Storing quality validation results in CityGML with the QualityADE

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

[1] Hochschule für Technik Stuttgart, 70174 Stuttgart, Germany - (matthias.betz, alexander.riegel, volker.coors)@hft-stuttgart.de

Keywords: CityGML, ADE, Validation, Quality Management, Quality Assurance

Abstract

In this paper a concept for storing validation information in CityGML as an extension is proposed. This work is an extension of our previous work in which we improve and evaluate more of the validation process and the data storage itself(Betz and Coors, 2021) . With this model companies, states and municipalities can manage the quality of their 3D models based on CityGML more easily. The information can be further used in other application software to improve their handling of imperfect 3D models as their input. Without quality information it is almost always unknown whether an issue stems from a problem in their application or from errors in the input data itself. CityGML creators can now distribute CityGML files with the quality information contained in them providing a standardized way of storing the data alongside the 3D models. The quality management information also further improves the process of 3D data creation. As issues are detected with a proper quality management tool and stored in the data, the validation information can be used to find problems in the workflow and remove them.

1. Introduction

3D spatial models of cities or entire landscapes, henceforth both referred to as city model, have been and are going to be increasingly more popular for visualization, simulation, disaster management, AR and many more applications. Each application asserts specific requirements for the city model; therefore, a process is necessary to ensure the compliance of the model data with these requirements. The widely used standard for storing spatial data from a city model is an XML (Bray et al., 1997) based data format called CityGML (City Geography Markup Language) (Kutzner et al., 2023). However, the CityGML standard does not offer the ability to store information on the accuracy or validity of its geometric and semantic data and thus requires an extension. CityGML has a system for extensions called ADE (Application Domain Extension) (Kutzner et al., 2023) that can be used to add new object types or properties to CityGML's data structure. In this work, we define an ADE for storing validation results in CityGML. This ADE allows applications, for example, simulations such as SimStadt (Nouvel et al., 2015), to produce more reliable results because the quality and validity of the city model are known.

We already developed the concept in (Betz and Coors, 2021) and have improved on the data structures with our experience in using it. This has lead to improvements regarding usage in databases as well as the collection of more evaluation data for the validation and the QualityADE.

This ADE is also useful for CityGML manufacturers as it enables them to deliver validated city models with the validation information contained in the model itself. Additionally, the validation results can be used to increase the quality of the created models by improving the process of the geometry generation or by fixing the geometries in a post processing step. The quality management process and improvement is shown in Figure 1.

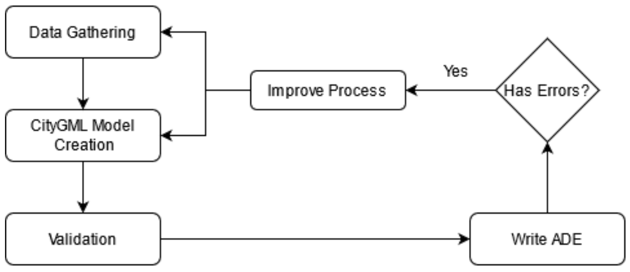


Figure 1. Quality Management Process

2. State of the Art

There are many papers already describing algorithms and processes for validating city models, see (Coors et al., 2020), (Ledoux and Wagner, 2016),(Biljecki et al., 2016).

(Ledoux, 2018) show a validation workflow for CityJSON as well as CityGML. The error reporting of val3dity is external as well, meaning the meta data is stored outside the dataset as html. The software also focuses on geometric errors and does not support semantic validation.

To be able to work with massive amounts of data, meta data management becomes a necessity (Kavisha, 2020). One of the possibilities for storing meta data is creating and using an ADE to extend existing CityGML models (Labetski et al., 2018). While metadata can be stored as CityGML generic attributes, the data would lack structure and a clearly defined interface. Based on the state of the art it was decided to develop an ADE for CityGML to enable the storage of quality-management related data in a structured and consistent way.

3. Concept

The process of validation is shown in figure 2. The validation plan defines the requirements and their respective parameters. The validation software implements checks that verify the compliance with the defined requirements. The result of these checks is then stored in the ADE. The result status indicates whether a feature contains an error, is error-free, or has not yet been validated. This is essential for providing an overview of which features have been checked and their corresponding validation results. Without this status information, it would be unclear whether the data structure has been validated.

