TU Delft

Delft University of Technology

Modelling and managing massive 3D data of the built environment

Kavisha, K.

DOI
10.4233/uuid:47218911-c93d-4295-a3de-231d023c1743

Publication date
2020

Document Version
Final published version

Citation (APA)
Kavisha, K. (2020). *Modelling and managing massive 3D data of the built environment*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:47218911-c93d-4295-a3de-231d023c1743

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

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Modelling and managing massive 3D data of the built environment

Kavisha

Modelling and managing massive 3D data of the built environment

Kavisha

Modelling and managing massive 3D data of the built environment

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus Prof. dr. ir. T. H. J. J. van der Hagen
chair of the Board of Doctorates
to be defended publicly on Wednesday 14 October 2020 at 10.00 o'clock

by

Kavisha

Master of Technology in Remote Sensing and GIS, IIRS, Dehradun, India
born in Moradabad, UP, India.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus	chairperson
Prof. dr. J.E. Stoter	Delft University of Technology, promotor
Dr. H. Ledoux	Delft University of Technology, promotor

Independent members:

Prof. dr. P.W. Chan	Delft University of Technology
Prof. dr. V. Coors	Hochschule für Technik Stuttgart, Germany
Prof. dr. H.J. Scholten	Vrije Universiteit Amsterdam
Dr. C. Ellul	University College London, United Kingdom
Dr. E.J. Meijers	Delft University of Technology

This work is part of the research programme Maps4Society with project number 13740 which is partly financed by the Netherlands Organisation for Scientific Research (NWO), and partly by the Ministry of Economic Affairs.

Keywords: 3D city models, CityGML, ADE, LandInfra, metadata.
Printed by: printenbind.nl
Copyright ©2020 by Kavisha
ISBN: 978–94–6366–316–8

A digital version of this thesis is available at http://repository.tudelft.nl.

This thesis is released using the CC BY 4.0 license, for more information visit https://creativecommons.org/licenses/by/4.0/.

Contents

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisors Hugo Ledoux and Jantien Stoter. I was quite fortunate to have them as supervisors who guided my research in the right direction, and gave me a great degree of freedom to explore topics of interest to fulfil my scientific curiosity. Thank you for creating such a wonderful working environment.

I would like to thank the Netherlands Organisation for Scientific Research (NWO) for funding my PhD project, the NWO program directors Margriet Jansz and Paul Blank, and the members of the NWO user committee for their participation and valued input.

During the project I have closely collaborated with Anna Labetski, who is also a PhD candidate in the same group. Her inexhaustible enthusiasm is acknowledged with great pleasure. Special thanks to everyone who is or used to be part of the 3D geoinformation group: Abdoulaye Diakité, Balázs Dukai, Clara García-Sánchez, Filip Biljecki, Francesca Noardo, Giorgio Agugiaro, Ken Arroyo Ohori, Liangliang Nan, Ravi Peters, Stelios Vitalis, Tom Commandeur, and Weixiao Gao. I really enjoyed being a part of the group. I am also grateful to Danielle Karakuza, Karin Visser, Margo van der Helm, and Martine de Jong-Lansbergen for their administrative and organisational support. I would also like to thank Richard Schmidt (DGMR) and Theo Verheij (DGMR) for their inputs and fruitful discussions.

I also would like to thank Ankit for his moral support and help in designing the cover of this thesis. Last but not least, I would like to express my gratitude to my parents and my brother for their continued love and support during my PhD research.

Kavisha *Eindhoven, August 2020*

Part I

Introduction, research questions, and scope

Chapter 1

Introduction

1.1 Motivation

Cities are progressively embracing 3D city models. Several major cities like Rotterdam, Brussels, Singapore, New York, and Berlin, to name a few, have already created 3D city models [241], and the main goal is to use them for different urban applications [19]. 3D city models offer additional insights when using conventional 2D maps. For instance, calculating the noise levels is more accurate when done using 3D city models rather than 2D maps. The variations in noise levels at different heights and due to the presence of other features like buildings and noise barriers in the path of noise propagation cannot be properly modelled using 2D maps [135].

Practitioners engaged in the development and utilisation of 3D city models of large cities often encounter several issues and limitations. One such issue is that these 3D city models get massive in size, especially when semantics and other properties need to be stored with the geometry and topology of different city objects in the model. When the size of semantic 3D city models exceeds the terabyte mark, difficulties arise in storing, managing, and using these models efficiently. One such example is the 3DTOP10NL[1] [117], the 3D city model of the Netherlands. It covers the whole country including buildings, terrain, roads, canals, etc. as one triangulation with more than one billion triangles (Figure 1.1). It takes more than 700 GB of storage space just to store the geometry of the 3DTOP10NL terrains (without any topological information and semantic attributes) in files according to the international 3D city modelling standard CityGML [136].

[1] The 3DTOP10NL dataset was launched just before the start of my PhD research in 2015. It is the dataset of interest in 3D4EM project.

I investigate in this thesis how to better model such massive and semantically enriched 3D city models to efficiently store, manage, and use them in different applications.

Currently, the 3D city models are developed by different organisations and are available in different standards and formats. A number of standards exists for representing 3D data, such as IFC (Industry Foundation Classes) [114] in BIM (Building Information Modelling) domain; CityGML in 3D GIS (Geographical information Systems); COLLADA[2] (COLLAborative Design Activity), glTF[3] (GL Transmission Format) in computer graphics, and so on. One would need to convert between different standards and formats to use these 3D city models in different applications. Practitioners often struggle while shuttling these massive semantic 3D datasets back and forth between different standards and formats, which often results in loss of information [258].

Much work has been done for developing frameworks for converting 3D datasets between different standards, particularly IFC and CityGML [185], the two popular open standards in the BIM and 3D GIS domains [3, 52, 63, 70, 72, 98, 175]. In addition, new standards have been recently developed that integrate concepts from different standards to represent an integrated semantic 3D city model. One such standard is the OGC LandInfra (and InfraGML) [188] which integrates concepts from IFC and CityGML. Despite several attempts, these new standards remain disconnected, owing primarily to low adoption in the 3D community and lack of software support [138, 139]. There is a lack of research into the development of these new open standards for interoperable semantic 3D city modelling. A standard that has been tested and implemented in software is an important step of the standardisation process, which increases the usability of the standard.

I focus in this thesis on different open standards for 3D city models, including CityGML, IFC, and LandInfra/InfraGML. I investigate how the development of new multi-disciplinary standards such as the LandInfra/InfraGML can contribute to the convergence of interoperability issues between different 3D domains, specifically BIM and GIS. Further, I discuss various applications where in the 3D city models modelled using different standards can be used.

The 3D city models can also be enriched with application specific information such as attributes related to noise mapping, flood modelling, energy simulations, and so on. The use of such 3D city models can greatly improve environmental analysis in 3D. However, there is a dearth of such enriched 3D city models for use in applications. The application of 3D city models which is covered in this thesis is noise simulation. Monitoring and mapping of noise is an active area of research which is drawing substantial public attention [135]. I discuss the

