Leveraging Knowledge Graphs and Semantic Web Technologies for Validating 3D City Models

Abstract

The widespread use of 3D (three-dimensional) city data plays a significant role in various applications such as mixed reality, infrastructure facility management, solar potential analysis, navigation and so on. Ensuring high spatial and semantic quality in these endeavours is crucial to gathering proper results. Ensuring quality means verifying that the data adheres to relevant standards. Although these relevant standards are openly published, there are issues in names of interoperability and reusability in academic studies and software development efforts. In this study, these issues are addressed using semantic web technologies. 3DCMs (3D city models) are treated as knowledge graphs (KG) with this approach. The main contribution of the study is a web-based interoperable tool for validation of CityGML LOD2 (Level of Detail 2) 3DCMs which is compatible with relevant standards. Besides, an open-source 3DCM-to-KG converter and an open validation ontology are published as by-products while accomplishing the main goal. By virtue of the KG approach, the 3DCM KG becomes capable of carrying its own validation constraints which come from the validation ontology. With these efforts, this study provides a practical interoperable solution to improve the quality and usability of 3DCMs and validation plans, fostering consistency across applications while aligning with established standards in the field.

Keywords: 3D City Models, Data Quality, Semantic Web Technologies, Knowledge Graphs, Validation

1. Introduction

Geographic or spatial data constitutes 80% of the approximately 2.5 exabytes of data produced daily in the world and 60% of the data held in existing databases is spatial data [1, 2]. This sheer amount of spatial data underpins numerous applications, from navigation systems to urban planning, where the need for detailed, accurate, and structured representations of the physical world is ever-growing. Given this dominance of geospatial data, 3D city models (3DCMs) emerge as a critical tool, transforming raw spatial information into structured, three-dimensional representations of urban environments. 3DCMs are digital representations of urban objects with three-dimensional geometry, foremost buildings, and are essential subjects for a large range of tasks which are vital for our understanding and planning of cities today [3]. 3DCMs should ensure high spatial and semantic quality to gather accurate results from these tasks. For example, in a solar analysis, defining surfaces that are not roofs in use as roofs or errors in the surface geometries will be misleading in terms of the solar panel placement. Therefore, purchasing and using invalid data can cause economic drawbacks [4] and ensuring quality has been a crucial research topic in the field. With the mentioned use of 3D city models across various applications and domains, CityGML was introduced as a standard data model and an exchange format [5], and data quality studies in the field have usually been carried out on this model.

The validation on CityGML datasets generally consists of three main tasks which are schema validation, geometry validation and semantic validation. Schema validation is the validation of CityGML documents against CityGML schema documents, and it can be considered a well-known solution due to its being widely performed in the literature. Geometry validation is the validation of city objects in a CityGML dataset against the definitions in the ISO:19107 Spatial Schema Standard [6, 7]. This standard provides a description set for representing and manipulating geographic entities through vector-based geometric primitives, such as points, curves, surfaces, and solids, and topological relationships, ensuring that the geometry of city objects adheres to consistent and mathematically defined structures. As an example of the geometry validation control, it can be given that GML rings' are topologically closed, and the first and last points of GML rings have to be identical. Semantic validation is performed even if the objects in the CityGML data pass schema validation and geometric validation to detect if it contains unrealistic representations in object classes or attribute fields. An example of a

semantic error is that the value in the attribute field expressing the measured height of a building does not match the height value calculated from the coordinate arrays, or, a wall or balcony surface is represented as a roof. The structured list of which of all these controls will be applied on a CityGML document and in what order is called a validation plan [8].

Studies before 2013, they do not include semantic validation [9, 10, 11]. These studies only dealt with the geometric validation of city models. The semantic validation concept was obviously emphasised in the study done by [12] for the first time. In addition to performing semantic validation, this study emphasises interoperability and the fact that data exchange should satisfy the data provider and the client.

With the experiment (Quality Interoperability Experiment, QIE, 2016) done under the editorial of Wagner and Ledoux [8], validation checks of CityGML 3DCMs are classified and tested by various software. In their study, requirements for the validation of CityGML-formatted city models are defined in detail. The tested validation requirements for CityGML data consist of semantic checks for the “Building” thematic class in addition to schema and geometry validation. While the study shows an extensible requirement coding system for other thematic classes, the tested requirements in this study can be considered as a minimum consensus basis as buildings are essential elements of 3DCMs. Although this study is a well-organized guide in the field, there is still a lack of agreement with the minimum basis in the applied validation plans, even in most current studies. This situation is a drawback to interoperability because the mentioned methodologies are not able to produce an identical validation result for the same 3DCM due to the lack of consensus. Most of the current studies on the quality are mostly focusing on the positional accuracy of models regarding ground truth data [13,14,15]. In [15], they also employed val3dity [16] software to perform geometry validation besides the positional accuracy. The study of [17] performs several semantic checks without depending on any basis, such as QIE. They also did not perform any geometric checks so while semantic validation requires being valid geometrically, the results of performed checks are not meaningful. In [18], they perform a validation process on IFC-formatted (Industry Foundation Class) BIM (Building Information Model). As stated in the study, there is a lack of standardization work in the BIM field for interoperability in which quality requirements are defined so this study is in a state of a prototype.

The first realised application of QIE is the study [19]. They adopted a validation plan based on QIE for the context of heating demand simulation, using CityDoctor [20] software. With the last version of CityDoctor software, a realised tool for QIE is freely available to use. The error codes and the minimum validation requirements in QIE are implemented in this last version of CityDoctor. Besides, the validation results are stored in the same CityGML file through DataQualityADE [21]. The implementation is remarkable progress for interoperability while the rest of the available tools are not serving identical results even if they are applying the same checks [8]. The difference between error codes and results for the same checks also hinders the reusability of validation results because comparing and evaluating these results together into the same process at different times sequentially requires additional code-matching processes. The interoperability and reusability problems can be demonstrated in Figure 1.

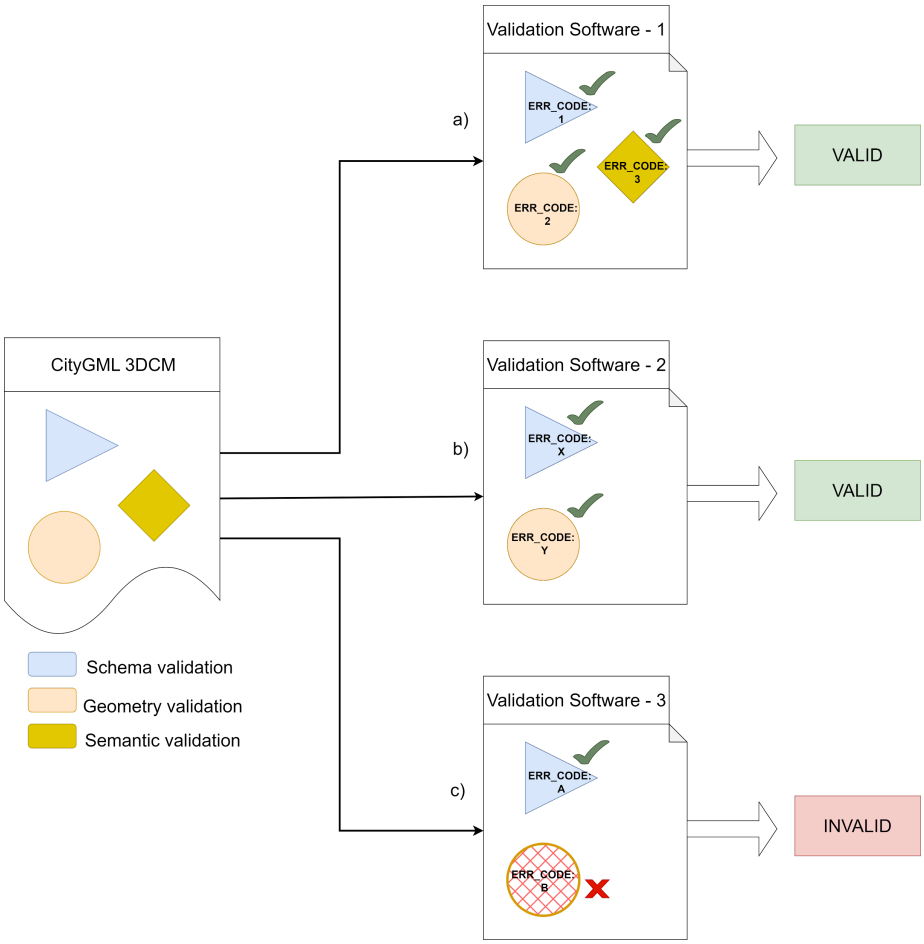


Figure 1. Validation results of three fictional validation software against the same 3DCM data. Software-1 and Software-2 claim the 3DCM is valid but Software-2 does not include semantic validation. Software-1 and Software-2 include the same kind of checks but produce different results due to the need for more consensus between applied checks in the context of geometry validation. Besides, while different providers are using different software, melting their results into the same pot requires a code-matching phase due to the difference in error codes, even if the applied checks are the same.

