Delft University of Technology

OGC® CityGML Quality Interoperability Experiment

Ledoux, Hugo; Wagner, D.

Publication date
2016

Document Version
Final published version

Citation (APA)
Ledoux, H., & Wagner, D. (Eds.) (2016). *OGC® CityGML Quality Interoperability Experiment*. Open Geospatial Consortium (OCG).

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

OGC® CityGML Quality Interoperability Experiment

Warning

This document is not an OGC Standard. This document presents a discussion of technology issues considered in an initiative of the OGC Interoperability Program. This document does not represent an official position of the OGC. It is subject to change without notice and may not be referred to as an OGC Standard. However, the discussions in this document could very well lead to the definition of an OGC Standard.

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable

Contents Page

iii

Figures

Tables

Page

OGC® CityGML Quality Interoperability Experiment

1   Introduction

1.1   Preface

The aim of the CityGML Quality Interoperability Experiment was to define data quality requirements for a general CityGML data specification, to provide recommended implementation guidance for 3D data, and to provide a suite of essential quality checking tools to carry out quality assurance on CityGML data. The data requirements and recommended implementation guidance were obtained by experimentation and may serve as input to extend and refine the CityGML standard. These requirements and guidance should meet the OGC Membership and community "thirst" for better implementation guidance for OGC standards, in this case CityGML.

The QIE was carried out as a joint activity between OGC, SIG3D and EuroSDR.

1.2   Scope

This OGC Engineering Report specifies the results and findings of the CityGML Quality Interoperability Experiment. Guidelines were developed for the following concepts:

- Definition of data quality;
- Data quality requirements and their specification;
- Quality checking process of CityGML data; and
- Description of validation results.

The desired outcomes of this Interoperability Experiment are to improve the interoperability of CityGML data by removing some ambiguities from the current standard and formally defining data quality requirements for a general CityGML data specification. Further, the results of this work provides to the community (organizations invested in capturing, procuring, or utilizing CityGML data) recommended implementation guidance for 3D data and a suite of essential quality checking tools to carry out quality assurance on CityGML data.

1.3   Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization
Carsten Rönsdorf (CR)	Ordnance Survey
Detlev Wagner (DW)	University of Tehran
Simon Thum (ST)	Fraunhofer IGD
Dean Hintz (DH)	Safe Software

Hugo Ledoux (HL)	Delft University of Technology
Filip Biljecki (FB)	Delft University of Technology
Jantien Stoter (JS)	Delft University of Technology, Geonovum, and EuroSDR
Egbert Casper (EC)	SIG3D
Joachim Benner (JB)	Karlsruhe Institute for Technology
Volker Coors (VC)	Stuttgart University of Applied Sciences
Lucas van Walstijn (LW)	virtualcitysytems

1.4 Revision history

Date	Release	Editor Contributor	Primary clauses modified	Description
2015-02-08		DW		Draft of report structure
2015-04-30		HL		Draft of all parts related to the geometric validation part
2015-06-01		DW		Draft version for discussion
2015-07-27		DW		Draft version for discussion
2015-08-10		ST	7.6, 9.2	Fraunhofer input
2015-08-13		DH	7.6.2, 9.2.3	Safe Software input
2015-08-14		DW		Overall review
2015-08-17		JS		Overall review and update previously self-written parts
2015-09-22		LW	9.4.1	Added input on conformance requirements
2016-02-15		JS, VC, HL, EC, DW		Discussion and preparation of final version
2016-02-18		EC, HL, DW	4, 7, 9, 10, 11, 12	Last additions
2016-03-08		VC	11, 12	Final editing and change requests
2016-05-26		LW	9.4, 6.2	Added missing information on conformance requirements
2016-07-11		Scott Simmons	All	Preparation for final publication

1.5 Forward

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

2 References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

OGC 12-019, *OGC® City Geography Markup Language (CityGML) Encoding Standard*

ISO 19107:2003, *Geographic information -- Spatial schema*

SIG3D Modeling Handbook, http://wiki.quality.sig3d.org

Dutch CityGML implementation report, http://www.geonovum.nl/onderwerpen/3d-standaarden/documenten/3d-final-report

In addition to these documents, this report includes a Solid definition for GML features according to ISO 19197 as specified in Annex A.

3 Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the CityGML Encoding Standard [OGC 12-019] shall apply. In addition, the following terms and definitions apply.

3.1
Validation
Process of validating a CityGML data set against a specified set of requirements. A valid data set is conformant to these requirements.

3.2
Requirements
Rules and restrictions to define data structure and content unambiguously. Requirements can be derived from the CityGML Standard document or be defined separately as refinement of the CityGML standard to avoid ambiguities of the standard and/or specify further requirements (user/application dependent).

3.3
Check
Algorithmic implementation to check if a requirement is met in a validation software.

3.4

Error

Result of a check in case of non-conformance.

3.5

Validation Plan

Structured list of requirements, usually depending on use case.

4 Conventions

4.1 Abbreviated terms

2D	Two Dimensional
3D SIG	3D Special Interest Group Netherlands
3D	Three Dimensional
CAD	Computer Aided Design
CityGML	City Geography Markup Language
DTM	Digital Terrain Model
EuroSDR	European Spatial Data Research Organisation
GML	Geography Markup Language
IFC	Industry Foundation Classes
ISO	International Organization for Standardisation
LOD	Level of Detail
OGC	Open Geospatial Consortium
QIE	Quality Interoperability Experiment
SIG 3D	Special Interest Group 3D of Spatial Data Infrastructure Germany
TIN	Triangulated Irregular Network
UML	Unified Modeling Language
XML	Extensible Markup Language

5 Report overview

The report contains results and recommendations for specification of data quality requirements of CityGML data, validation of those data, and how to report validation results.

In section 6, general issues of requirements specification are discussed to introduce a structured order of data quality requirements with respect to CityGML data.

Based on that structured order, a standardized encoding system for the specification of these requirements is presented in section 6.3. Accordingly, validation results should be reported with the error codes specified in the section 7.5. Mappings of vendor-specific error codes of available validation tools are included.

Section 8 describes unit tests to assure correct performance of a validation tool according to the requirements specified.

Section 9 explains which experiments were performed in this CityGML quality interoperability experiment.

A general strategy for validation for typical use cases is outlined in Section 10.

Section 11 summarizes recommendations for conformance requirements.

Further steps towards standardized validation of 3D city models is outlined in Section 12.

6   Validation of CityGML data

With the wide adoption and use of CityGML as an international standard, the volume of 3D datasets has increased rapidly in tandem with the range of applications to which 3D is being applied (e.g. solar mapping, noise modeling, cadastre, etc.). The quality of the data being used in these forms of spatial analyses is of the utmost importance to the value of the outputs. However, evaluation of the quality of CityGML data has not received the attention it deserves in practice. This is partly because GML---the "basis" by which CityGML models geometric primitives---is a generic standard and allows a certain freedom of implementation, and partly because practice has shown that CityGML only offers limited guidance on how to uniformly and unambiguously implement the standard. While conformance requirements do exist for CityGML, they do not cover integrity checking of CityGML geometries. Furthermore, implementation specifications do not currently exist for 3D primitives (with the exception of the modeling handbook published by SIG 3D's data quality working group[1] and the 3D SIG in the Netherlands[2]). Due to these ambiguities, it is likely that in the future, different interpretations of CityGML implementation between customer and contractor will continue to occur.

If a CityGML document is validated with different validation software tools, it is essential that the validation results are consistent (i.e. they are similarly defined in each software): validation results from different software tools cannot be compared if this is not the case. Hence, a basic requirement for quality checks must be that defined validation rules lead to the same result in different software. Unified validation rules must be defined as derived from the requirements defined in the CityGML standard.

Geometric validation of 3D city models is important to ensure that they are conformant with user/application requirements. The aim of this quality experiment is to give clarity and better guidance to allow providers and users to come to a common understanding of what is required for a CityGML city model and how to specify and validate these requirements. A crucial point is to understand if a fundamental set of requirements which proves useful for most models does exist. It should be possible to validate these requirements to confirm that CityGML data are compliant.

---

[1] http://www.sig3d.org/index.php?catid=2&themaid=8777960&language=en

[2] http://www.geonovum.nl/onderwerpen/3d-standaarden/documenten/3d-final-report

These requirements should cover both geometric and semantic aspects of the data. As semantics are less strictly defined in the CityGML standard compared to geometry, the validation process yields results on the plausibility of semantic information rather than strict compliance statements.

Different views have to be considered for the aim of validation:

☐ Data capture view: make it easier to capture CityGML data and offer consistent data;

☐ Procurement view: make it easier to procure data capture and be able to check the consistency of the data delivered;

☐ Software view: make it easier for software to interpret, visualize, analyze and manipulate CityGML data; and

☐ Data management view: make it easier to maintain CityGML data over time .

Different themes of validation to test are defined, as follows:

☐ XML-Schema validation;

☐ Conformance based on formal and non-formal requirements in the CityGML standard document;

☐ Referential integrity (within a CityGML document as well as to external data sources—the latter would need to be available for the tests as well);

☐ Geometry; and

☐ Semantics / attribute constraints such as deviation of attribute "measured height" and the height of the LoD2 geometry.

The goals of the validation strategy designed in this experiment are to:

☐ Enable a common understanding about key requirements from the four views outlined above;
☐ Suggest elements of a best practice data specification; and
☐ Suggest sensible data tests and test tools as well as an overarching test plan that contains:
    o a set of modular, well defined ("low level") tests;
    o a way to define a ("high level") testbed (a selection of low-level tests, perhaps rule-based);
    o an agreed schema to test results both for "low level" and "high level" tests; and
    o well known sample data.

6.1 Validation Process

Data validation requires a clearly defined rule set. This rule set may differ for different use cases and has to be defined by the data user, possibly in cooperation with a technical expert. Rules should be derived from existing standards by interpretation, as described in the process model for conformance requirements (cf. section 9.4.4). Derived rules must be unambiguous and machine processable. The set of rules defines the validation plan.

6.2 Use cases and requirements for geometry

During the kick-off phase of the experiment a collection of standard use-cases was defined.

The specific questions to be answered were as follows.

- Is there a universal, reasonably generic set of CityGML requirements that should be specified in addition to the CityGML standard?

- What are the detailed quality requirements?

- How can these requirements be tested to ensure that they have been adhered to?

The generic use cases to be explored are the creation and maintenance of CityGML models for national and regional mapping including visualization, and analysis such as:

- Line of sight;

- Shading;

- Flooding;

- Aggregation of floorspace for buildings/sites;

- Energy demand simulation; and

- Scenario evaluation in urban planning.

The initial goal to give recommendations for respective use-cases could not be reached within the CityGML Quality IE. The editors agree that it would be beneficial to the community if there were guidelines on how to validate CityGML instance documents in order that those instances be of appropriate data quality and structure for a certain standard use case.

The task above can also be solved on a national level to allow for consideration of local and regional differences.

6.3 Validation framework

Besides the validation of instance documents against the tests from one single experiment, it will be necessary to validate against any number of requirements from any of the domains (geometric, semantic, conformance requirements, application specific requirements, schema conformance). For that purpose, a validation framework can be used. The concepts (as well as a prototype implementation) are developed in the Masters Thesis "Requirements for an Integral Testing Framework of CityGML Instance Documents" van Walstijn (2015). This validation framework builds on the concept of the unit-test (an atomic, isolated test). Unit-tests can cover any kind of test (geometric, semantic, schema, conformance, etc.) and always evaluate to either true or false. Figure 1 illustrates the unit-test concept.
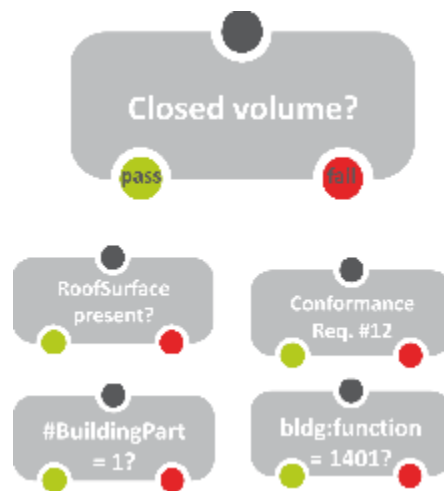
Figure 1: **Examples of individual unit-tests. Unit-tests can represent geometric, semantic, application specific or conformance requirement.**

These unit-tests can be implemented as binary code: for example, Java or C++. It is also possible to implement unit-tests which consist of an FME Workbench or even an XML or Schematron schema. Moreover, it is possible to define unit-tests based on software libraries such as CityDoctor or val3dity.

These individual unit-tests can be combined and chained in a workflow-description, an example workflow-description can be visualized as illustrated in Figure 2.
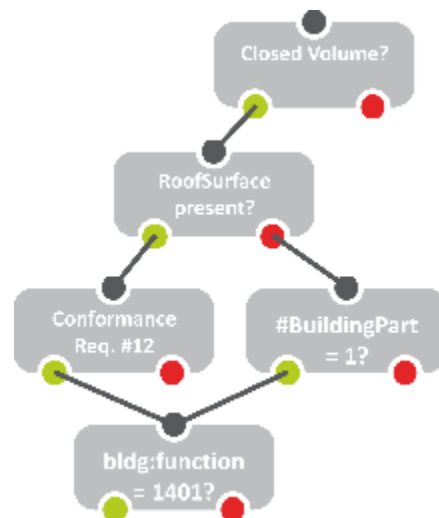
Figure 2 Chaining of unit-tests.

By connecting input and output ports of individual unit-tests, arbitrary validation plans can be built. Such a validation framework could be made accessible over an OGC WPS Interface (see Figure 3). Typically, a client sends an (XML) based validation plan together with a CityGML instance document to the WPS. The WPS in turn validates the instance document against the specified unit-tests based on the provided validation plan. A standardized error report is returned to the user. More information and details on how such a validation framework can be realized, the WPS interface is used, and test reports comply with ISO 19157 Geographic Information – Data quality, can be found in van Walstijn (2015).
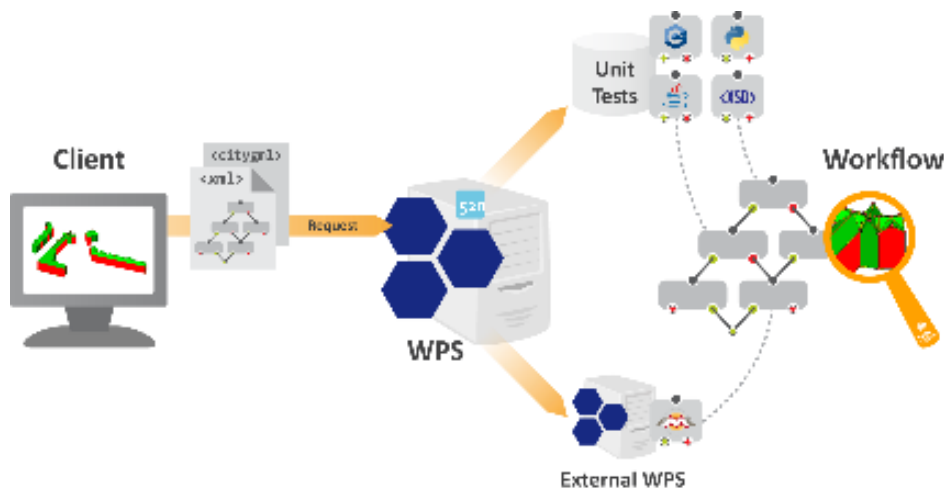
Figure 3: Architecture overview of a validation framework. Prototypically implemented in the virtualcity VALIDATOR.

9

7 Prerequisites

7.1 Schema Validation

Tools for XML Schema validation are available and give reliable results. Hence schema validation was not considered a major task for the CityGML QIE, although well-formed XML document data is required as input for most validation tools.

Schema related problems with the CityGML structure, such as the order of elements in sequences, were not discussed. For example, the order of child elements is different for _AbstractBuilding in LOD 2 and LOD 3, A change request for the next version of CityGML is suggested (cf Section 11.5).

7.2 Definition of a Solid according to ISO 19107

ISO 19107 defines geometric primitives for geodata. A Solid definition is derived from this standard to describe geometric validation criteria of volumetric CityGML objects in detail. The full definition can be found in Annex A.

7.3 System for encoding of data quality requirements

Suggestions how data quality requirements should be specified in CityGML or in addition to CityGML were discussed and evaluated. The following approaches were considered.

☐ Add data-quality based conformance requirements that are more prescriptive in a future version of the CityGML standard.

☐ Give additional guidance for implementation in a particular community in documentation separate from the CityGML standard (profiling, adding constraints, documenting conventions/best practice, implementers' agreements).

☐ Add additional metadata to CityGML modeling constructs used. This could be implemented in a future version of the CityGML standard or as an Application Domain Extension.

7.4 Specification of data quality requirements and requirement identifiers

For clarity, a validation criteria coding system needs to be defined. Note that every single validation criterion is called a requirement in this document.

The coding system uses the following structure:

XX-namespace:YY-ZZZZ

XX is the domain identifier. The following values are allowed:

value	description
SC	Schema Requirements
CO	Conformance Requirements
GE	Geometry Requirements
SE	Semantic Requirements
RI	Referential Integrity

YY is the element identifier. It defines the CityGML element upon which the requirement operates. The following values are allowed (NB: this is just a first step and needs to be extended in the future):

value	description
bldg:AB	Abstract Building
bldg:BU	Building (specific to Building in addition to Abstract Building)
bldg:BP	Building Part (specific to BuildingPart in addition to Abstract Building)
bldg:BS	BoundarySurface
bldg:WS	WallSurface
bldg:GS	GroundSurface
bldg:RS	RoofSurface
bldg:OFS	OuterFloorSurface
bldg:OCS	OuterCeilingSurface
gml:SO	Solid
gml:MS	MultiSurface
gml:PO	Polygon
gml:LR	Linear Ring

ZZZZ is the requirement identifier for the element in the given domain. These identifiers need to be defined by each working group. For example, for conformance requirements, the identifiers could simply be the number of the conformance requirement from the standard.

7.5 Error codes for geometric errors

In accordance with the QIE naming schema for the requirements, all the geometric errors are in the "GE" domain. Also, since the geometric validation is performed at 3 different levels (based on the primitives), the requirements also contain the level:

- Ring: R

- Polygon: P

- Shell: S

Error codes follow the same system as the requirements mentioned in section 7.4. A violation of each respective data quality requirement should result in an error which must be reported with an error code according to the requirement ID.

