DOI: 10.1111/tgis.12710

RESEARCH ARTICLE

Transactions in GIS WILEY

Reference study of CityGML software support: The GeoBIM benchmark 2019—Part II

Francesca Noardo[1] | Ken Arroyo Ohori[1] | Filip Biljecki[2,3] | Claire Ellul[4] | Lars Harrie[5] | Thomas Krijnen[1] | Helen Eriksson[5] | Jordi van Liempt[1] | Maria Pla[6] | Antonio Ruiz[6] | Dean Hintz[7] | Nina Krueger[8] | Cristina Leoni[9] | Leire Leoz[10] | Diana Moraru[11] | Stelios Vitalis[1] | Philipp Willkomm[8] | Jantien Stoter[1]

[1]3D Geoinformation, Delft University of Technology, Delft, The Netherlands

[2]Department of Architecture, National University of Singapore, Singapore, Singapore

[3]Department of Real Estate, National University of Singapore, Singapore, Singapore

[4]Department of Civil, Environmental and Geomatic Engineering, University College London, London, UK

[5]Department of Physical Geography and Ecosystem Science, Lund University, Lund, Sweden

[6]Institut Cartogràfic i Geològic de Catalunya, Barcelona, Spain

[7]Safe Software, Surrey, Canada

[8]M.O.S.S. Computer Grafik Systeme GmbH, Taufkirchen, Germany

[9]Department of Civil, Constructional and Environmental Engineering, Sapienza University of Rome, Rome, Italy

[10]Tracasa, Pamplona, Spain

[11]Ordnance Survey, Southampton, UK

Correspondence
Francesca Noardo, 3D Geoinformation, Delft University of Technology, Delft, The Netherlands.
Email: f.noardo@tudelft.nl

Funding information
International Society of Photogrammetry and Remote Sensing (ISPRS) - Scientific Initiatives 2019 - GeoBIM benchmark project; European association for Spatial Data Research (EuroSDR) - GeoBIM benchmark project.; European Union's Horizon 2020 Research & Innovation Programme, Grant/Award Number: 677312; Marie Sklodowska-Curie, Grant/Award Number: 707404

Abstract

OGC CityGML is an open standard for 3D city models intended to foster interoperability and support various applications. However, through our practical experience and discussions with practitioners, we have noticed several problems related to the implementation of the standard and the use of standardized data. Nevertheless, a systematic investigation of these issues has never been carried out, and there is thus insufficient evidence for tackling the problems. The GeoBIM benchmark project is aimed at finding such

evidence by involving external volunteers, reporting on various aspects of the behavior of tools (geometry, semantics, georeferencing, functionalities), analyzed and described in this article. This study explicitly pointed out the critical points embedded in the format as an evidence base for future development. A companion article (Part I) describes the results of the benchmark related to IFC, the counterpart of CityGML within building information modeling.

KEYWORDS

3D city models, CityGML, data models, GeoBIM, interoperability, standards, software support

1 | INTRODUCTION

Interoperability through open standards is critical for the effective reuse and exchange of data and it is essential for reciprocal integration of different types of data. The integration of 3D city models with building information models (BIMs) has become a widely discussed topic in recent research. Two open standard data models considered for accomplishing such an integration are the Open Geospatial Consortium CityGML (http://www.citygmlwiki.org) for 3D city models, and buildingSMART Industry Foundation Classes (IFC, https://www.buildingsmart.org/standards/bsi-standards/industry-foundation-classes/) for BIM models.

However, even as open standards are highly desirable, there is significant debate surrounding the CityGML standard. As examples, Ledoux et al. (2019) and Arroyo Ohori (2020) report some of the issues of the CityGML standard from the developers' point of view.

Meanwhile, various users on the web also reported related issues, even if in a less academic context.[1] The overall narrative is that the great number of ways in which the same models can be represented increases the implementation effort for software developers and reduces the quality of their implementations, therefore reducing interoperability and falling below the expectations of end-users.

Nevertheless, these issues are mostly discussed informally by practitioners and academics and have not been tested systematically. In order to gain more insight into the topic, the support of CityGML in software was investigated as part of the GeoBIM benchmark project (https://3d.bk.tudelft.nl/projects/geobim-benchmark/, see Section 2) and reported in this article. Within the project, the approach to the study of the support for the two standards involved in the GeoBIM integration (IFC and CityGML) was conceived in parallel, also with the aim of understanding whether one of the two offered more effective solutions that could be possibly borrowed by the other one in future developments. However, the final outcomes of the two different tasks are very specific for each standard and deserve to be presented and discussed separately, considering the specificities of each case. For these reasons, this article, which focuses on the results about the benchmark Task 3 (support for CityGML), is written in tandem with Noardo, Ellul, et al. (2020), which describes Task 1 covering the support for IFC. In order to allow each article to be read on its own, the two articles share some information (i.e., the first part of Section 2, explaining the general context and motivation of the study; Section 3.1 covering the initial part of the methodology about the entire GeoBIM benchmark set-up, and Section 3.3 concerning some similarities in the methodology). One further article explores the parts of the project more directly related to the subject of integration, namely, conversion procedures and useful tools to georeference IFC models (Noardo, Harrie et al., 2020).

2 | THE GeoBIM NEEDS AND THE IDEA BEHIND THIS STUDY

Two kinds of 3D information systems have been developed, studied and used in recent times, revealing their potential in related fields:

- *3D city models*, which are used to represent city objects in three dimensions and advance previous 2D maps and other cartographic products, in order to support city analysis and management, city planning, navigation, and so on (e.g., Bartie, Reitsma, Kingham, & Mills, 2010; Biljecki, Stoter, Ledoux, Zlatanova, & Çoltekin, 2015; Egusquiza, Izkara, & Gandini, 2018; Jakubiec & Reinhart, 2013; Kumar, Ledoux, Commandeur, & Stoter, 2017; Liang, Gong, Li, & Ibrahim, 2014; Nguyen & Pearce, 2012; Peters, Ledoux, & Biljecki, 2015);
- *building information models*, which are used in the architecture, engineering and construction fields to design and manage buildings, infrastructure and other construction works, and which also have features useful to project and asset management (e.g., Azhar, 2011; Haddock, 2018; Petri, Kubicki, Rezgui, Guerriero, & Li, 2017).

Several international standards exist to govern the representation of the built environment in a shared manner, to foster interoperability and cross-border analysis and, consequently, actions, or to reuse tools, analysis methods and data themselves for research and, possibly, government. Some examples of international standards are: the European Directive for an infrastructure for spatial information in Europe (INSPIRE) (https://inspire.ec.europa.eu), aimed at the representation of cross-border territories in Europe, for common environmental analysis; the Land and Infrastructure standard (LandInfra, https://www.ogc.org/standards/landinfra), by the Open Geospatial Consortium (OGC), aimed at land and civil engineering infrastructure facilities representation; and the green building data model (gbXML, https://www.gbxml.org) aimed at the representation of buildings for energy analysis.

Nonetheless, the two dominant reference open standards for those two models are CityGML (citygmlwiki.org), by the OGC, focusing on urban-scale representation of the built environment, and the Industry Foundation Classes (ISO, 2013, https://technical.buildingsmart.org/standards/ifc/), by buildingSMART, aimed at the very detailed representation of buildings and other construction works for design and construction objectives, first, but also intended to enable project management throughout the process, and asset and facility management in a following phase. Those standards are both intended to be very comprehensive and are therefore very wide and articulated. They both use complex data models allowing for a wide variety of models using object-oriented representations, even if that comes at the cost of slower and more inconsistent implementations.

Due to the overlapping interests in both fields (meeting in the building-level representation), increasing attention is being paid to 3D city model-BIM integration (GeoBIM), where the exchange of information between geospatial (3D city models) and BIM sources enables the reciprocal enrichment of the two kinds of information with advantages for both fields, e.g., automatic updates of 3D city models with high-level-of-detail features, automatic representation of BIM in their context, automated tests of the design, and so on (Aleksandrov, Diakite, Yan, Li, & Zlatanova, 2019; Arroyo Ohori, Diakite, Krijnen, Ledoux, & Stoter, 2018; Fosu, Suprabhas, Rathore, & Cory, 2015; Kang & Hong, 2015; Kumar, Labetski, Arroyo Ohori, Ledoux, & Stoter, 2019; Lim, Tauscher, & Biljecki, 2019; Liu et al., 2017; Niu, Yang, & Pan, 2019; Noardo et al., 2019; Stouffs, Tauscher, & Biljecki, 2018; Sun, Mi, Olsson, Paulsson, & Harrie, 2019).

The GeoBIM subject can be divided into several sub-issues:

1. The harmonization of data themselves, which have to concretely fit together, with similar (or harmonizable) features (e.g., accuracy, kind of geometry, amount of detail, kind of semantics, georeferencing).
2. Interoperability, which is a fundamental key in the integration. It is important to note here, that before enabling the interoperability among different formats (e.g., GIS and BIM), which is the theme of point 3 below, the interoperability GIS-to-GIS and BIM-to-BIM itself is essential. That means that the formats of data have to be

understood and correctly interpreted uniquely by both any person and any supporting software. Moreover, an interoperable data set is supposed to remain altogether unchanged when going through a potentially infinite number of imports and exports by software tools, possibly converting it to their specific native formats and exporting it back.

3. The effective conversion among different formats, that is, transforming one data set in a (standardized) format to another one in compliance with the end format specifications and features.
4. The procedures employing 3D city models and the ones based on BIM should be changed in order to obtain better advantages by the use of both, integrated, since those systems enable processes which are usually more complex than just the simple representations.

The many challenges implied by the points above are still far from being solved, and one of the essential initial steps is actually to outline such challenges more sharply.

In particular, the second point (interoperability and involved standards) is often considered to be solved by standardization organizations. It is desirable to rely on open standards for this, because the well-documented specifications of open standards would enable longer-term support, as well as their genericity with respect to different software vendors, as opposed to closed point-to-point solutions that merely connect one proprietary system to another (and might be discontinued or stop working at any moment). However, our previous experiences suggest that the support for open standards in software often includes shortcomings.

The researchers promoting this study (as users of data, advocates of open standards and developers of tools adopting such standards) have noticed, over their research and work activities, how the use of those standards in data and their implementations in software were not always straightforward and not completely consistent with the standard specifications either. Many tools, when managing standardized data, do not adequately support features or functionalities as they do when the data is held in the native formats of the software. In addition, software tools have limitations with respect to the potential representation (geometry, semantics, georeferencing) of data structured following these standards, or can generate errors and erroneous representations by misinterpreting them.

The standards themselves are partly at fault here, since they often leave some details undefined, with a high degree of freedom and various possible interpretations. They allow high complexity in the organization and storage of the objects, which does not work effectively towards universal understanding, unique implementations and consistent modeling of data. This is probably due to the fact that such standards often originate as amalgamations of existing mechanisms and compromises between the various stakeholders involved. These experiences have been informally confirmed through exchanges within the scientific community and especially with the world of practitioners, who are supposed to work with (and have the most to gain from) those standardized data models and formats. However, more formal evidence on the state of implementation of these open standards and what problems could be connected to the standard themselves have not been compiled so far.

For this reason, the GeoBIM benchmark project (https://3d.bk.tudelft.nl/projects/geobim-benchmark/, https://www.isprs.org/society/si/SI-2019/TC4-Noardo_et_al_WG-IV-2-final_report.pdf) was proposed and funded in 2019 by the International Society for Photogrammetry and Remote Sensing (ISPRS, https://www.isprs.org) and the European association for Spatial Data Research (EuroSDR, http://www.eurosdr.net). The aim of the benchmark was to get a better picture of the state of software support for the two open standards (IFC and CityGML) and the conversions between them, in order to formulate recommendations for further development of the standards and the software that implements them. In addition, we tested two known major technical issues related to GeoBIM integration and which are known to be solved only partially in practice: the ability of tools and methods to georeference IFC and the conversion procedures between IFC and CityGML Noardo et al. (2020).

The relevant outcomes regarding the OGC CityGML standard are the subject of this article.

2.1 | OGC CityGML: Overview and knotty points

CityGML (citygmlwiki.org and citygml.org), by Open Geospatial Consortium (OGC, 2012), is the most internationally widespread standard to store and exchange 3D city models with semantics in the geospatial domain. It establishes a structured way to describe the geometry and semantics of city objects.

