The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume L-4/W3-2026
10th International Conference on Smart Data and Smart Cities (SDSC), 29 September–1 October 2026, Sofia, Bulgaria

A Data Model for Geometric, Semantic and Spatial Topology Validation of CityGML Datasets

Luna Riegel[1], Matthias Betz[1], Volker Coors[1]

[1] Stuttgart Technical University of Applied Sciences, Faculty of Geomatics, Computer Science and Mathematics, 70174 Stuttgart, Germany - (luna.riegel, matthias.betz, volker.coors)@hft-stuttgart.de

Keywords: CityGML, Validation, Software Architecture, Data processing, GIS

Abstract

3D city models are gaining importance across applications such as urban analytics and digital twins, where geometric and semantic data quality directly determine downstream usability. While tools for validating the geometric and semantic consistency of individual CityGML features already exist, support for validating the spatial topology between distinct features remains absent from current validation tools.
This paper presents ongoing research extending the data model of CityDoctor2 (CD2), an open-source Java library for CityGML validation and repair, to enable spatial topology validation at city-model scale. We describe CD2's existing data model and detail the architectural adaptations required to support topology checks: replacing the original streaming-based parser with a persistent, database-backed storage layer complemented by an in-memory cache, enabling spatial indexing, multi-threaded validation, and candidate-pair queries via bounding-box predicates. Performance evaluation on a CityGML test dataset shows the new architecture incurs a considerable increase in runtime when compared to the previous streaming approach, in exchange for the ability to perform spatial topology validation on datasets too large to hold entirely in memory.

1. Introduction

Germany's national digitalisation strategy includes a phased plan for smart cities (German Federal Ministry of Housing, Urban Development and Construction., 2024). A key component of this plan is the creation of so-called urban digital twins. Urban digital twins are digital 3D city models, which are enriched by connections to other systems, platforms, databases and datasets. They are thus designed to act as a consolidating interface for the aforementioned services. While the primary purpose of a digital 3D city model (hereafter: city model) in an urban digital twin is visualization, it also serves as a valuable dataset for analytical applications. One example is the Simstadt platform (Nouvel et al., 2015), which, among other functions, simulates urban energy and heating demand based on city models.

To enable the exchange of semantically enriched 3D data across entities, a common data model is required. CityGML (Devys et al., 2021) is a widely used open standard for storage and exchange of city models. As of Version 3.0 CityGML also includes a conceptual model, defining the abstract semantic classes and their structure, allowing the usage of other encodings, such as CityJSON (Ledoux et al., 2019).

CityGML models are commonly generated from existing spatial datasets, such as 2D cadastral building contours and aerial laser scans. However, these generation processes can produce XML-syntactically correct models containing geometrical errors. XML validation tools are thus unable to detect these errors, as the model is schema compliant. Critically, these geometric errors do not necessarily produce visual artifacts, allowing erroneous models to appear valid. While this may be acceptable for visualization, it can lead to significant issues in analytical applications. Assessing the suitability of a model for a given application therefore requires evaluation of its geometric consistency. Due to the scale of city models, manual processes for evaluation become unfeasible, thus requiring automated methods (Coors et al., 2020).

CityDoctor2 (CD2) is a free, open-source software Java library for validation and repair of the geometric and semantic consistency of CityGML models. This paper presents the data model underlying CD2, tailored for the geometric computation of CityGML city models while preserving semantic and structural information, and the current research efforts focusing on extending and adapting the data model for implementing the validation and repair of the spatial topology of city models.

2. Definition of Terms

CityObject (CO): The abstract core class of the CityGML conceptual model, as all classes containing geometrical information extend from it. In this work `CityObject` is used as a stand-in term for any CityGML class that contains `Geometry` objects.

Feature: Used in this work as shorthand for CityGML's TopLevelFeature. TopLevelFeature is a stereotype of the conceptual model denoting distinct classes of entities within the city model, such as Buildings or Bridges.

Geometry: To avoid confusion, `Geometry` will be capitalized and in `teletype` font when referring to the CityGML class.

