OGC WPS 2.0 standard and its associated domain logic. The server consumes GML data from the CSW and produces IndoorGML compliant with the requirements of Indoor Navigation for the Pre-planning Tool Client. The front-facing WPS server is 52°North's Java based implementation of WPS 2.0, JavaPS. This WPS server is hosted on a Tomcat instance on a Docker-based Debian GNU/Linux 8 distribution, hosted on a Microsoft Azure Cloud server. The backend Processing Service is a Java-based implementation for transforming the relevant CityGML-to-IndoorGML data, in part utilizing the open source library citygml4j. The WPS provides asynchronous, scalable capability by containerizing these Processes using Docker.

7.3.1. Lessons Learned

CityGML appears to be a very large standard (i.e., over 270 pages). This standard covers various Levels of Detail (LOD), which it was later discovered that only LOD4 pertained to the pilot. The LOD4 describes the exterior and interior geometries of buildings, openings (e.g., doors, windows, etc.), and interior objects such as furniture. These interior objects could include public safety features such as fire extinguishers, shut off valves, etc. which apply to the public safety scenario of this pilot.

The sponsor requirements mandated, however, that additional metadata was needed to describe certain public safety specific features regarding interior objects such as the direction a door swings inward/outward, symbology, and hazards such as oxygen tanks, etc. In order to incorporate this data, the Public Safety ADE was developed and incorporated into CityGML. The Public Safety ADE extends features within CityGML with additional properties such as whether a door is locked and direction it opens.

Automatic generation of IndoorGML SpaceLayers was very challenging without utilizing existing enterprise tools. Complex polygons, such as a horseshoe shaped room, are difficult to conduct using fast and efficient algorithms. As such, transformations such as Delaunay Triangulations of CityGML Geometry were required. Our approach was achieved using FME Workbench to generate a TIN of the topographical space. Even a semi-manual process such as this may result in bugs or stranded States. Additionally, it is highly difficult to utilize all IndoorGML specifications in an automated fashion based purely on Geometry alone for converting CityGML to IndoorGML due to CityGML's more generalized and abstracted semantic definitions. An example of this is CityGML's standard Room definition. IndoorGML specifies subtype of Rooms, such as hallways or archways. Without additional semantic information, it is very difficult for an automated process to differentiate between a very long and narrow room and a hallway. Some manual definition of these spaces is required in order to utilize some the respective IndoorGML definitions in the case of this pilot.

In order to convert CityGML to IndoorGML, the Java Architecture for XML Binding (JAXB) was used for transformation of the XML data types. It was discovered that an open source bindings based on XML-Java Binding (XJB) exist, specifically for CityGML to use with JAXB, called ogc-schemas. These files are bindings that are required in order to transform CityGML markup into Java class representations. Using the XJC command line interface (CLI) and the ogc-schemas bindings, the CityGML java objects can be generated. Once the objects are generated, additional properties can be annotated. Then the objects and be transformed into different representations such as IndoorGML.

Some challenges were encountered during development. Writing XJB bindings for such a large schema such as CityGML is too complex due to the number of dependencies required considering the scope and the development time allocated for the pilot. It was also discovered that there is a mismatch of GML versions between CityGML and IndoorGML. CityGML is an application schema of GML version 3.1.1 whereas IndoorGML targets GML version 3.2.1 which created issues. For example, a solid property type in CityGML is different than a solid property type in IndoorGML. It was learned that objects of type GML 3.1.1 could not be casted to objects of type GML 3.2.1. For example, CityGMLSolid = new IndoorGMLSolid() would result in a Typecast compilation error. Since IndoorGML and CityGML both inherit from GML types, manual generation of GML 3.2.1 was required to transform the CityGML to IndoorGML.

NOTE CityGML3 - the new version of CityGML – is currently being developed to support GML 3.2.1 (https://github.com/opengeospatial/CityGML-3.0Encodings).

As both CityGML and IndoorGML geometric representations use GML, and GML did not change the core representation of objects composed of primitive types such as character arrays, int, long, float etc., a developmental workaround was able to be implemented. The workaround is shown in the following Java listing.

```
private net.opengis.gml._3.SurfacePropertyType

CreateNewSurfaceMembers(List<Double> list){
  net.opengis.gml._3.DirectPositionListType outputPosList = new
net.opengis.gml._3.DirectPositionListType();
  outputPosList.getValue().addAll(list);
  net.opengis.gml._3.LinearRingType myring = new net.opengis.gml._3.LinearRingType();
  myring.setPosList(outputPosList);

JAXBElement<net.opengis.gml._3.LinearRingType> ring = objfac.createLinearRing(myring);
  net.opengis.gml._3.AbstractRingPropertyType absRing =
objfac.createAbstractRingPropertyType();
  absRing.setAbstractRing(ring);
  net.opengis.gml._3.PolygonType poly = objfac.createPolygonType();
  poly.setExterior(absRing);
  poly.setSrsDimension(BigInteger.valueOf(3));
  JAXBElement<net.opengis.gml._3.PolygonType> absSurface = objfac.createPolygon(poly);
  net.opengis.gml._3.SurfacePropertyType surfacePropertyType
=objfac.createSurfacePropertyType();
  surfacePropertyType.setAbstractSurface(absSurface);
  return surfacePropertyType;
}
```

First, the CityGML geometry primitives of GML type 3.1.1 were extracted. The geometry primitives are posList, which is a collection of doubles that represent geometric coordinates. Then, a new PosList of type GML 3.2.1 was created. JAXB object factories were used in this process to generate compliant GML classes. The CityGML primitive collection of type GML 3.1.1 would then be copied onto this newly generated 3.2.1 PosList. The geometric representation would then propagate up the abstraction hierarchy such that PosLists are nested into AbstractRings into Polygons into Surface Members. These surface members represent the walls, doors, and openings of a building space. However, this approach is prone to error because new code must be generated to map the appropriate version types for each mismatch, for example GML 3.1.1 to GML 3.2.1. This could become a continued issue and should be carefully considered in a future version of IndoorGML and the target application of the GML schema versions for interoperability.

Chapter 8. Building Model Repository

8.1. Compusult Building Model Repository

The Building Model Repository stores CityGML, IndoorGML, and Point Cloud URLs inside a Catalog service using a Catalog Service for Web-Transactional (CSW-T) interface. A client can search for information to discover the data and the available services. The CSW-T provides a response containing links to the data and/or services. A WFS-T is not provided since CSW-T can provide the necessary transactional requests to discover and access the services and data. For this pilot, the data is stored in an FTP location, and the URL to the data is stored in the CSW.

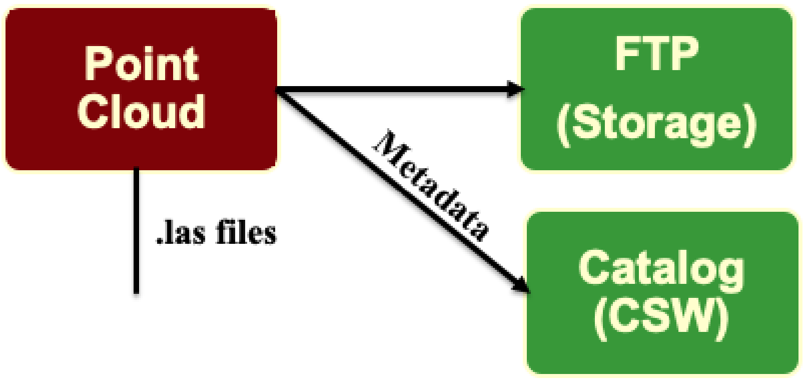


Figure 59. Building the Model Repository

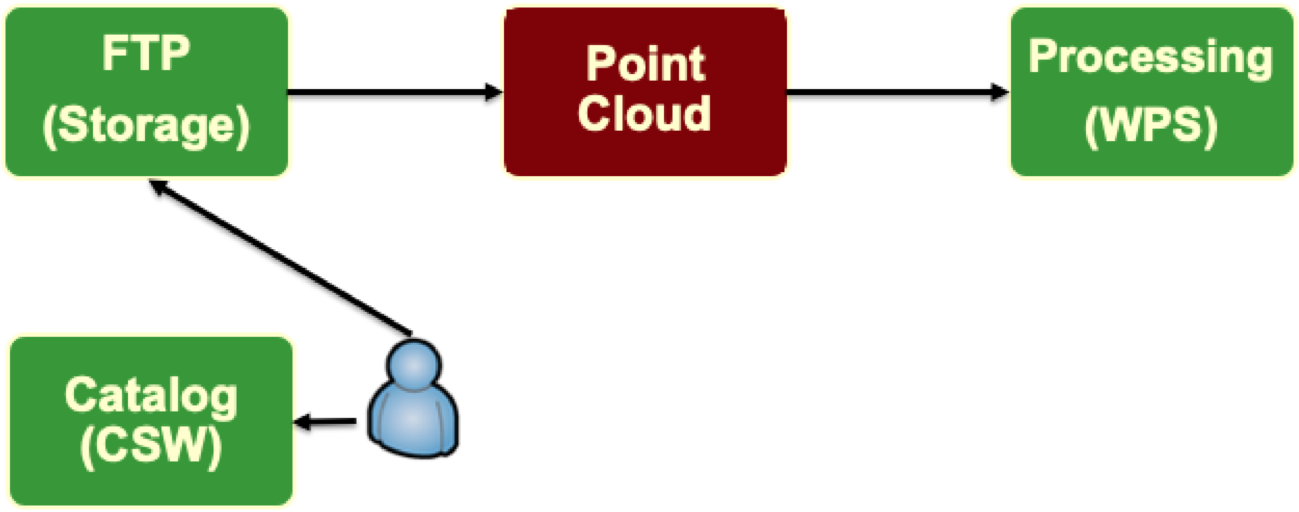


Figure 60. Requesting Data from the Repository

Compusult has a product called Web Enterprise Suite (WES) which provides the capability to interface with the CSW to retrieve Point Cloud, CityGML, and IndoorGML data and convert them into 3D Tiles and store them within a GeoPackage. Additionally, IndoorGML data is natively stored in a GeoPackage Related Tables extension. The IndoorGML Route is used for navigation purposes. In these formats, they are delivered via 3DPS to a Pre-planning Tool Client provided by the Compusult GoMobile Client. This enables offline viewing and navigation routing of the content on a mobile device in a disconnected environment, also known as Denied, Degraded, Intermittent, or Limited (DDIL) Bandwidth environment.

The Building Model Repository provided by Compusult performs according to the following sequence diagram.

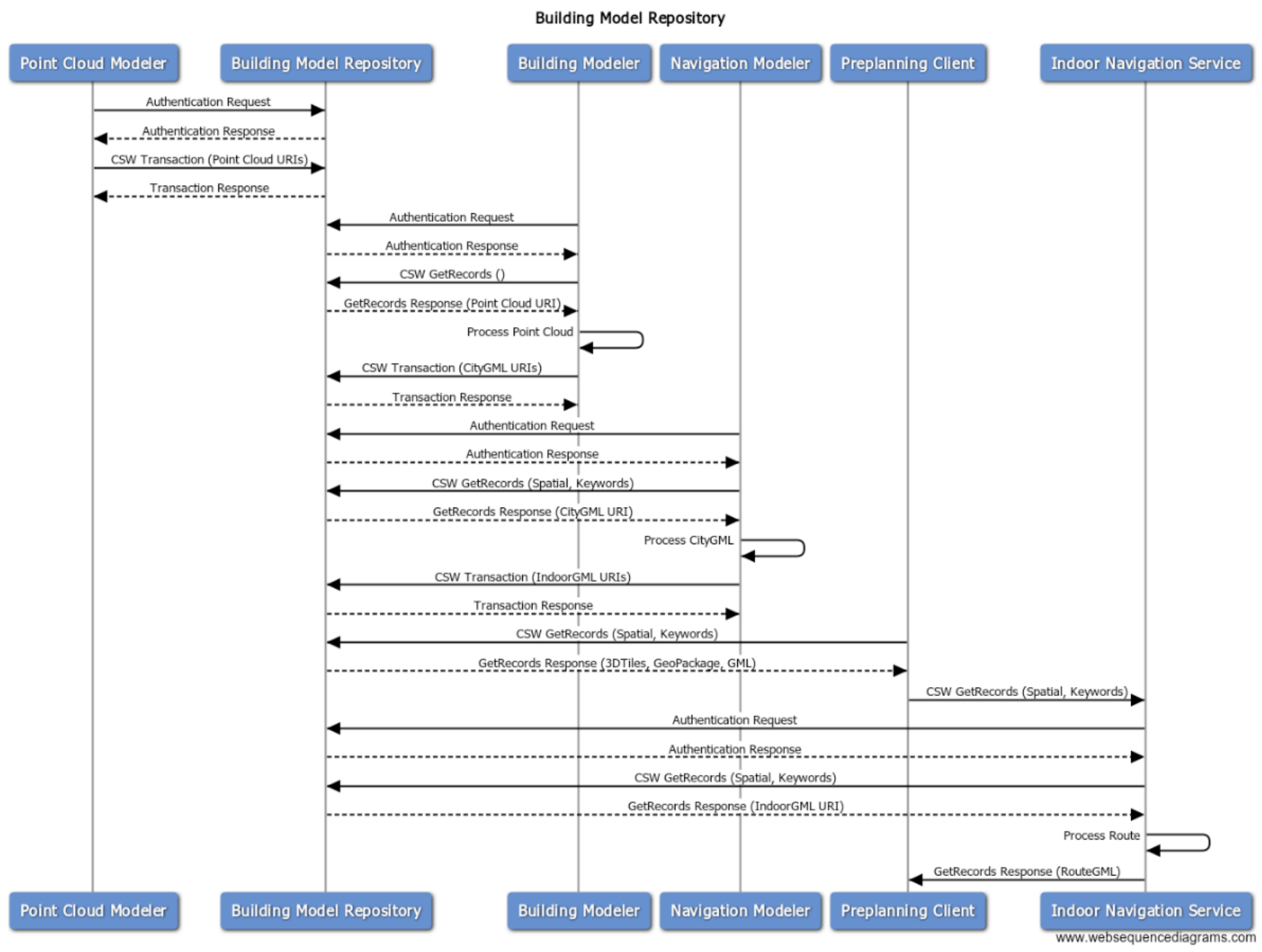

Building Model Repository
Point Cloud Modeler
Building Model Repository
Building Modeler
Navigation Modeler
Preplanning Client
Indoor Navigation Service
Authentication Request
Authentication Response
CSW Transaction (Point Cloud URIs)
Transaction Response
Authentication Request
Authentication Response
CSW GetRecords ()
GetRecords Response (Point Cloud URI)
Process Point Cloud
CSW Transaction (CityGML URIs)
Transaction Response
Authentication Request
Authentication Response
CSW GetRecords (Spatial, Keywords)
GetRecords Response (CityGML URI)
Process CityGML
CSW Transaction (IndoorGML URIs)
Transaction Response
CSW GetRecords (Spatial, Keywords)
GetRecords Response (3DTiles, GeoPackage, GML)
CSW GetRecords (Spatial, Keywords)
Authentication Request
Authentication Response
CSW GetRecords (Spatial, Keywords)
GetRecords Response (IndoorGML URI)
Process Route
GetRecords Response (RouteGML)
Point Cloud Modeler
Building Model Repository
Building Modeler
Navigation Modeler
Preplanning Client
Indoor Navigation Service
www.websequencediagrams.com

Figure 61. Building Model Repository Sequence Diagram

a. Users authenticate with the CSW service

b. Authenticated users execute a CSW Insert, Update or Delete Transaction using an HTTP POST request. The Insert transaction's POST data contains the URLs to the raw Point Cloud data, IndoorGML and/or the CityGML along with the Metadata. The Update transaction's POST data contains the filter criteria of the CSW record(s) to be updated to the Point Cloud URL, IndoorGML URL and/or CityGML URL data along with the Metadata. The Delete transaction's POST data contains the filter criteria of the CSW record(s) to be deleted.

c. The CSW object classifications shall be of the following types:

   ◦ urn:ogc:def:ebRIM-ObjectType:OGC:Dataset:3DData:Lidar

   ◦ urn:ogc:def:ebRIM-ObjectType:OGC:Dataset:3DData:CityGML

   ◦ urn:ogc:def:ebRIM-ObjectType:OGC:Dataset:3DData:IndoorGML

   ◦ urn:ogc:def:ebRIM-ObjectType:OGC:Dataset:3DData:IndoorGMLRoute

d. Clients execute a GetRecords or GetRecordByID request to the CSW to find Indoor Mapping Service records.

e. Clients extract the raw data URL to download the data files.

Interface Calls

- CSW INSERT, UPDATE and DELETE Transactions

    - Input
        - POST data with the URL(s) to the raw data files, CSW Classification Type, and associated Metadata
    - Output
        - <csw:TransactionResponse>
- CSW GetRecords
    - Input
        - Spatial, Keyword, and Classification Type (see CSW Classification Types above)
    - Output
        - <csw:GetRecordsResponse>
- CSW GetRecordByID
    - Input
        - RecordID
    - Output
        - <csw:GetRecordByIdResponse>

8.1.1. CSW Transactions

Examples are shown in Annex A - Appendix A, *Building Repository Examples*.

CSW Insert Transaction

The **CSW Insert Transaction** Post request initiates the ingestion of metadata describing the data with a link to the raw product. (see Section A.1, “CSW Insert Transaction POST Request Example”)

CSW Update Transaction

The **CSW Update Transaction** Post request updates the metadata and/or raw product link. (see Section A.3, “CSW Update Transaction Request Example”)

CSW Delete Transaction

The **CSW Delete Transaction** Request deletes the records in the CSW matching the criteria specified. (see Section A.5, “CSW Delete Transaction Request Example”)

CSW GetRecords Transaction

The **GetRecords** Request returns the records that meet the criteria specified. (see Section A.7, “CSW GetRecords Request Example”)

CSW GetRecordByID Transaction

The **GetRecordByID** Request returns the record for the specified ID. (see Section A.11, “CSW GetRecordByID Response Example”)

8.1.2. Lessons Learned

Compusult did not have access to a stable basic dataset to work with at the beginning of the pilot.

Most datasets that were initially provided by participants had issues (e.g., CRS not specified, missing GML tags, etc.), which led to a series of iterations to try to determine the issues and work with the corresponding participants to get the data cleaned up. In order to create a complete scenario with building models, navigation routes, and public safety features, all datasets (point cloud, IndoorGML, CityGML, and Public Safety) are required. Sometimes only a subset of these were available. Without IndoorGML, the navigation routing cannot be provided. Without CityGML, the building model cannot be represented. Without Public Safety feature point-of-interest, we cannot render things such as fire extinguishers, gas shut off valves, etc.

Another challenge faced by Compusult was that participants did not always implement the CityGML or IndoorGML standards in the same way, which caused inconsistencies in the data and resulted in errors or unexpected outputs when new data was received. To avoid such pitfalls in the future, validation tools could be developed for the datasets to ensure contributions follow the standards specifications more closely.

Compusult worked with the other participants to identify the discrepancies and work through the issues. The following issues were identified:

- Missing Coordinate Reference System (CRS)
- Use of different coordinate systems without specifying CRS
- Inconsistent use of namespaces
- Missing datasets (e.g., Point Cloud and CityGML existed but not the corresponding IndoorGML, having CityGML and IndoorGML datasets but missing Point Cloud data (Victoria Airport), or missing CityGML Public Safety ADE or IndoorGML PS Extension.)

Chapter 9. Indoor Navigation Service

The purpose of the **Indoor Navigation Service** is to provide the optimal route for a firefighter to safely reach their target within the building. Conceptually, the IndoorGML data should already be annotated with the PS extension and provide notable cues for guidance throughout the navigable path. The indoor navigation service is intended to help first responders to navigate the interior spaces of a building during an emergency. It creates a route between two points, a start and an end point. The service is also intended to support navigation to public safety features along the way such as fire extinguishers, shut-off valves, etc. or avoid hazards such as locked doors, rooms with oxygen tanks, etc. The indoor navigation service takes IndoorGML as an input and parameters from the pre-planning client to generate an IndoorGML route.

9.1. Skymantics Indoor Navigation Service

Skymantics' Indoor Navigation Service was developed using Java, specifically JDK 1.8. This includes libraries such as the Apache Commons networking library to extend FTP and HTTP access, as well as the JAXB XML processing libraries for the consumption and generation of XML data based on the IndoorGML schema specifications. This domain logic was integrated into JavaPS [https://github.com/52North/javaPS], an Open Source WPS implementation from 52North. The WPS was hosted on a Tomcat 8.0 instance contained in Docker which was then deployed to an Azure Cloud server. An FTP endpoint specified by the client is used to extract the relevant IndoorGML for a given navigation area. The JAXB library is then used to create IndoorGML Java Objects from their respective markup classes using .xjb binding files provided by the Open-Source ogc-schemas repository.

The Navigation Service extracts the State-Transition Network from the IndoorGML topographical SpaceLayer and proceeds to parse the graph, associating the various states to their respective transitions, whilst also checking the semantic definition of States or transitions to delineate hazards (e.g., locked doors). These public safety enhanced state / transitions are assigned specific weights if none are provided. For example, the connected Transition of any public safety hazard has all its inbound edges set to a weight of Infinity. Likewise, Transitions that represent locked doors also have their weight set to Infinity. This guarantees that an inaccessible node will not be considered during the Navigation process. An implementation of Dijkstra’s algorithm was used to navigate interior spaces, utilizing IndoorGML’s SpaceLayer architecture to calculate the most eligible route for a given Start and End point. Functionally, the Navigation Service takes two IndoorGML states from the topographical SpaceLayer. These states are chosen by the end user in the pre-planning client application. The Navigation Service then calculates the shortest, most viable path from the Start state to the End state. The result of this process is an ordered set of **Route** objects. This list is then deserialized using the relevant .xjb binding to produce schema-compliant IndoorGML that represents a Route, with the relevant states as RouteNodes and transitions between them as RouteSegments. This markup is then sent as a response to the requesting Client tool. The Indoor Navigation Service provided by Skymantics performs according to Figure 62.

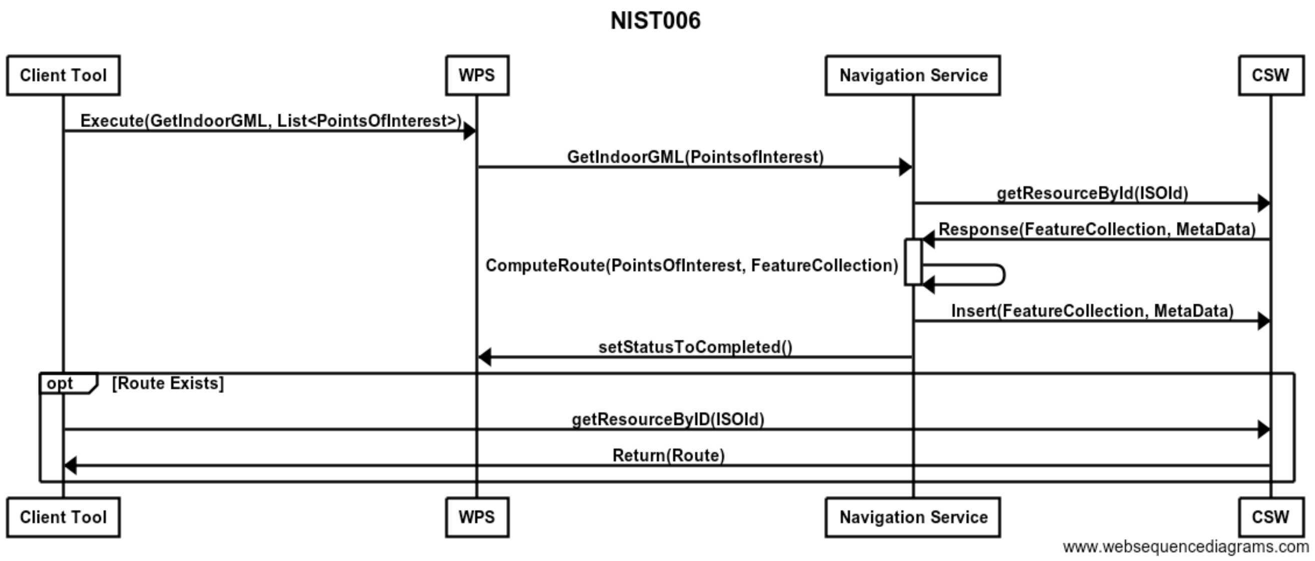

NIST006
Client Tool
WPS
Navigation Service
CSW
Execute(GetIndoorGML, List<PointsOfInterest>)
GetIndoorGML(PointsofInterest)
getResourceById(ISOId)
Response(FeatureCollection, MetaData)
ComputeRoute(PointsOfInterest, FeatureCollection)
Insert(FeatureCollection, MetaData)
setStatusToCompleted()
opt
[Route Exists]
getResourceByID(ISOId)
Return(Route)
www.websequencediagrams.com

Figure 62. NIST006 Indoor Navigation Service

1. Client application sends selected Start and End StateId's and accompanying Catalog GUID to the WPS instance
2. WPS obtains the IndoorGML data associated with the GUID via the IndoorPilot FTP Server
3. WPS transforms this data via JAXB into an annotated Java Object
4. Domain Logic consumes this Java Object to extract the State-Transition network and transforms it into a graph representation
5. The Start and End states are fed into a graph traversal function based on Dijkstra's algorithm, and the most optimal route is calculated using the Weight property of the Transition class
6. The series of States visited are inserted into a Stack, which is then unwound and transformed into a Route object that mirrors IndoorGML Navigation modules Route schema
7. The WPS returns this Route Object in XML format back to the Client tool, adhering to the schema put forth for the IndoorGML navigation module.