CityGML 2.0 (current version, considered in this project) contains classes structured into 12 modules, each of them extending the core module, containing the most general classes in the data model, with city object-specific classifications, (e.g., Building, Bridge, WaterBody, CityFurniture, LandUse, Relief, Transportation, Tunnel, Vegetation). These modules contain one or more classes representing specific types of objects, which differ in the way they are structured into smaller parts and the attributes that are expected for each. The most developed and most used module in practice is the Building module.

Moreover, CityGML supports the possibility of further extending the schema through a standardized application domain extension (ADE) mechanism (Biljecki, Kumar, & Nagel, 2018). Some existing official ADEs, which could be useful for future tests of this project, are, for example, the Noise ADE, the Energy ADE, and the Utility network ADE. However, it is known that ADEs have poorer software support, since most implementations would need to specifically encode the new objects and attributes added by an ADE.

CityGML proposed a very attractive management of useful concepts for user communities. For example, it covers the most basic 3D city information with meaningful object-oriented representation, with deep hierarchies and complex relationships, which would be a more faithful representation of reality than a simpler relational database one. However, the downside of this is that complex and unusual connections of information to internal/external sources are sometimes used (e.g., the prevalence of xlink-connected geometries within a file, the complex set of attributes used to store addresses, or the possibility of using references to external files through URIs), which could be a problem for the implementation of such a model.

CityGML geometries are essentially the same for most classes: objects are represented as boundary surfaces embedded in 3D and consist of triangular or polygonal faces, possibly with holes.

CityGML as a data format is implemented as an application schema for the Geography Markup Language (GML) (CityGML uses version 3.1.1 of GML) (OGC, 2004). It is an open format and human readable, which means that the information could potentially be retrieved even if losing backward compatibility in software. However, GML presents many issues from a software developer point of view, since, for example, too many alternatives (http://erouault.blogspot.com/2014/04/gml-madness.html) are allowed even for simple objects, and a supporting application is supposed to foresee all possible combinations of them. The result of this complexity is that few software programs completely support all possible combinations, and most of its richness and power is lost (as the results of this research further demonstrate). An additional consequence of the kind of storage of such models is about their computational requirements: usually very large and complex files are produced, and it can be time- and resource-intensive to manage them properly in software.

As a possible solution to those issues, CityJSON (https://www.cityjson.org) version 1.0.0 (Ledoux et al., 2019) was recently released, providing a JSON encoding for a subset of the CityGML 2.0.0 data model. CityJSON follows the philosophy of another (non-standardized but working) encoding of CityGML: 3DCityDB (Yao et al., 2018): to store the models efficiently and allow practitioners to access features and their geometries easily. The deep hierarchies of the CityGML data model are replaced by a simpler representation. Furthermore, some more restrictions are applied and one and only one way is allowed to represent the semantics and the geometries of a specific feature. CityJSON is in the process of becoming an OGC community standard. There is already a broad consensus around it, among users who are choosing it as an alternative to CityGML and many tools already developed to effectively work with it. Some of the tests performed within this study use CityJSON as a pathway to process CityGML in order to manage the data effectively within software.

In this article, the framework of the GeoBIM benchmark methodology is described, with a specific focus on the part of the project regarding the investigation of the support for CityGML and related results.

3 | METHODOLOGY

3.1 | The GeoBIM benchmark general set-up

The benchmark was intended as a way to combine the expertise of many people with different skills, coming from several fields and interests, in order to describe the present ability of current software tools to use (i.e., read, visualize, import, manage, analyze, export) CityGML and IFC models and to understand their performance while doing so, in terms of both information management functionalities, and possible information loss. Moreover, since the large size of such standardized data sets often generates difficulties in their computational management, the ability to handle large data sets was a further part of the tests.

In particular, the four topics investigated in the benchmark are:

Task 1. What support is there for IFC within BIM (and other) software?
Task 2. What options for georeferencing BIM data are available?
Task 3. What support is there for CityGML within GIS (and other) tools?[2]
Task 4. What options for conversion (software and procedural) (both IFC to CityGML and CityGML to IFC) are available?

For this purpose, a set of representative IFC and CityGML data sets were provided (Noardo et al., 2019) and used by external, voluntary participants in the software they would like to test in order to check the support in it for the open standard considered (Noardo, Biljecki, et al., 2019).

Full details about the software tested and a full list of participants can be found in the respective pages of the benchmark website.[3] The significant number of participants, balance in skills, fields of work, levels of confidence about the software tested (asked them to be declared) offered the possibility to limit the bias in the results.

The participants described the behavior of the tools tested following detailed instructions and delivered the results in a common template with specific questions, provided as online forms. In the end, they delivered both their observations and the models as re-exported back to the original standardized format (CityGML or IFC).

In order to cover the widest part of the list of software potentially supporting the investigated standards, we completed the testing ourselves, by searching the online documentation of both the standards and the potential software.

In the final phase of the project, the team coordinating the study analyzed the participants' observations, descriptions and delivered further documentation (screenshots, log files, related documents and web pages). From this review, an assessment of the performances and functionalities of the tools tested was derived. Moreover, the models delivered were validated and analyzed using available tools, when possible, and/or through manual inspection (Section 3.3). This approach enabled the inquiry about the level of interoperability given by the standard and its software implementation, by comparing the results of the export with the imported model features.

It is important to notice that the test results are not intended to substitute the official documentation of each software program. Moreover, there were no expertise nor skill requirements to participate in the benchmark tests. Therefore, some information could be wrong or inaccurate, due to little experience with the software tested or the topics managed. The declaration of the level of expertise was intended to lower this possible bias. Moreover, the benchmark team and the authors tried to double-check the responses (at least the most unexpected ones) as much as possible, but the responses reported in the data were generally not changed from the original ones. Any discrepancies between the best potential software performances and what was tested could anyway be showing a low level of user-friendliness of tools (and thus a degree of difficulty in achieving the correct result).

3.2 | The CityGML data sets provided

A number of data sets from different sources were identified, pre-processed and validated for this benchmark activity (for details, see Noardo et al., 2019). The data sets were chosen to test both the most common features of such data and the main detected issues regarding interesting but tricky aspects of the format.

Therefore, a large level of detail 1 (LoD1) file was chosen to test the support for a quite simple but extended data set: the whole city of Amsterdam in LoD1, covering the representation of many city-related objects and useful for testing software- and hardware-related performance. Furthermore, a two-LoD file (LoD1 and LoD2) representing a district in Rotterdam was used to test the support of different LoDs stored in the same file. Finally, the more complex geometries (and related different semantics) needed for an LoD3 representation were tested by means of the synthetically generated file *BuildingsLoD3.gml* (Table 1). LoD4 models were not included in the data sets provided, since few if not even any examples can be found in practice.

3.3 | Analysis of responses about the support for CityGML

The methodologies for analyzing the results about the support of software for IFC (Task 1) and CityGML (Task 3) are very similar, since they were also conceived to test similar issues concerning interoperability and the ability of software to keep files consistent with themselves after import–export phases.

The initial part of results analysis (Section 4.2) is qualitative, providing the description of software support and functionality based on the responses delivered.

The complete responses and documents delivered in the online templates (Noardo et al., 2020b, http://doi.org/10.5281/zenodo.3966987) were double-checked for correctness and consistency with respect to the questions asked. However, due to the nature of the initiative, we trusted the delivered information about the software, double-checking it with new tests only in cases of inconsistent responses in different tests about the same software, or possibly unexpected responses. In these cases, we also considered the level of expertise of the participant to assess if further checks were actually needed.

The responses delivered in the templates were critically assessed and cross-checked with the different tests about the same software and the attached screenshots. A score for each aspect considered for the assessment of general support and software functionalities was assigned (1, full support; 0.5, partial support; 0, no support). Those were synthesized in a table (Table 3) in order to more easily deduce possible patterns across many issues for a single software package or across many software packages for a single issue.

The definition of software groups is growing increasingly fuzzy, since their functionalities are continuously being extended and now tend to overlap with each other. However, in the tables, and more generally in the analysis, in order to help the detection of possible patterns, the software tested was classified according to the criteria that usually guide the choices made by users, based on their different needs for specific tasks:

- *GIS* are expected to combine different kind of geodata and layers and carry out analysis on them, structured in a database, in a holistic system.
- *"Extended" 3D viewers* are likely software that was originally developed for visualizing 3D semantic models, including georeferencing, and querying them. They were (sometimes later) extended with new functions for applying symbology or making simple analysis.
- *Extract transform and load* (ETL) software, and conversion software, is expected to apply some defined transformations or computations to data.
- *3D modeling tools* have good support for geometry editing, but are not intended to manage georeferenced data or semantics.
- *Analysis software* is intended specifically for few kinds of very specific analysis (e.g., energy analysis),

TABLE 1 Provided CityGML data for the GeoBIM benchmark 2019

Name	Description	Dimension	Source	Aim
Amsterdam.gml	Seamless city model covering the whole city of Amsterdam, including several CityGML city entities (vegetation, roads, water, buildings, etc.) LoD1.	4.06 GB	Generated through 3dfier by TUDelft (https://github.com/tudelft3d/3dfier)	Test of the hardware- and software-connected performance (it is a very heavy model), and support for the included city classes
RotterdamLoD12.gml	Textured CityGML model of one district in Rotterdam, including only buildings in LoD1 and LoD2	33.91 MB/154.4 MB (with textures)	Municipality of Rotterdam (NL)	Test of the support for multiple LoDs and textured files
BuildingsLoD3.gml	Procedurally modeled buildings in LoD3	1.33 MB	Generated through Random3Dcity (Biljecki, Ledoux, & Stoter, 2016, https://github.com/tudelft3d/Random3Dcity)	Test of the support for LoD 3 files and related classes

- *BIM* software is intended to design buildings or infrastructure according to BIM methods.

The issues investigated, reflected in the different sections of the templates provided, regarded mainly the support of the software for the two standards (how the software reads and visualizes the data sets) and the functionalities allowed by the software with standardized data sets (what is it possible to do with such data). In particular, the test for the support was intended to check: how the georeferencing information in the files is read and managed; how the semantics are read, interpreted and kept after importing; and the state of the geometry after importing. Georeferencing is about the ability of the software to locate and visualize the data at the correct georeferenced coordinates, together with recognizing the correct coordinate reference system (CRS), as read within the file. Semantics are the thematic data associated to the objects of the model, which are structured in hierarchies of classes, reciprocally related by means of the relationships defined in the data model. In this case, the reference data model is ruled by the CityGML standard. Further information is associated to such entities as attributes. The third point is about geometry — the way each object is modeled spatially and how such geometry is stored (e.g., as a solid, as a surface).

Moreover, some additional questions for Task 3 were intended to investigate if any additional step or conversion was necessary to use the file within the tested software, or a straightforward workflow was sufficient (e.g., opening the software and importing the file by pushing a button).

Additional questions therefore are: what kind of formats can be managed (is a file supported through the GML encoding, or a conversion is eventually needed in addition or alternatively, to the CityJSON format or to a database, mainly SQL)? Does the software support the file out of the box, or is some specific plug-in or add-on needed? Finally, we are also concerned with the following issues:

- What kind of visualization is enabled (3D, 2D, with textures, with specific themes).
- What kind of editing is possible (attributes, geometry, georeferencing).
- What kind of query (query the single object to read the attributes, selection by conditions on attributes, spatial query, computation of new attributes).
- What analyses are allowed. This topic is more complex, since very different analyses are possible. Therefore we summarized it by defining two analysis types: "Type 1" is any kind of analysis regarding the model itself (e.g., geometric or semantic validation), and "Type 2" is the simulations and analyses of the performance of the object represented (e.g., a building) with respect to external factors, in the city or environment (e.g., shadow, noise, energy).
- Finally, is it possible to export back to CityGML?

One more aspect that was asked of the testers of Task 3 (support for CityGML), and checked in the delivered results, was the support for ADEs, although it was just checked in theory, since no ADE data sets were provided.

Moreover, the support for each of the delivered data sets was noted, given the specific features as follows: multi-LoD management (through *RotterdamLOD12.gml* data set), LoD3 management (through *BuildingsLOD3.gml*) and a large LoD1 model (the *Amsterdam.gml* data set).