For each error code, extra information can (and should) be returned. For instance:

- if a ring is not closed (GE_R_NOT_CLOSED) then the ID of the ring (e.g., its position [first, second..] in the polygon) should be returned;
- if a polygon is not planar (GE_P_NON_PLANAR_POLYGON_DISTANCE_PLANE) then the ID of the polygon and the max. deviation from the reference plane should be returned; or
- if a shell is not 'watertight' (GE_S_NOT_CLOSED) then the location of the hole(s) should be returned.

In this QIE the following requirements are defined for the three respective geometric levels (adopted from Ledoux (2013):

RING level

- GE_R_TOO_FEW_POINTS (<3 points)
- GE_R_CONSECUTIVE_POINTS_SAME (2 consecutive points are the same)
- GE_R_NOT_CLOSED (first-last points are not the same)
- GE_R_SELF_INTERSECTION (self-intersects, i.e., a bowtie)
- GE_R_COLLAPSED (is point or line)

POLYGON level

- GE_P_INTERSECTION_RINGS (2+ rings intersect)
- GE_P_DUPLICATED_RINGS (2+ rings identical)
- GE_P_NON_PLANAR_POLYGON_DISTANCE_PLANE (with respect to tolerance)
- GE_P_NON_PLANAR_POLYGON_NORMALS_DEVIATION with respect to tolerance)
- GE_P_INTERIOR_DISCONNECTED (interior is not connected)
- GE_P_HOLE_OUTSIDE (1 or more interior rings are located outside the exterior ring)
- GE_P_INNER_RINGS_NESTED (interior ring is located inside other)
- GE_P_ORIENTATION_RINGS_SAME (exterior and interior rings have same orientation)

SHELL level

- GE_S_TOO_FEW_POLYGONS (<4 polygons)
- GE_S_NOT_CLOSED (there is 1+ hole(s) on the surface)
- GE_S_NON_MANIFOLD_VERTEX
- GE_S_NON_MANIFOLD_EDGE
- GE_S_MULTIPLE_CONNECTED_COMPONENTS (1+ polygons not connected to main shell)
- GE_S_SELF_INTERSECTION
- GE_S_POLYGON_WRONG_ORIENTATION (orientation of a polygon not correct)
- GE_S_ALL_POLYGONS_WRONG_ORIENTATION (normals all pointing in wrong direction)

7.6 Description of geometry checks

RING

GE_R_TOO_FEW_POINTS

A ring should have at least 3 points. For GML rings, this error ignores the fact that the first and the last point of a ring are the same (see GE_R_NOT_CLOSED), i.e., a GML ring should have at least 4 points.

For instance, this ring is invalid:

```
<gml:LinearRing>
  <gml:pos>0.0 0.0 0.0</gml:pos>
  <gml:pos>1.0 0.0 0.0</gml:pos>
```

```
  <gml:pos>0.0 0.0 0.0</gml:pos>
</gml:LinearRing>
```

GE_R_CONSECUTIVE_POINTS_SAME

Points in a ring should not be repeated (except first-last in case of GML, see GE_R_NOT_CLOSED). This error is for the common error where 2 *consecutive* points are at the same location. Error GE_R_SELF_INTERSECTION is for points in a ring that are repeated, but not consecutive.

For instance, this ring is invalid:

```
<gml:LinearRing>
  <gml:pos>0.0 0.0 0.0</gml:pos>
  <gml:pos>1.0 0.0 0.0</gml:pos>
  <gml:pos>1.0 0.0 0.0</gml:pos>
  <gml:pos>1.0 1.0 0.0</gml:pos>
  <gml:pos>0.0 1.0 0.0</gml:pos>
  <gml:pos>0.0 0.0 0.0</gml:pos>
</gml:LinearRing>
```

GE_R_NOT_CLOSED

This applies only to GML rings. The first and last points have to be identical (at the same location).

For instance, this ring is invalid:

```
<gml:LinearRing>
  <gml:pos>0.0 0.0 0.0</gml:pos>
  <gml:pos>1.0 0.0 0.0</gml:pos>
  <gml:pos>1.0 1.0 0.0</gml:pos>
  <gml:pos>0.0 1.0 0.0</gml:pos>
</gml:LinearRing>
```

GE_R_SELF_INTERSECTION

A ring should be *simple*, i.e., it should not self-intersect. The self-intersection can be at the location of an explicit point, or not.
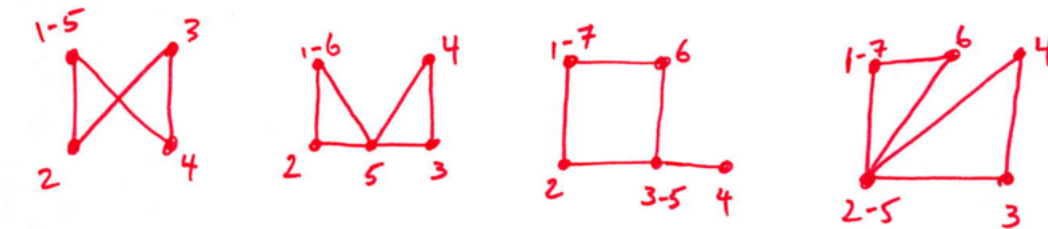
Figure 4: GE_R_SELF_INTERSECTION

GE_R_COLLAPSED_TO_LINE

A special case of self-intersection (GE_R_SELF_INTERSECTION): the ring is collapsed

to a line. If the geometry is collapsed to a point, then GE_R_TOO_FEW_POINTS / GE_R_CONSECUTIVE_POINTS_SAME should be used.
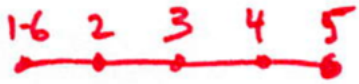
Figure 5: GE_R_COLLAPSED_TO_LINE

POLYGON

GE_P_INTERSECTION_RINGS

Two or more rings intersect, these can be either the exterior ring with an interior ring or only interior rings.

Figure 6: GE_P_INTERSECTION_RINGS

GE_P_DUPLICATED_RINGS

Two or more rings are identical.

GE_P_NON_PLANAR_POLYGON_DISTANCE_PLANE

A polygon must be planar, i.e., all of its points (used for both the exterior and interior rings) must lie on a plane. To verify this, we must ensure that the distance between every point and a plane is less than ϵ_1, a given *tolerance* (e.g., 1cm). This plane should be a plane fitted with least-square adjustment.

GE_P_NON_PLANAR_POLYGON_NORMALS_DEVIATION

To ensure that cases illustrated in Figure 7, below are detected (the top polygon is clearly non-planar, but would not be detected with GE_P_NON_PLANAR_POLYGON_DISTANCE_PLANE and a tolerance of 1cm for instance), another requirement is necessary: the distance between every point forming a polygon and *all* the planes defined by all possible combinations of 3 non-colinear points is less than ϵ_1. In practice this assessment can be implemented with a triangulation of the polygon (any triangulation): the orientation of the normal of each triangle must not deviate more than a certain user-defined tolerance ϵ_2 (e.g., 1

degree).
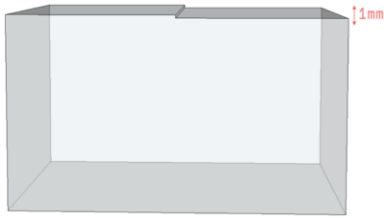
Figure 7: Tolerance for planarity

GE_P_INTERIOR_DISCONNECTED

The interior of a polygon must be connected. The combination of different valid rings can create such an error.
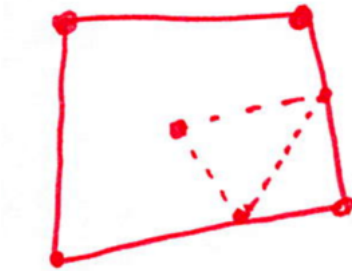
Figure 8: GE_P_INTERIOR_DISCONNECTED

GE_P_HOLE_OUTSIDE

One or more interior ring(s) is (are) located completely outside the exterior ring. If the interior ring intersects the exterior ring, then error GE_P_INTERSECTION_RINGS should be returned.
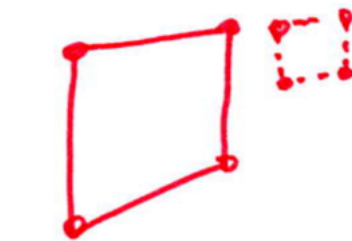
Figure 9: GE_P_HOLE_OUTSIDE

GE_P_INNER_RINGS_NESTED

One or more interior ring(s) is (are) located completely inside another interior ring.

Figure 10: GE_P_INNER_RINGS_NESTED

GE_P_ORIENTATION_RINGS_SAME

The interior rings must have the opposite direction (clockwise vs counterclockwise) when viewed from a given point-of-view. When the polygon is used as a bounding surface of a shell, then the rings have to have a specified orientation (see GE_S_POLYGON_WRONG_ORIENTATION / GE_S_ALL_POLYGONS_WRONG_ORIENTATION).
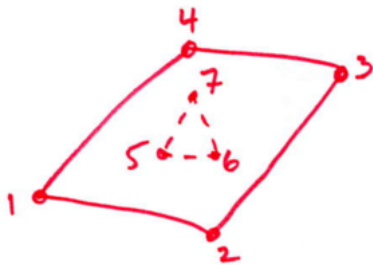
Figure 11: GE_P_ORIENTATION_RINGS_SAME

SHELL

GE_S_TOO_FEW_POLYGONS

A shell should have at least 4 polygons - the simplest volumetric shape in 3D is a tetrahedron.

GE_S_NOT_CLOSED

The shell must not have 'holes', i.e., it must be 'watertight'. This refers only to the topology of the shell, not to its geometry (see GE_S_SELF_INTERSECTION).

In Figure 12, the left solid is invalid while the right one is valid (since the hole is filled with other polygons).
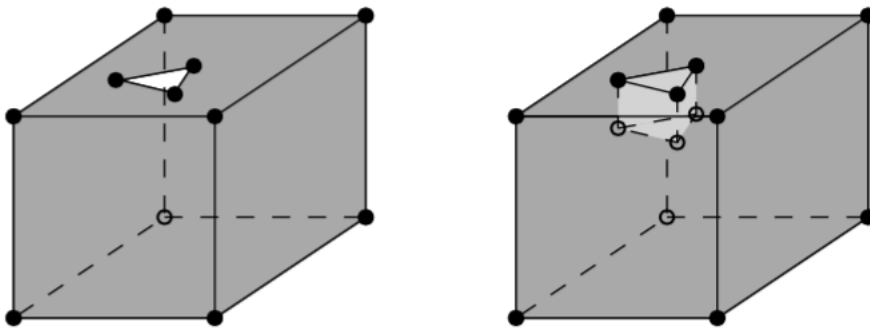
Figure 12: GE_S_NOT_CLOSED

GE_S_NON_MANIFOLD_VERTEX

Each shell must be simple, i.e. it must be a 2-manifold. A vertex is non-manifold when its incident polygons do not form one `umbrella.'
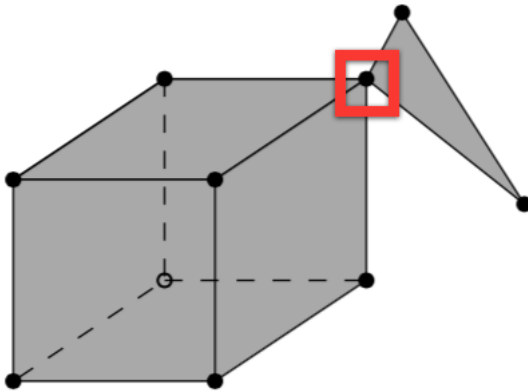
Figure 13: GE_S_NON_MANIFOLD_VERTEX

GE_S_NON_MANIFOLD_EDGE

Each edge of a shell should have exactly 2 incident polygons.
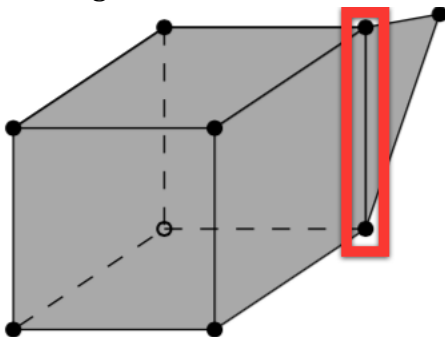
Figure 14: GE_S_NON_MANIFOLD_EDGE

GE_S_MULTIPLE_CONNECTED_COMPONENTS

Polygons that are not connected to the shell should be reported as an error.
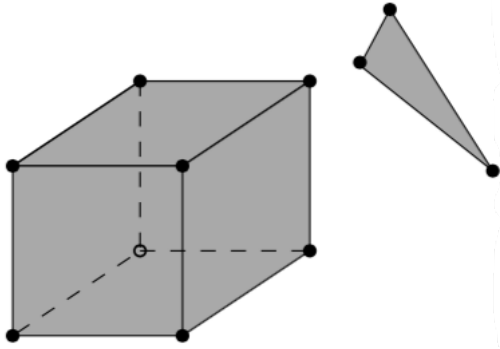
Figure 15: GE_S_MULTIPLE_CONNECTED_COMPONENTS

GE_S_SELF_INTERSECTION

If topology of the shell is correct and the shell is closed (thus no error GE_S_TOO_FEW_POLYGONS / GE_S_NOT_CLOSED / GE_S_NON_MANIFOLD_VERTEX / GE_S_NON_MANIFOLD_EDGE / GE_S_MULTIPLE_CONNECTED_COMPONENTS), it is possible that the geometry introduces other errors, e.g., intersections. For instance, the topology of both shells in Figure 16 is identical, but the geometry differs. The left shell is valid while the right one is invalid.
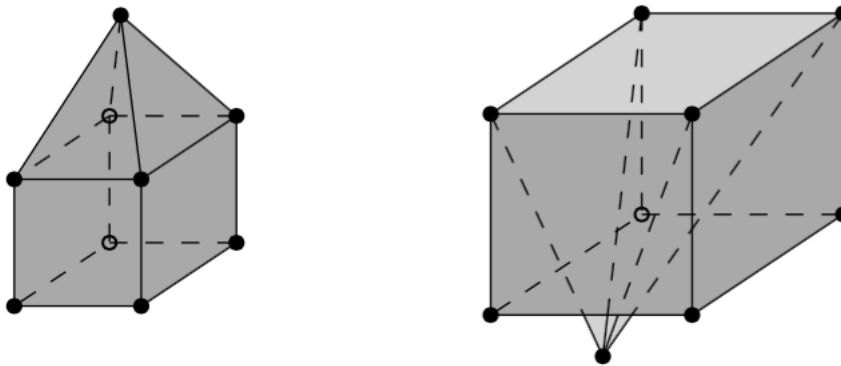
Figure 16: GE_S_SELF_INTERSECTION

GE_S_POLYGON_WRONG_ORIENTATION

If one polygon is used to construct a shell, its exterior ring must be oriented in such a way that when viewed from outside, the shell points are ordered counterclockwise.

GE_S_ALL_POLYGONS_WRONG_ORIENTATION

Where all the polygons have the wrong orientation (as defined in GE_S_POLYGON_WRONG_ORIENTATION), i.e. they all point inwards.

7.7 Implementation of error codes and references to existing error code systems in test tools

Some of the existing validation tools return vendor-specific error codes. A mapping of these custom error codes to the error codes agreed upon in the CityGML QIE is given in the table below to enable comparison of validation results.

Table 1: Check and Error Identifiers

	QIE naming convention	CityDoctor	FME	val3dity
1	GE_R_TOO_FEW_POINTS	CP_NUMPOINTS	Too few points	101
2	GE_R_CONSECUTIVE_POINTS_SAME	CP_DUPPOINT	Duplicate Consecutive Points	102
3	GE_R_NOT_CLOSED	CP_CLOSE	Polygon not closed*	103
4	GE_R_SELF_INTERSECTION	CP_SELFINT	Self-Intersections in 2D	104
5	GE_R_COLLAPSED_TO_LINE	CP_NULLAREA	Duplicate Consecutive Points in 3D, Degenerate or Corrupt Geometries, Self-Intersections in 2D **	105
6	GE_P_INTERSECTION_RINGS	+	Self-Intersections in 2D - Donut: Overlapping Rings	201
7	GE_P_DUPLICATED_RINGS	CS_SELFINTNATIVE	Self-Intersections in 2D - Donut: Duplicate Rings, Donut: Touching Rings	202
8	GE_P_NON_PLANAR_POLYGON _DISTANCE_PLANE	CP_PLANNATIVE	Non-Planar Surfaces	203
9	GE_P_NON_PLANAR_POLYGON _NORMALS_DEVIATION	CP_PLANTRI	Non-Planar Surfaces	204
10	GE_P_INTERIOR_DISCONNECTED	+	Self-Intersections in 2D - Donut: Disjoint Interior	205
11	GE_P_HOLE_OUTSIDE	+	Self-Intersections in 2D - Donut: Hole Outside Shell	206
12	GE_P_INNER_RINGS_NESTED	+	Self-Intersections in 2D - Donut: Nested Hole	207
13	GE_P_ORIENTATION_RINGS_SAME	+	Invalid Solid Boundaries, Invalid Solid Voids - Surface Not Closed; Invalid Solid Boundaries	208
14	GE_S_TOO_FEW_POLYGONS	CS_NUMFACES	Invalid Solid Boundaries, Invalid Solid Voids - Not Enough Faces	301
15	GE_S_NOT_CLOSED	CS_OUTEREDGE	Invalid Solid Boundaries; Invalid Solid Voids - Surface Not Closed	302
16	GE_S_NON_MANIFOLD_VERTEX	CS_UMBRELLA	Invalid Solid Boundaries, Invalid Solid Voids - Dangling Faces	303
17	GE_S_NON_MANIFOLD_EDGE	CS_OVERUSEDEDGE	Invalid Solid Boundaries, Invalid Solid Voids - Surface Not Closed	304
18	GE_S_MULTIPLE_CONNECTED _COMPONENTS	CS_CONCOMP	Invalid Solid Boundaries, Invalid Solid Voids - Free Faces	305
19	GE_S_SELF_INTERSECTION ***	CS_SELFINT	Invalid Solid Boundaries,	306

			Invalid Solid Voids - Surface Self Intersects	
20	GE_S_POLYGON_WRONG _ORIENTATION	CS_FACEORIENT	Invalid Solid Boundaries, Invalid Solid Voids - Dangling Faces	307
21	GE_S_ALL_POLYGONS_WRONG _ORIENTATION	CS_FACEOUT	Invalid Solid Boundaries, Invalid Solid Voids - Surface Normals Bad Orientation	308

Comments

Citydoctor[3]

(1) CP_DUPPOINT detects consecutive and non-consecutive duplicate points. The later may be detected by other software as self-intersection (e.g., FME).
(2) CS_SELFINTNATIVE covers different types of intersections of several polygons within one solid. GE_P_DUPLICATED_RINGS is just one special case. CS_SELFINTNATIVE will be split into several error types to reflect this in future versions.
(3) Checks marked with a + sign are implemented after the CityGML QIE and will be available in future versions.