9.1.1. Lessons Learned

One recommendation to the IndoorGML SWG is to eliminate or refactor the Node and Edge property of the SpaceLayer Class (see **SpaceLayerType** in Section B.3, "SpaceLayerType Example"). Specifically, the unbounded **maxOccurs** property of both Edge and Node classes, in conjunction with the **minOccurs** property of the Edge class, can result in IndoorGML markup in which States are stranded with no Transition. It may also result in improperly formatted Markup, in which connected states are erroneously logically separated into different Node collections. An expansion of the IndoorGML schema document of the NodeType and EdgeType classes, and their precise purpose, in regards to the schema, may eliminate some confusion associated with these particular objects.

9.2. Compusult Indoor Navigation Service

The Indoor Navigation Service provided by Compusult has the following functionality:

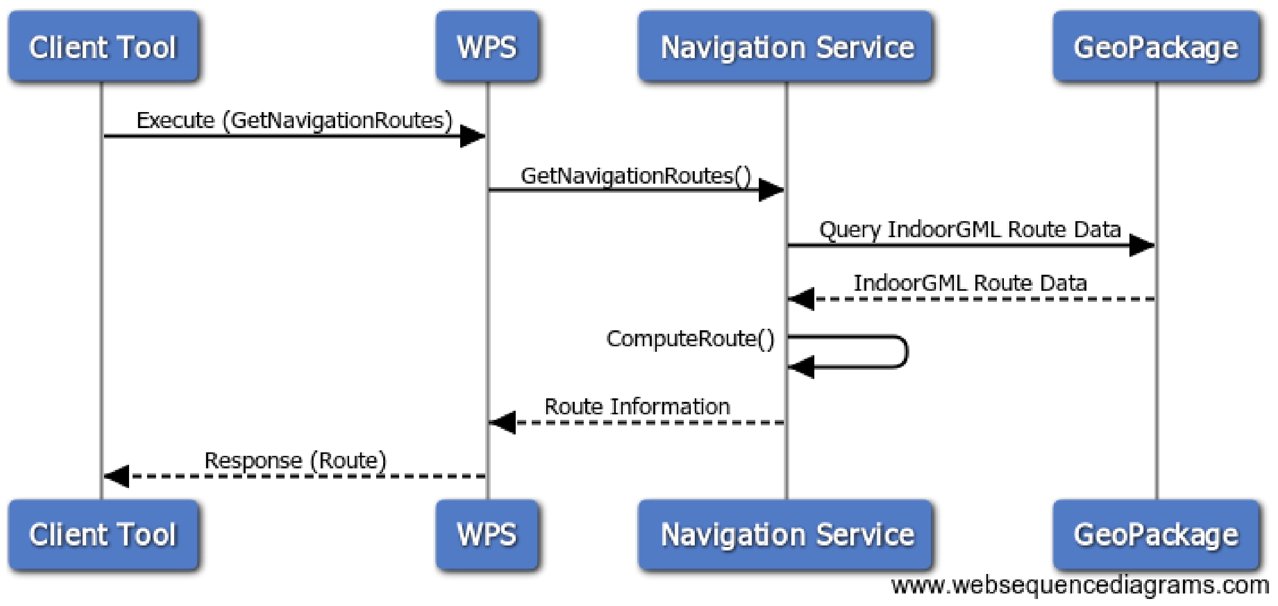


Figure 63. Data flow among components

Process Flow

a. Users will initiate a navigation request by specifying a start point, an end point in the GetNavigationRoutes request.

b. The Indoor Navigation Service will request IndoorGML data.

c. IndoorGML Data will be returned to the Indoor Navigation Service.

d. The Indoor Navigation Service will calculate the route.

e. The Route info (GML) will be passed back to the User for visualization in the Indoor Mapping Client

Interface Calls

- WPS GetNavigationRoutes
  - Input
    - IndoorGML ID
    - Start Point
    - End Point
  - Response
    - GML Navigation Routes (including Metadata)

Examples of interface calls are provided in Appendix C, *Compusult Indoor Navigation Service Examples*.

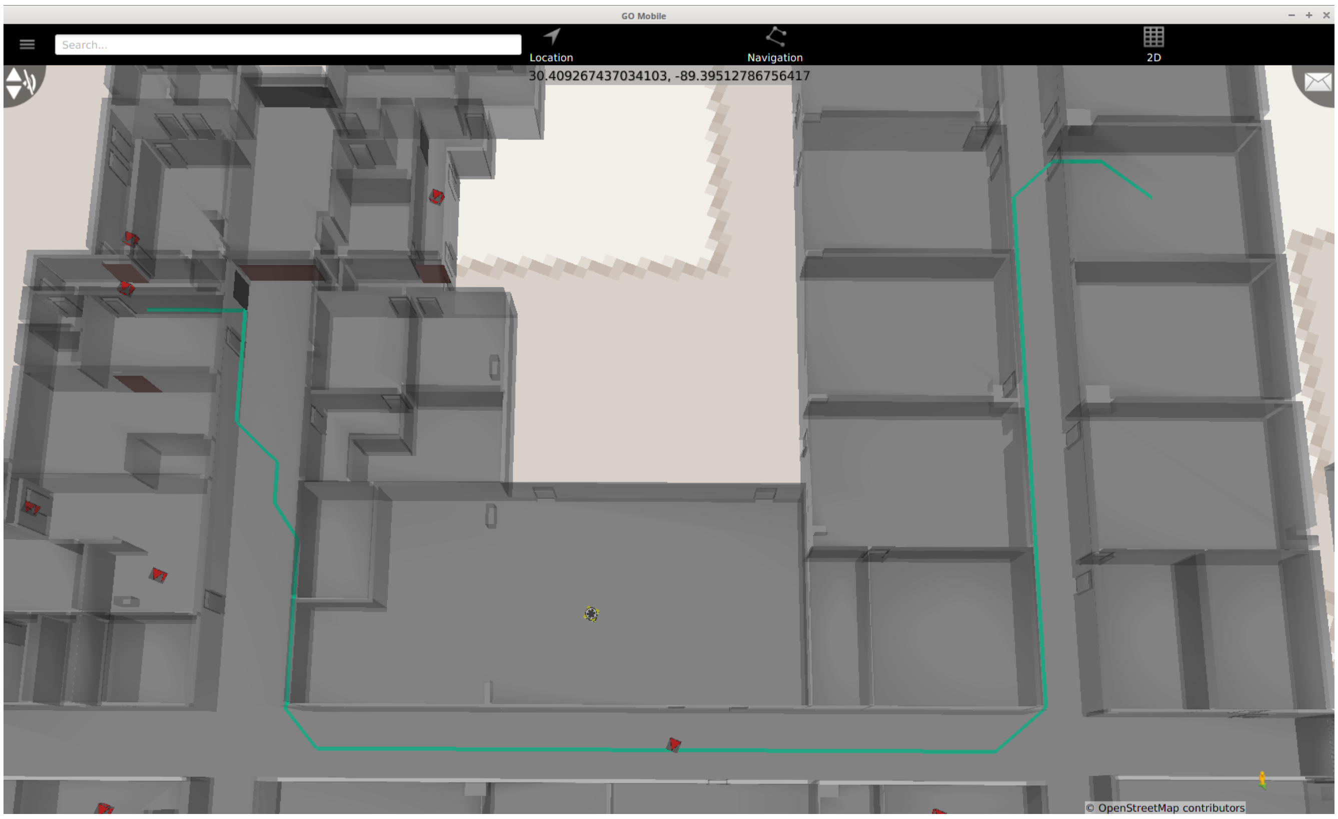


Figure 64. Indoor Navigation Route Displayed in Pre-Planning Client

9.2.1. Lessons Learned

It was found that the Coordinate Reference System (CRS) was specified inconsistently or sometimes missing altogether in CityGML / IndoorGML. The following examples show different representations of the srsName. In the future there may need to be a standardized way of representing the srsName to increase the interoperability between various clients. Some examples include the following [1: Implementors should refer to OGC Naming Authority polices (https://www.opengeospatial.org/standards/na) for a description of the correct structure of URNs and URIs that identify CRS.]:

- srsName="EPSG3850"
- srsName="EPSG:3850"
- srsName="EPSG::3850"
- srsName="urn:ogc:def:crs,crs:EPSG::25832,crs:EPSG::5783"

The pre-planning client has the capability to turn on and off the display of specific floors. Initially, there was no specification of a floor / story so Compusult attempted a few approaches for the representation of a floor (such as using the z-value). However, a good solution could not be reached, so it was recommended to add a "Storey" attribute to IndoorGML which would represent the floor level. This was initially implemented as text in the comment for the element which is a suboptimal approach.

There was also some discussion about adding the Story as a parent element and grouping other child elements inside the parent element. This also highlights an issue between the various worldly spellings of some words. Storey is the British spelling and story is American English. [2: OGC officially uses American English in its documents.]

The Storey attribute is required to identify the floor or story being represented in the building. Knowing this would allow the client software to be able to turn on/off all the feature data for a given floor. For example, if the user only wanted to see the navigation points and public safety features for floor 4, this attribute would allow the software to identify the floors and disable all but Storey 4.

When defining the start and end navigation points for a route, there were differences in approaches to mapping the start and end points to elements in IndoorGML in an interoperable way. Initially, Compusult's implementation provided the start and end points as x, y, z values whereas Skymantics provided the start and end points as named points. All points in the IndoorGML model have a unique ID which is used to name the point. It was decided to change the implementation to match the Skymantics implementation to facilitate interoperability between the routing services. Both of these methods are viable, however, it was determined that using the named method was cleaner because there was no additional work required to find the closest point to a user's selection.

The routing information is stored in IndoorGML so it was decided to store the IndoorGML in the GeoPackage for offline routing in a Denied, Degraded, Intermittent, or Limited Bandwidth (DDIL) environment. This was accomplished by using the GeoPackage Related Tables extension to store the IndoorGML in its native format. This gave the additional benefit of being able to implement routing in DDIL environments as we could read the GML directly from the GeoPackage without the need to interface with the WPS service.

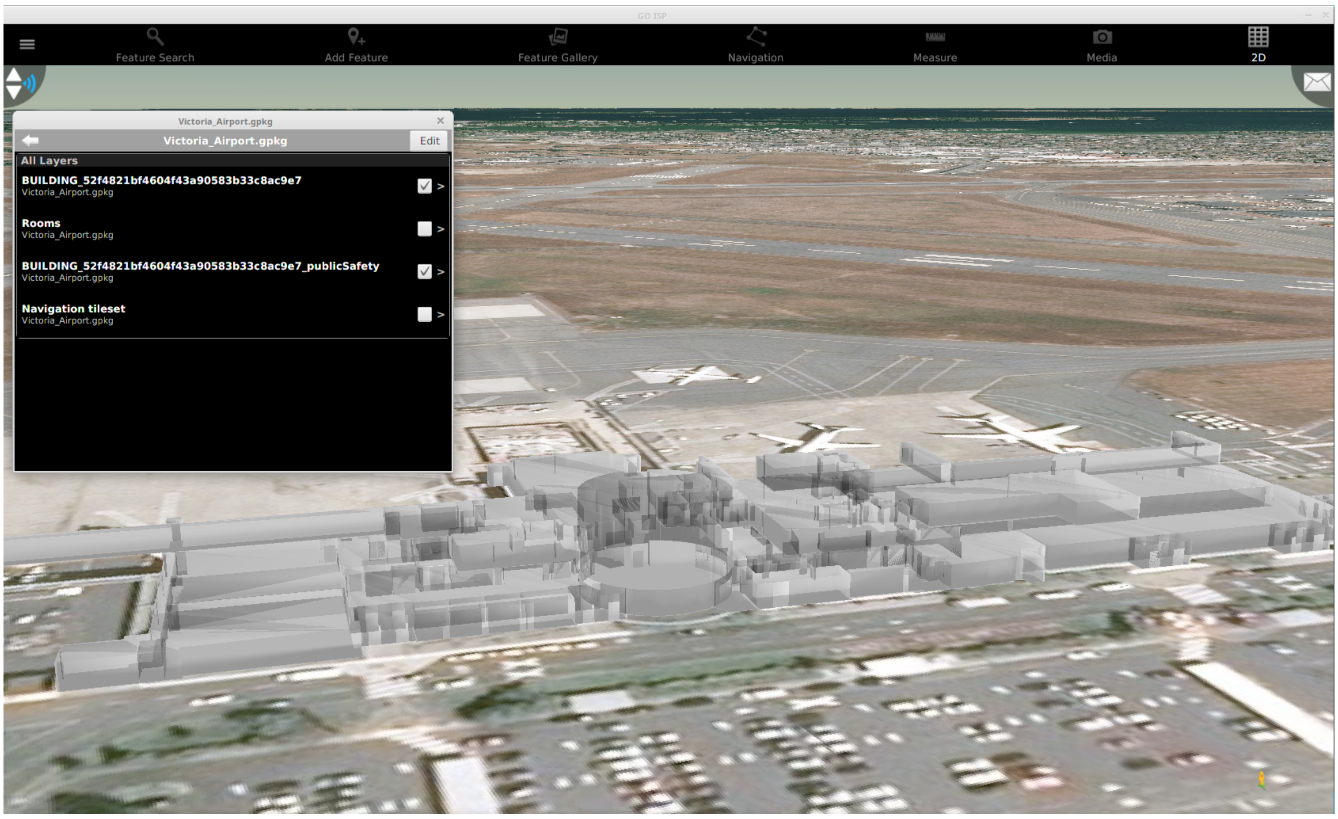
The Indoor Navigation Service was implemented as a WPS instead of the suggested WFS-T due to flexibility required for the implementation. The WPS allowed for a simpler interface to pass the start and end points as well as the IndoorGML to the navigation service and to return the route information as IndoorGML. A Web Feature Service (WFS) would not work for this as it is simply a query interface to return feature information. There was no clear way to pass the IndoorGML data using a WFS.

Chapter 10. Pre-planning Tool Client

The pre-planning tool client is the user interface that a first responder or technician would interact with during a pre-planning scenario. The client provides 3D and 2D views of the captured point cloud, converted CityGML geometries, IndoorGML indoor spaces, and IndoorGML navigational networks. Using the pre-planning client, a user can access the building repository to retrieve the stored datasets, display and manipulate the viewing angles, and generate routes by specifying start and end points. The routes are displayed within the client along with the public safety features. In the future, turn-by-turn directions could be incorporated to provide the optimal route safely throughout the building and avoid hazards.

10.1. Compusult Pre-planning Tool Client

Compusult extended their current desktop and mobile client offering called GoMobile to support CityGML, IndoorGML, and Point Cloud datasets. The GoMobile Pre-planning Tool client can display 2D maps and images, Point Clouds, 3D maps (See Figure 65), and 3D Indoor data (See Figure 66).



Victoria_Airport.gpkg

All Layers

BUILDING_52f4821bf4604f43a90583b33c8ac9e7 ✓
Victoria_Airport.gpkg

Rooms
Victoria_Airport.gpkg

BUILDING_52f4821bf4604f43a90583b33c8ac9e7_publicSafety ✓
Victoria_Airport.gpkg

Navigation tileset
Victoria_Airport.gpkg

Figure 65. GoMobile Pre-planning Tool Displaying Victoria Airport

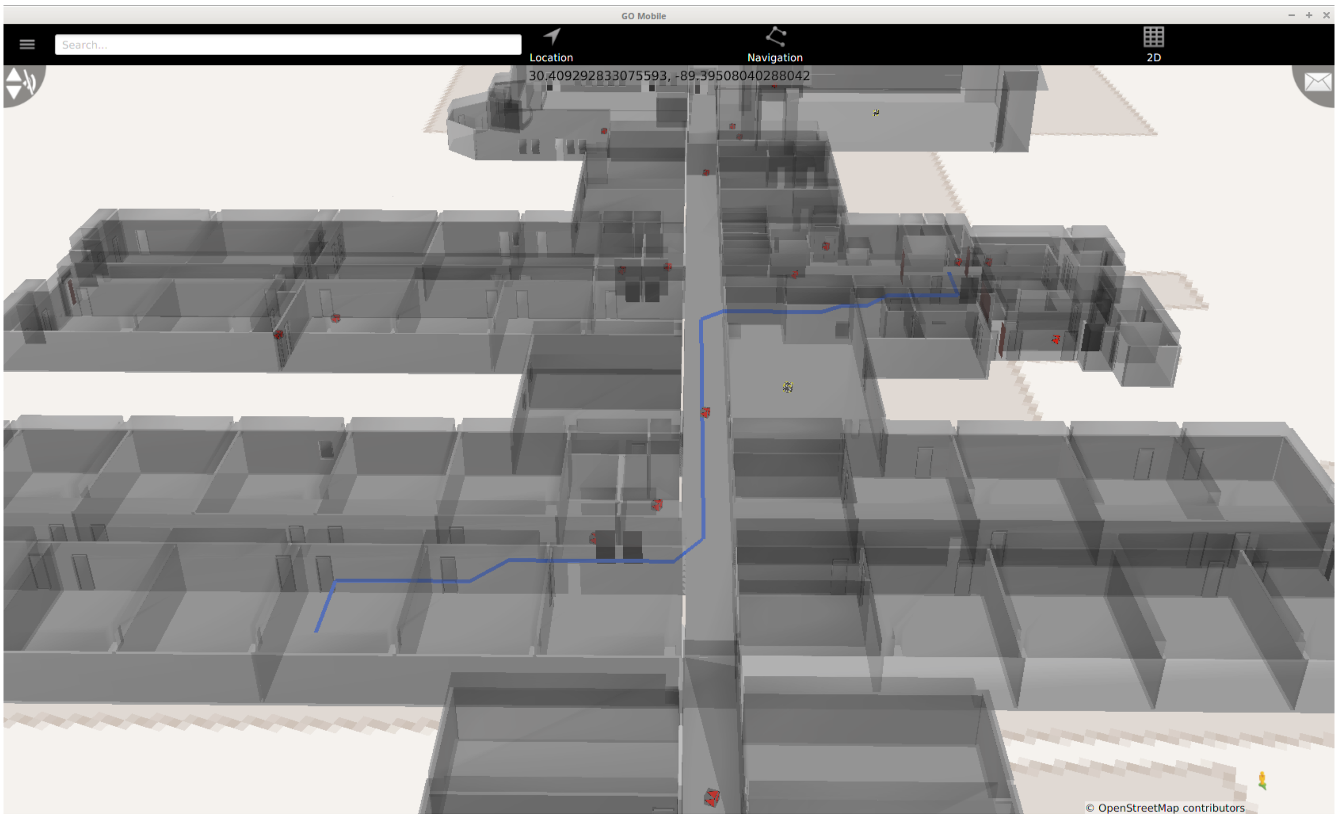


Figure 66. GoMobile Pre-planning Tool Displaying 3D Indoor Data

10.1.1. Workflow

GoMobile has the ability to query, render and navigate indoor scenes according to the following workflow:

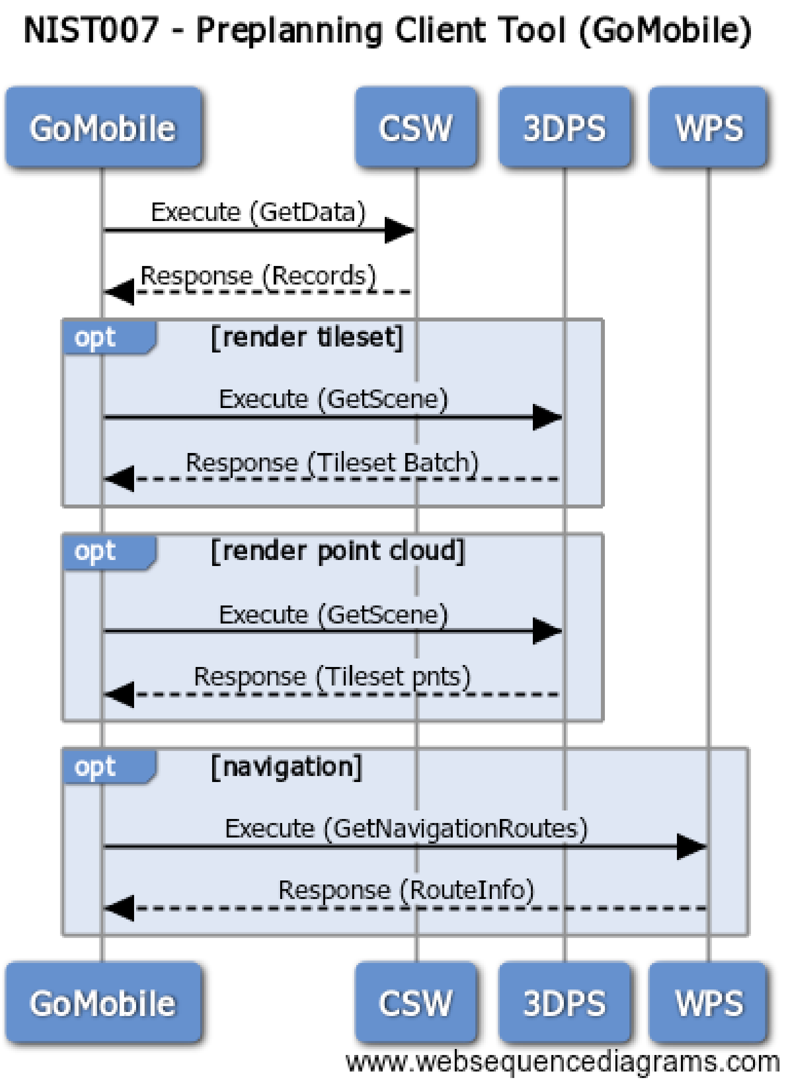


Figure 67. Pre-planning Tool Client Sequence Diagram

1. GoMobile initiates a getData request to Compusult's Web Enterprise Suite (WES) platform which returns an ISO document with URLs to the map, point cloud, navigation resources
2. GoMobile calls the 3DPS to retrieve the 3D map or the point cloud data.
3. Users can use GoMobile to initiate a WPS navigation request by specifying a start point, an end point in the WPS GetNavigationRoutes request.
4. The Route info (GML) is passed back to the user for visualization in GoMobile client application.

10.1.2. Lessons Learned

The first challenge was how to pick the start and end point in 3D space. Due to the nature of 3D rendering, a user can rotate the model so that multiple points can be oriented behind each other. This presented a challenge to determine which point the user meant to click on when they click in 3D viewer. It was assumed that the user meant to click the closest point, which is used for the route start or end point.