This first part provides a reference about the tools themselves for people intending to use standardized information. In addition, the most challenging tasks and most frequent issues for the management of standards were supposed to be pointed out.

A second, more quantitative, part of the analysis considers the delivered models exported back to CityGML (Noardo et al., 2020a, https://doi.org/10.5281/zenodo.3966915) from the tested software (Section 4.4). The numbers and types of features of such files were calculated and compared to the same features in the initial data sets that were provided for the test.

The semantics were checked, in terms of number of entities and relationships, as computed by the statistics tool related to the KIT FZKViewer (https://www.iai.kit.edu/1302.php). Moreover, the presence and consistency

of attributes was also checked by means of manual inspection in 3D viewers (FZK and azul, https://github.com/tudelft3d/azul).

In addition, the CityGML schemas were validated by means of the GML schema validator related to the FZKViewer and the CityGML schema validator (http://geovalidation.bk.tudelft.nl/schemacitygml).

The number and kind of geometries were also counted by the FZK statistics tool, further supported by manual inspection in some cases. Finally, The val3dity (http://geovalidation.bk.tudelft.nl/val3dity) validation tool (Ledoux, 2013) allowed testing of the validity of re-exported geometries.

This allowed us to assess the level of interoperability that the connected standards-tools can actually achieve in the different cases: that is, can the data be imported and re-exported without any change?

A further assessment (Section 4.3) was intended to evaluate the software- and hardware-connected performance. The times declared by the testers were compared for the three data sets to see if their computational weight could affect their management within software.

Given the complexity of measuring software performance to the nearest second, this was not requested of the users. Instead, they were asked to provide an approximate timing value for each test, according to a classification that was proposed following the way they could affect the perception or the work of a user, as explained in the following list:

- It is almost immediate (good!).
- Less than a minute (ok, I will wait).
- 1–5 min (I can wait, if it is not urgent).
- 5–20 min (in the meantime I can do other things).
- 20 min–1 hr (I cannot rely on it for frequent tasks),
- More than 1 hr (I launch my process and go home, definitely ineffective for regular work).

Other options included reporting whether the software crashed or whether the task was not possible with the software provided, and participants were also asked to provide information about the specification of the machine, as this may impact overall performance of the software. Due to their diverse levels of size and complexity, timing results are summarized for the individual data sets.

4 | RESULTS: SUPPORT OF SOFTWARE FOR CityGML

4.1 | Software tested

In order to ensure the coverage of most of the software solutions to manage CityGML, we checked that the main currently used GIS software packages were tested, and we tried to ensure that the tools declaring some support for CityGML were tested as well.

In total, 15 software packages were tested, with several tests for some of them, likely covering most of the possible solutions. The full list is reported in Table 2. In the table, they are organized based on the kind of software and divided into open source, proprietary and freeware (but not open source) software. Moreover, the levels of expertise of participants making the tests (from Level 1, the least expert, to Level 4, the most expert) are also reported.

Some of the software packages were tested several times, especially the best known and most popular GIS tools, where the inclusion of 3D city model data could be the most interesting and useful, as well as natural, since they are the current tools employed to manage geoinformation. *QGIS* (https://www.qgis.org/en/site/) was tested four times, plus once partially, by participants having different levels of expertise (beginners, current users and experts). The four tests conducted adopted different approaches, though: built-in support for GML, import of the file after conversion to CityJSON and the use of the CityJSON loader plug-in. Additional methods are the import

TABLE 2 Software tested for Task 3: support for CityGML

	Open source	Proprietary	Freeware
GIS software	QGIS [L1+L2+L3]	ESRI ArcGIS [L1+L2]	
		ESRI ArcGIS Pro [L2]	
'Extended' 3D viewers		Safe Software FME Data Inspector [L3]	FZKViewer [L1+L2]
		eveBIM [L4]	Hexagon AB tridicon CityDiscoverer Light [L1]
		M.O.S.S. novaFACTORY+WEGA-3D [L3]	
		1Spatial Elyx 3D [L2]	
ETL and conversion software	3DcityDB[L3]	Safe Software FME [L3]	
3D modeling software	Blender+CityJSON plug-in [L1]	ESRI CityEngine [L1]	
Analysis software		Kaemco CitySim Pro [L1]	
BIM software		Autodesk Infraworks [L1]	

through *citygml4j*, and the use of *GMLAS* (https://gdal.org/drivers/vector/gmlas.html) However, the *GMLAS* procedure was not successful at importing the data.

The CityJSON (https://www.cityjson.org/software/) implementation was used to enable the import of the data sets in Blender, by means of the CityJSON plug-in (https://github.com/cityjson/Blender-CityJSON-Plugin), which could be useful to model and edit the models. *ESRI ArcGIS* was also tested multiple times, both in the standard version (three tests) and in the Pro version (two tests). Other popular software packages were *FME* (Feature Manipulation Engine, https://www.safe.com) from Safe Software, tested twice, and the freeware extended 3D viewer KIT *FZKViewer* (https://www.iai.kit.edu/1648.php), also tested twice (by testers having levels of expertise 1 and 2).

Moreover, other applications were considered (raising the number of software packages tested to 26), selected on the base of information found on the CityGML Wiki under 'software' (http://www.citygmlwiki.org/index.php/Commercial_Software, http://www.citygmlwiki.org/index.php?title=Freeware, http://www.citygmlwiki.org/index.php?title=Open_Source) or on the websites of software programs declaring support for CityGML. However, some of them, the ones for which the full test was not performed, did not actually support the data or were no longer available:

1. *Autodesk Landxplorer CityGML Viewer* is quite outdated, it is possible to find information about it until around 2011 (http://download.autodesk.com/us/landxplorer/docs/LDX11_Studio/index.html?topic.htm) and no download was found except for an external website publishing an executable for its 2009 version (https://www.cadforum.cz/cadforum_en/download.asp?fileID=1064)
2. *Bentley Map* and *Bentley Microstation* were tested, but it was not possible to import CityGML data sets.
3. *SketchUP+CityGML plugin* (https://forums.sketchup.com/t/citygml-plugins-for-sketchup/24921) is outdated: it worked for *SketchUP* v.2015, while the newest one is v.2019.
4. *Cesium Viewer* and *Google Earth* work only through conversion. There are different tools available for that purpose but the participants in this study chose the *3DCityDB* converter. *3DCityDB* loads CityGML into relational tables in *PostGIS* or *Oracle Spatial* and can export into CityGML, KML/COLLADA and glTF formats. Rendering is very efficient with these formats at the expense of losing complex semantics and relationships.

TABLE 3 Synthesis table of the delivered tests of support for CityGML (benchmark Task 3)

Support

Group	Software	GML support	Multi LoDs consistent	Lod3	georeferencing consistent (georef)	elevation (georef)	no need to set crs manually (georef)	avg georeferencing	entities consistent (semantics)	hierarchy consistent (semantics)	attributes (semantics)	relationships (semantics)	avg semantics	geometry	AVG SUPPORT
GIS	QGIS - json	0	0	1	1	1	0.5	0.8	0.5	0	0.5	0.5	0.4	0.5	0.6
GIS	QGIS - plain gml	0.5	0	0	1	1	1	1	1	0	1	0	0.5	0.5	0.7
GIS	ArcGIS	0.5	0	1	1	1	1	1	1	0.5	0.5	0.5	0.6	0.5	0.7
GIS	ArcGIS Pro TOTAL	0.5	0	1	1	1	0	0.7	0.5	0.5	1	0.5	0.6	1	0.8
	avg GIS software	0.4	0	0.8	1	1	0.6	0.9	0.8	0.3	0.8	0.4	0.5	0.6	0.7
Enhanced 3D viewers	tridicon CityDiscoverer Light	1	0	1	/	/	/	/	/	/	/		/	1	1
Enhanced 3D viewers	FZKViewer	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Enhanced 3D viewers	FMEDataInspector	1	0	1	1	1	1	1	1	0.5	1	0.5	0.8	1	0.9
Enhanced 3D viewers	azul	1	0	1	1	1	1	1	1	1	1	1	1	1	1
Enhanced 3D viewers	eveBIM	1	1	1	1	1	1	1	1	1	1	0	0.8	1	0.9
Enhanced 3D viewers	novaFACTORY + WEGA-3D	1	1	1	1	1	0	0.7	1	1	1	1	1	1	0.9
Enhanced 3D viewers	1Spatial Elyx 3D*	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	avg 3D viewers	1	0.6	1	1	1	0.8	0.9	1	0.9	1	0.8	0.9	1	1
ETL and conv	a) 3DCityDB	1	0.5	1	1	1	0	0.7	1	1	1	1	1	1	0.9
ETL and conv	FME 2019.2	1	0.5	1	1	1	1	1	1	0.5	1	0.5	0.8	1	0.9
	avg ETL software	1	0.5	1	1	1	0.5	0.8	1	0.8	1	0.8	0.9	1	0.9
3D modelers	CityEngine	0	0	1	1	1	0	0.7	1	0.5	0	0.5	0.5	1	0.7
3D modelers	Blender	0	1	1	0	1	1	0.7	1	1	1	1	1	1	0.9
	avg 3D modeling sw	0	0.5	1	0.5	1	0.5	0.7	1	0.8	0.5	0.8	0.8	1	0.8
Analysis	CitySim Pro	1	0.5	1	/	/	1	1	/	/	/	/	/	1	1
BIM sw	Infraworks	0	0	1	1	/	1	1	/	/	/	/	/	1	1
	AVERAGE SUPPORT	0.7	0.4	0.9	0.9	1	0.7	0.9	0.9	0.7	0.9	0.6	0.8	0.9	0.9

Functionalities (visualisation, edit, query)

Group	Software	view 3D	view 2D	textures	thematic visualization (entities)	avg visualisation	edit attributes	edit geometry	edit georeferencing	avg editing	query objects	select by attribute	spatial query	compute attribute	avg query
GIS	QGIS - json	1	1	0	0	0.5	0.5	1	1	0.8	1	1	0.5	0	0.6
GIS	QGIS - plain gml	1	1	0	0	0.5	0	0	0	0	1	1	0.5	0	0.6
GIS	ArcGIS	1	1	0	0	0.5	1	1	1	1	1	1	1	1	1
GIS	ArcGIS Pro TOTAL	1	1	1	0	0.8	1	1	1	1	1	1	1	1	1
	avg GIS software	1	1	0.3	0	0.6	0.6	0.8	0.8	0.7	1	1	0.8	0.5	0.8
Enhanced 3D viewers	tridicon CityDiscoverer Light	1	0	/	0	0.3	0	0	/	0	0	0	0	0	0
Enhanced 3D viewers	FZKViewer	1	0	0	1	0.5	0	0	0	0	1	0.5	0.5	0.5	0.6
Enhanced 3D viewers	FMEDataInspector	1	1	1	/	1	0	0	0	0	1	0.5	0.5	0.5	0.6
Enhanced 3D viewers	azul	1	0	/	0.5	0.5	0	0	0	0	1	/	/	/	1
Enhanced 3D viewers	eveBIM	1	0	1	/	0.7	1	0	0	0.3	1	1	1	1	1
Enhanced 3D viewers	novaFACTORY + WEGA-3D	1	1	/	0.5	0.8	0.5	0.5	1	0.7	1	1	1	/	1
Enhanced 3D viewers	1Spatial Elyx 3D*	1	1	1	0.5	0.9	0	0	0	0	1	1	1	1	1
	avg 3D viewers	1	0.4	0.8	0.5	0.7	0.2	0.1	0.2	0.2	0.9	0.7	0.7	0.6	0.7
ETL and conv	a) 3DCityDB	na	na				0	0	0	0	0	0	0	0	0
ETL and conv	FME 2019.2	1	1	1			1	1	1	1	1	1	1	1	1
	avg ETL software						0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
3D modelers	CityEngine	1	0	1	0	0.5	1	1	0	0.7	1	1	1	0	0.8
3D modelers	Blender	1	0	0	0.5	0.4	1	1	0	0.7	1	0.5	0.5	0.5	0.6
	avg 3D modeling sw	1	0	0.5	0.3	0.4	1	1	0	0.7	1	0.8	0.8	0.3	0.7
Analysis	CitySim Pro	1	0	0	0.5	0.4	0	0	0	0	0	0	0	0	0
BIM sw	Infraworks	1	0	1	/	0.7	0	0	0	0	0	0	0	0	0
	AVERAGE SUPPORT	1	0.5	0.5	0.3	0.6	0.4	0.4	0.3	0.4	0.8	0.7	0.6	0.4	0.6