FME

* FME's GML reader automatically closes unclosed polygons. To detect errors such as '103.gml polygon not closed,' datasets were read with FME's XML reader and searched for LinearRings to see if there are any that are not closed.
** Some tests were uniquely mappable using a truth table applied to the output of the GeometryValidator within the test FME workspace. For example, test 105 was the only test that produced the FME errors: 'Duplicate Consecutive Points in 3D, Degenerate or Corrupt Geometries, Self-Intersections in 2D' which was detected and uniquely mapped to 'Ring collapsed into a line.' This could be more conclusively characterized by an FME workspace that detects ring features that have length but no area.
*** A number of the tests datasets (cf. Section 8) provided had more than one variant per unit test. Consequently, results varied depending on which variant was tested. For example, test 306 had 4 variants. Variant i306_3.gml was 'unit cube with one extra face inside another face' and yielded the FME error: 'Invalid Solid Boundaries, Invalid Solid Voids;' Detail: 'Free Faces.' Variant i306_4.gml was 'torus where the hole in the top/bottom faces touches the side surfaces' and yielded an FME Error: 'Self-Intersections in 2D' Detail: 'Donut: Self-Intersection Error.' The 306 unit test used in the above table was the result from testing i306_1.gml 'house with tip below the ground.'

---

[3] Wagner, D., Alam, N., Wewetzer, M., Pries, M., & Coors, V. (2015). METHODS FOR GEOMETRIC DATA VALIDATION OF 3D CITY MODELS. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *40*(1), 729.

CityServer3D

CityServer3D utilizes the CityDoctor library to implement geometric validity checks as part of a rule-based system. Accordingly, the CityDoctor error codes are used to report errors to the user, and CityDoctor restrictions apply.

val3dity

val3dity[4] returns both the number and the QIE string. For this, the "--qie" flag must be used.

8 Unit tests

All test models are listed in Annex B.

8.1 TU Delft

Unit tests were developed for geometric errors by 3D GeoInformation Group, TU Delft. Each GML file contains one and only one solid, and (ideally) a maximum of 1 error. A brief description is provided for each file:

- *i101_1.gml;cube with top face having only 2 points; and*

- *i102_1.gml;cube with one duplicate vertex (repeated in a ring).*

There are 3 "types" of files:

1. vXXX.gml: solid is valid;

2. iXXX_Y.gml: solid is invalid, the reason is "XXX" (see below for the codes); and

3. tXXX_Y.gml: validity is based on a tolerance; 1E-Y is the amount by which one vertex is moved, thus for t203_1.gml one vertex was moved by 10cm.

The codes for the errors are in accordance with the QIE naming schema for the requirements. For each error code, extra information can (and should) be returned. For instance:

- if a ring is not closed (GE_R_NOT_CLOSED) then the ID of the ring (e.g., its position [first, second..] in the polygon);
- if a polygon is not planar (GE_P_NON_PLANAR_POLYGON_DISTANCE_PLANE) then the ID of the polygon should be returned; or

---

[4] http://geovalidation.bk.tudelft.nl/val3dity/ and https://github.com/tudelft3d/val3dity

☐   if a shell is not 'watertight' (GE_S_NOT_CLOSED) then the location of the hole(s) should be returned.

The test data is available on https://github.com/tudelft3d/CityGML-QIE-3Dvalidation.

8.2   SIG3D

Five test data sets for three different cases are provided:

1) Test Case "Addresses;"

2) Test Case "Generic Attributes;" and

3) Test Case "Geometry."

Data available from
http://en.wiki.quality.sig3d.org/index.php/CityGML_2.0_Examples_and_Test_Data

8.3   HFT

15 models with geometric errors plus 15 models with non-geometric errors are provided by HFT Stuttgart.

Data is available here:

http://citydoctor.hft-stuttgart.de/pwiki/index.php/Specification_of_Test_Data

9   Experiments

The IE addressed the following experiments:

☐   Experiment #1: Schema validation ;

☐   Experiment #2: 3D geometry validation: test tools and ability to visualize/analyze in a number of clients;

☐   Experiment #3: Semantic and attribute consistency testing: test tools and ability to visualize/analyze in a number of clients ; and

☐   Experiment #4: Conformance requirements testing: test tools and ability to validate the conformance requirements that are within the test capability of the test tools.

9.1   Experiment 1: Schema validation

A valid CityGML document is expected to adhere to the XML Schema as defined by the standard. Thus, a valid XML structure is a prerequisite before any other validation step should be performed. Commercial tools and web-based services for schema validation are available (Oxygen, XML-Spy etc.). They are well suited to perform this step of the validation process, hence there is no need for detailed investigation of schema validation within the frame of the CityGML QIE.

9.2 Experiment 2: Geometry validation

Geometry for CityGML features must comply with the underlying geometry model of GML. The general use of geometry features is further restricted by the CityGML standard, where additional rules and recommendations for correct deployment are given. Nevertheless, the same physical geometry can be modeled in different ways and still be valid according to the open formulations in the standard. Thus, it might be necessary for the user to restrict these possibilities to one or more valid alternatives. The next section shows one simple building geometry which is modeled in alternative ways.

9.2.1 Modeling alternatives

Geometry can be modeled in different ways (e.g., Solid or MultiSurface, xLinks etc.). There are at least 10 different ways to structure a simple building in LOD 2, see Table 2.

Table 2: Modeling alternatives for a simple building in LoD 2

Image	Name	Description
The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.	SimpleSolid_SBS	A simple building modeled in CityGML as Solid geometry with the Boundary Surfaces exposing MultiSurface geometry *(conflict with building conformance requirement 10.3.9.4 (see CityGML 2.0 spec, 10.3.9, page 78))*
The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.	SimpleSolid_SBSx	A simple building modeled in CityGML as Solid geometry with the Boundary Surface tags existing but with empty geometry
The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.	SimpleSolid_Sx	A simple building modeled in CityGML as Solid geometry without Boundary Surfaces

	SimpleSolid_SBSref	A simple building modeled in CityGML as Solid geometry with the Boundary Surfaces referencing via xlinks the Solid geometry *(conflict with building conformance requirement 10.3.9.4 (see CityGML 2.0 spec, 10.3.9, page 78))*
	SimpleSolid_SrefBS	A simple building modeled in CityGML as Solid geometry which references via xlinks the MultiSurface geometry of the Boundary Surfaces
	SimpleSolid_xBS	A simple building modeled in CityGML using only Boundary Surfaces
	SimpleSolid_MSBS	A simple building modeled in CityGML as MultiSurface geometry with the Boundary Surfaces exposing MultiSurface geometry also *(conflict with building conformance requirement 10.3.9.4 (see CityGML 2.0 spec, 10.3.9, page 78))*
	SimpleSolid_MSBSref	A simple building modeled in CityGML as MultiSurface geometry with the Boundary Surfaces referencing via xlinks the MultiSurface geometry *(conflict with building conformance requirement 10.3.9.4 (see CityGML 2.0 spec, 10.3.9, page 78))*
	SimpleSolid_MSrefBS	A simple building modeled in CityGML as MultiSurface geometry which references via xlinks the MultiSurface geometry of the Boundary Surfaces

	SimpleSolid_MSx	A simple building modeled in CityGML as MultiSurface geometry without Boundary Surfaces
	SimpleSolidOverhangs_SrefBS	A simple building with overhangs modeled in CityGML as Solid geometry which references via xlinks the MultiSurface geometry of the Boundary Surfaces *(geometric error as the geometry is in fact not a solid geometry: overused edges)*
	SimpleSolidOverhangs_MSrefBS	A simple building with overhangs modeled in CityGML as MultiSurface geometry which references via xlinks the MultiSurface geometry of the Boundary Surfaces
	SimpleSolidOverhangs_SMSrefBS	A simple building with overhangs modeled in CityGML as Solid (includes only the faces that can form a solid) and MultiSurface (includes only the overhangs) geometry which references via xlinks the MultiSurface geometry of the Boundary Surfaces

All variants are conformant to the CityGML 1.0 and CityGML 2.0 schema. Some validate the conformance requirements given in the standard document. The models can be downloaded from here:
http://citydoctor.hft-stuttgart.de/pwiki/index.php/Specification_of_Test_Data

SIG3D has discussed variations in geometry modeling previously, which led to the publication of a modeling handbook with focus on German needs. This handbook considers the most common modeling alternatives and makes recommendations for deployment. The modeling handbook is available from the website of SIG3D:

Part 1: Basics (Rules for Validating GML Geometries in CityGML):
http://files.sig3d.org/file/ag-
qualitaet/201311_SIG3D_Modeling_Guide_for_3D_Objects_Part_1.pdf

Part 2: Modeling of Buildings (LoD1, LoD2, LoD3):
http://files.sig3d.org/file/ag-
qualitaet/201311_SIG3D_Modeling_Guide_for_3D_Objects_Part_2.pdf

Recommendation:

Buildings should be modeled as *Solid* elements, only roof overhangs as *Multisurface*
elements. In addition, *BoundarySurfaces* should contain the actual geometry, which is
then referenced by the *Solid* element.

9.2.2 Validation results

The results of validation reports of the unit tests, where provided by specific software
tools, are available at https://github.com/tudelft3d/CityGML-QIE-
3Dvalidation/tree/master/results.

A. val3dity (TU Delft)

The tool developed by TU Delft, called val3dity, is freely available under an open-source
license (GPL v3). The code can be obtained at https://github.com/tudelft3d/val3dity but it
is easier to use web interface at http://geovalidation.bk.tudelft.nl/val3dity as there is
nothing to install: just upload a CityGML file and get back a detailed report.

The validation of the solids is performed according to the international standard ISO
19107, and the tool is compliant to this standard.

Results: All the valid unit tests are validated as such, and all the ones containing errors
are also reported as invalid.

However, sometimes a different error is reported. For instance, val3dity cannot report
error 105--GE_R_COLLAPSED and instead reports 104--
GE_R_SELF_INTERSECTION, which is correct but less accurate (collapsing implies
self-intersection). Also, errors 303--GE_S_NON_MANIFOLD_VERTEX and 304--
GE_S_NON_MANIFOLD_EDGE cannot be differentiated and thus only 303 is reported.
Also, when tolerances are involved, the errors returned can differ since a fixed tolerance
was used for the testing.

Test Order:

DANGLING_FACES
VERTICES_NOT_USED
FREE_FACES
SURFACE_SELF_INTERSECTS
SURFACE_NOT_CLOSED
This means that if a 'DANGLING_FACES' error is encountered, that error code is
returned and testing is terminated. So subsequent checks for VERTICES_NOT_USED
and FREE_FACES will not be performed. Thus test order determines the error reported.
As the test order is somewhat arbitrary, as long as the results are specific enough to

explain the problem with the source data, they should be considered valid. A general overview about the order of checks is given in Figure 17.
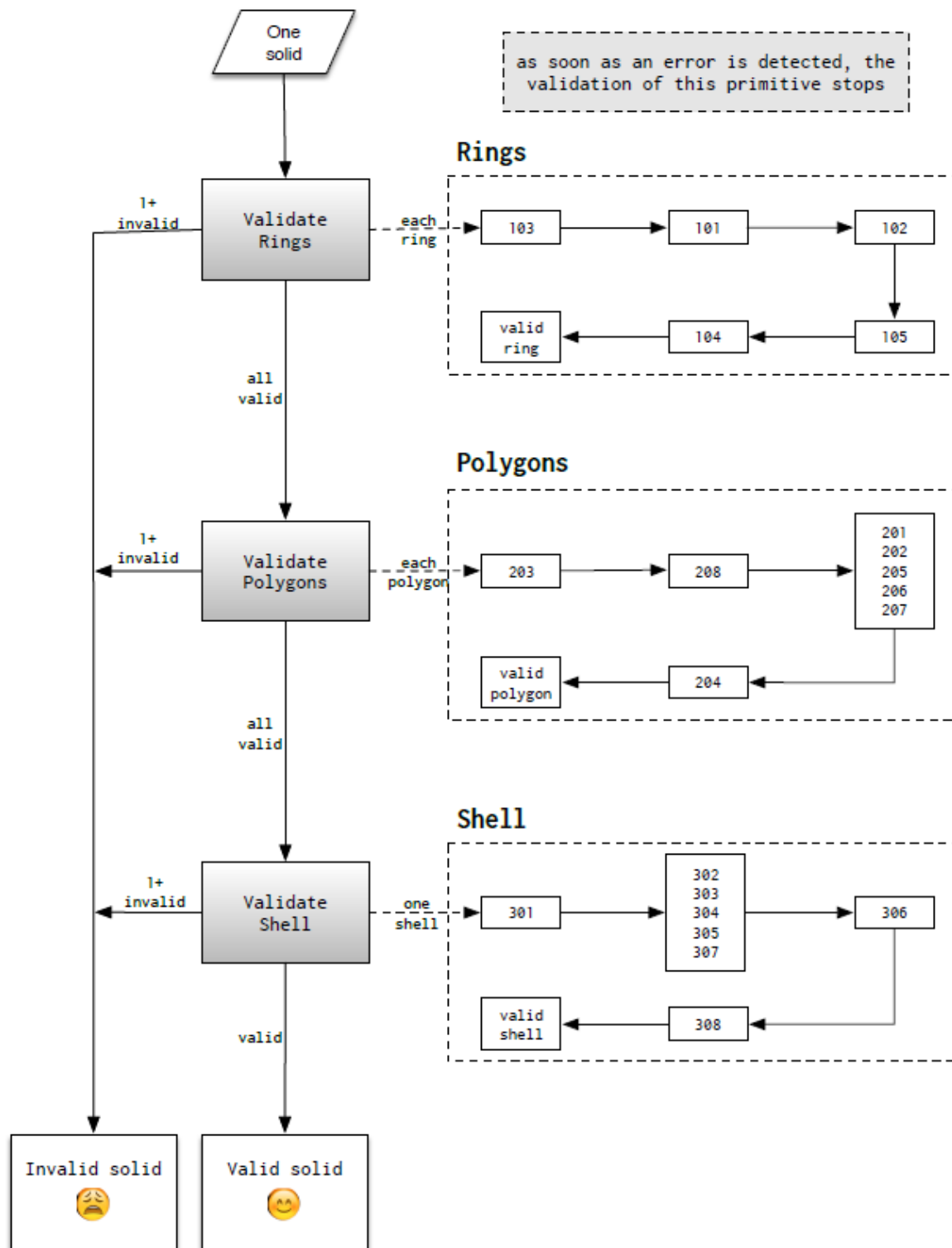
Figure 17: Execution order of geometry checks in val3dity

The above problem can be mitigated by chaining successive GeometryValidators in a workspace. This allows each GeometryValidator to focus on a specific test. For this

project, one GeometryValidator was used for each major group of tests (one for 100 level, one for 200 level etc.).

For any checks that the GeometryValidator cannot yet handle, there are typically approaches using other FME transformers that suffice. A separate process was used to encode checks for too few points for the ring test (101.gml) and to compute zero volume. It should be noted that FME contains many tools for geometry and schema manipulation and validation besides GeometryValidator.

Many readers tend to do a certain amount of data cleaning or interpretation which can make error detection difficult. FME's GML reader automatically closes unclosed polygons. To detect errors such as '103.gml polygon not closed,' datasets were read with FME's XML reader and searched for LinearRings to see if there are any that are not closed.

There are a wide range of bad geometries that are not present in the unit tests provided for the QIE. In other contexts, datasets have been observed that contain null sub elements, such as solids with null faces, or coordinate lists with NaN values etc. All the unit tests here were in an arbitrary x/y/z Cartesian space. There are problems that can arise during validation with datasets that are in geographic coordinates such as LL-WGS84 etc., and during conversion between spherical and projected coordinates. Also, the unit test datasets provided did contain geometry instances, including invalid or missing links to reference geometries. Finally, most datasets were contrived for tests composed of a single or a few small features designed to focus on one issue. More testing needs to be done on production scale datasets with more complex geometries and textures.

B. CityDoctor

CityDoctor was developed at HFT Stuttgart as result of a research project from 2010 to 2013. Results of the project can be found at http://citydoctor.hft-stuttgart.de. The geometry checks are described in (Wagner, Alam, Wewetzer, Pries, & Coors, 2015).

The tool transforms CityGML data sets into an internal data model which did not consider inner rings at that time (future versions have this included to be conformant with ISO 19107).

CityDoctor is continuously improved and maintained and will be available from http://www.citydoctor.eu.

Results: As the current implementation did not support inner rings, some errors returned are false positives, or, not detected. There's a "cascading effect" that causes solids containing polygons with inner rings to be returned as invalid although they are valid according to the definition in section 7.2. The tolerance used for the validation of planarity is 0.01 m and 0.01 rad resp.

The checks are performed in a hierarchical order (which is roughly reflected in the requirements section 7.4). This means that for certain checks to be executed, other lower-level checks must be passed as a prerequisite. A solid is only considered valid if all required checks are passed. The dependencies are listed in Table 4.