Checks, and their respective errors, can be categorized into three distinct classes: geometric, semantic, and topological.
Geometric checks evaluate CityObjects for internal geometric consistency, ensuring that the model does not contain errors such as missing or non-planar polygons, open rings, self-intersections, or non-manifold edges.
Semantic checks assess the meaning and correctness of CityObjects, for example, verifying that buildings contain a valid solid representation or that required attributes exist and their values fall within specified ranges or meet defined constraints. The necessary semantic checks depend heavily on the intended application or task and can therefore vary significantly between different models.
Finally, topological checks examine the geometric relationships between different CityObjects, such as verifying that features are properly connected to the terrain model, ensuring that objects do not intersect inappropriately, and checking compliance with spatial regulations—for instance, root protection zones around trees or ensuring that streets maintain an appropriate slope.

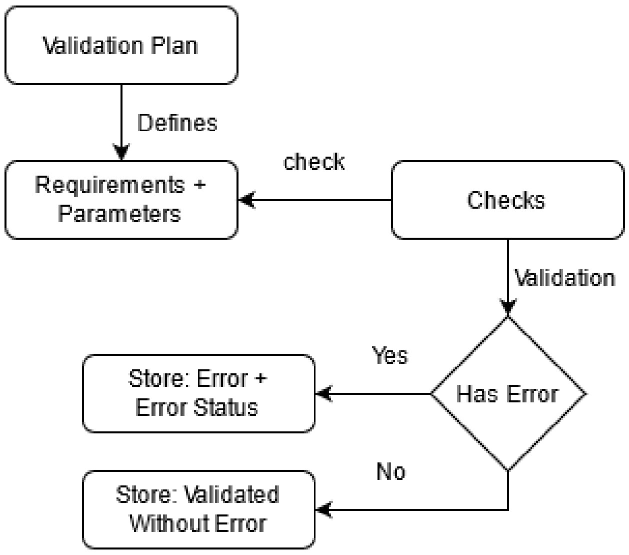

Validation Plan
Defines
Requirements + Parameters
check
Checks
Validation
Yes
Has Error
Store: Error + Error Status
No
Store: Validated Without Error

Figure 2. Validation Process

The paper (Coors et al., 2020) defines the scope of the geometric validation and the possible resulting errors. The ADE extension does not define how the results are obtained, it only creates a space for the results to be saved in the model itself. The definition of each error type and where it can occur is defined in the paper (Coors et al., 2020). The semantic requirements can vary significantly between the different applications. Most semantic requirements are simply the presence of attribute or that its value follows a prescribed set of rules. To include this information the ADE contains endpoints to store whether an attribute is missing or has the wrong value.

3.1 Data Structures

There are two main data structures in the Quality ADE. Firstly, the Validation structure which stores the validation plan, its configuration for the validation process and its metadata. The configuration contains the information which requirements are needed, including their respective configuration parameters. An overview of available requirements and their parameters can be found on the CityDoctor homepage [1]. Quality ADE currently supports the reporting of geometric and semantic errors.

The validation plan also provides the option to filter top-level features, enabling the skipping of feature types that are not of interest for a particular validation run.

The validation structure is for storing of the results of an actual validation run. It contains information about the system and date, as well as a reference to the used validation plan and the error statistics, which records the numbers and types of found errors.

Secondly, the ValidationResult structure (see figure 6) associated with a CityObject stores the result of the validation, such as found errors, for the respective feature as well as a result status showing if the feature was validated at all. While the ValidationResult structure can be attached to any CityObject, it is intended to be used solely on top-level features such as Buildings or Vegetation. Every error contains information about the cause of the error, e.g. the geometry of an unclosed ring or a missing attribute, to aid with the identification of the issue. Detailed information about every error and the included information can also be found on the CityDoctor homepage.
Following are examples for geometric and semantic errors to demonstrate how the errors are modeled.

Figure 4 shows the geometric errors on ring level. The errors are defined as shown in (Coors et al., 2020). Two points are considered the same when they are within a pre-defined distance (default value: 0.0001 m) to each other. There are four type of ring errors:

- **GE_R_TOO_FEW_POINTS:** Signals that a ring does not have enough distinct points (< 3) to form a valid ring.
- **GE_R_NOT_CLOSED:** Occuring when the start and end point of a ring are not the same point.
- **GE_R_CONSECUTIVE_POINTS_SAME:** Used when two consecutive points within a ring are the same point. The Error contains two DirectPosition structures to store both points.
- **GE_R_SELF_INTERSECTION:** Used when the ring intersects with itself, meaning a point is non consecutively repeated; two edges intersecting with each other; or a point being too close to another edge, effectively splitting the ring into two rings. The error stores the cause of the self intersection as an enumeration, as well as the meta information if applicable.

Figure 5 shows how semantic issues are reported in the Quality ADE. These errors can be for generic attributes, as well as attributes that are defined by the CityGML standard. There are three types of errors:

- **SE_ATTRIBUTE_WRONG_VALUE:** Used if an attribute exists in the CityObject but its value is wrong.
- **SE_ATTRIBUTE_MISSING:** Used if an attribute should exist in the CityObject but is missing.