Functionalities (analysis) and export

Group	Software	typoe 1	type 2	visibility	buffers	overlay	measurements	shadows	xml schema validation	sun analysis	more analysis	height profiles	can export	AVG FUNCTIONALITIES
GIS	QGIS - json	0.5	0.5	0	0.5	0	1	1	0	1	1	1	0.5	0.6
GIS	QGIS - plain gml	0.5	0.5	0	0	0	0	0	0	0	0	0	0	0.3
GIS	ArcGIS	0.1	0.5	0	0	0.5	1	1	0	0	0	0	0.5	0.7
GIS	ArcGIS Pro TOTAL	0.5	0.5	0	0.5	0.5	1	1	0	0	0	0	0.5	0.8
	avg GIS software	0.4	0.5	0	0.3	0.3	0.8	0.8	0	0.3	0.3	0.3	0.4	0.6
Enhanced 3D viewers	tridicon CityDiscoverer Light	0	0	0	0	0	0	0	0	0	0	0	0	0.1
Enhanced 3D viewers	FZKViewer	1	0.5	0	0	0	1	0	1	1	0	0	0	0.3
Enhanced 3D viewers	FMEDataInspector	0	0	0	0	0	1	0	0	0	0	0	0	0.3
Enhanced 3D viewers	azul	0	0	0	0	0	1	0	0	0	0	0	0	0.3
Enhanced 3D viewers	eveBIM	0	0.5	0	0	1	1	1	0	0	0	0	1	0.7
Enhanced 3D viewers	novaFACTORY + WEGA-3D	1	1	1	0	0	1	1	0	1	0	1	1	0.9
Enhanced 3D viewers	1Spatial Elyx 3D*	0	1	1	1	0	1	0	0	0	0	0	1	0.8
	avg 3D viewers	0.3	0.4	0.3	0.1	0.1	0.9	0.3	0.1	0.3	0	0.1	0.4	0.5
ETL and conv	a) 3DCityDB	0	0	0	0	0	0	0	0	0	0	0	1	0.3
ETL and conv	FME 2019.2	1	1	1	1	1	1	1	1	1	1	0	1	1
	avg ETL software	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	1	0.6
3D modelers	CityEngine	1	1	1	0	0	0	0	0	0	1	0	0	0.6
3D modelers	Blender	1	0	1	0	0	0	0	0	0	1	0	0.5	0.4
	avg 3D modeling sw	1	0.5	1	0	0	0	0	0	0	1	0	0.3	0.5
Analysis	CitySim Pro		0.5	0	0	0	0	0	0	0	0.5	0	0	0.2
BIM sw	Infraworks	0	0	0	0	0	0	0	0	0	0	0	0	0.1
	AVERAGE SUPPORT	0.4	0.4	0.3	0.2	0.2	0.6	0.4	0.1	0.2	0.3	0.1	0.4	0.5

Legend: purple = managed through CityJSON; orange = managed through FME conversion

Notes: The color scale from green to red is assigned according to the scores from 0 (no support) to 1 (full support). In the table, the software tools supporting CityGML through conversions to other formats, namely CityJSON or through *FME*-based scripts, are indicated in purple and orange, respectively.

5. *Sidefx Houdini* was proposed for testing by one of the participants, but it does not work with CityGML.
6. *iTOWNS* (http://www.itowns-project.org) does not support CityGML, unless probably through some conversion to *PostGIS* (https://github.com/Oslandia/workshop-3d-itowns/blob/master/0_Data/1_Buildings.md).
7. *HALE Studio* (https://www.wetransform.to/products/halestudio/) should also be able to support CityGML, but the function to import CityGML was not actually found in the software.

4.2 | Software support for CityGML

In this section the qualitative analysis of the responses of participants (Noardo et al., 2020b, http://doi.org/10.5281/zenodo.3966987) describing the software tools and the tests is reported. First the observations about how the GML format is read by software are summarized in Section 4.2.1. A second level of the test was about checking that the data were interpreted and read correctly, meaning their georeferencing information (Section 4.2.2), semantics (Section 4.2.3) and geometry (Section 4.2.4) were not lost in the conversion to the native formats read by software but remained interoperable instead. Those responses are summarized in Table 3.

4.2.1 | Loading of GML data

First, the support for loading (City)GML files directly by the software was evaluated. This functionality would allow conversions between different formats to be avoided, ensuring easier use of such data by experts of other domains. Also, it would prevent the introduction of further errors and inconsistencies into the data, due to the conversion process.

Notes: The color scale from green to red is assigned according to the scores from 0 (no support) to 1 (full support). In the table, the software tools supporting CityGML through conversions to other formats, namely CityJSON or through *FME*-based scripts, are indicated in purple and orange, respectively.

In the case of GIS software (*ArcGIS* and *QGIS*), direct import of GML files was only correctly done with LoD1 data, which is basically 2.5D. Moreover, very few functionalities are enabled in this case, it being possible only to visualize and query the data, without editing. The other two data sets (i.e., the LoD3 and multi-LoD files, which are both "real" 3D) were interpreted in a completely incorrect manner: geometries were only loaded as points for the multi-LoD data set; and only attributes with no geometries were loaded for the LoD3 data set.

To import the data consistently, additional software or specific plug-ins were required in both cases. For *ArcGIS*, the 'Data Interoperability' toolbox was used, which is based on the Safe Software Feature Manipulation Engine (FME). For *QGIS* there are several plug-ins, of which only one worked (CityJSON Loader, https://github.com/tudelft3d/cityjson-qgis-plugin) which requires conversion to the CityJSON format. The conversion to a relational database through *3DcityDB* would also work in both cases, but again that involves a conversion and external tools in the process.

The group of 'extended 3D viewers' are a different matter, since they were implemented specifically to work with the (City)GML format and can, therefore, read it directly. The same is true for the ETL and conversion tools, able to interpret the GML format, since their aim is generally to transform it to something else. Also in *CitySim Pro* a specific import for CityGML was implemented, and works.

That is not the case for 3D modelers and BIM software, where external tools are needed: the proprietary software packages *ESRI CityEngine* and *Autodesk Infraworks* only manage their own proprietary format, therefore import through a connected *FME*-based converter is necessary. *Blender* (open source) is able to manage several more standardized formats, but a specific plug-in is needed and, furthermore, this is able to import the data in CityJSON, meaning that the data should be converted to CityJSON.[4]

To summarize this issue, 70% of the tested software programs can read GML directly, however, most of them (nine out of 15) were specifically programmed for CityGML. The remainder need conversions that usually go through *FME* processing (especially the proprietary ones) and/or through the conversion to CityJSON[5] or require the use of plug-ins.

Another point regarding the support of software for the CityGML data model concerns the more complex kinds of geometry management, once imported, namely: the consistent interpretation and visualization or use for analysis of multi-LoD data sets and functioning with LoD3 data.

The consistent interpretation of multi-LoD is intended as the possibility to read the information of the CityGML objects associated to the various LoDs geometries at once, among which it is possible to choose from the same interface, for both visualization or eventual use of that in analysis. This was tested through the *RotterdamLoD12.gml* data set. Only 40% of the software tested is able to manage them consistently:

1. No consistency in GIS (both read together and superimposed).
2. Partial support from the ETL and conversion tools (in *FME* it is either considered as a unique aggregate or it is possible to choose to upload them separately and in *3DcityDB* it is necessary to choose which LoD to work with).
3. Being based on *FME*, the same is true for *CityEngine* and *Infraworks*.
4. Partial support is provided in *CitySim Pro*, since only LoD2 and LoD3 are allowed, therefore it was difficult for us to test it.
5. The issue is well managed by most extended 3D viewers (except for *tridicon CityDiscoverer Light*, *FMEDataInspector* and *azul*).
6. *Blender* can also manage them consistently (through the CityJSON format).

LoD3, tested through the *BuildingsLoD3.gml* data set, is well supported by all of the software tested, once the necessary conversions are made (it is not read only by *QGIS* if the GML format is used).

4.2.2 | Loading georeferencing

Looking at georeferencing, the information about coordinate reference systems (CRS) can be consistently read by all the software managing the files (except for *Blender*, which is not designed to manage georeferenced objects), as well as heights. In a few cases it is necessary to set the CRS manually, for example in *QGIS* or ArcGIS in some cases, in *novaFACTORY* and it also has to be explicitly set in *FME* and, consequently, *CityEngine*. No relevant problems are found for this aspect, as expected in software for managing geoinformation.

4.2.3 | Loading semantics

More difficult management is observed for semantics:

1. 'Extended' 3D viewers as well as ETL and conversion tools can usually interpret semantics properly, as consistency of entities, hierarchies, attributes and further relationships, with some exceptions, for example, the hierarchy and relationships are managed through parent IDs in relational database fashion in some cases (*FME* and related software).
2. *Tridicon CityDiscoverer Light* has no functionality to read either georeferencing or semantics information.
3. In the GIS tools the semantics is converted to the internal, relational, structure of data, therefore some part of the information is always lost or managed through different tables, connected by means of IDs, that make the structure much more complex and less usable.

In *QGIS* the testers[6] pointed out that sometimes the same entity information can be lost (wall, door, building), but the attributes are listed under a generic *cityobjectmember*, when managed through the GML format. In the CityJSON case, the geometries are correctly recognized, but they cannot be accessed through the attribute table.

The entity name (Building, BuildingPart, etc.) is kept in the attribute 'type' of the CityJSON format. The relationships are managed through IDs stored in the attribute tables. The same management of attributes is sometimes made more complex by their storage in different tables, connected through IDs.

4.2.4 | Loading geometry

The tools tested had little possibility to assess the validity and correctness if geometry, and more results are found after analyzing the exported models (Section 4.4). However, the geometries look generally good, except in GIS software when directly importing GML files without conversion. In that case we already discussed that 3D (LoD3) or multi-LoD geometries were not read correctly.

A secondary question about the support for CityGML concerned the possibility of managing ADE information. This is managed by *FME* and some of the software using *FME*-based procedures (*ArcGIS*), *FZKViewer*, through the addition of the schemas in the software files, *eveBIM*, *novaFACTORY* and *Blender*. In none of them is it possible, however, to use the information for analysis, except for *FME* and *ArcGIS*, as stated in the responses. In the other cases such information can only be viewed and sometimes queried.

4.2.5 | Using CityGML data

The functionalities requested for testing were visualization, editing, query and analysis.

All the tools tested can visualize the data in 3D and some of them (the GIS software, the ETL and conversion tools, plus *1Spatial Elyx 3D* and *FME Data Inspector*) can also do so in 2D, which is not, however, the priority of those tools or of those data.

There is also partial support for the visualization of textures, consistently provided by half of the software tools: *ArcGIS Pro*, *FME Data Inspector*, *FME*, *CityEngine*, *Infraworks* (all based on *FME* conversion and enabled to visualize textures in their native formats), *eveBIM* and *1Spatial Elyx3D*. The thematic visualization (i.e., the application of symbology based on variables such as the name of entities, attribute values, and queries) is enabled as association of different colors to different entities in *azul*, *novaFactory*, *1Spatial Elyx3D*, *Blender* and *CitySim Pro*. *FZKViewer* also allows some thematic symbology based on the values of some attributes. A drawback of this is that only a limited amount of pre-set thematization is allowed, without full customization being possible. But it is still the most advanced application for this functionality, which could be very useful for current GIS users to use 3D information. *QGIS* also allows the application of symbology to 3D data in the 3D viewer.

The possibility to edit the data is also important for users. In this task, the GIS tools offer better support. Usually, it is possible to edit attributes, geometry (e.g., by moving vertexes) and georeferencing. However, this is not possible with GML data in *QGIS*. Limited editing is possible in enhanced 3D viewers: for example, it is only possible to remove objects from the *FZKViewer*, *novaFACTORY* can edit attributes and geometry through a module called 'Feature3D' and additional plug-ins, and *1Spatial Elyx 3D* can edit attributes only if the data are converted to relational DBMS and imported in that format.

We observe that, unfortunately, the tools which can best read the GML format (the enhanced 3D viewers) are the least able to manage editing. This is likely because good functionality is achieved partly through a conversion to an optimized internal data model, from which it is hard to export back to CityGML.