Table 3: Check dependencies for geometry checks of CityDoctor ValidationTool

	R1	R2	R3	R4	R5	P1/2	S1	S2	S3
R1 CP_NUMPOINTS									
R2 CP_CLOSE									
R3 CP_DUPPOINT	X	X							
R4 CP_NULLAREA	X	X	X						
R5 CP_SELFINT	X	X	X	X					
P1 CP_PLANNATIVE	X	X	X	X					
P2 CP_PLANTRI	X	X	X	X					
P3 *GE_P_INTERSECTION_RINGS*									
P4 *GE_P_INTERIOR_DISCONNECTED*									
P5 *GE_P_HOLE_OUTSIDE*									
P6 *GE_P_INNER_RINGS_NESTED*									
P7 *GE_P_ORIENTATION_RINGS_SAME*									
P8 GE_P_DUPLICATED_RINGS*									
S1 CS_NUMFACES	X	X	X	X	X				
S2 CS_SELFINT	X	X	X	X	X	X	X		
S3 CS_OUTEREDGE	X	X	X	X	X				
S4 CS_OVERUSEDEDGE	X	X	X	X	X				
S5 CS_FACEORIENT	X	X	X	X				X	
S6 CS_FACEOUT	X	X	X	X		X	X	X	X
S7 CS_CONCOMP	X	X	X	X				X	
S8 CS_UMBRELLA	X	X	X	X				X	

Checks in italics were not part of CityDoctor Validation tool at time of QIE. Will be implemented with the next version.
* GE_P_DUPLICATED_RINGS is integrated as part of CS_SELFINT

C. Safe FME

Safe Software produces FME – Feature Manipulation Engine – a toolset for supporting spatial data integration and harmonization. For this project, Safe provided FME to validate the 3D geometry unit tests. Most validation was conducted with the tools built into the GeometryValidator transformer. The diagnostic report on all the unit datasets was auto-generated as an Excel spreadsheet using an FME workspace. The code used is based on val3dity for the 3D part. The validation of the 2D primitives (of the planes forming a solid for instance) is done using other FME methods.
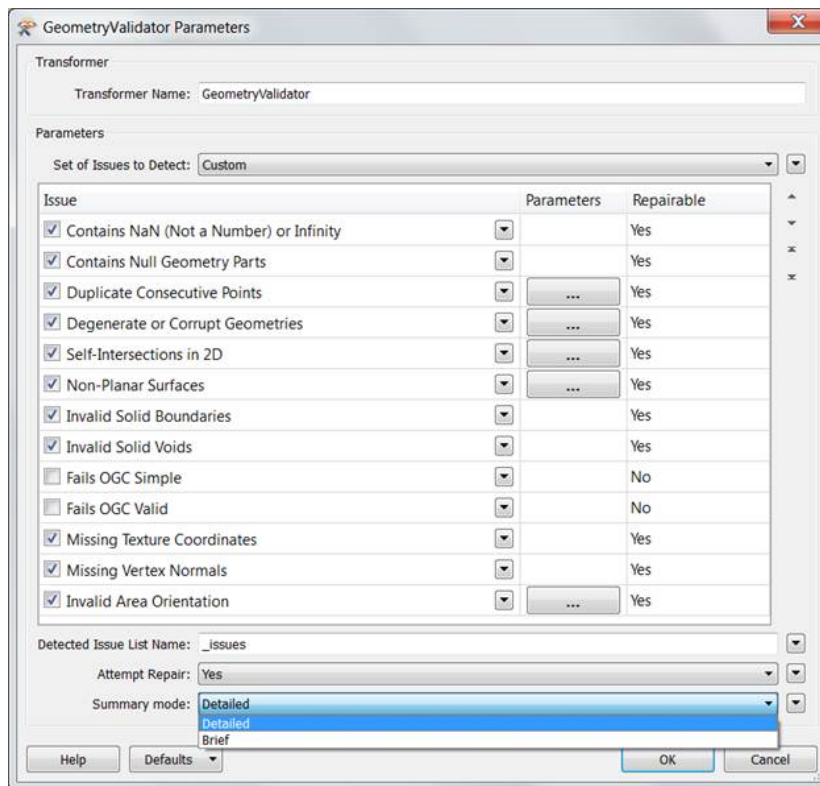
Figure 18: GeometryValidator transformer used to validate 2D and 3D geometries for QIE

Significant improvements were made to GeometryValidator over the course of the project, to provide more granular results and add new tests where needed. Before, FME would only generate a general error message such as 'degenerate geometry', or 'self intersection' for each error feature. Now supplementary_info items are added such as 'too few points'. The error features have x,y,z coordinates so that the exact location of the error can be tracked.

Many new tests added to FME relate to the 200 level polygon and shell unit tests. Additional tests have been added for solids, with work ongoing. Several solid problems are reported as invalid solids, but unit tests 303, 304 and 307 did not produce results directly relatable to the QIE descriptions. For 303, 304, while FME did correctly flag these features as invalid, FME not identify the specific errors related to manifold edges or vertices. For 307, FME did not identify the face orientation problem, but generated the error 'Invalid Solid Boundaries - Dangling Faces' instead. Also, 204 produced the same error as 203 and so is not readily distinguishable.

In all, about 90% of the test results produced messages equivalent to QIE descriptions. Overall, FME was able to detect geometry problems in 100% of the unit tests provided, whether or not the actual error messages exactly matched the QIE descriptions.

In some cases, validity depended on tolerances. Tolerances used include absolute distance of 0.0005 units, and angular tolerance of 5 degrees. These were applied

primarily to the planarity checks (thickness and surface normal respectively). The distance tolerance was also used to assess endpoint accuracy for degenerate geometries.

One of the challenges is the order the tests are performed. In FME, when a specific test fails, often further tests are ignored.

D. CityServer3D

The Department for Spatial Information Management at Fraunhofer IGD produces the CityServer3D package, a 3D geospatial database with CityGML support. CityServer3D is capable of executing rules against spatial data stored in the server or loaded into its supporting "AdminTool" (the main client). In this fashion, a large amount of spatial data (e.g., from a city) can be checked in one run.

The execution of rules on the database has been used to perform CityDoctor checks on the QIE unit-tests. The results deviate from CityDoctor as CityServer3D is missing inner ring support in faces and generally imposes internal validity constraints up front.

There are currently a few outliers, as given in the following table.

Table 4: CityServer3D Test Results (outliers only)

Test case(s)	Expected result	Actual result
i204_*, i308_*	Invalid 204/308	valid
I201, i202, i205, i206, i207, i208	Invalid polygon (2xx)	Invalid ring (101)
I305_2	Invalid	Valid
I303_1, i303_2	Invalid (303)	Invalid (302)
I306, i305_1, i304,	Invalid (with certain error)	Invalid (different errors)
V005	Valid	Invalid (102)
V011, v013	Valid	Invalid (302)
V012, v014	Valid	Invalid (101)

9.2.3 Validation test conclusions

In summary, the results show that there is not yet consistent error handling amongst validation tools. In general, basic validity scores much better than precision of the error(s) reported. This could be addressed by clearly specifying which errors are more

accurate than others, so a report could demote less accurate errors if more accurate ones exist on the primitive.

The CityServer's approach to combining checks in a rule-based system has worked well and will probably continue to do so when more parameterization, e.g., the thresholds required for some checks, are available in the CityDoctor API.

9.3   Experiment 3: Semantic validation

CityGML contains an independent semantic model, which usually references geometric elements accordingly. CityDoctor is the only tool in the context of CityGML QIE that provides semantic checks. Table 5 lists available checks which have been tested during the QIE.

Validation of semantics can be described in different ways, as follows.

- Sole validation of the semantic model against a rule set which has to be derived from the referenced thematic model. This would include tests for pure semantic elements, for example if a *BuildingPart* is always a child element of a *Building*. Checks to validate the relationship of *Building* and *BuidingPart* elements are proposed for CityDoctor[5].This is not part of this QIE, as there currently are no implementations of this concept.

- Semantic validation in the context of 3D city models using additional information (e.g., from cadastre) or limited to plausibility checks of semantic entities with relation to geometry. Results of the latter are considered plausible if they are coherent with the geometry (Stadler 2007). This type of semantic validation is discussed in the following paragraphs.

- Semantic validation in the context of 3D city models using additional information (e.g., from cadastre) or limited to plausibility checks of semantic attributes with relation to geometry. These checks can often be substituted by schema validation using proper rule sets. Checks for mandatory attributes and the valid range of attribute values will be available in the next version of CityDoctor. Rule sets can in many cases be defined with concepts such as Schematron or GeoSparQL.

Table 5: List of semantic checks.

BoundarySurface
SE-bldg:RS-FN (Orientation of RoofSurface)
SE-bldg:WS-FN (Orientation of WallSurface)

---

[5] http://citydoctor.hft-stuttgart.de/pwiki/index.php/Building-Pr%C3%BCfungen

SE-bldg:GS-FN (Orientation of GroundSurface)
SE-bldg:OFS-FN (Orientation of OuterFloorSurface)
SE-bldg:OCS-FN (Orientation of OuterCeilingSurface)

_AbstractBuilding

SE-bldg:AB-ASA   (Comparison sum of story heights with height of building geometry)
SE-bldg:AB-AMH   (Comparison attribute value with height of building geometry)

Building structure

SE-bldg:WS-PL (WallSurface contains connected polygons with similar normal vectors)
SE-bldg:RS-PL (RoofSurface contains connected polygons with similar normal vectors)
SE-bldg:GS-PL (GroundSurface contains connected polygons with similar normal vectors)

Description of checks

SE-bldg:RS-FN

The face normal of a *RoofSurface* has a positive z-direction.

SE-bldg:WS-FN

The face normal of a *WallSurface* has a zero z-coordinate (within a tolerance).

SE-bldg:GS-FN

The face normal of a *GroundSurface* has a negative z-direction.

SE-bldg:OFS-FN

The face normal of an *OuterFloorSurface* has a positive z-direction.

SE-bldg:OCS-FN

The face normal of an *OuterCeilingSurface* has a negative z-direction.

SE-bldg:AB-ASA

The sum of all storey heights of the attribute *storeyHeightsAboveGround* is equal to the total height of the building geometry (within a tolerance).

SE-bldg:AB-AMH

The attribute value of *measuredHeight* is equal to the total height of the building geometry (within a tolerance).

SE-bldg:WS-PL

A *WallSurface* is composed of several connected polygons with the same face normal direction (within a tolerance).

SE-bldg:RS-PL

A *RoofSurface* is composed of several connected polygons with the same face normal direction (within a tolerance).

SE-bldg:GS-PL

A *GroundSurface* is composed of several connected polygons with the same face normal direction (within a tolerance).

A synthetic model with randomly assigned BoundarySurface types[6] was useful to validate the applicability of this approach. 1795 BoundarySurface elements which are not coherent with the geometry, i.e., their face normal direction, were found in 394 out of 900 buildings of the LOD2 model. As the total number of errors in this model is not exactly known, further research concerning the stability of the algorithms is necessary.

This strategy could also be used for the implementation of certain checks for Conformance Requirements (cf. Experiment 4, section 9.4).

9.4 Experiment 4: Validation of Conformance Requirements

Conformance requirements (CRs) are defined in the CityGML standard for a number of topics. They describe necessary conditions or restrictions for rules of standard definitions and are present in many parts of the CityGML Standard document. From these editors' point of view, the importance of testing conformance requirements follows in importance

---

[6] Biljecki, F., Ledoux, H., & Stoter, J. (2014). Error propagation in the computation of volumes in 3D city models with the Monte Carlo method. ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., II-2, 31–39. http://doi.org/10.5194/isprsannals-II-2-31-2014

just behind checking the syntax of a model. Only now is there a formal way to check a CityGML document's compliance.

The aim of Experiment 4 of the QIE was to analyze the conformance requirements in the standards document and to find an automatic and formal correction mechanism for testing data against the conformance requirements of CityGML.

The strategy for this experiment can be described as follows. First, all conformance requirements of the standard were collected and classified according to the agreed common coding system. To facilitate the discussion on the transformation of conformance requirements into testable statements and keep formal track of the discussion during the QIE meetings, a workflow was developed.

☐ The conformance requirement is split into several citations. This is necessary because conformance requirements are generally multiple sentences long and make various logical statements which need to be considered separately.

☐ Next, individual citations are analyzed and interpretations made. These interpretations do not need to be mutually consistent and are used as input for the discussion. Interpretations are the first non-trivial step towards formalizing the conformance requirements. This step has proven to be important because it made the differences in the understanding of the citations very clear and showed the conflicts between different interpretations.

☐ Based on the discussion of the interpretations, definitions are formed. Each definition should be fully unambiguous and follow the line of the interpretation from which it came.

☐ Finally, each definition should ideally be accompanied by a formalization. These formalizations contain the same information as the definitions, however they are expressed in a formal language.

This exercise collected 128 conformance requirements from the standard, which can be broken down as shown in Table 6.

Table 6: CityGML 2.0 Conformance Requirements

# of CR's	Module of CityGML 2.0	Conformance Requirements in Detail
9	**CityGML Core module**	**CO-core:Core-001 (CityModel Containment)** **CO-core:Core-002 (external References)** **CO-core:Core-003 (addresses)** **CO-core:Core-004 (implicit geometry part of Core module)** **CO-core:Core-005 (Xlink for city objects)** **CO-core:Core-006 (Xlink for addresses)** **CO-core:Core-007 (implicit geometries used in extensions)** **CO-core:Core-008 (MIME Type)** **CO-core:Core-009 (Xlink for implicit geometry)**

12	Appearance module	not part of QIE
32	Bridge module	not part of QIE
28	**Building module**	**CO-bldg:BU-001 (Building --- BuildingPart)** **CO-bldg:BU-002 (lod0FootPrint and lod0RoofEdge)** **CO-bldg:BU-003 (lodXSolid and lodXMultiSurface)** **CO-bldg:BU-004 (boundedBy)** **CO-bldg:BU-005 (lodXMultiCurve)** **CO-bldg:BU-006 (outerBuildingInstallation)** **CO-bldg:BU-007 (outerBuildingInstallation - boundedBy)** **CO-bldg:BU-008 (opening)** *CO-bldg:BU-009 (interiorRoom)* *CO-bldg:BU-010 (interiorBuildingInstallation)* *CO-bldg:BU-011 (interiorBuildingInstallation - boundedBy)* **CO-bldg:BU-012** **CO-bldg:BU-013** **CO-bldg:BU-014** **CO-bldg:BU-015** **CO-bldg:BU-016** **CO-bldg:BU-017** **CO-bldg:BU-018** **CO-bldg:BU-019** **CO-bldg:BU-020** **CO-bldg:BU-021** **CO-bldg:BU-022** **CO-bldg:BU-023** **CO-bldg:BU-024** **CO-bldg:BU-025** **CO-bldg:BU-026** **CO-bldg:BU-027** **CO-bldg:BU-028** **Note: Conformance requirement 9 - 11 deal with LoD4 and are not part of the QIE**
1	CityFurniture module	not part of QIE
3	CityObjectGroup module	not part of QIE
3	Generics module	not part of QIE
1	LandUse module	not part of QIE
4	Relief module	not part of QIE
4	Transportation module	not part of QIE
25	Tunnel module	not part of QIE
1	Vegetation module	not part of QIE
5	WaterBody module	not part of QIE

QIE Experiment 4 was limited to the Core and Building modules. The remaining modules will be completed within the SIG3D Quality Working Group. The complete results will be published by the SIG3D after completion.

9.4.1 Interpretation of conformance requirements

Initially, a requirement was analyzed if it was correctly applicable to the domain of this experiment. In case of missing application domains like a geometry-oriented CR, this CR should be defined in the geometry test experiment instead of another CR experiment to ensure that each rule is only tested once.

In an attempt to find an exact formulation for testing, a standardized test documentation was created for each CR which consists of:

☐ a complete citation and decomposition of the CR by splitting up complex formulations into small isolated pieces;

☐ an analysis of the created isolated pieces and, if necessary, a further decomposition until basic, but exact formulations are found;

☐ an interpretion and classification of basic rules into `mandatory/optional/obsolete;`

☐ the transformation of basic rules into a formal language: Schematron was chosen as an appropriate language for this purpose;

☐ validation of CR; and

☐ creation of test data for CR.

As an example of this process, consider the conformance requirement CO-bldg:BU-003 found in the CityGML 2.0 document on page 78 section 10.3.9(3) dealing with the use of lodXSolid and lodXMultiSurface in buildings:

3. The *lodXSolid* and *lodXMultiSurface*, X ∈ [1..4], properties (*gml:SolidPropertyType* resp. *gml:MultiSurfacePropertyType*) of *_AbstractBuilding* may be used to geometricllay represent the exterior shell of a building (as volume or surface model) within each LOD. For LOD1, either *lod1Solid* or *lod1MultiSurface* must be used, but not both. Starting from LOD2, both properties may be modelled individually and complementary.

Figure 19: Conformance Requirement 10.3.9(3) according to CityGML 2.0

This conformance requirement is split into the following citations:

A. The lodXSolid and lodXMultiSurface, X ∈ [1..4], properties (gml:SolidPropertyTyperesp. gml:MultiSurfacePropertyType) of AbstractBuilding may be used to geometrically represent the exterior shell of a building (as volume or surface model) within each LOD.

B. For LOD1, either lod1Solid or lod1MultiSurface must be used, but not both.

C. Starting from LOD2, both properties may be modelled individually and complementary

These citations lead us to the following interpretations.

[I.1] Citation A is already part of the schema and does not declare anything new.

[I.2] Citation B: if lod1Solid is used in a building, lod1Multisurface cannot be used or vice versa.

[I.3] Citation C: For LoDx, x ∈ [2..4], either one of the geometry representations (solid | multisurface) may be used, or both. This is already part of the schema. So it forms no additional requirement.

Therefore, the only remaining rule (I.2) leads us to the following definition:

[D.1] From [I.2] : a Building or BuildingPart element shall not consist of both an lod1Solid property and an lod1MultiSurface property.

This can be transformed as follows to Schematron:

```
[D.1] a Building or BuildingPart element shall not consist of
both an lod1Solid property and an lod1MultiSurface property

                              ↓

<rule context="bldg:BuildingPart | bldg:Building">
   <report test="bldg:lod1Solid and bldg:lod1MultiSurface">
    <name/> with id:<value-of select="@gml:id"/> consists of both Solid
    and MultiSurface LoD1 geometry, this is not allowed.
   </report>
</rule>
```

Figure 20: Mapping of a definition to Schematron (adapted from van Walstijn (2015))

All other rules of the CityGML Core and Building module are analyzed in a similar way. The resulting documents can be found in the Wiki page for this experiment or in the annex of this document.

Test data and examples are provided or will be provided later on the Wiki page of this experiment.

