humantech

D2.2 – Open-source BIM authoring tools

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement nº 101058236. This document reflects only the author's view, and the EU Commission is not responsible for any use that may be made of the information it contains.

Project Title	Human-Centred Technologies for a Safer and Greener European Construction Industry.
Project Acronym	HumanTech
Grant Agreement No	101058236
Instrument	Research & Innovation Action
Topic	HORIZON-CL4-2021-TWIN-TRANSITION-01-12
Start Date of Project	June 1, 2022
Duration of Project	36 months

Name of the Deliverable	Open source BIM authoring tools: Framework and Software Modules
Number of the Deliverable	D2.2 (D6)
Related WP Number and Name	WP 2 – BIMxD Formats and Standardization
Related Task Number and Name	T2.2 – Open source BIM authoring tools
Deliverable Dissemination Level	PU – Public
Deliverable Due Date	29.02.2024
Deliverable Submission Date	29.02.2024
Task Leader/Main Author	RPTU – Fabian Kaufmann, Marius Schellen
Contributing Partners	All
Reviewer(s)	Jason Rambach, Rachele Bernardello

Keywords

Open source BIM, BIM authoring tools, IFC, FreeCAD, BlenderBIM, IfcOpenShell, BIMxD, IFC update, bSDD, IFC editing

Revisions

Version	Submission date	Comments	Author
v0.1	29.02.2024	Submitted version	Marius Schellen, Fabian Kaufmann
v0.2			
…			
…			
…			

Disclaimer

Acronyms and definitions

Acronym	Meaning
API	Application Programming Interface
BCF	BIM Collaboration Format
BIM	Building Information Modelling
bSDD	buildingSmart Data Dictionary
CAD	Computer Aided Design
CAM	Computer-Aided Manufacturing
CFD	Computational Fluid Dynamics
CNC	Computer Numerical Control
FEA	Finite Element Analysis
GUI	Graphical User Interface
IDS	Information Delivery Specifications
IFC	Industry Foundation Classes
MEP	Mechanical, Electrical and Plumbing
SMEs	Small and medium-size enterprises

Abstract

In this Deliverable, open-source tools are presented to create BIMxD objects from reconstructed semantics and geometry, object filtering and information extraction, inheritance of semantic labels from IFC objects and BIMxD update are presented.

These tools for open-source BIM authoring and the implementation in the openbimxd module provides the backbone for interacting with BIMxD representations in the HumanTech project. With the submission of the delivery, the code will be accessible on github https://github.com/humantecheu/openbimxd with a complete documentation and application examples. Naturally, the source code and documentation are part of this deliverable.

The HumanTech project

The European construction industry faces three major challenges: increase the safety and wellbeing of its workforce, improve its productivity, and become greener, making efficient use of resources.

To address these challenges, HumanTech proposes to develop **human-centred cutting-edge technologies** such as wearables for workers' safety and support and robots that can harmoniously coexist with human workers while contributing to the ecological transition of the sector.

HumanTech aims to achieve major advances in cutting-edge technologies that will enable a safe, rewarding, and digital work environment for a new generation of highly skilled construction workers and engineers.

These advances will include:

- **Robotic devices equipped with vision and intelligence** that allow them to navigate autonomously and safely in highly unstructured environments, collaborate with humans and dynamically update a semantic digital twin of the construction site in which they are.
- **Smart, unobtrusive workers protection and support equipment**. From exoskeletons activated by body sensors for posture and strain to wearable cameras and XR glasses that provide real-time workers' location and guidance for them to perform their tasks efficiently and accurately.
- An entirely new breed of **Dynamic Semantic Digital Twins (DSDTs) of construction sites** that simulate in detail the current state of a construction site at the geometric and semantic level, based on an extended Building Information Modelling (BIM) formulation that contains all relevant structural and semantic dimensions (BIMxD). BIMxDs will act as a common reference for all human workers, engineers, and autonomous machines.

The **HumanTech consortium** is formed by 22 organisations — leading research institutes and universities, innovative hi-tech SMEs, and large enterprises, construction groups and a construction SME representative — from 10 countries, bringing expertise in 11 different disciplines. The consortium is led by the German Research Center for Artificial Intelligence's Augmented Vision department.

Contents

1 Introduction

The following deliverable presents an open-source framework for editing and updating BIMxD data representation based on Industry Foundation Classes (IFC) and is related to the results from task T2.2 “open-source BIM authoring tools” of WP2 “BIMxD Formats and Standardization”. In this task the work of the tasks T2.1 “BIMxD Formats and Specifications” and T2.4 “buildingSmart Data Dictionary (bSDD) specifications and IFC extension” will be included to develop the foundation framework for the “Dynamic Semantic Digital Twin (DSDT) Generation” in WP3. Additionally, the developed framework will be applied in the integration of the Pilots in WP7 “Pilots, Evaluation and Validation”.

In general, Building Information Modelling (BIM) offers a digital technology for managing information of buildings and infrastructure facilities including planning, building, maintenance, and deconstruction. BIM goes beyond traditional 2D drawings by creating a collaborative 3D model including additional data and information throughout the project lifecycle. This fosters collaboration among various stakeholders in the construction process, including architects, engineers, contractors, and facility managers. The shared model allows real-time collaboration, reducing errors and enhancing communication. With IFC an international widespread accepted and adopted standard for BIM data was gained. IFC acts as a common language for BIM data, enabling seamless communication and data exchange between different software platforms. This interoperability is crucial for integrated and efficient project workflows, even though it was only meant for data exchange without the possibility to make changes directly to the exchanged IFC files.

The aim of WP2 is to define and extend this static standard by additional information, necessary for a safer, greener, and more efficient future of construction to reach a dynamic and flexible IFC representation. Therefore, T2.2 develops and delivers the needed frameworks to generate these IFC representations, based on open-source solutions.

1.1. Why Open BIM: The value of open-source tools

Open-source tools for BIM authoring have gained significant attention and popularity in the construction and architectural industries in recent years. The adoption of open-source principles has the potential to revolutionize the industry, especially in terms of interoperability and collaboration of small to medium-size enterprises (SMEs) and lager companies in an Open BIM character. Utilizing an Open BIM strategy with open-source tools can enhance both interoperability and collaboration within the project processes and across the entire lifespan of building assets.

Open BIM and Closed BIM represent two distinct approaches in digital construction processes, each with its set of advantages. Open BIM stands out for its commitment to interoperability, enabling seamless data exchange among diverse software applications used by architects, engineers, contractors, and other stakeholders. This interoperability is made possible through the adoption of standardized file formats and protocols, fostering collaboration across different disciplines and software platforms. One of the primary advantages of Open BIM lies in its ability to break down silos and promote collaboration in the construction industry. With standardized data formats, project teams can effortlessly share information, enhancing communication and coordination. This collaborative environment extends to various disciplines, such as architecture, structure, and MEP, allowing for more integrated and efficient design and construction processes. Moreover, Open BIM embraces vendor neutrality, empowering stakeholders to choose the most suitable software tools for their specific tasks. This flexibility ensures that each discipline can leverage specialized software while still contributing to a unified, interoperable BIM model. In contrast, Closed BIM often ties users to a specific vendor's software suite, limiting flexibility and potentially hindering the selection of tools based on individual preferences or project requirements. Open BIM excels in adapting to changes throughout the project lifecycle. Its interoperable nature allows for the smooth transfer and integration of data across different software platforms, enabling project teams to efficiently adjust workflows and incorporate modifications. In contrast, Closed BIM systems may encounter challenges in responding to changes, as the closed nature of the system can impede the seamless exchange of updated information between different disciplines and project phases. Long-term data accessibility is another notable advantage of Open BIM. By utilizing standardized formats, Open BIM reduces the risk of data obsolescence, ensuring that project information remains accessible and usable even as software tools evolve over time. In contrast, Closed BIM systems may face the risk of data lock-in, where information stored in proprietary formats becomes challenging to access or migrate to new software solutions in the future, posing potential difficulties for maintenance, renovation, or facility management. In summary, Open BIM stands as a paradigm for collaboration, interoperability, and flexibility in the construction industry. Its emphasis on standardized formats and protocols facilitates a more efficient and integrated building process compared to Closed BIM systems, which may confine collaboration within a proprietary environment.

In terms of BIM authoring tools Open BIM projects can still be implemented by using closed and proprietary software solutions. For model creation tools such as Autodesk Revit, Nemetschek Allplan, etc. are used to create the BIM model, which is stored primarily in the native data format. From the native file, IFCs are exported, which can be used for interoperability and collaboration. In such procedures, two major disadvantages can be identified: (i) Typically, the IFC exporters are 'black box', thus there is only limited options to control the mapping of native objects to IFC elements, geometry representation, properties, etc. During data exchange, this can lead to extra efforts for semantics engineering to extract the information. (ii) During the lifetime of the asset, it is likely that the native data will be partially lost, as it is reported from attempts to convert older native files to recent versions.