[2] https://www.khronos.org/collada/
[3] https://www.khronos.org/gltf/

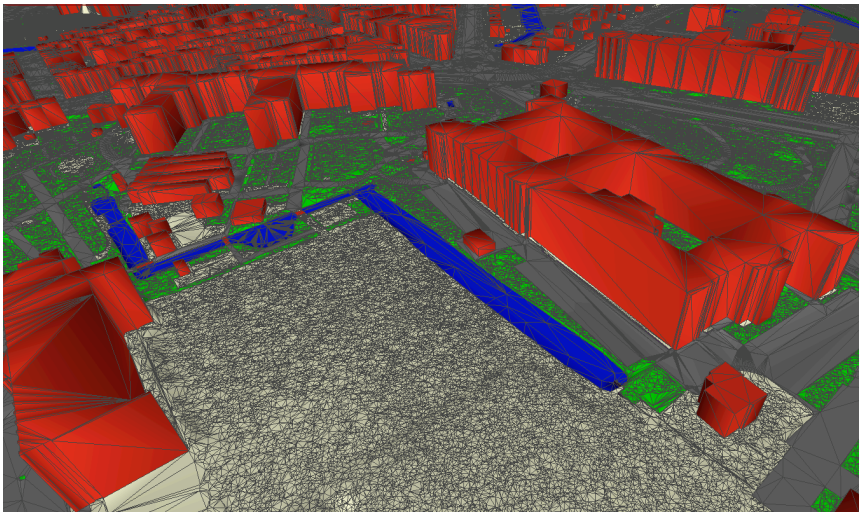


Figure 1.1: Snapshot of 3DTOP10NL dataset of a part of Delft, the Netherlands. Note that the terrain is one massive TIN with buildings, roads, water bodies, and other features

need to have a harmonised semantic data model to represent 3D city models for urban noise simulations. The goal is to standardise the input and output data for noise simulation to be able to compare the outcomes of different noise studies and to assure that different noise studies using the same simulation method yield the same results. Further more, storing metadata related to the 3D city models can play important role in *3D data discovery* i.e. with the help of metadata, a user can find relevant 3D datasets for a specific application [143]. However, the specifications to model metadata related to 3D city models in a structured way are missing in CityGML. I examine in this thesis, the metadata needs specific to 3D geospatial datasets and present an ISO 19115 compliant solution to add metadata to the 3D city models.

1.2 Problem description

During the course of my research [120, 132, 134, 136, 137, 138, 139, 143] in the field of 3D city modelling, I have identified three problems which are often faced by the practitioners using 3D datasets.

First Problem: *There is a lack of robust solution to efficiently store, manage massive terrains in the context of 3D city models.*

In practice, the applications of 3D city models are mostly centred around buildings; other city features, such as terrain, vegetation, roads, and water bodies are often ignored. Furthermore, the formal specifications for modelling buildings in 3D space are often more prominently defined than other city features. I investigate in this thesis the terrain in a 3D city model, and particularly its storage and management as TINs (Triangulated Irregular Networks), which, apart from grids, are more used in practice. Terrains often require a massive amount of storage space for the geometry, topology, and other associated attributes (if any), since they are often created from the LiDAR datasets.

The international 3D GIS standard CityGML supports the storage of terrains as TINs but I show in this thesis that it is not efficient for storing and managing massive TINs. The CityGML datasets can become very large for massive TINs because of the redundancy in the underlying data structure, which greatly hinders their use in applications, web rendering, and exchange of data [136]. Moreover, topology information can only be stored limitedly in the underlying data structure for storing TINs in CityGML, which prevents us from efficiently using the datasets for analysis. This problem of storing and managing massive TIN terrain models persists also in the database implementation of the CityGML, i.e. the 3D City Database (3DCityDB). I present in this thesis my implementation for a robust solution to compress massive TIN terrains in flat file systems and databases.

Furthermore, CityGML has the concept of LODs (0-4) wherein features become much more detailed in their geometry and semantic differentiation with each increasing LOD [185]. The concept is very well established for buildings and bridges, but is vague in case of terrains and land use. The CityGML specifications do not distinguish between different terrain LODs at geometrical and semantic level although its possible to model different levels of terrain [158]. I present in this thesis a proposal for modelling terrains at different LODs in CityGML.

Second Problem: *There is a lack of interoperable standards for semantic 3D city modelling.*

A plethora of standards exists for representing 3D city models, e.g. 3D graphical standards such as OBJ, glTF; 3D GIS standards such as CityGML; BIM standards such as IFC, gbXML (Green Building XML); and so on. These standards are developed by different organizations and differ in modelling approach used for geometry, semantics, underlying schema, etc. Problems like loss of information, improper conversion, loss of relationships, topological inconsistencies arise while converting and combining 3D city models from different formats [258]. This complicates the exchange and use of 3D city models across different applications, thereby making interoperability a crucial issue.

The BIM and 3D GIS domains are often faced with the data interoperability issues when converting 3D city models between the IFC and the CityGML standard. Recent steps taken by the standardisation organisations to address these issues include the development of the multi-disciplinary open standards integrating concepts from different domains, e.g. the open LandInfra standard by the OGC to connect BIM and 3D GIS. The LandInfra standard has substantial overlaps with CityGML and IFC [188]. Although it has the potential to bring the BIM and GIS views onto a common footing, the standard is not well known in the BIM or GIS communities. Furthermore, LandInfra has no software support yet and is barely used in practice. I investigate in this thesis the harmonisation of already existing (i.e. CityGML and IFC) and newly developed open standards (i.e. LandIfra/InfraGML) for the representation and utilisation of semantics 3D city models. I discuss how these new open standards can contribute to the convergence of interoperability issues between different 3D domains, specifically BIM and GIS, and how that helps for different applications in practice.

Third Problem: *There is a lack of standardised semantic data models to discover 3D city models for use in different applications related to the built environment.*

The use of 3D city models for urban applications has tremendously increased in the past years. Since 3D city models are widely being created and used, storing metadata related to the 3D city models can be useful for *3D data discovery* i.e. the users can discover (on the web or in a portal) relevant 3D datasets for a specific application [143]. Metadata can play a key role in the management, retrieval, and dissemination of these massive models [143]. Metadata can ensure that data creators and data users from different 3D domains can understand and communicate about data requirements and its usability [164]. Further, it can be useful for *3D data discovery* i.e. the users can discover (on the web or in a portal) relevant 3D datasets for a specific application. While there exist international standards for geospatial metadata (ISO 19115 [108]), these are rarely used in practice for 3D data. Furthermore, CityGML does not offer a mechanism to store metadata related to 3D city models in a structured way. I examine in this thesis, the metadata needs specific to 3D geospatial datasets and present an ISO 19115 compliant solution to add metadata to the 3D city models. Having metadata in CityGML files, which are in practice often very large and complex, would provide us with the ability to quickly understand the relevance and the nature of a dataset i.e. geometry, semantics, LODs, etc. [143].

With the possibility to enrich 3D city models with application specific information, these city models are being used as input/output models for environmental analysis in 3D. For instance, urban environmental applications such as noise simulation require a multitude of data coming from different sources, such as data related to the source of noise (e.g. number of vehicles, speed of vehicles, operating hours of industrial machineries, etc.), obstructions in the path of noise

(e.g. height of noise barriers and buildings), geographic locations of people (e.g. buildings), etc. [51, 135]. The 3D city models supplemented with such extensive information can aid noise simulation. However, different countries have developed their own methods for estimating noise, and the utilisation of this input data can differ based on the method used to assess the noise levels. Furthermore, the results are affected by the completeness, accuracy, and reliability of the spatial data used in the simulation. The heterogeneity in these methods and utilised input data makes it difficult to obtain comparable results which is important for strategic noise mapping [173, 180]. I present in this thesis my approach for the development of a semantic data model to represent 3D city models for use in urban noise simulations.

1.3 Research questions

The main research question of this thesis is:

> *How to better model massive and semantically rich 3D city models, coming from heterogeneous data sources, for re-use in different built environment applications?*

Based on the identified problems in Section 1.2, the sub research questions can be formulated as:

1. How to efficiently store and manage massive terrains in the context of 3D city models in flat files and database systems? *(Problem 1)*

2. How can we model a terrain at different levels of detail in a 3D city model? *(Problem 1)*

3. How can the development of new multi-disciplinary standards such as LandInfra and InfraGML contribute to the re-use of data amongst different application domains such as BIM and 3D GIS? *(Problem 2)*

4. How to harmonise the already existing (i.e. CityGML/IFC) and the newly developed open standards (i.e. LandIfra/InfraGML) for semantically rich 3D city models? *(Problem 2)*

5. How to effectively model the metadata associated with 3D city models for 3D data discovery and applications? *(Problem 3)*

6. How to develop a semantically enriched data model for integrating data from different sources for urban noise simulation? *(Problem 3)*

1.4 Thesis outline

1.4.1 Thesis outline

This thesis is based on different scientific articles that I have published during the course of my PhD research. Some parts have been updated since their original publication to reflect the updates in the methodology and results. The thesis is organised into 3 parts and 10 chapters, as follows:

Part I: Introduction, research questions, and scope

In Chapter 1, I discuss the motivation behind this research, the problems and the research questions addressed in this thesis. In Chapter 2, I provide an overview of the state of art in 3D city modelling standards for the built environment.