Parameter	Value
Req Id	CO-bldg:BU-003
Short Desc	(lod0FootPrint and lod0RoofEdge)
Citation	**(A)** The *lodXSolid* and *lodXMultiSurface*, X ∈ [1..4], properties (*gml:SolidPropertyType* resp. *gml:MultiSurfacePropertyType*) of *_AbstractBuilding* may be used to geometrically represent the exterior shell of a building (as volume or surface model) within each LOD. **(B)** For LOD1, either *lod1Solid* or *lod1MultiSurface* must be used, but not both. **(C)** Starting from LOD2, both properties may be modelled individually and complementary.
Source	CityGML 2.0⊞, 10.3.9 (3), p. 78
Interpretation	**[I.1]** (A): This is already part of the schema **[I.2]** (B): if lod1Solid is used, lod1MultiSurface cannot be used or vice versa **[I.3]** (C): For a given LoD, X ∈ [2..4]): Either one of the geometry representations (solid | multisurface) may be used or both, which is already part of the schema
Definition	**[D.1]** from [I.2]: if lod1Solid is used, lod1MultiSurface cannot be used or vice versa

Formalization	**[F.1]** from [D.1]:
	
	 Schematron:
	
	 `<schema xmlns="http://purl.oclc.org/dsdl/schematron⊡" queryBinding="xslt2">`
	 `    <ns uri="http://www.opengis.net/gml⊡" prefix="gml"/>`
	 `    <ns uri="http://www.opengis.net/citygml/building/2.0⊡" prefix="bldg"/>`
	 `    <pattern>`
	 `        <rule context="bldg:BuildingPart | bldg:Building">`
	 `            <report test="child::bldg:lod1Solid and child::bldg:lod1MultiSurface">`
	 `                <name/> with id:<value-of select="@gml:id"/> consists of both Solid and MultiSurface LoD1 geometry, this is not allowed.`
	 `            </report>`
	 `        </rule>`
	 `    </pattern>`
	 `</schema>`

Exceptions	--
Precondition	--
Expected Behavior	--
OGC URN	--
Return Code	RET_CO-bldg:BU-003
Classification	Error
CityGML Version	2.0
Example	
Comments	HFT Validated by CityDoctor SEM_LOD1_ASSOLID KIT (2014-09-08) Sentences (S1) and (S3) are not necessary
Status	[D.1]: accepted, mandatory

Figure 21 Wiki page example

9.4.2   Conformance Requirements – General findings

Some of the conformance requirements have not been described exactly in a formal language, but are formulated colloquially. This causes the following problems.

☐ Undefined terminology
Several CRs use terminology that is not properly defined. For example, the base requirement makes a distinction between buildings consisting of one homogeneous part and buildings consisting of individual structural segments. However, a clear, unambiguous explanation which formally distinguishes both cases from another is missing. Therefore, it is impossible to formally validate this requirement. Another example is the use of the expression *LoD* in the CRs. In many requirements, formulations like "From LoD X, ..." or "This element should not be used for LoD X" are used. This can at least be interpreted in three different ways. Besides an intuitive overview of what each LoD consists of, there is no formal description what constitutes exactly an LoD. This makes it rather difficult to formalize and validate CRs which depend on such a vocabulary. Because the main purpose of CRs is to express testable statements to which the data should

41

conform, all terminology used in the formulation of a CR should be unambiguous and precisely defined.

☐ Modeling guidelines.

Closely related to the previous category is the problem that a number of CRs consist of modeling guidelines. An example from BU-006: "BuildingInstallation elements shall only be used to represent outer characteristics of a building which do not have the significance of building parts." Such statements are very difficult to test, because "significance" is not formally defined. Another example is citation D of BU-008 which states that the embrasure surface of windows or doors shall belong to the BoundarySurface element and not the Opening element. Such statements specify how reality should be mapped to CityGML elements, which can never be validated incontestably and are of the type: "bridges should not be modeled as Building element." Such modeling guidelines can only be inspected manually because there is no objective way of checking that the contents of a Building element, is in fact a building. Such guidelines could be formulated in a best practice document which describe how adopted OGC documents shall be used or implemented.

☐ Incomplete formulation

Some CRs appear not be to complete. An example is the CR concerned with the Room element. BU-009 states that the Room element should only be used from LoD4. Accordingly, the Room element only consists of lod4Solid and lod4MultiSurface properties to reference geometry. Besides these direct properties, the Room element can also reference boundary surface elements to semantically classify the walls of the room. This boundary surface element has also an lod2MultiSurface- and lod3MultiSurface property besides the lod4MultiSurface property. It seems logical to exclude the use of LoD2 and LoD3 property elements although it is not explicitly stated in the CR. A similar situation has been observed for BU-005.

☐ Redundancy

Quite a number of CRs contain an introductory statement which is obvious from the text in the thematic module itself and the UML diagram. Also, some CRs contain redundant information. That is, they contain information that is already formulated in another CR. To reduce any ambiguities about the content of CRs, redundancy should be avoided.

This discussion shows that when the above mentioned points are not considered, it is difficult to create validation tests from the CRs. This is problematic since CRs have the purpose to be used as a test of the validity of instance documents. In fact, a CR is defined by the OGC as an "expression in the content of a document conveying criteria to be fulfilled if compliance with the document is to be claimed and from which no deviation is

permitted". Furthermore, it is noted that: "Anything without a defined test is a priori not testable and thus would be better expressed as a recommendation." [7]

For that purpose, it is suggested that each CR should be stated as a logical material condition with a structure as shown in the following figure.

Figure 22: Proposed structure of a conditional sentence consisting of 4 structural elements for formulating conformance requirements. (adapted from van Walstijn (2015))

 The protasis consists of a condition on a subject and formulates unambiguously the situation in which the CR should apply. The apodosis or consequence consists of a combination of object, modal and predicate. The modal is used to express the level of strictness between the object and the predicate and takes the form of a modal verb. Naturally the exact meaning of these modal verbs should be defined. In fact, The OGC has defined the following modal verbs to be used for OGC standards.

- **Shall** verb form used to indicate a requirement to be strictly followed to conform to this standard, from which no deviation is permitted.

- **Must** equivalent to shall.

- **Should** verb form used to indicate desirable ability or use, without mentioning or excluding other possibilities.

- **May** verb form used to indicate an action permissible within the limits of this standard.

- **Can** verb form used for statements of possibility.

Furthermore, whenever subject and object are not one and the same, the relation between them should be made explicit. With this structure, it should be possible to formulate CRs that are free from any ambiguity. Nonetheless each CR should be always checked to ensure that all vocabulary in either the condition, the predicate, or specification of object or predicate is fully unambiguous and leaves no room for interpretation. During the specification of CRs, it should always be considered that a software implementation of the CR needs to have a boolean domain. So the evaluation of a CR on test data shall always result in either *true* or *false*.

---

[7] C Reed. Policy Directives for Writing and Publishing OGC Standards: TC Decisions. OGC Doc 06-135r11

Also, it is possible to specify the CRs as a Schematron schema. Such schemas could then be accompanied by natural language to clarify the rule. By making these Schematron schemas leading or normative, all ambiguity would be removed from the CRs.

All CRs are identified by the common coding system agreed in the QIE as *CO-namespace:YY-ZZZZ*, e.g. `CO-bldg:BU-004` for the fourth CR regarding buildings in the *building* namespace.

The editors recommend a return coding and classification system for testing CRs which is directly linked to the underlying test *as RET-CO-namespace:YY-ZZZZ, classes Error/Warning*, e.g. `RET-CO-bldg:BU-002 class Error`

Requirements Coding System

value	Description
CO - ns:YY - ZZZZ	Conformance Requirement
XX - **ns:YY** - ZZZZ	CityGML Modules / Domain
XX - ns:YY - ###	The requirement by number as found in CityGML 2.0

Figure 23: Requirements Coding System

9.4.3   Validation results

virtualcityVALIDATOR

As part of the Master thesis "Requirements for an Integral Testing Framework of CityGML Instance Documents" by Lucas van Walstijn (2015), a validation framework has been prototypically implemented which is based on the concepts of the validation framework described in chapter 6.1 Validation framework. This Web Processing Service (WPS) based tool has been used to validate CityGML instance documents against all the QIE developed Schematron schemas. This prototype formed the basis for the virtualcityVALIDATOR, a product of virtualcitySYSTEMS GmbH.
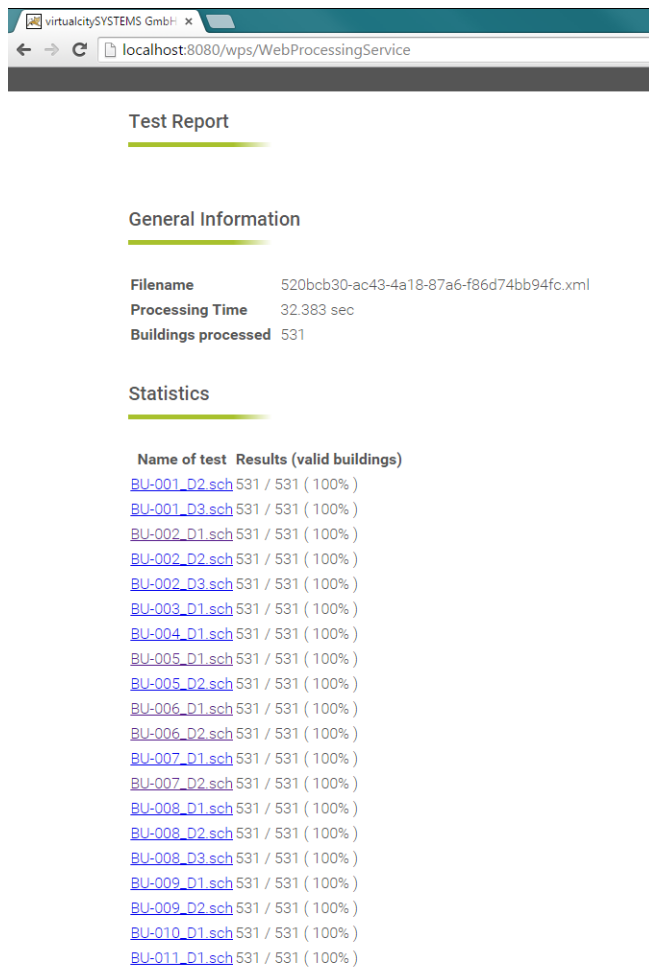
Figure 24: Screenshot of a test-report created by the virtualcityVALIDATOR. Input data is taken from the Berlin 3D download portal. (adapted from van Walstijn, (2015))

9.4.4   Process model

All tasks based on the previous explanations of the formulation of test conditions for CRs can be formalized and described as a standardized process, which contains the elements in Figure 25.
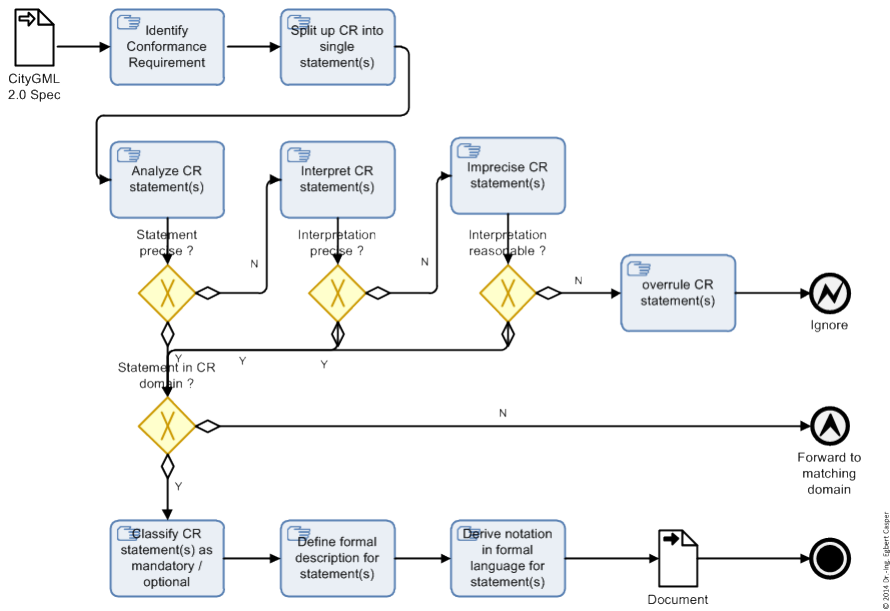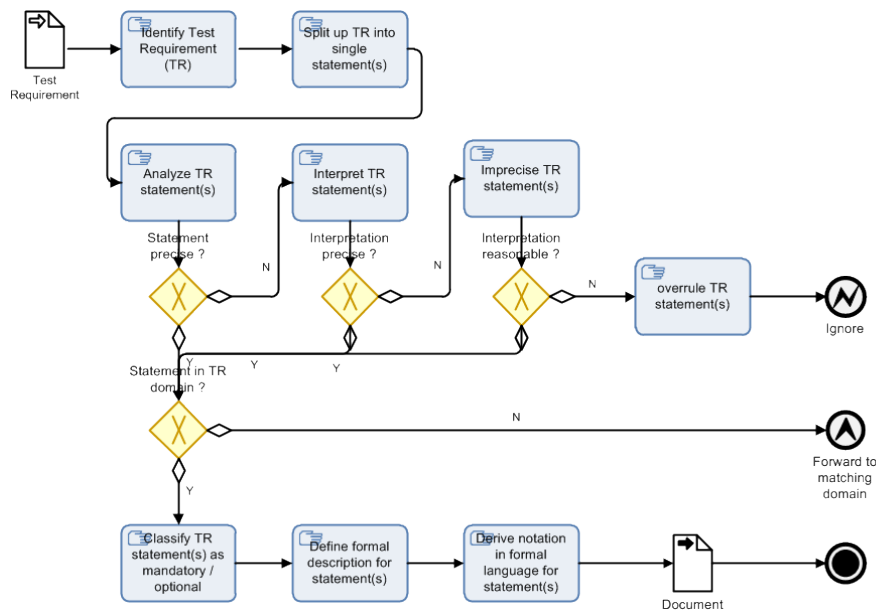
Figure 25: BPMN process model for analyzing conformance requirements

The following steps have to be performed:

1. Identify and extract CRs in a given document (in this case the CityGML 2.0 specification);

2. Split up complex formulations into small isolated pieces;

3. Analyze isolated pieces and split up further parts until basic but exact formulations are found;

4. Rate and classify testing formulations (mandatory / optional / obsolete);

5. Transform testing formulations into a formal language; and

6. Create a document containing at least all steps of transformation.

Upon closer inspection of the figure above, one will notice that this process model can be generalized for all kinds of tests in content of CityGML, e.g., geometry tests or semantic tests and a generalized process model can be denoted as shown in Figure 26.

Figure 26: BPMN process model for analyzing general tests

One lesson learned in experiment 4 of the QIE is that CRs can actually be formalized and thus tested.

As a consequence of this experiment, for future versions of CityGML, it is recommended that a much more precise formulation of the CRs be made. Furthermore all elementary definitions of CityGML should be defined explicitly, so that no interpretation is needed: e.g., what is a building and what is a buildingPart?

10 Use cases and requirements for geometry

During the kick-off phase of the experiment, a collection of standard use-cases was defined.

The specific questions to be answered were:

☐ Is there a universal, reasonably generic set of CityGML requirements that should be specified in addition to the CityGML standard;

☐ What are the detailed quality requirements to be specified; and

☐ How can these requirements be tested to ensure that they have been adhered to?

The generic use cases to be explored are the creation and maintenance of CityGML models for national and regional mapping including visualization and analysis such as:

☐ Line of sight;

☐ Shading;

☐ Flooding;

☐ Aggregation of floorspace for buildings/sites;

☐ Energy demand simulation; and

☐ Scenario evaluation in urban planning.

The initial goal to give recommendations for respective use-cases could not be reached within the CityGML Quality IE. The project participants agreed that it would be beneficial to the community if there were guidelines on how to validate CityGML instance documents in order to be of appropriate data quality and structure for a certain standard use case.

This task can also be solved on a national level to allow for consideration of local and regional differences.

11 Conclusion and Recommendations

This project has come a long way towards refining both testing methods and unit test datasets in the area of both CityGML, specifically and 3D geospatial data, in general.

The QIE team made great strides in bringing together a wide range of users from a variety of 3D communities to promote collaboration in the area of validating CityGML and 3D data. This is in itself a great accomplishment, since work in the area of developing common standards around validation, specifically as related to CityGML, has been lacking. More work needs to be done to refine the unit tests and better document those tests in terms of validity status, error type, error priority, etc. Even basic graphical depictions of what the error actually looks like, developed in the course of the project, were very helpful to promote discussion and common understanding of the issues at play.

11.1   Recommendations for geometry and semantics

CityGML as an open standard allows many different modeling alternatives of one and the same geometric structure (cf. Section 9.2.1). Validation of a given geometry is not possible without clear specification of the requirement, e.g. whether a Solid geometry or MultiSurface geometry is considered valid, or even both. In a validation plan, all aspects which are not restricted in an unambiguous way by the CityGML standard must be clearly specified with sufficient details.

This report recommends introducing the concept of tolerance for the geometric validation of objects. Neither ISO19107 nor OGC GML address this issue, but this experiment suggests that validation can only be performed when a tolerance is defined.

This is especially true for polygon planarity (i.e., all the points of a polygon forming a surface of a solid must lie on a plane), which is a requirement.

This reports proposes three requirements for tolerance.

1. The distance between every point forming a polygon and a plane is less than a given tolerance e_1 (eg 1mm). This plane should be a plane fitted with least-square adjustment.

2. The distance between every point forming a polygon and all the planes defined by all possible combinations of 3 non-collinear points is less than e_1. This is to ensure that surfaces having a very small 'fold' are detected.

3. A tolerance for the snapping of input vertices should be defined. Solids stored in GML are modelled with very little topological relationships. For instance, all six surfaces of a cube are stored independently, and thus the coordinates (x,y,z) of a single point (where 3 surfaces 'meet') is stored 3 times. For the validation, one needs to identify that these 3 points are the same, and thus a tolerance should be used.

Most important is the method for modeling geometric features. The editors recommend that buildings should be modeled as *Solid* elements, only roof overhangs should be *Multisurface* elements. In addition, *BoundarySurfaces* should contain their actual geometry, which is then referenced by the *Solid* element.

To enable comparison of validation results from different tools, it is necessary to define check algorithms in a consistent and comparable way. Furthermore, the order of execution of interdependent geometric checks can influence the validation result. It is thus required to execute the checks in a standardized order. Recommendations for this order are given in section 9.2.2.

Semantics are an important design concept of CityGML. Semantics and geometry must be coherent. Semantic validation is thus focused on the relationship of semantics and geometry. For example, a *RoofSurface* element has a face normal which is directed "upwards." This example shows that the validation criteria is rather imprecise: the validation result can be regarded as the plausibility of a geometric feature having certain well defined semantics. Exceptions might occur, and the angular range of "upwards" must be set according to users' needs.

Another aspect of semantic validation is be the plausibility check of attributes with relation to geometry, such as *measuredHeight*. For the validation results, the same considerations have to be made as stated above.

In some cases, it is difficult to separate semantics from conformance requirements. However, the same approach as described in Section 9.3 can be used to derive semantic restrictions from the CityGML standard, which might be useful for further development.

CityDoctor was the only tool which provided semantic validation during the QIE. The current implementation considered checks of *BoundarySurface* orientation and attributes with relation to geometry. Within a tolerance, these semantic elements can be validated against their geometry. It is important to define a suitable tolerance and keep in mind that exceptional cases can still cause a validation error, although the geometric-semantic coherency was not violated. These cases have to be inspected manually.