Requirement: A rule, condition, or constraint that a dataset must fulfill. Requirements are derived either from the CityGML standard or from application-/implementation-specific restrictions. A requirement can have parameters (e.g., a floating-point equality tolerance) and can depend on other requirements being met, forming a dependency chain.

Validation plan: A collection of requirements, together with their parameters, that a dataset must satisfy. The validation plan is the formal specification of a dataset's target state and is central to the validation process.

Error: A recorded violation of a requirement, documenting its cause and the objects or entries involved. Each requirement is associated with one or more error types.

Check: An algorithm that determines whether an entry of a dataset satisfies a specific requirement. Each check is associated with

an error type used to classify detected errors and produces a check result recording its findings.

Validation report: The output document of a validation run. It consists of the validation plan, metadata (e.g., datetime, filename, validation tool used), check results, and detected errors. Insights derived from validation reports can subsequently be used for refining data generation processes or supporting repair of identified errors within a dataset.

Geometric validation: The validation of geometric primitives. Its objective is to determine whether a geometry elements conform to the structural requirements of their geometry type defined by a standard.

Semantic validation: The validation of attribute values, code lists, classification consistency, and other domain-specific business rules.

Geometric topology: Qualitative information and intrinsic properties of geometries which are not changing under continuous deformation, such as adjacency, connectivity or closeness(Carlsson, 2009).

Spatial topology: Spatial relationships and properties between different geometries, such as intersection, connection or adjacency.

Topology validation: The validation of the spatial topology of a city model. For example, detecting overlapping buildings, violations of clearance distances or intersection angles between utility-line objects. Topological requirements are not derived directly from the CityGML specification, but from application requirements, regulations, and assumptions about plausible real-world spatial configurations.

3. Related Work

Validation of city models has already been a focus of many research efforts. Wagner et al. (Wagner et al., 2013) developed systematic rules for geometric-semantical consistency validation and demonstrated that CityGML models can produce erroneous results even while remaining standard-compliant, motivating creation of rule-based validation beyond schema checks. Building on this, Alam et al. (Alam et al., 2014) proposed a workflow for detecting both geometrical and semantic errors in CityGML models as a precursor to automated repair. Ledoux et al. (Ledoux, 2013) surveyed the geometric and topological quality of real-world CityGML datasets at scale, validating Solid and MultiSurface primitives together with semantic correctness of building boundary surfaces, and found that geometric and semantic errors are widespread, even in production datasets. Val3dity is a free open source library for geometry validation according to the ISO 19107 standard (Ledoux, 2018) .

Borrmann and Rank (Borrmann and Rank, 2009) defined 3D topological predicates based on the 9-intersection model, and implemented these as part of a spatial query language for building models. Salleh et al. (Salleh et al., 2021) analyzed different spatial databases for their support of 3D topological data structures, and in earlier work (Salleh et al., 2022) examined 3D topological relationships for representing legally binding cadastral parcels, addressing a two-dimensional analogue of the problem: cadastral topology rules require that parcels be gapless and overlap-free. Most directly relevant is Salleh et al. (Salleh et al., 2020), who validated a Compact Abstract Cell Complex topological data structure for LOD2 CityGML buildings, demonstrating 3D topological validation within individual building geometries.