In this study, we have aimed to solve interoperability and reusability problems by leveraging the knowledge graph (KG) approach and semantic web (SW) technologies. With this approach, validation plans can be directly transferred and accessible online with the associated 3DCM KG as validation graphs. Section 1.1. describes the relationship between semantic web technologies and the validation process in detail while exposing the effect of the KG approach on the solution of the interoperability and reusability problems. We have described our solution and developed web-based software in Section 2. Section 3 discusses the experimental results according to the current state of other relevant solutions in the literature. Finally, future plans and prospects of the adopted approach are put forward in Section 4.

1.1. Knowledge Graphs, Semantic Web and Validation

Semantic web (SW) is an extension of the well-known document-based web which enables the cooperation of people and computers using the information with well-defined meaning [22, 23]. Today we know this cooperation as interoperability and the mentioned well-definition of the meaning is possible thanks to the

links between resources. The SW paradigm uses RDF (Resource Description Framework) language, or format, for interlinking between knowledge resources [24]. These interlinked resource networks defined in RDF format are also known as knowledge graphs (KGs) which are directed and labelled graphs [25].

RDFs are constructed with triples which are subject, predicate and object sequences. Subjects and objects denote entities that represent objects from the world with URIs (Unique Resource Identifiers), and predicates denote relationships between these entities. Therefore, subjects and objects become nodes and predicates become edges of a RDF-formatted KG. KGs are used either to represent application domains or individual relations in any domain [26]. If a KG represents classes and properties of an application domain, the KG is named as an ontology, a T-Box, or a namespace [27]. We define the ontologies using OWL (Web Ontology Language), an RDF derivative syntax which is a W3C recommended standard for the modelling of ontologies [28]. Suppose a KG represents relations between individuals in any domain, represented by a T-Box (the T stands for "terminology"), or attribution ownership, the KG is named an A-Box (the A stands for "assertions"). This manner enables an A-Box to carry its relevant definitions through a T-Box over the web so, this semantic richness of data transfer between different information systems or applications in various domains eases the interoperability. This reliably compatible transfer mechanism is even more useful for the data among applications or situations that include unforeseen schemas [29, 30].

The idea of KG is also currently being researched in the architecture, engineering, and construction (AEC) industry, especially for validation, as adopted in this study. While most of the efforts in the literature are focused on BIM data [31, 32, 33, 18], there is still a research gap for a noteworthy effort for CityGML models. In this study, we adopted the KG approach for validation of CityGML, specifically using the Shapes Constraint Language (SHACL) [34] which is a W3C recommendation language and standard to validate RDF-formatted KGs [36].

The SHACL engine validates RDF graphs by defining shapes which are sets of constraints, and by comparing these constraints against nodes of the RDF graph. The constraints in a shape are mandatories or restrictions that RDF data must and are defined according to an ontology, which is also what the RDF graph nodes are defined in [35, 36]. A set of SHACL shapes is also known as a shapes graph, hence, a SHACL document can also be considered as another KG to validate the RDF-formatted data graph [37]. For the validation process, constraints in a SHACL shape are compared to the associated data graph node. This associated node is also known as the target node. If the SHACL processor finds an inconsistency between a shape and a node, it reports a violation for the target node regarding the shape in question in the validation report. The relation between SHACL shapes and data graph nodes is demonstrated in Figure 2, and the unity as connected data and shapes graphs is shown in Figure 3.

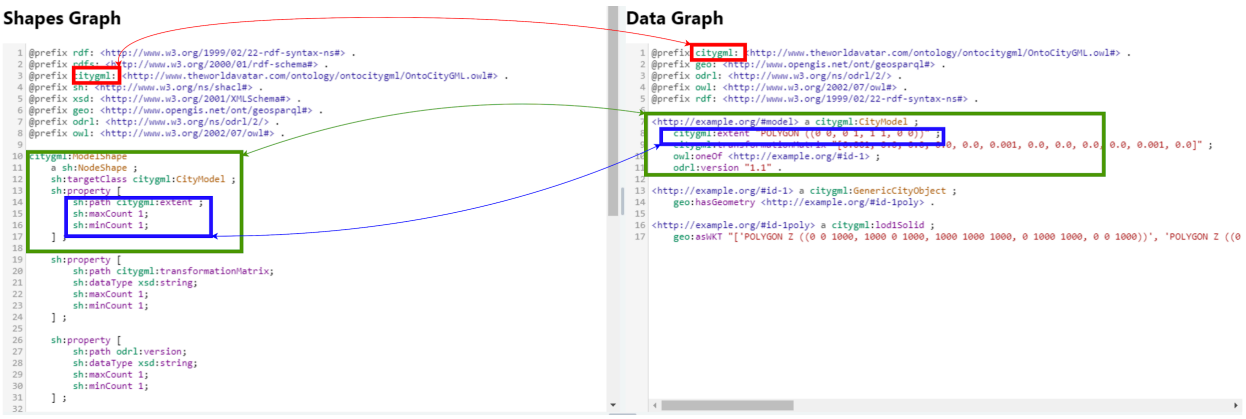


Figure 2. Prefixes at the top of both graph documents represent URIs of the used ontologies. The prefix "citygml" (highlighted with red) belongs to the ontology in which data and shapes graphs are both defined. The shape "citygml:ModelShape" (highlighted with green) contains constraints to validate the individual, or instance, that belongs to the CityModel class in "citygml" ontology. The constraint "citygml:extent" (highlighted with blue) contains the restrictions, which are about the count of occurrence for the "extent" property belonging to the CityModel instance.

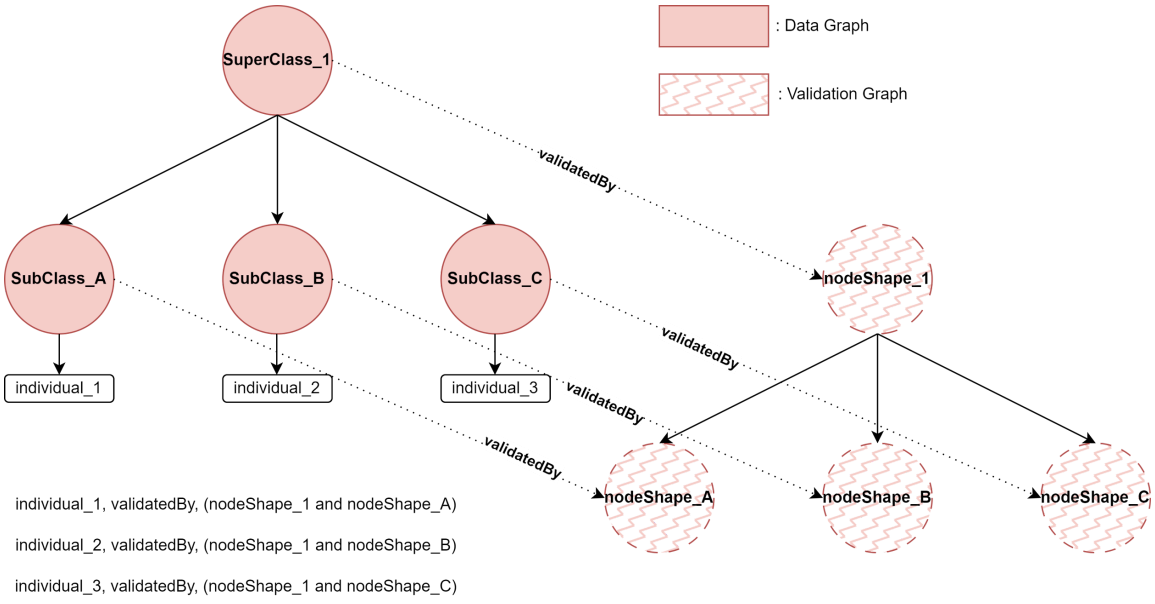


Figure 3. The connectivity between data and shapes graphs. Each shape in the shapes graph validates an individual, or a target node, within the same hierarchy as the data graph.