Moreover, any kind of change is possible through the workflows of *FME*, and partially in *CityEngine* and *Blender*, where it is possible to edit attributes and geometry. The format edited in *CityEngine* is the native software format as converted by *FME*, while in *Blender* the CityJSON format is managed and can be edited. Finally, *CitySim Pro* allows the editing of some energy-related parameters, external to the data set, according to its scope. Also,

3DcityDB does not include editing, query and analysis functionalities, but of course those are not the aim of the software and it is not considered a limitation.

The query functionality is another important requirement for working with data effectively. Table 3 shows that most of the software tested, especially GIS and 3D viewers (except *tridicon CityDiscoverer Light*), allow objects to be queried directly (by 'clicking' on them) in order to read the object attributes. *CityEngine* and *Blender* also allow that. More complex queries, such as the selection of entities based on rules, are possible in some of the software. They are well managed in GIS, although the 2D footprint of the geometry is often considered when running spatial queries. In 3D viewers, except for *tridicon CityDiscoverer Light*, all sort of queries are also possible, even if reduced in some cases, for example, very simple queries can be performed in *FME Data Inspector*, like looking for one attribute in a table, and in the *FZKViewer* they are quite predefined. In the case of *FME* and *Blender*, any kind of query is supported but it has to be specifically programmed by the user—through the use of *FME* transformers in *FME* or in Python in the case of *Blender*.

The data analysis functionalities of the software (regarding geometry, semantics and georeferencing validity) and the simulation of interaction of objects with their context (Type 1 and Type 2, respectively) were considered separately. These topics will not be treated exhaustively, since not all the participants reported on them with the best possible accuracy and, moreover, the least expert of them could not know how to get to the analysis tools that were not easy to access from the user interface or less documented. Nevertheless, from the tests it is possible to see that, generally, partial support for analysis is provided in GIS tools, which mainly work with 2D information. Sometimes additional plug-ins or toolboxes can perform further 3D analysis. Such tools are quite new and little developed even for 3D information in the same software native format, therefore we did not have the chance to determine whether they work differently when used on CityGML data. In *ArcGIS*, the *3D Analyst* extension is necessary; this is used to deal with 2.5D information such as digital terrain and surface models, and is probably also being extended for real 3D. Moreover, it is possible to program more analysis through Python scripts, although these are not so user-friendly for the least expert users. Very little analysis is possible within 3D viewers: most of the time measurements are possible on the model, some energy analysis can be done in *FZKViewer* and *novaFACTORY*, shadows in some cases (*eveBIM* and *novaFACTORY*), overlay and buffers and visibility analysis can be performed in *novaFACTORY*, *1Spatial Elyx 3D*, *CityEngine* and *Blender*. Moreover, validation of the schema can be done in *FZKViewer*; the same, plus some geometric validation and many other analyses, can be done in *FME*, which is the one most supporting functionalities of this kind, through building workflows by means of its transformers. The specific goal of *CitySim Pro* is to run energy analyses. However, in this test those did not work with the CityGML data sets provided. The implementation of analysis tools in software supporting CityGML is therefore not yet well developed, with some exceptions for visibility and shadow analyses and some energy tools. The availability of tools analyzing 3D information is generally still limited with respect to the 2D counterparts, anyway.

Finally, few software packages can export 3D city models back to CityGML. Notably, this is true even when the software is able to import the CityGML data through *FME* (e.g., *CityEngine*, *Infraworks*), which itself has CityGML export functionality. Some GIS software can export data to CityGML through the same tools they used for the import (*ArcGIS Data Interoperabilty toolbox* and *CityGML toolbox for CityJSON* in *QGIS*). In some 3D viewers, the data are not converted at all, therefore they are just saved with any new changes without being exported (which implies a conversion). Moreover, the ETL and conversion tools are developed on purpose to run import–export processing, and therefore are able to export too.

4.3 | Software performance with CityGML data

A total of 21 different reports were returned, for 15 different software packages. In particular, we received multiple results for *Esri ArcGIS Pro* (2 sets), *Esri ArcGIS* (3 sets), *QGIS* (3 sets) and *FME* (2 sets). These offer the opportunity for timing comparisons to investigate the impact of hardware on software performance.

Figure 1 gives a summary of the success rates returned for the tests on the three data sets. Figure 2 gives the count of the different timing values for the successful tests.

Note that in some cases users reported results for some of the tests but did not report results for all of the tests ("No result reported"). Additionally, some users typed in comments such as "no error" instead of giving specific timing. These are included in the "No result reported" count.

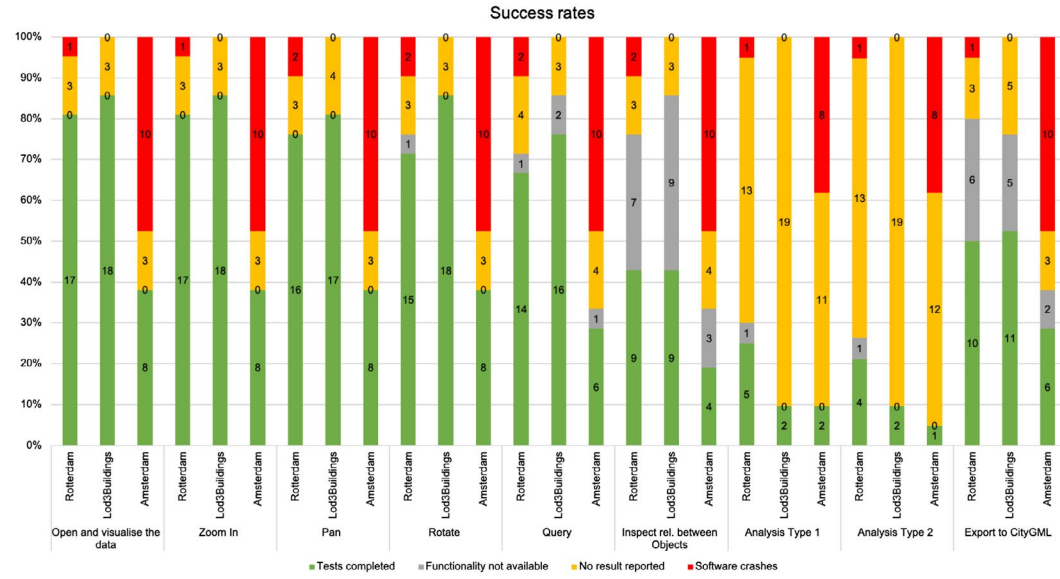

FIGURE 1 Graph reporting the success rate of the tested tools in the various tasks (in the bars, the number of tools for each case).

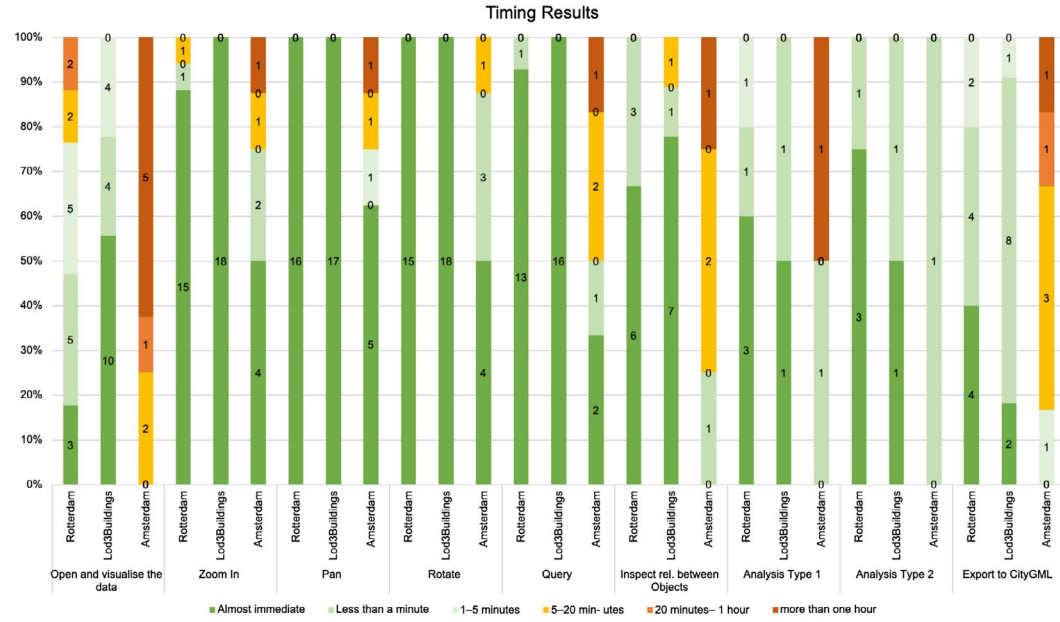

FIGURE 2 Graph reporting the timing results of the tested tools in the various tasks (in the bars, the number of tools for each case).

For the *RotterdamLoD12.gml* data set, two out of 17 of the successfully completed tests took over 20 min to open the data set (*City Engine* and *ArcGIS*, with the second *ArcGIS* test reporting that this task took between 5 and 20 min). The vast majority of the software packages tested could zoom and pan the data immediately, with only one package taking between 5 and 20 min to zoom into the data set (*eveBIM*). Only 10 out of the 21 tests report a successful export to CityGML, with all of these taking less than 5 min to execute.

With the *BuildingsLoD3.gml* data set, all 18 of the successful attempts to open the data set took less than 5 min. This reflects the small size of this data set, and similarly all 11 successful attempts to re-export the data took less than 5 min. Rotation, zoom and plan are very rapid—but one report notes a time of between 5 and 20 min to inspect relationships between objects (*Esri ArcGIS*).

The results with the *Amsterdam.gml* data set demonstrate clearly the impact of the larger data set on the tests carried out, with only eight out of 21 reports indicating that the software was able to handle the data, with an additional 10 reporting software crashes. The three "no result reported" instances here relate to *QGIS*, where the testers either displayed the data as points or first converted the data to CityJSON, which is not what the task required.

None of testers reported a time of less than 5 min to open the data set, and five reported a time of over 1 hr. *3DCityDB* reported an export time of 1–5 min, perhaps due to its bespoke development for CityGML, in contrast to the generic functionality offered by many of the other software packages.

4.3.1 | Multiple tests on same software packages

The crowdsourcing approach taken in this project resulted in multiple participants testing the same software, providing an opportunity for comparison. However, the three *QGIS* tests were eliminated from the comparison as one imported the data into points and the other two created CityJSON data.

1. Comparing the *FME Data Inspector* tests, many of the results obtained were identical in terms of performance time. A slight difference (1–5 min versus less than 1 min for the import of the *RotterdamLoD12.gml* data) could perhaps be attributed to the different RAM values, with the first participant having 16 GB and the second 32 GB. However, this did not make a difference in terms of the export time for the *Amsterdam.gml* data set, reported at 5–20 min by both participants.
2. For the two *ArcGIS Pro* tests, one user reports that the *RotterdamLoD12.gml* data set caused the software to crash when zooming and panning, meaning that results could not be compared. All other results (*Amsterdam.gml* and *BuildingsLoD3.gml*) were similar.
3. For the *Amsterdam.gml* data set, two out of three of the *ArcGIS* testers managed to import and visualize the data, whereas the third user reported that this crashed their system. This user also reported slower export times (1–5 min) for the *BuildingsLoD3.gml* and *RotterdamLoD12.gml* data sets (in comparison to less than a minute reported by the other two users). Examining the hardware used, it can be noted that all three users had 16 GB of RAM in their machines and dedicated graphics cards, and the only potentially significant difference between the machines is that the first two were run on an Intel i7 machine and the third was run on an Intel i5 machine; however, the latter has a reported CPU speed (from the user) of 3.2 GHz whereas the former two both use the Intel i7–8750H chip which reports a speed of between 2.2 and 4.1 GHz. Similar amounts of hard drive space were available for all three tests (270, 210 and 298 GB). Perhaps importantly, it was not specified whether the hard drives were solid state, which could be a significant factor when importing a large data set and writing it temporarily to disk.
4. The first *FZKViewer* respondent was able to perform some timing tests on analytical tasks using the Amsterdam data, and reports that tests of Type 1 took more than 1 hr (XML Schema validation) and less than a minute (distance measurement). However, while the first report for *FZKViewer* reported timings for some analytical tasks, the second report did not include this information. The second *FZKViewer* report did not include time measurements for panning the *BuildingsLoD3.gml* data set. While visualization times for both *FZKViewer* reports

were equal for the *Amsterdam.gml* data set, a marked difference was evident when it came to zooming into and panning around the model, with one user reporting a time of more than 1 hr and the other "it is almost immediate". For query and inspection, the first test reported 5–20 min whereas the second reported "more than 1 hr". There is no particular difference in the hardware used for testing that would account for this difference (both machines had 32 GB RAM and ran Windows 10 with an Intel i7 processor). Additionally, for the other visualization test (rotation) very similar results are reported.

