A lightweight BIM–GIS Integration alternative via DIN 18290-1 Multi-Model Container - a prototype implementation

Submitted by:
Iveli de Paula Sousa

June 2025

Technische Universität Berlin
Fakultät VI - Planen Bauen Umwelt
Institut für Bauingenieurwesen
Fachgebiet Bauinformatik

Introduction

This research explores how to bring together Building Information Modeling (BIM) and Geographic Information Systems (GIS) into a single, standards-compliant container. BIM models carry rich semantic and geometric detail about individual building elements, while GIS datasets have a broader spatial context and analysis capabilities. In practice, however, these two domains have historically remained separate, even though their integration could improve decision-making from early design through facility management. To address this gap, the project develops a simple prototype that packages an IFC-based BIM model and a GeoJSON GIS layer into a Multi-Model Container (MMC) archive, conforming to the DIN 18290-1 standard. The resulting MMC not only preserves each model's native format but also embeds metadata to express their spatial relationship.

The main objectives are to ensure that (1) the prototype can reliably extract georeferencing parameters from the IFC file (or default, if they are missing), (2) the GeoJSON's coordinate reference system (CRS) is captured automatically and (3) both models are correctly listed in the XML descriptors according to MMC schema. By doing so, the research examines how lightweight containers can, without requiring complete data conversions, bridge the semantic depth of BIM and GIS.

State of the art

Bringing together Building Information Modeling (BIM) and Geographic Information Systems (GIS) has become an essential concern for AEC professionals seeking to combine detailed asset descriptions with broader spatial context. Because these two domains address complementary needs, their integration promises more informed decision-making throughout a project's lifecycle, which might range from early design and site planning to maintenance and urban policymaking. Yet, as numerous authors have documented, the very richness of BIM and GIS models creates semantic and technical mismatches that complicate seamless interoperability (Sani & Rahman, 2018; Guyo, Hartmann, & Ungureanu, 2021).

BIM/GIS Data Standards and Semantic Alignment

BIM and GIS rely on open but distinct standards that serve different needs. In the BIM community, the Industry Foundation Classes (IFC) schema (ISO 16739-1) is widely used to exchange rich, object-level semantic information and parametric geometry. IFC can represent walls, doors, beams and other building elements with detailed attributes such as materials, properties and construction schedules. GIS practitioners, on the other hand, more often use standards like CityGML or GeoJSON to emphasize geographic features, topological relationships and spatial analysis. Because these standards were designed for different purposes, attempts to convert an BIM model directly into GIS dataset (or vice versa) tend to degrade geometry and discard semantic details. For example, curved surfaces in BIM model may be approximated by flat polygons in a GIS layer, while GIS dataset's broad land-use categories rarely match the finely detailed object classes in IFC. As a result, direct one-to-one conversion often fails to preserve the richness of either data model (Noardo et al., 2020).

To avoid information loss, some researchers nowadays advocate for approaches that keep each format in its native form while linking them explicitly. Krischler et al (2024) demonstrated this by extending the Information Container for Linked Document Delivery (ICDD) to bundle IFC and GeoJSON files alongside an RDF manifest. In their workflow, each IFC and GeoJSON retains its full detail, and relationships are stored as triples that describe how features intersect or contain one another. Similarly, Krischler et al. (2025) extended the Multi-Model Container (MMC) schema so that multiple geometric, topological and mereological links can be expressed in XML. This strategy preserves the component-level depth of IFC and the spatial simplicity of GeoJSON without forcing a single unified schema.

Semantic barriers also appear from mismatches in class definitions and granularity. In IFC, "IfcWindow" is a specific class with properties that describe glazing, framing and manufacturer. In GIS files, a window may simply be part of a "Building" feature, lacking dedicated semantics. CityGML's "Bridge" category, for example, does not map directly to any IFC class unless extension schemas are used. Existing automated mapping tools often have to rely on manually created dictionaries (Guyo, Hartmann and Ungureanu, 2021; Sani and Rahman, 2018) or machine learning models to resolve these mismatches (Diakite & Zlatanova, 2020). Even when class names align, differences in attribute hierarchies, units of measure and geometry representations are an issue for the integration

Integration techniques

To integrate have grouped integration strategies into three main categories: 1) Data-standard conversion, 2) Data-standard integration, and 3) Linked-data federation."

Guyo, Hartmann and Ungureanu (2021) mentioned three main categories for connecting BIM and GIS data: 1) data-standard conversion, 2) data-standard integration and 3) linked-data federation.