Thus, the only viable option for a long-term accessibility and benefit of BIMs are open standards. This does not only apply to the use of open standards. For further improvement of the open-source principle Open BIM can be extended by the development and usage of open-source BIM authoring tools. These tools allow users to access, modify, and distribute the software's source code freely, which can significantly reduce software licensing costs for individuals, SMEs, or organizations with limited budgets. This fosters a community-driven approach to BIM, encouraging collaboration among diverse stakeholders and facilitating continuous improvement. The open-source nature of these tools promotes interoperability, allowing users to exchange data seamlessly with other BIM software. This not only supports collaboration but also helps mitigate issues related to proprietary formats and closed ecosystems. For example, variations in the quality of exported IFC data can be observed, potentially resulting in information losses. Open-source BIM authoring tools contribute to a more inclusive and adaptable approach to digital construction, empowering users to tailor software to their specific needs and driving innovation across the industry. In terms of transparency and security, open-source software's code is openly accessible and reviewable, facilitating the identification and resolution of security vulnerabilities. The openness of the code not only fosters user trust in the software, allowing them to understand its inner workings, but also provides potential for education and training. With free access for learning and research, educational institutions and students can easily get into understanding and developing both old and new solutions, resulting in the next generation of architects, engineers, and construction professionals. Open-source BIM tools play a crucial role in facilitating the development of sustainable and green building design and analysis. They allow for the integration of environmental modelling tools to assess the environmental impact of construction projects. While open-source BIM tools offer numerous benefits, they may also come with limitations, such as the potential requirement for more technical expertise to set up and customize. Additionally, some users may still prefer some advanced features, a more user-friendly environment and support offered by proprietary BIM software. The choice between open-source and proprietary tools should carefully consider the specific needs, resources, and goals of the users or organizations involved in construction and design projects.

1.2. Available open-source BIM authoring tools

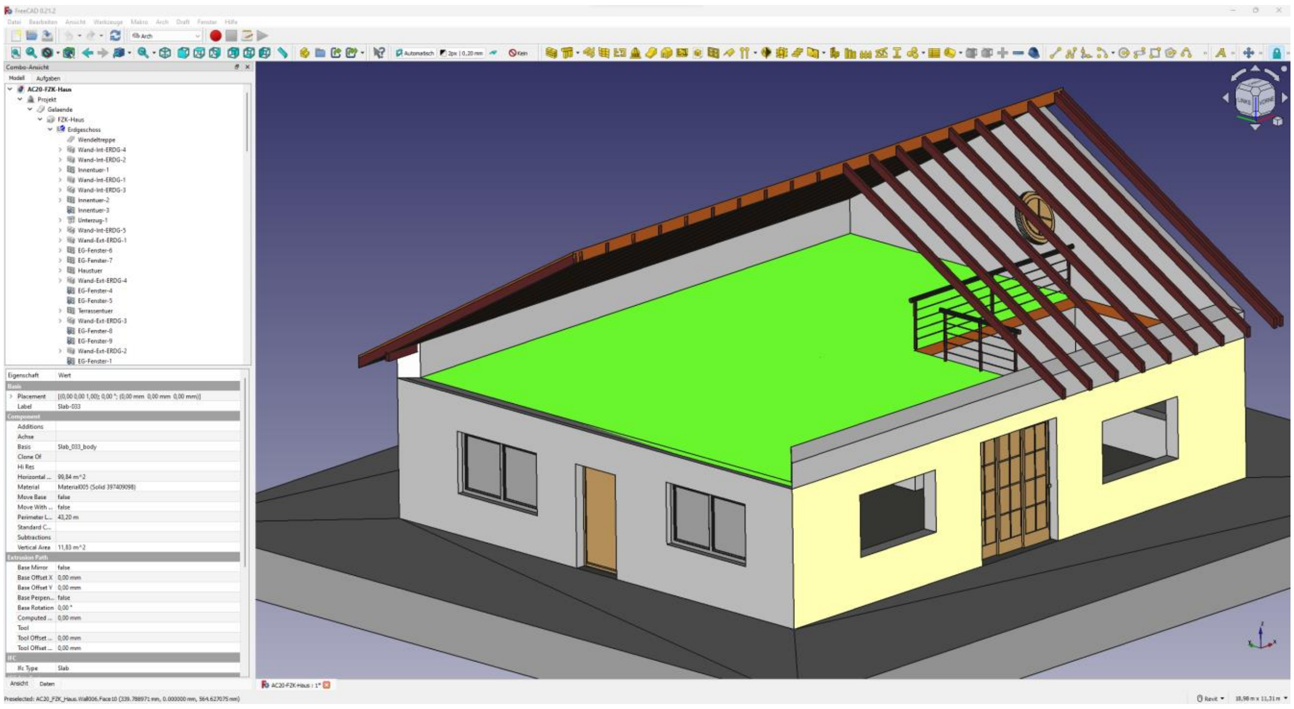
There are several open-source BIM authoring tools available that can be used for architectural and construction projects. Some of the widely used solutions are listed below. Each tool has its own features, strengths, and limitations, which are described in detail.

1.1.1 FreeCAD

Figure 1 FreeCAD Logo

FreeCAD is an open-source, customizable 3D modeler primarily designed for product design, mechanical engineering, and architecture. It is suitable for a wide range of users, including hobbyists, programmers, experienced Computer Aided Design (CAD) users, students, and teachers. FreeCAD is a parametric modeler, which means it allows users to design real-life objects of any size where the shape of the 3D objects is controlled by parameters. This parametric modelling feature enables users to easily modify their designs by going back into the model history to change its parameters. For instance, the shape of a brick might be controlled by three parameters: height, width, and length. These parameters are part of the object and remain modifiable at any time, even after the object has been created. The software is equipped with a variety of tools to meet different needs. It includes dedicated BIM, modern Finite Element Analysis (FEA) tools, experimental Computational Fluid Dynamics (CFD), Geodata or Computer-Aided Manufacturing (CAM)/Computer Numerical Control (CNC) workbenches, and a robot simulation module that allows users to study robot movements. FreeCAD also allows users to sketch geometry-constrained 2D shapes and use them as a base to build other objects. It contains many components to adjust dimensions or extract design details from 3D models to create high-quality, production-ready drawings. FreeCAD is a cross-platform software that runs on Windows, macOS, and Linux operating systems. It is based on OpenCascade, a powerful geometry kernel, and features an Open Inventor-compliant 3D scene representation model provided by the Coin 3D library. It also has a broad Python API, and its interface is built with Qt. FreeCAD supports several different file formats for import and export of models. This enables compatibility with various CAD and BIM software and includes popular file formats such as e.g. IFC; STEP, IGES, Collada DWG, and DXF.

FreeCAD, like many modern design applications such as Revit or other commercial authoring tools, is built upon the idea of Workbench. A workbench can be considered as a set of tools specially grouped for a certain task. When users switch from one workbench to another, the tools available on the interface change. Toolbars, command bars and possibly other parts of the interface switch to the new workbench, but the contents of user's scene don't change.



Eigenschaft	Wert
Basis	
Placement	[(0,00 0,00 1,00); 0,00 °; (0,00 mm 0,00 mm 0,00 mm)]
Label	Slab-033
Component	
Additions	
Achse	
Basis	Slab_033_body
Clone Of	
Hi Res	
Horizontal ...	99,84 m^2
Material	Material005 (Solid 397409098)
Move Base	false
Move With ...	false
Perimeter L...	43,20 m
Standard C...	
Subtractions	
Vertical Area	11,83 m^2
Extrusion Path	
Base Mirror	false
Base Offset X	0,00 mm
Base Offset Y	0,00 mm
Base Perpen...	false
Base Rotation	0,00 °
Computed ...	0,00 mm
Tool	
Tool Offset ...	0,00 mm
Tool Offset ...	0,00 mm
IFC	
Ifc Type	Slab

Preselected: AC20_FZK_Haus.Wall006.Face10 (339.788971 mm, 0.000000 mm, 564.627075 mm)

Figure 2 Overiew FreeCAD GUI with IFC in the Architecture Workbench

Although, FreeCAD is a powerful open-source parametric 3D CAD modelling software, there are limitations. These limitations may vary depending on the version used and the context in which it is used. While efforts have been made to improve the user interface over the years, some users may still find it less intuitive compared to commercial CAD software like AutoCAD or SolidWorks. The software stability can vary depending on the complexity of projects and the system they are running on. Crashes and instability issues can occur, especially when dealing with large and complex assemblies. Further limitations of other work areas are not explained in this context.

In summary, FreeCAD is a versatile and powerful 3D modelling software that offers a wide range of features and tools for various engineering and design needs. Its open-source nature and parametric modelling capabilities make it a flexible solution for professionals, hobbyists, and researchers. The full documentation of FreeCAD can be found under[1].

[1] https://wiki.freecad.org/Power_users_hub

1.1.2 IfcOpenShell

Figure 3 IfcOpenShell Logo

IfcOpenShell is an open-source software library that provides tools for working with IFC files. It uses Open Cascade internally to convert the implicit geometry in IFC files into explicit geometry that any CAD software or modelling package can understand. It enables the reading, writing, and editing of IFC files and supports various IFC versions including IFC2X3, IFC4, IFC4.3. Additionally, it can read and write IFC-SPF, IFCJSON, IFCXML, and IFCHDF5.

IfcOpenShell is primarily a collection of C++ libraries, however, as it has Python bindings, it can be integrated with programs like FreeCAD and Blender. Also, a variety of libraries and applications are included, for instance, it includes a Python library for IFC manipulation, a clash detection library, a library and CLI app to export and import schedules from IFC, and a tool to compare changes between IFC models. It also includes a command-line tool, IfcGeomServer, which allows processing geometry in a crash-safe manner using a child process with dynamic linking. The Python API also provides extensive support for 4D and 5D BIM operations such as cost formulas, critical path analysis, and calendar-based scheduling propagation, as well as many other features of IFC that are typically not supported by other tools. Additionally distinctive are its HDF5 cache support, voxel analysis for handling erroneous BIM geometry, and semantic SVG-based drawing production.

A huge benefit is the high level native IFC authoring API for hundreds of tasks. Complex authoring like copying objects, cost calculation or 4D simulation can be performed with single lines of code. From authoring new projects with geometries and additional properties to editing existing IFC files in a few lines of code, IfcOpenShell offers a wide bandwidth of possibilities. The main limitation in terms of usage in this project lies in the missing geometry creation engine. Hence, IfcOpenShell has no direct GUI to enable real 3D geometry modelling, it uses the Open Cascade internally for geometry processing. The full documentation of IfcOpenShell can be found under[2].