[1] `https://transfer.hft-stuttgart.de/pages/citydoctor/citydoctorhomepage/en/geometric/`

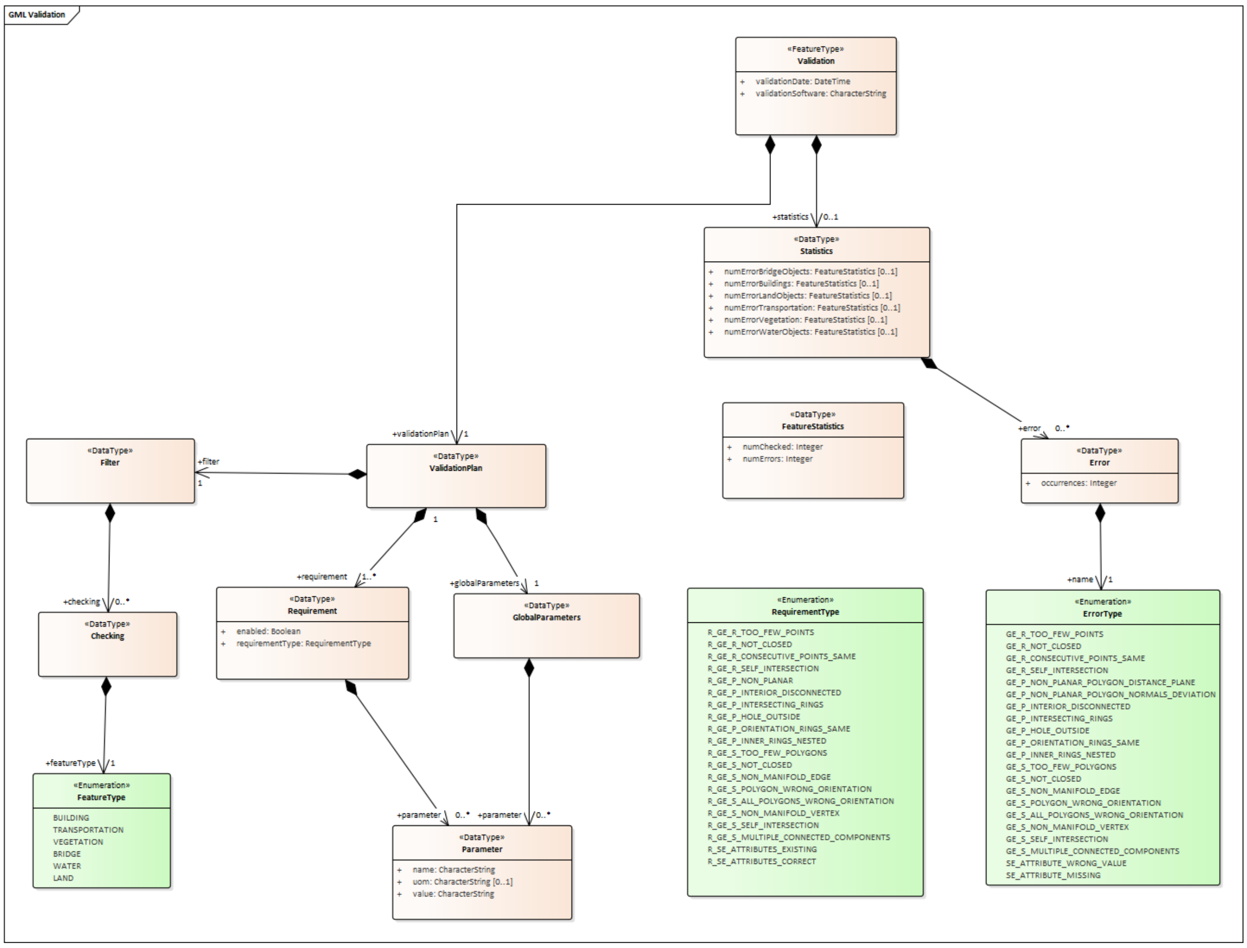

GML Validation
«FeatureType» Validation
+ validationDate: DateTime
+ validationSoftware: CharacterString
+statistics 0..1
«DataType» Statistics
+ numErrorBridgeObjects: FeatureStatistics [0..1]
+ numErrorBuildings: FeatureStatistics [0..1]
+ numErrorLandObjects: FeatureStatistics [0..1]
+ numErrorTransportation: FeatureStatistics [0..1]
+ numErrorVegetation: FeatureStatistics [0..1]
+ numErrorWaterObjects: FeatureStatistics [0..1]
«DataType» FeatureStatistics
+ numChecked: Integer
+ numErrors: Integer
+error 0..*
«DataType» Error
+ occurrences: Integer
+name 1
«Enumeration» ErrorType
GE_R_TOO_FEW_POINTS
GE_R_NOT_CLOSED
GE_R_CONSECUTIVE_POINTS_SAME
GE_R_SELF_INTERSECTION
GE_P_NON_PLANAR_POLYGON_DISTANCE_PLANE
GE_P_NON_PLANAR_POLYGON_NORMALS_DEVIATION
GE_P_INTERIOR_DISCONNECTED
GE_P_INTERSECTING_RINGS
GE_P_HOLE_OUTSIDE
GE_P_ORIENTATION_RINGS_SAME
GE_P_INNER_RINGS_NESTED
GE_S_TOO_FEW_POLYGONS
GE_S_NOT_CLOSED
GE_S_NON_MANIFOLD_EDGE
GE_S_POLYGON_WRONG_ORIENTATION
GE_S_ALL_POLYGONS_WRONG_ORIENTATION
GE_S_NON_MANIFOLD_VERTEX
GE_S_SELF_INTERSECTION