By virtue of this mechanism, in this study, we have adopted the KG approach to validate LOD2 CityGML models. With this approach, it can be enabled to carry a 3DCM KG with its own shapes graph which contains validation constraints for the application domain of the 3DCM. Besides, it could be also enabled to process multidomain data in different formats into the same application frame thanks to the RDF meeting point and the data's having self-definitions. This approach will play a key role in ensuring interoperability by informing the data client about what the data is being validated against and restricting the data provider about the standards according to which the data should be presented. In addition to the merits mentioned above, semantic

web technologies allow for the inference of new information using the existing relationships in the data [38]. This enables the discovery of implicit relationships or, in our context, implicit validation rules that are not explicitly stated in the KG. This capability also opens a way to recover missings or eliminate inconsistencies from ontology statements [39]. Our methodology, which is explained in detail in Section 2., consists of the main steps below to implement this KG approach;

- ➢ Creating a "Validation Ontology" based on QIE 2016,
- ➢ Extracting the shapes graph from "Validation Ontology" using automated web tools,
- ➢ Converting the CityGML data into an RDF graph,
- ➢ Validation of the RDF graph against the shapes graph,
- ➢ Usage of the pipeline through the web interface with different CityGML datasets,

2. Methodology

2.1. General pipeline

The general pipeline of the implemented validation process is demonstrated in Figure 4. According to the pipeline, the client accesses the validation software and uploads their CityJSON formatted city model data. Following file upload, the client should choose the ontology against which the data will be validated. In our implementation, regarding our needs, the only ontology option is the validation ontology which is created based on QIE 2016 (2. component in Figure 4). After this selection, the selected ontology and the CityJSON data are sent to the server for the validation process. On the server side, the CityJSON file is converted to an RDF file, in other words, 3DCM KG (1. component in Figure 4.). The KG is validated against the SHACL graph generated regarding the validation ontology via automated OWL to SHACL web tool (3. component in Figure 4). The validation process produces a human-readable validation report that includes all geometric and semantic violations for every object level in a city model such as Ring, Polygon and Shell (4. component in Figure 4). Then, this report is sent back to the client for the examination. The rest of the section continues with explanations of the components of the architecture.

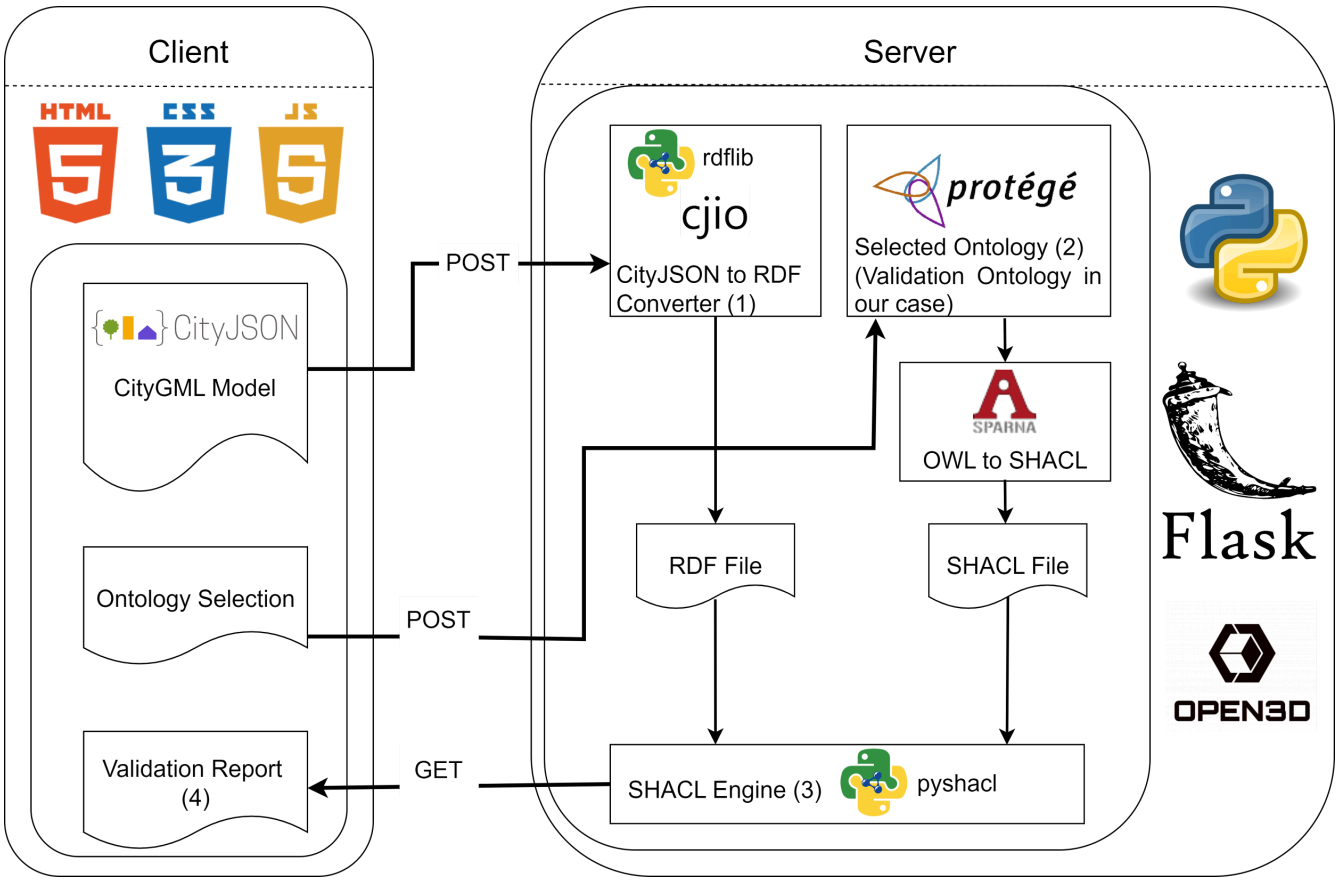


Figure 4. The general pipeline of the study

2.2. Creation of the validation ontology

At this phase, as a crucial need for the validation, a validation ontology has been created which includes the validation check definitions in QIE 2016. This ontology plays an essential role in generating both SHACL and RDF graphs. Regarding the definitions in QIE 2016, the validation ontology must also be inherited from GeoSPARQL[40], CityGML [41, 42], and Simple Features [43] namespaces in addition to RDFS and OWL built-in ontologies. Hence, the ontologies of GeoSPARQL, CityGML and Simple Features were used as base T-Boxes. At first, the base ontologies mentioned before which are RDFS, OWL, CityGML and Simple Features were imported into a workspace of the Protege ontology editor [44]. The required data properties of QIE 2016 were included in the workspace by linking the relevant classes of base ontologies. The properties were divided into 2 classes which are geometric and semantic properties regarding the rule classification in QIE 2016. The geometric properties were also divided into 3 subclasses which are shell properties, ring properties and polygon properties. All properties have values as boolean literals, as True or False. Lastly, the additional class, "validity," states the data's validity status and is created to store validation results after the validation process. After the consequent property classification and property-base class linking processes, the created ontology can be exported in any desired format such as OWL, Turtle, JSON-LD, N-triples, etc. The JSON-LD (JSON for linking data), which is the chosen format for this study, is gaining popularity in the field thanks to the JSON format's wide usage in web applications. The properties of the validation ontology and the linked base ontology classes are given in Table 1. In Table 2, definitions of the properties are given considering the QIE 2016 study.