1. Data-Standard Conversion

Conversion involves transforming one format into another, for instance exporting IFC to CityGML or importing CityGML terrain into IFC. Conversion tools can be useful for one-off analyses, such as feeding city models into an energy simulation or terrain analysis. However, this process often degrades semantics or geometry. Curved surfaces in IFC become flattened, and parametric wall details may be lost. The GeoBIM Benchmark 2019 showed that many tools struggle to preserve high-fidelity geometry or complex parametric constructs. Bidirectional conversion (IFC⟶CityGML⟶IFC) is even less reliable because each conversion step introduces the possibility of mismatch and data loss (Noardo et al., 2020; Guyo, Hartmann and Ungureanu, 2021).

2. Data-Standard Integration

Integration tries to reconcile multiple data models into a single, unified schema. This can look like a database that stores both BIM and GIS features or an extended schema that subsumes elements from IFC and CityGML. The greatest advantage of this kind of integration is that it allows more complex queries, such as "identify roads within two meters of a building wall," but it sacrifices domain specificity. Maintaining such a unified schema is costly since any change to IFC or CityGML standards requires updating the integration model. In practice, enterprise data architectures that adopt integration layers often either lose parametric richness or fall out of sync with standard updates. (Guyo, Hartmann and Ungureanu, 2021)

3. Linked-Data for interoperabiliy

This method keeps each file in its native format—IFC remains IFC, GeoJSON remains GeoJSON—and stores only cross-references in an external graph (typically RDF or OWL). For example, an IfcWall instance might receive a unique URI and be linked to a CityGML:Road polygon through triples that express spatial relations such as "intersects" or "within." Because IFC and GeoJSON each retain their full geometry and semantics, there is no need to flatten or convert. Researchers can issue SPARQL queries or use ontology alignment services to traverse both domains. Early work in this direction (Krischler et al. 2023) showed how the Information Container for Linked Document Delivery (ICDD) can bundle IFC and GeoJSON files with a minimal RDF manifest that records spatial relationships (e.g., "IfcSite intersects GeoJSON:Parcel123"). Later, Krischler et al. (2025) applied the same principle to the Multi-Model Container (MMC) standard, embedding Geospatial_Link, Geometric_Link, and Topological_Link entries as XML so users can unzip an MMC archive and still explore both BIM and GIS files in familiar software. The original semantics and parametric definitions remain intact, and no expensive monolithic schema needs updating whenever a new version of BIM/GIS standards is released.

Because conversion and unified-schema approaches degrade essential details, and because linked-data federation allows each domain to retain its native richness, our work adopts the linked-data paradigm, specifically an MMC-based workflow. By using a standard container (MMC 2.1.0) to hold IFC and GeoJSON together with cross-reference links, we preserve provenance, minimize information loss, and allow integration at both file-level and component-level granularity.

Another topic worth being mentioned, is that among each one of those approaches, georeferencing is a critical prerequisite for all integration methods. Christian and Görne (2019) proposed six "Levels of Georeferencing" to describe how much spatial information an IFC file contains. At the lowest level, an IFC might only include a postal address with no precise coordinates. At mid-levels, the model includes a projected coordinate system and basic IfcMapConversion parameters, allowing accurate placement in a projected space. At the highest level, control points tie the IFC to a geodetic network with centimeter-level accuracy. The GeoBIM Benchmark found that most commercial IFC models only reached mid-level georeferencing (Noardo et al., 2020). When spatial anchors are missing or inconsistent, even the best semantic links cannot be exploited. To overcome incomplete georeferencing, Diakité and Zlatanova (2020) proposed an automated heuristic that extracts 2D building footprints from IFC, compares them to GIS footprints from sources such as OpenStreetMap, and computes a best-fit translation and rotation. This would recover spatial alignment when the IFC lacks explicit coordinate system references.

Current gaps and possible future directions

When BIM and GIS data are effectively linked, a range of practical applications becomes possible. In infrastructure planning, Krischler et al. (2023) used ICDD bundles to combine IFC models of bridges, cadastral parcels and noise maps into a single interface for rapid scenario analysis. In construction progress monitoring, linked point-cloud data can be compared directly against BIM geometry while also referencing site topography stored in GIS. At the urban scale, combined datasets feed energy modeling, line-of-sight analysis and evacuation simulations by leveraging BIM's semantic detail alongside GIS's spatial analytics (Noardo et al., 2020).