Part II: Data modelling and management of massive terrains

The second part of this thesis focuses on the storage and management of massive terrains in the context of 3D city models. In Chapter 3, I present my improved representation for compactly storing massive terrains as TINs. I review the different data structures for compactly representing massive terrains as TINs and explore how they can be used integrated in flat file systems. Chapter 4 presents the DBMS implementation of the selected data structures for efficient storage and management of TIN terrains in a database. In Chapter 5, I present a proposal for modelling terrains at different levels of detail in a 3D city model.

Part III: Harmonising 3D standards for the built environment

The third part of the thesis focuses on the different open standards for representing semantic 3D city models. Chapter 6 provides a detailed comparative analysis of the LandInfra/InfraGML standards with CityGML and IFC, and explores how it can contribute to the built environment applications. Chapter 7 describes the harmonised mapping between the LandInfra and the CityGML standard. This chapter also gives an overview of a few use cases where this integration can be useful for urban applications.

Part IV: 3D data modelling and discovery for the built environment applications

The fourth part of the thesis focuses on 3D data modelling and discovery for applications. In Chapter 8, I investigate how metadata can help in 3D data discovery for applications, and present an ISO 19115 compliant solution to add metadata to the 3D city models. Chapter 9 focuses on implementing a semantically enriched data model for urban noise simulation for the European Union.

Chapter 10 concludes the thesis with the key takeaways, answers to the research questions, main contributions of the research, and a roadmap for future work.

1.4.2 Open science

The main software that I have developed in this research are released as open-source. Moreover, the generated datasets have also been publicly released as open data. All the publications that form the basis of this thesis are available as open access.

1.4.3 Personal pronouns

In this thesis, I use 'we' as a courtesy to my co-authors in the publications. I use 'I' in parts exclusive to this thesis, such as answering the research questions.

Further, I do not officially have a last name. In my publications, I use the last name *'Kumar'*.

1.4.4 About this thesis

The work described in this thesis represents the results of the research I carried out at the 3D Geoinformation group, Delft University of Technology, the Netherlands and is based on the following journal papers:

1. **A Harmonized Data Model for Noise Simulation in the EU**. Kavisha Kumar, Hugo Ledoux, Richard Schmidt, Theo Verheij, and Jantien Stoter. *ISPRS International Journal of Geo-Information* 9(2), 2020, p.121, doi: https://doi.org/10.3390/ijgi9020121

2. **The LandInfra standard and its role in solving the BIM-GIS quagmire**. Kavisha Kumar, Anna Labetski, Ken Arroyo Ohori, Hugo Ledoux, and Jantien Stoter. *Open Geospatial Data, Software and Standards* 4(1), 2019, p.1 , doi: https://doi.org/10.1186/s40965-019-0065-z

3. **Harmonising the OGC Standards for the Built Environment: A CityGML Extension for LandInfra**. Kavisha Kumar, Anna Labetski, Ken Arroyo Ohori, Hugo Ledoux, and Jantien Stoter. *ISPRS International Journal of Geo-Information* 8(6), 2019, p.246, doi: https://doi.org/10.3390/ijgi8060246

4. **A metadata ADE for CityGML**. Anna Labetski, Kavisha Kumar, Hugo Ledoux, and Jantien Stoter. *Open Geospatial Data, Software and Standards* 3(1), 2019, p.16, doi: https://doi.org/10.1186/s40965-018-0057-4

5. **Compact representation of massive terrains represented as TINs in 3D city models**. Kavisha Kumar, Hugo Ledoux, and Jantien Stoter. *Transaction in GIS* 22(5), 2018, p.1152, doi: https://doi.org/10.1111/tgis.12456
6. **CityGML Application Domain Extension (ADE): overview of developments**. Filip Biljecki, Kavisha Kumar, and Claus Nagel. *Open Geospatial Data, Software and Standards* 3(13), 2018, doi: https://doi.org/10.1186/s40965-018-0055-6

Chapter 6, 7, and 8 of this thesis are based on the journal papers 2, 3, and 4 repectively, written in collaboration with Anna Labetski, PhD candidate in the 3D Geoinformation group. Chapter 5 is based on the following conference paper written also in collaboration with Anna Labetski.

1. **An improved LOD framework for the terrains in 3D city models**. Kavisha Kumar, Anna Labetski, Hugo Ledoux, and Jantien Stoter. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* IV-4/W8, 2019, p.75, doi: https://doi.org/10.5194/isprs-annals-IV-4-W8-75-2019

1.4.5 Project

The research described in this thesis is part of the project '3D for Environmental Modelling (3D4EM)' with project number 13740 in the Maps4Society programme. The project is partly funded by the Netherlands Organisation for Scientific Research (NWO), and partly by the Ministry of Economic Affairs. The aim of this research project is to develop a nationwide 3D information infrastructure (an SDI (Spatial Data Infrastructure)) for integrated 3D environmental modelling. The project was carried out in collaboration with 14 partner organisations (universities, government and companies). Furthermore, the project was closely associated with the activities of the Open Geospatial Consortium (OGC).

Chapter 2

State of art in 3D standards for built environment modelling

This chapter describes the state of art in different 3D standards covered in this thesis for modelling built environment information.

2.1 CityGML

CityGML is an open 3D standard for the representation, storage, and exchange of 3D city models [185]. It models the geometry, semantics, and graphical appearance associated with 3D city models. It is implemented as an application schema of the GML3 (Geography Markup Language version 3) [185] The representation of 3D geometries in CityGML is in accordance with the ISO 19107 model [100].

The data model of CityGML comprises of a core module and several thematic modules such as Building, Relief, Bridge, Transportation, Vegetation, and WaterBody. The core module defines the abstract base classes from which thematic classes are derived, e.g. the abstract `_CityObject` class in the core module is the base class of all the thematic classes in CityGML. It also defines non-abstract data, such as basic data types and attributes which can be used by the thematic classes e.g. `creationDate`, `demolitionDate`. The thematic model of CityGML provides different thematic classes to store city objects, such as buildings, water, terrain, roads, vegetation, bridges, tunnels and their associated semantic properties. For instance, in case of buildings, it is possible to store building properties, such as year of construction `(yearOfConstruction)`, year of demolition `(yearOfDemolition)`, type of roof `(roofType)`, etc. as attributes. Furthermore, CityGML supports hierarchical decomposition of an object into semantic surfaces depending upon the required LOD, e.g. a building in LOD2 can be differentiated into walls `(WallSurface)`, roofs `(RoofSurface)` and ground surface

(GroundSurface). This is useful for many applications such as estimation of urban energy [1] or noise levels on building facades [135]. Further, objects which are not explicitly modelled in the thematic model of CityGML such as pipes and road noise barriers can be stored by extending the data model using either generic city objects/attributes or ADEs (Application Domain Extensions) (see Section 2.1.1[1]).

2.1.1 CityGML extension modelling

Depending upon the application requirements, users may want to model objects and attributes of 3D city models which are not covered in the data model of CityGML. An example of such a case is the application of 3D city models in the energy domain: a method employed to estimate the energy demand of a building may necessitate specific and less common information such as building occupancy [1], an attribute that CityGML does not envisage by default [25]. The developers of CityGML have foreseen situations such as this one, and thus for this purpose two ways to support augmenting the CityGML data model beyond its initial scope are offered: through (i) generic objects and attributes (*Generics* module), and (ii) using the ADE mechanism [25].