[2] https://blenderbim.org/docs-python/

1.1.3 Blender BIM

Figure 4 BlenderBIM Logo

Blender is an open-source 3D content creation suite that offers a comprehensive set of capabilities, including 3D modelling, rendering, animation, video editing, VFX, compositing, texturing, and numerous simulation types. The cross-platform compatibility, customizable Python scripts, and active community support enhance its appeal as a versatile tool for 3D architecture and efficient creation workflows. An extension of this functionality is demonstrated through the BlenderBIM add-on, a tool designed to facilitate OpenBIM reading, writing, and analysis. Rooted in the IfcOpenShell project, this add-on enables interaction with the IFC file format directly.

BlenderBIM showcases several noteworthy features. By offering a platform to analyse, create, and modify OpenBIM via Blender, the add-on affords multiple settings within the Properties Panel for project and object-specific configurations. This includes Scene Properties for overarching project settings and Object Properties for detailed settings for specific building components such as walls or doors (Figure 5). Additional settings can also be found in the mesh data, materials, and the N-Panel.

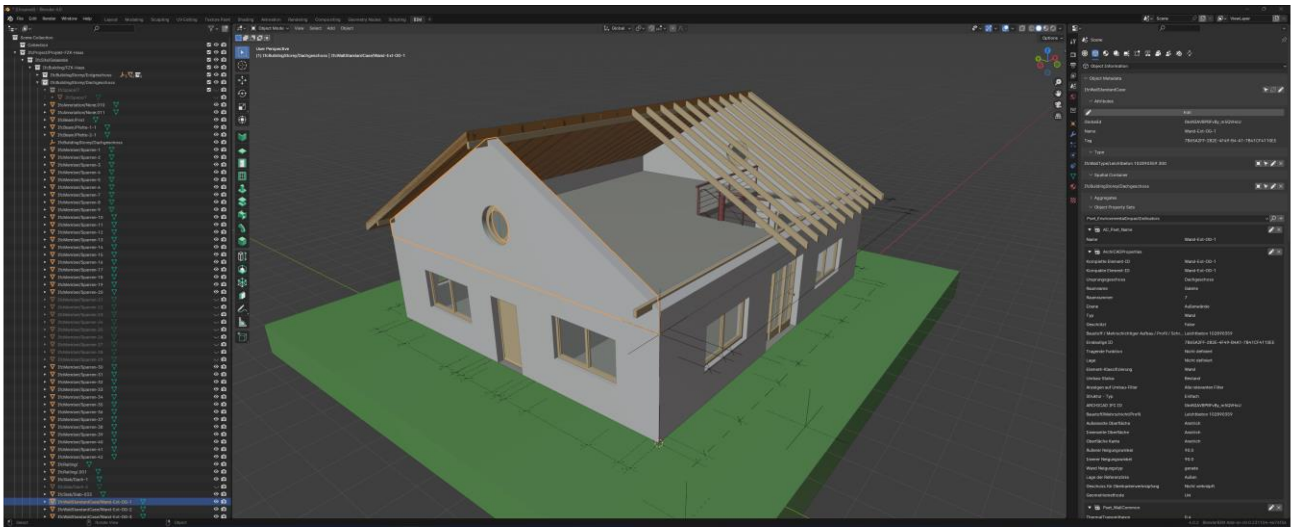


Figure 5 Overview of the BlenderBIM GUI with the IFC Entities (left), 3D Model (middle), Entity property sets (rigt)

This add-on proves particularly advantageous when creating intricate and data-rich OpenBIM. While it permits the addition of new attributes to objects from Blender, an occasional limitation is that the IFC models exported from Blender may not always retain these new properties. It supports the import of .ifc, .ifczip, and .ifcxml formats, and the export of .ifc, .ifczip, and .ifcjson formats, enabling a wide range of interoperability with other BIM tools. The add-on enables the viewing of models, including their spatial relationships and physical attributes. This includes the ability to edit and extract characteristics and attributes, as well as modify geometry and move objects within the model. BlenderBIM also provides tools for managing classification systems, document, and library references, which are essential for organizing and referencing information in a BIM project. It can generate 2D drawings, schedules, and create sheets, facilitating the production of detailed documentation.

Also, it enables the creation of new objects using library elements, providing a way to reuse components and standardize elements across a project. Within BlenderBIM its possible to use the advanced 3D modelling capabilities of Blender to assign IFC Entities to the 3D mesh geometries (Figure 6). It also supports the investigation and editing of structural analysis models, which is crucial for understanding and optimizing the structural performance of a building. BlenderBIM supports the creation of construction schedules, critical path analysis, and the generation of sequence animations, which are essential for planning and managing the construction process. The add-on includes features for creating cost schedules, using formulas, and deriving quantities from model elements, which are important for cost estimation and budgeting. It includes tools for clash detection and managing issues for model coordination, which are crucial for identifying and resolving conflicts in a BIM model. BlenderBIM also includes features for checking IFC data against an Information Delivery Manual with MicroMVDs, which is important for ensuring that a BIM model meets the requirements of a project. Finally, BlenderBIM provides an interface to manage IFC data, including assigning IFC classes, assigning attributes to IFC elements, assigning properties and properties sets to IFC elements, assigning quantities and quantities sets to IFC elements, and calculating quantities of IFC geometry. These features provide a comprehensive set of tools for managing and manipulating IFC data within a BIM model.

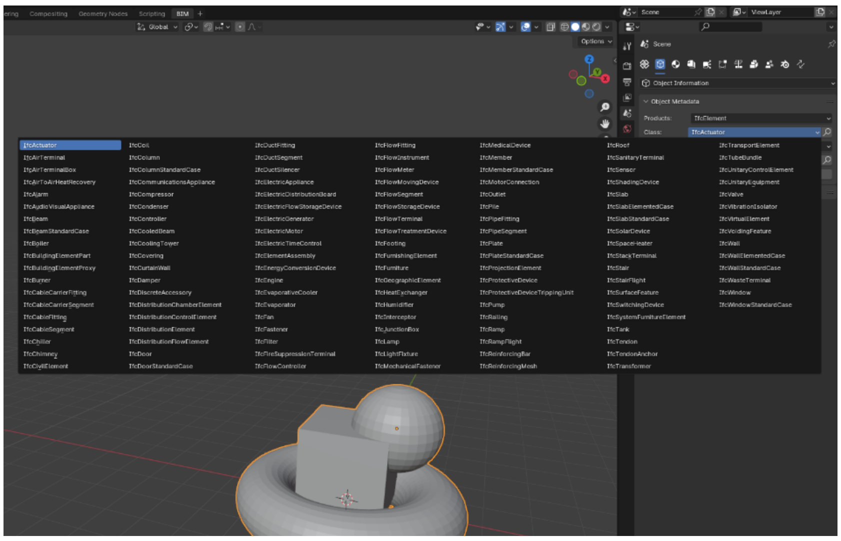

Products: IfcElement
Class: IfcActuator

Figure 6 BlenderBIM tool to generate an IFC entity out of every possible geometry

In conclusion, Blender is a robust tool for 3D content creation, with the BlenderBIM add-on broadening its scope to encompass OpenBIM operations. This positions it as a valuable asset for professionals within the building and construction industry. Nevertheless, as is the case with any intricate tool, its full utilization demands a degree of practice and learning. The full documentation of BlenderBIM can be found here[3].

[3] https://blenderbim.org/docs/index.html

2 The Human Tech approach

After a thorough assessment of the available open-source BIM authoring solutions, it was decided to rely on IfcOpenShell for programmatic creation, update, and exploitation of IFC data, and BlenderBIM as the GUI application. As both are from the same eco-system, there is a tight integration, and procedures developed using the GUI in BlenderBIM can be transferred to IfcOpenShell and vice versa. Both can be used via pyhton interfaces, which in turn ideally suits other solutions based on python e.g., Machine Learning pipelines, point cloud manipulation, etc. In particular, stability problems of FreeCAD when using more extensive BIM models have led to the exclusion of FreeCAD.

Compared to FreeCAD, the modelling capabilities of IfcOpenShell are limited. Nevertheless, IfcOpenShell offers basic geometry creation solutions for walls, doors, etc. and even more advanced ones to define profiles to deliver a detailed representation typically used for BIM modelling of steel structures. However, there is no such flexibility as in FreeCAD which offers a fully parametric modelling environment. If advanced parametric modelling is needed in the further course of the project, Blender and the BlenderBIM add-on with its wide range of modelling capabilities or Open Cascade technologies with respective python interfaces (pyocc) could be used for geometric modelling. IfcOpenShell would then be used to integrate the geometric representations from pyocc into IFC entities. Through the python interface, Blender can also be used for geometry creation.

Besides the IFC model creation, other use cases such as IFC update, object filtering and information extraction apply to the HumanTech solutions related to BIMxD. To deliver the respective solutions, IfcOpenShell offers an exhaustive library as a basis to deliver the solutions. All these use cases and respective implementations are described in section 3.

The above remarks apply to IFC authoring and editing. IFC is the open data standard backbone of the BIMxD implementation in HumanTech. Extending IFC is achieved by formulating new entities and respective properties in a buildingSMART Data Dictionary (bSDD), which in turn can be integrated into the open source BIM authoring tools to deliver instances of the defined entities. The bSDD formulation is part of T2.4, details can be found in the HumanTech bSDD specifications (D2.4).