11.2 General recommendations for conformance requirements

☐ Each CR should be verified through the CR process model as described in Figure 25. Reaching the "ignore" exit leads to an invalid CR which may be ignored.

☐ Each CR should be formulated precisely. Colloquial formulations should be avoided, otherwise a translation or transformation into a set of exact formulations is needed.

☐ Each CR should be uniquely associated to a module or domain (for example `building`) and should not span across domains (neither technically, geometrically, semantically, nor schematically).

☐ If a basic definition used in a CR is ambiguous, it must be defined in a dictionary. This basic definition should be used consistently in all CRs.

☐ These recommendations should be applied to existing, future and additional CRs.

11.3 Recommendations for existing conformance requirements

The following CRs or parts of a CR of the CityGML 2.0 building module are classified against the general recommendations.

Conformance Requirement	Deals with	Sentences	Status	Basic Requirement	Classification
`co-bldg:BU-001`	Building - BuildingPart	(A) (B) (C)	`valid` `valid` `valid`	(D1) (D2) (D3)	mandatory optional mandatory
`co-bldg:BU-002`	lod0FootPrint and lod0RoofEdge	(A) (B)	`valid` `valid`	(D1) (D2) (D3)	mandatory mandatory mandatory

`co-bldg:BU-003`	lodXSolid and lodXMultiSurface	(A)
		(B)
		(C)
			`invalid`
			`valid`
			`invalid`
				(D1)	mandatory

`co-bldg:BU-004`	boundedBy	(A)
		(B)
		(C)
			`valid`
			`invalid`
			`valid`
				(D1)
				(D2)
					mandatory
					(D2) difficult to translate into schematron

`co-bldg:BU-005`	lodXMultiCurve	(A) (B)	valid invalid	(D1)	optional
`co-bldg:BU-006`	outerBuildingInstallation	(A) (B) (C)	valid invalid invalid	(D1)	optional
`co-bldg:BU-007`	outerBuildingInstallation - boundedBy	(A) (B)	invalid valid	(D1)	mandatory
`co-bldg:BU-008`	opening	(A) (B) (C) (D)	valid invalid invalid invalid	(D1)	optional

Hence, the invalid parts are recommended as to be treated as obsolete.

It is recommended that an official change request should be submitted if either a CR is classified as invalid but meaningful or desirable or optional but necessary.

11.4 Recommendations for future conformance requirements

It is recommended for the development of CityGML 3.0 that each CR should be built in accordance with the general recommendations for CRs as described in chapter 10.2.

11.5 Change requests

☐ The order of child elements is different for _AbstractBuilding in LOD 2 and LOD 3. For a building the sequence <lod2Solid> … </lod2Solid> followed by <boundedBy> …</boundedBy> is correct, while <lod3Solid> … </lod3Solid> followed by <boundedBy> …</boundedBy> is invalid. The sequence has to be the other way around in this case (<boundedBy> …</boundedBy> followed by <lod3Solid> … </lod3Solid>) due to the definition in the schema. This is confusing and also often not taken into account by the software generating the models. As a consequence, most LOD3 building models do not validate against the schema as the order of elements is wrong. This report recommends to change the order of elements: put the <boundedBy> element either before the <lod0FootPrint> element or before the <consistsOfBuildingPart> element.

☐ The editors recommend introducing the concept of tolerance for the geometric validation of objects. Neither ISO19107 nor OGC GML address this issue, but this experiment suggests that validation can only be performed when a tolerance is defined.

☐ Tolrance is especially important for polygon planarity (i.e., all the points of a polygon forming a surface of a solid must lie on a plane) which is a requirement in CityGML (for details, see section 11.1).

☐ Though there are some ways to model a CityGML building by using Solids or MultiSurfaces, a volume model of a building should be modeled solely using

gml:Solid geometry elements and should meet the requirements of a solid geometry.

☐ If BoundarySurfaces are defined in the model, they contain the actual geometry as defined in CR 10.3.9-4 of the CityGML 2.0 standard document. In addition, it is strongly recommended that the building geometry as Solid element referencing the Boundary Surface geometry must be mandatory.

☐ Change CR 10.3.9-1 as "main part" of a building is very difficult to validate. A building with buildingParts containing the entire geometry shall be valid.

12 Next steps

Further improvement of the validation process is required. Tools available to date do not deliver the identical validation results when seemingly deploying the same checks. Differences in implementation, algorithms, and tolerances or the order of check execution can cause this divergence. Stronger definitions of checks could help to solve this problem. However, there is a general need to certify/validate the validation tools. Strategies to guarantee unified and certified validation results should be worked out in future. This process should include a discussion of numeric accuracy in order to avoid instable computations and/or random results to floating point errors. Specific future work should include:

- Unification of workflows;

- Harmonization of specification of validation plans;

- Harmonization of error codes (concept is in the report, but just as a recommendation);

- Define profiles / restrictions for specific applications;

- Focus on buildings from the QIE, then extend to other features;

- Interaction with different features; and

- Repair / healing.

It should be noted that GeometryValidator as well as CityDoctor have a repair mode, and that many problems can be repaired. While repairs can themselves introduce new problems, the results of automated repairs can be revalidated. This can be helpful for managing large volumes of 3D data.

Another area worth exploring is deploying validation and repair as a web service such as via REST, WFS-T or WPS. With FMEServer, any workspace that accepts or produces XML/GML/CityGML can be deployed as a data streaming service.

More work needs to be done to refine unit tests and better document those tests in terms of validity status, error type, and error priority. For example, basic graphical depictions of the errors (see 7.5) were very helpful to promote discussion and common understanding of the issues at play.

13 Resources

City Doctor (Deutsch): Methoden und Metriken zum Qualitätsmanagement virtueller Stadtmodelle

City Doctor (English): Methods and metrics for quality management of virtual city models (Access information from Volker Coors)

CityGML_QIE_Master_Contact_List.docx: CityGML QIE Contact List

CityGML_QualityIE_Activity_Plan_v0.4.5.docx: Latest activity plan

Kick-off meeting resources

TU Delft Github Site: Repository for the test datasets for the CityGML QIE Geometric Validation

Random3Dcity - synthetic multi-LOD CityGML data, with several versions of datasets that contain intentional errors of multiple classes (topology, semantics, ...)

SIG3D Modeling Guide (English): Modeling guide of the SIG3D for 3D objects - Rules for validating GML geometries in CityGML (part 1) and rules for modeling buildings (part 2)

SIG3D Quality Working Group Wiki (Access information form Egbert Casper)

SIG3D Test Cases

van Walstijn, Lucas (2015). *Requirements for an Integral Testing Framework of CityGML Instance Documents* (Unpublished master thesis). TU Berlin, Germany.[8]

---

[8] For information: lvanwalstijn@virtualcitySYSTEMS.de

Annex A

Three-dimensional primitives in the context of the CityGML QIE

1 What is an ISO 19107 solid?

ISO 19107 defines different geometric primitives[9]: 0D is a GM_Point, 1D is a GM_Curve, 2D is a GM_Surface, and 3D is a GM_Solid. A primitive is built with lower-dimensional primitives, e.g., in Figure 27 the GM_Surface is composed of 2 (closed) GM_Curves, which are composed of several GM_Points.

| 0D | 1D | 2D | 3D |
| GM_Point | GM_Curve | GM_Surface | GM_Solid |
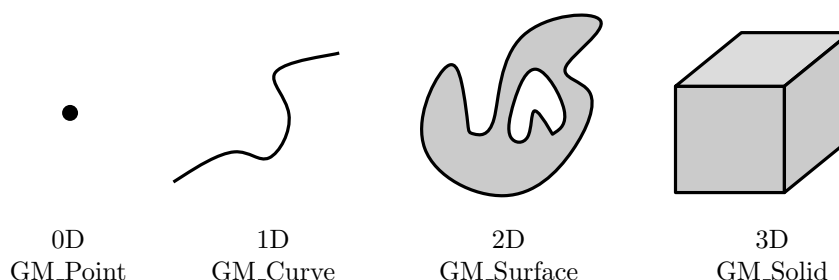
Figure 27: ISO 19017 primitives.

Observe that primitives do not need to be linear or planar, i.e., curves defined by mathematical functions are allowed.

In our context, the following three definitions from ISO (2003) are relevant:

Definition 1 *A GM_Solid is the basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces. The boundaries of GM_Solids shall be represented as GM_SolidBoundary. [...] The GM_OrientablesSurfaces that bound a solid shall be oriented outward.*

Definition 2 *A GM_Shell is used to represent a single connected component of a GM_SolidBoundary. It consists of a number of references to GM_OrientableSurfaces connected in a topological cycle (an object whose boundary is empty). [...] Like GM_Rings, GM_Shells are simple.*

Definition 3 *A GM_Object is* simple *if it has no interior point of self-intersection or self-tangency. In mathematical formalisms, this means that every point in the interior of the object must have a metric neighbourhood whose intersection with the object is isomorphic to an n-sphere, where n is the dimension of this GM_Object.*

The bounding surfaces of a shell thus form a *closed* and *orientable two-dimensional manifold* (or 2-manifold for short). A 2-manifold is a space that is topologically equivalent to R^2, the 2D

---

[9] All the geometric primitive have the prefix 'GM_'

Euclidean space. An obvious example is the surface of the Earth, on which near any point the surrounding area is topologically equivalent to a plane. If a shell is stored in a data structure, it implies that each edge is guaranteed to have a maximum of two incident faces, and that around each vertex the incident faces form one 'umbrella,' as Figure 28 shows.
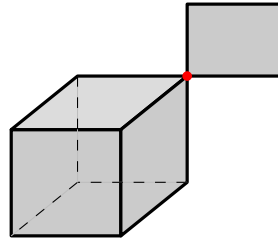
Figure 28: The red vertex is a non-manifold vertex since the neighborhood around it is not topologically equivalent to a plane.

To be valid shell, the 2-manifold should be closed, i.e., there should not be 'holes' in the surface (in other words, it should be watertight).

Figure 29 shows a solid that respects the definition above.
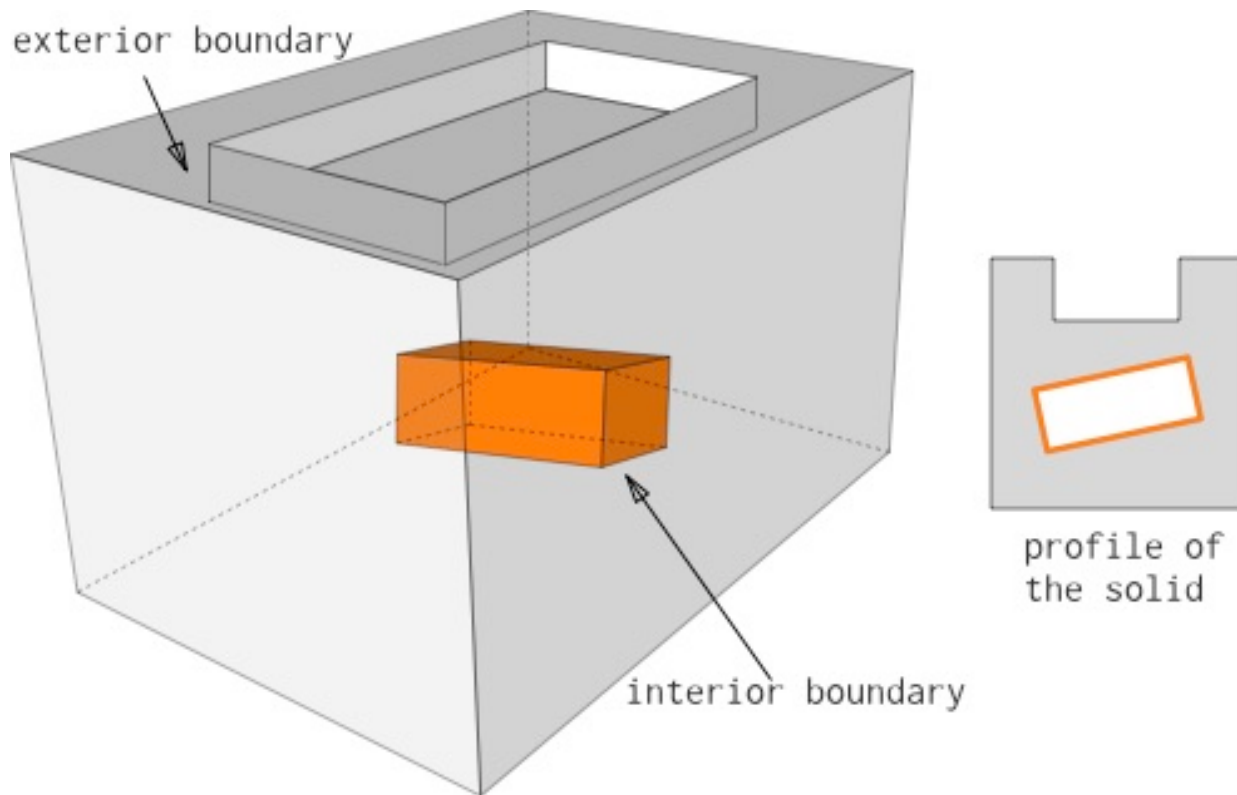
Figure 29: One solid which respects the international definition. It has one exterior shell and one interior shell (forming a cavity).

First, observe that the solid is composed of two shells (both forming boundaries), one being the exterior and one being the interior shell. The exterior shell has eleven surfaces, and the interior one six. An interior shell creates a cavity in the solid—cavities are also referred to as "voids" or holes in a solid. A solid can have no inner shells, or several. Observe that a cavity is not the same as a hole in a torus (a donut) such as that in Figure 30: it can be represented with one exterior shell having a genus of 1 and no interior shell.
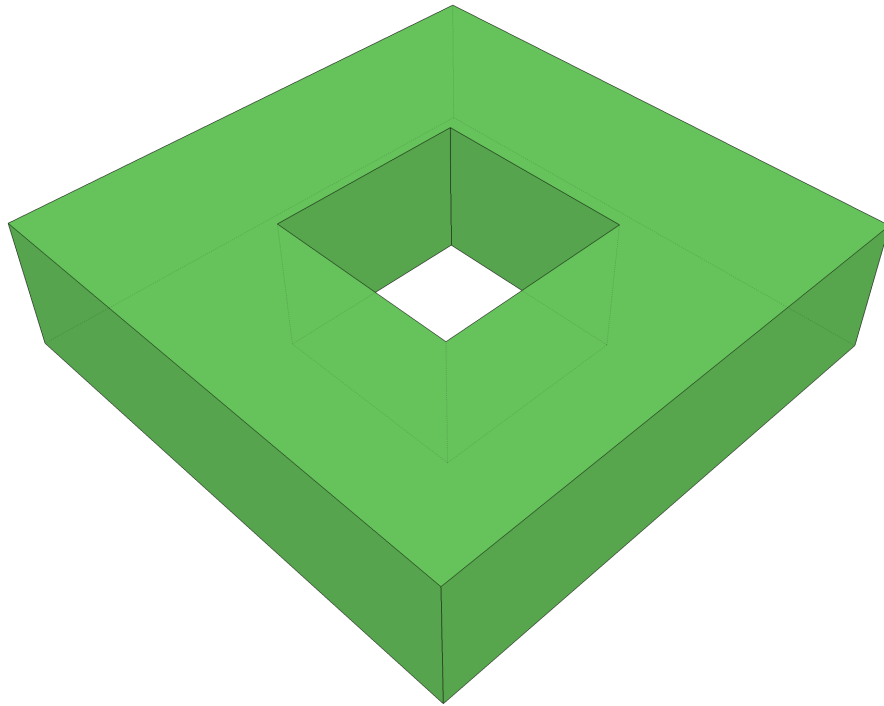
Figure 30: A 'squared torus' is modelled with one exterior boundary formed of ten surfaces. Notice that there is no interior boundary.

2 Primitives in CityGML

CityGML uses the ISO 19107 geometric primitives for representing the geometry of its objects. However, as shown in Figure 31
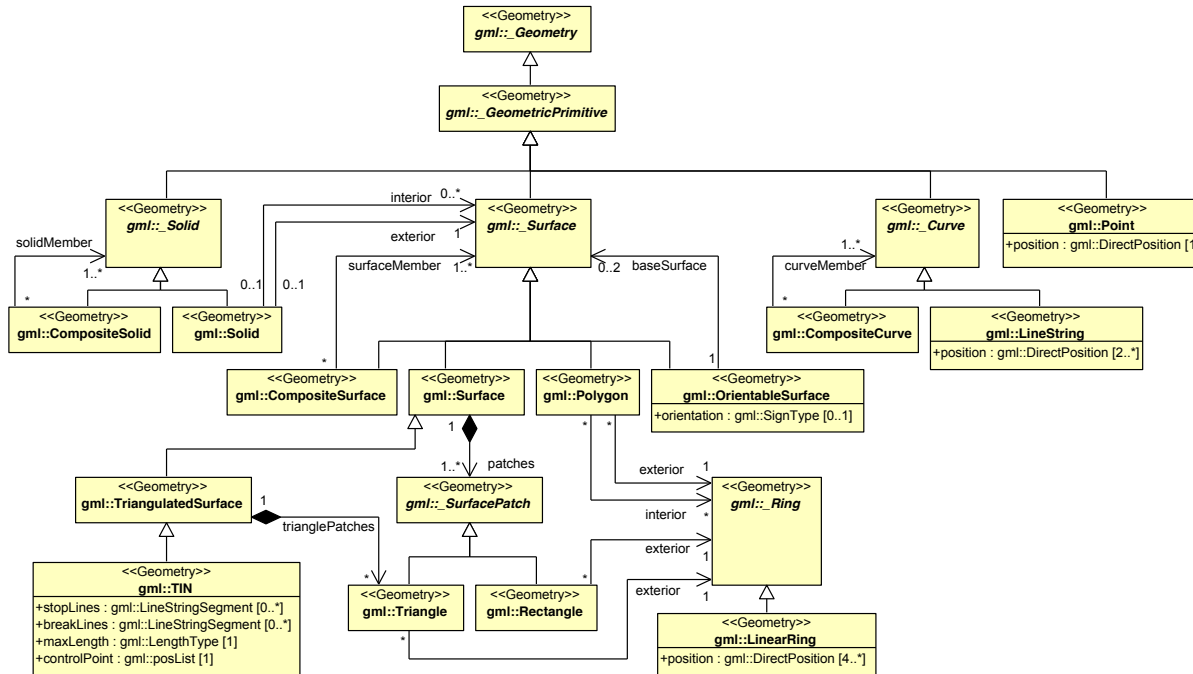
Figure 31: UML diagram of the CityGML geometry model.

only a subset is used, with the following two restrictions: (1) GM_Curves can only be *linear* (thus only LineStrings and LinearRings are used); (2) GM_Surfaces can only be *planar* (thus Polygons are used). The primitives for constructing Shells and Solids are shown in Figure 32.
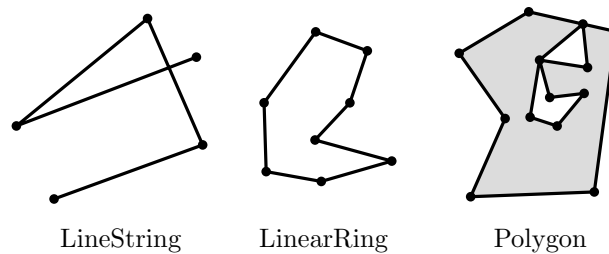
Figure 32: 2D CityGML primitives.