Generics

Generics is a semi-structured extension mechanism where the city objects are extended with additional objects and attributes without making any changes in the CityGML schema. An example from practice is the 3D city model of the Hague in the Netherlands[2]. Each building contains the attribute of the height of its eaves, expressed as a generic attribute, for example:

```
<gen:doubleAttribute name="RelativeEavesHeight">
  <gen:value>5.162</gen:value>
</gen:doubleAttribute>
```

The CityGML 2.0 Generics also has a `GenericAttributeSet` element which supports grouping of related generic attributes `(_genericAttribute)` under a common name. A `GenericAttributeSet` can contain any number of arbitrary generic attributes `(_genericAttribute)` and its name serves as an identifier for the entire set. It also has an optional `codeSpace` attribute to specify the authority, e.g. the organisation or community that defined a generic attribute set and its contained attributes.

[1] The section 2.1.1 is based on our paper: Biljecki F, Kumar K, and Nagel C. CityGML Application Domain Extension (ADE): overview of developments. *Open Geospatial Data, Software and Standards*, 2018. doi: https://doi.org/10.1186/s40965-018-0055-6

[2] https://data.overheid.nl/data/dataset/3d-stadsmodel-den-haag-2018

```
<gen:genericAttributeSet name="BaseHeights"
    codeSpace="https://www.example_authority.com">
   <gen:doubleAttribute name="RelativeEavesHeight">
       <gen:value>6.609</gen:value>
   </gen:doubleAttribute>
   <gen:doubleAttribute name="AbsoluteEavesHeight">
       <gen:value>13.189</gen:value>
   </gen:doubleAttribute>
   <gen:doubleAttribute name="RelativeRidgeHeight">
       <gen:value>9.587</gen:value>
   </gen:doubleAttribute>
   <gen:doubleAttribute name="AbsoluteRidgeHeight">
       <gen:value>16.167</gen:value>
   </gen:doubleAttribute>
</gen:genericAttributeSet>
```

Using Generics has certain limitations. For instance, CityGML datasets with generic objects and/or attributes cannot be validated against the schema because their names and data types are not formally defined in the schema. Moreover, name conflicts of the generic attributes and objects may occur. Consequently, using Generics has very limited semantic and syntactic interoperability.

ADEs

ADE is a structured mechanism for enriching the data model with new feature classes and attributes, while preserving the semantic structure of CityGML, and it is an important component of the standard since its early versions. Its main purpose is supporting additional requirements by certain use cases, which is accomplished by specifying extensions to the data model.

While Generics are created at runtime without introducing any changes in the CityGML schema, an ADE is formally specified in a separate XSD (XML Schema Definition) file and has its own namespace [185]. ADEs are actively used by information communities to create application specific extensions, such as the Energy ADE for energy modelling [181], the GeoBIM ADE for BIM-IFC integration with CityGML [52], the IMGeo ADE for modelling Dutch topographic data in CityGML [30], the Noise ADE for noise modelling [185]; see Biljecki et al. [25] for a detailed overview of existing ADEs.

The advantage of using ADEs is that the extensions are formally specified which ensures semantic and syntactic interoperability for the exchange of application specific information. The extended CityGML instances can be validated. Additionally, it is possible to use more than one ADE in the same dataset. ADEs do not need a formal approval by any standardisation body and can be developed by anyone.

ADEs can be modelled directly in the XML schema or can be generated by extending the UML (Unified Modelling Language) model of CityGML with application specific information and later deriving the XML schema from it. However, the CityGML documentation describes only the former approach. Afterwards, a document explaining the development of an ADE through a UML was released: the OGC best practice 'Modeling an application domain extension of CityGML in UML' [205]. [30] describe six different ways to create a CityGML ADE using UML. I provide here a brief overview of the most preferred ways of modelling an ADE in XSD and UML.

Using hooks: Every CityGML feature class has a GML 'hook' of the form `'_GenericApplication PropertyOf<Featuretypename>'` in its XML schema definition which allows to attach additional attributes to it by ADEs [185]. For example, `bldg:_GenericApplicationPropertyOfBuilding` can be used to attach new attributes (e.g. numberOfInhabitants) to the existing Building class:

```
<element name="numberOfInhabitants"type="xsd:positiveInteger"
    substitutionGroup="bldg:_GenericApplicationropertyOfAbstractBuilding"/>
```

Using subclasses: Another favoured approach for modelling an ADE is by extending the CityGML model with new classes for new feature types as subclasses of the existing CityGML classes. In this case, the hook mechanism is not used because a new class is added and not only properties to an existing CityGML class. The subclasses are also marked using the same stereotype `(«featureType»)` as the CityGML classes and are not suppressed in the XML Schema. As an example, Figure 2.1 depicts the extension of CityGML Building class with a new feature type `Garage` as a subclass with `«featureType»` stereotype. The new class `Garage` also has a new attribute `numberOfCarsParked`.

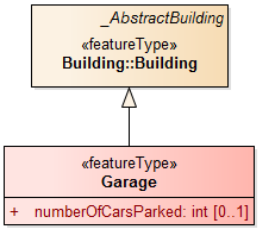


Figure 2.1: Example UML model depicting the extension of CityGML `Building` class with a new feature type `Garage` as a subclass with `«featureType»` stereotype. The new class `Garage` also has a new attribute `numberOfCarsParked`

The XML schema for the ADE can be generated from the UML model using the ShapeChange[3] tool. ShapeChange is a JAVA based tool which implements UML to GML encoding rules described in ISO 19136 [101], ISO 19118 [103], and ISO 19109 [111]. The XML schema is generated only for the ADE classes and attributes and not for the entire CityGML data model because it is already publicly available. The generated CityGML ADE schema only needs to correctly import the CityGML schema (see snippet below).