The specification lacked guidance on how to represent the 3D model so it was implemented with gray translucent walls, floors, and ceilings, gray non-public safety doors, and brownish red public safety doors. This type of information could have been more clearly defined in the appearance module of the CityGML. Without the appearance module definitions individual clients could render the building differently. **Additional guidance for public safety viewing.**

There appears to be inconsistent use of the boundary surface thematic classes (e.g., <bldg:RoofSurface>, <bldg:CeilingSurface>, <bldg:FloorSurface>, etc.) as opposed to LOD4MultiSurface. Therefore, the client could not reliably represent surfaces based on the element class in some models. Use of the thematic classes would allow for finer detailed representation of the building parts. For example, a client could render the floor a different color than the roof or ceiling. It would also allow for parts of the building to be turned on/off in the display, giving the user the ability to turn off the roof, for example.

Different uses of the standard CityGML namespaces (e.g., generic vs building vs bridge) presented challenges for parsing CityGML data. Different datasets used different elements / namespaces. Use of namespaces such as 'generic' instead of 'building' caused parsing errors when using the CityGML.

There is a lack of a specification on how to represent Public Safety features. It was decided to use a lookup dictionary for mapping the icon to the Public Safety feature type. This means users of the ADE have to parse the XML to get the type of the feature, then take the type and do a lookup to find the URL to the icon, then download the icon for use. This limits certain types (e.g., fire extinguisher) to a single icon. It would be simpler if the ADE XML itself contained the URL directly to the icon and by-pass the lookup. However, there is the benefit that if the URL to the icon changes it only needs to be updated in the lookup table and not in all places throughout the XML.

It was found that there were different versions of GML used between CityGML and IndoorGML. CityGML used V3.1.1 whereas IndoorGML used v3.2.1 which led to parsing issues. Initially, it was expected that the CityGML and IndoorGML needed to be consistent in versions. However, the issue was overcome by adding support for both versions of GML.

Point Cloud data was received in three formats (PCD, LAS and LAZ). In order to process the various formations, Compusult used a 3rd party tool to manually convert all Point Clouds to LAZ before being converted into 3DTiles in .PNTS format which was used by the client for display. This introduced a manual step in the process which is not desirable.

The Point Cloud datasets did not contain any information for georeferencing so a manual process was introduced to georeference, scale, and rotate the point cloud so that it displayed corrected on the maps in Compusult's 3D mapping client. The specification does allow for georeferencing information which, if provided, would allow rendering of the building in the correct location. In the absence of a geo-location to draw the points our client would default to 0,0.

10.2. EcoDomus Pre-planning Tool Client

EcoDomus is the software platform to create Digital Twins of facilities and infrastructure objects. The software tracks every object's attributes (e.g., from a rooftop chiller to a receptacle) and allows visualizing objects in 3D and 2D. For this research project, EcoDomus expanded its 3D Viewer capabilities to support CityGML and IndoorGML.

EcoDomus Viewer is able to render CityGML geometry and show properties for the selected objects. The software allows loading multiple CityGML files and aligning the objects (facilities) by moving them within the 3D scene. An example below (See Figure 68) shows a campus 3D map using CityGML (white buildings) alongside the BIM model (yellow-brown building, exported from Revit):

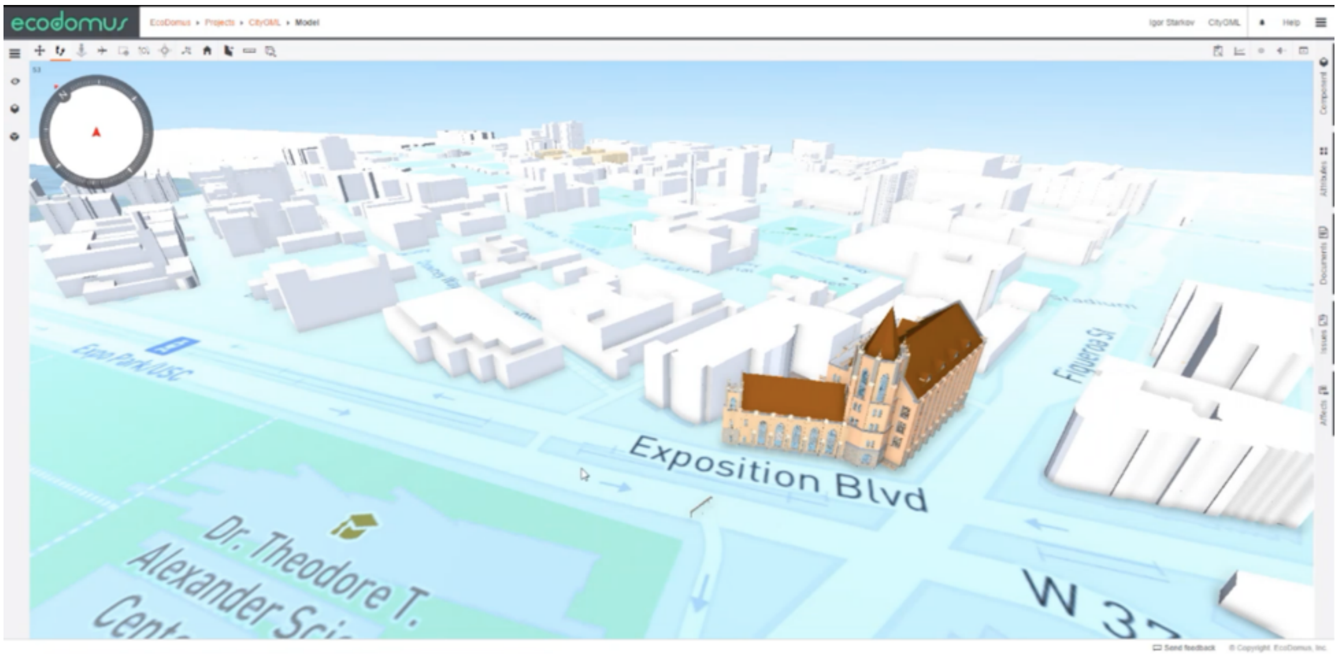


Figure 68. EcoDomus Viewer

CityGML-rendered facilities can also be merged within the 3D scene with point clouds as shown in Figure 69 below.

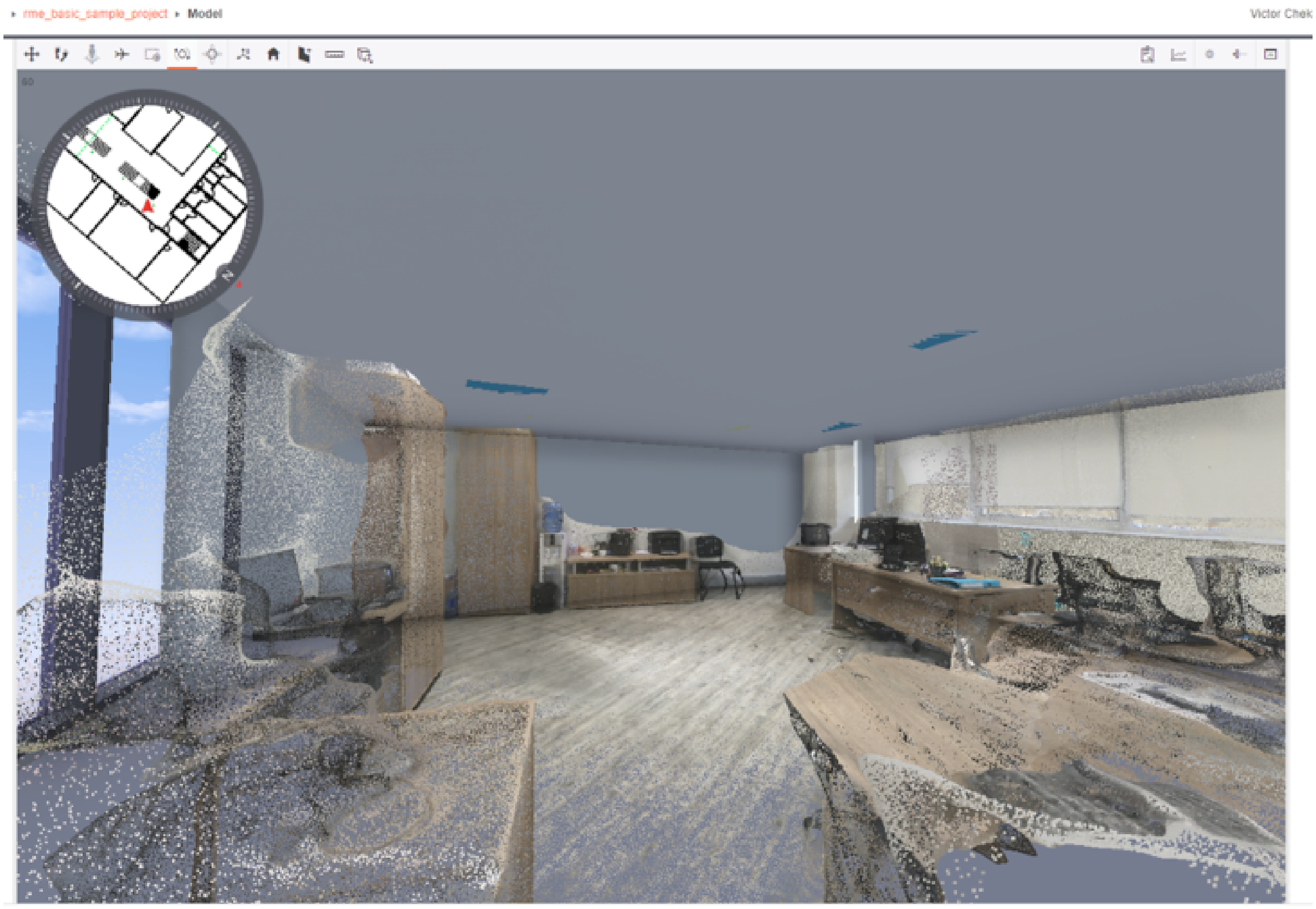


Figure 69. EcoDomus Viewer Interior View

10.2.1. Workflow

EcoDomus Viewer operates in a standard browser (e.g., Google Chrome) via WebGL and does not require special plugins. Asset data is stored in MS SQL server and can be edited by the authorized users.

1. The authorized user logs into EcoDomus software and opens the Viewer.
2. The user enters some facility's address into a textbox in the menu.
3. EcoDomus software accesses CSW hosted by another project participant and searches by a keyword agreed upon (e.g., "Hancock County").
4. EcoDomus software retrieves the CityGML file for the facility and loads it into the Viewer.
5. The user can navigate within the scene using standard controls (walk, orbit, pan, zoom, etc.), select objects and see/edit its data. The user can hide selected objects, isolate objects making the objects semi-transparent, etc.

EcoDomus Viewer operates in a standard browser (Chrome) via WebGL and does not require special plugins. Asset data is stored in MS SQL server and can be edited by the authorized users. For this project, EcoDomus added IndoorGML processor. An example of a project file is shown below:

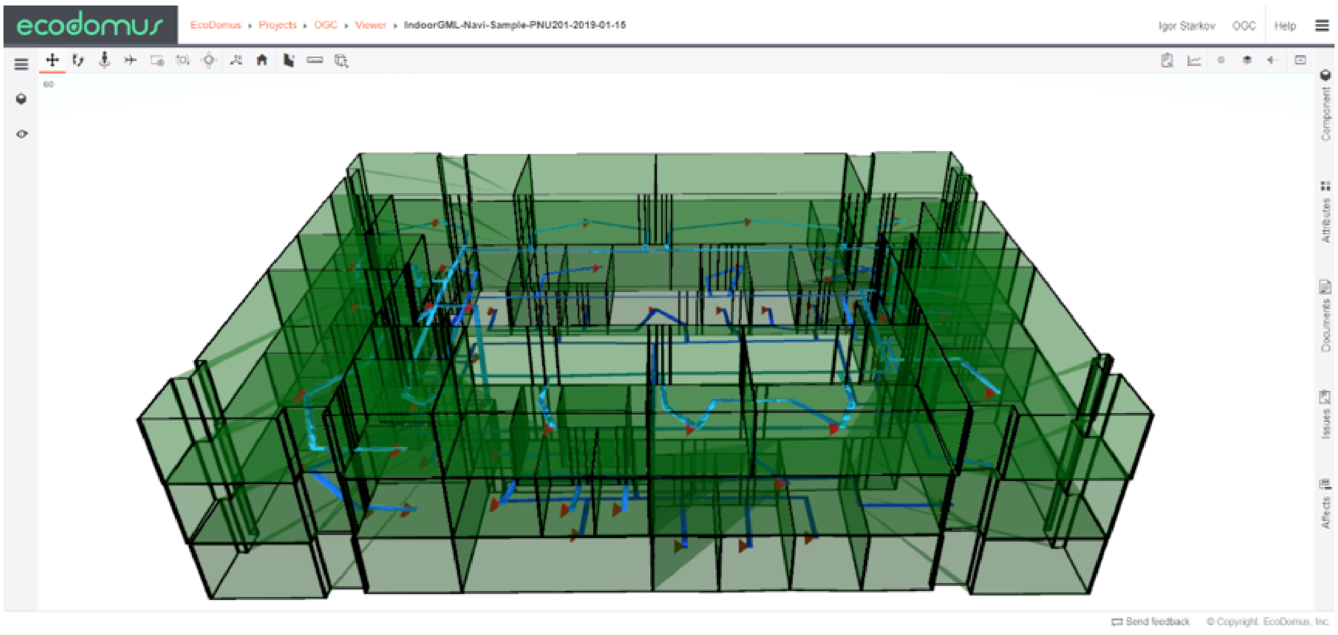

ecodomus
EcoDomus ▸ Projects ▸ OGC ▸ Viewer ▸ IndoorGML-Navi-Sample-PNU201-2019-01-15
Igor Starkov
OGC
Help
Component
Attributes
Documents
Issues
Affects
Send feedback
© Copyright. EcoDomus, Inc.

Figure 70. EcoDomus Viewer IndoorGML Processor

Red objects show the nodes and blue lines show the suggested navigation between the nodes.

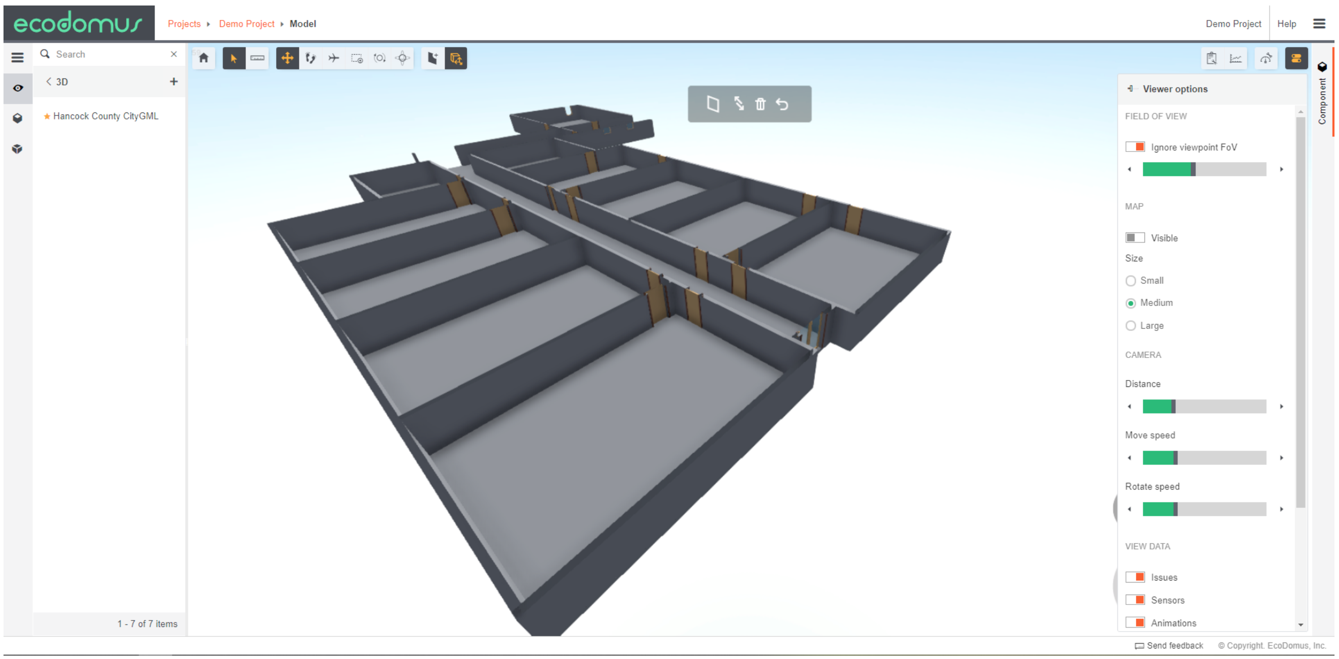


Figure 71. EcoDomus Viewer displaying Hancock CityGML

Figure 71 shows the resulting Hancock data in the Ecodomus Viewer.

10.2.2. Lessons Learned

- The different CityGML files have different coordinate reference systems. The Faramoon data does not contain a CRS, and using a local cartesian reference worked fine. The Safe software data does and opens correctly on first try. So, the CRS should be specified in the CityGML data to avoid interoperability issues.
- Need standards for units of measure such as millimeters or meters.
- In order to properly view the navigational routes, they need to be overlaid on top of the CityGML view.

Chapter 11. Public Safety Scenario (Demonstration)

The NIST Public Safety Communications Research Division is the primary federal laboratory conducting research, development, testing, and evaluation for public safety communications technologies. This OGC Innovation Pilot initiative is focused on the research and development of Indoor Mapping technologies and public safety extensions using CityGML and IndoorGML derived from point clouds and other 3-dimensional data from automated scans. The goal is to incorporate open standards to support the public safety objectives and increase adoption of these standards to improve tooling and support from the geospatial community.

The scenario for this initiative is based on pre-planning for firefighting and emergency first-responder operations. First responders typically survey high-risk facilities in their jurisdiction for pre-incident planning, but they are often forced to create their own hand-drawn maps during the process. LiDAR and image scans can be conducted periodically in large buildings to capture any changes in the building construction, remodeling, expansion, and placement of public safety features including fire extinguishers, fire exits, water pumps, etc. Using mobile mapping systems (see Figure 72) equipped with LiDAR and a 360-degree camera, technicians can efficiently capture 3D point clouds (see Figure 73), transform them into vector formats (CityGML, IndoorGML), and calculate and modify navigable indoor routes for incident response. The scenario demonstrates the use of point cloud data to support firefighting and emergency operations pre-planning.

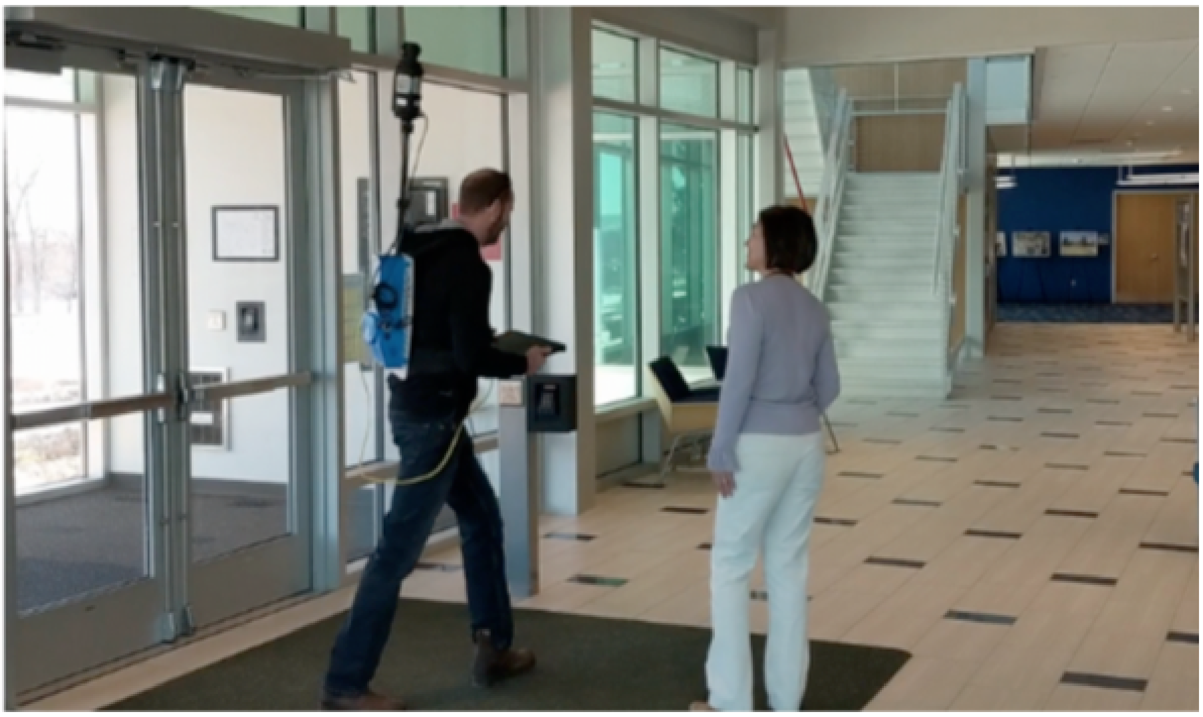


Figure 72. Mobile LiDAR Scanning Operation

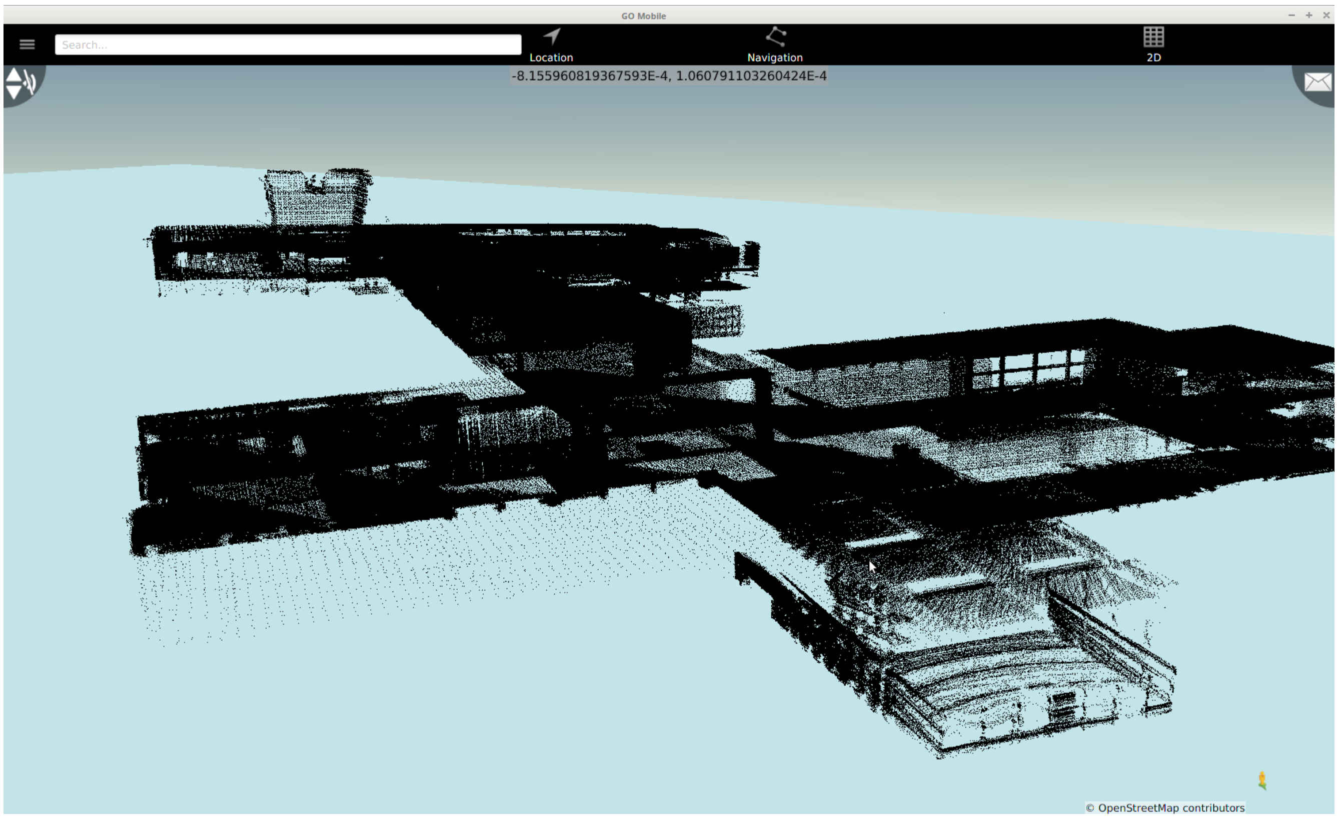

GO Mobile
Search...
Location
Navigation
2D
-8.155960819367593E-4, 1.060791103260424E-4
© OpenStreetMap contributors

Figure 73. 3D LiDAR Point Cloud

The process begins with the conversion of point cloud data into CityGML format (see Figure 74). The resulting CityGML data can be enriched with Public Safety features captured from image scans, point cloud object identification, or manual insertion. New Public Safety features are taken from the NAPSG Public Safety library of features. (https://www.napsgfoundation.org). The Public Safety ADE provides the CityGML schema extension for these features which are then converted to Public Safety Extension in IndoorGML to provide navigational support for indoor routing.