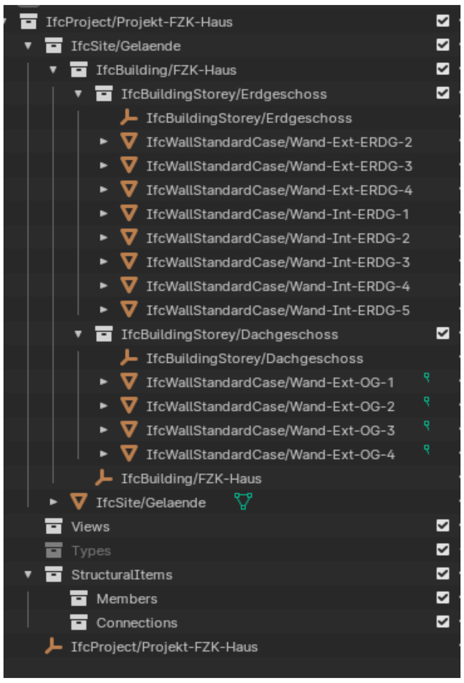
In the following sections, the implementation of open source BIM authoring tools for the HumanTech project will be described. The modules and source code are part of this deliverable and are available at github[4].

[4] https://github.com/humantecheu/openbimxd

3 Open BIMxD software module for BIMxD generation, update, and information delivery

3.1 IFC model creation for scan to BIMxD

3.1.1 File structure



```
IfcProject/Projekt-FZK-Haus
  IfcSite/Gelaende
    IfcBuilding/FZK-Haus
      IfcBuildingStorey/Erdgeschoss
        IfcBuildingStorey/Erdgeschoss
        IfcWallStandardCase/Wand-Ext-ERDG-2
        IfcWallStandardCase/Wand-Ext-ERDG-3
        IfcWallStandardCase/Wand-Ext-ERDG-4
        IfcWallStandardCase/Wand-Int-ERDG-1
        IfcWallStandardCase/Wand-Int-ERDG-2
        IfcWallStandardCase/Wand-Int-ERDG-3
        IfcWallStandardCase/Wand-Int-ERDG-4
        IfcWallStandardCase/Wand-Int-ERDG-5
      IfcBuildingStorey/Dachgeschoss
        IfcBuildingStorey/Dachgeschoss
        IfcWallStandardCase/Wand-Ext-OG-1
        IfcWallStandardCase/Wand-Ext-OG-2
        IfcWallStandardCase/Wand-Ext-OG-3
        IfcWallStandardCase/Wand-Ext-OG-4
      IfcBuilding/FZK-Haus
    IfcSite/Gelaende
  Views
  Types
  StructuralItems
    Members
    Connections
  IfcProject/Projekt-FZK-Haus
```

Figure 7 IFC file structure

The initial step of creating an IFC model is to set up the overall file structure, which is defined by a hierarchy of spatial objects. Typically, it consists of an IfcProject object, one or more IfcSites, IfcBuildings and IfcBuildingStoreys to separate the levels of the building[5], see Figure 7. The IfcProject defines the units, the geometric representation context, the project coordinate system and coordinate space dimension, and the precision used withing the geometric representations. The IfcSite contains the piece of property or more general the area of land the project will be delivered on. On the site, an IfcBuilding defines the structure itself. The building is organiszed using IfcBuildingStoreys, which serves as the spatial container all elements of an elevation level are assigned to.

The IFC model hierarchy itself is defined by IfcRelAggregates objects, which relate either of the objects to its parent elements as shown in Figure 7. The location of IfcSite, IfcBuilding and IfcBuildingStorey is defined by an IfcLocalPlacement. All placements can relate to each other, or be defined against the global coordinate frame defined by the project global positioning concept in the IfcProject entity.

3.1.2 From bounding box to BIM geometry

BIM representations comprise both geometric and semantic information. For the former, the previously reconstructed geometry needs to be transferred into BIM geometric representations. The geometric representation is derived from the geometric reconstruction, mostly bounding boxes, and other basic geometric representations. Bounding boxes need to be transferred to parametric BIM representations.

The purpose of this work is not to discuss the overall structure of the IFC standard. However, it must be noted, that an IFC element, if the standard allows it to encorporate geometry, will be assigned an IfcRepresentation, which is assigned an IfcShapeRepresentation. Among basic geometries such as points or curves, an IfcShapeRepresentation can include parametric solid geometries such as SweptSolid, Boundary representation and Constructive solid geometry. All mentioned concepts have particular

[5] https://standards.buildingsmart.org/IFC/DEV/IFC4_2/FINAL/HTML/

advantages. However, for cuboid-like geometry as reconstructed previously, the IfcSweptSolid is the primary choice. Although this may contain different style of parametric geometry creation, the basic principle is to use a profile and extrude it along a curve. Thus, to create a box as an IfcSweptSolid, a rectangle is used as a profile, which is then extruded along a line with a given dimension.

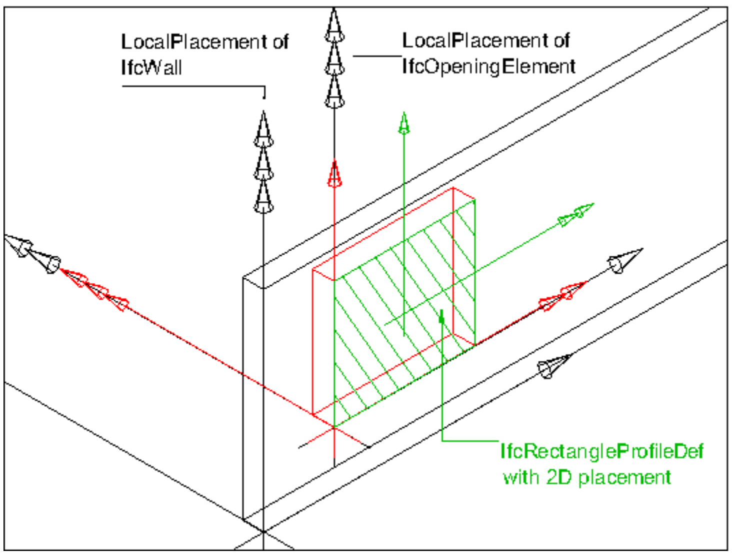

LocalPlacement of IfcWall
LocalPlacement of IfcOpeningElement
IfcRectangleProfileDef with 2D placement

Figure 8 IfcLocalPlacement of IfcWall and IfcOpeningElement

Technically, such parametric IFC geometry could be created using global coordinates i.e., the corner points of the bounding boxes previously reconstructed. However, IFC models follow the concept of local placements. A local placement is a local coordinate frame defined by its axes as vectors and the coordinates of the origin. The concept of local placement applies to the overall structure of the IFC model, including IfcProject, IfcSite, IfcBuilding and finally the building components. According to the standard, all IfcElements (IfcWall, IfcColumn, etc.) should be placed relative to the local placement of its container e.g., IfcBuilding[6]. The same principle applies to child elements such as openings that are associated to a wall, as illustrated in Figure 8.

The described concept requires the global bounding box coordinates to be transformed into a local coordinate frame, or a local coordinate frame relative to the parent element respectively. To achieve this, the bounding box corner points are ordered. More specifically, the points are ordered in counter-clockwise direction from bottom to top. Additionally, two conditions are enforced. (i) the edge connecting the first and second point must be longer than the one connecting the second and third point i.e., the edge of the first and second point is along the length of the object. (ii) the first points X and Y coordinate must be smaller than the second points X and Y coordinate. With this sorting strategy, the IfcLocalPlacement origin can be set to the first bounding box point, the X axis is the norm of the vector connecting the first and second point, the Y axis can be set as the vector connecting the first and fourth point. The Z-axis is set vertical. To ensure that all axes are perpendicular, only the origin and rotation are passed to IfcOpenShells util.placement.rotation() function.

For the IfcLocalPlacement of any opening a copy of the wall local placement that has been transformed to the opening position can be used. As the door has been projected into the wall bounding box, the

[6] https://standards.buildingsmart.org/IFC/DEV/IFC4_2/FINAL/HTML/index.htm

transformation vector is the one of the first points of the wall and door bounding boxes. An example scene with reconstructed walls and doors can be found in Figure 9.