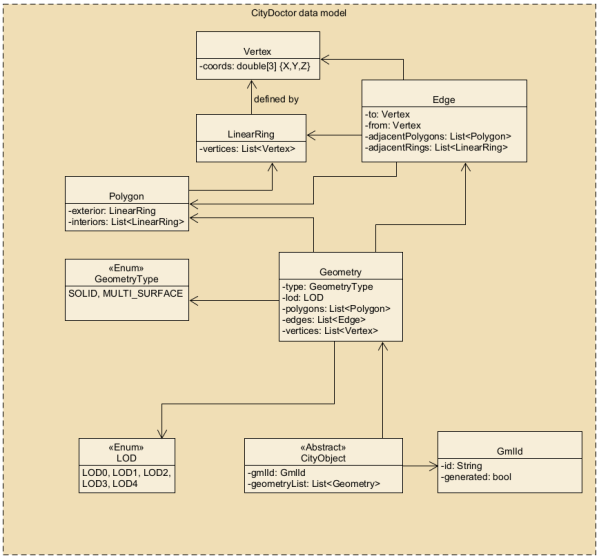


Figure 1. CityDoctor's data structure

However, there exists a gap in the research and tooling regarding validation of spatial topology rules. The State of the art is the validation of geometric and semantic consistency of single top-level-features. While 3D spatial-topology functions are being partially offered by some tools, such as spatial databases, a validation of the consistency of spatial topological relationships between Features is not being offered directly by any of these tools. This paper introduces the work of a current research effort aimed at implementing spatial topological validation of urban digital twins in CityGML.

4. System Architecture

4.1 Data Structure

Figure 1 illustrates the implementation of CityGML's conceptual model in `CityObjects` CD2's data structure. CityGML's `CityObject` class is the core of the conceptual model, as it is the superclass of all space and boundary stereotypes.

GML's encodes geometries as polygonal primitives, containing flattened lists of the vertex coordinate triplets. CD2 maps the vertiex triplets to 3D vector `Vertex` objects and de-duplicates the vertices of a `Feature` after parsing. De-duplication is performed using a virtual equidistant grid: vertices are first snapped to the nearest grid position and then compared using a distance tolerance to determine equivalence. This process is restrained to each `Geometry` Object, thus it does not extend across `CityObjects`, even if both belong to the same Feature. This simplifies many geometric calculations, as this enables the creation of auxiliary mappings for easier traversal and querying of geometries.

4.2 Tree Structure and Traversal

The validation process is comprised of several stages, such as parsing-cleanup, pre-checking preparation, checking, post-checking cleanup, error-statistics generation and validation report creation. During each of these stages, repeated traversals over the Features are required. To support this, CD2's data model constructs a tree-like data structure from the CityGML model (see Figure 2), enabling the use of the Visitor pattern. This generalizes the traversal

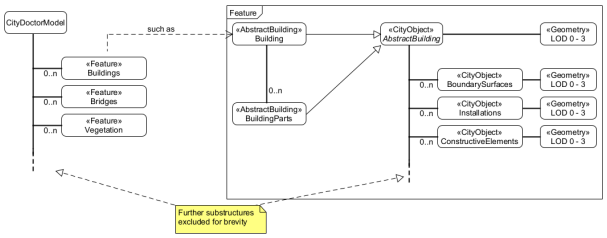


Figure 2. Visualization of CD2's tree-like datastructure

over Features, reducing the risk of errors and improves the maintainabilty and extendsibility of the data model. Moreover, the double-dispatch mechanism of Visitors allows tasks to retrieve a Feature's specific `CityObject` subclass without relying on casting or `instanceof` checks.

Moreover, as part of the pre-check preparation, auxiliary mappings are constructed on all `Geometries` within a `Feature`. These mappings explicitly encode geometrical topology information, such as adjacency and edge connections. As these mappings are computationally cheap and quick to create, but introduce noticeable memory overhead, they are only retained while a Feature is being validated.

4.3 Geometry Instancing (`ImplicitGeometry`)

CityGML offers mechanisms for `Geometry` instancing in the form of `ImplicitGeometry` objects. An `ImplicitGeometry` is constructed from a prototype geometry and a 4x4 transformation matrix describing the translation, rotation, and scale of the instanced geometry. Prototype geometries can be bound in two ways: as `LibraryObjects` or as `RelativeGeometry` objects. A `RelativeGeometry` references a `Geometry` contained within the same CityGML file, while a `LibraryObject` references a path to an external CityGML file.

CD2 resolves implicit geometries as follows. For a `RelativeGeometry`, CD2 first checks whether the prototype geometry has already been parsed. If so, a copy of the prototype is created, transformed and added to the `CityObject`; otherwise, an empty `Geometry` is inserted and registered in an indexing map for deferred resolution. Once the referenced `RelativeGeometry` has been processed by the parser, all deferred references are resolved and the placeholder `Geometries` are replaced. Resolving a `LibraryObject` is simpler: the first time it is referenced, the parser starts a sub-parser for retrieving the prototype geometry from the referenced file. Then a copy of the prototype is transformed and added to the `CityObject`. Subsequent `LibraryObjects` referencing the same file then skip the sub-parsing step.