4.4 | Writing of CityGML files

The data exported from the tested tools (Noardo et al., 2020a, https://doi.org/10.5281/zenodo.3966915) and delivered by participants were analyzed by means of the tools described in Section 3.3.

The information about georeferencing is always kept unchanged, with two exceptions: *novaFACTORY* for exporting *Amsterdam.gml* and one of the *ArcGIS* tests exporting *BuildingLoD3.gml* data set. The *Amsterdam.gml* data set exported by *novaFACTORY*, reports the reference system Amersfoort/RD New, EPSG:28992, Dutch national reference system for plane coordinates, instead of EPSG:7415, which simply associates the same CRS to the Dutch national height system ('NAP height'). Consequently, the *z*-values of the file bounding box are different, although it is possible to change the reference system to the correct one for exporting. In one of the tests with *ArcGIS* for the *BuildingsLoD3.gml* data set, the coordinates of the bounding box are moved slightly from minimum (−0.7, −0.61, 0) to (0, 0, 0) and maximum (70.39, 67.54, 16.7) to (69.93, 67.21, 16.7), with related consequences in the total dimension of the bounding box and area of the base surface.

Tables 4–6 show the differences between the statistics and computed features (related to both semantics and geometry) in the original data sets and the ones exported by the tools tested. In the left-hand panels the absolute

TABLE 4 Comparison between the *Amsterdam.gml* models exported back to CityGML by participants with respect to the original one

Amsterdam	Difference absolute values						Difference percentage				
File Statistics	NovaFACTORY	FMEDataInspector	FMEDataInspector	3DcityDB	FME	ORIGINAL FILE	NovaFACTORY	FMEDataInspector	FMEDataInspector	3DcityDB	FME
Entities	-178785	0	-2682	-12885	0	319118	-56.02	0.00	-0.84	-4.04	0.00
bldg:Building	-90260	0	0	0	0	90260	-100.00	0.00	0.00	0.00	0.00
brid:Bridge	-2682	0	-2682	0	0	2682	-100.00	0.00	-100.00	0.00	0.00
core:CityModel	0	0	0	0	0	1	0.00	0.00	0.00	0.00	0.00
genobj:GenericCityObject	-10504	0	0	0	0	10504	-100.00	0.00	0.00	0.00	0.00
landuse:LandUse	0	0	0	0	0	103043	0.00	0.00	0.00	0.00	0.00
trpt:Road	-62454	0	0	0	0	97668	-63.95	0.00	0.00	0.00	0.00
veget:PlantCover	-12885	0	0	-12885	0	12885	-100.00	0.00	0.00	-100.00	0.00
water:WaterBody	0	0	0	0	0	2075	0.00	0.00	0.00	0.00	0.00
Relations	-178785	0	-2682	-12885	0	319117	-56.02	0.00	-0.84	-4.04	0.00
GmlRelFeature	-178785	0	-2682	-12885	0	319117	-56.02	0.00	-0.84	-4.04	0.00
Geometry Information											
Number of points:	-29775282	0	-4077698	-8353908	0	45038667	-66.11	0.00	-9.05	-18.55	0.00
Number of polygons:	-974319	0	-71906	0	0	974319	-100.00	0.00	-7.38	0.00	0.00
Number of triangles:	-8560678	0	-1230696	-2784636	0	13648473	-62.72	0.00	-9.02	-20.40	0.00
Number of faces:	-7514018	0	-292112	-2784636	0	12601813	-59.63	0.00	-2.32	-22.10	0.00
Area of Base Surface	0	0	0	0	0	1.22264e+08	0.00	0.00	0.00	0.00	0.00
Volume:	-7.785E+14	0	0	0	0	1.62699e+10	-47.85	0.00	0.00	0.00	0.00
Faulty faces	-2994	0	-334	-163	0	4384	-68.29	0.00	-7.62	-3.72	0.00
Geometry type											
Face	-6502991	0	-292112	-2784636	0	11590786	-56.10	0.00	-2.52	-24.02	0.00
GmlSolid	-90260	0	0	0	0	90260	-100.00	0.00	0.00	0.00	0.00
MultiSurface	-269045	0	-183202	-12885	0	409376	-65.72	-44.75	0.00	-3.15	0.00

Note: Orange is used for missing entities and green denotes no change.

TABLE 5 Comparison between the *BuildingsLoD3.gml* models exported back to CityGML by participants with respect to the original one

Buildings LoD3	Difference absolute values										
	With validated data						With initial data (non validschema)				
File statistics	ArcGIS	ArcGIS Pro	NovaFACTORY	FMEWorkbench	CityEngine	ORIGINAL FILE	ArcGIS	FME Data Inspector	FME Data Inspector	3DcityDB	ORIGINAL FILE
Entities	776	1905	1	0	1905	**204**	-325	12	18	-3	666
bldg:Building	-16	0	0	0	0	**16**	-15	0	0	0	16
bldg:BuildingInstallation	-4	-4	0	0	-4	**4**	166	0	0	-3	4
bldg:ClosureSurface	-6	-6	0	0	-6	**6**	170	6	6	-6	0
bldg:Door	-16	-16	0	0	-16	**16**	-16	0	0	0	16
bldg:GroundSurface	-16	0	0	0	0	**16**	-16	0	0	0	16
bldg:OuterFloorSurface	-1	0	0	0	0	**1**	-1	-1	0	0	1
bldg:RoofSurface	-55	-1	0	0	-1	**55**	-55	-1	0	0	55
bldg:WallSurface	-89	1916	0	0	1916	**89**	-77	8	12	0	77
core:CityModel	0	0	0	0	0	**1**	0	0	0	0	1
bldg:BuildingPart	0	16	0	0	16	**0**	0	0	0	0	0
bldg:Window	0	0	0	0	0	**0**	-480	-1	0	0	480
Relations	776	1905	17	0	1905	**203**	-325	12	18	-3	665
GmlRelFeature	776	1905	17	0	1905	**203**	-325	12	18	-3	665
genobj:GenericCityObject	85	0	1	0	0	0	170	1	0	0	0
genobj:intAttribute	83	0	0	0	0	0	170	0	0	0	0
genobj:stringAttribute	785	0	0	0	0	0	0	0	0	0	0
Geometry information											
Number of points:	-520	16130	0	0	15992	**16364**	-1818	106	2124	1836	18200
Number of polygons (convex polygons):	-238	1874	0	0	1943	**2021**	-462	28	432	459	2480
Number of triangles:	144	2878	0	0	2740	**2760**	10	-2	132	0	2760
Number of faces:	-94	2034	0	0	2034	**2118**	-457	29	498	459	2577
Geometry type											
CompositeSurface	0	16	0	0	16	0	0	0	0	0	0
Face	-94	2034	0	0	2034	**2118**	2120	488	498	459	2577
MultiSurface	-94	1909	0	0	1909	**183**	170	491	498	462	645
SetOfGeometry	0	0	0	0	0	0	0	3	3	0	0

Difference percentage									
With validated data						With initial data (non valid schema)			
ArcGIS	ArcGIS Pro	NovaFACTORY	FMEWorkbench	CityEngine	ArcGIS	FME Data Inspector	FME Data Inspector	3DcityDB
380.39	933.82	0.49	0.00	933.82	-48.80	1.80	2.70	-0.45
-100.00	0.00	0.00	0.00	0.00	-93.75	0.00	0.00	0.00
-100.00	-100.00	0.00	0.00	-100.00	4150.00	0.00	0.00	-75.00
-100.00	-100.00	0.00	0.00	-100.00				
-100.00	-100.00	0.00	0.00	-100.00	-100.00	0.00	0.00	0.00
-100.00	0.00	0.00	0.00	0.00	-100.00	0.00	0.00	0.00
-100.00	0.00	0.00	0.00	0.00	-100.00	-100.00	0.00	0.00
-100.00	-1.82	0.00	0.00	-1.82	-100.00	-1.82	0.00	0.00
-100.00	2152.81	0.00	0.00	2152.81	-100.00	10.39	15.58	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					-100.00	-0.21		
382.27	938.42	8.37	0.00	938.42	-48.87	1.80	2.71	-0.45
382.27	938.42	8.37	0.00	938.42	-48.87	1.80	2.71	-0.45
-3.18	98.57	0	0	97.73	-9.99	0.58	11.67	10.09
-11.78	92.73	0	0	96.14	-18.63	1.13	17.42	18.51
5.22	104.28	0	0	99.28	0.36	-0.07	4.78	0
-4.44	96.03	0	0	96.03	-17.73	1.13	19.32	17.81
-4.44	96.03	0	0	96.03	-17.73	18.94	19.32	17.81
-51.37	1043.17	0	0	1043.17	-73.64	76.12	77.21	71.63

Note: Orange is used for missing entities, red for added entities and green denotes no change.

TABLE 6 Comparison between the *RotterdamLoD12.gml* models exported back to CityGML by participants with respect to the original one

Rotterdam LoD12	Difference absolute values							Difference percentage					
File statistics	ArcGIS	ArcGIS	FMEDataInspector	FMEDataInspector	3DcityDB	FME	ORIGINAL FILE	ArcGIS	ArcGIS	FMEDataInspector	FMEDataInspector	3DcityDB	FME
Entities	-13559	54388	0	-16	0	-1528	16056	-84.45	338.74	0.00	-0.10	0.00	-9.52
bldg:Building	-252	-252	0	0	0	0	252	-100.00	-100.00	0.00	0.00	0.00	0.00
bldg:BuildingInstallation	-4	-4	0	0	0	0	4	-100.00	-100.00	0.00	0.00	0.00	0.00
bldg:BuildingPart	-398	-398	0	0	0	0	398	-100.00	-100.00	0.00	0.00	0.00	0.00
bldg:ClosureSurface	-2234	-2234	0	0	0	0	2234	-100.00	-100.00	0.00	0.00	0.00	0.00
bldg:GroundSurface	-591	-591	0	0	0	0	591	-100.00	-100.00	0.00	0.00	0.00	0.00
bldg:OuterCeilingSurface	-125	-125	0	-125	0	0	125	-100.00	-100.00	0.00	-100.00	0.00	0.00
bldg:RoofSurface	-2580	-2751	0	-4	0	0	2751	-93.78	-100.00	0.00	-0.15	0.00	0.00
bldg:WallSurface	-7827	-7827	0	-12	0	0	7827	-100.00	-100.00	0.00	-0.15	0.00	0.00
core:CityModel	0	0	0	0	0	0	1	0.00	0.00	0.00	0.00	0.00	0.00
core:xalAddress	-174	-345	0	0	0	0	345	-50.43	-100.00	0.00	0.00	0.00	0.00
Relations	-13559	54388	0	-16	0	-1528	16055	-84.45	338.76	0.00	-0.10	0.00	-9.52
GmlRelFeature	-13559	54388	0	-16	0	-1528	16055	-84.45	338.76	0.00	-0.10	0.00	-9.52
genobj:GenericCityObject	562	7827	0	125	0	0	0						
genobj:intAttribute	-162	7103	-724	-724	0	-724	724	-22.38	981.08	-100.00	-100.00	0.00	-100.00
genobj:stringAttribute	226	53985	724	724	0	-804	804	28.11	6714.55	90.05	90.05	0.00	-100.00
Geometry information													
Number of points:	-41514	-193346	-164042	-6537	78836	78836	242878	-17.09	-79.61	-67.54	-2.69	32.46	32.46
Number of polygons (conv	-4257	-16770	-17653	-36	9423	9423	27076	-15.72	-61.94	-65.20	-0.13	34.80	34.80
Number of triangles:	-7236	-40485	-29180	-2122	13362	13362	42542	-17.01	-95.16	-68.59	-4.99	31.41	31.41
Number of faces:	-7129	-23837	-24595	-141	10305	10305	34900	-20.43	-68.30	-70.47	-0.40	29.53	29.53
Geometry Type													
Face	10321	-6387	0	-141	0	0	17450	59.15	-36.60	0.00	-0.81	0.00	0.00
GmlSolid	-1092	-1092	0	0	0	0	1092	-100.00	-100.00	0.00	0.00	0.00	0.00
MultiPoint	-345	-345	0	0	0	0	345	-100.00	-100.00	0.00	0.00	0.00	0.00
MultiSurface	-12966	-5701	0	-141	0	0	13528	-95.85	-42.14	0.00	-1.04	0.00	0.00
Vertex	-345	-345	0	0	0	0	345	-100.00	-100.00	0.00	0.00	0.00	0.00

Note: Orange is used for missing entities, red for added entities and green denotes no change.

numbers of features that are lost (shaded orange) or generated (shaded red gradients) with respect to the original files are counted. Only when the value is 0 (green) were there no changes in the files and was the data exchange successful. These tables also include the entities and geometric elements which were not present in the original files, but appeared instead in the exported ones.