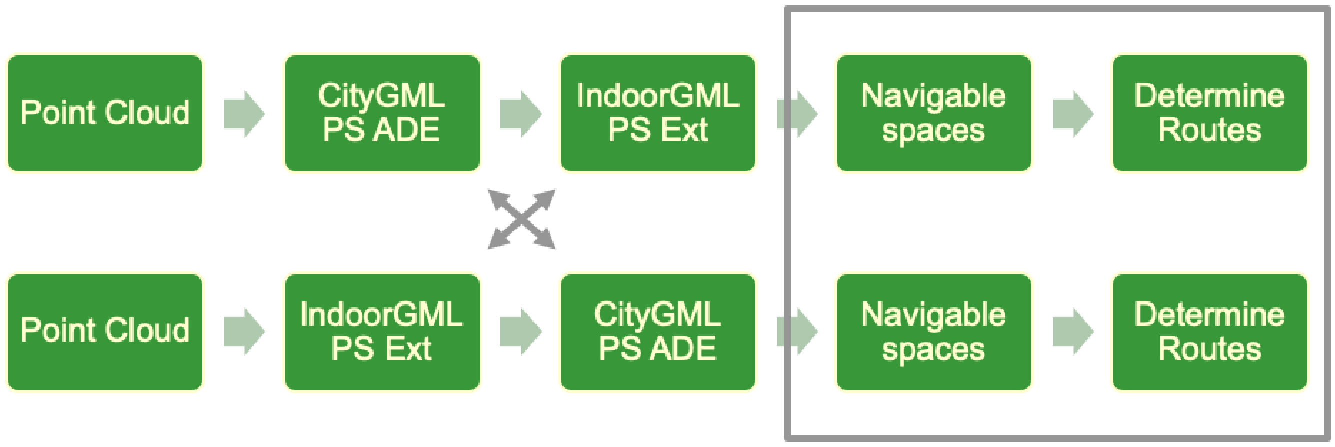
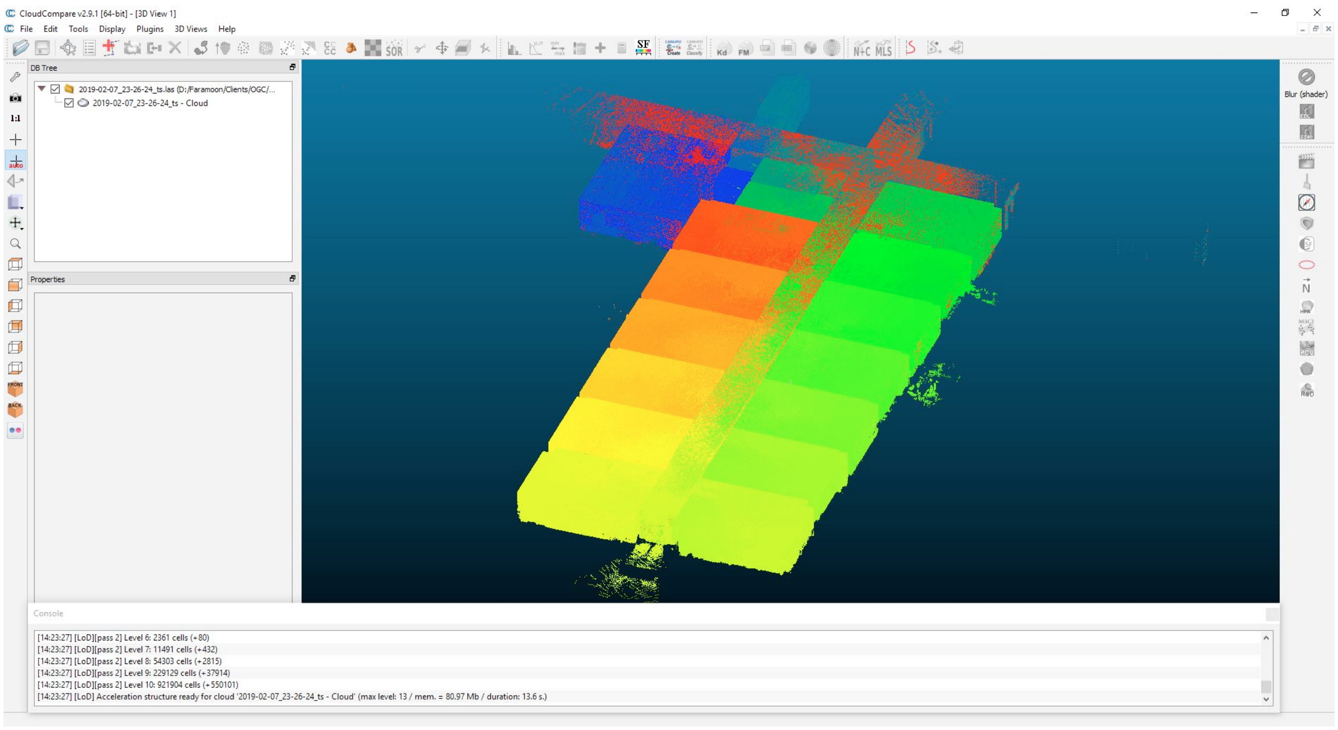


Figure 74. OGC Indoor Mapping & Navigation Pilot Overview

Once the conversion process is complete, a planner can support training operations for emergency response using a pre-planning client to view the information in a graphical user interface. Having knowledge of public safety features and the best route through an interior space during an emergency would help in firefighting situations and support emergency operations.

11.1. Point Cloud to CityGML Data Conversion

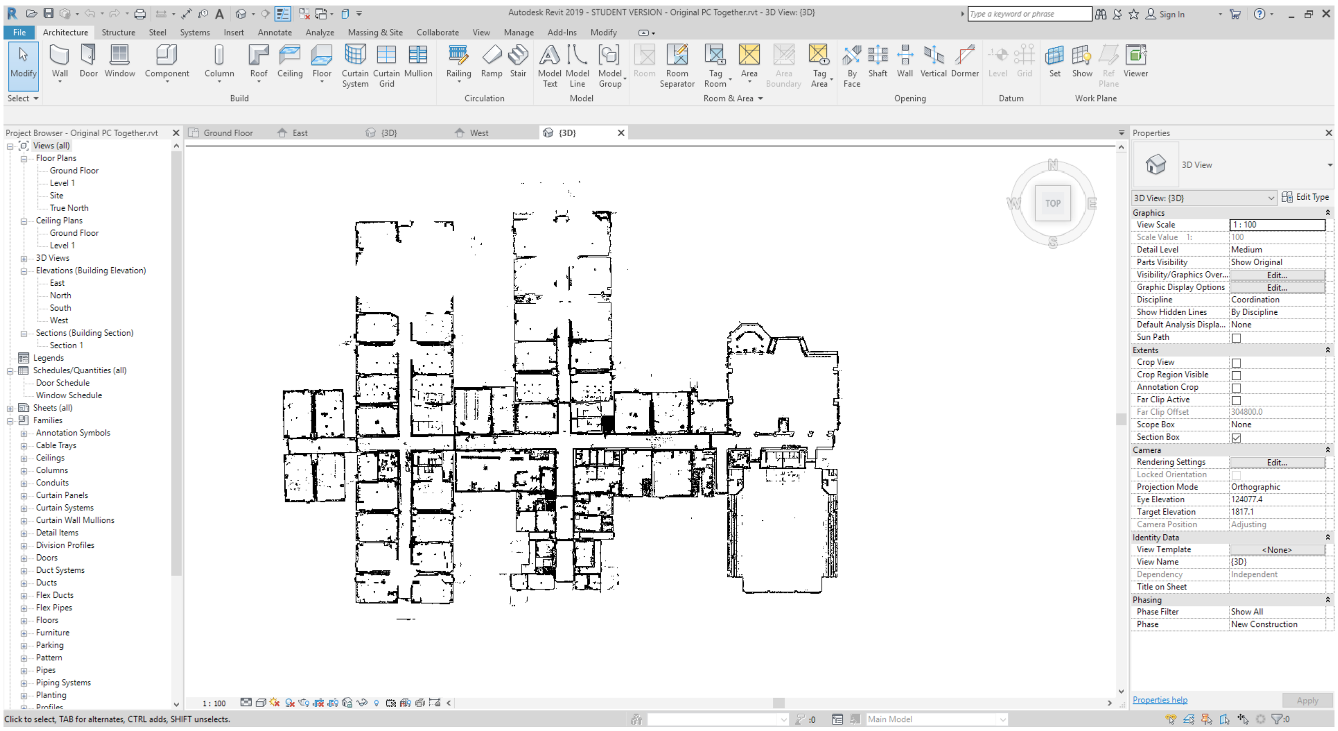
Initially, the initiative began testing with readily available point cloud data scans and textured images of Korea University and CityGML data of Victoria International Airport. Later in the initiative, official Hancock County data was provided which contained captured point cloud data and image scans. In the beginning, the team attempted to automate the conversion of the point cloud data. However, point cloud data tends to be very noisy, and the common result was a data output with false walls, missing doors, and incorrect geometries, etc. Notice the noise at the bottom and right sides of Figure 75 likely caused by LiDAR scans leaking through transparent door or window glass (see Figure 76).


CloudCompare v2.9.1 [64-bit] - [3D View 1]
File Edit Tools Display Plugins 3D Views Help
DB Tree
2019-02-07_23-26-24_ts.las (D:/Faramoon/Clients/OGC/...
2019-02-07_23-26-24_ts - Cloud
Properties
Blur (shader)
Console

```
[14:23:27] [LoD][pass 2] Level 6: 2361 cells (+80)
[14:23:27] [LoD][pass 2] Level 7: 11491 cells (+432)
[14:23:27] [LoD][pass 2] Level 8: 54303 cells (+2815)
[14:23:27] [LoD][pass 2] Level 9: 229129 cells (+37914)
[14:23:27] [LoD][pass 2] Level 10: 921904 cells (+550101)
[14:23:27] [LoD] Acceleration structure ready for cloud '2019-02-07_23-26-24_ts - Cloud' (max level: 13 / mem. = 80.97 Mb / duration: 13.6 s.)
```

Figure 75. Hancock County Point Cloud Data Noise

The Hancock County point clouds came as separate files and had to be merged.



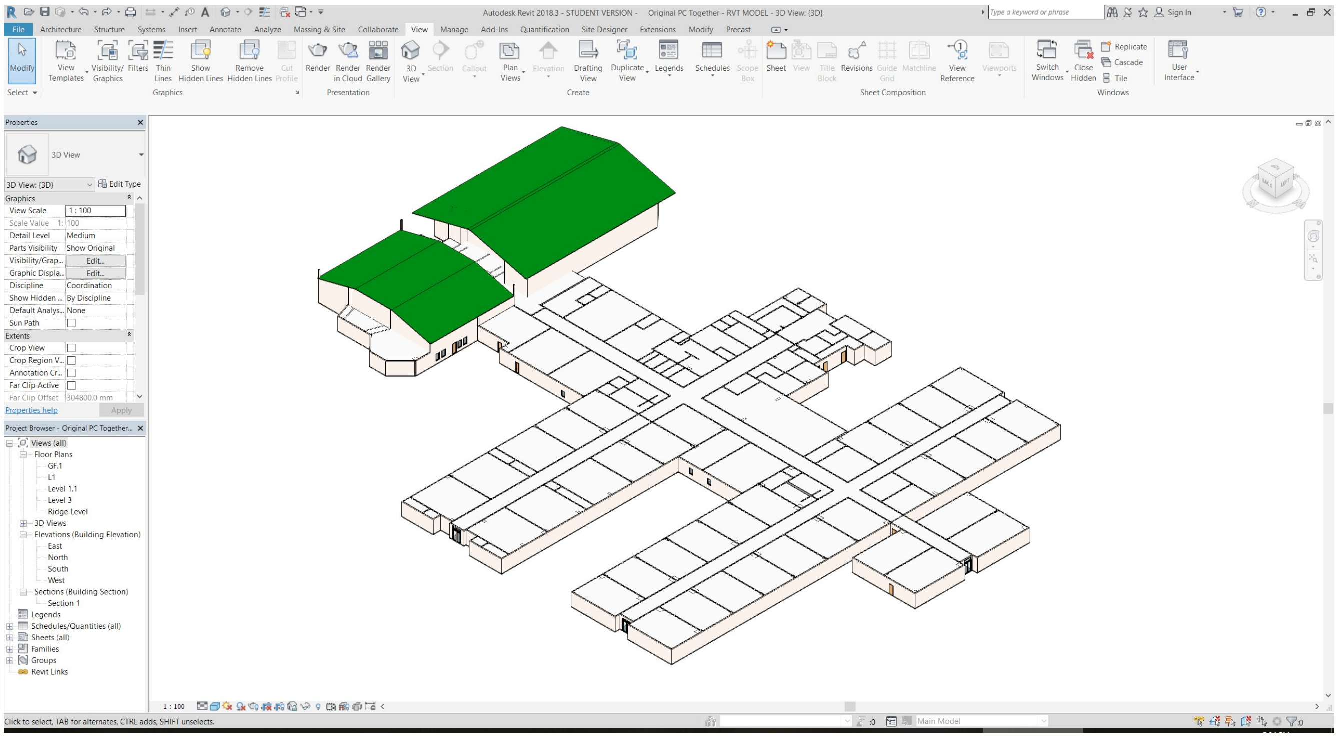
Autodesk Revit 2019 - STUDENT VERSION - Original PC Together.rvt - 3D View: {3D}

Properties — 3D View — 3D View: {3D}

Graphics	
View Scale	1 : 100
Scale Value 1:	100
Detail Level	Medium
Parts Visibility	Show Original
Visibility/Graphics Over...	Edit...
Graphic Display Options	Edit...
Discipline	Coordination
Show Hidden Lines	By Discipline
Default Analysis Displa...	None
Sun Path	☐
Extents	
Crop View	☐
Crop Region Visible	☐
Annotation Crop	☐
Far Clip Active	☐
Far Clip Offset	304800.0
Scope Box	None
Section Box	☑
Camera	
Rendering Settings	Edit...
Locked Orientation	☐
Projection Mode	Orthographic
Eye Elevation	124077.4
Target Elevation	1817.1
Camera Position	Adjusting
Identity Data	
View Template	<None>
View Name	{3D}
Dependency	Independent
Title on Sheet	
Phasing	
Phase Filter	Show All
Phase	New Construction

Figure 76. Hancock County Point Cloud Data Merge in Revit

The consensus is that current software tools and libraries are not yet mature enough to produce clean geometries from point cloud data, and the data needed to be cleansed manually and enriched in order to be used for the demonstration. Attempts to use open source tools proved to be difficult due to immature developmental status and lack of existing documentation for the software. In general, the team was able to use automation to output either CityGML or IFC data (BIM standard). Since editing tools are limited for CityGML formats, the team had to manually cleanse the data and output the data in IFC format (See Figure 77 and Figure 78). Once the data was output in IFC format and georeferenced, it is considered clean enough to use for demonstration purposes.

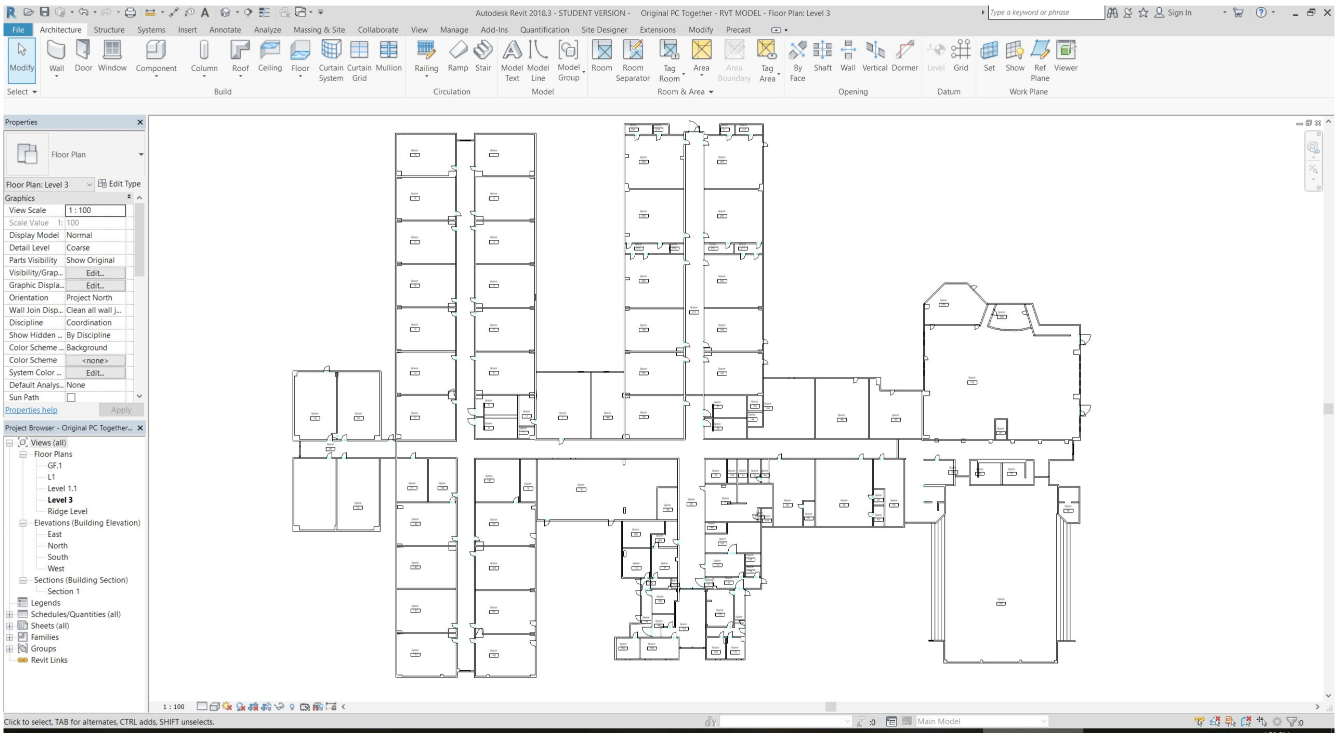


Autodesk Revit 2018.3 - STUDENT VERSION - Original PC Together - RVT MODEL - 3D View: {3D}

Properties — 3D View — 3D View: {3D}

Graphics	
View Scale	1 : 100
Scale Value 1:	100
Detail Level	Medium
Parts Visibility	Show Original
Visibility/Grap...	Edit...
Graphic Displa...	Edit...
Discipline	Coordination
Show Hidden ...	By Discipline
Default Analys...	None
Sun Path	☐
Extents	
Crop View	☐
Crop Region V...	☐
Annotation Cr...	☐
Far Clip Active	☐
Far Clip Offset	304800.0 mm

Figure 77. Hancock County IFC Data Output


Autodesk Revit 2018.3 - STUDENT VERSION - Original PC Together - RVT MODEL - Floor Plan: Level 3

Floor Plan: Level 3

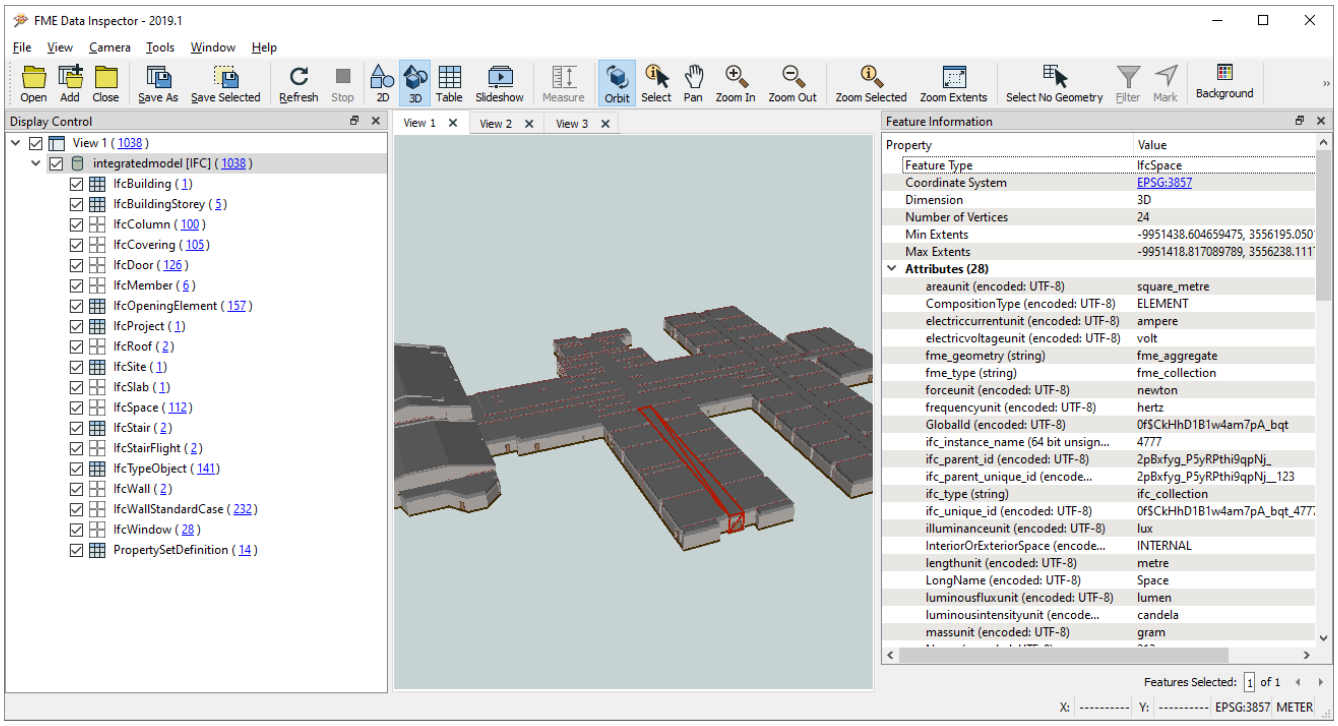
Graphics	
View Scale	1 : 100
Scale Value 1:	100
Display Model	Normal
Detail Level	Coarse
Parts Visibility	Show Original
Visibility/Grap...	Edit...
Graphic Displa...	Edit...
Orientation	Project North
Wall Join Disp...	Clean all wall j...
Discipline	Coordination
Show Hidden ...	By Discipline
Color Scheme ...	Background
Color Scheme	<none>
System Color ...	Edit...
Default Analys...	None
Sun Path	☐

Project Browser - Original PC Together...
- Views (all)
  - Floor Plans
    - GF.1
    - L1
    - Level 1.1
    - Level 3
    - Ridge Level
  - Elevations (Building Elevation)
    - East
    - North
    - South
    - West
  - Sections (Building Section)
    - Section 1
- Legends
- Schedules/Quantities (all)
- Sheets (all)
- Families
- Groups
- Revit Links

Click to select, TAB for alternates, CTRL adds, SHIFT unselects.