4.4 XLink Reference Resolution

Although GML supports referencing through XLinks, the CityGML specification treats Features as self-contained entities. Consequently, use of cross-feature geometric references therefore introduces dependencies between otherwise independent objects. These dependencies complicate the extraction, partitioning, and exchange of subsets of a CityModel, since the complete geometry of an object may only be recoverable through external references. They also increase parsing complexity, as referenced geometries may not yet have been encountered during sequential processing.

CD2 resolves such references in two stages. At the start of parsing each Feature, an empty PolygonMap is created and used to resolve inner-Feature Hrefs while the Feature Object is being constructed; the map is cleared once construction completes. Any Href that remains unresolved against this map is treated as a cross-feature reference, barring one exception: if the Feature being resolved holds its own copy of the referenced polygon, and the Href points to the duplicate of the polygon in the another Feature, the reference is not flagged as cross-feature.

References classified as cross-feature are recorded in a global index, keyed by the referenced polygon identifier and storing the identifiers of all Features that reference it, and are resolved only once the entire document has been parsed. At that point, each entry is checked against the set of all parsed polygons. If the referenced polygon is found elsewhere in the document, the reference is resolved, but the occurrence is logged as a warning, together with the identifiers of the referencing Features, since it indicates a genuine cross-feature reference, a pattern that violates CD2's assumption of geometrically self-contained CityObjects. If no polygon with the referenced identifier can be found anywhere in the CityModel, the reference remains unresolved and is reported as a data error, since it points to geometry that does not exist within the document.

4.5 Graphical User Interface

As CD2's data model uses n-gonal polygons it is not suitable to be used directly for visualization, as rendering requires triangulated meshes. Thus, visualization requires a preprocessing step in the form of triangular tesselation. CD2 utilizes the Java OpenGL tesselator and collects the resulting triangles of all tessellated polygons in one mesh.

4.6 Healing

The healing process has a similar structure as validation. The healer uses the Validation Plan to collect all respective applicable `HealingMethod` visitors. The healer then begins traversing the model till it encounters a Feature containing Errors. The Healer then begins the healing loop, iterating over the list of `HealingMethods` Visitors and applying them to the Errors, till either a `HealingMethod` changed a geometry, or all Methods have been attempted to apply without changes. In the latter case, the Feature is not healable and thus, the Healer stops the loop and continues traversing the Model. If a Method changed a geometry the Healer stops iterating and checks the Feature again. The Healer repeats the healing loop until either A: the Feature does not contain Errors anymore, B: determines that the Feature is not healable or C: Has reached the limit of Healing loop iterations (default: 200). In all cases the Healer continues traversing the model. Case C is a safety against infinite loops, as the changes of a healing method can produce new Errors, which in some cases creates an Error-loop, for example: A missing Polygon in the shell of a feature creates a non-manifold edge when inserted, causing the Healer to insert and then remove this Polygon repeatedly.

4.7 Adaptations and Extensions for Topological Validation

Current development and research efforts are focused on extending CD2 to allow the execution of spatial topology checks. These checks introduce architectural challenges to the existing data structure and validation process, as geometric and semantic validation of Features depend only on information contained within itself and is independent of the remainder of the CityModel. Validation of spatial topology, however, necessitates repeated access to all other Features within the city model, as it analyzes the relationship between a Feature and the remainder of the CityModel. As operational CityGML datasets can reach filesizes of several Gigabytes, older versions of CD2 processed CityGML files by creating a data stream to sequentially validate the Features, circumventing the filesize limitation that would be imposed by retaining

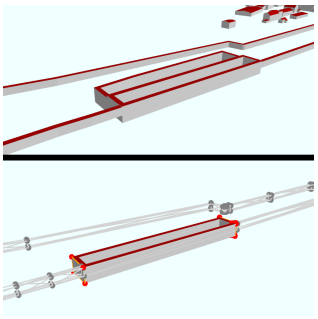


Figure 3. Example of a Feature collision

the entire model in memory. However, this method makes it impossible to access other Features at random times. While it would be possible to create another input stream from the CityGML file, it would introduce significant redundant computational overhead, as streams have no indexing, requiring linearly processing the entire file.