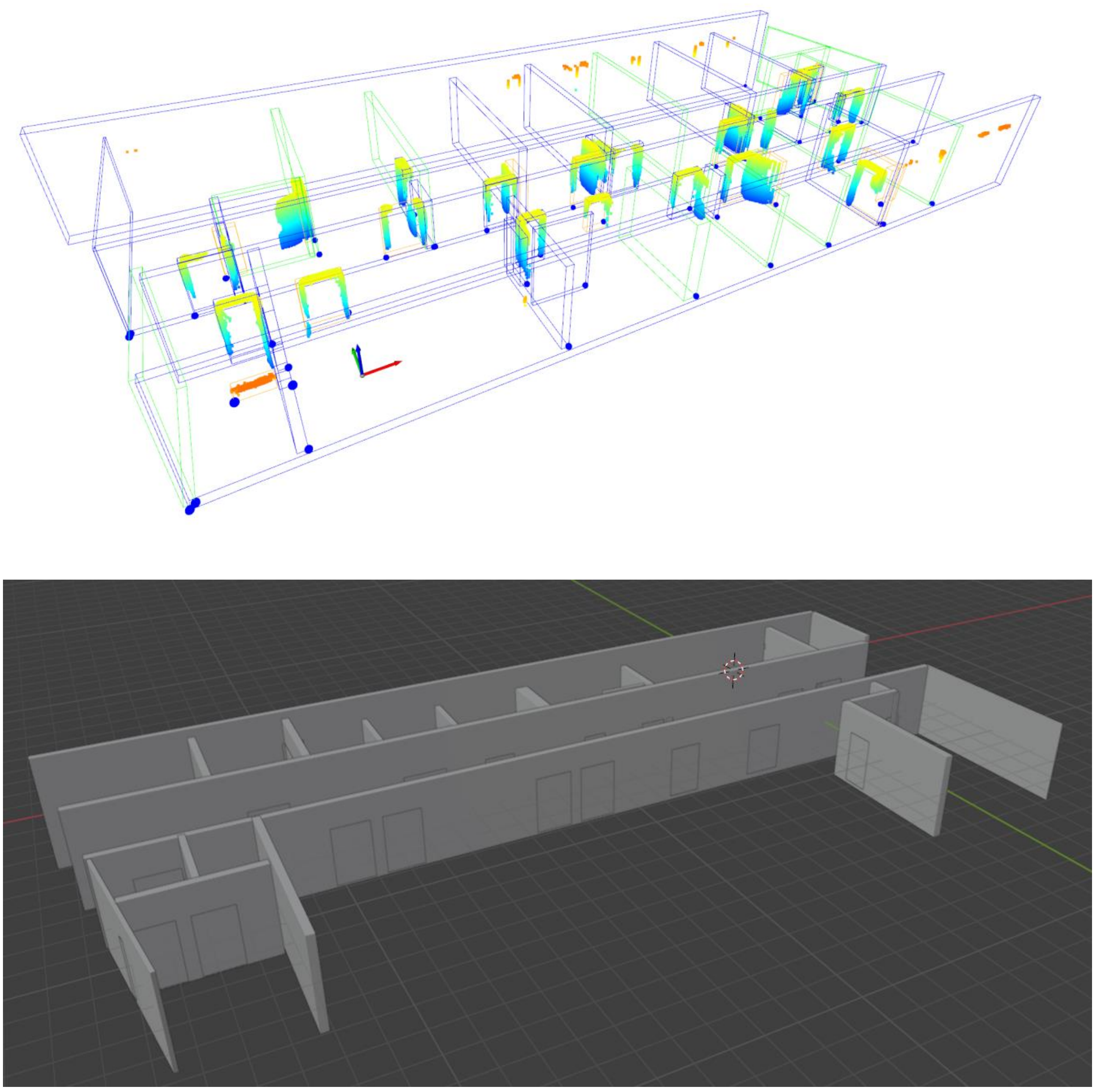


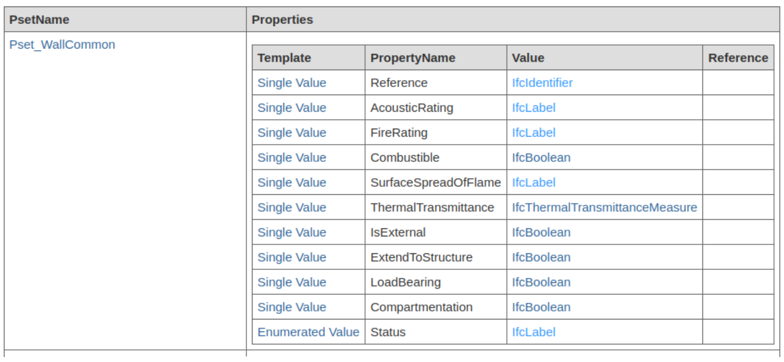
Figure 9 IfcWall and IfcDoors. Top: IFC geometry with door points. Bottom: Resulting IFC file (some objects hidden).

The described concept and procedure can be used for any object with cuboid geometry (walls, slabs, square columns, …) and respective child elements (doors, shafts, openings, …). It is implemented in the openbimxd elements module.

3.1.3 Semantic IFC object creation

Based on the geometric representation and local placement created previously, IFC objects are created for the respective classes. Obviously, IFC entities are not limited to a class and geometry. Thus, further concepts are used to assign more semantic information to the created objects.

In IFC, types are used to share common properties, material (layer) information and common shape representations among multiple instances of a type. For instance, this applies to interior dry walls with a common layer configuration, which can be represented as a material layer set. Same applies for any other object with types such as doors, windows, etc.



PsetName	Properties			
	Template	PropertyName	Value	Reference
Pset_WallCommon	Single Value	Reference	IfcIdentifier	
	Single Value	AcousticRating	IfcLabel	
	Single Value	FireRating	IfcLabel	
	Single Value	Combustible	IfcBoolean	
	Single Value	SurfaceSpreadOfFlame	IfcLabel	
	Single Value	ThermalTransmittance	IfcThermalTransmittanceMeasure	
	Single Value	IsExternal	IfcBoolean	
	Single Value	ExtendToStructure	IfcBoolean	
	Single Value	LoadBearing	IfcBoolean	
	Single Value	Compartmentation	IfcBoolean	
	Enumerated Value	Status	IfcLabel	

Figure 10 Property set wall common

To organize properties, IFC offers predefined property sets that apply for common occurrences of objects. Such property sets contain the name of the property and define its IFC data type. As an example, the properties contained Pset_WallCommon are presented in Figure 10. This property set can be used to add common properties to walls, that apply to most of the occurrences. Beyond that, more specific property sets e.g., for prefabricated concrete walls, are available. In openbimxd, the Pset_WallCommon is added to all walls created. However, the properties are only populated based on the information available from the previous reconstruction, segmentation, and classification methods.

For objects consisting of one homogenous material, an IfcMaterial object can be created which then is assigned to the object using IfcRelAssociateMaterials objects. Following this principle, one instance of an IfcMaterial can be linked to multiple objects, thus avoiding redundant storage of material information. For objects with multiple layers an IfcMaterialLayerSet can be used which combines IfcMaterialLayers to represent multi-layer constructions such as drywalls, timber frame construction, plastered brick walls, etc. Ideally, IfcMaterialLayerSets are used in conjunction with IfcWallTypes to ensure consistent modelling. Note, that the width of the geometric representation must comply with the thickness of all layers of the IfcMaterialLayerSet.

In the current implementation, objects reconstructed from scans will on be assigned an IfcMaterial based on secondary information (plans, documentation, ...) as semantic reconstruction methods to identify layers and materials of objects have not been integrated into the scan to BIMxD pipeline (T3.6).

3.2 Object filtering and information extraction for subtasks

For specific use cases, only a part or even only single objects and their respective information in a BIM model are needed. For robotic navigation, all objects defining the topology of the building such as walls, doors, windows, etc. are of interest. For demolition task planning (T5.1), the object with the respective opening is needed to derive the opening geometry for demolition execution planning. To inherit semantic labels from BIM objects towards point clouds (see section 3.3), all objects of a specific class need to be filtered. Similarly, all child objects of a parent (all windows and doors within a wall) can be relevant. Thus, filtering by class and / or name and properties, as well as selecting by semantic relationships and spatial filtering can be defined as general use cases. All use cases will be illustrated using a model of a storey of an office building, which is used in WP4 for camera localization testing as well. This model is shown in Figure 11.

3.2.1 Filter by IFC class

A very common case is filtering by IFC class. As a result, all elements of a specific IFC class can be obtained. This does not only include objects that have a shape representation such as walls, doors, columns, etc. but also more abstract elements such as IfcStorey being spatial structures and any other elements. This filtering is the starting point for any class-based use cases such as Ifc to label (section 3.3). In IfcOpenShell, any Ifc model can be searched for objects of a specific class using the by_type() method. Same applies for BlenderBIM as the GUI application.

An example of IfcWall elements filtered from the example IFC model is presented in Figure 11.

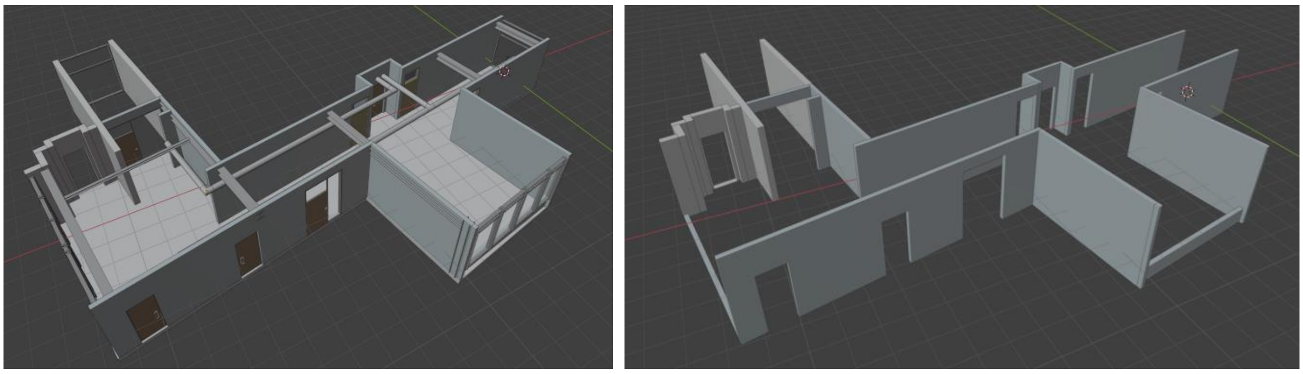


Figure 11 Filtered IfcWall Elements

Note, that in the example above, openings of doors and windows are still present. In other cases, the geometry of openings is lost which can lead to faulty results of subsequent processes. This is caused by various types of geometric representations of openings. In the example of Figure 11, the geometry of the walls is created excluding the doors i.e., the door openings are part of the wall geometric representation. An IfcPolygonalFaceSet has been used in this case, which allows to represent the multiple faces of the wall geometry including the openings. In the second example (Figure 12[7]), after the filtering the openings are not present in the walls anymore. This is the expected behaviour for IfcSweptSolid shape representations.