Definition 4 *A LineString is a Curve with linear interpolation between each Point; each two consecutive Points defines a line segment. A LinearRing is a LineString that is both closed and simple.*

Definition 5 *A Polygon is a surface patch that is defined by a set of boundary curves and an underlying surface to which these curves adhere. The default is that the curves are coplanar and the polygon uses planar interpolation in its interior.*

Each shell of a solid is thus composed of Polygons, and these can have inner rings (which are often referred to as holes). Observe that the top polygon of the solid in Figure 29 has one inner ring, but that other polygons "fill" that hole so that the exterior shell is "watertight" (i.e., it has no holes and is thus closed).

3 QIE = no cavities

It should be noticed that during the QIE, only buildings in LOD1, LOD2, and LOD3 were considered, and, as a consequence, cavities in solids are ignored.

This implies that a solid has exactly *one* shell representing its exterior boundary. However, this does *not* mean that inner rings in the boundary surfaces are excluded, because simple LOD1 buildings having for instance, an inner yard, require inner rings, as Figure 30 shows; an alternative to representing inner rings is to decompose the face into several polygons, e.g., to triangulate the face.

4 Requirements for validity of the 3D primitives

Each primitive used to construct a higher-dimensional primitive should be valid. This means that in order to validate a solid, one must also ensure that each ring and polygon used is valid. For rings and polygons, observe that these will be embedded in 3D (i.e., the points used to construct rings will have (x,y,z) coordinates).

4.1 Rings & Polygons

According to the ISO 19107 abstract specification, the different boundaries of a polygon are allowed to interact with each other, but only under certain circumstances. The implementation specifications defined by the OGC (OGC, 2006) gives clear requirements:

1. Polygons are topologically closed;
2. The boundary of a Polygon consists of a set of LinearRings that make up its exterior and interior boundaries;
3. No two Rings in the boundary cross and the Rings in the boundary of a Polygon may intersect at a Point but only as a tangent, e.g.,

    $\Box P \Box Polygon, \Box c1, c2 \Box P.Boundary(), c1 \neq c2,$

    $\Box p, q \Box Point, p, q \Box c1, p \neq q, [p \Box c2 \Box q \Box c2];$

4. A Polygon may not have cut lines, spikes or punctures e.g.:

    $\Box P \Box Polygon, P = P.Interior.Closure;$

5. The interior of every Polygon is a connected point set; and

6. The exterior of a Polygon with 1 or more holes is not connected. Each hole defines a connected component of the exterior.

Some concrete examples of invalid polygons are shown in Figure 33.
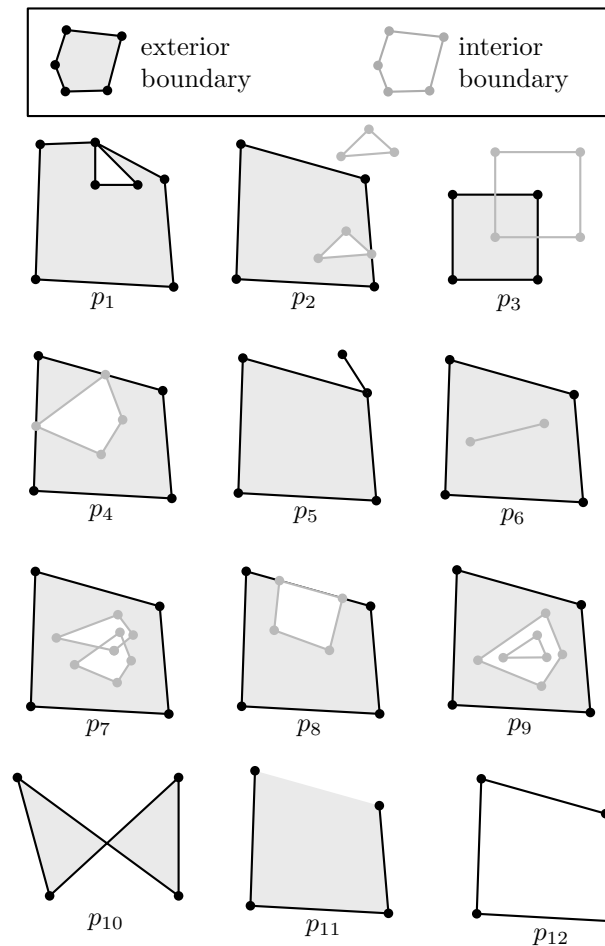
Figure 33: Some examples of invalid polygons. Polygon p_{12} has its exterior and interior rings defined by the same geometry.

Below are explanations for some of the polygons in Figure 33.

1. Each ring should be closed (p_{11}): its first and its last points should be the same.

2. Each ring defining the exterior and interior boundaries should be *simple*, i.e., non-self-intersecting (p_1 and p_{10}). Notice that this prevents the existence of rings with zero-area (p_6), and of rings having two consecutive points at the same location. It should be observed

that the polygon p_1 is not allowed (in a valid representation of the polygon, the triangle should be represented as an interior boundary touching the exterior boundary).

3. The rings of a polygon should not cross (p_3, p_7, p_8 and p_{12}) but may intersect at one tangent point (the interior ring of p_2 is a valid case, although p_2 as a whole is not since the other interior ring is located *outside* the interior one). More than one tangent point is allowed, as long as the interior of the polygon stays connected (see below).

4. A polygon may not have cut lines, spikes or punctures (p_5 or p_6); removing these is known as the *regularization* of a polygon (a standard point-set topology operation).

5. The interior of every polygon is a connected point set (p_4).

6. Each interior ring creates a new area that is disconnected from the exterior. Thus, an interior ring cannot be located outside the exterior ring (p_2) or inside other interior rings (p_9).

4.2 Planarity requirement

A polygon must be planar, i.e., all its points (used for both the exterior and interior rings) must lie on a plane. Interestingly, the concept of *tolerance* is not mentioned in the standards by ISO and OGC.

For the QIE, two requirements are proposed:

1. the distance between every point forming a polygon and a plane is less than ε_1, a given tolerance (e.g., 1mm). This plane should be a plane fitted with least-square adjustment; and

2. the distance between every point forming a polygon and all the planes defined by all possible combinations of 3 non-collinear points is less than ε_1.

The second requirement is to ensure that cases such as that in Figure 34 are detected.
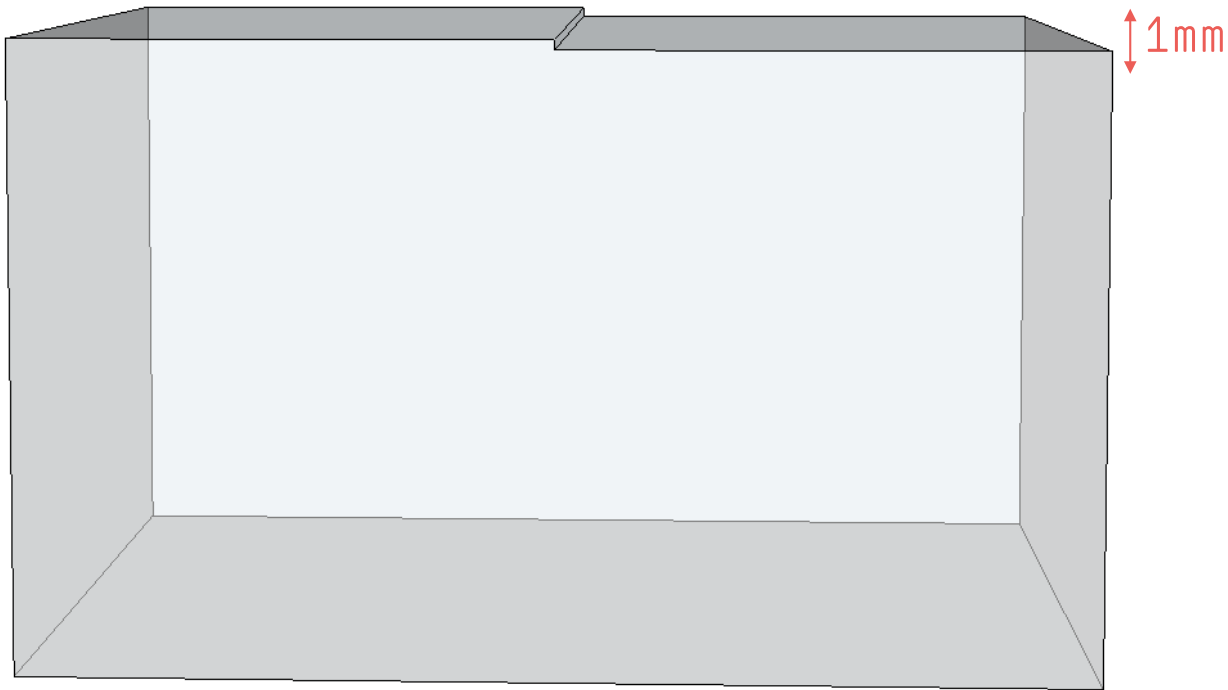
Figure 34: All the points of the top polygon are within 1mm but the polygon cannot be considered planar.

From algorithmic point of view, the definition is not very efficient, but in practice it can be implemented with a triangulation of the polygon (any triangulation): the orientation of the normal of each triangle must not deviate more than a certain user-defined tolerance ε_2 (e.g., 1 degree).

4.3 Snapping tolerances for vertices

Geometries modeled in CityGML, and in GML in general, store amazingly very little topological relationships. For instance, all six surfaces of a cube are stored independently. This means that the coordinates (x,y,z) of a single point (where 3 polygons "meet") is stored 3 times. It is possible that these 3 vertices are not exactly at the same location, e.g., (0.01, 0.5, 1.0), (0.011, 0.49999, 1.00004) and (0.01002, 0.5002, 1.0007), and that would create problems when validating since there would be tiny cracks/overlaps in the cube. The snap tolerance basically gives a threshold that says: "if 2 points are closer then ε_3, then we assume that they are the same." This value should be defined by the user.

4.4 Orientation requirement

For a polygon embedded in the 2D plane, the orientation of its exterior ring must be the opposite of that of its interior rings (e.g., clockwise versus counterclockwise).

If one polygon is used to construct a shell, its exterior ring must be oriented in such a way that when viewed from outside the shell the points are ordered counterclockwise. Figure 35 shows an example.
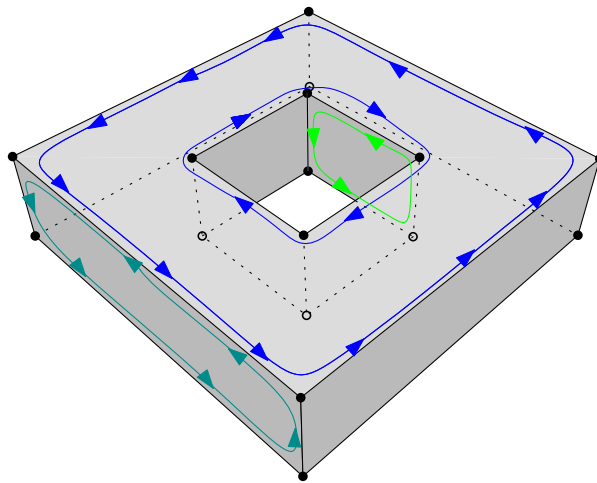
Figure 35: One solid and the orientation of 3 of its polygons (different colors).

In other words, the normal of the surface must point outwards if a right-hand system is used, i.e., when the ordering of points follows the direction of rotation of the curled fingers of the right hand, then the thumb points towards the outside. If the polygon has interior rings, then these must be ordered clockwise.

If the polygon is part of a MultiSurface, then there is no prescribed orientation other than the outer ring must have a different orientation than the inner ring(s).

4.5 Requirements for shells and solids

To understand the requirements for shells and solids, we can simply generalize the following assertions: polygons become solids, rings become shells, and holes become cavities.

Figure 36 shows 9 solids, some of them valid some not.
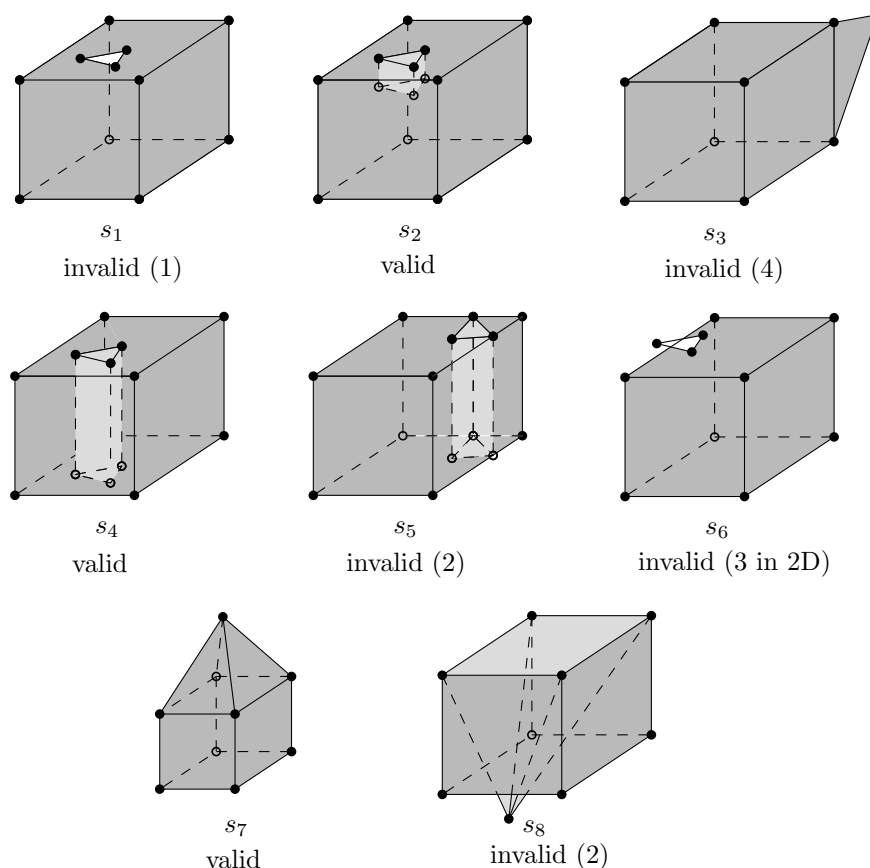
Figure 36: Nine solids, the number between brackets indicates which assertion(s) from the OGC Simple Features is/are violated.

The first assertion means that a solid must be closed, or 'watertight'. The solid s_1 is thus not valid but s_2 is since the hole in the top surface is 'filled' with other faces.

The second assertion implies that each shell must be simple, i.e., that it is a 2-manifold. s_5 and s_8 are thus invalid.

The fourth assertion states that a shell is a 2-manifold and that no dangling pieces can exist (such as that of s_3); it is equivalent to the *regularization* of a point-set in R^3.

The other assertions refer to solids having interior shells, which are out of scope for the QIE. These are thus ignored.

Comment on 4.5 (M.Wewetzer, D. Wagner)

By simply projecting assertions for 2D polygons to 3D shells, certain configurations can be problematic.

ISO 19107 allows tangent points for interaction of different rings of one polygon. By extruding such a polygon, e.g., to create an LoD 1 Building with Solid geometry, a non-manifold edge will be part of the structure (Figure 37). Validation will fail at least for the checks GE_S_NON_MANIFOLD_VERTEX and GE_S_NON_MANIFOLD_EDGE. Depending on the implementation, there might be a GE_S_SELF_INTERSECTION error in addition.
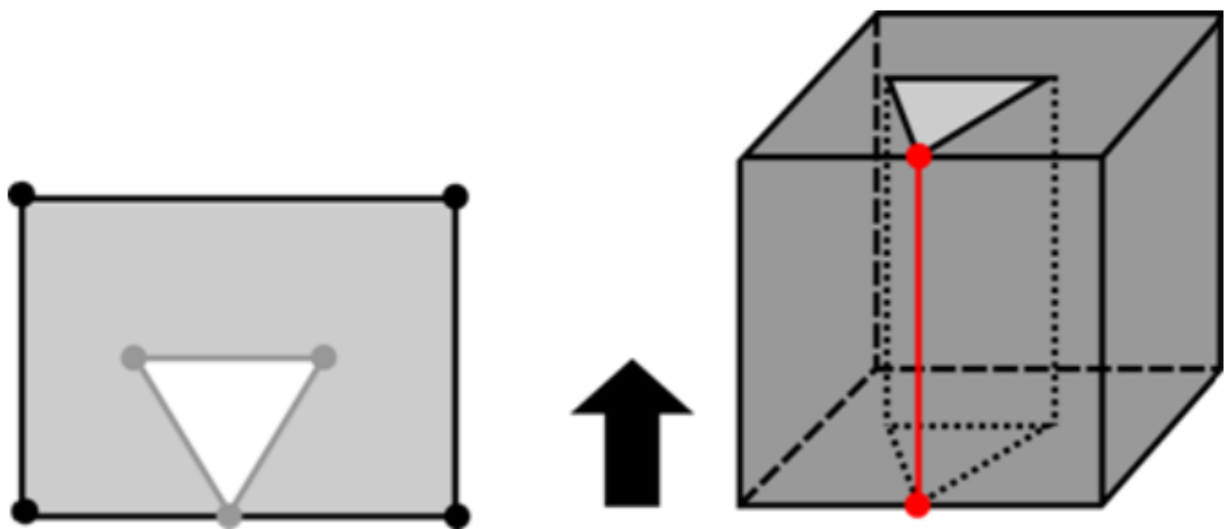
Figure 37: Valid 2D polygon, which results in an extrusion body with a non-manifold edge (red).