GE_S_MULTIPLE_CONNECTED_COMPONENTS
SE_ATTRIBUTE_WRONG_VALUE
SE_ATTRIBUTE_MISSING
+validationPlan 1
«DataType» ValidationPlan
«DataType» Filter
+filter 1
+checking 0..*
«DataType» Checking
+featureType 1
«Enumeration» FeatureType
BUILDING
TRANSPORTATION
VEGETATION
BRIDGE
WATER
LAND
+requirement 1..*
«DataType» Requirement
+ enabled: Boolean
+ requirementType: RequirementType
+globalParameters 1
«DataType» GlobalParameters
+parameter 0..*
+parameter 0..*
«DataType» Parameter
+ name: CharacterString
+ uom: CharacterString [0..1]
+ value: CharacterString
«Enumeration» RequirementType
R_GE_R_TOO_FEW_POINTS
R_GE_R_NOT_CLOSED
R_GE_R_CONSECUTIVE_POINTS_SAME
R_GE_R_SELF_INTERSECTION
R_GE_P_NON_PLANAR
R_GE_P_INTERIOR_DISCONNECTED
R_GE_P_INTERSECTING_RINGS
R_GE_P_HOLE_OUTSIDE
R_GE_P_ORIENTATION_RINGS_SAME
R_GE_P_INNER_RINGS_NESTED
R_GE_S_TOO_FEW_POLYGONS
R_GE_S_NOT_CLOSED
R_GE_S_NON_MANIFOLD_EDGE
R_GE_S_POLYGON_WRONG_ORIENTATION
R_GE_S_ALL_POLYGONS_WRONG_ORIENTATION
R_GE_S_NON_MANIFOLD_VERTEX
R_GE_S_SELF_INTERSECTION
R_GE_S_MULTIPLE_CONNECTED_COMPONENTS
R_SE_ATTRIBUTES_EXISTING
R_SE_ATTRIBUTES_CORRECT

Figure 3. UML of the Validation and ValidationPlan structure.

