PRACTICAL EXAMPLES ON BIM-GIS INTEGRATION BASED ON SEMANTIC WEB TRIPLESTORES

S. Schilling [1], C. Clemen [1]

[1] HTW Dresden - University of Applied Sciences, Faculty of Spatial Information, Dresden, Germany
(sebastian.schilling, christian.clemen)@htw-dresden.de

Commission V, WG V/7

KEY WORDS: BIM, GIS, Semantic Web, Linked Data, Ontology, Triplestore, Object-relational Database

ABSTRACT:

The integration of geodata and building models is one of the current challenges in the AECOO (architecture, engineering, construction, owner, operation) domain. Data from Building Information Models (BIM) and Geographical Information Systems (GIS) can't be simply mapped 1:1 to each other because of their different domains. One possible approach is to convert all data in a domain-independent format and link them together in a semantic database. To demonstrate, how this data integration can be done in a federated database architecture, we utilize concepts of the semantic web, ontologies and the Resource Description Framework (RDF). It turns out, however, that traditional object-relational approaches provide more efficient access methods on geometrical representations than triplestores. Therefore we developed a hybrid approach with files, geodatabases and triplestores. This work-in-progess-paper (extend abstract) demonstrates our intermediate research results by practical examples and identifies opportunities and limitations of the hybrid approach.

1. INTRODUCTION

In the introduction we want to briefly explain the general background of BIM / GIS integration in order to motivate our specific and application-oriented research question.

1.1 Background

A direct and complete conversion of data between the BIM and GIS domain is not possible, although they may describe the same things, e.g. buildings. The reason is a fundamentally different perspective on things, which causes a lack of interoperability (Beck et al., 2021). While e.g. buildings are often described within the context of a city model in GIS, buildings in BIM are described in context for building construction or operation. The fundamental goal of numerous research projects (Noardo et al., 2020a, Karimi and Iordanova, 2019) is to evaluate and improve the BIM→GIS, GIS→BIM and the common BIM↔GIS communication without data loss.

One possible approach is to the utilization of semantic web technologies. Instead of converting BIM and GIS data to each other, all data is converted into a format of the Resource Description Framework (RDF). RDF is the standard model for data interchange in the semantic web [1]. It is readable for humans and machines, uses URI's (Uniform Resource Identifier) to identify each resource and can be used to link resources from different sources. All relations between resources are structured as triples. Each triple contains a subject, predicate and object. RDF is used to describe web resources and can be stored in files, triplestores or general graph databases.

For each conversion to RDF exists an ontology, a schema which describes the data of a source format in RDF. The main modeling languages for describing RDF data are the Web Ontology Language (OWL) and the Resource Description Framework Schema (RDFS). On this schema level, different ontologies can be linked by mapping classes, which define the same individuals or creating links by adding new predicates and classes. On this basis links can be created on data level between resources.

For the data integration on the persistence level, all converted data may be stored in a single triplestore, then all data, domain-ontologies, link-ontology and previously created links are physically and logically united. The advantage of this approach is, that it is now possible to make queries over all data. E.g. to answer questions, which are needed by both, the ACEOO and geospatial domain. In this approach all information is stored within one triplestore. This approach is very useful for many data integrations, but geospatial data can't be stored and queried as efficient as in object-relational databases. In particular, no or hardly any spatial functions are available. In a triplestore, each geometry is typically stored as WKT (Well Known Text) string and typed with a URI for the coordinate reference system (CRS). Every geometry string needs this URI, even if they have the same CRS because so far there is no standardized CRS ontology. When a geometry is queried by the client, triplestores cannot perform datum transformations or coordinate conversions. They only export the geometry in the original CRS, while an object-relational database can output geometries in almost any desired CRS. However, if you neglect these typical geospatial aspects, triplestores offer very effective queries for linked data.

1.2 Problem statement

That's why we believe a hybrid approach with

- geodatabase and
- triplestore

[1] https://www.w3.org/RDF/

could utilize the advantages of both concepts. In this work-in-progress paper we show, how we've tried to implement and test this approach with a very small dataset.

1.3 Structure of the paper

The paper is structured as follows. In the next section (2) we give a short overview of previous data integration methods with semantic web technologies between BIM and GIS. Then, in Section 3, we will explain how we've implemented our federated system as micro service architecture. After that we will show how to apply the system on a very simple test dataset (Section 4). In the end we will discuss the results and finally give a short conclusion and outlook (Section 5).

