AN APPLICATION DOMAIN EXTENSION FOR STORING VALIDATION RESULTS OF CITYGML STRUCTURES

Matthias Betz[1,*], Volker Coors[1]

[1] Stuttgart University of Applied Sciences, Germany

KEY WORDS: CityGML, ADE, Validation, XML

ABSTRACT:

With the increase of applications using 3D models as base for visualization or simulation the requirements for having valid models rises. While there are many programs that validate models there is no standard for storing the results. The simulation programs also may implement their own validation checking to verify if a model can be simulated which may be error prone or not extensive enough for their purpose. This paper provides an extension for the CityGML format to store the validation result. With the available validation information software using the extension can improve their process of working with models containing errors. CityGML model manufacturers may also improve their creation process of the model as well as deliver them validated.

1. INTRODUCTION

3D spatial models of cities or whole landscapes have been and are going to be increasingly more popular visualization, simulations, disaster management and many more applications. The more requirements the application has for the model the more important it gets that those requirements are actually fulfilled by the available data. The widely used standard for storing 3D spatial data is a XML (Consortium et al., 2006) based data format called CityGML (City Geography Markup Language) (Gröger et al., 2012). The current CityGML standard version 2 has no way to store data containing information about the correctness or validity of the geometric and semantic data in a data set. As the data format is based on the XML standard it can be and is intended to be extended. CityGML has created a system for those extensions called an ADE (Application Domain Extension) (Gröger et al., 2012) which can be used to inject additional modules into existing CityGML data structures. This is used in this paper to create data structures for storing validation data for features. With this information available in the data itself a simulation such as SimStadt (Nouvel et al., 2015) can give more credible results as it can guarantee that the input is correct or not. It is also possible to outright exclude parts of the data and simulate only the parts that are correct.

It is also useful for CityGML manufacturers as they are able to create models and deliver them validated with the validation information contained in the model itself. They can also use the information to improve the process of the geometry generation or fix geometries in a post processing step to increase the quality of the created models. The quality management process and improvement is shown in figure 1.

2. STATE OF THE ART

There are many papers already describing algorithms and processes for validating 3D city models, see (Coors et al., 2020), (Ledoux and Wagner, 2016),(Biljecki et al., 2016).

To be able to work with massive amounts of data, meta data management becomes a necessity (Kavisha, 2020). One of the possibilities for meta data storing is creating and using an ADE to extend existing CityGML models (Labetski et al., 2018).

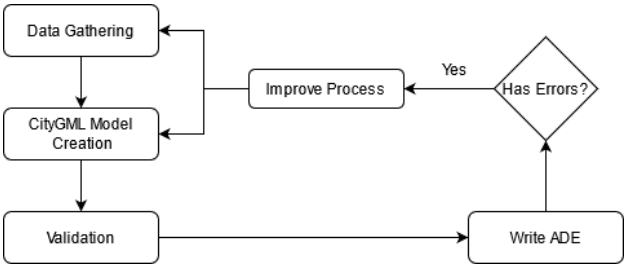

Figure 1. Quality Management Process

Based on the state of the art it was decided to create an ADE for CityGML so allow storage of quality management related data. Therefore the CityGML Quality ADE was designed.

3. CONCEPT CITYGML QUALITY ADE

The process of validation is shown in figure 2. The validation plan defines the requirements and the parameters for the requirements. The validation software implements checks that test the defined requirements. The result is then stored in the ADE. The result includes both the error case as well as the OK case or the not validated case if the feature has been excluded from the validation process. This is important to give a complete overview of what was tested and which were the exact result in all cases.

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 space for the results to be written into the model itself. The definition of each error type and where it can occur is defined in the paper (Coors et al., 2020). Additionally the semantic requirements can be very different between each application. Most of those requirements are simply the presence of attributes or the correctness of an attribute. To include these information the ADE contains endpoints to store whether an attribute is missing or has the wrong value.