Table 1. Property classes and associated object classes

Property Class	**Subclass of Geometric Properties**	**Property Name**	**Linked Object Class**
Geometric	Shell Properties	areAll3AnglesConnected	citygml:SolidType
		hasSelfIntersections	
		isCorrectOriented	
		isEdgeManifold	
		isVertexManifold	
		isWatertight	
		tooFewPolygons	
	Ring Properties	consecutiveSamePoints	sf:Surface
		isClosed	
		isCollapsedtoLine	
		noSelfIntersection	
		tooFewPoints	
	Polygon Properties	hasDuplicatedRings	sf:Polygon
		hasHoleOutside	
		hasInnerNestedRings	
		hasInteriorDisconnected	
		hasIntersectedRings	
		isCcwise	
		isCoplanar	
		isNormalsDeviated	
Semantic	Semantic	wallSurfaceNormals	citygml:WallSurfaceType
		wallSurfacePolygonNormals	citygml:BuildingType
		roofSurfaceNormals	citygml:RoofSurfaceType
		roofSurfacePolygonNormals	citygml:BuildingType
		groundSurfaceNormals	citygml:GroundSurfaceType

		groundSurfacePolygonNormals	citygml:BuildingType
		outerCeilingSurfaceNormals	citygml:OuterCeilingSurfaceType
		outerFloorSurfaceNormals	citygml:OuterFloorSurfaceType
		storeyHeightEqualGeometry	citygml:AbstractBuildingType
		attributeHeightEqualsGeometry	citygml:AbstractBuildingType
Validity	-	validity	citygml:CityModelType

Table 2. Property definitions of the validation ontology

Property Name	**Definition**	**Desired Value**
areAll3AnglesConnected	States if all triangles of the shell in question are connected.	True
hasSelfIntersections	States if the shell in question has self-intersections.	False
isCorrectOriented	If one polygon is used to construct the shell, this property states if the points of the exterior ring are ordered counterclockwise.	True
isEdgeManifold	States if each edge of the shell has exactly 2 incident polygons.	True
isVertexManifold	States if each vertex of the shell constructs an umbrella with its incident polygons.	True
isWatertight	States if the shell in question is watertight.	True
tooFewPolygons	States if the shell in question has at least 4 polygons.	False
consecutiveSamePoints	States if the ring in question has repeated points, except first and last points.	False
isClosed	States if the points of the ring construct a closed loop.	True
isCollapsedtoLine	States if the points of the ring collapse to a line.	False
noSelfIntersection	States if the ring in question has self-intersections.	True
tooFewPoints	States if the ring in question has at least 3 unique points, except the first and last points.	False
hasDuplicatedRings	States if the polygon in question has two or more identical rings.	False

hasHoleOutside	States if the polygon in question has any exterior rings.	False
hasInnerNestedRings	States if any of the interior rings of the polygon are completely inside another interior ring.	False
hasInteriorDisconnected	States if any of the interior rings of the polygon are disconnected.	False
hasIntersectedRings	States if any rings of the polygon are intersected.	False
isCcwise	States if the points of the exterior rings of the polygon are all oriented counterclockwise.	True
isCoplanar	States if all rings of the polygon in question are coplanar within a tolerance value.	True
isNormalsDeviated	States if all rings of the polygon in question have deviated surface normals within a tolerance value.	False
wallSurfaceNormals	States if the face normal of the WallSurface in question has zero Z direction within a tolerance value.	True
wallSurfacePolygonNormals	If the WallSurface consists of several connected polygons, this property states if these polygons have deviated surface normals within a tolerance value.	True
roofSurfaceNormals	States if the face normal of the RoofSurface in question has a positive Z direction within a tolerance value.	True
roofSurfacePolygonNormals	If the RoofSurface consists of several connected polygons, this property states if these polygons have deviated surface normals within a tolerance value.	True
groundSurfaceNormals	States if the face normal of the WallSurface in question has a negative Z coordinate within a tolerance value.	True
groundSurfacePolygonNormals	If the WallSurface consists of several connected polygons, this property states if these polygons have deviated surface normals within a tolerance value.	True
outerCeilingSurfaceNormals	States if the face normal of the OuterCeilingSurface in question has a negative Z direction within a tolerance value.	True
outerFloorSurfaceNormals	States if the face normal of the OuterFloorSurface in question has a positive Z direction within a tolerance value.	True
storeyHeightEqualGeometry	States if the sum of all storey heights, which is stored in the storeyHeightsAboveGround attribute, is equal to the total height of the building geometry.	True
attributeHeightEqualsGeometry	States if the value of the measuredHeight attribute is equal to the total height of the building geometry.	True

validity	This field includes the validation report with a unique URL	String

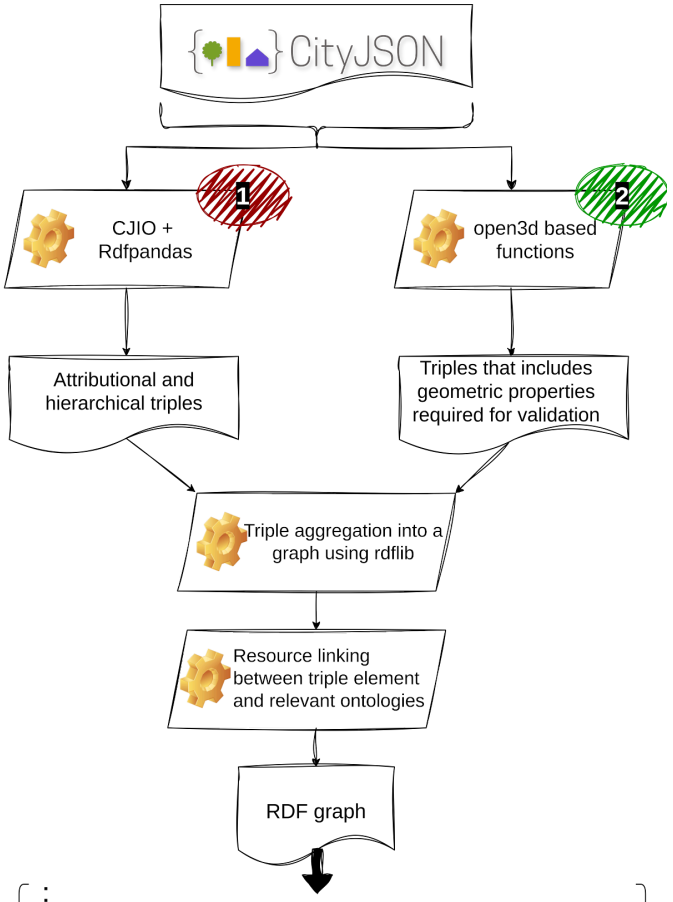
2.3. CityJSON to RDF conversion

This phase consists of consequent processes of CityJSON parsing, creating triples, and linking triple elements with relevant definitions in the validation ontology. The CityJSON encoding of the CityGML data model is chosen for this study thanks to its comfort mapping to Python dictionaries and its less nested structure than GML encoding [45]. The parsing of the CityJSON file is carried out using the cjio Python library which is created for CityJSON file management [46]. The CityJSON file is converted into a DataFrame with this parsing process to ease the triple extraction for the RDF output. The nature of DataFrames is similar to Python dictionaries, which is also similar to JSON files, hence it becomes easier to create subject-predicate-object combinations from key-value pairs such as "key-is a-value". The Rdfpandas Python library is used to extract triples from DataFrames [47]. In this extraction, the row indices of the DataFrame become subjects, the column names become predicates, and the corresponding values become objects. The triple store is an RDF-like property graph without any resource linking in this state [48]. While all processes till this state can be automated using the mentioned libraries before linking triple elements and the validation ontology, further information from the CityJSON data must be extracted to satisfy the properties of the validation ontology. This is needed because the cjio library only transfers attributes to the DataFrame, not geometries and thematic classes [49]. The CityJSON-to-RDF converter includes additional operations to overcome this problem. The thematic classes and parent-child relations are extracted with the additional Python functions which traverse the CityJSON data to gather this information.