2. RELATED WORK

The BIM-GIS integration with a semantic web approach is already discussed for a lot of years by many researches (El-Mekawy and Östman, 2010, Guyo et al., 2021, Hor and Sohn, 2021). A comprehensive overview of the use of semantic web technologies in construction (BIM) is presented by (Pauwels et al., 2017). The research presented by (Roxin and Hbeich, 2019) and (Jetlund, 2021) is more specifically addressing BIM/GIS integration with open standards. They evaluate and demonstrate the benefits of linking data with the semantic web technology stack.

In our presented work we tried to convert all data of BIM and GIS into RDF and import it into a triplestore. With the growing number of conversion tools, this has been an easy task and is not the subject of our research. Some examples, e.g. the GIS converters, are presented in (Ulutaş Karakol et al., 2018). For the building models in Industry Foundation Classes (IFC) we used the IFCtoRDF converter[2]. More tools for BIM-GIS integration are discussed e.g. in (Noardo et al., 2020b).

For the linking between spatial data we used the GeoSPARQL ontology as in (Battle and Kolas, 2011). GeoSPARQL is a standardized geospatial extension of the SPARQL Protocol and RDF Query Language, which enables describing and querying of geometries in RDF[3]. Geometries are expressed as WKT or GML strings, however we used the WKT serialization only.

Not every triplestore is able to store geometry and answer spatial questions. Also, triplestores supports GeoSPARQL in many different ways. (Jovanovik et al., 2021) tested the GeoSPARQL support of some triplestores and pointed out, that the choice of the right triplestore is important for a good geometry support. Their results also show, that there is no triplestore which fully supports GeoSPARQL. We choose a tiplestore with a good GeoSPARQL support for our tests, namely GraphDB[4].

3. TECHNICAL IMPLEMENTATION

After importing all data we tried to answer questions by querying with SPARQL and GeoSPARQL. Because of our different sources from GIS and BIM, we had 2D and 3D geometries in different CRS. However GeoSPARQL only supports 2D geometries, so just the x and y coordinates are used for calculations. Calculations are always made in the CRS of the first geometry in the query.

These limitations show that neither the GeoSPARQL standard nor the triplestores are able to provide functionality comparable to geodatabases at this time. This is the point, where we want to start with our hybrid solution, which will link a geodatabase with a triplestore.

For the implementation we decided to develop a microservice architecture. Microservices allow us to create little lightweight and independent pieces of software, which can be linked together to achieve a common goal. This architectural style has the advantages, that services can be easily added, updated, tested and deleted.

In this section we will explain, how we developed this architecture, how each service works and how the services play together.

3.1 Used Software Tools

The services are mainly developed with the Spring Framework for Java, because it provides many functionalities for microservices[5]. Table 1 lists the principal Spring packages, that were used for our microservices.

main package	sub package
spring-boot	actuator jersey jpa neo4j security test thymeleaf web
spring-cloud	bootstrap config config-server eureka-client eureka-server

Table 1. Main Spring packages used for the microservice architecture.

All services shall be accessible over an API, which is documented with OpenAPI Specification 3.0. OpenAPI 3.0 can serve the access on applications over HTTP requests in a simple, reusable, well documented, machine- and human-readable way.

With the concept of *content negotiation* the response can be outputted in various formats. This could be useful in the future, if we want to deliver geospatial and BIM data in different formats. The services, designed and developed within our research, use API's for the communication in the microservice architecture only.

As semantic storage we use a Java based triplestore with SPARQL and GeoSPARQL support. Geometries are stored in a PostgreSQL database with the PostGIS extension. An advantage of PostGIS is that it can be asked for geometries in a wide range of coordinate systems. If needed, the geometry will be converted before output.

All components of the microservice architecture are wrapped in Docker Container Images. Docker Containers are ideal for

[2] https://github.com/pipauwel/IFCtoRDF
[3] http://schemas.opengis.net/geosparql/1.0/geosparql_vocab_all.rdf
[4] https://www.ontotext.com/products/graphdb/
[5] https://spring.io/microservices

microservice architectures, because they are easy to install, update and delete. Docker Images are immutable and can be used several times, side by side.

These basic components of our system, will be extended by more services as Docker in the future. Docker offers flexibility and scalability to our system.