Figure 78. Hancock County IFC Floor Plan

The edited IFC data was then post-processed using FME for geometric processing to convert IFC into CityGML. This process required experimentation to create the proper processing to convert into CityGML. Once a satisfactory method was achieved, this processing step was mostly automated. This process is shown in Figure 79 and Figure 80.


FME Data Inspector - 2019.1
File View Camera Tools Window Help
Open Add Close Save As Save Selected Refresh Stop 2D 3D Table Slideshow Measure Orbit Select Pan Zoom In Zoom Out Zoom Selected Zoom Extents Select No Geometry Filter Mark Background
View 1 View 2 View 3

Display Control
- View 1 (1038)
  - integratedmodel [IFC] (1038)
    - IfcBuilding (1)
    - IfcBuildingStorey (5)
    - IfcColumn (100)
    - IfcCovering (105)
    - IfcDoor (126)
    - IfcMember (6)
    - IfcOpeningElement (157)
    - IfcProject (1)
    - IfcRoof (2)
    - IfcSite (1)
    - IfcSlab (1)
    - IfcSpace (112)
    - IfcStair (2)
    - IfcStairFlight (2)
    - IfcTypeObject (141)
    - IfcWall (2)
    - IfcWallStandardCase (232)
    - IfcWindow (28)
    - PropertySetDefinition (14)

Feature Information

Property	Value
Feature Type	IfcSpace
Coordinate System	EPSG:3857
Dimension	3D
Number of Vertices	24
Min Extents	-9951438.604659475, 3556195.050
Max Extents	-9951418.817089789, 3556238.111
Attributes (28)	
areaunit (encoded: UTF-8)	square_metre
CompositionType (encoded: UTF-8)	ELEMENT
electriccurrentunit (encoded: UTF-8)	ampere
electricvoltageunit (encoded: UTF-8)	volt
fme_geometry (string)	fme_aggregate
fme_type (string)	fme_collection
forceunit (encoded: UTF-8)	newton
frequencyunit (encoded: UTF-8)	hertz
GlobalId (encoded: UTF-8)	0f$CkHhD1B1w4am7pA_bqt
ifc_instance_name (64 bit unsign...	4777
ifc_parent_id (encoded: UTF-8)	2pBxfyg_P5yRPthi9qpNj_
ifc_parent_unique_id (encode...	2pBxfyg_P5yRPthi9qpNj__123
ifc_type (string)	ifc_collection
ifc_unique_id (encoded: UTF-8)	0f$CkHhD1B1w4am7pA_bqt_477
illuminanceunit (encoded: UTF-8)	lux
InteriorOrExteriorSpace (encode...	INTERNAL
lengthunit (encoded: UTF-8)	metre
LongName (encoded: UTF-8)	Space
luminousfluxunit (encoded: UTF-8)	lumen
luminousintensityunit (encode...	candela
massunit (encoded: UTF-8)	gram

Features Selected: 1 of 1

X: ---------- Y: ---------- EPSG:3857 METER

Figure 79. IFC source data from Building Modeler for Hancock County

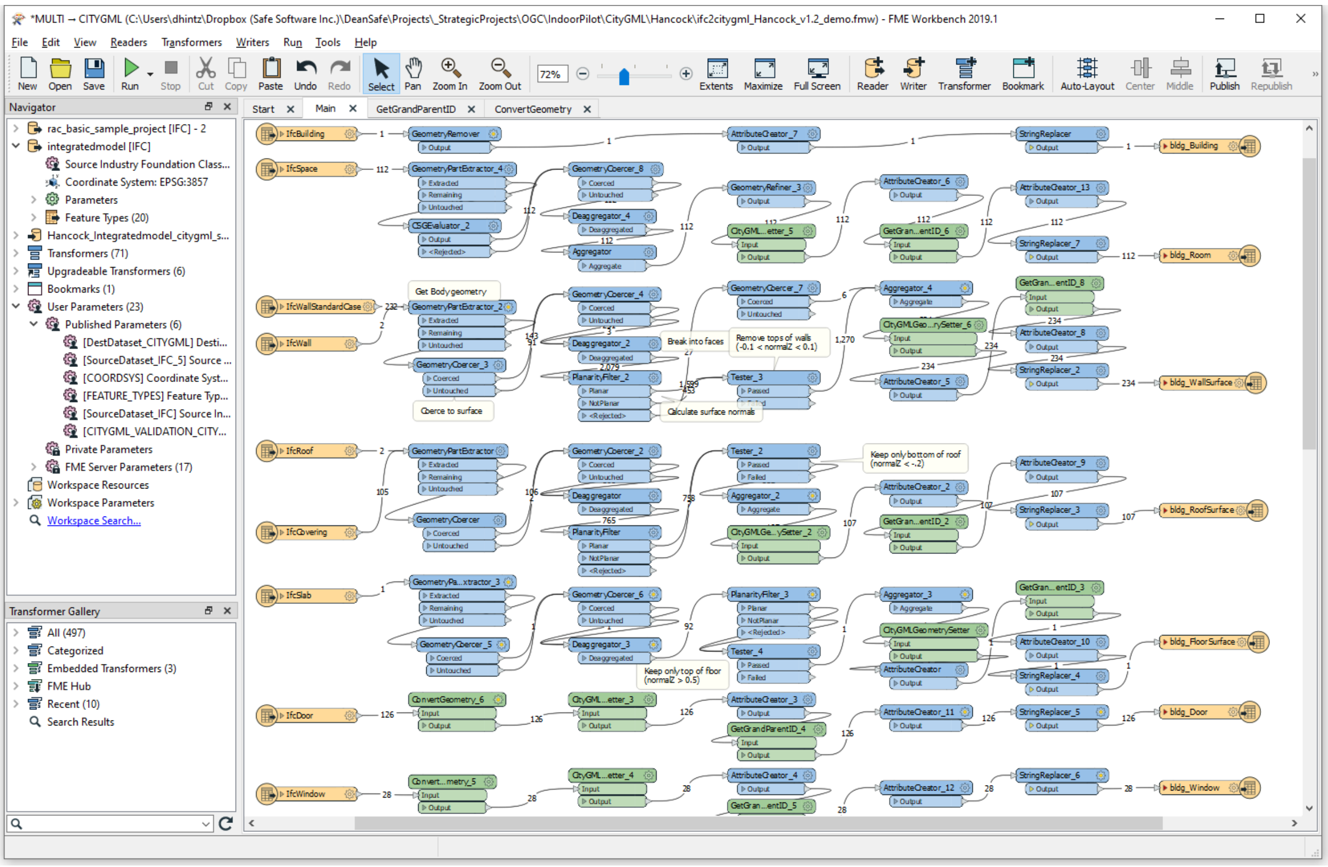
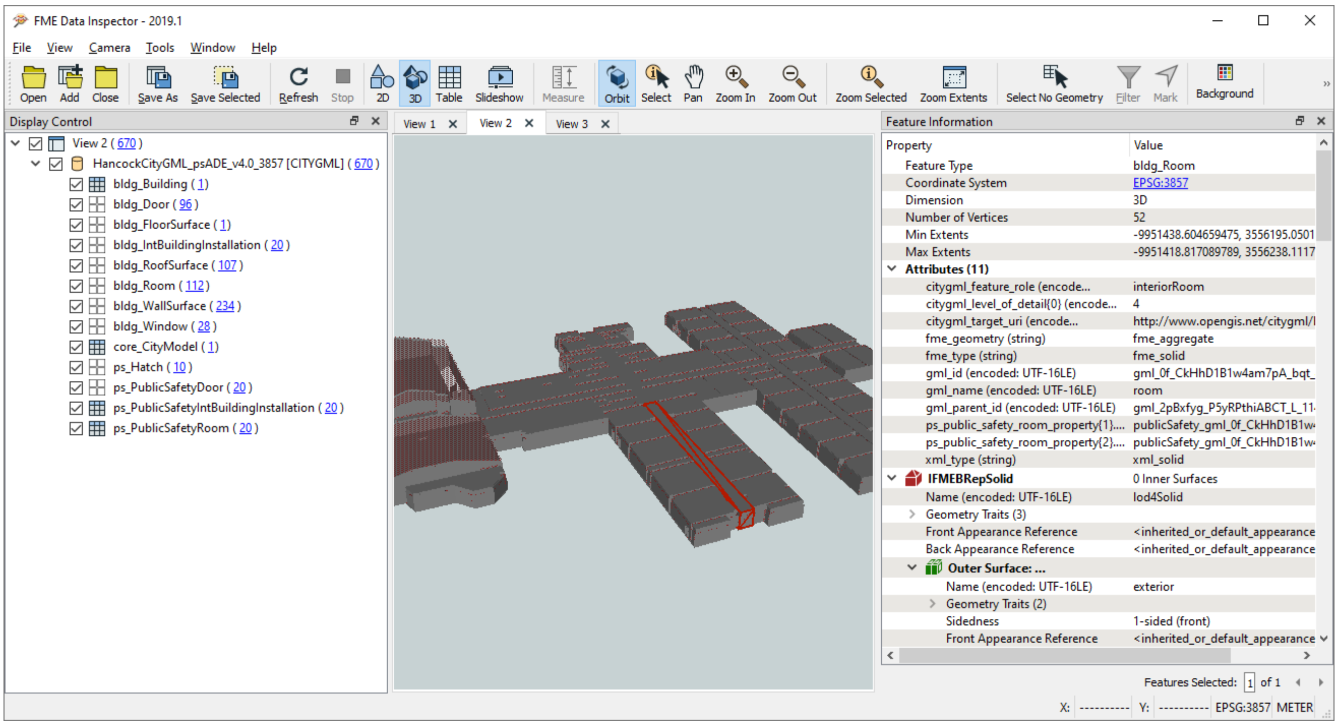

*MULTI → CITYGML (C:\Users\dhintz\Dropbox (Safe Software Inc.)\DeanSafe\Projects_StrategicProjects\OGC\IndoorPilot\CityGML\Hancock\ifc2citygml_Hancock_v1.2_demo.fmw) - FME Workbench 2019.1
File Edit View Readers Transformers Writers Run Tools Help
Navigator
Transformer Gallery
Start Main GetGrandParentID ConvertGeometry
IfcBuilding
IfcSpace
IfcWallStandardCase
IfcWall
IfcRoof
IfcCovering
IfcSlab
IfcDoor
IfcWindow
Get Body geometry
Coerce to surface
Break into faces
Remove tops of walls (-0.1 < normalZ < 0.1)
Calculate surface normals
Keep only bottom of roof (normalZ < -.2)
Keep only top of floor (normalZ > 0.5)
bldg_Building
bldg_Room
bldg_WallSurface
bldg_RoofSurface
bldg_FloorSurface
bldg_Door
bldg_Window

Figure 80. Spatial ETL (FME) process to convert Hancock County data from IFC to CityGML

In IFC, indoor geometries are typically represented as solids. However, for CityGML, geometries typically are represented as surfaces, and therefore need to be converted from solids into multi-surface elements to properly represent the room boundaries. CityGML contains "walls" and "rooms" whereas IndoorGML contains "Cell" or "GeneralSpace" which are mapped from CityGML "rooms". In order to support this conversion to CityGML, a new feature called <ifcSpace> was created which were then converted into CityGML "rooms". During this step, often too many surfaces are generated. Extra surfaces were filtered and discarded in FME with a process using surface normal generation and analysis. For more information see Section 5.3.1, “Conversion to IFC”.



View 2 (670)
HancockCityGML_psADE_v4.0_3857 [CITYGML] (670)
bldg_Building (1)
bldg_Door (96)
bldg_FloorSurface (1)
bldg_IntBuildingInstallation (20)
bldg_RoofSurface (107)
bldg_Room (112)
bldg_WallSurface (234)
bldg_Window (28)
core_CityModel (1)
ps_Hatch (10)
ps_PublicSafetyDoor (20)
ps_PublicSafetyIntBuildingInstallation (20)
ps_PublicSafetyRoom (20)

Property	Value
Feature Type	bldg_Room
Coordinate System	EPSG:3857
Dimension	3D
Number of Vertices	52
Min Extents	-9951438.604659475, 3556195.0501
Max Extents	-9951418.817089789, 3556238.1117
Attributes (11)	
citygml_feature_role (encode...	interiorRoom
citygml_level_of_detail{0} (encode...	4
citygml_target_uri (encode...	http://www.opengis.net/citygml/
fme_geometry (string)	fme_aggregate
fme_type (string)	fme_solid
gml_id (encoded: UTF-16LE)	gml_0f_CkHhD1B1w4am7pA_bqt_
gml_name (encoded: UTF-16LE)	room
gml_parent_id (encoded: UTF-16LE)	gml_2pBxfyg_P5yRPthiABCT_L_11
ps_public_safety_room_property{1}....	publicSafety_gml_0f_CkHhD1B1w
ps_public_safety_room_property{2}....	publicSafety_gml_0f_CkHhD1B1w
xml_type (string)	xml_solid
IFMEBRepSolid	0 Inner Surfaces
Name (encoded: UTF-16LE)	lod4Solid
Geometry Traits (3)	
Front Appearance Reference	<inherited_or_default_appearance
Back Appearance Reference	<inherited_or_default_appearance
Outer Surface: ...	
Name (encoded: UTF-16LE)	exterior
Geometry Traits (2)	
Sidedness	1-sided (front)
Front Appearance Reference	<inherited_or_default_appearance

Features Selected: 1 of 1

X: ---------- Y: ---------- EPSG:3857 METER

Figure 81. CityGML data converted from IFC data with FME

In another method, the team used a MatLab application called PolyFit to manually cleanse the point cloud data and redraw 3D surfaces. The output of the PolyFit is a clean point cloud data file. Once the data is cleaned and surfaces are reconstructed using PolyFit, it is processed using PointNet functions to manually convert the data into mesh data file set.

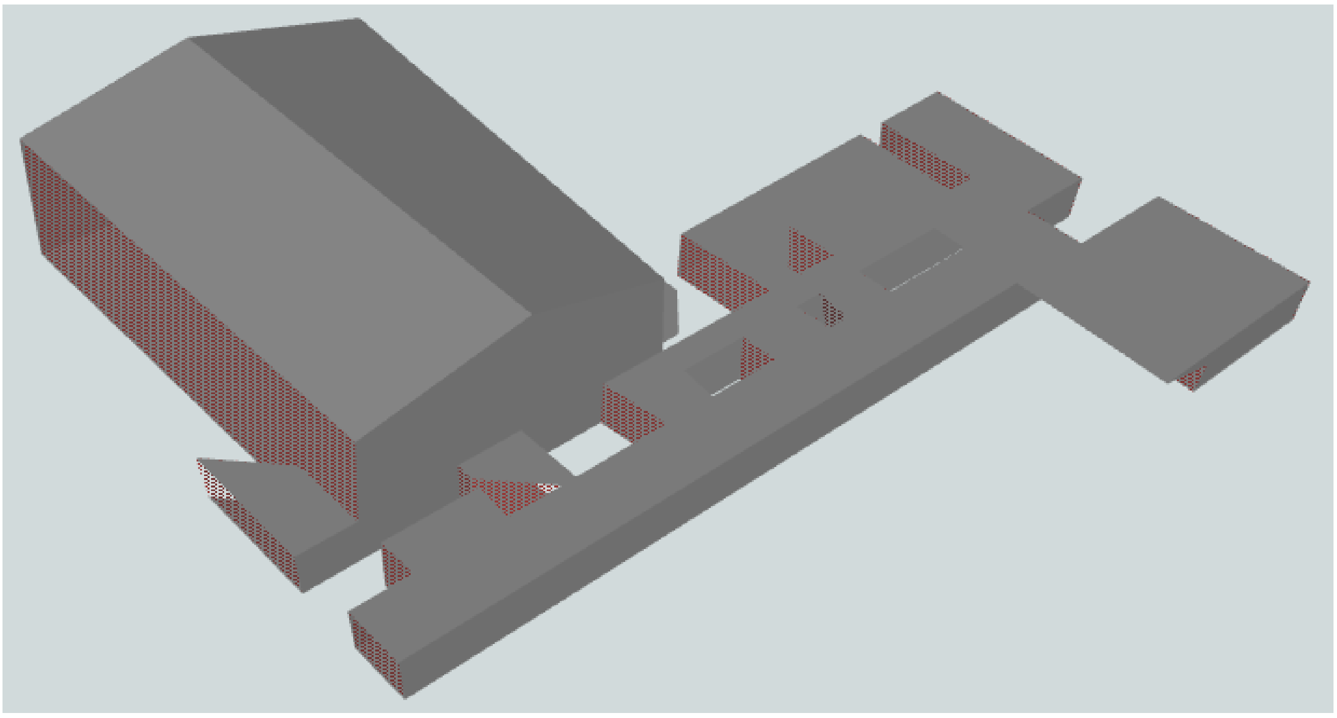


Figure 82. Hancock County Data Cleansed and ReDrawn with PolyFit.

11.2. CityGML to Indoor GML Data Conversion

The team implemented a couple different methods to generate routing networks. One method was to calculate the centroid of the segmented sub-spacing, the centroid of segmented subspaces within the corridors, the door centroids, and generate network nodes and edges between these center points. This is shown in Figure 83. This methodology is also described in more detail in Section 7.1.3, “IndoorGML Network Sub-Spacing”.

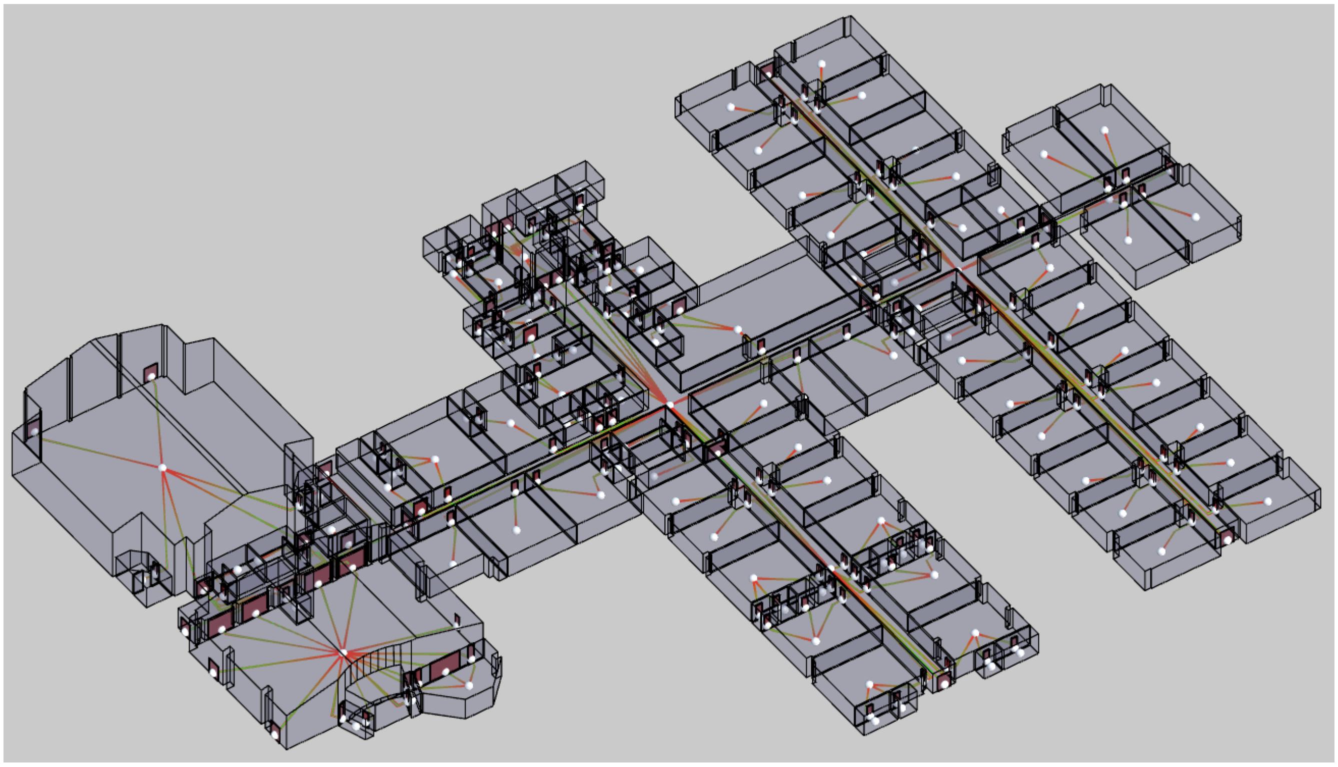


Figure 83. Simple Center Point Network of Points for Indoor Routing

The other more fine-grained method is to generate a dense network of points within each indoor space and calculate the most efficient method of routing within each space. This can be seen in Figure 84. This is done by overlaying a mesh over the space, and using walls and doors (non-navigable objects) to cut the mesh, resulting in a navigable node and edge network. For this reason, correct doors and walls are required to create an optimal network mesh and routes.

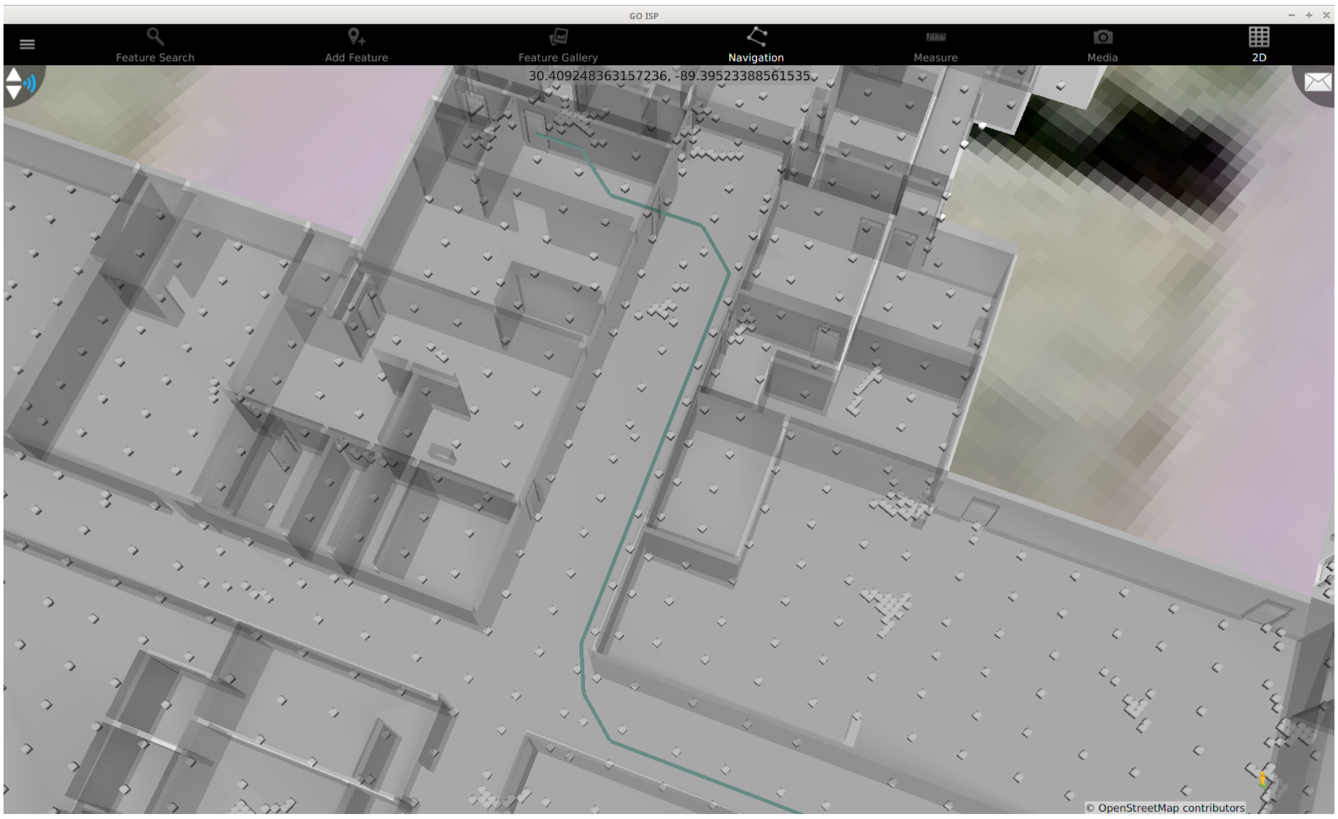


Figure 84. Fine-Grained Network of Points for Indoor Routing

11.3. Visualizing Indoor Mapping for Pre-Planning Events

By defining a start and end point, an optimal route can be generated based on various algorithms (e.g., A-Star algorithm). Once the Navigation Modeler has completed the navigational route calculations, the route can be displayed in the pre-planning client in 3D to show the navigable path for a first-responder.