* Corresponding author

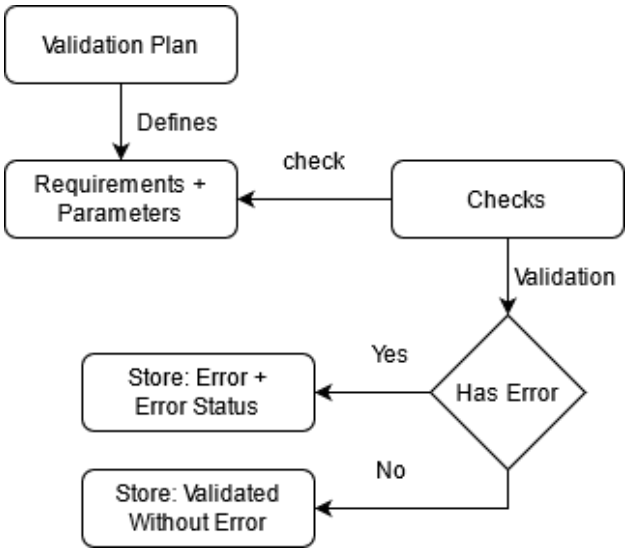

Figure 2. Quality Management Process

4. DATA STRUCTURES

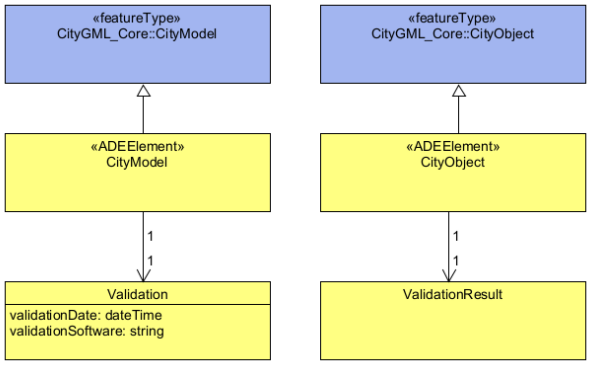

Figure 3. Main data structures

There are two main data structures (see figure 3) in the ADE, first is the validation structure which is associated to a CityModel CityGML structure. Second the validationResult structure associated to CityObjects. While the validationResult structure can be written in each CityObject it is intended to use it solely on top level features such as buildings, transportation objects, vegetation objects etc.

4.1 Validation Structure

The validation structure contains meta information of the validation process. It contains the date of the validation as well as a string identifying the validation software. Furthermore it contains a statistic of the amount of features checked and how many of them contain any error and a statistic of all errors found in the model. The most important part of the validation structure is the validation plan. It contains which requirements where validated and with which parameters. This is essential to be able to reproduce the results in different validation runs or to compare the results with results of different validation software. Without the plan the results would be meaningless.

4.2 Validation Plan

The validation plan consists of a list of requirements (see table 1 and uml diagram in figure 4) with each requirement containing a list of parameters. There is also a section for global parameters where a list of parameters that apply to many or all requirements are defined.

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

Table 1. Defined Requirements

An example validation plan would look like this in the final document (excerpt):

```
<qual:validationPlan>
  <qual:globalParameters>
    <qual:parameter name="numberOfRoundingPlaces">
          8</qual:parameter>
    <qual:parameter name="minVertexDistance"
          uom="m">1.0E-4</qual:parameter>
    <qual:parameter name="schematronFile"/>
  </qual:globalParameters>
  <qual:requirement
     name="R_GE_S_ALL_POLYGONS_WRONG_ORIENTATION"
     enabled="true"/>
  <qual:requirement
     name="R_GE_P_INNER_RINGS_NESTED"
     enabled="true"/>
  <qual:requirement
     name="R_GE_S_NOT_CLOSED" enabled="true"/>
  <qual:requirement
     name="R_GE_S_POLYGON_WRONG_ORIENTATION"
     enabled="true"/>
  <qual:requirement
     name="R_GE_P_NON_PLANAR" enabled="true">
    <qual:parameter name="angleTolerance"
        uom="degree">1</qual:parameter>
    <qual:parameter name="distanceTolerance"
        uom="m">0.01</qual:parameter>
    <qual:parameter name="type">distance
        </qual:parameter>
  </qual:requirement>
  <qual:requirement
     name="R_GE_R_TOO_FEW_POINTS"
     enabled="true"/>
```

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-VIII-4-W1-2021-11-2021 | © Author(s) 2021. CC BY 4.0 License.