3.2 Microservice Architecture

Our microservice architecture actually consists of seven main components (Figure 1). Beside the databases of PostgreSQL and the triplestore there are a Config and a Discovery Server. Both are typical for a microservice architecture, because they are used for the administration of all services. The Config Server is used to serve configurations to the services, with parameters related to all or single services.

The Discovery Server manages the communication between services within our architecture. If a service gets started, he first registers to the Discovery Server to get part of the network. If any service wants to communicate with another, the Discovery Server establishes contact.

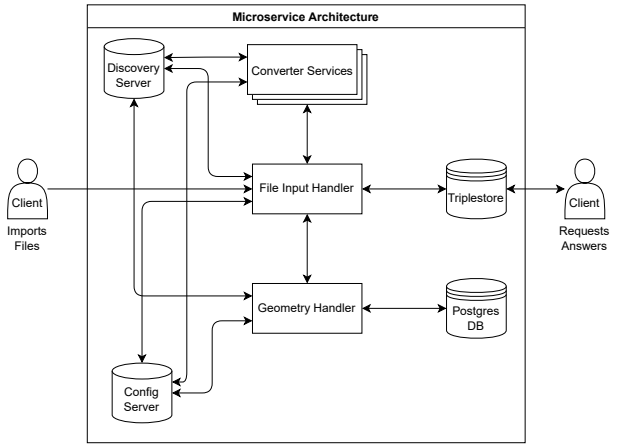

Figure 1. The basic microservice architecture and the connections between components. Clients can interact with the File Input Handler and the Triplestore.

These four services (triplestore, PostgreSQL with PostGIS, Config Server and Discovery Server) are the general backbone of the system, but not specifically related to BIM/GIS integration. Currently, we have implemented three specific services:

- The File Input Handler orchestrates the processing of all incoming files and imports RDF data into the triplestore.
- The Geometry Handler extracts geometry from files and imports it into the PostgreSQL database.
- Converter Services summarizes all converters, which can convert different file types to an RDF file. Examples of this are CSVtoRDF, XMLtoRDF or IFCtoRDF converters.

All Services have an API, over which they can be reached and data can be requested. It should be noted that we use the open source object store minio[6] for file handling. The communication and file transfer only takes place via the http(s) protocol.

[6] https://min.io/

3.3 General Information Flow

As a starting point of a use-case, we consider the user, who wants to upload some files to our system. Currently (January 2022) files of the following types can be uploaded:

- CSV/TXT (may containing a geo-feature as WKT) for Geodata and Annotation (GIS)
- LandXML, especially for Digital Terrain Models (DTM)
- IFC for building models (BIM)
- TTL and RDF/XML for any generic information

Figure 2 shows, how data is proceeded through the microservice architecture.

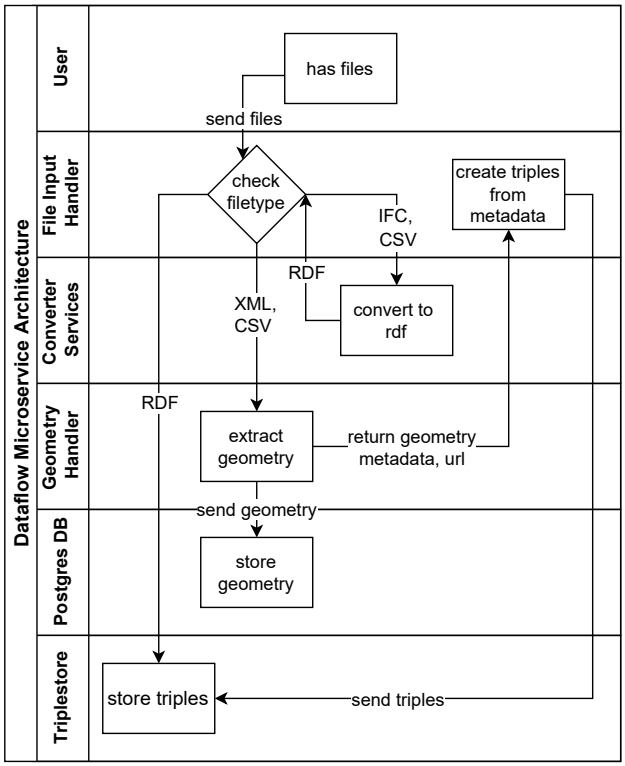

Figure 2. The Dataflow for data input through the components.