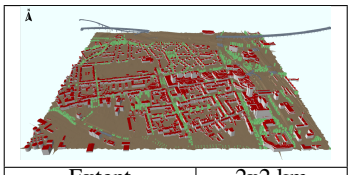
4.7.1 Persistent Model Storage To address the requirement of accessing `Features` at arbitrary times, a persistant storage for `Features` is necessary. CD2 employs a local, file-based geospatial database as an implementation of persistent storage. After a CityObject has been parsed it is then serialized into the database. This database is only intended as an interim storage, and is only maintained for the lifetime of a validation session, and is removed as part of the shutdown process. However, while this allows accessing Features at arbirtrary times, the database slows down the overall processing speed, as (de-)serialization of Objects from a drive is significantly slower compared to accessing Objects that are held in the systems RAM. To adress this issue, the database is complemented by an in-memory cache. This allows recently accessed CityObjects to remain temporarily available in memory, reducing access times and database congestion. While the cache is not useful during geometric and semantic validation, as models containing more Features than the cache size-limit will cause it to repeatedly ”roll over” during the processing stages, it will aid in accelerating the spatial topology validation.

4.7.2 Spatial Queries Spatial topology checks involve complex and thus computationally expensive geometric calculations. As the number of distinct unordered pairs of Features within a CityModel grows exponentially with the number of Features, naively analyzing all pairings quickly grows to an unfeasible scale. Thus, to manage this large volume of data, cheap methods for quickly filtering out Features warranting the expensive calculations (”Candidates”) are required. For example, the Feature collision check (see Figure 3) can immediately exclude all Features whose bounding-box does not intersect the subject’s bounding-box, as there is no possibility of volumetric overlap.

A benefit of the database-backed storage is the access to spatial indexing and geospatial querying capabilities. As the database only supports 2D bounding-boxes, the 3D bounding-boxes of Features are flattened to 2D and used for the spatial index. While this removes some of the granularity, making the filtering less precise, the time saved by avoiding (de-serialization) makes this an acceptable trade-off.

5. Evaluation

To test the performance of the different parsing approaches a series of small tests were run. The model used for running these tests was an extracted CityGML tile from the City of Leverkusen,



Extent	2x2 km
Features	3,700
Distinct Vertices	1,430,337
Filesize	165 MB

Table 1. CityGml tile from the city model of Leverkusen (district Wiesdorf)

Version	3.17.3	3.18.3			
Operation	stream	parallel, cached	parallel, no cache	not parallel, cached	not parallel, no cache
Time (s)	36	52	54	170	464

Table 2. Runtime test results (Geometric validation only)

specifically the district of Wiesdorf (see table 1). This model was chosen as it contains a spread of different Feature-types, including a TIN-Relief DTM, and CityObjects using `Implicit-Geometry`. There were no spatial topology checks performed to be able to compare the CD2 versions. These validation routines can only be performed by the newer version.

5.1 Test setup

The latest public release of CD2 was tested against Version 3.17.3, as it was the last version using the streaming approach. CD2 sets the worker-thread pool-size for parallelization to 2 times the amount of available CPU cores, and has a cache size of 3000 Features. These values were changed to 1 to simulate the disabling of their respective mechanisms, as they were intended to always be active in actual releases of the library.

The tests were run in two stages. In the first stage only the Geometric validation was run to establish a baseline time value of the validation. In the second stage the Feature collision check was activated and the tests were run again, excluding Version 3.17.3 as it does not offer spatial topology checks.

The runtime was recorded from call to the command-line interface till termination. For the memory consumption test a high water mark was determined during runtime. For the remaining modes of operation the memory usage was recorded after the entire model was validated. The tests were ran on a system using an Intel i7-8850H CPU (6 Cores @ 2.6 GHz).