Despite these advances, three significant gaps remain. First, automated link discovery remains largely manual. Connecting building components with GIS features often requires human intervention. While some experiments in machine learning and shape-matching have started to show potential correspondences, these methods are still experimental and lack generality. Second, scalability and data governance at city scale have not been fully tested. When linking thousands of IFC files and millions of GIS features, it might be add version control, access management and performance, which are the challenges that most academic and industry prototypes have yet to tackle. Third, tool support for linked-data workflows remains limited outside research contexts. Mainstream BIM and GIS platforms rarely offer built-in RDF endpoints or SPARQL interfaces, so organizations often depend on custom scripts or academic code that is not maintained over time.

Methods

This project aimed to develop a prototype tool that packages a Building Information Model (BIM) file in IFC format together with a GIS dataset in GeoJSON format into a standards-compliant Multi-Model Container (MMC) according to DIN 18290-1. The prototype produces a single `.mmc` archive that, when opened, contain a MultiModel.xml descriptor, a LinkModel.xml descriptor, and a models/ folder with the original IFC and GeoJSON files. The following sections describe the data sources and software environment, the overall processing workflow to create this prototype, how the metadata was extracted from the `.ifc` and `.geoJSON` files and, finally, how the XML descriptors and the final archive are created.

Data and Software Environment

The prototype expects two properly geo-referenced inputs placed in a predetermined folder structure: the IFC file (in this case, the `Ifc4_SampleHouse.ifc`) placed in the `input/bim` and a GeoJSON file (for this example, `landuse_residential_berlin.geojson`) placed in the `input\gis` folder. Both files should already contain valid coordinate reference information so that any spatial relationships can later be linked. Development was done using Python 3.11, with the following key libraries and modules:

- **ifcopenshell**, for parsing IFC files and extracting schema information and its georeferencing entities.
- **geopandas** for loading GeoJSON files, accessing their coordinate reference system (CRS) and geometry collections;
- **xml.etree.ElementTree** for creating the XML descriptors.
- Python's built-in **shutil** and **zipfile** modules to manage file copying and to produce the final `.mmc` archive;
- A configuration module (`config.py`) for the user to edit the file paths, constants, and schema settings.

MMC Generation

Once the BIM and GIS metadata have been parsed, the next task is to assemble them into a single, cohesive container that conforms exactly to the DIN 18290-1 standard. In practice, this means producing two XML descriptors—MultiModel.xml (listing model metadata) and LinkModel.xml (defining spatial relationships)—and packaging them alongside the original IFC and GeoJSON files. By the end of this process, the user has a single `.mmc` archive that any standard ZIP tool or compliant validation platform can inspect. The whole MMC folder generation happens as follows:

1. **Preparation of input files**: preprocessed BIM and GIS files need to be placed into the `models/` folder.

2. **Editing parameters**: the user needs to edit configuration parameters (input and output paths) in `mmc_builder/config.py`. By default, directory structure is created under `output/mmc/`, with a dedicated `models/` subfolder to host the IFC and GeoJSON files.

3. **Metadata parsing**: metadata from both input files are extracted to be later included in the XML files. For the IFC file, the parser identifies the schema version if available and extracts geolocation and transformation data from `IfcSite` and `IfcMapConversion` entities. From the GeoJSON file, it identifies its Coordinate Reference System (CRS), which is added to the XML. The script also captures the EPSG code and counts the feature types—information that can be used in the future to expand linking logic or validation.

4. **Creation of the XML files**: both `MultiModel.xml` and `LinkModel.xml` are created as structured XML trees with identifiers to maintain consistency. The `MultiModel.xml` document lists the BIM and GIS models, including format, version, and file paths. The `LinkModel.xml` defines spatial relationships, which in this case is a `<Geospatial_Link>`. For the initial test, placeholder values were used for position and orientation, though real values from the IFC would be correctly inserted if available.
5. **Assemble the MMC archive**: finally, Python's `zipfile` module packages the two XML descriptor files and the entire `models/` folder (complete with the original `.ifc` and `.geojson` files) into a `.mmc` archive. Because MMC is defined as a ZIP-based container, users can open or extract it with any standard unzip tool. The final file thus conforms to DIN 18290-1 while remaining easy to inspect or validate.

Metadata Extraction and Descriptor Population

During the metadata extraction step, the IFC parser focuses on two primary tasks. First, it calls `ifcopenshell.ifc_file.schema` to determine the IFC schema version, by storing that string for the `<Format>` or `<Version>` element in `MultiModel.xml`. Second, the parser searches for `IfcSite` entities to read geodetic (WGS84) or projected (CRS) parameters; if an `IfcMapConversion` object is present, its seven parameters (scale X, scale Y, scale Z, Easting, Northing, OrthogonalHeight, and rotation) are captured as a dictionary. These values are saved as attributes in a Python dictionary so that, when constructing LinkModel.xml, the placeholder `<Position>` and `<Orientation>` tags can be updated with real numbers.