The files are sent to the File Input Handler. On the basis of file type or metadata, this handler decides, how to process the data. CSV files and IFC models are forwarded to their Converter Services, which return the converted data as RDF file. The File Input Handler imports the returned RDF data into the triplestore.

Files containing the RDF/XML or Turtle Syntax are imported directly to the triplestore.

LandXML and CSV files are forwarded to the Geometry Handler. CSV files, which were already processed by the RDF converter, are checked, if there is any geometry in the data. In that case, the geometry is extracted from the WKT and inserted into the PostgreSQL database by an insert SQL query. The same is done with the extracted geometry from LandXML files. Depending on type and dimension the geo-feature are inserted to different tables. These tables are simply named by these two

classifications: type and dimension. For example, 2D polygons are inserted in the table *polygon_2d*. During the import, all geometries (points, lines, polygons in 2D and 3D) are transformed in one coordinate reference system (CRS) for easier storage and calculations. Please note, that this simple and generic approach comes without any problem-specific data model. It only provides geospatial functionality. Any other types of relations are queryable from the triplestore.

For each geometry, the Geometry Handler collects metadata and sends them back to the File Input Handler. The metadata contain geometry type, coordinate dimension, source file name, id in PostgreSQL database and some other. Additionally, an IRI (Internationalized Resource Identifier) is part of it. This IRI is created by the Geometry Handler. It provides the link to the geometry in PostgreSQL database. This is realized by the API. The API is developed in conformance to the standard *OGC API–Feature–Part 1: Core*[7] which gives a schema, how features should be accessible over an API. A specific feature should be accessed over the path:

```
/collections/{collectionId}/items/{featureId}
```

We use this to create our IRI. It looks like this:

```
https://geodatabase.org/collections/polygon_2d/
items/ce8007d3-0798-4712-894a-8ce0c071300d
```

In our opinion GeoSPARQL is the best ontology to describe geometry in RDF at the moment. Other ontologies, like the WGS84, GeoNames or NeoGeo Vocabulary ontology, are less extensive and mostly support only geometries in WGS84 coordinate system.

The geometry with GeoSPARQL is represented as Well Known Text (WKT) or Geography Markup Language (GML) and can be typed with a coordinate reference system. But if calculations with coordinates of different coordinate reference systems (CRS) are required, the triplestore uses only the CRS of the firstly inserted geometry resource.

With GeoSPARQL most basic calculations are accessible from the triplestore. The specification includes, for example, the calculation of topological relations between geometries, creation of buffers or convex hulls. But until now, with GeoSPARQL can't be calculated more complex things like areas, volumes and 3D geometries in general. That's why we created our own ontology by extending the GeoSPARQL ontology. For example, we reorganized the topological predicates and added the class 'GeoLink' to the class 'Geometry', appended by the Predicate 'url'. The ontology snippet looks like this:

```
@prefix tto: <https://test-ontology.org/> .

tto:GeoLink rdf:type owl:Class .

tto:url rdf:type owl:ObjectProperty ;
        rdfs:domain geo:Geometry ;
        rdfs:range tto:GeoLink .
```

For better querying later we modeled some relations between the topological predicates. We added the expression 'owl:disjointWith' between the GeoSPARQL predicates 'geo:sfIntersects' and 'geo:sfDisjoint' and defined all other topological predicates (equals, touches, within, contains, overlaps, crosses, covers, coveredBy) as sub properties of 'geo:sfIntersects'. Instead of calculating the topological relations with the GeoSPARQL functions, we only use the predicates to describe the topological relations explicitly, which we calculated in PostgreSQL database. This is also done via an API. The API sends SQL (Structured Query Language) queries to the PostgreSQL database, which calculate the topological relations between all geometries. The results are mapped to the topological predicates and turned into triples with the two geometry representations in the triplestore as subject and object. Then they are imported into the triplestore. We use the topological relations as part of our Linked Data concept to link geometries of different sources.

All imported files are represented as an instance of class 'Document' in the triplestore. All metadata of one file are appended to this instance. With the predicate 'hasSource' every 'Feature' is linked to its source file.

Figure 3 shows the communication between the microservice components, as the client imports data into the system.