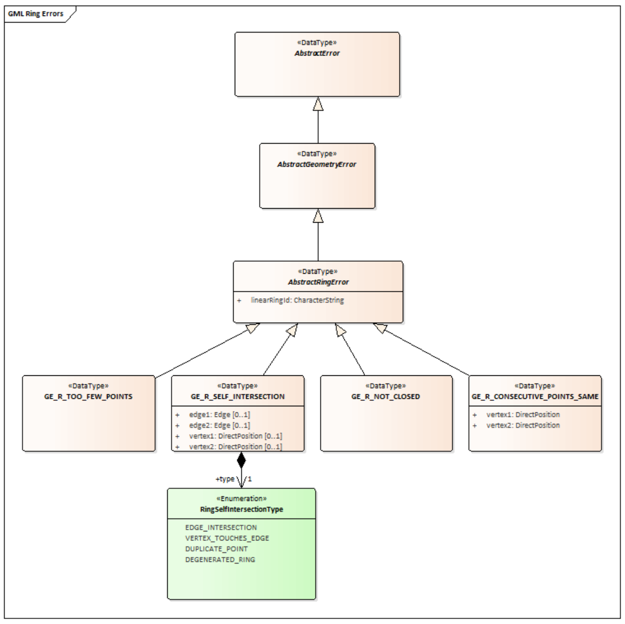

GML Ring Errors
«DataType» AbstractError
«DataType» AbstractGeometryError
«DataType» AbstractRingError
+ linearRingId: CharacterString
«DataType» GE_R_TOO_FEW_POINTS
«DataType» GE_R_SELF_INTERSECTION
+ edge1: Edge [0..1]
+ edge2: Edge [0..1]
+ vertex1: DirectPosition [0..1]
+ vertex2: DirectPosition [0..1]
«DataType» GE_R_NOT_CLOSED
«DataType» GE_R_CONSECUTIVE_POINTS_SAME
+ vertex1: DirectPosition
+ vertex2: DirectPosition
+type 1
«Enumeration» RingSelfIntersectionType
EDGE_INTERSECTION
VERTEX_TOUCHES_EDGE
DUPLICATE_POINT
DEGENERATED_RING

Figure 4. Ring error structures

- **SE_ATTRIBUTE_INVALID:** Used if an attribute exists that should not exist in this place. This is typically used if a subset of the CityGML standard should be used for modeling features. These restriction have to be validated and can result in attributes that may be allowed in the CityGML standard but not in the subset therefore are invalid for this application.

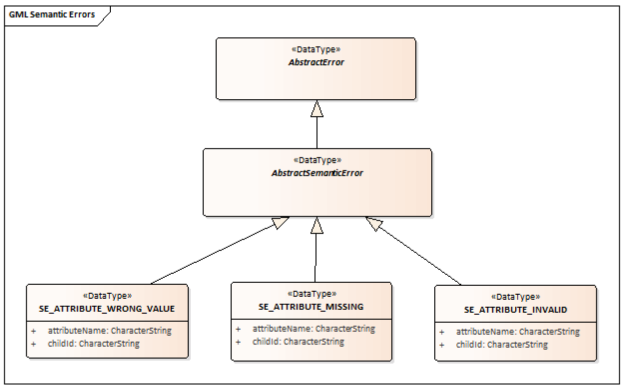

GML Semantic Errors
«DataType» AbstractError
«DataType» AbstractSemanticError
«DataType» SE_ATTRIBUTE_WRONG_VALUE
+ attributeName: CharacterString
+ childId: CharacterString
«DataType» SE_ATTRIBUTE_MISSING
+ attributeName: CharacterString
+ childId: CharacterString
«DataType» SE_ATTRIBUTE_INVALID
+ attributeName: CharacterString
+ childId: CharacterString

Figure 5. Semantic error structures

Each of the semantic errors contains two values which are used to provide more information on the error.

- **attributeName:** The name of the attribute for which the error occured.
- **childId:** The childId refers to the gmlId of a CityObject that is contained within the structure of a top level Feature. For example, if a building contains a building part that has a missing attribute, the childId contains the gml:id of the building part.

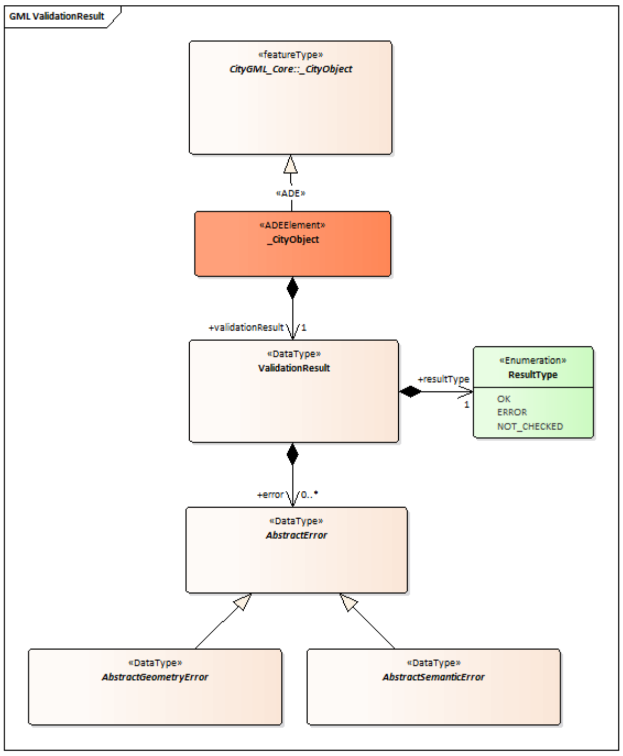


Figure 6. UML of the ValidationResult structure

4. Implementation

An ADE for CityGML Version 2.0 has been implemented and published in accordance with the designed concept[2]. This also includes an plugin for the parser library citygml4j[3].