To extract and include geometries into triples, the open3d Python library is used [50]. Regarding the data properties in the validation ontology, the extracted geometric information is attached to the triple store as Boolean values. The open3d library provides the necessary 3D computational geometry components to create additional functions to calculate geometric Boolean properties from the coordinate sequences (Polygon Z WKTs (Well Known Text)) obtained from the CityJSON data. After the calculation of all required properties, all triple elements in the store are linked to the relevant definitions in the validation ontology. As a result of the linking process, the triple store of CityJSON data becomes a 3DCM KG regarding the validation ontology and can be exported in any RDF format, such as Turtle, JSON-LD, N-triples, etc. The RDFLib Python library is chosen for the RDF file creation [51]. Figure 5 includes the workflow of RDF creation process and a part of a created RDF file.

The bottom side of the Figure 5 includes a chunk of triple samples from created RDF files. The red highlighted triples include attributional, hierarchical and object class information. These triples are gathered from DataFrames generated by cjio library, and by CityJSON file traversing for extracting hierarchical, parent-child or hasGeometry, relations. The green highlighted triples include geometric properties which are required for check of validation requirements such as watertightness, having self-intersections, being vertex manifold, and so on. These geometric properties are calculated using open3d library-based functions. Following the triple extraction, all generated triples are aggregated into a graph, then elements of these triples (subject-predicate-object) are linked into relevant T-Boxes. The "valid: " prefix in the figure states the validation ontology, and the part after the colon states the validation properties and corresponding calculated values. Similarly, the "geo: " prefix states the GeoSPARQL ontology, and the "citygml: " prefix states the CityGML ontology. As the data part in the figure, the object with the

“ex:GUID_DBDABF53-7DD5-4C2F-BE7F-51F29A0CBA16_1” URI is an instance of BuildingPart class of the CityGML. The object “has geometries” as six surfaces. The object’s “geometry type” is Lod2Solid of the CityGML data model. The object’s parent is the Building object with another URI. The object’s watertightness is calculated as “True” regarding the validation ontology. Further inspections or exemplifications are possible through the figure or the repository of the project which is already given in the Conclusions section.




```
ex:GUID_DBDABF53-7DD5-4C2F-BE7F-51F29A0CBA16_1 a citygml:BuildingPart,
      citygml:SolidType ;
   geo:hasGeometry ex:GUID_DBDABF53-7DD5-4C2F-BE7F-51F29A0CBA16_1_0,
      ex:GUID_DBDABF53-7DD5-4C2F-BE7F-51F29A0CBA16_1_1,
      ex:GUID_DBDABF53-7DD5-4C2F-BE7F-51F29A0CBA16_1_2,
      ex:GUID_DBDABF53-7DD5-4C2F-BE7F-51F29A0CBA16_1_3,
      ex:GUID_DBDABF53-7DD5-4C2F-BE7F-51F29A0CBA16_1_4,
      ex:GUID_DBDABF53-7DD5-4C2F-BE7F-51F29A0CBA16_1_5 ;
   valid:areAll3AnglesConnected true ;
   valid:attributeHeightEqualsGeometry true ;
   valid:groundSurfacePolygonNormals true ;
   valid:hasSelfIntersections false ;
   valid:isCorrectOriented true ;
   valid:isEdgeManifold true ;
   valid:isVertexManifold true ;
   valid:isWatertight true ;
   valid:roofSurfacePolygonNormals true ;
   valid:tooFewPolygons false ;
   valid:wallSurfacePolygonNormals true ;
   citygml:GeometryType citygml:lod2Solid ;
   citygml:SolidType citygml:lod2Solid ;
   citygml:height 3.134e+00 ;
   citygml:measuredHeight 3.133e+00 ;
   citygml:parent ex:GUID_DBDABF53-7DD5-4C2F-BE7F-51F29A0CBA16 ;
   citygml:roofType "1000" .
```

Figure 5. RDF creation workflow and a sample part of RDF data.

2.4. Generating the SHACL graph and carrying out the validation

The SHACL graph generation process is actually a conversion from property restriction axioms in the ontology, which are given in Table 2 for our purpose, to the SHACL shapes. The process can also be summarised as a conversion between statements like “Object-has value-X” and “Object-must have a value-X” to enable the detection of violations in the data graph using the SHACL engine.

The conversion between the ontology and the SHACL graph is certainly accomplished in an automated manner without any handwritten editing process. The “owl2shacl” conversion tool by SPARNA [52] is used for an automated process. Table 3 shows two samples from the restriction axioms and the corresponding SHACL shapes. The SHACL graph, which includes such shapes, is used for the validation process. The validation process is carried out using the pySHACL library, which handles SHACL graph manipulations and validation processes [53]. The final validation report includes such violations by individuals of every class which has property restrictions in the validation ontology. In this way, invalid objects of RDF data can be inspected by the client thanks to detailed information for each individuals. Figure 6 shows a few samples from validation processes between RDF statements and the relevant SHACL shapes.

Table 3. Two samples of ontologic restrictions and corresponding SHACL shapes

Ontologic Property Restriction Definitions	**SHACL Shapes**
:isWatertight rdf:type owl:DatatypeProperty ; rdfs:subPropertyOf :shellProperties . … <citygml#SolidType> rdf:type owl:Class ; rdfs:subClassOf <citygmll#GeometricPrimitiveType> , [rdf:type owl:Restriction ; owl:onProperty :isWatertight ; owl:hasValue "true"^^xsd:boolean] ,	<citygml#SolidType-isWatertight> a sh:PropertyShape ; sh:hasValue true ; sh:path <valid#isWatertight> .
:tooFewPolygons rdf:type owl:DatatypeProperty ; rdfs:subPropertyOf :shellProperties . … <citygml#SolidType> rdf:type owl:Class ; rdfs:subClassOf <citygmll#GeometricPrimitiveType> , [rdf:type owl:Restriction ; owl:onProperty :tooFewPolygons ; owl:hasValue "false"^^xsd:boolean] ,	<citygml#SolidType-tooFewPolygons> a sh:PropertyShape ; sh:hasValue false ; sh:path <valid#tooFewPolygons> .