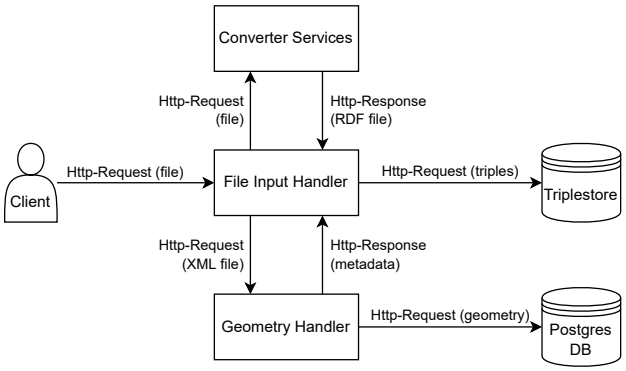

Figure 3. Communication between microservices while data input.

3.4 Requesting data

After the import and transformation the data are accessible with SPARQL requests. Later this endpoint will be replaced by an REST API. With this (future) REST API the user doesn't need any SPARQL skills and the answers are reproducible. The API is also needed to interpret the geometry link to the PostgreSQL database, if geo-feature are the requested output. Because of the Linked Data, SPARQL requests can be queries across many data source. These sources were previously (before import to our system) distributed over many files. Also the queries utilize cross-source links, that were created by our architecture, e.g. the persistent topological links.

Figure 4 visualizes, how a query on Linked Data is answered by the system. The client sends a request to the API of the Event Handler. The Event Handler translate the incoming request into a SPARQL query and forward the requests to the triplestore. For now, we use a SPARQL endpoint in order to directly send the query to the triplestore. If any geometry is part of the response, the link to PostgreSQL needs to be interpreted by the client. As a matter of work-in-progess reserach this is done manually/hardcoded until now. Later the Event Handler will interpret the link by sending a request to the Geometry Handler, who returns the geometry as WKT string. WKT and non-geometric response of the SPARQL request is then eventually forwarded to the client.

[7] https://www.ogc.org/standards/ogcapi-features

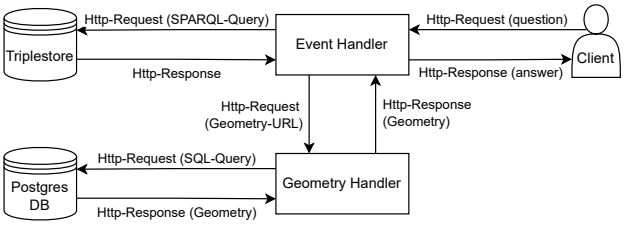

Figure 4. Communication for answering a clients question.

4. SIMPLE TEST-SCENARIO

In this section we will show one of our simple tests. Currently we convert a complete IFC model to RDF and then import the RDF document to the triplestore. However, in Section 5, we will postulate that a pure import is not efficient, due to the highly decomposed structure of IFC geometries as triples in RDF.

In our simple test we want to prove our hybrid concept for BIM/GIS integration. We used geospatial data attached with additional non-geometric environmental information and a digital terrain model (DTM). The geo-feature are given as WKT strings in a text file (CSV) and the DTM is uploaded as LandXML format. In the test scenario a residential house is delivered as georeferenced BIM model in IFC (Figure 5).

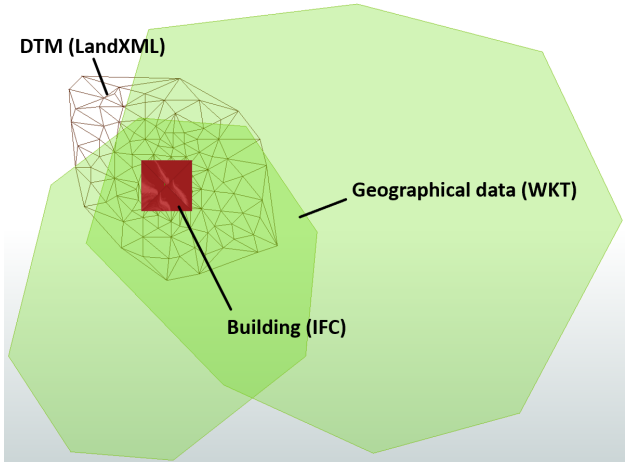

Figure 5. 2D view of test BIM and GIS data from heterogeneous sources.

Our aim is to link these data in order to be able to answer advanced queries. A use-case could be a building permit subroutine, that checks if it is allowed to build the house in this area or which type of land use is affected by the building. This queries need the relations between the geometries to be answered.