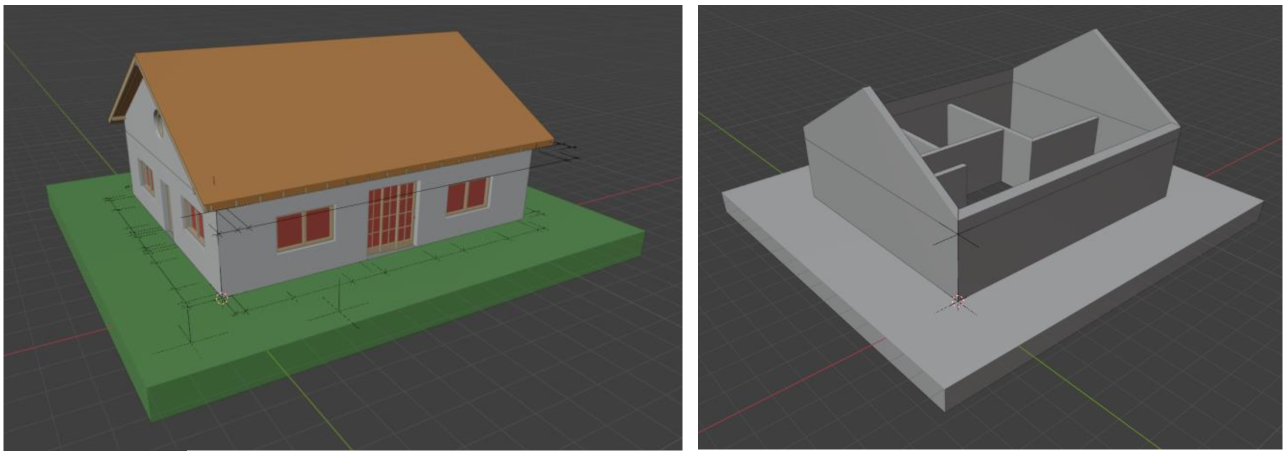


Figure 12 Filtering of IfcWall elements. Left: original IFC model. Right: IfcWall objects without openings.

Whenever openings are required for subsequent processes more advanced filtering strategies must be applied to preserve the openings in the walls. Such filtering methods considering spatial or semantic relationships will be presented in the latter.

Regardless of this specific behaviour, filtering by IFC classes is a very versatile and simple filtering strategy to explore and extract information from IFC files.

3.2.2 Filter by Name or properties

Obviously, filtering by class can result in a multitude of search results with the need of further distinction. As IFC entities are rich of information, many features can be used to find specific elements. Among others, these are the material, type, location, and any property within property sets. To identify specific elements, the GlobalID can be used. This is useful for frequent BIMxD update of one specific element.

To implement advanced filters, IfcOpenShell offers a selector syntax that allows for a combination of different search parameters, including regular expression style search patterns. The most common search queries are:

- IfcSlab, IfcWall, …: filter elements one or more IFC classes
- IfcSlab, material=concrete filter all slabs with material concrete
- IfcDoor, Name=D02 filter all doors with the given name

[7] Institute for Automation and Applied Informatics (IAI) / Karlsruhe Institute of Technology (KIT)

Further options for searching can be found in the IfcOpenShell documentation[8]. An example of the filter string *IfcWall, ! 3rPX_Juz59peXXY6wDJl18* applied to an IFC model is given in Figure 13. This results in all walls, except the one with the given ID.

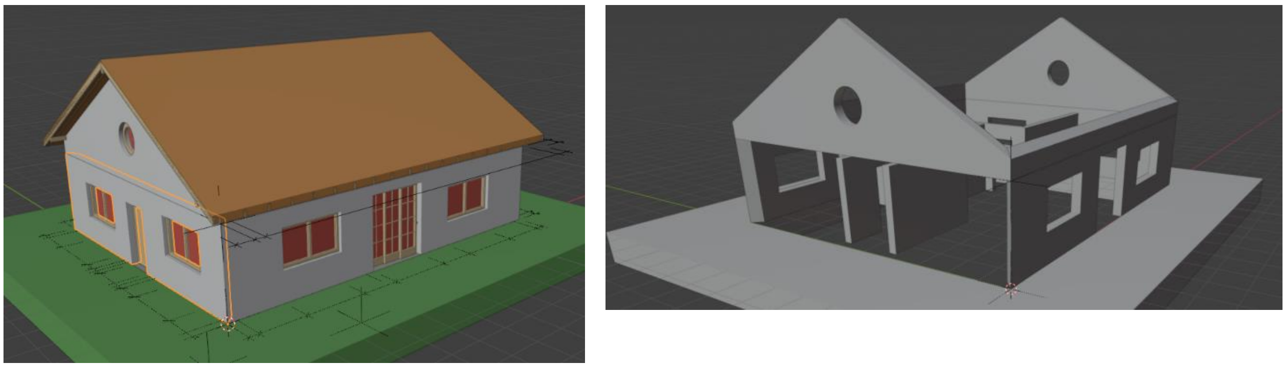


Figure 13 Filtering of IfcWall, except one wall identified by a given GUID. Left: original IFC model, right filtered walls

3.2.3 Filter by relationships

The previously described filtering methods rely on the match of specific strings in the IFC file, and the retrieval of all assigned entities (IfcLocalPlacement, shape representations, …). However, in many cases it is beneficial to exploit the relationships of objects. These can be twofold: (i) the spatial or semantic composition of an object i.e., subelements of an element collection such as an IfcElementAssembly or elements related to a parent element such as openings. (ii) spatial proximity of objects e.g. objects enclosed in a certain space defined by a bounding box. The spatial and semantic decomposition can be retrieved by using the ifcopenshell.util.element.get_decomposition()[9] method. From an element as an input, this method retrieves all related sub-elements for further processing. This method is used primarily to retrieve openings inside parent objects as shown in Figure 14. In this implementation, the get_decomposition() method is used to retrieve the relationships assigned to specific objects and thus all objects related to the parent element. However, for larger spatial containers such as storeys the inverse operation ifcopenshell.util.element.get_container()[10] is used to find the spatial container of all objects and reassign them after filtering. Beyond the relationship-based query, IfcOpenShell offers a method to build an unbalanced geometry tree of a model.[11] Such a tree can be used to run queries based on a point, a sphere, elements, and rays. When querying an element all elements contained in, containing, and intersecting and the element itself are returned. This method is particularly useful to find elements to be updated by reconstructed geometry.

[8] https://blenderbim.org/docs-python/ifcopenshell-python/selector_syntax.html
[9] https://blenderbim.org/docs-python/autoapi/ifcopenshell/util/element/index.html#ifcopenshell.util.element.get_decomposition
[10] https://blenderbim.org/docs-python/autoapi/ifcopenshell/util/element/index.html#ifcopenshell.util.element.get_container
[11] https://blenderbim.org/docs-python/ifcopenshell-python/geometry_tree.html

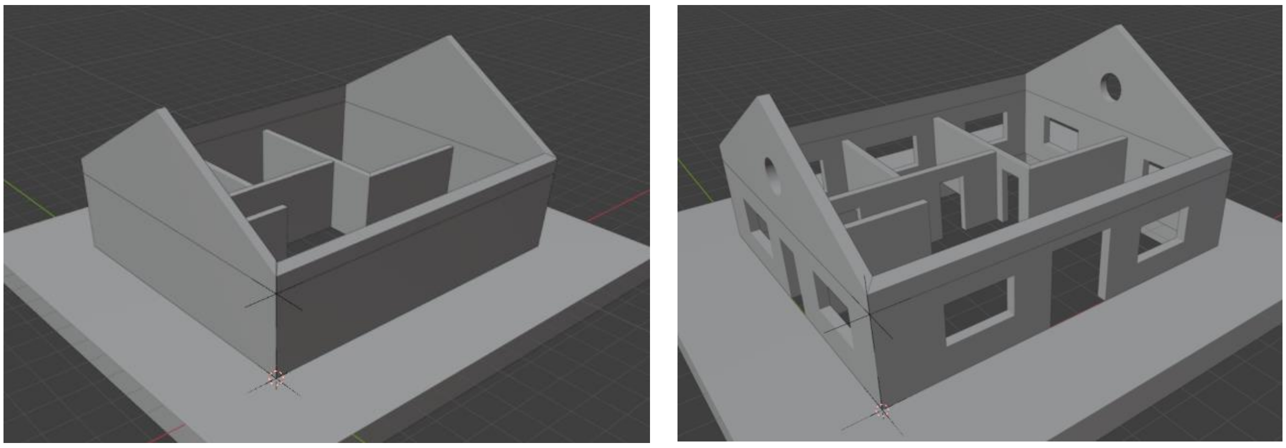


Figure 14 Retrieving openings of wall parent elements. Left: walls without openings, Right: walls with openings

3.2.4 Export and use of filtered IFC files

All filtered objects are added to a new file preserving their properties, material, GUID, etc. This reliably avoids any artefact objects present in the file as only the true positive i.e., the filtered elements and related objects are present in the file. The overall structure of the file (see 3.1.1) is preserved as in the original file. The result of the filtering methods described above can be twofold. Either geometric, semantic information or quantities are directly derived from the objects or a new IFC file is created containing all the filtered objects.

An example of geometry derived from the IFC objects is described in section 3.2. Generally, the shape representation, which contains the geometry is extracted e.g., to derive a bounding box of the object. In the HumanTech ecosystem, bounding boxes are a common representation e.g., to check for points inside the geometry, or in the further course to identify changes of objects for respective updates.

In the latter case, the structure of the initial IFC file will be transferred to the newly created filtered IFC file. To retrieve a fully functional model, all assigned relationships need to be transferred to the filtered model. Among others, this involves the assignment of the objects to spatial containers (typically storeys), and the assignment of parent-child relationships (walls to doors and windows). Advanced semantic filtering methods from the IfcOpenShell are used to either identify and assign the spatial container of an object, or to assign parent-child relationships.