4.1 CityDoctor

The ADE has been implemented in the validation software CityDoctor (Betz and Riegel, 2021). CityDoctor can validate CityGML files and write the results back into the GML file. Together with XML validation via Schematron it is capable of validating the semantic requirements for different applications. This allows validation of generic cases like missing attributes or wrong values, which covers the typical use cases for semantic validation of CityGML files. CityDoctor is implemented using the programming language Java and uses the citygml4j library for reading/writing of CityGML files. CityDoctor has been integrated in tools and workflows of multiple companies, demonstrating that interest in quality management of city model datasets exists.

CityDoctor is capable of validating all CityGML versions, including the recently released version 3.0. However, the current implementation of the Quality ADE is not compatible with CityGML 3.0, as such it is currently not possible to store the validation results of CityGML 3.0 files using the ADE.

CityDoctor is fully free and open source and can be viewed on our gitlab: `https://transfer.hft-stuttgart.de/gitlab/citydoctor/citydoctor2`.

[2] `https://transfer.hft-stuttgart.de/gitlab/citydoctor/qualityade`

[3] `https://github.com/citygml4j/citygml4j`

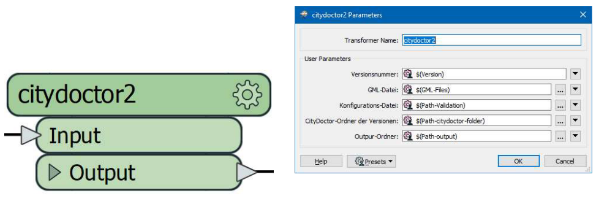
4.1.1 Semantic Validation CityDoctor uses the XML validation framework Schematron to validate sematic issues in the CityGML. As each application imposes unique requirements on specific attributes and may also include constraints for generic attributes, a flexible workflow was needed to allow adaptation for these varying requirements. By integrating Schematron users can attach a Schematron rules file to the validation plan, which is then executed as part of the validation process. The Quality ADE provides generic error types to allow storage of the Schematron validation results.

4.2 citygml4j Plugin

The library citygml4j is used to parse and write CityGML files in Java. The library offers support for plugins that enable the parsing and writing of ADEs. For the integration of the Quality ADE in CityDoctor an open source plugin has been developed and is provided on a gitlab instance[4]. The plugin currently only supports CityGML 2.0, an update for CityGML 3.0 support is currently in development.

4.3 FME Transformer

CityDoctor has also been integrated as an FME Transformer to be used within a FME Workbench (see figure 7). The transformer is able to handle the output that CityDoctor is writing to be used further in a workflow.



Transformer Name: citydoctor2

User Parameters

Versionsnummer: $(Version)
GML-Datei: $(GML-Files)
Konfigurations-Datei: $(Path-Validation)
CityDoctor-Ordner der Versionen: $(Path-citydoctor-folder)
Outpur-Ordner: $(Path-output)

Figure 7. FME Transformer for CityDoctor

5. Evaluation

The implementation of the validation and QualityADE was tested on multiple datasets. We will focus here on two datasets one from the region of Stöckach and one from Niedernhall. Both datasets are DLM (Digital Landscape Model) tiles with an area of two square kilometers. These models not only contain buildings but also vegetation objects, transportation objects, bridges, water bodies and land use objects. The geometry of all of those objects was validated. In addition to the geometry a semantic validation for buildings was also performed which checks if a solid geometry exists in said building.

5.1 File size and errors

Table 2 shows the impact of the ADE on the file size of both datasets. figure 8 and figure 9 show what kind of errors were found in the tiles.

Tile	Features	With errors	Percentage
Stöckach	25685	1182	5%
Niedernhall	5737	955	17%

Table 1. Number of features per tile

[4] `https://transfer.hft-stuttgart.de/pages/citydoctor/citydoctorhomepage/en/qual_ade/`

	File Size [MB]	
Tile	**Original**	**With ADE**
Stöckach	3943	3950
Niedernhall	3245	3248

Table 2. File Sizes of Tiles With and Without ADE

The Quality ADE increased the file sizes of the Stöckach and Niedernhall tiles by 7 MB (0.17%) and 3 MB (0.09%) respectively. This increase is negligible in this case but it also depends on the kinds of errors that are in the files as for each error metadata about the error is written. This results in varying amount of data as part of the ADE.

For an overview of the found errors see table 3. For more information on the errors, what they are and how they work visit the CityDoctor homepage where we list more detailed information[5].