This approach has been accepted as the best practice by the OGC [205]. Another factor supporting this approach is that the concept of subclasses and inheritance is easy to understand with basic knowledge of UML.

```
....
....
<import namespace="http://www.opengis.net/citygml/2.0"
    schemaLocation="http://schemas.opengis.net/
   citygml/2.0/cityGMLBase.xsd"/>
<import namespace="http://www.opengis.net/citygml/building/2.0"
    schemaLocation="http://schemas.opengis.net/
   citygml/building/2.0/building.xsd"/>
....
....
<element name="Garage" type="GarageType"
    substitutionGroup="bldg:_AbstractBuilding"/>
<complexType name="GarageType">
  <complexContent>
    <extension base="bldg:BuildingType">
      <sequence>
        <element name="numberOfCarsParked" type="xs:integer"
          minOccurs="0" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

2.2 LandInfra and InfraGML

LandInfra [188] was proposed as the successor to LandXML [145]. LandXML is an XML based, open data model for the representing civil engineering and survey measurement data [145]. It is not recognised as an official standard by any standards organization like OGC or ISO, which created a confusion in the marketplace concerning the future of the standard. To align LandXML with the OGC standards, a LandGML Interoperability Experiment [183] was initiated by the OGC in 2004 to make LandXML compliant with the OGC GML standard for geospatial data. Following this effort in 2013, LandInfra SWG (Standards Working Group) reviewed LandXML and made efforts to determine how to continue

[3] http://shapechange.net

its support to the existing users in the best possible manner. Several problems with the LandXML-1.2 were discovered and likewise documented in Scarponcini [223, 224]. Further, there is no formally published documentation, user guide, requirements definition, or underlying conceptual model of LandXML. Therefore, a fresh OGC standard LandInfra was developed, based on a subset of LandXML functionality, but implemented with GML (as InfraGML) and supported by a UML conceptual model.

LandInfra covers both topography and subsurface information and partners the needs of surveying to locate infrastructure facilities on the terrain in compliance with interests in land [188]. It thus includes land and civil engineering infrastructure facilities, e.g. roads, buildings, railways, projects, alignment, survey, and land features; as well as the division of land based on administration, i.e. jurisdictions and districts; and interests in land, e.g. land parcels, easements and condominiums.

InfraGML is the GML based encoding of the LandInfra data model, which is published as an 8 part OGC standard: LandInfra Core (Part 0) [191], Land Features (Part 1) [192], Facilities and Projects (Part 2) [193], Alignments (Part 3) [194], Roads (Part 4) [195], Railways (Part 5) [196], Survey (Part 6) [197], and Land Division (Part 7) [198]. Each part has a separate schema (XSD file).

LandInfra has 10 requirements classes (Figure 2.2), of which `LandInfra` is the only mandatory one and is implemented in InfraGML standard Part 0 (summarised in Table 2.1). LandInfra allows land features to be collected into a `Facility`. Facilities include collections of buildings and civil engineering works but only provides general support for the facilities themselves and allows subsequent requirements classes to focus on specific facilities. Projects are activities that are related to the improvement of a facility, this includes design and/or construction. LandInfra requirement classes `Facility` and `Project` are implemented in InfraGML standard Part 2. An `Alignment` is a positioning element providing a linear referencing system necessary for locating elements and is implemented in InfraGML standard Part 3. The LandInfra requirement class `Survey` supports information in relation to survey work such as equipments used for survey, survey results, etc. and is implemented in InfraGML standard Part 6 (Survey). LandInfra `Road`, and `Railway` provide support for modelling roads and railways within the facilities and are implemented in InfraGML standard Part 4 and Part 5, respectively. `LandFeature` focuses on features of a land and specifically naturally occurring water and vegetation features while `LandDivision` models the division of land either public or private. `LandFeature` and `LandDivision` are implemented in InfraGML standard Part 1 and Part 7, respectively. LandInfra requirement class `Condominium` deals with the ownership of private and public units in a multi-unit building. It is also implemented in InfraGML standard Part 7.

Table 2.1: Main LandInfra requirements classes

#	Class	Summary	InfraGML part
1	`LandInfra`	Mandatory core with dataset information and common types	0
2	`Facility`	Collection of buildings, civil engineering works and their siteworks	2
3	`Project`	Activity related to the improvement of a facility	2
4	`Alignment`	Positioning element for locating physical elements	3
5	`Road`	Roads with 3D elements	4
6	`Railway`	3D railway elements and track geometry	5
7	`Survey`	Information related to surveys, e.g. equipment, results, etc.	6
8	`LandFeature`	Whether natural or man-made, in the surface or subsurface	1
9	`LandDivision`	Public (political, judicial, or executive) or private land divisions	7
10	`Condominium`	Ownership of private and public units in a multi-unit building	7

The development of LandInfra and InfraGML is an important milestone in the direction of open standards for the integration of geospatial information and the information about the built environment. Since it is based on the functionality of LandXML, LandInfra can easily substitute LandXML in the surveying, roads, and highway transportation sector. LandInfra can also be used in the AEC industry for urban facility management and life cycle maintenance of facilities and projects. Further, integration of LandInfra with other OGC standards, such as CityGML, can be useful for different urban applications such as estimating the level of noise exposure on buildings, or how much solar irradiation a building will receive. Unlike CityGML, LandInfra explicitly models the materials of road surfaces and terrain, geometry and semantics of railways, type of road elements (pavements, hard shoulders, soft shoulders, etc.), construction materials of buildings, and information about the observation/measurement points, to name a few. Such information is useful for environmental applications such as urban noise and flood mapping.

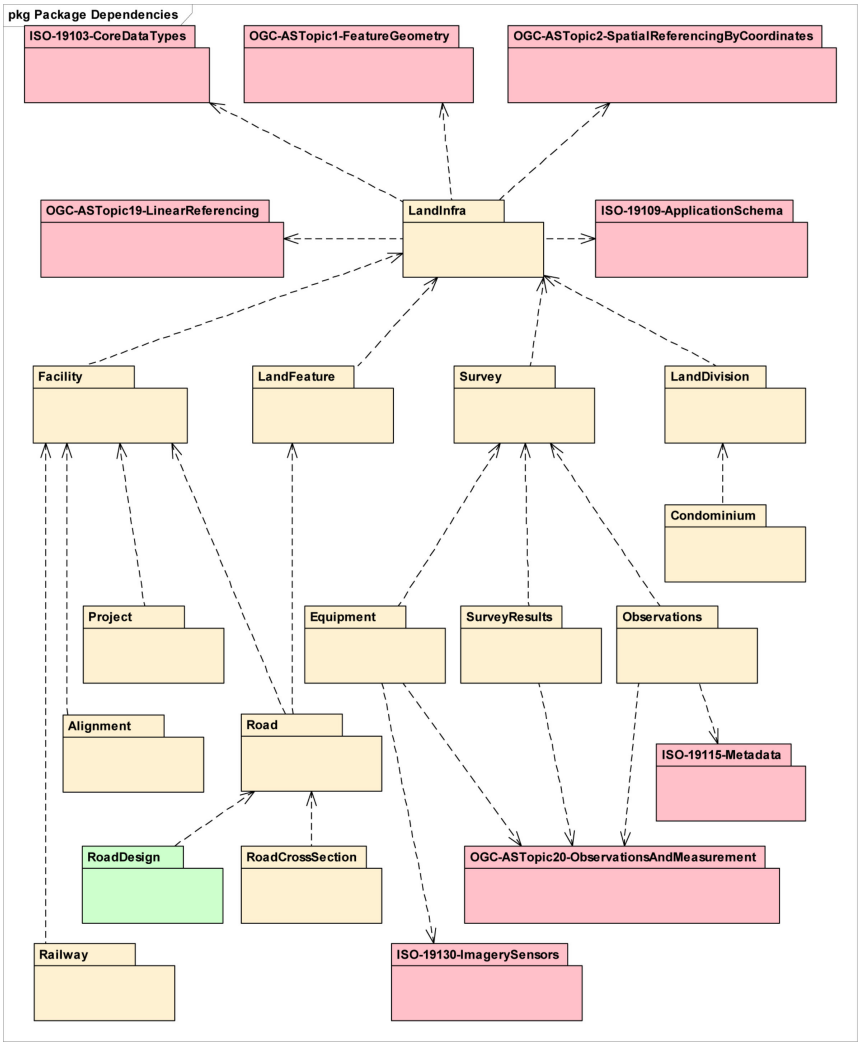


Figure 2.2: LandInfra Requirements Classes (Source: OGC [188]

2.3 IFC

IFC [113] is an open, international standard used in the BIM domain for the exchange of 3D models of buildings and infrastructure projects, such as bridges and viaducts. The standard is developed by the buildingSMART consortium, which comprises software and construction companies, transportation network operators and government agencies[4].

IFC files can contain many types of classes (130 defined types, 207 enumeration types, 60 select types and 776 entities in IFC 4 Addendum 2 [34]. Among others, there are classes to model the semantics of products (which include building elements such as `IfcDoor` or `IfcColumn`), organisations, rules, processes, and resources. For the purposes of this research, the most interesting ones are products, which include the definition of locations, such as building sites (`IfcSite`) and spaces (`IfcSpace`), and also the physical elements in buildings (e.g. `IfcBeam`, `IfcColumn`, `IfcDoor`, `IfcWindow`) and can have their own materials.

The geometry of physical elements can be created using a variety of representation paradigms, which include:

- **Primitive instancing:** an object is represented based on a set number of predefined parameters. IFC uses this paradigm to define various forms of 2D profiles, as well as volumetric objects such as spheres, cones and pyramids.
- **CSG and Boolean operations:** an object is represented as a tree of Boolean set operations (union, intersection and difference) of volumetric objects. Half-spaces are often used to cut out the undesired parts of surfaces or volumes.
- **Sweep volumes:** a solid can also be defined by a 2D profile (a circle, a rectangle or an arbitrary polygon with or without holes) and a curve along which the surface is extruded.
- **B-rep:** an object is represented by its bounding surfaces, either triangulated meshes, polygonal meshes or topological arrangements of free-form surfaces.

These paradigms can be used independently or combined with each other in a hierarchy. For this, elements are modelled in local coordinate systems defined by a hierarchical set of transformations, which correspond to the levels in a decomposition structure (typically a site, project, building and individual floors).

[4] https://www.buildingsmart.org/members/member-directory/

Part II

Data modelling and management of massive terrains

Chapter 3

Compact representation of massive TIN terrains in 3D city models

This chapter is based on the paper:

Kumar K, Ledoux H, Stoter J. 2018. Compactly representing massive terrain models as TINs in CityGML. *Transaction in GIS*, 22(5), pp.1152-1178. doi: https://doi.org/10.1111/tgis.12456

3.1 Introduction