3.3 Ifc to label

In this project, some tasks such as T3.4, T3.5, T3.6, T4.3, and the respective pilot implementations in WP7 rely on ground truth labels in point clouds i.e., a semantic label per point representing the class of an object. In many cases, these are added manually to the data. If a sufficiently accurate BIM model is available, it can be used to retrieve the ground truth labels.

The implementation is closely related to the pystruct3d module which is currently being developed and will be part of D3.6. In this library, basic geometry processing algorithm for 3D point cloud geometry extraction are available. Among others, the implementation of the bounding box class and respective methods such as to find all points within the bounding box are used here to assign the labels based on the IFC objects.

The method is implemented as follows. First, all vertices are retrieved from the geometry representations of the IFC objects. Effectively, the vertices are all corner points of the object's geometry, including all openings and features. From these vertices, the eight extremal points can be used to obtain the bounding box of the object. The extremal points are these with the maximum distance from the centroid of all vertices. With these eight points a bounding box is initiated which is used to retrieve all points inside the bounding box. To all points inside a specific bounding box the respective class is assigned.

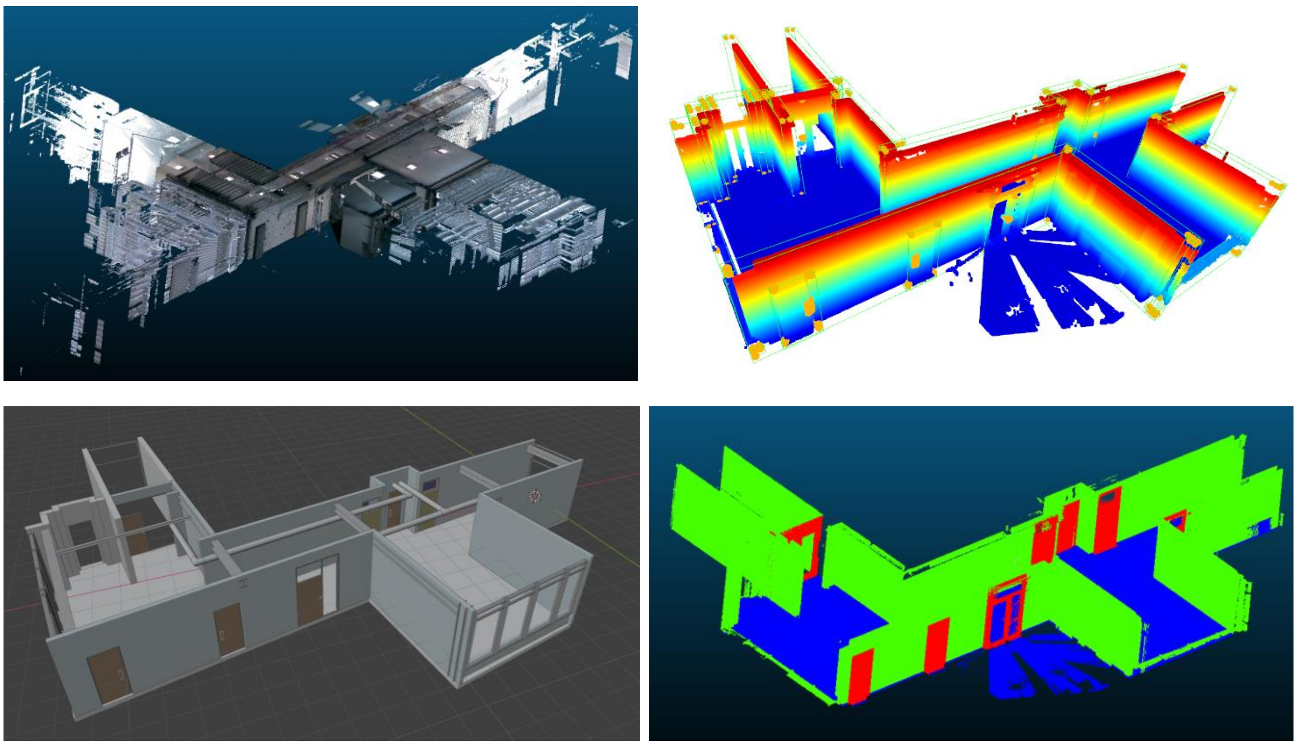


Figure 15 IFC to label. Top left: original point cloud, Top right: bounding boxes and point cloud, Bottom left: BIM model, Bottom right: retrieved labels.

3.4 BIMxD update

Updating IFC models is not a very common application for this type of data format which has mainly been developed to exchange models and use these to coordinate the design process or to reference design domains against a typically architectural model. With common BIM tools updating is a difficult task, as it is normally achieved in the native file format followed by an export of the updated model. A direct update of the IFC model itself could not be achieved, as no tools were available. Based on the IfcOpenShell a python module to instantly update the geometry, attributes and properties of IFC objects is presented.

In HumanTech an update of IFC objects is useful for different use cases:

- Adding / updating building objects based on scanned data.
- Updating the state of dynamic moving objects such as robots (WP5), or workers localized using wearable cameras (WP4)
- Updating generic objects representing information such as falling hazards, occupied zones, etc.

An example of updating the location and properties of the mobile robotic platform used in HumanTech is displayed in Figure 16. Note, that the geometric representation of the robot is a simplified mesh derived from the CAD model used for the assembly of the robot. Although simplified, it is still quite detailed and requires a considerable amount of storage. Depending on the use case an even more simplified geometric representation might be more appropriate.

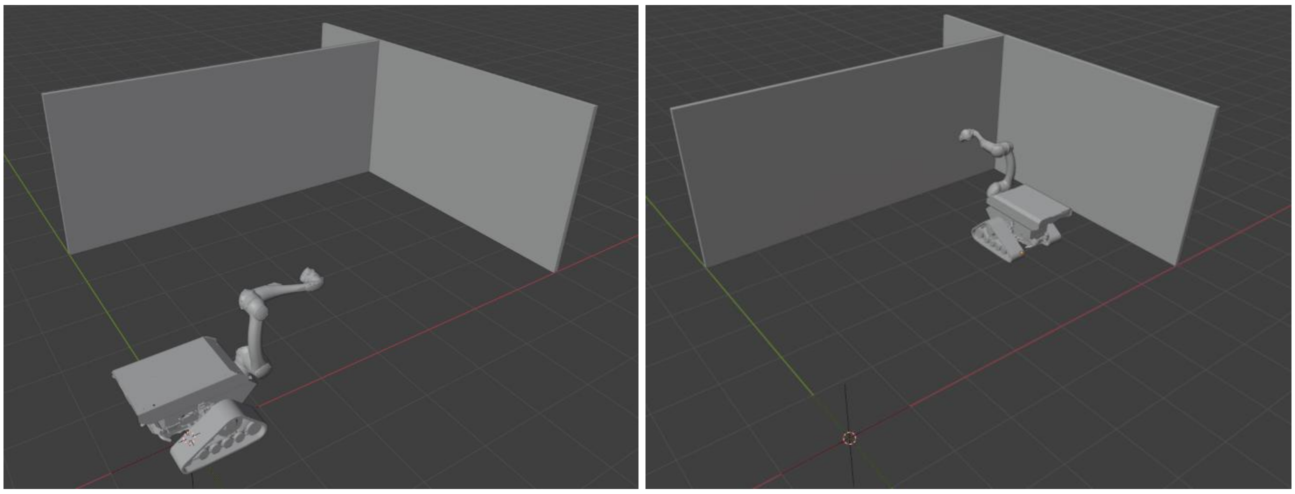


Figure 16 Update of robot location and properties. Left: original, Right: After update.

For the robot a custom property set PSet_Robot has been set up, which has four properties: is_active (Boolean), battery (ratio 0…1), current_task (text) and is_moving (Boolean). During the update, two of the valueas are updated. The battery level is set to 0.82 from 0.89 and is_moving is set to False. The new properties are passed to the update_property() method as a python dictionary, with full flexibility and integration into any python environment and module.

The update is implemented as follows. The UpdateElement class is initialized with an IFC model and one of the containing elements. The update_location() method updates the IfcLocalPlacement of the object with a new origin point and angle. If only a vector is available, the new IfcLocalPlacement origin can be derived by adding the vector to the original origin coordinates. On the semantics, the update_material() method changes the material with a given IfcMaterial. The IfcMaterials need to be created in the IfcMaterial module. The update_property() method reads the existing property set(s), exchanges properties of an existing property set or adds a new property set using a python dictionary. Finally, the updated file is saved. Note, that the GUID is preserved when updating the model which is crucial for the application of associated technologies such as BCF.

Semantic information is not only included in property sets, however this is the most common option and will be primarily used in HumanTech. Property sets facilitate a very structured way to organize properties, especially to ensure that the same properties are assigned for each instance of a class or type of objects. With the same procedures as described above, other properties and attributes can be edited as well.

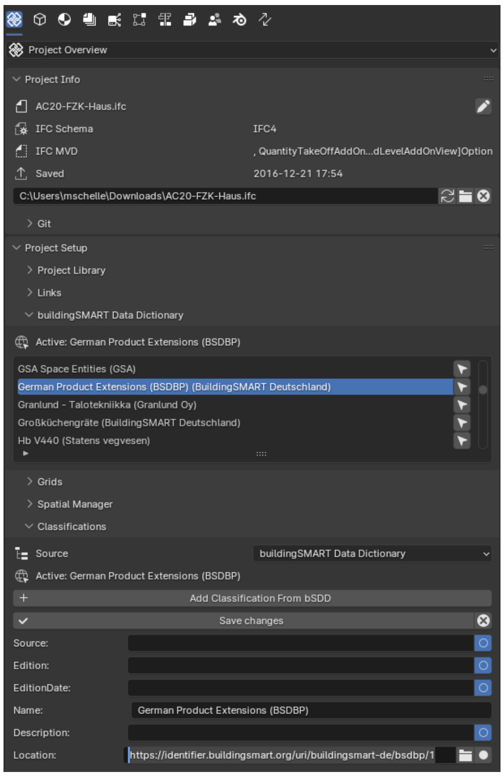
Updating the location of an element is a very basic geometry-related update procedure. However, more advanced operations might be needed in the context of updating a model based on geometry reconstructed from scans of the asset. In these cases, one approach is to exchange the geometric representation with one generated as described in section 3.1.2. Indeed, this is the most practical option for many update cases. However, some constraints derived from the old geometry might have to be considered such as height or connections to neighbouring objects.