For the practical integration of BIM and GIS we created the georeferenced footprint of the IFC Building as WKT string to make topological calculations. The footprint is imported into the PostgreSQL database as the other geometries and linked to the IFC building resource in the triplestore. In the future, the footprint will be calculated automatically on import of the IFC file. In the future we intend to add the footprint-service to our dockered microservice architecture.

Then the Event Handler triggers the calculation of topological relations by PostGIS. The results are saved explicitly into the triplestore using our extended GeoSPARQL ontology. As a first result, our triplestore is filled with Linked Data with a connection to the geometries in the PostgreSQL database.

In the next step we can answer SPARQL queries on the linked and extended data. Figure 6 shows a query, where we want to know, which land uses are affected, if the building is constructed on this position. Additionally, we want to get the geometries of affected land uses. The query first searches for a building, which has a footprint geometry object. This footprint object intersects with other geometry objects. From this other geometry objects is searched for the feature resources and their land use types. The last triple asks for the link of each land use object to the geometry in the geodatabase.

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX tto: <https://test-ontology.org/>
PREFIX lu: <https://example.org/landuse/>
PREFIX geodb: <https://geodatabase.org/>

select ?building ?landUseType ?urlToGeoDatabase where {
    ?building geo:hasGeometry ?footprint .
    ?footprint geo:sfIntersects ?landUseGeo .
    ?landUse geo:hasGeometry ?landUseGeo .
    ?landUse lu:landUseType ?landUseType .
    ?landUseGeo tto:url ?urlToGeoDatabase.
}
```

Figure 6. SPARQL query for the question: 'Which forms of land use are affected by constructing the building? Give the geometry of these land uses.'

The query especially asks for the building resource, the types of affected land uses and their geometry links. The query results are presented in figure 7. The results show two affected land uses, what is consistent with the two green polygons what can be seen in figure 5. The full link to the geodatabase can be executed and returns the geometry as Well Known Text and the id in JSON format. The output format may be changed by *content negotiation*.

	building	landUseType	urlToGeoDatabase
1	bldg:Building_1	"Forest"	geodb:collections/polygon_2d/items /ce8007d3-0798-4712-894a-8ce0c071300d
2	bldg:Building_1	"Grassland"	geodb:collections/polygon_2d/items /f48a0c20-98ae-4e9f-a816-713e25a7b87b

Figure 7. Result of the query, containing the building resource, the affected land use types and the links to their geometry in the geodatabase.