For the GeoJSON file, geopandas provides immediate access to `.crs` (the coordinate reference system) and `.geometry.geom_type` (the geometry types present, such as "Polygon" or "Point"). The script extracts the EPSG code (e.g., `EPSG:4326`) and the total feature count. These metadata are inserted into the `<Format>` (set to "GeoJSON"), `<Version>` (assumed "1.0" unless otherwise specified), and a `<CoordinateSystem>` tag in `MultiModel.xml`. Although this prototype does not yet exploit geometry counts for linking, storing such metadata lays the groundwork for future development of semantic or topological links between BIM elements and GIS features.

All XML trees are serialized with UTF-8 encoding and human-readable indentation. The `MultiModel.xml` begins with a root element `<MultiModelContainer>`, under which each `<Model>` child holds subelements `<ID>`, `<Format>`, `<Version>`, and `<Path>`. Similarly, `LinkModel.xml` has `<Links>` as its root, containing each with subelements `<SourceModelID>`, `<TargetModelID>`, `<Position>`, and `<Orientation>`. Even though the prototype currently inserts generic defaults for `<Position>` (e.g., "0,0,0") and `<Orientation>` (e.g., "0,0,0,1"), replacing those default tags with the IFC's `IfcMapConversion` values requires only a minor adjustment to the parser's output dictionary.

Results

The prototype generated a valid Multi-Model Container archive during the tests performed. Each resulting `.mmc` file opened without error in a standard ZIP utility, with two XML descriptors (MultiModel.xml and LinkModel.xml) alongside the `models/` folder with the original IFC and GeoJSON inputs. Examining these descriptors confirmed that the BIM and GIS entries appeared as expected. Users are able to substitute their own IFC and GeoJSON files simply by placing them in the designated `input/bim/` and `input/gis/` directories and updating the file paths in config.py. For those familiar with basic scripting, this process is very straightforward.

The prototype's behavior in some cases is still not ideal. In the current implementation, the code will revert to default georeferencing values, setting the `<outerPose>` to (0,0,0) when no map conversion is found and omitting a `<Meta key="CRS">` entry if the GeoJSON's CRS is undefined. However, because no test inputs lacking these attributes have been provided, the script's ability to report missing metadata or to fail in such scenarios was not verified so far. Finally, a survey of publicly available tools and literature revealed no other open-source or commercial implementations that generate DIN 18290-1, compliant with MMC archives for IFC + GeoJSON. In that sense, this prototype appears to be the first documented code for this specific packaging task.

Discussion

At its core, the tool fulfills the primary goal of embedding an IFC and a GeoJSON into a single MMC. By extracting schema information from the IFC and reading georeferencing parameters when available, the prototype ensures that both models are represented accurately by the XML descriptors. From a usability perspective, allowing users to swap input files by editing a single configuration file lowers the barrier for individuals who have minimal scripting experience. Nonetheless, it is important to acknowledge that many professionals in architecture, engineering, and construction field are still resistent to edit even a small piece of code. Adding a simple graphical interface or a drag-and-drop feature for choosing files could be an alternative to bridge this gap and encourage wider adoption.

Currently, a key limitation is how the prototype handles georeferencing. Right now, it looks for either an `IfcMapConversion` or an `IfcSite` entry in the IFC. If neither exists, it defaults to placing everything at (0,0,0), which is unlikely to match the real-world location. Without a warning or check, a user might unknowingly generate an archive that places the BIM and GIS data in completely different places. The same issue arises if a GeoJSON lacks a declared CRS. The resulting MMC simply omits that information and users may not even realize their maps will be misaligned. To avoid these pitfalls, the script should perform basic checks up front, by verifying that the IFC has some form of georeference and that the GeoJSON has a CRS, so it can alert the user before generating a container with errors.

Because no other MMC-generation tools were discovered during the literature review, this prototype creates an initial starting point. There is still a great need for richer linking logic, in which multiple `<Geospatial_Link>` entries could represent finer-grained associations, such as matching individual building footprints to parcel boundaries. To achieve that level of detail, the prototype would need to iterate over IFC elements, extract 2D geometries, and make spatial queries against the GIS data layer. One possible approach could be to compare the X–Z extents (2D geometry) of the IFC footprints to the polygons queried from OpenStreetMap via an Overpass Turbo API. By calculating shape-matching scores, the algorithm could find a best-fit translation and rotation, thereby automatically georeferencing IFCs that lack embedded coordinates. This machine learning/heuristic methodology could be very useful given that many existing IFC models in practice are often not properly georeferenced.