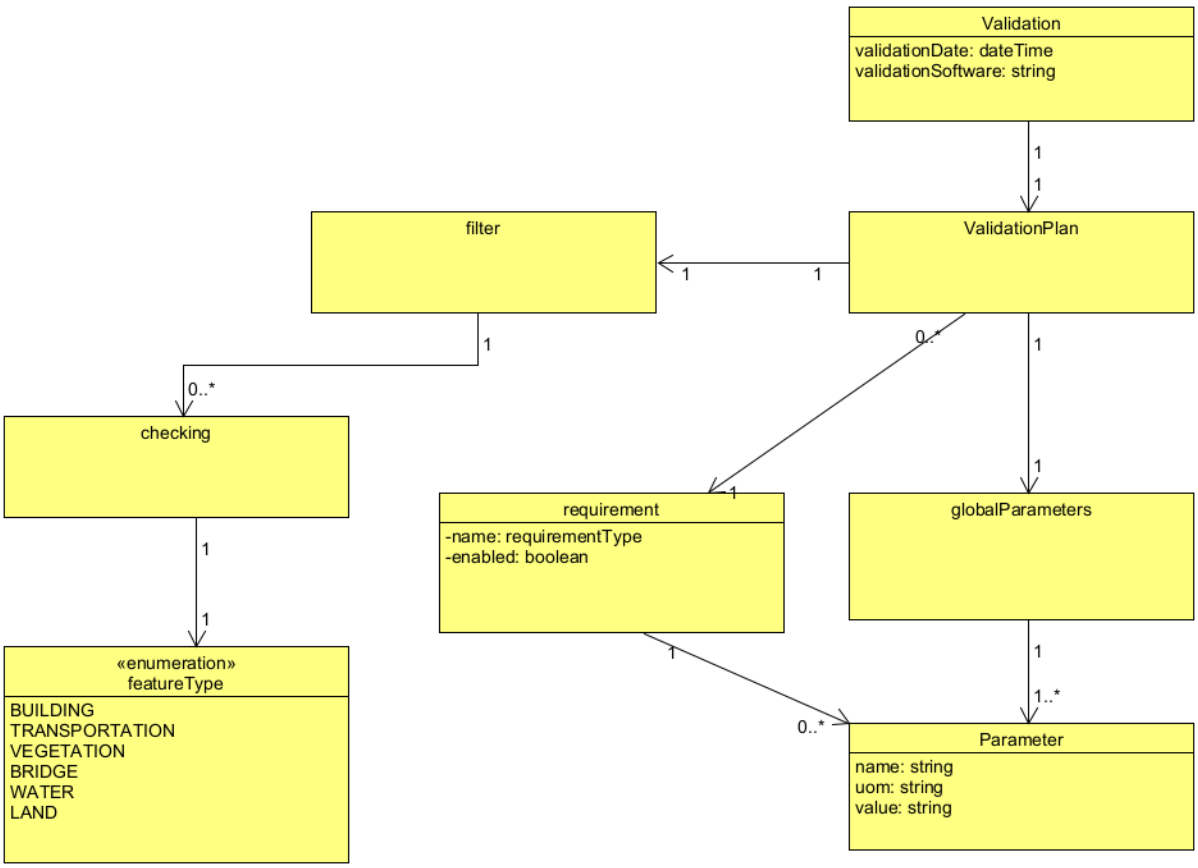

Figure 4. Validation Plan Structure

4.3 Statistics

The statistics structure has 5 variables for storing the amount of top level features that were validated and the amount of them having errors. See table 2 for which variable counts which CityGML type.

Variable	CityGML Type
numErrorBuildings	BuildingType
numErrorVegetation	AbstractVegetationObjectType
numErrorLandObjects	LandUseType
numErrorBridgeObjects	BridgeType
numErrorWaterObjects	AbstractWaterObjectType
numErrorTransportation	AbstractTransportation ObjectType

Table 2. CityGML Type to error variable mapping

4.4 ErrorStatistics

The error statistics is a simple list of error type associated with the number of times it occurred in the model. A list of possible error types is listed in table 3

ErrorType
GE_R_TOO_FEW_POINTS
GE_R_NOT_CLOSED
GE_R_CONSECUTIVE_POINTS_SAME
GE_R_SELF_INTERSECTION
GE_P_NON_PLANAR_POLYGON_NORMALS_DEVIATION
GE_P_NON_PLANAR_POLYGON_DISTANCE_PLANE
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

Table 3. Possible error types

4.5 ValidationResult

This is the data structure that should be associated with top level features in a CityModel. It contains information about whether the validation was successful, an error occurred or even checked at all. In case of errors there will also be information about the error, the error type and additional error information depending on the type. Figures 5, 6, 7 and 8 show the data structures needed to store the errors in detail.