The use of 3D city models for urban planning and management has increased in recent years. However, in practice, their applications are mostly centred around buildings; other features like vegetation, roads, terrain, and water bodies are often ignored. Furthermore, the formal specifications for modelling buildings in 3D space are often more prominently defined than other urban features. For example, in the international 3D GIS standard CityGML, the concept of LODs is very well established for buildings and bridges, but is vague in case of terrains and landuse [185]. In this chapter, I focus on the representation of a terrain, and particularly on its representation as a TIN (Triangulated Irregular Network), which, apart from grids, is often being used in practice. It should be emphasised that storing TINs is more complicated than storing grids because we need to not only store the geometry of the TIN but also efficiently storing the topological relationships between the triangles. Terrains often require a massive amount of storage space for the geometry, topology, and other associated attributes (if any), since they are often created from the LiDAR datasets.

While CityGML supports the storage of terrain models as TINs, I argue in this chapter that it is not efficient for storing massive TIN terrains. As shown in Section 3.5, the CityGML datasets can become very large for massive TINs because of the redundancy in the underlying data structure, which greatly hinders web-based rendering, exchange, and use of data in applications. Moreover, there is very little topological information stored, which greatly hinders their use in applications, web rendering, and exchange of data [136]. In addition, it is also not possible to store semantics and other attributes with the TIN in CityGML. For instance, CityGML requires more than 700 GB of storage space just to store the geometry of the 3DTOP10NL[117] terrains (without any topological information and semantic attributes) [136].

In this chapter, I present an improved representation to store massive terrains as TINs in CityGML. I discuss different data structures for compactly representing TINs, and explore how they can be implemented in CityGML to efficiently store massive TINs. I introduced three existing compact TIN data structures, namely *indexed triangles* [214], *triangle strips* or *tristrips* [230], and *stars* [147], as new geometry types in CityGML for representing TINs. A CityGML extension called the iTINs ADE is developed so that these data structures can be included in the data model of the CityGML in a structured manner. Experiments with massive real-world terrains show that, with this approach, it is possible to compress CityGML files up to 20x with one billion+ triangles, and our method has the added benefit of explicitly storing the topological relationships of a TIN model.

3.2 State of art in modelling terrains as TINs

In this section, I provide an overview of different representations for terrains, and data structures for modelling TINs.

3.2.1 TIN structure

A TIN is a network of non-overlapping triangles formed by the interconnection of points that are usually irregularly spaced [140]. For a given set of points, different triangulations can be constructed [67, 218]. TINs in GIS applications are generally constructed with the Delaunay triangulation to avoid long and skinny triangles [50]. A Delaunay triangulation (DT) of a point set S (Figure 3.1) is defined as a triangulation of S such that no point in S lies inside the circumcircle of any other triangle in the triangulation [50]. The DT maximizes the minimum angle, among all possible triangulations, to avoid long and skinny triangles.

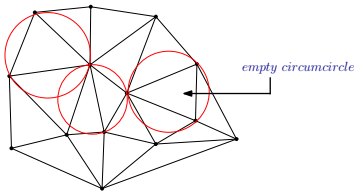


Figure 3.1: Delaunay Triangulation of a set of points

If the point set is associated with some constraints (segments, polygons, etc.) then a CDT (Constrained Delaunay Triangulation) can be constructed. A CDT is similar to DT but every input segment appears as an edge of the triangulation [226]. For instance, the 3DTOP10NL terrain is a constrained TIN (Figure 1.1). The number of triangles in a TIN is roughly two times the number of vertices used in triangulation. For instance, if you consider a dataset with n number of vertices with m number of vertices in the boundary of the convex hull, then there are ($2n$ - 2 - m) triangles in the triangulation [50]. In practice, $m<<n$ for real-world datasets as found in the GIS domain.

3.2.2 Representation of Terrains

Over the last few decades, grids and TINs have become the two most popular models for representing terrains. These are also referred to as field representations in GIS [42, 140]. A *field* is a model of spatial variation of an attribute over a spatial domain [148]. Fields are generally used to represent continuous

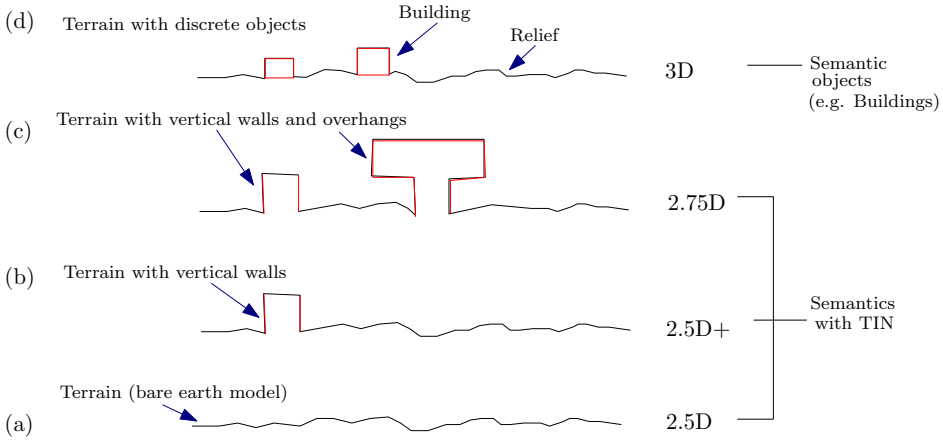


Figure 3.2: Different TIN representations for modelling terrains considered in this research. Semantics are attached to the entire TIN in 2.5D/2.5D+/2.75D and to the discrete objects (e.g. Buildings) embedded in the TIN in 3D

geographical phenomena, such as elevation of a terrain, surface temperature, etc. [42, 148]. A terrain can be modelled as a field, by a function $f(x, y)$ mapping each (x, y) location in the spatial domain to an elevation value z, i.e. $z = f(x, y)$ (Figure 3.2(a)). Modelling terrains by storing only one elevation value z for any (x, y) location is referred to as 2.5D (Figure 3.2(a)).

Grids are 2.5D in nature, but TINs can be more than 2.5D. It is not possible to represent features like vertical walls, roof overhangs, caves/tunnels, and overfolds like balconies and dormers with 2.5D field models. For instance, 3DTOP10NL terrain data has vertical walls. Modelling it in 2.5D will result in loss of information representing the vertical walls. Therefore, the focus is on geometrical representations which extend the field based 2.5D model to handle such features. In Figure 3.2(b), an example is shown where a location (x, y) has more than one elevation value (z) to model the vertical walls of natural or man-made objects like buildings. It is a so-called "2.5D+ model" which is topologically equivalent to 2.5D model as it is still a 2-manifold [207].

The ISO19107:2003 Spatial Schema [100] standard defines the *GM_ TIN* geometry type for representing TIN models, which in theory should allow vertical triangles in a TIN and therefore can be referred to as a 2.5D+ data structure. Features like balconies, and overhangs of rocks and roof surfaces are not covered by these models and are described using 2.75D models [85, 239]. A 2.75D model is a 2.5D+ model extended to model any 2-manifold surface with features like

balconies and overhangs (Figure 3.2(c)). These models are described in the context of TINs and not grids (except they could be described with voxels). They are sufficient for applications like visualization and watershed modelling [160].

However, for some applications, even 2.5D+ and 2.75D models have limitations. For instance, applications estimating population and building energy demand using 3D city models require to compute the volume of buildings [19] which is not possible to calculate using these terrain models. To compute the volume of a building, it should be closed at the base, i.e. modelled as solids. Based on the above argument, 3D model of a terrain is refered to as a 2.5D+/2.75D model with buildings modelled as solids (Figure 3.2(d)). The boundary surfaces of the solid can be modelled using TINs (triangles) or polygons.

3.2.3 TIN representations

Several data structures have been proposed in different domains to represent and store TINs. The simplest way of representing a TIN is to store each of its triangles as a list of vertex coordinates. *Simple Features* [204] is an example of such data structure. It stores each triangle as a closed linear ring of the coordinates of its vertices (Figure 3.7) [132]. It is simple to store and represent, and is supported by CityGML (GML) and almost all other spatial databases. However, it has certain limitations. First, the structure exhibits data redundancy, i.e. the first vertex of every linear ring is repeated as the last vertex of the ring (Figure 3.7). If we assume that the vertices follow a Poisson distribution, the average degree of a vertex in a 2D Delaunay triangulation is exactly 6 [202]. This suggests that on average each vertex is stored $6+(6/3) = 8$ times in the Simple Features [132]. The size of the dataset increases considerably with this repeated storage of vertex information for every triangle. Secondly, it does not explicitly store the adjacency relationships between the triangles which are necessary for traversing the TIN and for several spatial analysis operations, such as adjacency, connectivity, interpolation, derivation of slope/aspect, etc.

The need for storage efficient representations of triangular meshes has contributed to the development of a number of compact data structures which have different goals, such as compression and/or explicit storage of topological relationships, e.g. *indexed triangles* (similar to *OBJ*), *triangles with adjacency information* [29, 226], *stars* [27, 147] *triangle strips or tristrips*[230], *half-edge or DCEL* [163, 172], *SQuad* [86], *grouper* [159], *Laced Ring (LR)* [87], *zipper* [88], and *tripod* [228].

The TIN data structures that I consider in this research are *indexed triangles*, *stars*, and *triangle strips (or tristrips)*. The other data structures are also capable of reducing the storage requirements for a TIN and ensuring an efficient implementation with respect to run-time and mesh operations. They can be useful for streaming and visualization of large TINs. CityGML, on the other

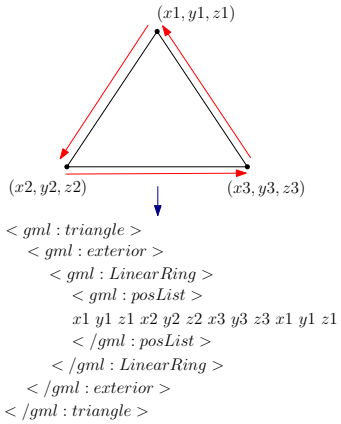
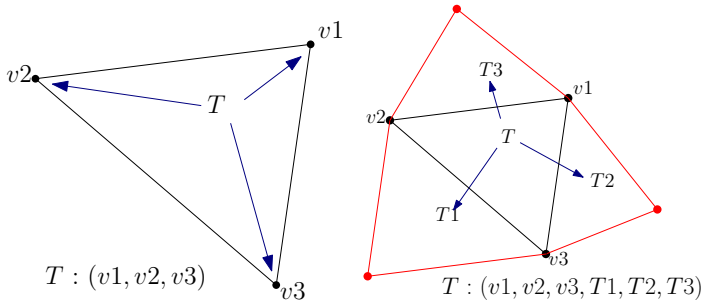


Figure 3.3: Simple Features representation for a triangle in GML [133]. The first vertex $(x1, y1, z1)$ of every triangle is repeated as the last vertex $(x1, y1, z1)$ to close the linear ring

hand, is a XML based data model for storing and representing 3D city objects. Visualization of data is not the main task of CityGML. Storing data in XML format with highly compressed data structures would require more preprocessing and later on extensive decoding for comprehensibility. Therefore, I only consider solutions that fit in the data model of CityGML and still ensure interoperability. I provide here the details of the selected TIN data structures.