Error Type	Stöckach	Niedernhall
GE_R_SELF_INTERSECTION	612	223
GE_R_TOO_FEW_POINTS	2987	1532
GE_R_CONSECUTIVE_POINTS_SAME	9	25
GE_P_NON_PLANAR_POLYGON_DISTANCE_PLANE	0	82
GE_S_MULTIPLE_CONNECTED_COMPONENTS	0	251
GE_S_NOT_CLOSED	0	90
GE_S_NON_MANIFOLD_VERTEX	0	25
GE_S_NON_MANIFOLD_EDGE	0	13
SE_ATTRIBUTE_MISSING	0	112

Table 3. Error counts by type and location for the tiles Stöckach and Niedernhall

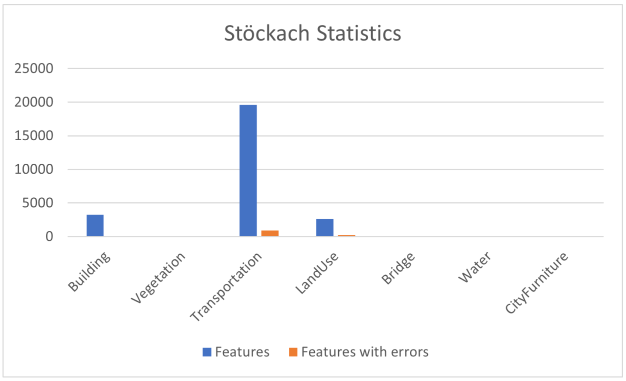


Figure 8. Statistics of top level features in the Stöckach tile.

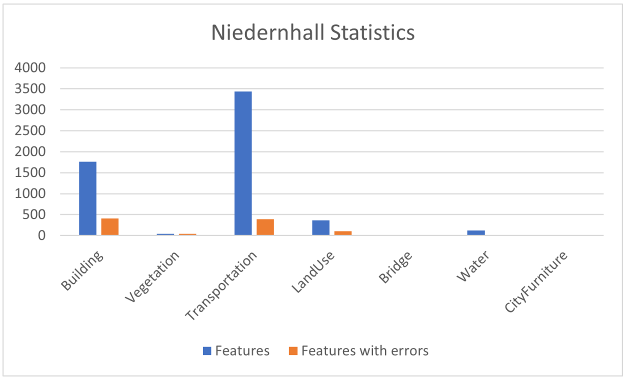


Figure 9. Statistics of top level features in the Niedernhall tile.

[5] https://transfer.hft-stuttgart.de/pages/citydoctor/citydoctorhomepage/en/geometric/

5.2 Performance

The performance of the validation process is difficult to predict, as it is influenced by multiple factors that affect the total runtime. The primary factors are the number of features, the complexity of these features, and the clock speed of the system's CPU. Secondary factors are the the number and types of errors encountered. Certain checks have dependencies on certain errors not being present in a feature, and will be skipped should their dependency not be met.

For the DLM tiles the duration of the validation including writing the complete CityGML with the ADE is shown in table 4.

Tile	Duration [mm:ss]	per Feature [ms]
Stöckach	08:04	18.8
Niedernhall	08:20	87.2

Table 4. Processing durations per tile and feature

The AdV[6] provided runtime data for approximately 50 million buildings from their validation of LOD-2 models of the federal states of Germany (see table 5) with CityDoctor2. The AdV has the geometry data of all buildings of all states of Germany in LOD-2 encoded in CityGML[7] (see figure 10) and is validating them.

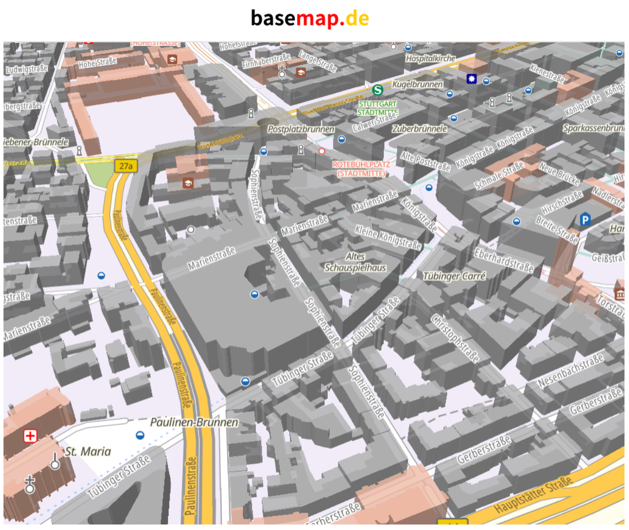


Figure 10. 3D models in a viewer provided by the AdV

While a report is generated, their integration of the CityDoctor validation library is not using the Quality ADE due to technical reasons. As the integration is not writing the CityGML back with the ADE this is resulting in faster validation times as writing CityGML is a time consuming process relatively to the validation process.