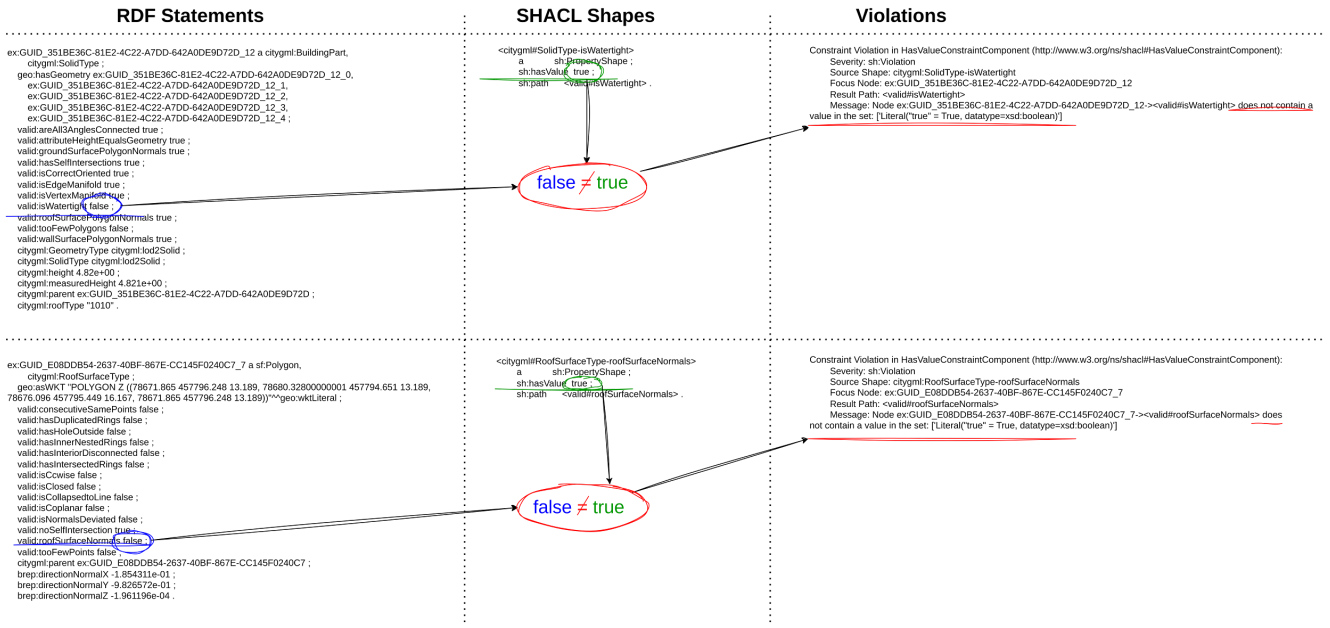


Figure 6. Samples from validation processes carried out using pySHACL engine

2.5. Usage of the pipeline via the web interface

The web interface of the application is developed using Python's Flask library [54] to manage web services, alongside core web technologies such as HTML (HyperText Markup Language) and JavaScript (Figure 7). The pipeline starts by storing the data online. Users begin by selecting a CityJSON file from their local file system and uploading it to the interface. Once uploaded, the user selects an ontology to determine the validation criteria. At present, the only available ontology is the validation ontology described earlier. After these initial steps, the data and selected ontology are transmitted to the server by clicking the "Process Data" button. On the server side, a SHACL graph for the validation ontology and an RDF graph of the CityJSON data are generated. The validation process is then executed, and the resulting validation report is stored with a URL in the "validity" field of the RDF data. This report is also sent back to the client, enabling them to download and review it.

The web interface offers tools to visually examine both invalid city objects and the complete city model. Once CityJSON data is uploaded, users can view the entire city model by selecting the “View Full Model” button located on the left side of the interface. After the validation process, a list of invalid Building and BuildingPart object URIs will appear on the right panel. Users can investigate the missing or damaged triangles causing the geometric issues by clicking on the respective object URIs. Simultaneously, clicking an object URI highlights that object with a red color within the full model view. The 3D rendering of these objects is powered by the Three.js JavaScript library [55].

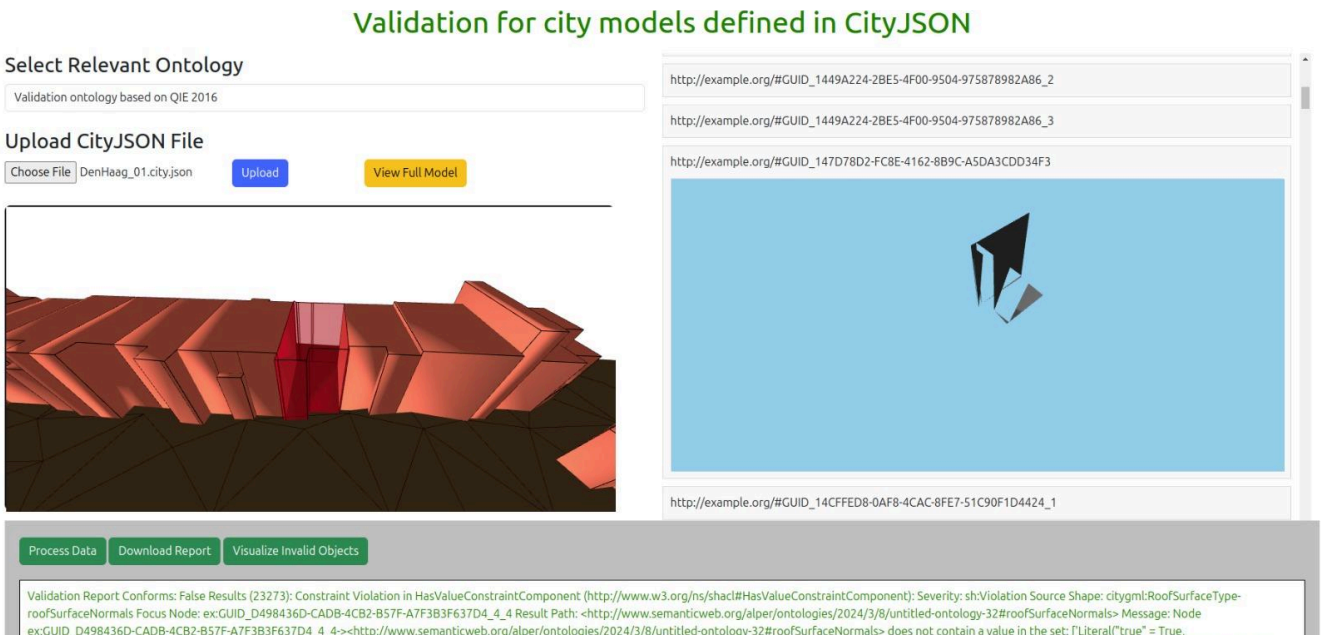


Figure 7. The user interface of the application

3. Results and Discussion

The described pipeline indicates that an interoperable validation process is possible for both geometrical and semantical aspects by the virtue of using the KG approach. The data begins to be stored online with the validation ontology and the report after the upload and the validation processes. By the automated real-time creation of the shapes graph for the data, the data brings its own validation rules to the client by preventing the client from planning a rule set. By storing the validation report with a time stamp in a node of the data graph, possible new clients of the same data or data providers are also be informed about the validity of the data. Table 4 shows the server configuration, runtimes, object counts, and file sizes for two LoD2 (Level of Details-2) CityJSON test models with no texture, Rotterdam, Vienna and Den Haag, which are published openly from [56]. Table 5. includes percentages of affected Building or BuildingPart objects for the validation violations, or invalidities, detected by the developed solution which test models have.

Considering the file sizes and the runtimes, the product application can be plugged into real-world data validation routines of concerned institutions with more powerful server configurations and user authentications. Regarding the runtime limits of popular cloud computing services, runtime results for both models are acceptable without implementing any acceleration technique if cloud systems were preferred instead of physical servers [57]. However, real-world scenarios can require larger models with more objects. In such a situation, further clustering and acceleration techniques with higher server configurations can be considered.

Before the examination of the validation results in Table 5, it should be considered that the SHACL results are not totally human-friendly for reading, especially for people who are not closely familiar with such a mechanism. Hence, the results are converted into a more readable form in the table for better understanding. The granularity level was a crucial decision for a better understanding. For urban applications, usually we concern about solid level validity, or city object level, because the ring and polygon level invalidities are usually the atomic reasons of solid level invalidities. For example, the most concerned validity aspect will likely be the

watertightness for a flood simulation, and connection problems between triangles or false holes that came from wrong-oriented rings are possible reasons of the watertightness. Based on this, we summarized the percentages of geometric invalidities in solid level granularity rather than thousands of polygon level invalidities. Besides, as a novel aspect of our study, we also share the semantic invalidities, or semantically wrong defined surfaces, in polygonal surfaces but in polygon level. The interface of the solution provides validation results as both the SHACL engine-generated form and also more human-readable with individual or error grouped form as well.

The table includes the semantic invalidities that cannot be identified by the currently in-use validation software. The most notable aspect of this functionality is that both semantic and geometric invalidities can be revealed and stored in connection with associated individuals online. Beyond the table, the most frequently encountered semantic invalidity across all models is the misdefinition of ground and roof surfaces. These errors are detected by examining the orientations of surface normals and comparing them to directional restrictions in the validation ontology by the SHACL engine. Furthermore, the geometric invalidity ratio can be seen as more broadly different from that of actual validation software. This difference is related to the chosen snap tolerance parameter for the solution. For example, a snap tolerance of 0.01 is set as the default in val3dity, and the ratio of invalid objects changes according to this tolerance value, which is similarly applicable in our solution. The snap tolerance is an inevitable parameter to select in such studies, as it directly impacts the consensus between different software. For this reason, the dynamic selection of this parameter based on the inherent nature of the input data, such as the reference system of the model or the coordinate precision of the data collection technique, could be a topic for further exploration experiments.