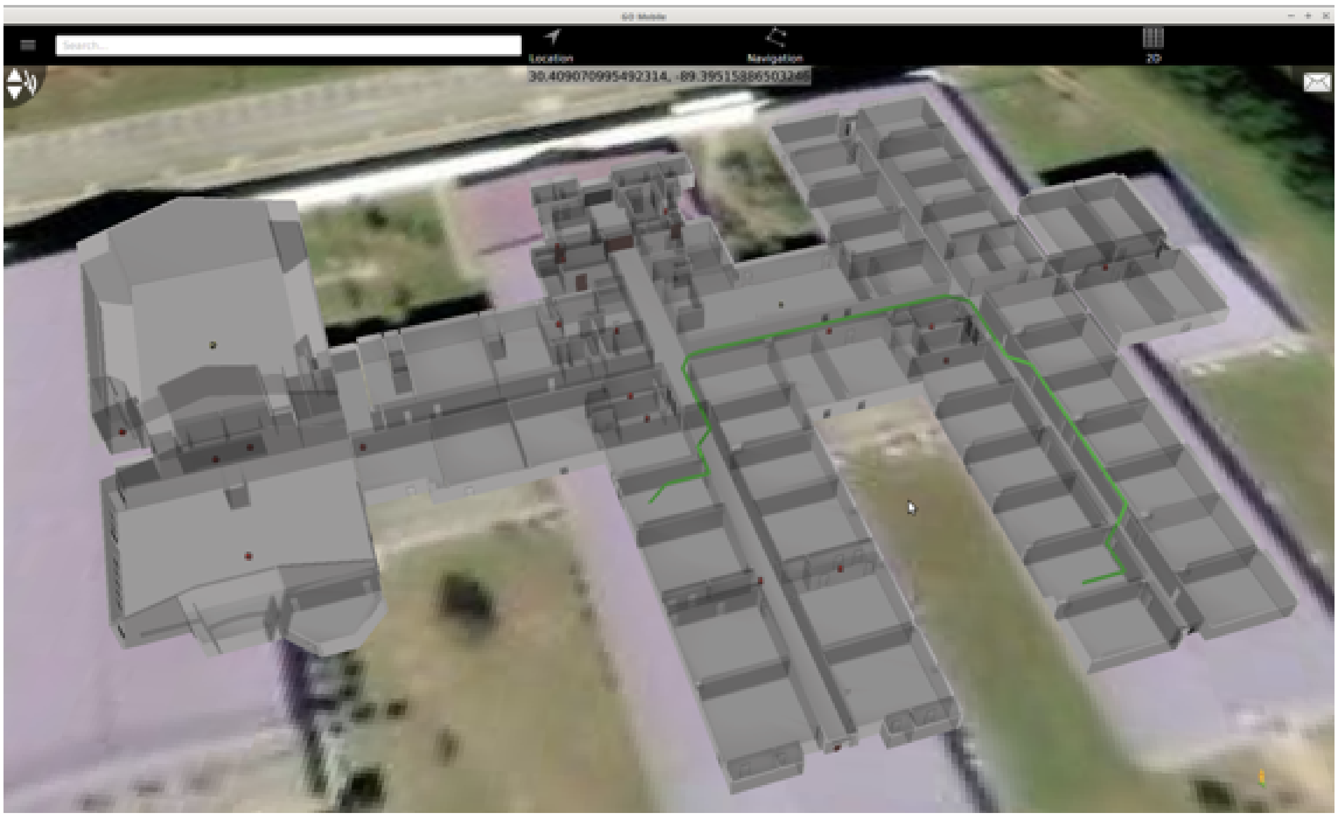


Figure 85. Navigating the IndoorGML Route

Additionally, public safety features can be visualized as part of the pre-planning walkthrough as seen in the example in Figure 86. These features include fire pumps, public safety doors and windows, and gas shutoff valves, and can be placed throughout the building as required.

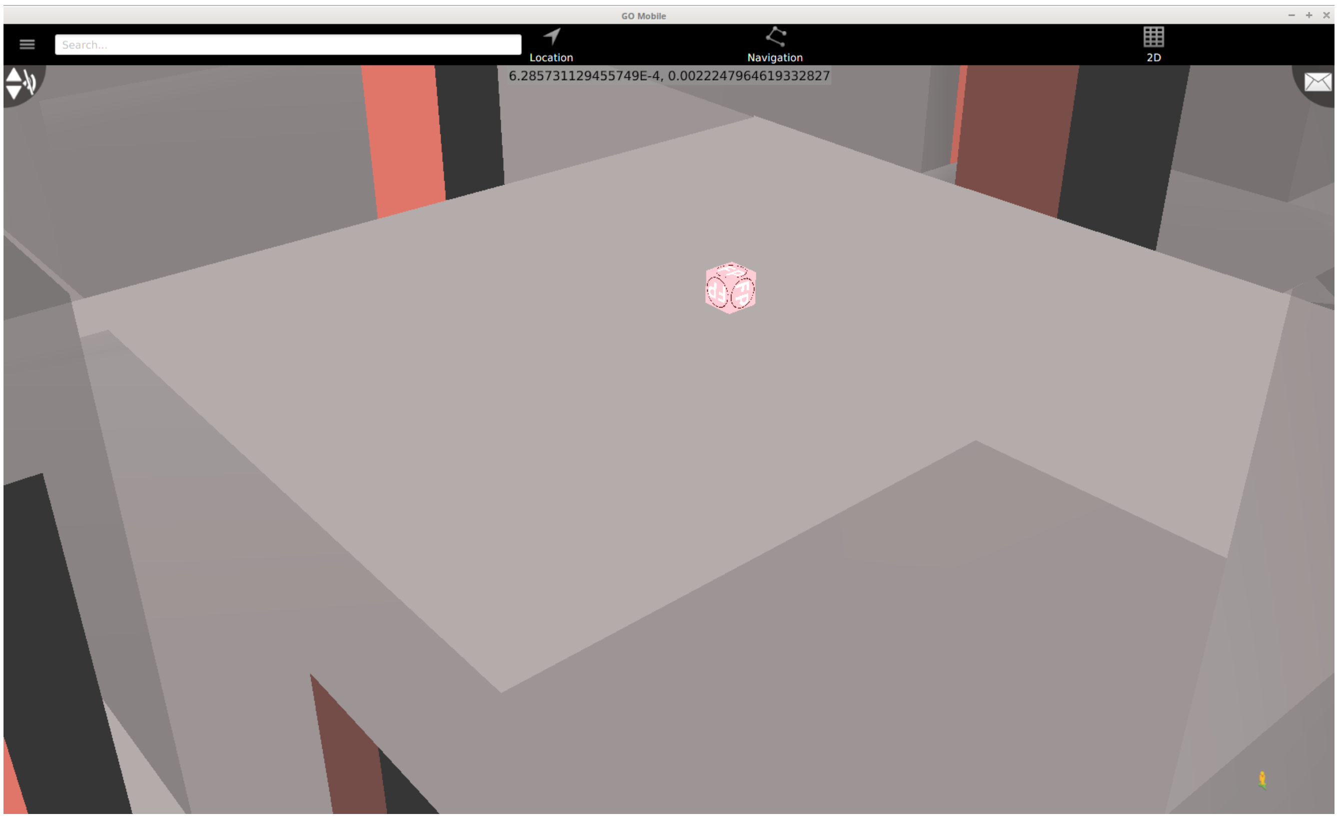


Figure 86. Visualizing Public Safety Feature Enriched in IndoorGML

11.4. Conclusions

From the perspective of the approach of this pilot, creating maps of indoor spaces is very demanding, and one of the promising approaches is to use point clouds to create indoor maps.

The team demonstrated that it is possible to transform a point cloud to CityGML Public Safety ADE and IndoorGML with a Public Safety Extension, and then visualize the public safety data in the Pre-planning Tool using open standards and basic navigation capabilities. Open standards made the process possible for data processing, modeling, exchange, and display. While some steps were automated, some manual effort is still required to get from point cloud to building features. The team had some success with semi-automated tools, and observed there is potential for automation to be improved. IndoorGML is primarily focused on indoor navigation, but this cannot be completely separated from the physical view of CityGML for the representation of the building.

Some integration of the two models would improve the use and capabilities of each. More work needs to be done to improve the capability of point cloud feature extraction, editors, converters, and develop quality, symbology and validation rules for indoor mapping and navigation data. Potential gaps between indoor and outdoor maps and simplified data stream profiles for mobile use should also be explored. Finally, enhancing client tools by integrating indoor maps with real-time environmental information and augmented reality would significantly enrich the information provided to first responders. Combined, these have the potential to significantly improve safety and enhance effectiveness when responding to a wide array of potential emergencies.

11.4.1. Summary of Lessons Learned

- Calculating the centroid of a space for IndoorGML causes anomalies if the spaces are not properly segmented. In the example shown in Figure 87, using only one centroid for a long hallway with corners caused an anomaly with route generation in the center of the hallway and caused all routes for that hallway to map to a single node. This can be solved by sub-spacing such spaces.

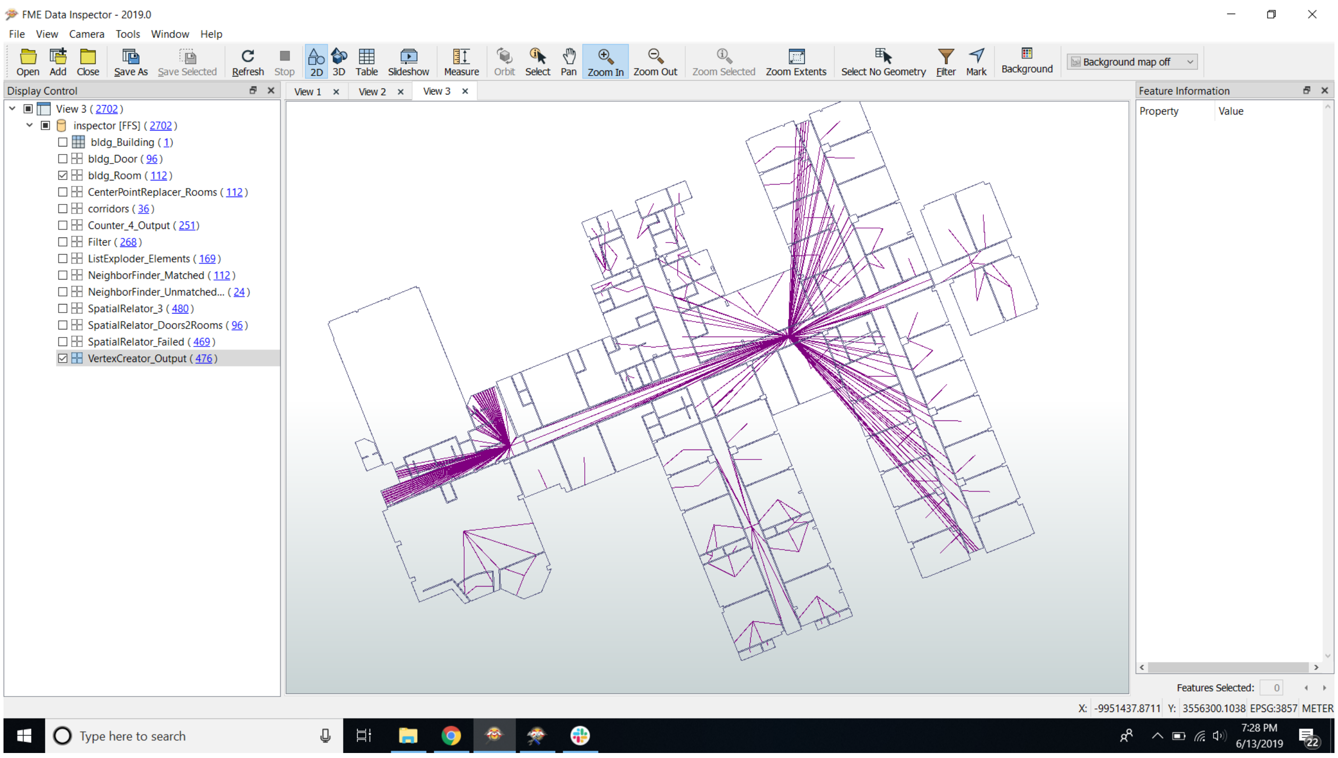

FME Data Inspector - 2019.0
File View Camera Tools Window Help
Open Add Close Save As Save Selected Refresh Stop 2D 3D Table Slideshow Measure Orbit Select Pan Zoom In Zoom Out Zoom Selected Zoom Extents Select No Geometry Filter Mark Background
Background map off
Display Control
View 3 (2702)
inspector [FFS] (2702)
bldg_Building (1)
bldg_Door (96)
bldg_Room (112)
CenterPointReplacer_Rooms (112)
corridors (36)
Counter_4_Output (251)
Filter (268)
ListExploder_Elements (169)
NeighborFinder_Matched (112)
NeighborFinder_Unmatched... (24)
SpatialRelator_3 (480)
SpatialRelator_Doors2Rooms (96)
SpatialRelator_Failed (469)
VertexCreator_Output (476)
View 1 View 2 View 3
Feature Information
Property Value
Features Selected: 0
X: -9951437.8711 Y: 3556300.1038 EPSG:3857 METER
Type here to search
7:28 PM
6/13/2019

Figure 87. Anomaly when generating a route network without proper space segmentation

- When generating a mesh of points for indoor spaces, it is important to rotate the overlaid graph to ensure alignment with the georeferenced IndoorGML. The orientation of the IndoorGML model affects the way in which the routing nodes are generated and requires some calibration of the orientation angle in order to ensure points are aligned. This can be seen in Figure 88. Early network generation in Hancock data was not aligned with the corridors yielding a network that produced irregular paths.

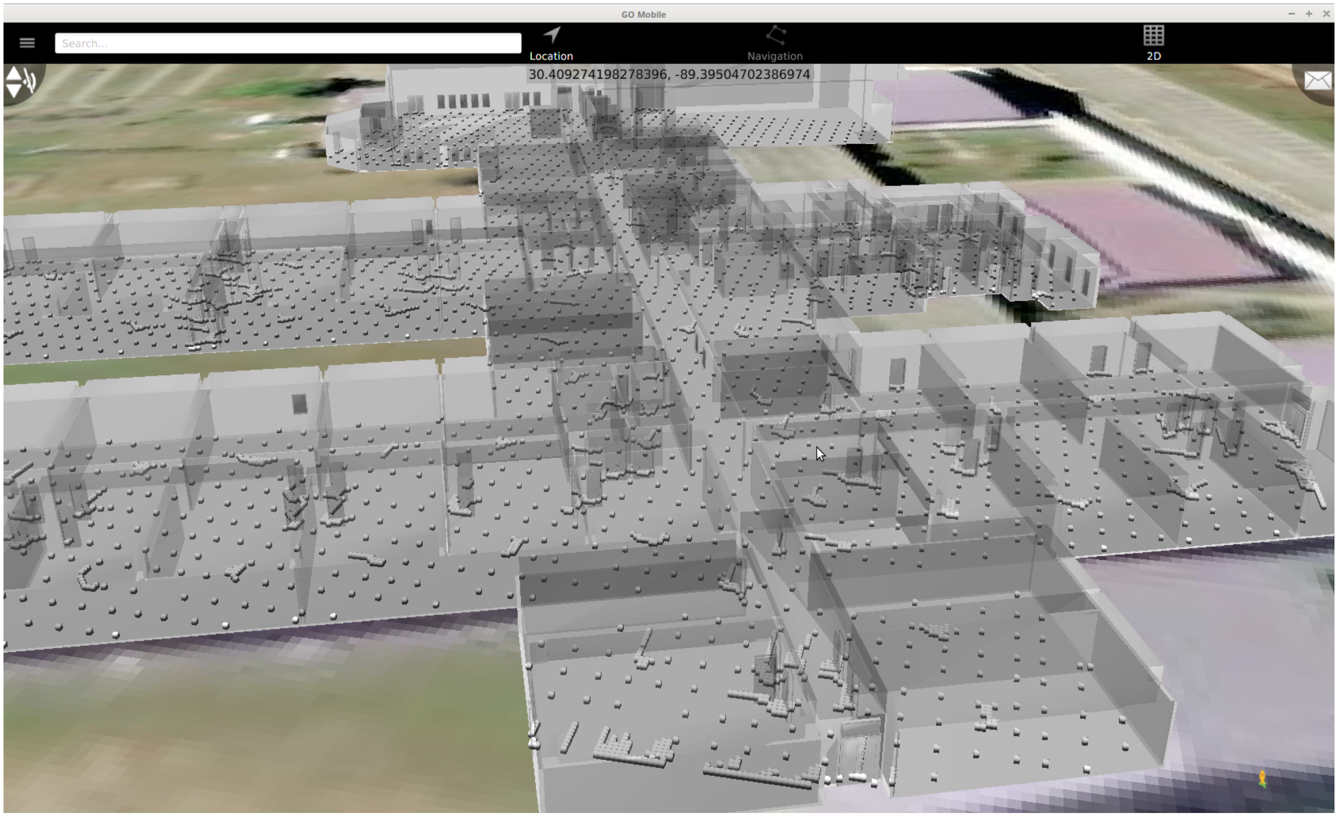


Figure 88. Grid Misalignment Overlayed on a Map

- This pilot would have benefited from the use of clean reference data for developing standards. The primary data still introduced issues that were unresolvable from automated data conversion which impacted the overall project. From a public safety perspective, data capture and structure are critical, particularly the RGB data of a point cloud. This affects how doors and windows can be identified. Having an accurate representation of doors is critical to a public safety scenario including which doors are closed and/or locked. Indoor navigation routing requires these doors, but point cloud scans do not easily detect closed doors. During the scanning process, the person executing the scan needs to make every effort to identify every door (and open it if possible) and identify locked doors. Richer data leads to better results.
- Automatic conversion of point cloud data into CityGML resulted in significant levels of noise including false walls and incorrect geometries. Reflective floors and transparent glass cause artifacts in the point cloud data. Therefore, it was determined that the output data needed to be cleansed manually in order to be used for the pilot demonstration. Automation tools and processes are not yet mature enough yet to generate models for navigation purposes without manual intervention and data cleansing.
- Existing CityGML tools did not provide the type of editing functionality required. The IFC standard was easier to edit using REVIT. For this reason, Faramoon chose to convert first to IFC for editing and cleaning, and then to CityGML. PNU also used an editing tool - in this case TICA - to clean up their GML data. Some issues may have been created due to the conversion of PC data into IFC data and then IFC into CityGML. These issues have to do with differences in the data model formats. Potential issues may be mitigated by reducing the number of conversions. So, there is a trade-off between editing tool functionality and the conversions required to make use of that tool. Ideally it would be best to convert directly from PC to CityGML. However, the output of both approaches proved to be usable for indoor navigation purposes. In the future, further development of tools for this application is needed to improve the maturity, capability, and ease of use. Also, quality control and validation methods can mitigate issues raised by data model differences and conversion.
- IndoorGML Core does not provide all requirements for Public Safety applications, so this is why the Public Safety Extension is needed.
- IndoorGML can be used for improving navigation instructions within a building. However, the data model could be harmonized with other indoor formats from industry standards (e.g., Apple IMDF, ESRI Indoor, etc.) to promote cross interoperability.
- Data conversions between data formats with established open geospatial standards such as CityGML and IndoorGML proved to be relatively straightforward to apply to new datasets. Once the transform processes or models were defined, developed and validated, the same conversion processes could be applied to new datasets with significantly less reconfiguration or development effort with some variations depending on tools or libraries employed. In short, the use of open standards directly supported efforts towards increasing automation and scalability, both of which are critical for any potential new system to move beyond prototype to production.

11.4.2. Recommendations for Future Work

- This pilot focused on working with indoor data. For the purposes of public safety and other mapping applications, the distinction between indoor and outdoor is somewhat arbitrary. Often public safety incidents may involve multiple buildings and resources need to take into account road and water networks and other accessibility considerations. Future work related to indoor mapping should take into account how best to interface with outdoor mapping and utilizes data sources such as CityGML which can model both.

- Rendering the symbology for Public Safety is not well defined in the context of IndoorGML and CityGML. For the demonstration, some notations can be made in CityGML and IndoorGML for symbology. The NAPSG symbology within the CityGML Public Safety ADE and the IndoorGML Public Safety Extension could be encoded using Style Layer Descriptors (SLD). SLD currently supports 2-dimensional rendering, but future support for 3-dimensional rendering could be considered.
- Consideration could be made toward the development and improvement of tools (e.g., TICA) that combine machine learning and manual processes with interactive guidance to intelligently identify surfaces generated using ML or Deep Learning processes.
- The CityGML data output created from Point Cloud data may be valid, but not usable when converted into IndoorGML. The spaces need to be properly segmented into subspaces with each space defining a fully enclosed volume. For example, some of the early approaches yielded building models composed of just a few large objects. From a CityGML perspective, this result might look adequate in a visual representation, but not be usable for navigation purposes because the individual rooms and doors are not defined. For use in a navigational model, additional checks are needed to ensure connectivity and sub-spacing is adequate. Future work should include investigation to formalize what business rules, checks, and validations are needed to define a successful conversion from CityGML to IndoorGML.
- CityGML 3.0 has some enhancements that relate to indoor mapping and hold promise as an intermediary format. Future work on IndoorGML should consider closer coordination with CityGML 3.0 developments.
- Refine approaches for generating navigation routes in IndoorGML for turn-by-turn navigation and for optimized routes. Optimization methods could be developed for calculating by distance, by hazard avoidance, and potentially by taking into account breakable windows and walls.
- In this pilot, most enrichment related to Public Safety information was done manually as Public Safety information was not readily available for the test areas. On the one hand, most indoor applications require frequent updates in order to remain current and relevant. On the other hand, geometry processing is typically expensive in terms of processing and effort. Thus, typically the most efficient way of providing updates is via tables of attribute information related to a common key such as room ID. Going forward, future applications should provide a means of combining non-spatial room attribute table information with room spaces based on a common ID. This would enable many updates to be performed autonomously.
- Due to the scope and early stage of this pilot, work was necessarily focused on data generation and modeling. Future work would benefit from a greater focus on web and mobile clients. IndoorGML is designed as an exchange format for indoor data, and is optimized for that. It is not an ideal client for many end user applications, especially for mobile clients. Typically, mobile clients are easier to support using lightweight data streams based on JSON. Future work should look at integrating with the emerging OGC OpenAPIs and perhaps explore an Indoor light data model deployable via GeoJSON over WFS3.
- Another area that will benefit from significant research and development is studying how indoor maps and building models can be used to guide first responders during emergency events. Integration with real-time position and situational information will be critical,

combined with methods of continually providing updates of the building structures, accessibility, and environmental conditions. Methods for enriching awareness through the use of Augmented Reality (AR) should also be explored as this would be very beneficial for first responders working within a restricted visibility environment, as well as adding the ability to see through barriers to assess possible hazards such as gas lines or hot spots. Ways of combining multiple technologies could also be explored. With a comprehensive BIM-oriented 3D model as a frame of reference, point cloud updates combined with drone and robot derived infrared imagery could be used to develop a dynamic model of a fire in progress which could then inform AR systems guiding responders.