An example validationResult structure with a GE_R_CONSECUTIVE_POINTS_SAME error would look like the following XML example and is situated in the ADE of a CityObject. It contains information in which ring the error occured and which consecutive vertices are the same.

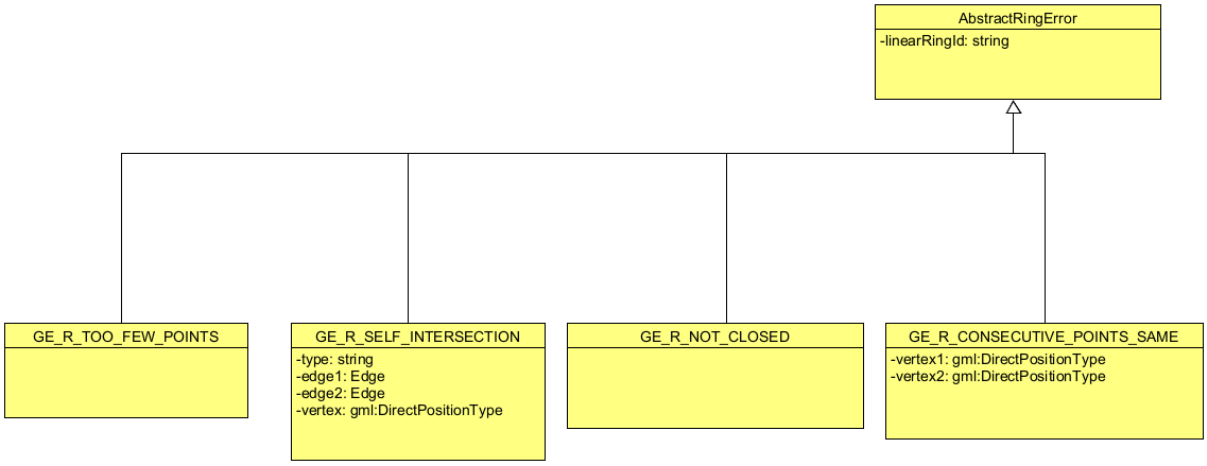

Figure 5. Ring Error Datastructures

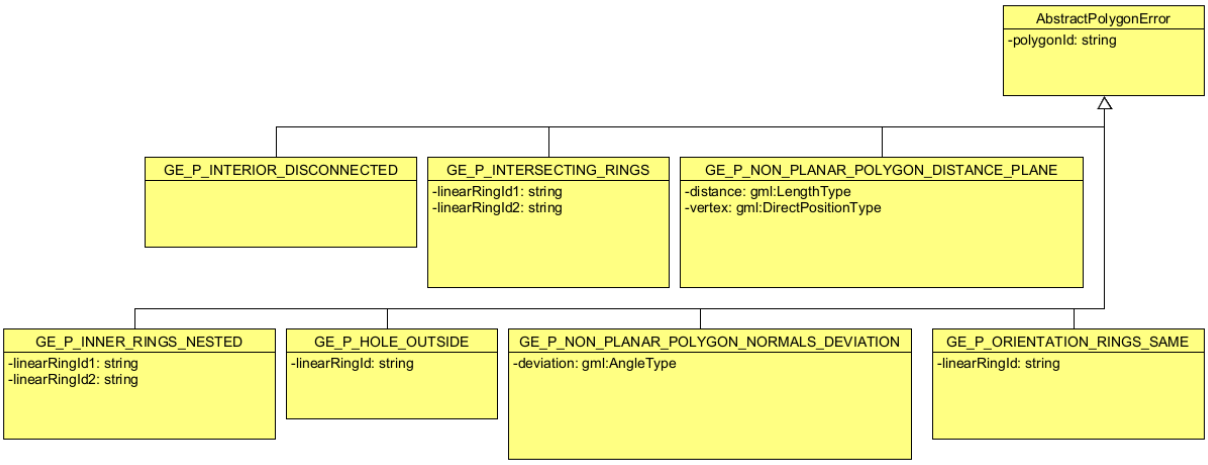

Figure 6. Polygon Error Datastructures

```
<qual:validationResult result="ERROR">
  <qual:GE_R_CONSECUTIVE_POINTS_SAME>
    <qual:linearRingId>_Simple_BD.1_PG.6_LR.1
        </qual:linearRingId>
    <qual:vertex1>11.5 10.0 4.5</qual:vertex1>
    <qual:vertex2>11.5 10.0 4.5</qual:vertex2>
  </qual:GE_R_CONSECUTIVE_POINTS_SAME>
</qual:validationResult>
```