The developed solution stands out from its peers in terms of comprehensiveness and interoperability. As detailed in Section 2, the most notable studies in the literature lack a standard to reach consensus, provide geometrical and semantical validation, and provide reusable validation plans and reports. Based on this, our solution introduces a standardised framework that addresses these gaps, ensuring consistent validation processes and improved interoperability across systems.

The most obvious limitation of this study lies in its confinement to the CityGML Building schema and LoD2 models. Both the validation ontology and the RDF converter were specifically designed for LoD2 buildings. To expand the application's scope to encompass other application schemas within the CityGML data model and higher LoDs, it would be necessary to extend the current validation ontology and enhance the RDF converter to align with this updated framework. However, considering the application's ability to address gaps in the existing literature and the predominant focus of most studies on LoD2 buildings, this limitation does not undermine the novelty of the application. Instead, it establishes a foundation for future research and advancements in the field.

Table 4. Observables of application runtime

Server Configuration	**City Model**	**Objects**	**Size (KBs)**	**Size (RDF; KBs)**	**Time (mins)**
CPU: Intel Core i7-7700HQ 2.80 GHz **RAM:** 32 GB DDR4	Den Haag	2498	16251	22673	10
	Vienna	1322	5504	24905	13
	Rotterdam	853	5769	9737	5

Table 5. Detected invalidities for each test model (T: Total instances, V: Violations, P: Percentage)

Invalidity Type	Invalidity Percantages per Test Model								
	Den Haag			Vienna			Rotterdam		
	T	V	P	T	V	P	T	V	P
Solid Level Geometric Invalidities	1990	3	0.15	2204	456	20.69	853	807	94.61
Semantic Surface Invalidities	16209	44	0.27	43260	203	0.47	15482	1051	6.79

4. Conclusions

This study aims to prove the ability to implement the KG approach and SHACL-based validation on 3DCMs by introducing an interoperable web tool. The mechanism indicated above enables 3DCM KGs to explain themselves by carrying their own definitions as ontologies and to inform the data-sharing parties about which constraint set the data is validated against or which set the data should be validated against. While a wide variety of institutions are putting efforts into ensuring standardisation and settling interoperable workflows, we believe that such a study can be a notable milestone, especially in the domain of 3DCMs. The application can be downloaded and compiled from [58].

Besides the virtues discussed earlier, thanks to making the 3DCM data alive on the web as a KG, the study workflow can also be used for time-dynamic city data operations in collaborative information systems, which we will frequently encounter soon. If we simply take a look at the nature of the KGs, it can be easily seen that every validation activity or every partial edit on a 3DCM KG can be stored in separate nodes with time stamps and editor, or accessor, information to inform shared parties about the history of the data.

It is also noticeable that the usability of the developed mechanism is not restricted to the CityGML data model or the 3DCM domain. In other words, the customisation of the mechanism for the CityGML data model is handled using relevant open ontologies and developing a CityGML(or CityJSON)-to-RDF converter. Based on this, the extension of the use of this mechanism to other domains or research fields can be done using domain-specific new ontologies and developing conversions for the relevant format-to-KG. In the future, where it is inevitable that all structured or unstructured data will come together in a unified information source, just as a Guide to the Galaxy, making the multi-domain worldwide data interlinked and alive-on-web with such a mechanism will play a crucial role in the decision-making and solution-producing processes for experts in all fields. Considering the extensive use of geospatial data over all other types of data on the web, it is important to carry out and disseminate such studies in the GIS field for accomplishing the unification path stated above.

Our future plans include the healing of the 3D objects that are detected as invalid using the workflow explained above. The next nearest move is to implement overlap and intersection tests for neighbour city objects like BuildingParts, Buildings, or other objects that are defined in other thematic classes except the Building schema. Due to the current process of only checking the geometric validity of individual objects separately, the validation report does not include any violations of inter-object geometric invalidities. The other purpose of this plan is to create an end-to-end pipeline that accomplishes both the validation and the healing processes on the

same 3D city KG to keep the model up-to-date and valid with minimum user interruption. The literature has various studies and software to handle the healing process on 3D objects but regarding our preliminary inspections, it can be seen that there is still no matured and fully-automatized method. After completing a mature validation and healing pipeline, one of our other next steps will also be the visualisation of the invalid and healed geometries together to make the pipeline a comprehensive framework for 3DCM quality management.

References

1. Vopham, T., Hart, J. E., Laden, F., & Chiang, Y.-Y.. (2018). Emerging trends in geospatial artificial intelligence (geoAI): potential applications for environmental epidemiology. Environmental Health, 17(1). https://doi.org/10.1186/s12940-018-0386-x
2. Burns, R. and Thatcher, J. (2014). Guest editorial: what's so big about big data? finding the spaces and perils of big data. GeoJournal, 80(4), 445-448. https://doi.org/10.1007/s10708-014-9600-8
3. Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., & Çöltekin, A. (2015). Applications of 3D City Models: State of the Art Review. ISPRS International Journal of Geo-information, 4(4), 2842–2889. https://doi.org/10.3390/ijgi4042842
4. Krämer, M., Haist, J., & Reitz, T. (2007). Methods for Spatial Data Quality of 3D City Models. In Eurographics Italian chapter conference (pp. 167-172).
5. Gröger, G., Kolbe, T. H., Czerwinski, A., & Nagel, C. (2012). OpenGIS® city geography markup language (CityGML) encoding standard. Version: 1.0. 0, OGC 08-007r1.
6. ISO, T. (2003). 211. ISO 19107 Geographic information–Spatial schema.
7. Biljecki, F., Ledoux, H., Du, X., Stoter, J., Soon, K. H., & Khoo, V. H. S.. (2016). The Most Common Geometric And Semantic Errors In Citygml Datasets. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, IV-2/W1, 13–22. https://doi.org/10.5194/isprs-annals-iv-2-w1-13-2016
8. Ledoux, H., & Wagner, D. (2016). OGC® CityGML Quality Interoperability Experiment.
9. Ledoux, H., & Meijers, M.. (2011). Topologically consistent 3D city models obtained by extrusion. International Journal of Geographical Information Science, 25(4), 557–574. https://doi.org/10.1080/13658811003623277
10. Ledoux, H. (2013). On the Validation of Solids Represented with the International Standards for Geographic Information. Computer-aided Civil and Infrastructure Engineering, 28(9), 693–706. https://doi.org/10.1111/mice.12043
11. Alam, N., Wagner, D., Wewetzer, M., Von Falkenhausen, J., Coors, V., & Pries, M.. (2013). Towards Automatic Validation And Healing Of Citygml Models For Geometric And Semantic Consistency. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, II-2/W1, 1–6. https://doi.org/10.5194/isprsannals-ii-2-w1-1-2013
12. Wagner, D., Wewetzer, M., Bogdahn, J., Alam, N., Pries, M., & Coors, V.. (2013). Geometric-Semantical Consistency Validation of CityGML Models. In Lecture Notes in Geoinformation and Cartography (pp. 171–192). Lecture Notes in Geoinformation and Cartography. https://doi.org/10.1007/978-3-642-29793-9_10
13. Singla, J. G., & Trivedi, S.. (2022). 3D building reconstruction and validation using high-resolution stereo data. Current Science, 122(8), 900. https://doi.org/10.18520/cs/v122/i8/900-906
14. Gabara, G., & Sawicki, P.. (2021). Quality Evaluation Of 3d Building Models Based On Low-Altitude Imagery And Airborne Laser Scanning Point Clouds. The International Archives of the Photogrammetry,