Appendix A: Building Repository Examples

A.1. CSW Insert Transaction POST Request Example

```
https://ogc-indoor-pilot.compusult.net/wes/serviceManagerCSW/cswt
```

```
<?xml version="1.0" encoding="UTF-8"?>
<csw:Transaction xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"   xmlns:dc=
"http://purl.org/dc/elements/1.1/" xmlns:dct="http://purl.org/dc/terms/"   xmlns:gmd=
"http://www.isotc211.org/2005/gmd" xmlns:gml="http://www.opengis.net/gml" xmlns:ogc=
"http://www.opengis.net/ogc" xmlns:ows="http://www.opengis.net/ows"   xmlns:rim=
"urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0" xmlns:wrs=
"http://www.opengis.net/cat/wrs/1.0"  xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"    service="CSW"
  version="2.0.2" xmlns:xml="http://www.w3.org/XML/1998/namespace"
xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2
http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0 http://docs.oasis-open.org/regrep/v3.0/schema/rim.xsd
http://www.opengis.net/cat/wrs/1.0
http://schemas.opengis.net/csw/2.0.2/profiles/ebrim/1.0/csw-ebrim.xsd">
  <csw:Insert>
    <wrs:ExtrinsicObject id="c7694e02-01b7-4267-8827-2eef63b84807"          mimeType=
"text/xml" objectType="urn:ogc:def:ebRIM-ObjectType:OGC:Dataset:3DData:Lidar">
      <rim:Slot name="http://purl.org/dc/terms/modified"                slotType=
"urn:oasis:names:tc:ebxml-regrep:DataType:DateTime">
        <rim:ValueList>
          <rim:Value>2019-01-17T13:01:11</rim:Value>
        </rim:ValueList>
      </rim:Slot>
      <rim:Slot name="http://purl.org/dc/terms/created"              slotType=
"urn:oasis:names:tc:ebxml-regrep:DataType:DateTime">
        <rim:ValueList>
          <rim:Value>2019-01-17T13:01:11</rim:Value>
        </rim:ValueList>
      </rim:Slot>
      <rim:Slot name="http://purl.org/dc/elements/1.1/language"             slotType=
"urn:oasis:names:tc:ebxml-regrep:DataType:String">
        <rim:ValueList>
          <rim:Value>en</rim:Value>
        </rim:ValueList>
      </rim:Slot>
      <rim:Slot name="http://purl.org/dc/terms/dateSubmitted"               slotType=
"urn:oasis:names:tc:ebxml-regrep:DataType:DateTime">
        <rim:ValueList>
          <rim:Value>2019-01-17T01:01:11</rim:Value>
        </rim:ValueList>
      </rim:Slot>
      <rim:Slot name="http://purl.org/dc/terms/spatial" slotType=
"urn:ogc:def:dataType:ISO-19107:GM_Envelope">
        <wrs:ValueList>
          <wrs:AnyValue>
            <gml:Envelope srsName="EPSG:4326">
              <gml:lowerCorner>-180.0 -90.0</gml:lowerCorner>
              <gml:upperCorner>180.0 90.0</gml:upperCorner>
            </gml:Envelope>
          </wrs:AnyValue>
        </wrs:ValueList>
      </rim:Slot>
      <rim:Slot name="Editable"             slotType="urn:oasis:names:tc:ebxml-
regrep:DataType:Boolean">
        <rim:ValueList>
          <rim:Value>true</rim:Value>
        </rim:ValueList>
      </rim:Slot>
      <rim:Slot name="Transfer Option Type"             slotType=
"urn:oasis:names:tc:ebxml-regrep:DataType:String">
        <rim:ValueList>
          <rim:Value>FTP</rim:Value>
        </rim:ValueList>
      </rim:Slot>
      <rim:Slot name="Transfer Option Name"             slotType=
"urn:oasis:names:tc:ebxml-regrep:DataType:String">
        <rim:ValueList>
          <rim:Value>Lidar data of Korean Mall</rim:Value>
        </rim:ValueList>
      </rim:Slot>
      <rim:Slot name="Transfer Option Description" slotType="urn:oasis:names:tc:ebxml-
regrep:DataType:String">
          <rim:ValueList>
              <rim:Value>Description of Lidar data of Korean Mall</rim:Value>
          </rim:ValueList>
      </rim:Slot>
      <rim:Slot name="Transfer Option URL" slotType="urn:oasis:names:tc:ebxml-
regrep:DataType:URI">
          <rim:ValueList>
          <rim:Value>
              ftp://user:password@ftp.somewhere.org/PointCloudData/filename.zip
          </rim:Value>
      </rim:ValueList>
      </rim:Slot>
      <rim:Name>
        <rim:LocalizedString charset="UTF-8" xml:lang="en"                 value=
"Lidar data of Korean mall" />
      </rim:Name>
      <rim:Description>
        <rim:LocalizedString charset="UTF-8" xml:lang="en" value="Description for
Lidar data of Korean mall"/>
      </rim:Description>
      <wrs:repositoryItemRef                xlink:href="http://ogc-indoor-
pilot.compusult.net/wes/serviceManagerCSW/csw"                xlink:type="simple" />
    </wrs:ExtrinsicObject>
    <rim:Classification classificationNode="urn:ogc:def:ebRIM-
ObjectType:OGC:Dataset:3DData:Lidar"          classificationScheme=
"urn:ogc:def:ObjectType:OtherResources"            classifiedObject="c8694e02-01b7-
4267-8827-2eef63b84807" id="dd27a7b5-e8f9-c2a9-633c-d61afa453012" />
  </csw:Insert>
</csw:Transaction>
```

A.2. CSW Insert Transaction Response Example

```
<?xml version="1.0" encoding="UTF-8"?>
<csw:TransactionResponse xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
  <csw:TransactionSummary>
    <csw:totalInserted>1</csw:totalInserted>
    <csw:totalUpdated>0</csw:totalUpdated>
    <csw:totalDeleted>0</csw:totalDeleted>
  </csw:TransactionSummary>
</csw:TransactionResponse>
```

A.3. CSW Update Transaction Request Example

```
https://ogc-indoor-pilot.compusult.net/wes/serviceManagerCSW/cswt
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<csw:Transaction xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"   xmlns:dc=
"http://purl.org/dc/elements/1.1/" xmlns:dct="http://purl.org/dc/terms/"   xmlns:gmd=
"http://www.isotc211.org/2005/gmd" xmlns:gml="http://www.opengis.net/gml" xmlns:ogc=
"http://www.opengis.net/ogc" xmlns:ows="http://www.opengis.net/ows"   xmlns:rim=
"urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0" xmlns:wrs=
"http://www.opengis.net/cat/wrs/1.0"  xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"    service="CSW" version="2.0.2"
xmlns:xml="http://www.w3.org/XML/1998/namespace"  xsi:schemaLocation=
"http://www.opengis.net/cat/csw/2.0.2 http://schemas.opengis.net/csw/2.0.2/CSW-
discovery.xsd urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0 http://docs.oasis-
open.org/regrep/v3.0/schema/rim.xsd http://www.opengis.net/cat/wrs/1.0
http://schemas.opengis.net/csw/2.0.2/profiles/ebrim/1.0/csw-ebrim.xsd">
  <csw:Update>
    <wrs:ExtrinsicObject id="c8694e02-01b7-4267-8827-2eef63b84807"          mimeType=
"text/xml" objectType="urn:ogc:def:ObjectType:OGC:GeoPackage">
      <rim:Name>
        <rim:LocalizedString charset="UTF-8" xml:lang="en"                 value=
"http://mark-UPDATED-dev.compusult.net/wes/serviceManagerCSW/csw" />
      </rim:Name>
      <rim:Description>
        <rim:LocalizedString charset="UTF-8" xml:lang="en"                 value=
"Uploaded file: http://mark-UPDATED-dev.compusult.net/wes/serviceManagerCSW/csw" />
      </rim:Description>
    </wrs:ExtrinsicObject>
  </csw:Update>
  <csw:Update>
    <wrs:ExtrinsicObject id="c9694e02-01b7-4267-8827-2eef63b84807"          mimeType=
"text/xml" objectType="urn:ogc:def:ObjectType:OGC:GeoPackage">
      <rim:Name>
        <rim:LocalizedString charset="UTF-8" xml:lang="en"                 value=
"http://mark-UPDATED2-dev.compusult.net/wes/serviceManagerCSW/csw" />
      </rim:Name>
      <rim:Description>
        <rim:LocalizedString charset="UTF-8" xml:lang="en"                 value=
"Uploaded file: http://mark-UPDATED2-dev.compusult.net/wes/serviceManagerCSW/csw" />
      </rim:Description>
    </wrs:ExtrinsicObject>
  </csw:Update>
</csw:Transaction>
```

A.4. CSW Update Transaction Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<csw:TransactionResponse xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
  <csw:TransactionSummary>
    <csw:totalInserted>0</csw:totalInserted>
    <csw:totalUpdated>2</csw:totalUpdated>
    <csw:totalDeleted>0</csw:totalDeleted>
  </csw:TransactionSummary>
</csw:TransactionResponse>
```

A.5. CSW Delete Transaction Request Example

```
https://ogc-indoor-pilot.compusult.net/wes/serviceManagerCSW/cswt
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<csw:Transaction xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"   xmlns:dc=
"http://purl.org/dc/elements/1.1/" xmlns:dct="http://purl.org/dc/terms/"   xmlns:gmd=
"http://www.isotc211.org/2005/gmd" xmlns:gml="http://www.opengis.net/gml" xmlns:ogc=
"http://www.opengis.net/ogc" xmlns:ows="http://www.opengis.net/ows"   xmlns:rim=
"urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0" xmlns:wrs=
"http://www.opengis.net/cat/wrs/1.0"  xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"    service="CSW" version="2.0.2"
xmlns:xml="http://www.w3.org/XML/1998/namespace"  xsi:schemaLocation=
"http://www.opengis.net/cat/csw/2.0.2 http://schemas.opengis.net/csw/2.0.2/CSW-
discovery.xsd urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0 http://docs.oasis-
open.org/regrep/v3.0/schema/rim.xsd http://www.opengis.net/cat/wrs/1.0
http://schemas.opengis.net/csw/2.0.2/profiles/ebrim/1.0/csw-ebrim.xsd">
  <csw:Delete>
    <csw:Constraint version='1.0.0'>
      <ogc:Filter>
        <ogc:PropertyIsEqualTo>
          <ogc:PropertyName>DOES NOT MATTER, USES ID</ogc:PropertyName>
          <ogc:Literal>c7694e02-01b7-4267-8827-2eef63b84807</ogc:Literal>
        </ogc:PropertyIsEqualTo>
      </ogc:Filter>
    </csw:Constraint>
  </csw:Delete>
</csw:Transaction>
```

A.6. CSW Delete Transaction Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<csw:TransactionResponse xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
  <csw:TransactionSummary>
    <csw:totalInserted>0</csw:totalInserted>
    <csw:totalUpdated>0</csw:totalUpdated>
    <csw:totalDeleted>1</csw:totalDeleted>
  </csw:TransactionSummary>
</csw:TransactionResponse>
```

A.7. CSW GetRecords Request Example

```
https://ogc-indoor-pilot.compusult.net/wes/serviceManagerCSW/csw
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<csw:GetRecords xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:dct="http://purl.org/dc/terms/"
xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc" xmlns:ows="http://www.opengis.net/ows"
xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0" xmlns:wrs=
"http://www.opengis.net/cat/wrs/1.0"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
maxRecords="10" outputFormat="application/xml" outputSchema=
"http://www.isotc211.org/2005/gmd"
resultType="results" service="CSW" startPosition="1" version="2.0.2"
xmlns:xml="http://www.w3.org/XML/1998/namespace"
xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2
http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0 http://docs.oasis-open.org/regrep/v3.0/schema/rim.xsd
http://www.opengis.net/cat/wrs/1.0
http://schemas.opengis.net/csw/2.0.2/profiles/ebrim/1.0/csw-ebrim.xsd">
    <csw:Query typeNames="csw:Record Classification_resourceTypes">
        <csw:ElementSetName typeNames="csw:Record">full</csw:ElementSetName>
        <csw:Constraint version="1.1.0">
            <ogc:Filter>
                <ogc:And>
                    <ogc:Intersects>
                        <ogc:PropertyName>ows:BoundingBox</ogc:PropertyName>
                        <gml:Envelope>
                            <gml:lowerCorner srsName="EPSG:4326">-180.0 -
90.0</gml:lowerCorner>
                            <gml:upperCorner srsName="EPSG:4326">180.0
90.0</gml:upperCorner>
                        </gml:Envelope>
                    </ogc:Intersects>
                </ogc:And>
            </ogc:Filter>
        </csw:Constraint>
    </csw:Query>
</csw:GetRecords>
```

A.8. CSW GetRecords Response Example

```
<?xml version="1.0" encoding="UTF-8"?>
<csw:GetRecordsResponse xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" xmlns:dc=
"http://purl.org/dc/elements/1.1/" xmlns:dct="http://purl.org/dc/terms/" xmlns:gmd=
"http://www.isotc211.org/2005/gmd" xmlns:gml="http://www.opengis.net/gml" xmlns:ows=
"http://www.opengis.net/ows" xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
xmlns:wrs="http://www.opengis.net/cat/wrs/1.0" xmlns:xlink=
"http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xml="http://www.w3.org/XML/1998/namespace" xsi:schemaLocation=
"http://www.opengis.net/cat/csw/2.0.2 http://schemas.opengis.net/csw/2.0.2/CSW-
discovery.xsd">
    <csw:SearchStatus timestamp="2019-01-24T07:03:16"/>
    <csw:SearchResults elementSet="full" nextRecord="0" numberOfRecordsMatched="3"
numberOfRecordsReturned="3">
        <gmd:MD_Metadata xmlns:gco="http://www.isotc211.org/2005/gco" xmlns:gml=
"http://www.opengis.net/gml/3.2" xmlns:gmd="http://www.isotc211.org/2005/gmd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=
"http://www.isotc211.org/2005/gmd http://www.isotc211.org/2005/gmd/gmd.xsd
http://www.isotc211.org/2005/gmx http://www.isotc211.org/2005/gmx/gmx.xsd">
            <gmd:fileIdentifier>
                <gco:CharacterString>150ffeee-d3ec-47d2-97bf-
39df28c73be7</gco:CharacterString>
            </gmd:fileIdentifier>
            <gmd:language>
                <gmd:LanguageCode codeList="http://www.loc.gov/standards/iso639-2/"
codeListValue="eng"/>
            </gmd:language>
            <gmd:characterSet>
                <gmd:MD_CharacterSetCode codeList=
"http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/resources/
codelist/ML_gmxCodelists.xml#MD_CharacterSetCode" codeListValue="utf8"/>
            </gmd:characterSet>
            <gmd:hierarchyLevel>
                <gmd:MD_ScopeCode codeList=
"http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/resources/
codelist/ML_gmxCodelists.xml#MD_ScopeCode" codeListValue="dataset"/>
            </gmd:hierarchyLevel>
            <gmd:dateStamp>
                <gco:DateTime>2019-01-24T04:51:01</gco:DateTime>
            </gmd:dateStamp>
            <gmd:metadataStandardName>
                <gco:CharacterString>ISO 19119/2005</gco:CharacterString>
            </gmd:metadataStandardName>
            <gmd:metadataStandardVersion>
                <gco:CharacterString>1.0</gco:CharacterString>
            </gmd:metadataStandardVersion>
            <gmd:identificationInfo>
                <gmd:MD_DataIdentification>
                    <gmd:citation>
                        <gmd:CI_Citation>
                            <gmd:title>
                                <gco:CharacterString>IndoorGML Sample
Data</gco:CharacterString>
                            </gmd:title>
                            <gmd:date>
                                <gmd:CI_Date>
                                    <gmd:date>
                                        <gco:Date>2019-01-15</gco:Date>
                                    </gmd:date>
                                    <gmd:dateType>
```

```
                                        <gmd:CI_DateTypeCode codeList=
"http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/resources/
codelist/ML_gmxCodelists.xml#CI_DateTypeCode" codeListValue="creation"/>
                                    </gmd:dateType>
                                </gmd:CI_Date>
                            </gmd:date>
                            <gmd:edition gco:nilReason="missing">
                                <gco:CharacterString/>
                            </gmd:edition>
                        </gmd:CI_Citation>
                    </gmd:citation>
                    <gmd:abstract gco:nilReason="missing">
                        <gco:CharacterString/>
                    </gmd:abstract>
                    <gmd:status>
                        <gmd:MD_ProgressCode codeList=
"http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/resources/
codelist/ML_gmxCodelists.xml#MD_ProgressCode" codeListValue="latestAvailable"/>
                    </gmd:status>
                    <gmd:resourceMaintenance>
                        <gmd:MD_MaintenanceInformation>
                            <gmd:maintenanceAndUpdateFrequency>
                                <gmd:MD_MaintenanceFrequencyCode codeList=
"http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/resources/
codelist/ML_gmxCodelists.xml#MD_MaintenanceFrequencyCode" codeListValue="unknown"/>
                            </gmd:maintenanceAndUpdateFrequency>
                        </gmd:MD_MaintenanceInformation>
                    </gmd:resourceMaintenance>
                    <gmd:graphicOverview>
                        <gmd:MD_BrowseGraphic>
                            <gmd:fileName>
                                <gco:CharacterString>https://ogc-indoor-
pilot.compusult.net/wes/serviceManagerCSW/csw?request=GetRepositoryItem&amp;service=CS
W&amp;version=2.0.2&amp;id=84e2ee01-ecd7-720e-27f2-21f4e41067b1</gco:CharacterString>
                            </gmd:fileName>
                            <gmd:fileDescription>
                                <gco:CharacterString>Uploaded Browse
Graphic</gco:CharacterString>
                            </gmd:fileDescription>
                            <gmd:fileType>
                                <gco:CharacterString>png</gco:CharacterString>
                            </gmd:fileType>
                        </gmd:MD_BrowseGraphic>
                    </gmd:graphicOverview>
                    <gmd:descriptiveKeywords>
                        <gmd:MD_Keywords>
                            <gmd:keyword>
                                <gco:CharacterString>IMAGERY/BASE MAPS/EARTH
COVER</gco:CharacterString>
                            </gmd:keyword>
                            <gmd:keyword>
                                <gco:CharacterString>STRUCTURE</gco:CharacterString>
                            </gmd:keyword>
                            <gmd:thesaurusName>
                                <gmd:CI_Citation>
                                    <gmd:title>
                                        <gco:CharacterString>ISO Topic
Category</gco:CharacterString>
                                    </gmd:title>
                                    <gmd:alternateTitle>
                                        <gco:CharacterString>ISO Topic
Category</gco:CharacterString>
                                    </gmd:alternateTitle>
                                    <gmd:date>
                                        <gmd:CI_Date>
                                            <gmd:date>
                                                <gco:DateTime>2019-01-
24T03:59:14</gco:DateTime>
                                            </gmd:date>
                                            <gmd:dateType>
                                                <gmd:CI_DateTypeCode codeList=
"http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/resources/
codelist/ML_gmxCodelists.xml#CI_DateTypeCode" codeListValue="date"/>
                                            </gmd:dateType>
                                        </gmd:CI_Date>
                                    </gmd:date>
                                </gmd:CI_Citation>
                            </gmd:thesaurusName>
                        </gmd:MD_Keywords>
                    </gmd:descriptiveKeywords>
                    <gmd:descriptiveKeywords>
                        <gmd:MD_Keywords>
                            <gmd:keyword>
                                <gco:CharacterString>OFF-LINE DIGITAL
DATA</gco:CharacterString>
                            </gmd:keyword>
                            <gmd:thesaurusName>
                                <gmd:CI_Citation>
                                    <gmd:title>
                                        <gco:CharacterString>Data
Resources</gco:CharacterString>
                                    </gmd:title>
                                    <gmd:alternateTitle>
                                        <gco:CharacterString>Data
Resources</gco:CharacterString>
                                    </gmd:alternateTitle>
                                    <gmd:date>
                                        <gmd:CI_Date>
                                            <gmd:date>
                                                <gco:DateTime>2019-01-
24T03:59:14</gco:DateTime>
                                            </gmd:date>
```

```xml
                                            <gmd:dateType>
                                                <gmd:CI_DateTypeCode codeList=
"http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/resources/
codelist/ML_gmxCodelists.xml#CI_DateTypeCode" codeListValue="date"/>
                                            </gmd:dateType>
                                        </gmd:CI_Date>
                                    </gmd:date>
                                </gmd:CI_Citation>
                            </gmd:thesaurusName>
                        </gmd:MD_Keywords>
                    </gmd:descriptiveKeywords>
                    <gmd:language>
                        <gmd:LanguageCode codeList=
"http://www.loc.gov/standards/iso639-2/" codeListValue="eng"/>
                    </gmd:language>
                    <gmd:characterSet>
                        <gmd:MD_CharacterSetCode codeList=
"http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/resources/
codelist/ML_gmxCodelists.xml#MD_CharacterSetCode" codeListValue="utf8"/>
                    </gmd:characterSet>
                    <gmd:topicCategory>
                        <gmd:MD_TopicCategoryCode>
imageryBaseMapsEarthCover</gmd:MD_TopicCategoryCode>
                    </gmd:topicCategory>
                    <gmd:extent>
                        <gmd:EX_Extent>
                            <gmd:geographicElement>
                                <gmd:EX_GeographicBoundingBox>
                                    <gmd:westBoundLongitude>
                                        <gco:Decimal>-180</gco:Decimal>
                                    </gmd:westBoundLongitude>
                                    <gmd:eastBoundLongitude>
                                        <gco:Decimal>180</gco:Decimal>
                                    </gmd:eastBoundLongitude>
                                    <gmd:southBoundLatitude>
                                        <gco:Decimal>-90</gco:Decimal>
                                    </gmd:southBoundLatitude>
                                    <gmd:northBoundLatitude>
                                        <gco:Decimal>90</gco:Decimal>
                                    </gmd:northBoundLatitude>
                                </gmd:EX_GeographicBoundingBox>
                            </gmd:geographicElement>
                        </gmd:EX_Extent>
                    </gmd:extent>
                </gmd:MD_DataIdentification>
            </gmd:identificationInfo>
            <gmd:distributionInfo>
                <gmd:MD_Distribution>
                    <gmd:transferOptions>
                        <gmd:MD_DigitalTransferOptions>
                            <gmd:onLine>
```

```
                                <gmd:CI_OnlineResource>
                                    <gmd:linkage>

<gmd:URL>ftp://user:password@ftp.somewhere.org/IndoorGML-Sample.gml</gmd:URL>
                                    </gmd:linkage>
                                    <gmd:protocol>
                                        <gco:CharacterString>FTP</gco:CharacterString>
                                    </gmd:protocol>
                                    <gmd:name>
                                        <gco:CharacterString>IndoorGML-Navi-Sample-
PNU201-2019-01-15.gml</gco:CharacterString>
                                    </gmd:name>
                                    <gmd:description>
                                        <gco:CharacterString>IndoorGML Sample
Data</gco:CharacterString>
                                    </gmd:description>
                                    <gmd:function>
                                        <gmd:CI_OnLineFunctionCode codeList=
"http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/resources/
codelist/ML_gmxCodelists.xml#CI_OnLineFunctionCode" codeListValue=""/>
                                    </gmd:function>
                                </gmd:CI_OnlineResource>
                            </gmd:onLine>
                            <gmd:onLine>
                                <gmd:CI_OnlineResource>
                                    <gmd:linkage>

<gmd:URL>ftp://user:password@ftp.somewhere.org/PointCloudFile.zip</gmd:URL>
                                    </gmd:linkage>
                                    <gmd:protocol>
                                        <gco:CharacterString>FTP</gco:CharacterString>
                                    </gmd:protocol>
                                    <gmd:name>
                                        <gco:CharacterString>Point Cloud
Data</gco:CharacterString>
                                    </gmd:name>
                                    <gmd:description>
                                        <gco:CharacterString>KU-HanaSquare-
v2</gco:CharacterString>
                                    </gmd:description>
                                    <gmd:function>
                                        <gmd:CI_OnLineFunctionCode codeList=
"http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/resources/
codelist/ML_gmxCodelists.xml#CI_OnLineFunctionCode" codeListValue=""/>
                                    </gmd:function>
                                </gmd:CI_OnlineResource>
                            </gmd:onLine>
                        </gmd:MD_DigitalTransferOptions>
                    </gmd:transferOptions>
                </gmd:MD_Distribution>
            </gmd:distributionInfo>
            <gmd:dataQualityInfo>
                <gmd:DQ_DataQuality>
                    <gmd:scope>
                        <gmd:DQ_Scope>
                            <gmd:level>
                                <gmd:MD_ScopeCode codeList=
"http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/resources/
codelist/ML_gmxCodelists.xml#MD_ScopeCode" codeListValue="dataset"/>
                            </gmd:level>
                        </gmd:DQ_Scope>
                    </gmd:scope>
                </gmd:DQ_DataQuality>
            </gmd:dataQualityInfo>
        </gmd:MD_Metadata>
    </csw:SearchResults>
</csw:GetRecordsResponse>
```