5. IMPLEMENTATION

A reference implementation was created with a citygml4j plugin in java. The implementation is open source and available here: `https://transfer.hft-stuttgart.de/gitlab/citydoctor/qualityade`. It allows for an integration into the widely used citygml4j parser and working with the defined data structures. It can be used for writing and reading enriched CityGML files. The complete UML model and examples can be viewed on the wiki page here: `https://gitlab.com/volkercoors/CiD4Sim/-/wikis/validation/QualityADE`

The validation is not part of the implementation as there can be many different implementations of the process. The validation can be done with the open source software CityDoctor2 `https://transfer.hft-stuttgart.de/gitlab/citydoctor/citydoctor2` but it can also be validated with val3dity (Ledoux, 2018)

The citygml4j plugin mechanism relies on the Java functionality of the ServiceLoader where implementations of interfaces can be easily loaded at runtime without knowing the implementation. This is used for loading all ADEs with the interface name ADEContext. If the interface is correctly implemented it provides citygml4j with all the necessary information to integrate the ADE into the main software. Citygml4j is based on JAXB (Jakarta XML Binding) which transforms XML based data into Java classes. For the ADE to be able to use JAXB in conjunction with citygml4j the ADEContext interface declares methods so it can be integrated into the mapping process.

5.1 Usage

CityDoctor2 is using the ADE for storing the result of the validation process. For the validation of semantic issues in models a XML validation process called Schematron (Van der Vlist, 2007) is used. By defining an exact output format of schemat-