5.2 Results

For the memory consumption four values are relevant. In the older version there were two separate way to validate a CityGML file. One was by going through the file feature by feature and validating it or load the whole file into memory and then run the validation. With the new data model in place using the database it will always be ”fully” loaded into memory before any validation takes place. To increase performance a cache was added which incurs an additional memory cost as it retains objects in memory. With the cache disabled this additional memory is not needed as seen in table 4. In the older version the retaining memory consumption is obviously much higher as it retains the complete model in memory.

For the first stage of runtime testing (table 2) parallelization played a much bigger role as during the file based database queries the

Operation	parallel, cached	parallel, no cache	not parallel, cached	not parallel, no cache
Time (s)	101	158	662	1343

Table 3. Runtime test results (Geometric + Feature Collision check)

Version	3.17.3		3.18.3	
Operation	streaming	retaining	cache	no cache
Max memory (MB)	119	989	739	107

Table 4. Memory test results

program would idle and therefore can spare CPU resources for validation processes where the database query has already resolved. Therefore there is a massive difference between parallel and non parallel execution in the newer version. The older version did not support any parallelization. While the newer version was, at best, around 70% slower in the tests, the underlying changes to the system architecture are required for the implementation of spatial topology checks.

The results of the second stage of testing (table 3) show the significant acceleration that parallelization and caching have on the runtime of spatial topology checks. Similiar to the first stage of testing, parallelization had a bigger impact on the overall runtime, however, the cache still provides a noticeable acceleration of.

Using past runtime data provided by the the AdV[1] for validation of LOD-2 models of the federal states of Germany (see table 5) with CD2 (previously presented in (Betz et al., 2025)) - with the measured performance loss we can estimate that the same dataset would take 5251 (87.5 hours) minutes instead of 3089 minutes (51.5 hours) (see table 5) with the new version.

As CD2 did not yet support parallelization, only a single core of the CPU was utilized. With the provided data, the average runtime per Feature was calculated to be:

$$\frac{3089min \times 60s}{50221985} \approx 0.0037 \text{ s} = 3.7 \text{ ms}$$

	old model	new model (estimation)
Total Number of Features	50,221,985	=
Total runtime	3089 min	5251 min
CPU name	AMD EPYC 7313P	=
CPU clock speed	3 GHz, 3.7 GHz Boost	=

Table 5. Runtime data from the AdV with runtime estimation

5.3 Limitations

The limitation of the data model stems from it fundamental architecture. It is not capable to be rendered quickly by a GPU. The model is a polygonal model which has to be converted to a triangular model before it can be rendered. The delay imposed by a database query also reduces render time for not yet loaded models. For quick rendering of 3D models a datamodel like glTF (The Khronos® 3D Formats Working Group, 2021).

The data model is also not very well suited for web streaming. While it now can perform spatial queries and therefore is able to send data in chunks it is not a static file based model but requires a daemon to provide access to the database. Web streaming is also mostly used for visualization which includes the rendering aspect. For this application the 3D Tiles data model is much better suited (Open Geospatial Consortium, 2023)

[1] AdV = Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (engl: Working Committee of the Surveying Authorities of the Federal States of the Federal Republic of Germany)

6. Conclusion

While the adjustments of the CD2 data model caused the Geometric validation to be significantly slower, it is a necessary trade-off to allow the implementation of checks for the consistency of spatial topological relationships. Furthermore, the implemented in-memory cache, which was introduced to provide accelerations for the spatial topology checks, did provide a reduction of the validation runtime. However, some limitations apply, as the validation was only tested on a tile of a city model, with a Feature count close to the caches' size limit, and thus allows the majority of the Features to be retained in memory.

7. Future

7.1 Healing using Reference Model Data

A major desired goal of the research project is the integration of reference data (such as PointClouds, 3D Meshes or IFC) in the healing process. The aim is not to convert the reference data, but to establish methods for calculating a "fitting"-score of Features to the ideal state represented by the reference data. However, a simple value, such as the mean square error, is insufficient for this use-case, as LODs intentionally simplifies the model. The calculation and rating of the fitting score would allow implementation of a more dynamic and robust healing algorithm, as the healer could utilize the ratings different healing methods and ordering, and compare their effectiveness in repairing the model while retaining the accuracy of the city model's representation.