A.9. CSW GetRecordByID GET Request Example

```
https://ogc-indoor-
pilot.compusult.net/wes/serviceManagerCSW/csw?Service=CSW&Version=2.0.2&request=GetRec
ordByID&id=150ffeee-d3ec-47d2-97bf-
39df28c73be7&elementSetName=summary&outputSchema=http://www.isotc211.org/2005/gmd
```

A.10. CSW GetRecordByID POST Request Example

```
https://ogc-indoor-pilot.compusult.net/wes/serviceManagerCSW/csw
```

```
<csw:GetRecordById service="CSW" version="2.0.2" outputFormat="application/xml"
outputSchema="http://www.isotc211.org/2005/gmd" xmlns:csw=
"http://www.opengis.net/cat/csw/2.0.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2
http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd">
  <csw:Id>150ffeee-d3ec-47d2-97bf-39df28c73be7</csw:Id>
  <csw:ElementSetName>summary</csw:ElementSetName>
</csw:GetRecordById>
```

A.11. CSW GetRecordByID Response Example

```
<?xml version="1.0" encoding="UTF-8"?>
<csw:GetRecordByIdResponse xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" xmlns:dc=
"http://purl.org/dc/elements/1.1/" xmlns:dct="http://purl.org/dc/terms/" xmlns:gmd=
"http://www.isotc211.org/2005/gmd" xmlns:gml="http://www.opengis.net/gml" xmlns:ows=
"http://www.opengis.net/ows" xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
```

```
xmlns:wrs="http://www.opengis.net/cat/wrs/1.0" xmlns:xlink=
"http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xml="http://www.w3.org/XML/1998/namespace" xsi:schemaLocation=
"http://www.opengis.net/cat/csw/2.0.2 http://schemas.opengis.net/csw/2.0.2/CSW-
discovery.xsd">
  <gmd:MD_Metadata xmlns:gco="http://www.isotc211.org/2005/gco" xmlns:gml=
"http://www.opengis.net/gml/3.2" xmlns:gmd="http://www.isotc211.org/2005/gmd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <gmd:fileIdentifier>
      <gco:CharacterString>150ffeee-d3ec-47d2-97bf-39df28c73be7</gco:CharacterString>
    </gmd:fileIdentifier>
    <gmd:characterSet>
      <gmd:MD_CharacterSetCode codeList=
"http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/resources/
codelist/ML_gmxCodelists.xml#MD_CharacterSetCode" codeListValue="utf8"/>
    </gmd:characterSet>
    <gmd:hierarchyLevel>
      <gmd:MD_ScopeCode codeList=
"http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/resources/
codelist/ML_gmxCodelists.xml#MD_ScopeCode" codeListValue="dataset"/>
    </gmd:hierarchyLevel>
    <gmd:dateStamp>
      <gco:DateTime>2019-01-24T04:51:01</gco:DateTime>
    </gmd:dateStamp>
    <gmd:metadataStandardName>
      <gco:CharacterString>ISO 19119/2005</gco:CharacterString>
    </gmd:metadataStandardName>
    <gmd:metadataStandardVersion>
      <gco:CharacterString>1.0</gco:CharacterString>
    </gmd:metadataStandardVersion>
    <gmd:identificationInfo>
      <gmd:MD_DataIdentification xmlns:srv="http://www.isotc211.org/2005/srv">
        <gmd:citation>
          <gmd:CI_Citation>
            <gmd:title>
              <gco:CharacterString>IndoorGML Sample Data</gco:CharacterString>
            </gmd:title>
          </gmd:CI_Citation>
        </gmd:citation>
        <gmd:graphicOverview>
          <gmd:MD_BrowseGraphic>
            <gmd:fileName>
              <gco:CharacterString>https://ogc-indoor-
pilot.compusult.net/wes/serviceManagerCSW/csw?request=GetRepositoryItem&amp;service=CS
W&amp;version=2.0.2&amp;id=84e2ee01-ecd7-720e-27f2-21f4e41067b1</gco:CharacterString>
            </gmd:fileName>
          </gmd:MD_BrowseGraphic>
        </gmd:graphicOverview>
        <gmd:extent>
          <gmd:EX_Extent>
            <gmd:geographicElement>
```

```xml
                <gmd:EX_GeographicBoundingBox>
                  <gmd:westBoundLongitude>
                    <gco:Decimal>-180</gco:Decimal>
                  </gmd:westBoundLongitude>
                  <gmd:eastBoundLongitude>
                    <gco:Decimal>180</gco:Decimal>
                  </gmd:eastBoundLongitude>
                  <gmd:southBoundLatitude>
                    <gco:Decimal>-90</gco:Decimal>
                  </gmd:southBoundLatitude>
                  <gmd:northBoundLatitude>
                    <gco:Decimal>90</gco:Decimal>
                  </gmd:northBoundLatitude>
                </gmd:EX_GeographicBoundingBox>
              </gmd:geographicElement>
            </gmd:EX_Extent>
          </gmd:extent>
          <gmd:characterSet>
            <gmd:MD_CharacterSetCode codeList=
"http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19139_Schemas/resources/
codelist/ML_gmxCodelists.xml#MD_CharacterSetCode" codeListValue="utf8"/>
          </gmd:characterSet>
          <gmd:topicCategory>
            <gmd:MD_TopicCategoryCode>
imageryBaseMapsEarthCover</gmd:MD_TopicCategoryCode>
          </gmd:topicCategory>
        </gmd:MD_DataIdentification>
      </gmd:identificationInfo>
      <gmd:distributionInfo>
        <gmd:MD_Distribution xmlns:srv="http://www.isotc211.org/2005/srv">
          <gmd:transferOptions>
            <gmd:MD_DigitalTransferOptions>
              <gmd:onLine>
                <gmd:CI_OnlineResource>
                  <gmd:linkage>
                    <gmd:URL>
ftp://user:password@ftp.somewhere.org/filename.gml</gmd:URL>
                  </gmd:linkage>
                  <gmd:linkage>
                    <gmd:URL>
ftp://user:password@ftp.somewhere.org/pointcloud.zip</gmd:URL>
                  </gmd:linkage>
                </gmd:CI_OnlineResource>
              </gmd:onLine>
            </gmd:MD_DigitalTransferOptions>
          </gmd:transferOptions>
        </gmd:MD_Distribution>
      </gmd:distributionInfo>
    </gmd:MD_Metadata>
</csw:GetRecordByIdResponse>
```

Appendix B: Skymantics Indoor Navigation Service Examples

B.1. Indoor Navigation WPS Request Example

```
<wps:Execute xmlns:wps="http://www.opengis.net/wps/2.0"
  xmlns:ows="http://www.opengis.net/ows/2.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd"
service="WPS"
        version="2.0.0" response="document" mode="sync">
<ows:Identifier>org.n52.javaps.service.NIST006Test</ows:Identifier>
              <wps:Input id="indoorGMLId">
                 <wps:Data
mimeType="text/xml"><wps:LiteralValue>a989ea1c-360c-4e8d-a171-
b7f2dbccc828</wps:LiteralValue></wps:Data>
              </wps:Input>
             <wps:Input id="start">
                 <wps:Data
mimeType="text/xml"><wps:LiteralValue>St_188</wps:LiteralValue></wps:Data>
             </wps:Input>
             <wps:Input id="end">
                 <wps:Data
mimeType="text/xml"><wps:LiteralValue>St_229</wps:LiteralValue></wps:Data>
             </wps:Input>
       <wps:Output id="route" mimeType="application/xml" transmission="value"/>
</wps:Execute>
```

B.2. Indoor Navigation WPS Response Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns4:Route xmlns:ns1="http://www.opengis.net/gml/3.2" xmlns:ns2=
"http://www.w3.org/1999/xlink" xmlns:ns3="http://www.opengis.net/indoorgml/1.0/core"
xmlns:ns4="http://www.opengis.net/indoorgml/1.0/navigation">
    <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="
true"/>
    <ns4:routeNodes>
        <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil=
"true"/>
        <ns4:nodeMember>
            <ns4:RouteNode>
                <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
                <ns4:geometry>
                    <ns1:Point ns1:id="P23">
```

```
                        <ns1:pos>445538.526005699 5444902.33091888 3.44</ns1:pos>
                    </ns1:Point>
                </ns4:geometry>
            </ns4:RouteNode>
        </ns4:nodeMember>
        <ns4:nodeMember>
            <ns4:RouteNode>
                <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
                <ns4:geometry>
                    <ns1:Point ns1:id="P12">
                        <ns1:pos>445538.526005699 5444902.30232276 0.66</ns1:pos>
                    </ns1:Point>
                </ns4:geometry>
            </ns4:RouteNode>
        </ns4:nodeMember>
        <ns4:nodeMember>
            <ns4:RouteNode>
                <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
                <ns4:geometry>
                    <ns1:Point ns1:id="P23">
                        <ns1:pos>445538.526005699 5444902.33091888 3.44</ns1:pos>
                    </ns1:Point>
                </ns4:geometry>
            </ns4:RouteNode>
        </ns4:nodeMember>
        <ns4:nodeMember>
            <ns4:RouteNode>
                <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
                <ns4:geometry>
                    <ns1:Point ns1:id="P12">
                        <ns1:pos>445538.526005699 5444902.30232276 0.66</ns1:pos>
                    </ns1:Point>
                </ns4:geometry>
            </ns4:RouteNode>
        </ns4:nodeMember>
        <ns4:nodeMember>
            <ns4:RouteNode>
                <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
                <ns4:geometry>
                    <ns1:Point ns1:id="P12">
                        <ns1:pos>445538.526005699 5444902.30232276 0.66</ns1:pos>
                    </ns1:Point>
                </ns4:geometry>
            </ns4:RouteNode>
        </ns4:nodeMember>
        <ns4:nodeMember>
            <ns4:RouteNode>
```

```
                <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
                <ns4:geometry>
                    <ns1:Point ns1:id="P3">
                        <ns1:pos>445538.543473167 5444902.27372664 -2.02</ns1:pos>
                    </ns1:Point>
                </ns4:geometry>
            </ns4:RouteNode>
        </ns4:nodeMember>
        <ns4:nodeMember>
            <ns4:RouteNode>
                <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
                <ns4:geometry>
                    <ns1:Point ns1:id="P3">
                        <ns1:pos>445538.543473167 5444902.27372664 -2.02</ns1:pos>
                    </ns1:Point>
                </ns4:geometry>
            </ns4:RouteNode>
        </ns4:nodeMember>
        <ns4:nodeMember>
            <ns4:RouteNode>
                <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
                <ns4:geometry>
                    <ns1:Point ns1:id="P1">
                        <ns1:pos>445536.499779417 5444906.24858758 -2.02</ns1:pos>
                    </ns1:Point>
                </ns4:geometry>
            </ns4:RouteNode>
        </ns4:nodeMember>
    </ns4:routeNodes>
    <ns4:path>
        <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil=
"true"/>
        <ns4:routeMember>
            <ns4:RouteSegment>
                <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
                <ns4:weight>0.0</ns4:weight>
                <ns4:connects>
                    <ns4:RouteNode>
                        <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:nil="true"/>
                        <ns4:geometry>
                            <ns1:Point ns1:id="P23">
<ns1:pos>445538.526005699 5444902.33091888 3.44</ns1:pos>
                            </ns1:Point>
                        </ns4:geometry>
                    </ns4:RouteNode>
                </ns4:connects>
                <ns4:connects>
                    <ns4:RouteNode>
                        <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:nil="true"/>
                        <ns4:geometry>
                            <ns1:Point ns1:id="P12">
<ns1:pos>445538.526005699 5444902.30232276 0.66</ns1:pos>
                            </ns1:Point>
                        </ns4:geometry>
                    </ns4:RouteNode>
                </ns4:connects>
                <ns4:geometry>
                    <ns1:LineString ns1:id="LS24">
                        <ns1:pos>445538.526005699 5444902.33091888 3.44</ns1:pos>
                        <ns1:pos>445538.526005699 5444902.30232276 0.66</ns1:pos>
                    </ns1:LineString>
                </ns4:geometry>
            </ns4:RouteSegment>
        </ns4:routeMember>
        <ns4:routeMember>
            <ns4:RouteSegment>
                <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
                <ns4:weight>0.0</ns4:weight>
                <ns4:connects>
                    <ns4:RouteNode>
                        <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:nil="true"/>
                        <ns4:geometry>
                            <ns1:Point ns1:id="P23">
<ns1:pos>445538.526005699 5444902.33091888 3.44</ns1:pos>
                            </ns1:Point>
                        </ns4:geometry>
                    </ns4:RouteNode>
                </ns4:connects>
                <ns4:connects>
                    <ns4:RouteNode>
                        <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:nil="true"/>
                        <ns4:geometry>
                            <ns1:Point ns1:id="P12">
<ns1:pos>445538.526005699 5444902.30232276 0.66</ns1:pos>
                            </ns1:Point>
                        </ns4:geometry>
                    </ns4:RouteNode>
                </ns4:connects>
                <ns4:geometry>
                    <ns1:LineString ns1:id="LS24">
                        <ns1:pos>445538.526005699 5444902.33091888 3.44</ns1:pos>
                        <ns1:pos>445538.526005699 5444902.30232276 0.66</ns1:pos>
                    </ns1:LineString>
```

```xml
                </ns4:geometry>
            </ns4:RouteSegment>
        </ns4:routeMember>
        <ns4:routeMember>
            <ns4:RouteSegment>
                <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
                <ns4:weight>0.0</ns4:weight>
                <ns4:connects>
                    <ns4:RouteNode>
                        <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:nil="true"/>
                        <ns4:geometry>
                            <ns1:Point ns1:id="P12">
<ns1:pos>445538.526005699 5444902.30232276 0.66</ns1:pos>
                            </ns1:Point>
                        </ns4:geometry>
                    </ns4:RouteNode>
                </ns4:connects>
                <ns4:connects>
                    <ns4:RouteNode>
                        <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:nil="true"/>
                        <ns4:geometry>
                            <ns1:Point ns1:id="P3">
<ns1:pos>445538.543473167 5444902.27372664 -2.02</ns1:pos>
                            </ns1:Point>
                        </ns4:geometry>
                    </ns4:RouteNode>
                </ns4:connects>
                <ns4:geometry>
                    <ns1:LineString ns1:id="LS10">
                        <ns1:pos>445538.526005699 5444902.30232276 0.66</ns1:pos>
                        <ns1:pos>445538.543473167 5444902.27372664 -2.02</ns1:pos>
                    </ns1:LineString>
                </ns4:geometry>
            </ns4:RouteSegment>
        </ns4:routeMember>
        <ns4:routeMember>
            <ns4:RouteSegment>
                <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true"/>
                <ns4:weight>0.0</ns4:weight>
                <ns4:connects>
                    <ns4:RouteNode>
                        <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:nil="true"/>
                        <ns4:geometry>
                            <ns1:Point ns1:id="P3">
<ns1:pos>445538.543473167 5444902.27372664 -2.02</ns1:pos>
                            </ns1:Point>
                        </ns4:geometry>
                    </ns4:RouteNode>
                </ns4:connects>
                <ns4:connects>
                    <ns4:RouteNode>
                        <ns1:boundedBy xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:nil="true"/>
                        <ns4:geometry>
                            <ns1:Point ns1:id="P1">
<ns1:pos>445536.499779417 5444906.24858758 -2.02</ns1:pos>
                            </ns1:Point>
                        </ns4:geometry>
                    </ns4:RouteNode>
                </ns4:connects>
                <ns4:geometry>
                    <ns1:LineString ns1:id="LS1">
                        <ns1:pos>445538.543473167 5444902.27372664 -2.02</ns1:pos>
                        <ns1:pos>445537.914644321 5444904.59001251 -2.02</ns1:pos>
                        <ns1:pos>445536.499779417 5444906.24858758 -2.02</ns1:pos>
                    </ns1:LineString>
                </ns4:geometry>
            </ns4:RouteSegment>
        </ns4:routeMember>
    </ns4:path>
</ns4:Route>
```

B.3. SpaceLayerType Example

```
<xs:complexType name="SpaceLayerType">
            <xs:complexContent>
                        <xs:extension base="gml:AbstractFeatureType">
                                    <xs:sequence>
                                                <xs:element name="usage" type=
"gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
                                                <xs:element name="terminationDate"
type="xs:dateTime"  minOccurs="0" maxOccurs="1"/>
                                                <xs:element name="function" type=
"gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
                                                <xs:element name="creationDate" type=
"xs:dateTime"  minOccurs="0" maxOccurs="1"/>
                                                <xs:element name="class" type=
"SpaceLayerClassTypeType"  minOccurs="0" maxOccurs="1"/>
                                                <xs:element name="nodes" type=
"NodesType" minOccurs="1" maxOccurs="unbounded"/>
                                                <xs:element name="edges" type=
"EdgesType" minOccurs="0" maxOccurs="unbounded"/>
                                    </xs:sequence>
                        </xs:extension>
            </xs:complexContent>
</xs:complexType>
```

Appendix C: Compusult Indoor Navigation Service Examples

C.1. Indoor Navigation WPS DescribeProcess Request Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wps:DescribeProcess
    xmlns:ows="http://www.opengis.net/ows/2.0"
    xmlns:wps="http://www.opengis.net/wps/2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wps/2.0 ../wps.xsd"
    service="WPS"
    version="2.0.0">

    <ows:Identifier>GetNavigationRoutes</ows:Identifier>
</wps:DescribeProcess>
```

C.2. Indoor Navigation WPS DescribeProcess Response Example

The **DescribeProcess** Request will return a description of the inputs and outputs of the GetNavigationRoutes request.

```xml
<ProcessOfferings xmlns="http://www.opengis.net/wps/2.0" xmlns:ns=
"http://www.opengis.net/ows/2.0">
    <ProcessOffering jobControlOptions="sync-execute" outputTransmission="value">
        <Process>
            <ns:Title xml:lang="en">Get Navigation Routes</ns:Title>
            <ns:Abstract xml:lang="en">Get A list of Navaigation Routes based on
start/end points and a routing source.</ns:Abstract>
            <ns:Identifier>GetNavigationRoutes</ns:Identifier>
            <Input>
                <ns:Title xml:lang="en">Route Start Point</ns:Title>
                <ns:Abstract xml:lang="en">GML Representation Of Navigation Route
Start Point</ns:Abstract>
                <ns:Identifier>START_POINT</ns:Identifier>
                <ComplexData>
                    <Format mimeType="application/gml+xml" encoding="UTF-8" schema=
"http://schemas.opengis.net/gml/" default="true"/>
                </ComplexData>
            </Input>
            <Input>
                <ns:Title xml:lang="en">Route End Point</ns:Title>
                <ns:Abstract xml:lang="en">GML Representation Of Navigation Route End
Point</ns:Abstract>
                <ns:Identifier>END_POINT</ns:Identifier>
                <ComplexData>
                    <Format mimeType="application/gml+xml" encoding="UTF-8" schema=
"http://schemas.opengis.net/gml/" default="true"/>
                </ComplexData>
            </Input>
            <Input>
                <ns:Title xml:lang="en">IndoorGML ID</ns:Title>
                <ns:Abstract xml:lang="en">IndoorGML ID</ns:Abstract>
                <ns:Identifier>INDOOR_GML_ID</ns:Identifier>
                <ComplexData>
                    <Format mimeType="application/string" encoding="UTF-8" schema=
"http://schemas.opengis.net/gml/" default="true"/>
                </ComplexData>
            </Input>
            <Output>
                <Output>
                    <ns:Title xml:lang="en">Navigation Routes</ns:Title>
                    <ns:Abstract xml:lang="en">Available Navigation Routes based on
Start/End Points</ns:Abstract>
                    <ns:Identifier>NAVIGATION_ROUTES</ns:Identifier>
                    <ComplexData>
                        <Format mimeType="application/gml+xml" encoding="UTF-8"
schema="http://schemas.opengis.net/gml/" default="true"/>
                    </ComplexData>
                </Output>
            </Output>
        </Process>
    </ProcessOffering>
</ProcessOfferings>
```

C.3. Indoor Navigation WPS GetNavigationRoutes Request Example

The **GetNavigationRoutes** Request will return the navigable routes based on the criteria specified.

```xml
<wps:Execute  ...  service="WPS" version="2.0.0" response="raw" mode="sync">
    <ows:Identifier>GetNavigationRoutes</ows:Identifier>
    <wps:Input id="START_POINT">
      <wps:Data>
        <wps:ComplexData>
         <wps:Format mimeType="application/gml+xml" encoding="UTF-8" schema=
"http://schemas.opengis.net/gml/3.1.1/base/geometryBasic0d1d.xsd"/>
         <gml:Point srsName="EPSG:4326" srsDimension="3">
           <gml:description>The start point required to determine the navigation
routes</gml:description>
           <gml:identifier>START_POINT</gml:identifier>
           <gml:name>startPoint</gml:name>
           <gml:pos>1.0 1.0 1.0</gml:pos>
         </gml:Point>
        </wps:ComplexData>
      </wps:Data>
    </wps:Input>
    <wps:Input id="END_POINT">
      <wps:Data>
        <wps:ComplexData>
         <wps:Format mimeType="application/gml+xml" encoding="UTF-8" schema=
"http://schemas.opengis.net/gml/3.1.1/base/geometryBasic0d1d.xsd"/>
         <gml:Point srsName="EPSG:4326" srsDimension="3">
           <gml:description>The end point required to determine the navigation
routes</gml:description>
           <gml:identifier>END_POINT</gml:identifier>
           <gml:name>endPoint</gml:name>
           <gml:pos>1.0 2.0 1.0</gml:pos>
         </gml:Point>
        </wps:ComplexData>
      </wps:Data>
    </wps:Input>
    <wps:Input id="INDOOR_GML_ID">
      <wps:Data>hshshs-383838-ahahah</wps:Data>
    </wps:Input>
    <wps:Output id="NAVIGATION_ROUTES" mimeType="application/gml+xml" transmission=
"value"/>
</wps:Execute>
```

C.4. Indoor Navigation WPS GetNavigationRoutes Response Example

```
<gml:MultiGeometry xmlns:gml="http://schemas.opengis.net/gml/3.2.1">
   <gml:metaDataProperty>
      <gml:GenericMetaData>Any text, intermingled with:
         <!--any element-->
      </gml:GenericMetaData>
   </gml:metaDataProperty>
   <gml:description>The available routes based on user start and end
points</gml:description>
   <gml:descriptionReference/>
   <gml:identifier>AvailableRoutes</gml:identifier>
   <gml:name>AvailableRoutes</gml:name>
   <gml:geometryMembers>
    <gml:LineString>
        <gml:posList>48.6406778 -123.4316012 0 48.640678199999996 -123.4316055 0
48.640678199999996 -123.4316055 0.00015 48.6406778 -123.4316012 0.00015 48.6406778
-123.4316012 0</gml:posList>
    </gml:LineString>
    <gml:LineString>
        <gml:posList>48.6406778 -123.4316012 0 48.640678199999996 -123.4316055 0
48.640678199999996 -123.4316055 0.00015 48.6406778 -123.4316012 0.00015 48.6406778
-123.4316012 0</gml:posList>
    </gml:LineString>
   </gml:geometryMembers>
</gml:MultiGeometry>
```

Appendix D: Revision History

Table 1. Revision History

Date	**Editor**	**Release**	**Primary clauses modified**	**Descriptions**
August 16, 2019	C. Chen	1.0	various	DER for SWG Review