[6] 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)

[7] https://dev.adv-smart.de/index.html

The validation was being run one state at a time, with lower Saxony and Berlin at the time of writing not yet being validated. Each state-model only contains Building Features. Since CityDoctor does not currently support parallelization, only a single core of the CPU is utilized and thus the processor's boosted clock rate is being used. With the provided data, the average runtime per Feature is calculated to be:

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

Total Number of Features	50,221,985
Total runtime	3089 min
CPU name	AMD EPYC 7313P
CPU clock speed	3 GHz, 3.7 GHz Boost

Table 5. Runtime data provided by AdV

The difference in validation speed can be explained from writing the CityGML with the DLM tiles validation. While the added performance loss by adding the quality ADE is negligible writing the CityGML itself is a time consuming process.

6. Outlook

Due to the growing adoption of CityGML version 3.0 an update for the Quality ADE is currently in development.

This update requires extended changes to the datastructure, as CityGML version 3.0 introduces major changes to the CityGML data model. Most notably is the removal of LOD 4, including the decoupling of interior models from LOD 4, thus allowing any LOD to implement an interior representation.

Other notable changes include the introduction of the CityGML Core module, which serves as the root module extended by all other CityObject modules; the addition of the Construction module, which is a parent module for the Building, Bridge and Tunnel modules; and a restructured datamodel for Transportation Features. These changes modify the XML namespaces of the affected modules, thus requiring the Schematron rules to be updated accordingly. Furthermore, support for auxiliary data, e.g. sensor data, point clouds, and versioning were introduced. ClosureSurfaces have also been added for the virtual closure of hollow objects, such as tunnels, to allow the calculation of their volumes.

While CityGML is designed to be backward compatible, this does not necessarily apply to ADEs. Therefore, it is not yet possible to determine whether the upcoming update of the Quality ADE will maintain backward compatibility. However, a practical workaround to ensure backward compatibility is to integrate both the current and future versions of the ADE, selecting the appropriate version based on the detected version of the CityGML file.

CityDoctor is currently in active development with the aim of implementing the validation of topological requirements, which govern geometric relationships between different features. Examples for such relationships include the following: Buildings should not intersect with each other, except for shared walls; a tree has a regulated buffer area for its root system that must remain free of intersecting subterranean objects, such as utility lines; and CityObjects must correctly intersect with a Digital Terrain Model (DTM), ensuring that they are neither floating above nor buried within it. The validation of these types of relationships will necessitate the introduction of new error structures, which will require a future corresponding update of the Quality ADE.

Furthermore, we are intending to implement a plugin for 3dCityDB, allowing storage of the Quality ADE data in the widely used database (Yao et al., 2025).

Acknowledgements

The authors would like to thank the Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV) for providing performance datasets used in this research.

This work was sponsored by the Federal Ministry of Research, Technology and Space, funding program Forschung an Fachhochschulen, FKZ 13FH019KA2, Project UDigiTDoctor.

References

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://isprs-annals.copernicus.org/articles/VIII-4-W1-2021/11/2021/.

Betz, M., Riegel, A., 2021. CityDoctor2. `https://transfer.hft-stuttgart.de/pages/citydoctor/citydoctorhomepage/en/`. [Accessed 25-04-2025].

Biljecki, F., Ledoux, H., Du, X., Stoter, J., Soon, K. H., Khoo, V., 2016. THE MOST COMMON GEOMETRIC AND SEMANTIC ERRORS IN CITYGML DATASETS. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F., 1997. Extensible markup language (XML). *World Wide Web Journal*, 2(4), 27–66.

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.

Kavisha, K., 2020. Modelling and managing massive 3D data of the built environment.

Kutzner, T., Smyth, C. S., Nagel, C., Coors, V., Vinasco-Alvarez, D., Ishimaru, N., Yao, Z., Heazel, C., Kolbe, T. H., 2023. Ogc city geography markup language (citygml) part 2: Gml encoding standard. Standard OGC 21-006r2, Open Geospatial Consortium.

Labetski, A., Kumar, K., Ledoux, H., Stoter, J., 2018. A metadata ADE for CityGML. *Open Geospatial Data, Software and Standards*, 3(1), 1–16.

Ledoux, H., 2018. val3dity: validation of 3D GIS primitives according to the international standards. *Open Geospatial Data, Software and Standards*, 3(1), 1–12.

Ledoux, H., Wagner, D. (eds), 2016. *OGC® CityGML Quality Interoperability Experiment*. 16-064r1, Open Geospatial Consortium (OCG).

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.

Yao, Z., Nagel, C., Kendir, M., Willenborg, B., Kolbe, T. H., 2025. The New 3D City Database 5.0 - Advancing 3D City Data Management based on CityGML 3.0.