4 bSDD: extending IFC towards an enhanced standard

The buildingSMART Data Dictionary (bSDD) is an online service hosting classes (terms) and properties, allowed values, units, translations, relations between those and more. It provides a standardised workflow to guarantee data quality, information consistency and interoperability.

Besides national and international classification systems (e.g. Uniclass, CCI) and domain-specific standards (e.g. ETIM, IfcAirport), company-specific standards can be stored in bSDD as well. The bSDD implements the ideas from ISO 12006-3, ISO 23386 and Linked Data standards.[12] In terms of the HumanTech project bSDD specifications for BIMxD are identified in "T2.4 bSDD specifications and IFC extension", respective in "D2.4 bSDD specification from BIMxD".

At the heart of bSDD is a database with all dictionaries. The content of dictionaries can be related to each other, creating a connected graph. The main way to access the bSDD is through its APIs. This is how most BIM software and other apps can use the data stored in the bSDD. Apart from that, there is the bSDD Search page, where people can look up the content. Authors can publish content to bSDD through the API or the Management Portal. BIM modellers use the bSDD for easy and efficient access to all kinds of standards to enrich their models. BIM Managers use the bSDD to reference Information Delivery Specifications (IDS) and check BIM data for validity. Content creators benefit from having one entry point to various BIM tools and platforms. Within the HumanTech approach a HumanTech bSDD can be accessed and used via the BlenderBIM add-on. Therefore, the specific bSDD source has to be searched and chosen in the project info under "buildingSMART Data Dictionary" (Figure 17). In this case the "German Product Extension (BSDBP) (BuildingSMART Deutschland)" was used for example. After choosing the bSDD the data dictionary has to be added to the project under "Add Classification From bSDD". There, further information regarding the dictionary can be found, e.g. the name or the URL of the bSDD location.



AC20-FZK-Haus.ifc
IFC Schema IFC4
IFC MVD , QuantityTakeOffAddOn...dLevelAddOnView]Option
Saved 2016-12-21 17:54
C:\Users\mschelle\Downloads\AC20-FZK-Haus.ifc

Active: German Product Extensions (BSDBP)

GSA Space Entities (GSA)
German Product Extensions (BSDBP) (BuildingSMART Deutschland)
Granlund - Talotekniikka (Granlund Oy)
Großküchengräte (BuildingSMART Deutschland)
Hb V440 (Statens vegvesen)

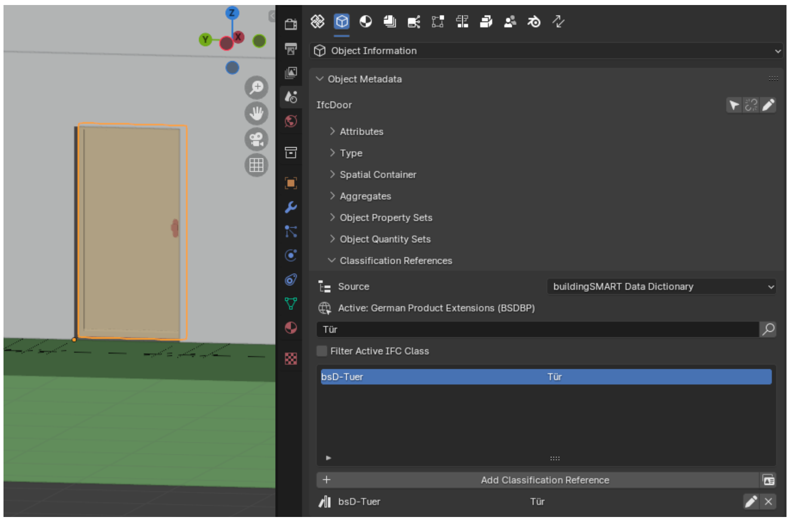
Source buildingSMART Data Dictionary
Active: German Product Extensions (BSDBP)

Source:
Edition:
EditionDate:
Name: German Product Extensions (BSDBP)
Description:
Location: https://identifier.buildingsmart.org/uri/buildingsmart-de/bsdbp/1

Figure 17 initializing bSDD in BlenderBIM

[12] https://www.buildingsmart.org/users/services/buildingsmart-data-dictionary/

After adding the classification source, the bSDD can be used to add specific "Classification References" to the IFC Objects in the model. In the shown case of Figure 18, the IfcDoor was given an additional classification reference "bsD-Tuer", defined in the mentioned bSDD "German Product Extension (BSDBP) (BuildingSMART Deutschland)". This classification reference can be seen as an additional, predefined property set of the IFC Object and could contain domain specific information, used in a variety of use cases. In the HumanTech approach e.g. possible predefined property sets for specific objects could be robot related information that later can be found in the planned BIM model and used by construction robots on site. The current state of the project the HumanTech bSDD is still under continuous development and will be extended and specified throughout the whole project.



Source buildingSMART Data Dictionary
Active: German Product Extensions (BSDBP)
Tür
bsD-Tuer Tür
bsD-Tuer Tür

Figure 18 Adding a classification reference from bSDD to an IFC Object

5 BIMxD backend integration

The BIMxD backend offers the platform to store and exchange information based on the IFC standard. The overall setup of the BIMxD platform and backend is described in D2.3. One of the main advantages is the collaborative environment the BIMxD platform offers, with functionalities to access and visualize the models as well as to assign issues based on BCF, etc. Related to openbimxd, two relations can be identified: (i) the BIMxD backend offers a REST API[13], which is documented and accessible for the entire HumanTech consortium. (ii) Services can be integrated into the backend, which is useful for such services that are used frequently by multiple users.

The Catenda Rest API offers methods to authenticate, manage projects and project members, add models and revisions, labels, libraries, etc. The most relevant methods are the ones related to IFC models that allow the user to download or create and update models. This applies for all the described procedures in openbimxd as all could use IFC files form the Backend and update the models on the backend respectively. However, offline processing can cause extra effort as all users need to set up a python environment for openbimxd and need to organize the file handling through API calls.

Thus, some of the procedures described will be integrated into the backend. At the beginning of the second period of the project these activities will be initiated and will terminate in the pilot implementations of WP7.

[13] https://hub.catenda.com/developers/reference/api/v2#catenda-rest-api-v2-stable

6 Link to other HumanTech tasks

As already mentioned in the previous subsections, openbimxd delivered by T2.2 is strongly linked to other HumanTech tasks.

Within WP2 this applies to T2.1 BIMxD formats and specifications, T2.3 BIMxD data exchange, interoperability and platform and T2.4 bSDD specifications and IFC extension. Openbimxd will be integrated into the BIMxD backend (T2.3) as described later in section 5. The bSDD specifications can be used to define IFC entities not available in the standard as well as respective property sets. The use and relation are described in section 4.

In WP3, T3.1 UGV&UAV scanning pipelines use information extracted from IFC for mission planning. T3.4 Synthetic data generation and domain adaptation uses IFC models to generate synthetic data. Although primarily implemented in Unreal, the tools provided here nevertheless support relating procedures. In T3.5 ground truth labels can be derived from IFC models as described in section 3.3. Finally, T3.6 uses openbimxd to create IFC objects using semantics and geometry extracted from scans as described in section 3.1. The geometry processing algorithms are implemented in the pystruct3d module (T3.6), both are strongly linked and use the same implementation of bounding boxes and the respective methods.

The use of openbimxd in WP4 is twofold. In T4.3 information can be derived from IFC models using openbimxd. However, the more relevant use case is the frequent update of localized objects such as humans using wearable devices or robots from attached sensors respectively. In both cases, openbimxd provides the functionality to update the location and properties as described in 3.4.

In WP5 mainly information is extracted from IFCs for related tasks such as demolition planning (T5.1). openbimxd provides the tools necessary to retrieve the information for advanced perception and robotic navigation (T5.4).

Generally, openbimxd can be applied in WP6 for training purposes. In this context, sample files have been created to provide simple-to-understand yet practically examples for training and teaching.

In WP7 openbimxd will serve as a toolbox to access the BIMxD backend and extract information for the respective use cases. Please refer to the initial architecture deliverable (D1.2) for details.

7 Conclusions

In this Deliverable, open-source tools are presented to create BIMxD objects from reconstructed semantics and geometry, object filtering and information extraction, inheritance of semantic labels from IFC objects and BIMxD update are presented.

These tools for open-source BIM authoring and the implementation in the openbimxd module provides the backbone for interacting with BIMxD representations in the HumanTech project. With the submission of the delivery, the code will be accessible on github https://github.com/humantecheu/openbimxd with a complete documentation and application examples. Naturally, the source code and documentation are part of this deliverable.

Indeed, it does not only serve the HumanTech project, but outlines a pathway towards an open standard construction industry fully relying on open-source tools. Open source does not only give the user control over the implementation and tools they are using, open source and open standards return the data back to the user, since the data is no longer trapped in data formats only accessible using proprietary tools. Considering the lifetime of assets of the build environment of 50 up to more than 100 years, long-term data access is crucial. This is granted by open data formats. Since the industry is currently adopting BIM as a method and technology, it is a major concern of the HumanTech project to foster the adoption of open BIM.