Indexed Triangle
It stores every triangle of the TIN as references to the IDs of the three vertices forming the triangle [132]. The vertices are stored in a separate list with IDs and are not repeated for every triangle like in Simple Features. For instance, in Figure 3.4, a triangle T has three vertices with IDs $\{v_1, v_2, v_3\}$, each with a tuple of location coordinates (x, y, z). 3D data formats like OBJ, and ITF (Intermediate TIN Format) [246] use this data structure for storing triangles. The information about the adjacency and incidence relationships between the triangles of a TIN can be easily derived using this data structure.

TriStrip
A TriStrip or a triangle strip is a sequence of $n + 2$ vertices that represents n triangles of a triangulation (Figure 3.5) [230]. TriStrips are based on the same concept like Indexed Triangles but are potentially capable of reducing the storage by a factor of 3 [230]. The vertex coordinates (x, y, z) are stored in a separate list with their IDs.

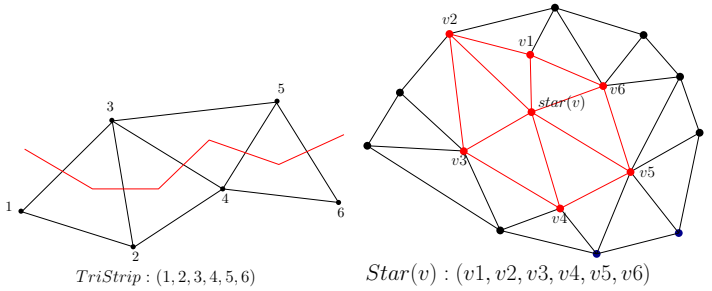


(a) Indexed Triangle [132]. Every triangle T is represented by the IDs of the three vertices (v_1, v_2, v_3) forming the triangle

(b) Triangle+ [132]. Every triangle T is represented by the IDs of the three vertices (v_1, v_2, v_3) forming the triangle and its three adjacent triangles $\{T1, T2, T3\}$

Figure 3.4: TIN data structures: Indexed Triangle and Triangle+

To generate a TriStrip, start with the three vertices of a triangle, then add a new vertex, and drop the oldest vertex to form the next triangle in sequence [230]. For instance, in Figure 3.5, the TriStrip $(1, 2, 3, 4, 5, 6)$ represents 4 triangles: $\Delta 123$ (formed by the first three vertices, $\Delta 234$ (formed by dropping the first vertex and taking up the next vertex in sequence), $\Delta 345$, and $\Delta 456$. OpenGL, and 3D graphics standards like COLLADA support triangle strips for representing the geometry of objects.



(a) TriStrip [230]. The first triangle ($\Delta 123$) is formed by the first three vertices and the next triangle ($\Delta 234$) is formed by dropping the first vertex and taking up the next vertex in sequence

(b) Star [132]. Every triangle incident to the vertex v is represented by v and the two consecutive vertices in the list v_i e.g. $\Delta v v_1 v_2$

Figure 3.5: TIN data structures: TriStrip and Star

Star
It is a vertex based, compressed, and pointerless data structure for compactly representing the triangular meshes [27]. The star of a vertex is represented as an ordered list (counter-clockwise) of IDs of the vertices incident to it [147], e.g. in Figure 3.5, the star of vertex v, $star(v)$, is represented by the vertex list $\{v_1, v_2, v_3, v_4, v_5, v_6\}$. The vertex coordinates (x, y, z) are stored in a separate list with their IDs. The triangles are not stored explicitly but computed on-the-fly. Every triangle incident to the vertex v is represented by v and two consecutive vertices in the list v_{i}, e.g. $\Delta v v_1 v_2$.

3.3 Terrains in CityGML

As mentioned before, CityGML is an open 3D standard for the representation, storage, and exchange of 3D city models [185]. The terrains are defined within the CityGML thematic module *Relief* and represented by the class `ReliefFeature` at 5 LODs (0-4) [185]. A terrain in CityGML can be represented either as a TIN `(TINReflief)`, or a grid `(RasterRelief)`, or a collection of points `(MasspointRelief)`, or break lines `(BreaklineRelief)`. It is also possible to represent a terrain as a combination of different terrain types in one CityGML dataset, e.g. as a TIN with break lines or as a coarse grid with some areas as TINs. The CityGML class that is of interest is `TINReflief`. It represents terrains as TINs using either `gml:TriangulatedSurface` or `gml:Tin` (GML3 geometry types). With `gml:TriangulatedSurface`, the geometry of the triangles (`gml:Triangle`) of a TIN are explicitly specified with Simple Features, whereas in `gml:Tin` a list of 3D control points is specified along with the triangles.

3.3.1 Problems in storing TIN terrains in CityGML

With advancements in 3D data acquisition and processing technologies, it is now possible to generate billions of 3D points for even an area of few square kilometres and, therefore, the TIN generated from these points is also massive in size. There are several problems in storing these massive terrains with CityGML [132].

1. *Massive size*: CityGML datasets becomes very large in size with the repeated storage of vertex information in Simple Features. Each of the triangles is specified with repetition of full vertex coordinate values which takes a lot of storage space.

2. *Little topology*: There is very little topological information stored with Simple Features. Each triangle is stored individually regardless of its neighbours, which hinders spatial analysis greatly.

3. *Badly modelled TIN geometry*: The class `gml:TriangulatedSurface` comprises of triangle patches forming the TIN. It is defined as a separate

subclass of `gml:Surface` but it could have been defined as a subclass of `gml:CompositeSurface`. A composite surface requires its elements to be disjoint, not to have overlapping interiors and must be topologically connected along their boundaries. These prerequisites are fulfilled in case of TINs.

4. *No specifications for terrain LODs*: There is no distinction between different LODs of a terrain in CityGML at geometrical and semantic level. The CityGML 3.0 provides extended LODs for the Building module only and the LOD specifications of other modules like *Relief*, etc. are left out [84, 155]. Giving only an attribute (e.g. `gml:lod`) does not solve the issue if we cannot identify the difference between LODs.

5. *Vertical triangles are not handled*: 3DTOP10NL has the skeletons of the urban objects like buildings, roads, etc. integrated in the terrain. In a way, it is not completely a 2.5D but a 2.8D model with vertical walls. A 2.8D models is a 2-manifold surface embedded in a 3D space [85]. When a 2.8D model is projected on a 2D surface, the vertical surfaces flatten out which distorts the geometry of the model. CityGML is implemented as an application schema of GML3 [185]. The `gml:Tin` is based on ISO 19107:2003 specification of `GM_TIN` which in theory is a *2.5D+* structure and can have vertical triangles. There is no mechanism in CityGML to mark out these vertical surfaces so as to remove them while transforming from 3D to 2D.

6. *No support for tiling*: The main memory of a system plays a key role in deciding the maximum size of a dataset that can be processed [97]. If the size of the datasets exceeds the available memory limit then it is split into small parts (called *tiles*). The concept of tiling the TIN cannot be extended to CityGML as there can be triangles spanning several tiles. Such triangles are repeated in the spanned tiles to complete the OGC SF closed linear ring structure thereby causing information redundancy in the CityGML datasets [132].

7. *Unclear TIN and grid combination*: The CityGML documentation describes having a combination of multiple terrain types in a single CityGML instance. However, there can be problems of mismatch of geometry when it comes to combining a TIN and a grid for a terrain dataset. For instance, in Figure 3.6, the terrain is represented by a grid along with a certain area represented by a TIN. The TIN vertices may lie anywhere on the grid and not necessarily at the centre of each grid pixel. Deriving the exact value of elevation of TIN vertices from grid pixels can be an expensive operation in this case.

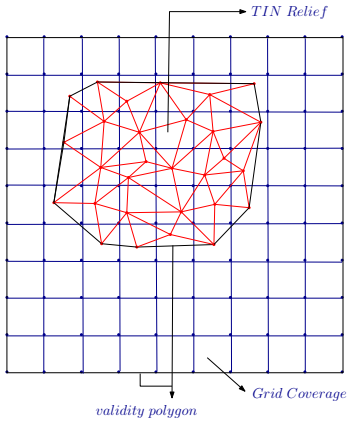


Figure 3.6: TIN + Grid combination in CityGML. TIN vertices may lie anywhere on the grid and not necessarily at the centre of each grid pixel.

3.4 Modelling a CityGML extension for massive TINs

The new data structures that I selected for compactly representing TINs, i.e. *indexed triangle, triangle strips, stars*, are not present in the data model of GML (and CityGML). Therefore, I introduce these new TIN types as an extension to the data model of CityGML. In Chapter 2, I provided an overview of the two available approaches for modelling an extension to CityGML, namely Generics and ADEs. After comparing the two alternatives, I adopted the ADE approach to model an extension to CityGML.

3.4.1 Modelling choices for new TIN geometry types

The geometry model of GML3 consists of geometric primitives such as points, lines, and polygons, which are combined to form complexes, aggregates, or composite geometries. Therefore, new geometry types are introduced in the GML3 geometry model and extend them to CityGML feature types as an ADE.

To avoid any name conflict with the existing GML elements, the new schema elements are defined in a separate XSD file `'iTIN_GML.xsd'` with a different namespace `'https://3d.bk.tudelft.nl/schemas/iTIN_GML'` and the `'igml'` identifier. We introduce new geometry types (primitives, aggregates, and composites) in this model for compactly representing TINs. New abstract classes for representing these geometry types are added so as not to disturb the original hierarchy of the GML3 model.

- `_iPointPrimitive`. It is an abstract class for modelling the point geometries. It is modelled as a type of `gml:_GeometricPrimitive`.
- `iPoint`. An `iPoint` (or an indexed Point) represents the geometry of an individual point (or vertex). It is modelled as a type of `igml:_iPoint-Primitive`. Each `iPoint` has an integer ID `(igml:id)` and a list of its coordinates (x, y, z) `(igml:coordinates)` (see Snippet 1).

Snippet 1: iPoint representation in iTINs_ADE