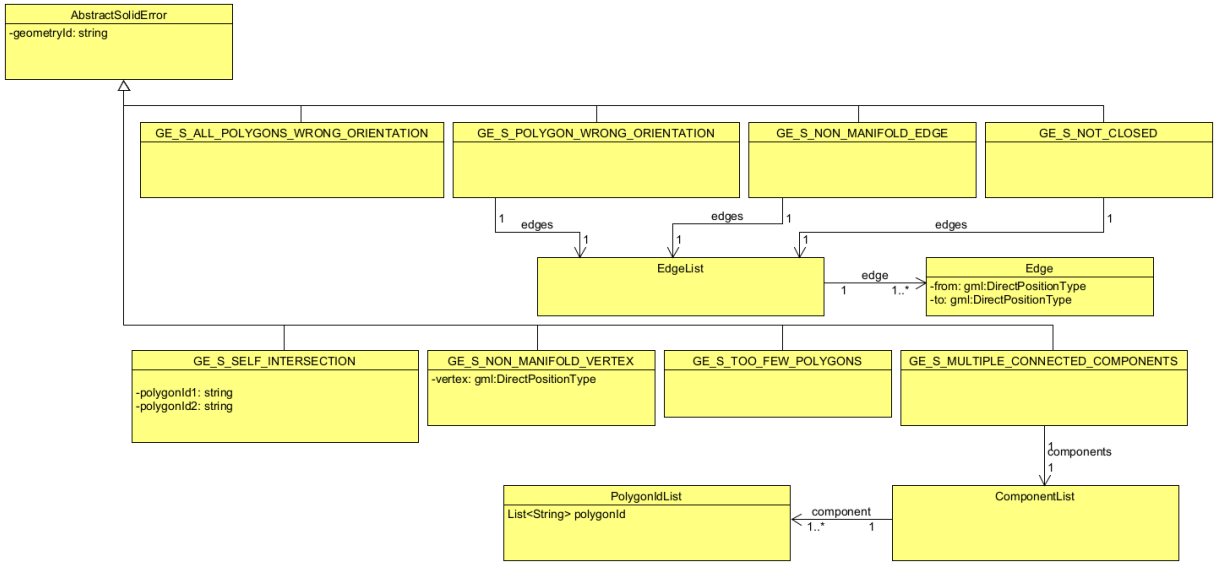

Figure 7. Solid Error Datastructures

Model Name	Errors
SimpleSolid_SrefBS-GE-gml-LR-0002-T0001-ADE	GE_R_CONSECUTIVE_POINTS_SAME
SimpleSolid_SrefBS-GE-gml-LR-0003-T0001-ADE	GE_R_NOT_CLOSED
SimpleSolid_SrefBS-GE-gml-PO-0001-T0001-ADE	GE_P_INTERSECTING_RINGS
SimpleSolid_SrefBS-GE-gml-PO-0002-T0001-ADE	GE_P_NON_PLANAR_POLYGON_DISTANCE_POLYGON
SimpleSolid_SrefBS-GE-gml-PO-0005-T0001-ADE	GE_P_INNER_RINGS_NESTED
SimpleSolid_SrefBS-GE-gml-SO-0004-T0001-ADE	GE_S_NON_MANIFOLD_EDGE
REKaiserwall-ADE	Various Errors

Table 4. Model examples with ADE information stored

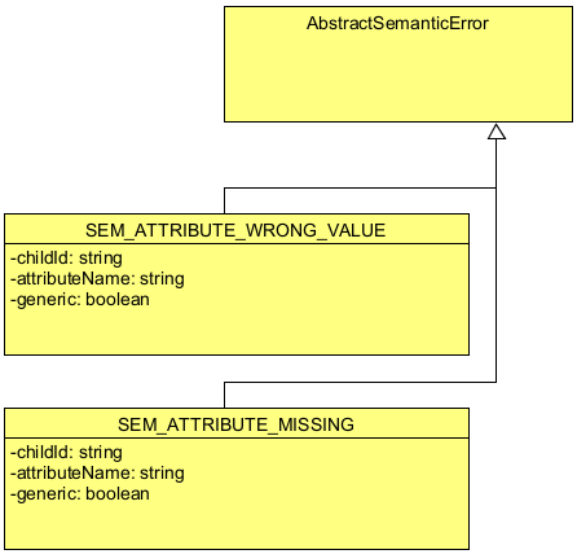

Figure 8. Semantic Error Datastructures

ron errors, they can be integrated into the CityDoctor validation process after the schematron validation process has finished.

6. DISCUSSION, CONCLUSION AND FUTURE WORK

An ADE extending CityGML has been developed that can be used to store validation results and has been successfully integrated into the open source software CityDoctor. The results can be used by different application software to be more aware of potential problems within the CityGML model.

6.1 Test Data

In table 4 a list of CityGML files was verified and the ADE was written into the files. The complete list with the original models and the models with ADE for downloading can be found here: `https://gitlab.com/volkercoors/CiD4Sim/-/wikis/validation/QualityADE`

6.2 Data Volume

A large CityGML file of around 4GB was validated and enriched with the ADE. The file size increased by around 2MB and has around 0.05% more lines.

6.3 Extensions

An extension for storing quality management meta data has been created and works as intended. The principle can be transferred to other encodings for 3D models such as CityJSON, the content and intention of the ADE is not based on the encoding of the underlying data.

With the citygml4j plugin there is a reference implementation for reading files but there is no database extension for 3DCityDB (Yao et al., 2018) or GeoRocket (Krämer, 2020) which would need to be implemented in the future.

The semantic error types can be extended as they are application specific most of the time. The current semantic errors are there to show that semantic errors are possible to store in the ADE.

6.4 Limitations

As the semantic errors can be very divers depending on the applications it is impossible to include error types which contain the necessary information for all possible semantic errors. As an example you can have the requirement that the roof surfaces of a building should only contain polygons that are planar to each other. This is important if the normal of roof surfaces containing multiple polygons should be calculated. That error would need a new type containing the relevant information like the normal and which polygon(s) are deviating too much.

REFERENCES

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.

Consortium, W. W. W. et al., 2006. Extensible markup language (XML) 1.1.

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.

Gröger, G., Kolbe, T. H., Nagel, C., Häfele, K.-H., 2012. OGC city geography markup language (CityGML) encoding standard.

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

Krämer, M., 2020. GeoRocket: A scalable and cloud-based data store for big geospatial files. *SoftwareX*, 11, 100409.

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.

Van der Vlist, E., 2007. *Schematron*. " O'Reilly Media, Inc.".

Yao, Z., Nagel, C., Kunde, F., Hudra, G., Willkomm, P., Donaubauer, A., Adolphi, T., Kolbe, T. H., 2018. 3DCityDB-a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML. *Open Geospatial Data, Software and Standards*, 3(1), 1–26.