In the right-hand panels the same values are expressed as percentages of the whole original amount, in order to allow a more meaningful representation. We can understand how the interoperability is only achieved by FME and only with the *Amsterdam.gml* and *BuildingsLoD3.gml* data sets. It does not completely maintain the features unchanged in the *RotterdamLoD12.gml*. This would not be acceptable for use in practice.

Since a software package could be tested more than once, with different results, the same software can appear more than once in the table, reporting the analysis of the delivered models.

With the *Amsterdam.gml* data set (Table 4) a lot of entities are lost. This may be due to the large size of the data set, requiring very high performance by computers, which may not be able to complete the export task completely (this is probably what happened with the *novaFACTORY* software). In other cases (i.e., one of the *FME Data Inspector* tests and *3DcityDB*), we observe that some specific entities are lost completely, since probably not read at all, and the related counts are consequently affected: *bldg:Bridges* in *FME Data Inspector* and *veget:PlantCover* in *3DcityDB*.

In the *BuildingsLoD3.gml* data set also some entities are completely lost. However, in this case some other objects appear, such as *bldg:WallSurface* or *bldg:BuildingPart*. This means that some internal undocumented transformation happens during import–export conversions, causing a change in entity storage or a split or duplication of objects.

A similar behavior is also recorded for the *RotterdamLoD12.gml* data set, in which, in addition, many *genobj:stringAttributes* unexpectedly appear. Some of them derive from the change of type of some attributes previously stored as *genobj:intAttributes*. There is no apparent explanation in other cases. Furthermore, the increase in the number of geometric elements in the *3DcityDB* and *FME* tests remains unexplained, as well as the loss of geometries in one of the *FME Data Inspector* tests, where no other changes are recorded in the number of entities.

5 | DISCUSSION

5.1 | Limited interoperability: A matter of complexity?

Although we expected clearer patterns in the results, which would make it possible to better understand the remaining problems of interoperability in CityGML models, the only clear result is that very little interoperability is actually achieved. There are very few tools able to read the standardized data sets correctly and even fewer that are able to export them consistently. The ability to uniquely interpret the models and to leave them consistent through the import–export phases is absolutely essential for interoperability and what it enables (data exchange, data reuse, etc.). At this stage it is not possible to trust standardized models though, even for simple file exchange.

Furthermore, the 3D city models are expected to be usable and powerful to support analysis, problem-solving and management actions. However, the inability of tools to manage 3D information, and especially standardized 3D models in CityGML, critically hinders this potential. This is much more apparent if looking at the discrepancy between the ability of software packages to correctly read and interpret the data, and their functionalities for editing, querying, and analyzing them. In fact, the tools which are better at supporting those functionalities (e.g., GIS) are not the best at interpreting the standardized data set contents, and vice versa. The frustration expressed informally, as reported in Section 1, about the gap between users' expectations and what is actually feasible (especially when they are just users and not geomatics or 3D modeling expert programmers) is therefore justified by our study.

Results were reported for 15 software packages, including both bespoke CityGML viewers but also generic GIS tools. Support for CityGML is very mixed, and in particular it is not directly supported (in the GML form) in perhaps the most popular open source GIS package, *QGIS*. CityGML is supported in the most popular commercial package, *ArcGIS* (including Pro), but only one out of five users was able to display an extended 3D city model (*Amsterdam.gml*) even though the display and management of an entire city model is likely to be a task that users will be interested in accomplishing.

While georeferencing information is supported quite well by the tools analyzed, problems were reported with semantic and geometric properties.

From the experience gained, we can see how probably some difficulties in the implementation lie in the standard itself. The management of semantics is sometimes problematic (e.g., loss, change), with the main issues related to the management of hierarchies and other relationships. Despite being one of the most exciting possibilities in models of this kind, the lack of suitable support should compel a rethink of complex relationships to make them simpler and effectively manageable. A clear definition of how to structure entities, priorities and limits is also needed. Similar points can be made for geometries: constrained validity rules would help a homogeneous implementation, resulting in homogeneous and consistent models.

A general recommendation is therefore a collaboration of software developers with standardization institutions, starting from the needs of practitioners and users.

For applications, it is often preferable to rely on a simpler model that is implemented consistently in all applications, than to dispose of a more complex one that will be implemented in fewer applications and with a lower degree of consistency. Most applications clearly load CityGML data sets into their own internal data models: something that can be seen most clearly in the differences in the imported and exported files, which proves that the assumption that the complex data model of CityGML is directly implemented in applications is false.

The issues and challenges shown by the results of this test bench are influenced by a number of factors, among which are the following:

1. The complexity of CityGML and implicitly of 3D city models; a complex data model such as CityGML is required to describe and render a wide range of characteristics of the city, from both a geometrical point of view but also from a semantic point of view.
2. A CityGML data set combines features and information from both 3D graphics and geoinformation domains, which in turn causes issues for various software packages which are not able to handle this data model easily.

Some software is tailored to more specific domains: geoinformation software (GIS, ETL, etc.) which does not always focus on complex data structures or 3D graphics; whereas other software is more tailored for 3D graphics/visualization, such as 3D modelers, but does not emphasize semantics.

3. Another factor which may influence interoperability issues is the way the standard is exploited and used by users (e.g., how they populate the schema and which encoding they use).
4. Last, but not least, issues may also arise due to the way software uses computational resources. For example, some software is more efficient in the use of CPU and GPU power and could handle complex CityGML data in a more efficient way, whereas other software only relies on CPU power (and may not use simultaneously more than one CPU thread), therefore causing issues when reading/writing/exporting data, which could also mean other types of data, not just CityGML.

5.2 | The computational load

While it is not possible to say which software package is fastest (the approach to timing used general timing categories rather than requiring the user to undertake the onerous task of time measurement), and performance will also depend on the hardware being used, we can report that none of the software packages managed to carry out the visualization task in under 5 min for the *Amsterdam.gml* data, although 13 packages achieved this with the *RotterdamLoD12.gml* data set and 18 for the smaller *BuildingsLoD3.gml* data set. For CityGML export 10 software packages managed to export the data in 5 min or less for *RotterdamLoD12.gml*, 11 for *BuildingsLoD3.gml* but only 1 for *Amsterdam.gml*.

A simplification of the geometry representation mechanisms could improve the related software performance, greatly affecting the time spent parsing and loading files, as well as the memory used while parsing the notoriously verbose GML geometry definitions. Since the industry trend is towards more and more complex and rich city models, including heavy monitoring and sensor data, there is an urgent need for a more agile system to store 3D city models.

5.3 | The voluntary participation

The fact that the inquiry is based on voluntary and completely open contributions can be considered both a strength and a limitation of this work. It is essential to cover the investigated object in the most thorough way: as many software packages as possible, with many experts involved, but also with the inclusion of less expert users to test user-friendliness. However, the downside is the incompleteness of the resulting tools review. This has been limited by the integration of additional packages in the testing that had not been considered at the beginning. Moreover, the tests reporting suspicious results according to the experiences of the promoting team (for both too good or too bad performances) were double-checked with new tests or by asking for clarifications. A further issue could be the inexperience of some testers, reporting inaccurately on tools' behavior. In mitigation, it was checked that all responses were given with sufficient care, whatever the level of expertise. Once verified, any conflicting responses with respect to the tools' actual potential could indicate a deficiency in the suitability of the tool for an inexpert user. The involvement of a large part of the community is also important to perceive the relevance of the topic, although dealing with an issue considered not to be a priority among the high-level standardization and academic communities.

6 | CONCLUSIONS

This study was designed to point out and provide evidence on the support and issues of available software for standardized information in CityGML version 2.0. Interoperability is essential for a number of use cases, and even

for merely exchange and reuse data. Standards are supposed to be enabling such interoperability and standardization is the essential premise for the development of any integration, including that of GeoBIM. In fact, the potential integration with other kinds of data, including BIM and respective standards, requires structures and formats to be respected reliably within data sets. Data which are compliant with a standard are not supposed to change or vary when produced or used by different tools, otherwise, it would be almost impossible to plan, design and implement effective solutions for mapping, conversions and object transformation to other formats.

CityGML is a popular open reference standard for 3D city model management and storage. However, a number of issues are informally reported in the use of CityGML as data format, preventing the effective use of such data sets. But no systematic proof was available as the basis for future improvements in implementations, in data modeling and in the standard itself.

Possible bias in the results could derive from the lack of expertise of participants making the tests, or from the initial inaccuracies in the data sets provided, modelled within real world practice. However, such data sets were validated and improved for the purpose of the benchmark, in order to limit their influence on the quality and reliability of results. Therefore, if residual inaccuracies were detected, despite great efforts to control them, it would reflect additional drawbacks of the standard itself: for example, lack of clarity about its use for the modeling of actual data sets, and difficulty in implementation, which could produce less intuitive tools.

This study shows the drawbacks of the CityGML (v. 2.0) standard and implementation, and the related difficulties, also due to the challenges to which it is intended to respond (e.g., representation of the information regarding a huge and complex field, flexibility for multiple needs). The outcomes are of great importance and can form the basis for future research in the field and development of concrete solutions, such as the addition of constraints and specific guidelines, simpler ways to store geometry, better selection of useful semantics, and so on.

Considering the results of this study as evidence, future work should be aimed at resolving the issues outlined — for example, the test for specific kinds of geometries, how to constrain them and how to guarantee that software can import, read, use and re-export them without any change. The same is true for semantics: which categories and relationships are essential for the models to be useful? How can they be simplified without loss of effectiveness? Moreover, when considering performance related to computational requirements, we can easily understand that the reduction of data size is urgent.

Work is already being done to solve such issues, towards less complex models offering more straightforward choices and easier implementation, for example with the introduction of CityJSON. On the other hand, CityGML version 3.0 is about to be released (Kutzner, Chaturvedi, & Kolbe, 2020). This new version introduces many features such as a new LoD concept (based on Löwner, Gröger, Joachim Benner, Biljecki, & Nagel, 2016), extended version management system (introducing bitemporal time-stamps, see Chaturvedi, Smyth, Gesquiere, Kutzner, & Kolbe, 2017), specific classes to facilitate easier conversion to IFC, and the possibility of introducing logical spaces (as a complement to physical spaces). The latter can be used, for example, for 3D cadastre units, enabling a connection between CityGML with the Land Administration Domain Model (International Organization for Standardization (ISO, 2012), without introducing an ADE (Sun, Olsson, Eriksson, & Harrie, 2020). CityGML version 3.0 has interesting features for establishing new national standards (Eriksson et al., 2020), but it is uncertain if this new standard will be able to address the technical interoperability issues evaluated in this benchmark. And there is a risk that the new features (even though interesting from several application perspectives) may even increase the complexity as highlighted in this study and therefore pose further interoperability challenges.