The definition for a GM_Ring in ISO 19107 states: "Even though each GM_Ring is simple, the boundary need not be simple. The easiest case of this is where one of the interior rings of a surface is tangent to its exterior ring" (ISO 2003). However, this statement can be overridden: "Implementations may enforce stronger restrictions on the interaction of boundary elements" (ibid).

In the domain of CAD data quality this kind of interaction is neither permitted nor should the distance between two boundary curves fall below a certain threshold.

Considering this, a recommendation to avoid tangent interactions between linear rings of one polygon and thus within the shell of a solid can be justifiable, as we can see no urgent need for their presence.

At least, we should not recommend or enforce non-manifold solid geometries as basic geometric primitives.

References

ISO (2003). ISO 19107:2003: Geographic information—Spatial schema. International Organization for Standardization.

OGC (2006). OpenGIS implementation specification for geographic information—simple feature access. Open Geospatial Consortium inc. Document 06-103r3.

Annex B

Test data sets of the CityGML QIE

Karlsruhe Institut of Technology

name		is_valid
Cube-01.gml		yes
Cube-02.gml		yes
Cube-03.gml		yes

Cube-04.gml
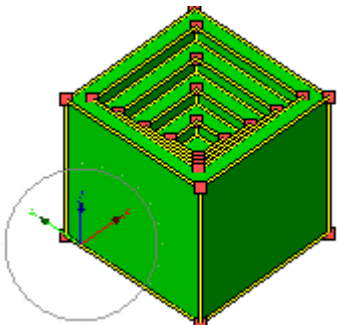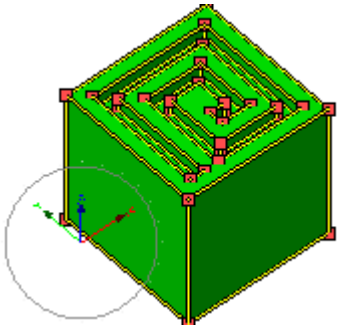
Cube-05.gml

Cube-06.gml

Cube-07.gml

yes

no

yes

yes

Cube-08.gml

Cube-09.gml

Cube-10.gml

Cube-11.gml

yes

no

no
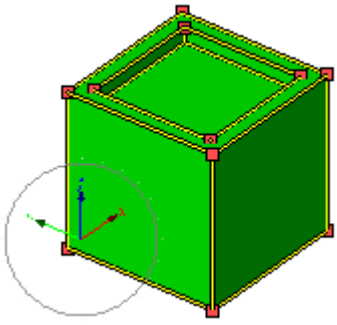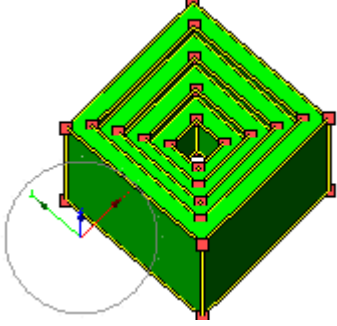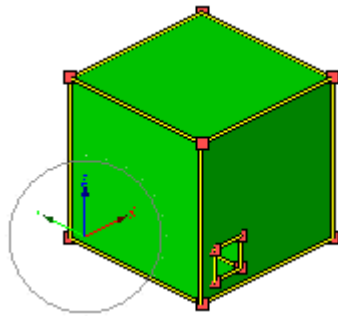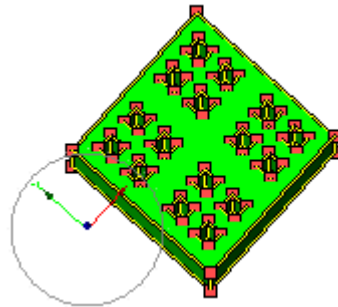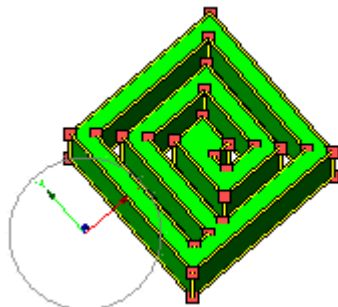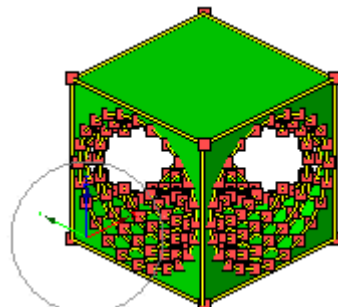
no
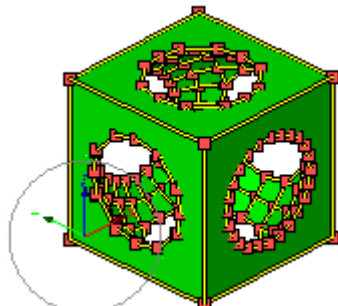
Cube-12.gml

Cube-13.gml

yes

Cube-14.gml

Cube-15.gml
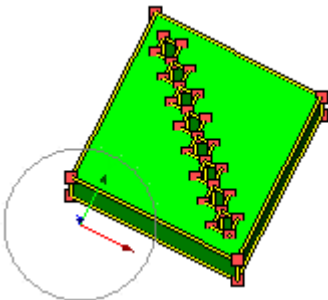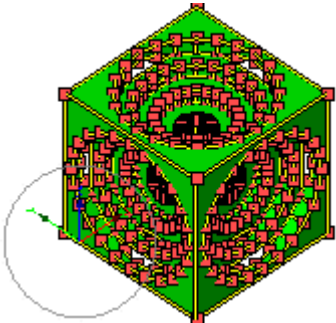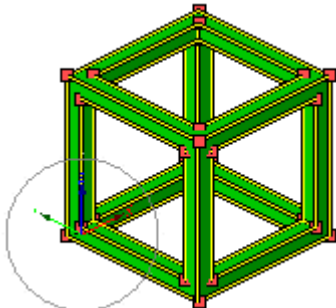
no

Cube-16.gml
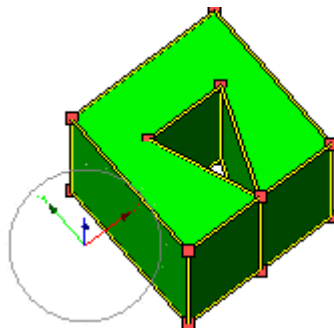
no

Cube-17.gml
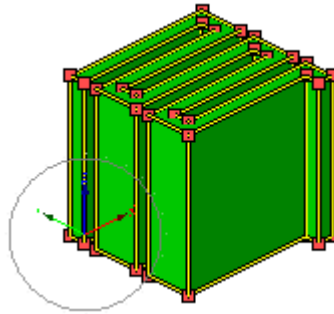
yes

Cube-18.gml
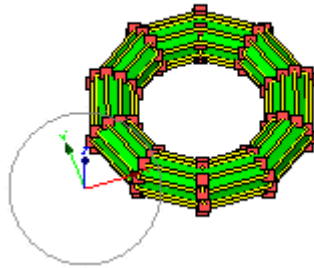
yes

Cube-19.gml

no

Cube-20.gml

yes
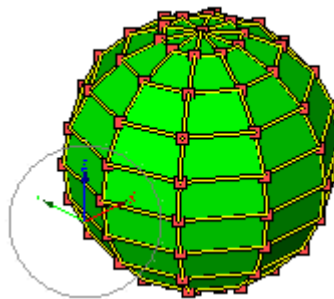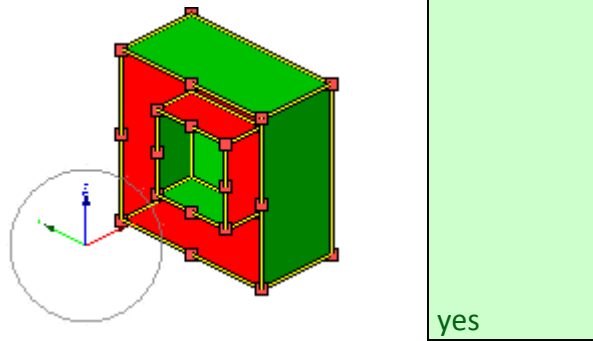
SIG 3D

1) Test Case "Addresses"

- Building LoD 1 with 2 Addresses
- Building LoD 3 with 2 Doors with one Address each

2) Test Case "Generic Attributes"

- Building LOD 1 with different generic Attributes and one Attribute Set
- Building LOD 2 with generic Attributes for Boundary Surfaces

3) Test Case "Geometry"

- Building LoD 2 with 3 Balconies (BuildingInstallation) with implicit Geometry
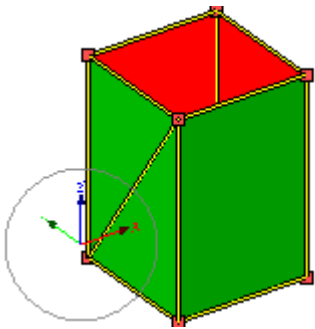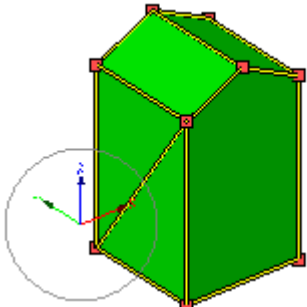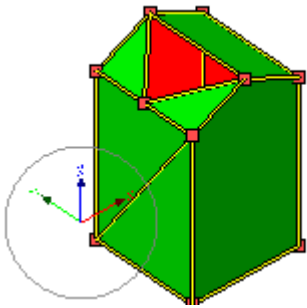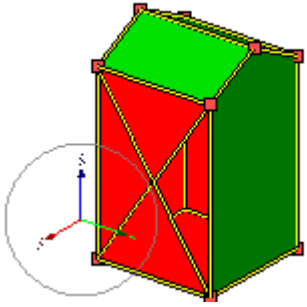
TU Delft

i101_1.gml;cube with top face having only 2 points
i102_1.gml;cube with one duplicate vertex (repeated in a ring)
i104_1.gml;cube where top face has a bow tie
i104_2.gml;unit cube with top face having a self-intersecting surface (2D invalid)
i105_1.gml;cube3 where inner ring is collapsed to a line
i201_1.gml;unit cube with a intersecting rings in top face
i202_1.gml;unit cube with a duplicate inner ring top face
i203_1.gml;surface #12 isn't planar
t203_1.gml;planarity: cube with one point of top surface moved upward by 1e-? units
t203_2.gml;? depends on tolerance
t203_3.gml;? depends on tolerance
t203_4.gml;? depends on tolerance
i204_1.gml;cube with an almost vertical fold in the top surface
i204_2.gml;same shift but vertical
i205_1.gml;unit cube with a polygon with interior disconnected in top face
i206_1.gml;unit cube with a hole in top face located outside

i207_1.gml;unit cube with a polygon with nested rings in top face
i208_1.gml;unit cube with a hole (inner ring) in the top face having same orientation as outer ring
i301_1.gml;volume with only 3 surfaces
i301_2.gml;flat cube
i302_1.gml;unit cube with one face missing (bottom one)
t302_1.gml;snap: cube with top surface having one vertex moved a bit (10cm)
t302_2.gml;snap: cube with top surface having one vertex moved a bit (1cm)
t302_3.gml;snap: cube with top surface having one vertex moved a bit (1mm)
i302_2.gml;unit cube with a hole (inner ring) in the top face
i303_1.gml;unit cube with one dangling face touching the cube at one point only
i303_2.gml;cube with one dangling face
i303_3.gml;2 unit cubes touching at one vertex
i304_1.gml;unit cube with an extra vertex on an edge. Only one of the 2 incident faces has it explicitly.
i304_2.gml;unit cube with one extra face in the middle
i305_1.gml;cube with one extra face "floating" in the air
i305_2.gml;2 unit cubes not touching at all
i306_1.gml;house with tip below the ground
i306_2.gml;house with tip touching the bottom face
i306_3.gml;unit cube with one extra face inside another face
i306_4.gml;torus where the hole in the top/bottom faces touches the side surfaces
i307_1.gml;unit cube with one face (face #1) with opposite orientation
i308_1.gml;axis-aligned cube with normals all pointing inwards
i308_2.gml;not axis-aligned cube with normals all pointing inwards
v001.gml;unit cube
v002.gml;not axis-aligned cube with normals all pointing outwards
v003.gml;unit cube with top face having 3 triangles forming the square
v004.gml;cube translated by (99999990
v005.gml;cube translated by (3333399999990
v006.gml;cube with -100.0 as z-coordinates
v007.gml;a real-life building from Munich
v008.gml;non-convex shape
v009.gml;cube with a pyramidal roof
v010.gml;Stanford bunny: valid
v011.gml;cube3 where there are extra faces that fill the hole on top face
v012.gml;cube5 where hole (inner ring) is touching the outer ring of a face
v013.gml;cube3 with one surface filling the hole
v014.gml;a "squared donut"
i103_1.gml;first polygon is not closed

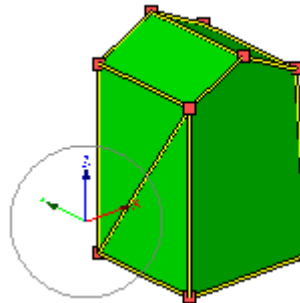**University of Applied Sciences Stuttgart**

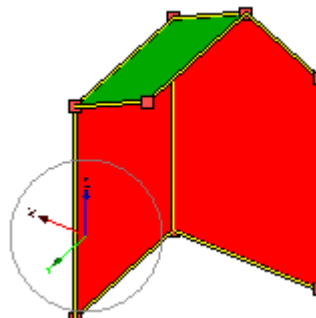| name		is_valid
Solid-010-CP-NUMPOINTS.xml		no
Solid-020-CP-CLOSE.xml		no
Solid-030-CP-DUPPOINT.xml		no
Solid-040-CP-SELFINT.xml		no

Solid-050-CP-PLAN.xml
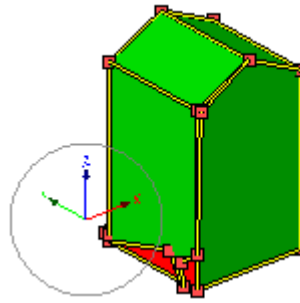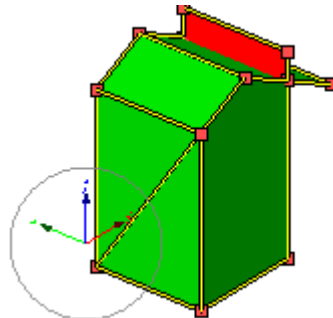

no

Solid-051-CP-PLAN.xml
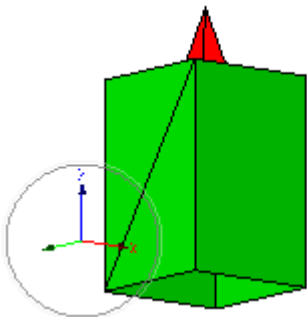

no

Solid-060-CS-NUMFACES.xml
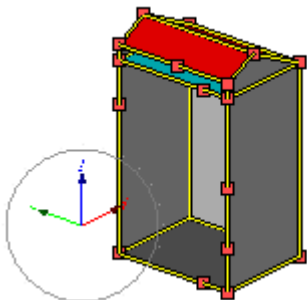

no

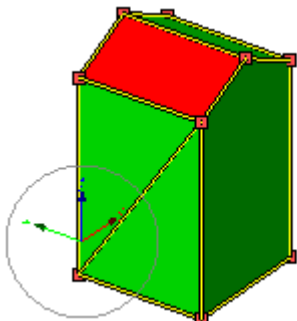Solid-070-CS-SELFINT.xml

no

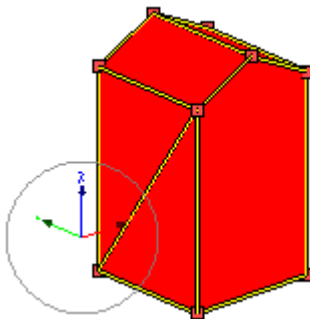Solid-071-CS-SELFINT.xml

no

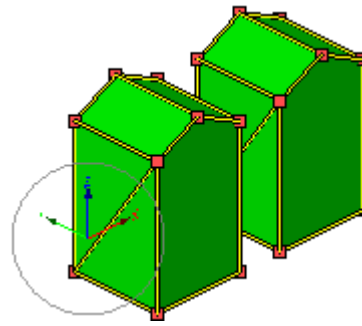Solid-080-CS-2POLYPEREDGE.xml

no
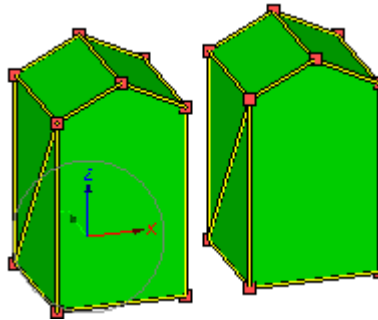
Solid-090-CS-FACEORIENT.xml

no

Solid-100-CS-FACEOUT.xml

no

Solid-110-CS-CONCOMP.xml
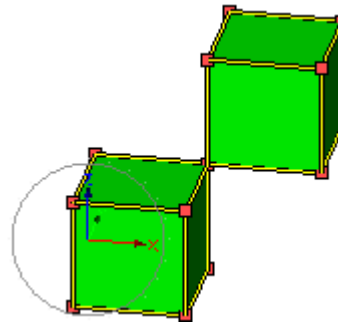

yes

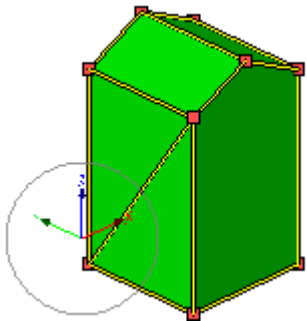Solid-111-CS-CONCOMP.xml

yes

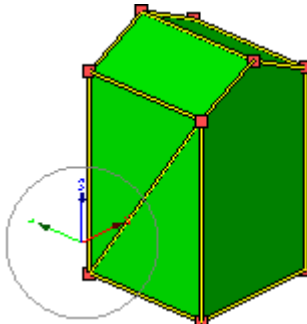Solid-112-CS-CONCOMP.xml
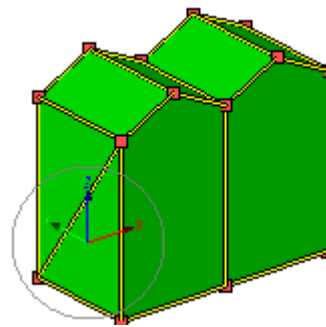
no

Solid-120-CS-UMBRELLA.xml

no

Solid-BS_COPLANAR_SURFACE.xml

yes

Solid-SimpleBldg.xml

yes