```
{
  "id": "ce8007d3-0798-4712-894a-8ce0c071300d",
  "geometry": "SRID=25832;POLYGON((685636.701199641 5646606.3536494,
              685686.011941892 5646602.44895838,685718.819502458 5646553.76775835,
              685713.569069366 5646496.58646396,685652.508571504 5646448.69792641,
              685605.234283641 5646452.58222739,685576.576094471 5646496.59665898,
              685596.186207196 5646575.49078808,685636.479635002 5646606.11546631,
              685636.701199641 5646606.3536494))"
}
```


Figure 8. The response of the first geometry link from the SPARQL query as WKT string together with id in JSON format.

This example shows, that the hybrid approach can be successfully applied for BIM-GIS integration.

For the semi-automatic linking with topological predicates footprints are needed, of each IFC building or for each IFC element. If these are not on place but needed for test-scenarios, the linking between BIM and GIS objects can be done manually in our system.

5. CONCLUSION AND OUTLOOK

It has turned out that geometry can not be saved and calculated in triplestores as performant as in geodatabases. The GeoSPARQL standard is a good basis, but both the standard and its implementation in the triplestores has to improve to compete with geodatabases.

That's the reason why we want to introduce a hybrid approach, which combines the advantages of a geodatabase, namely the geometry handling and CRS, with the possibilities of semantic linking from triplestores and the semantic web technologies in general. Geometries can be hold in a geodatabase, where the full range of geospatial calculations can be done on it. Basic calculations like the topological relations can be calculated ones and saved explicitly in the triplestore. They can be used for linking and querying, without calculating every time, which is good for the query performance.

Using Open API 3.0 makes it possible to reach all geometries over an IRI. The API is flexible because many requests over different tables and geometries can be made with one base query. An API provides a standardized access to the geometry data so that it can also be used in distributed environments.

Our hybrid approach gives us an idea, how geodatabases can be accessed by semantic web technologies, without the need of converting all data into RDF. The developed microservice architecture is flexible for adding more applications, which can be developed and tested while the basic system is running, thanks to the docker system.

In the next step, we will add the BIMserver[8]. The BIMserver can publish IFC documents and the single building elements can be accessed by an API. Because of the IFC structure, geometry is not presented very well in RDF. That's why a huge part of IFC triples in RDF is describing geometry. We think, that it would be better to save the geometry in domain specific databases (PostGIS, BIMServer). Only the remaining (non-graphical) parameters will be hold in the triplestore and the geometry would be linked through the BIMserver API. An API between BIMserver and triplestore could also solve the problem, that IFC elements have no absolute coordinates. The service behind the API could calculate the absolute position so that topological relations to other geometries can be found and be serialized in the triplestore. Also the Event Handler has to be implemented, that users can easily make standard requests without the need for SPARQL. The future API should also hide the link to the feature in the geodatabase, only providing the requested collection or feature in the desired format.

As future research we plan to extend our ontology to make other than only topological relations between geometries and use more problem-specific functionality.

In the course of the research work it has been shown that the many possibilities of modern IT systems (Semantic Web, Triplestore, Java Frameworks, Docker, openAPI, etc.) offer many possibilities if they are specifically applied to the application problem.

[8] https://github.com/opensourceBIM/BIMserver

ACKNOWLEDGEMENTS

This research was funded by the German Federal Ministry for Economic Affairs and Energy (BMWi), Central Innovation Programme for small and medium-sized enterprises (SMEs), Funding No. 16KN086446 *TerrainTwin*

REFERENCES

Battle, R., Kolas, D., 2011. Linking geospatial data with GeoSPARQL. *Semantic Web Journal – Interoperability, Usability, Applicability*. http://semantic-web-journal.org/sites/default/files/swj176.pdf.

Beck, S. F., Abualdenien, J., Hijazi, I. H., Borrmann, A., Kolbe, T. H., 2021. Analyzing Contextual Linking of Heterogeneous Information Models from the Domains BIM and UIM. *ISPRS International Journal of Geo-Information*, 10(12). https://www.mdpi.com/2220-9964/10/12/807.

El-Mekawy, M., Östman, A., 2010. Semantic mapping: an ontology engineering method for integrating building models in ifc and citygml. *3rd ISDE Digital Earth Summit, 12-14 June*.

Guyo, E., Hartmann, T., Ungureanu, L., 2021. Interoperability between bim and gis through open data standards: An overview of current literature. Technical report.

Hor, A.-H., Sohn, G., 2021. Design and Evaluation of a BimGis Integrated Information Model Using Rdf Graph Database. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 8, 175–182.

Jetlund, K., 2021. Harmonizing and linking conceptual models of geospatial information: Technologies for information modelling in GIS, ITS and BIM. Dissertation, NTNU – Norwegian University of Science and Technology.

Jovanovik, M., Homburg, T., Spasić, M., 2021. A GeoSPARQL Compliance Benchmark. 10(7), 487. PII: ijgi10070487.

Karimi, S., Iordanova, I., 2019. Integration of BIM and GIS for Construction Automation, a Systematic Literature Review. *Archives of Computational Methods in Engineering*, 28.

Noardo, F., Arroyo Ohori, K., Biljecki, F., Ellul, C., Harrie, L., Krijnen, T., Kokla, M., Agugiaro, G., Stoter, J., 2020a. Geobim benchmark – isprs scientific initiative 2019 – final report.

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., 2020b. Tools for BIM-GIS Integration (IFC Georeferencing and Conversions): Results from the GeoBIM Benchmark 2019. 9(9), 502. https://www.mdpi.com/2220-9964/9/9/502. PII: ijgi9090502.

Pauwels, P., Zhang, S., Lee, Y.-C., 2017. Semantic web technologies in AEC industry: A literature overview. 73, 145–165. https://www.sciencedirect.com/science/article/pii/S0926580516302928.

Roxin, A., Hbeich, E., 2019. Semantic interoperability between BIM and GIS – review of existing standards and depiction of anovel approach. hal-02279633. https://hal.archives-ouvertes.fr/hal-02279633.

Ulutaş Karakol, D., Kara, G., Yılmaz, C., Cömert, Ç., 2018. Semantic linking spatial RDF data to the web data sources. XLII-4, 639–645.