Remote Sensing and Spatial Information Sciences, XLIII-B2-2021, 345–352. https://doi.org/10.5194/isprs-archives-xliii-b2-2021-345-2021

15. Dukai, B., Peters, R., Vitalis, S., Van Liempt, J., & Stoter, J.. (2021). Quality Assessment Of A Nationwide Data Set Containing Automatically Reconstructed 3d Building Models. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLVI-4/W4-2021, 17–24. https://doi.org/10.5194/isprs-archives-xlvi-4-w4-2021-17-2021
16. Ledoux, H.. (2018). val3dity: validation of 3D GIS primitives according to the international standards. Open Geospatial Data, Software and Standards, 3(1). https://doi.org/10.1186/s40965-018-0043-x
17. Chadzynski, A., Li, S., Grišiūtė, A., Chua, J., Hofmeister, M., Yan, J., ... & Kraft, M. (2023). Semantic 3D city interfaces—Intelligent interactions on dynamic geospatial knowledge graphs. Data-Centric Engineering, 4, e20.
18. Pauwels, P., van den Bersselaar, E., & Verhelst, L. (2024). Validation of technical requirements for a BIM model using semantic web technologies. Advanced Engineering Informatics, 60, 102426.
19. Coors, V., Betz, M., & Duminil, E.. (2020). A Concept of Quality Management of 3D City Models Supporting Application-Specific Requirements. PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science, 88(1), 3–14. https://doi.org/10.1007/s41064-020-00094-0
20. https://transfer.hft-stuttgart.de/pages/citydoctor/citydoctorhomepage/en/ Last access: 05/03/2025
21. Betz, M., & Coors, V.. (2021). An Application Domain Extension For Storing Validation Results Of Citygml Structures. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, VIII-4/W1-2021, 11–16. https://doi.org/10.5194/isprs-annals-viii-4-w1-2021-11-2021
22. Lassila, O., Hendler, J., & Berners-Lee, T. (2001). The semantic web. Scientific American, 284(5), 34-43.
23. Cömert, Ç., Ulutaş, D., Akıncı, H., & Kara, G. (2010). Semantic web services for implementing national spatial data infrastructures. Scientific research and essays, 5(7), 685-692.
24. World Wide Web Consortium. (2014). RDF 1.1 Primer.
25. Pauwels, P., Zhang, S., & Lee, Y. C. (2017). Semantic web technologies in AEC industry: A literature overview. Automation in construction, 73, 145-165.
26. Nardi, D., & Brachman, R. J. (2003). An introduction to description logics. Description logic handbook, 1, 40.
27. Kostovska, A., Vermetten, D., Doerr, C., Džeroski, S., Panov, P., & Eftimov, T. (2021, July). Option: optimization algorithm benchmarking ontology. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp. 239-240).
28. Hitzler, P., Krotzsch, M., & Rudolph, S. (2009). Foundations of semantic web technologies. Chapman and Hall/CRC.
29. Vinasco-Alvarez, D., Samuel, J. S., Servigne, S., & Gesquière, G. (2020). From citygml to owl (Doctoral dissertation, LIRIS UMR 5205).
30. Van den Brink, L., Janssen, P., & Quak, W. (2013). From geo-data to linked data: automated transformation from GML to RDF. Linked Open Data-Pilot Linked Open Data Nederland.
31. Werbrouck, J., Senthilvel, M., Beetz, J., & Pauwels, P. (2019, September). A checking approach for distributed building data. In 31st forum bauinformatik, Berlin: Universitätsverlag der TU Berlin (pp. 173-81).
32. Hagedorn, P., & König, M. (2020, July). Rule-based semantic validation for standardized linked building models. In International Conference on Computing in Civil and Building Engineering (pp. 772-787). Cham: Springer International Publishing.

33. Hagedorn, P., Pauwels, P., & König, M. (2023). Semantic rule checking of cross-domain building data in information containers for linked document delivery using the shapes constraint language. Automation in Construction, 156, 105106.
34. Knublauch, H., & Kontokostas, D. (2017). Shapes constraint language (SHACL). W3C Candidate Recommendation, 11(8), 1.
35. https://www.ontotext.com/knowledgehub/fundamentals/what-is-shacl/ Last access: 05/03/2025
36. Cimmino, A., Fernández-Izquierdo, A., & García-Castro, R.. (2020). Astrea: Automatic Generation of SHACL Shapes from Ontologies. In Lecture Notes in Computer Science (pp. 497–513). Lecture Notes in Computer Science. https://doi.org/10.1007/978-3-030-49461-2_29
37. Pareti, P., & Konstantinidis, G. (2021). A review of SHACL: from data validation to schema reasoning for RDF graphs. Reasoning Web International Summer School, 115-144.
38. Zeshan, F., Ahmad, A., Babar, M. I., Hamid, M., Hajjej, F., & Ashraf, M.. (2023). An IoT-Enabled Ontology-Based Intelligent Healthcare Framework for Remote Patient Monitoring. IEEE Access, 11, 133947–133966. https://doi.org/10.1109/access.2023.3332708
39. Vinasco-Alvarez, D., Samuel, J., Servigne, S., & Gesquière, G.. (2021). Towards Limiting Semantic Data Loss In 4d Urban Data Semantic Graph Generation. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, VIII-4/W2-2021, 37–44. https://doi.org/10.5194/isprs-annals-viii-4-w2-2021-37-2021
40. http://www.opengis.net/ont/geosparql Last access: 05/03/2025
41. Chadzynski, A., Krdzavac, N., Farazi, F., Lim, M. Q., Li, S., Grisiute, A., ... & Kraft, M. (2021). Semantic 3D City Database—An enabler for a dynamic geospatial knowledge graph. Energy and AI, 6, 100106.
42. Quek, H. Y., Hofmeister, M., Rihm, S. D., Yan, J., Lai, J., Brownbridge, G., ... & Kraft, M. (2024). Dynamic knowledge graph applications for augmented built environments through "The World Avatar". Journal of Building Engineering, 91, 109507.
43. http://www.opengis.net/ont/sf Last access: 05/03/2025
44. Musen, M. A. (2015). The protégé project: a look back and a look forward. AI matters, 1(4), 4-12.
45. Ledoux, H., Arroyo Ohori, K., Kumar, K., Dukai, B., Labetski, A., & Vitalis, S.. (2019). CityJSON: a compact and easy-to-use encoding of the CityGML data model. Open Geospatial Data, Software and Standards, 4(1). https://doi.org/10.1186/s40965-019-0064-0
46. https://cityjson.github.io/cjio/index.html Last access: 05/03/2025
47. https://rdfpandas.readthedocs.io/en/latest/ Last access: 05/03/2025
48. Hartig, O. (2014). Reconciliation of RDF* and property graphs. arXiv preprint arXiv:1409.3288.
49. Akın, A. T., & Cömert, Ç.. (2024). "Cityjson2rdf" A Converter For Producing 3d City Knowledge Graphs. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLVIII-4/W9-2024, 15–19. https://doi.org/10.5194/isprs-archives-xlviii-4-w9-2024-15-2024
50. Zhou, Q. Y., Park, J., & Koltun, V. (2018). Open3D: A modern library for 3D data processing. arXiv preprint arXiv:1801.09847.
51. https://rdflib.readthedocs.io/en/stable/ Last access: 05/03/2025
52. https://github.com/sparna-git/owl2shacl Last access: 05/03/2025
53. https://github.com/RDFLib/pySHACL Last access: 05/03/2025
54. https://flask.palletsprojects.com/en/stable/ Last access: 05/03/2025
55. https://threejs.org/ Last access: 05/03/2025
56. https://www.cityjson.org/datasets/ Last access: 05/03/2025

57. Akin, A. T., Usta, Z., & Cömert, Ç.. (2024). Elaborating and performing optimization approaches for web-based 3D spatial analysis of 3D city models. Journal of Spatial Science, 69(4), 1225–1239. https://doi.org/10.1080/14498596.2024.2377564
58. https://github.com/alpertungakin/3DCMQuality/tree/main/Valid_app_gui Last access: 05/03/2025