Finally, expanding beyond IFC and GeoJSON would make the prototype even more useful across different fields. Supporting formats like CityGML or KML would require new parsing libraries and updates to the XML descriptors, but it would also encourage AEC professionals as well as city planners to bundle their 3D city and map data into one container. Likewise, integrating the script into an automated workflow, such as a small server that watches a folder for new IFC and GIS pairs, would bring a stronger error handling and a way to process multiple files at once. Each of these additions would move the tool closer to a reliable, production-ready system that helps lower the gap between building models and geographic data.

Conclusion

This research has produced a prototype that packages an IFC-based BIM model and a GeoJSON GIS dataset into a DIN 18290-1-compliant Multi-Model Container. In tests, each `.mmc` archive opened correctly and contained intact XML descriptors alongside the original `models/` folder. The prototype reliably extracts georeference data from IFC when available and captures the GeoJSON's CRS automatically. Substituting new input files requires only placing them in designated folders and updating a single configuration file.

Remaining current limitations include defaults when IFC georeference or GeoJSON CRS entries are missing and a single `<Geospatial_Link>` that cannot capture element-level associations. Addressing these issues will require adding input validation and extending the linking logic to match individual footprints, possibly by shape-matching using querying datasets from OpenStreetMap. Supporting additional formats beyond IFC and GeoJSON and automating the workflow in a server environment would further enhance usability. Overall, this work shows that a lightweight, link-oriented container can bridge BIM's detailed semantics and GIS's spatial context without full model conversion.

References

Christian, C., & Görne, H. (2019). LevelofGeoreferencing (LoGeoRef) Using IFC for BIM. Journal of Geodesy.

Diakité, A., & Zlatanova, S. (2020). Automated georeferencing of IFC surfaces to GIS footprints. International Journal of Geographical Information Science, 34(12), 2416–2434.

Guyo, E., Hartmann, T., & Ungureanu, L. (2021). Interoperability between BIM and GIS through open data standards: An overview of current literature. Proceedings of the EuroSDR Symposium on BIM and GIS Integration, 15–26.

Karan, E., Irizarry, J., & Haymaker, J. (2015). BIM and GIS integration and interoperability based on Semantic Web technology. Journal of Computing in Civil Engineering, 30(4), 04015043. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000519

Krischler, J., Schilling, S., Taraben, J., Sternal, M., Koch, C., & Clemen, C. (2024). A standards-based approach to BIM–GIS integration: Extending the Multi-Model Container schema. Journal of Information Technology in Construction, 29, 95–112.

Krischler, J., Schuler, P.-C., Taraben, J., & Koch, C. (2023). Using ICDD for BIM and GIS integration in infrastructure. Automation in Construction, 145, 104700.

Liu, Y., Nguyen, T., & Zhang, X. (2025). IoT and BIM integration for real-time resource monitoring in industrial facilities. Automation in Construction, 160, 104834.

Mushtaha, H., Elbeltagi, E., & Abdelmegid, M. (2024). Post-disaster BIM–GIS integration using ICDD bundles: A case study for rapid damage assessment. Natural Hazards Review, 25(1), 04023077.

Noardo, F., Harrie, L., Arroyo Ohori, K., Biljecki, F., Ellul, C., Krijnen, T., Eriksson, H., Guler, D., Hintz, D., Jadidi, M. A., Pla, M., Sanchez, S., Soini, V.-P., Stouffs, R., Tekavec, J., & Stoter, J. (2020). Tools for BIM–GIS integration (IFC georeferencing and conversions): Results from the GeoBIM benchmark 2019. ISPRS International Journal of Geo-Information, 9(9), 509. https://doi.org/10.3390/ijgi9090509

Sani, M. J., & Abdul Rahman, A. (2018). GIS and BIM integration at data level: A review. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLII-4/W9, 299–306. https://doi.org/10.5194/isprs-archives-XLII-4-W9-299-2018

Thao, L., Verrelst, J., & Van Meirvenne, M. (2025). Port planning with integrated BIM–GIS: A multi-criteria optimization approach. Journal of Urban Planning and Development, 151(2), 04021023.

Xiong, X., Zhang, X., & Li, L. (2013). Bridging BIM and GIS: A review of data interoperability. International Journal of Digital Earth, 6(5), 397–417.

Zlatanova, S., & Christian, C. (2020). Trends in BIM and GIS standardization: Report from the Joint ISO/TC 59/SC 13–ISO/TC 211 Working Group: GIS-BIM. ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences, V-2, 185–192.