7.2 Spatial Topology checks

With the new database in place CD2 is now capable of handling much more complex spatial validation tasks that stretches beyond the individual feature validation. We are currently working on implementing more spatial topology checks which can be performed on large data sets, such as city wide pipe networks.

In addition we are working on further improving the performance by using more batching to reduce the amount of slower queries to the database.

Acknowledgements

The authors would like to thank all project partners for their continuous support and feedback throughout the development of the software. Particular thanks are extended to the participating surveying authorities, municipalities, and industrial partners whose operational use cases and datasets have significantly contributed to the evolution and validation of the framework.

The authors further acknowledge the contributions of the AdV quality assurance working groups and all institutions that participated in testing and evaluating the software within operational workflows.

CD2 was developed through a cooperative effort involving academic institutions, geospatial software companies, and surveying authorities at municipal, regional, and national levels.

Funding

This work was funded by the German Federal Ministry of Research, Technology and Space (Bundesministerium für Forschung, Technologie und Raumfahrt, BMFTR) under grant no. 13FH019KA2.

References

Alam, N., Wagner, D., Wewetzer, M., von Falkenhausen, J., Coors, V., Pries, M., 2014. *Towards Automatic Validation and Healing of CityGML Models for Geometric and Semantic Consistency*. Springer International Publishing, Cham, 77–91.

Betz, M., Riegel, A., Coors, V., 2025. Storing quality validation results in CityGML with the QualityADE. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVIII-4/W15-2025, 31–37. https://isprs-archives.copernicus.org/articles/XLVIII-4-W15-2025/31/2025/.

Borrmann, A., Rank, E., 2009. Topological Analysis of 3D Building Models Using a Spatial Query Language. *Advanced Engineering Informatics*, 23(4), 370–385.

Carlsson, G., 2009. Topology and data. *Bulletin of the American Mathematical Society*, 46(2), 255–308.

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.

Devys, Coors, Hagler, Heazel, Kolbe, Kutzner, Nagel, Roensdorf, Smyth, 2021. Ogc city geography markup language. Technical Report Version 3.0, Open Geospatial Consortium.

German Federal Ministry of Housing, Urban Development and Construction., 2024. Stufenplan smarte städte und regionen. Technical report.

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.

Ledoux, H., 2018. val3dity: validation of 3D GIS primitives according to the international standards. *Open Geospatial Data, Software and Standards*, 3(1).

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), 4.

Nouvel, R., Brassel, K.-H., Bruse, M., Duminil, E., Coors, V., Eicker, U., Robinson, D., 2015. Simstadt, a new workflow-driven urban energy simulation platform for citygml city models. *Proceedings of International Conference CISBAT 2015 Future Buildings and Districts Sustainability from Nano to Urban Scale*, LESO-PB, EPFL, 889–894.

Open Geospatial Consortium, 2023. OGC 3D Tiles Specification 1.1. OGC Community Standard 22-025r4, Open Geospatial Consortium.

Salleh, S., Ujang, U., Azri, S., 2021. 3D Topological Support in Spatial Databases: An Overview. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVI-4/W5-2021, 473–478.

Salleh, S., Ujang, U., Azri, S., 2022. Topological Relationships in R3 for 3D Cadastre. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVIII-4/W3-2022, 165–170. https://isprs-archives.copernicus.org/articles/XLVIII-4-W3-2022/165/2022/.

Salleh, S., Ujang, U., Azri, S., Choon, T. L., 2020. 3D Topological Validation of Compact Abstract Cell Complexes (CACC) Data Structure for Buildings in CityGML. *International Journal of Built Environment and Sustainability*, 7(2).

The Khronos® 3D Formats Working Group, 2021. glTF™ 2.0 Specification. Specification, The Khronos Group. Version 2.0.1.

Wagner, D., Wewetzer, M., Bogdahn, J., Alam, N., Pries, M., Coors, V., 2013. *Geometric-Semantical Consistency Validation of CityGML Models*. Springer Berlin Heidelberg, Berlin, Heidelberg, 171–192.