ACKNOWLEDGMENTS

This work was possible thanks to the collaboration of the whole GeoBIM benchmark team (with their work as in-kind contribution to the project), all the data providers, the participants making the tests, listed in the GeoBIM benchmark website (https://3d.bk.tudelft.nl/projects/geobim-benchmark/participants.html) and the participants to the GeoBIM benchmark workshop (https://3d.bk.tudelft.nl/projects/geobim-benchmark/events.html). The benchmark was funded by ISPRS and EuroSDR. This project has also received funding from the European Research

Council (ERC) under the European Union's Horizon 2020 Research & Innovation Programme (grant agreement no. 677312," Urban Modelling in Higher Dimensions") and Marie Sklodowska-Curie (grant agreement no. 707404, "Multisource Spatial data Integration for Smart City Applications").

ORCID

Francesca Noardo https://orcid.org/0000-0003-2269-5336
Ken Arroyo Ohori https://orcid.org/0000-0002-9863-0152
Lars Harrie https://orcid.org/0000-0003-3252-1495
Cristina Leoni https://orcid.org/0000-0003-4906-5692
Stelios Vitalis https://orcid.org/0000-0003-1886-0722

ENDNOTES

[1] Some examples from social media include: https://twitter.com/jamesmfee/status/748270105319006208, https://twitter.com/hugoledoux/status/1226065325327822848, https://twitter.com/MikeTreglia/status/866114277261930496, https://twitter.com/OSMBuildings/status/582884586134343680, https://twitter.com/GmLfun/status/1093512032350097409, https://twitter.com/GmLfun/status/1109117945240788992, https://stackoverflow.com/search?q=citygml.

[2] This task is the subject of this article.

[3] https://3d.bk.tudelft.nl/projects/geobim-benchmark/software.html for the software tested and https://3d.bk.tudelft.nl/projects/geobim-benchmark/participants.html for the list of participants.

[4] For the conversion citygml-tools can be used: https://github.com/citygml4j/citygml-tools.

[5] An ***FME*** reader/writer for CityJSON has now been implemented (https://github.com/safesoftware/fme-CityJSON), so that it likely could also be used for importing CityJSON data. However, this functionality was not yet available during the data collection phase of the benchmark.

[6] In Noardo et al. (2020b), "QGIS test 3".

REFERENCES

Aleksandrov, M., Diakite, A., Yan, J., Li, W., & Zlatanova, S. (2019). Systems architecture for management of BIM, 3D GIS and sensors data. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, *4*(4/W9), 3–10.

Arroyo Ohori, K. (2020). azul: A fast and efficient 3D city model viewer for macOS. *Transactions in GIS*, *24*(5), 1165–1184.

Arroyo Ohori, K., Diakite, A., Krijnen, T., Ledoux, H., & Stoter, J. (2018). Processing BIM and GIS models in practice: Experiences and recommendations from a GeoBIM project in the Netherlands. *ISPRS International Journal of Geo-Information*, *7*(8), 311.

Azhar, S. (2011). Building information modeling (BIM): Trends, benefits, risks, and challenges for the AEC industry. *Leadership and Management in Engineering*, *11*(3), 241–252.

Bartie, P., Reitsma, F., Kingham, S., & Mills, S. (2010). Advancing visibility modelling algorithms for urban environments. *Computers, Environment & Urban Systems*, *34*(6), 518–531.

Biljecki, F., Kumar, K., & Nagel, C. (2018). CityGML Application Domain Extension (ADE): Overview of developments. *Open Geospatial Data, Software & Standards*, *3*, 13.

Biljecki, F., Ledoux, H., & Stoter, J. (2016). Generation of multi-LOD 3D city models in CityGML with the procedural modelling engine Random3Dcity. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, *4*(4/W1), 51–59.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., & Çoltekin, A. (2015). Applications of 3D city models: State-of-the-art review. *ISPRS International Journal of Geo-Information*, *4*(4), 2842–2889.

Chaturvedi, C., Smyth, C. S., Gesquiere, G., Kutzner, T., & Kolbe, T. H. (2017). Managing versions and history within semantic 3D city models for the next generation of CityGML. In A. Abdul-Rahman (Ed.), *Advances in 3D geoinformation* (Lecture Notes in Geoinformation and Cartography, pp. 191–206). Berlin, Germany: Springer.

Egusquiza, A., Izkara, J. L., & Gandini, A. (2018). Energy efficiency improvement in historic urban environments: From decision support systems to co-creation strategies. In *Proceedings of the Third International Conference on Energy Efficiency in Historic Buildings*, Visby, Sweden (pp. 576–584). Uppsala, Sweden: Uppsala University.

Eriksson, H., Johansson, T., Olsson, P.-O., Andersson, M., Engvall, J., Hast, I., & Harrie, L. (2020). Requirements, development, and evaluation of a national building standard: A Swedish case study. *ISPRS International Journal of Geo-Information*, *9*(2), 78.

Fosu, R., Suprabhas, K., Rathore, Z., & Cory, C. (2015). Integration of building information modeling (BIM) and geographic information systems (GIS): A literature review and future needs. In *Proceedings of the 32nd CIB W78 Conference*, Eindhoven, the Netherlands (pp. 27–29). https://itc.scix.net/pdfs/w78-2015-paper-020.pdf.

Haddock, A. (2018). *Building information modelling for asset and facilities management* (Unpublished MS thesis). Wellington, New Zealand: Victoria University of Wellington.

ISO. (2012). *Geographic information—Land administration domain model (ISO 19152)*. Geneva, Switzerland: International Organization for Standardization.

ISO. (2013). *Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries*. Geneva, Switzerland: International Organization for Standardization.

Jakubiec, J. A., & Reinhart, C. F. (2013). A method for predicting city-wide electricity gains from photovoltaic panels based on LiDAR and GIS data combined with hourly Daysim simulations. *Solar Energy*, *93*, 127–143.

Kang, T. W., & Hong, C. H. (2015). A study on software architecture for effective BIM/GIS-based facility management data integration. *Automation in Construction*, *54*, 25–38.

Kumar, K., Labetski, A., Arroyo Ohori, K., Ledoux, H., & Stoter, J. (2019). The LandInfra standard and its role in solving the BIM-GIS quagmire. *Open Geospatial Data, Software & Standards*, *4*(1), 1–16.

Kumar, K., Ledoux, H., Commandeur, T. J. F., & Stoter, J. E. (2017). Modelling urban noise in CityGML ADE: Case of the Netherlands. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, *4*(4/W5), 73–81.

Kutzner, T., Chaturvedi, K., & Kolbe, T. H. (2020). CityGML 3.0: New functions open up new applications. *Journal of Photogrammetry, Remote Sensing & Geoinformation Science*, *88*, 43–61.

Ledoux, H. (2013). On the validation of solids represented with the international standards for geographic information. *Computer-Aided Civil & Infrastructure Engineering*, *28*(9), 693–706.

Ledoux, H., Arroyo Ohori, K., Kumar, K., Dukai, B., Labetski, A., & Vitalis, S. (2019). CityJSON: A compact and easy-to-use encoding of the CityGML data model. *Open Geospatial Data, Software & Standards*, *4*, 4.

Liang, J., Gong, J., Li, W., & Ibrahim, A. N. (2014). A visualization-oriented 3D method for efficient computation of urban solar radiation based on 3D–2D surface mapping. *International Journal of Geographical Information Science*, *28*(4), 780–798.

Lim, J., Tauscher, H., & Biljecki, F. (2019). Graph transformation rules for IFC-to-CityGML attribute conversion. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, *4*(4/W8), 83–90.

Liu, X., Wang, X., Wright, G., Cheng, J. C.-P., Li, X., & Liu, R. (2017). A state-of-the-art review on the integration of Building Information Modeling (BIM) and Geographic Information System (GIS). *ISPRS International Journal of Geo-Information*, *6*(2), 53.

Löwner, M.-O., Gröger, G., Joachim Benner, J., Biljecki, F., & Nagel, C. (2016). Proposal for a new LOD and multi-representation concept for CityGML. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, *4*(2/W1), 3–12.

Nguyen, H. T., & Pearce, J. M. (2012). Incorporating shading losses in solar photovoltaic potential assessment at the municipal scale. *Solar Energy*, *86*(5), 1245–1260.

Niu, S., Yang, Y., Pan, W. (2019). Logistics planning and visualization of modular integrated construction projects based on BIM-GIS integration and vehicle routing algorithm. In *Proceedings of the 2019 Modular and Offsite Construction Summit*, Banff, Canada (pp. 579–586). Edmonton, Canada: University of Alberta.

Noardo, F., Arroyo Ohori, K., Biljecki, F., Ellul, C., Harrie, L., Krijnen, T., & Stoter, J. (2019). GeoBIM benchmark 2019: Design and initial results. *ISPRS International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, *42*(2/W13), 1339–1346.

Noardo, F., Arroyo Ohori, K., Biljecki, F., Ellul, C., Harrie, L., Krijnen, T., & Stoter, J. (2020a). *CityGML files exported within the GeoBIM benchmark 2019—Task 3*. Retrieved from http://dx.doi.org/info:doi/10.5281/zenodo.3966915.

Noardo, F., Arroyo Ohori, K., Biljecki, F., Ellul, C., Harrie, L., Krijnen, T., & Stoter, J. (2020b). *Reports about the software behaviour delivered within the GeoBIM benchmark 2019—Task 3*. Retrieved from http://dx.doi.org/info:doi/10.5281/zenodo.3966987.

Noardo, F., Biljecki, F., Agugiaro, G., Arroyo Ohori, K., Ellul, C., Harrie, L., & Stoter, J. (2019). GeoBIM benchmark 2019: Intermediate results. *ISPRS International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, *42*(4), 47–52.

Noardo, F., Ellul, C. C., Harrie, L., Overland, I., Shariat, M., Arroyo Ohori, K., & Stoter, J. (2020). Opportunities and challenges for GeoBIM in Europe: Developing a building permits use-case to raise awareness and examine technical interoperability challenges. *Journal of Spatial Science*, *65*(2), 209–233.

Noardo, F., Harrie, L., Arroyo Ohori, K., Biljecki, F., Ellul, C., Eriksson, H., … Stoter, J. (2020). Tools for BIM-GIS integration (IFC georeferencing and conversions): Results from the GeoBIM benchmark 2019. *ISPRS International Journal of Geo-Information*, *9*(9), 502.

Noardo, F., Krijnen, T., Arroyo Ohori, K., Biljecki, F., Ellul, C., Harrie, L., … Stoter, J. (2020). *Reference study of IFC software support: The GeoBIM benchmark 2019—Part I*. arXiv preprint, arXiv:2007.10951.

OGC. (2004). *OGC Geography Markup Language (GML) Implementation Specifications Version 3.1.1* (Document No. 03-105r1). Wayland, MA: Open Geospatial Consortium.

OGC. (2012). *OGC City Geography Markup Language (CityGML) Encoding Standard 2.0.0* (Document No. 12019). Wayland, MA: Open Geospatial Consortium.

Peters, R., Ledoux, H., & Biljecki, F. (2015). Visibility analysis in a point cloud based on the Medial Axis Transform. In *Proceedings of the Eurographics Workshop on Urban Data Modelling and Visualisation*, Delft, the Netherlands (pp. 7–12). Geneva, Switzerland: Eurographics.

Petri, I., Kubicki, S., Rezgui, Y., Guerriero, A., & Li, H. (2017). Optimizing energy efficiency in operating built environment assets through building information modeling: A case study. *Energies*, *10*(8), 1167.

Stouffs, R., Tauscher, H., & Biljecki, F. (2018). Achieving complete and near-lossless conversion from IFC to CityGML. *ISPRS International Journal of Geo-Information*, *7*(9), 355.

Sun, J., Mi, S., Olsson, P., Paulsson, J., & Harrie, L. (2019). Utilizing BIM and GIS for representation and visualization of 3D cadastre. *ISPRS International Journal of Geo-Information*, *8*(11), 503.

Sun, J., Olsson, P., Eriksson, H., & Harrie, L. (2020). Evaluating the geometric aspects of integrating BIM data into city models. *Journal of Spatial Science*, *65*(2), 235–255.

Yao, Z., Nagel, C., Kunde, F., Hudra, G., Willkomm, P., Donaubauer, A., Kolbe, T. H. (2018). 3DCityDB: A 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML. *Open Geospatial Data, Software & Standards*, *3*, 5.

How to cite this article: Noardo F, Arroyo Ohori K, Biljecki F, et al. Reference study of CityGML software support: The GeoBIM benchmark 2019—Part II. *Transactions in GIS*. 2020;00:1–27.
https://doi.org/10.1111/tgis.12710