```
<igml:iPoint>
  <igml:id>1234</igml:id>
  <igml:coordinates>
  85027.492 447446.125 1.51
  </igml:coordinates>
</igml:iPoint>
```

- `igml:iPointList`. An `iPointList` (or an indexed Point List) is a list of all the points (or vertices) of a surface defined by space separated values of all the coordinates (see Snippet 2). It is modelled as a type of `igml:_iPointPrimitive`.

Snippet 2: iPointList representation in iTINs_ADE

```
<igml:iPointList>
  <igml:coordinates>
    85027.492 447446.125 1.51
    85027.289 447446.156 1.31
    85049.219 447448.312 1.37
    85068.219 447447.332 1.64
    ....
  <igml:coordinates>
</igml:iPointList>
```

- `igml:iMultiPoint`. An `iMultiPoint` is a collection of all the points (i.e. vertices) of a surface. With `iMultiPoint`, it is possible to store points either as a collection of individual points or as one continuous list of points (see snippet 3).

Snippet 3: iMultiPoint representation in iTINs_ADE

```
<igml:iMultiPoint>
  <igml:iPointMember>
    <igml:iPoint>
      ....
    </igml:iPoint>
  <igml:iPointMember>
  ....
</igml:iMultiPoint>
```

- `igml:iLine`. An `iLine` (or an indexed Line) represents the geometry of an individual line segment (or curve). It is modelled as a type of `gml:_Curve` which is subtype of `gml:_GeometricPrimitive`. Any separate abstract base class (such as `_iLine`) is not introduced because it is a complete geometry (with points and indexes) and hence, can be reused with `gml:MultiCurve`. The existing hierarchy of elements in the GML model is followed for defining new classes in the model. Each iLine has an ID `(igml:id)` and a list of IDs of the points forming the line `(igml:indexes)` (see snippet 4). The `igml:indexes` lists the IDs of the points comprising the geometry instead of repeating the coordinate values of the points again.

Snippet 4: iLine representation in iTINs_ADE

```
<igml:iLine>
  <igml:id>D23</igml:id>
  <igml:iPoints>
  ....
  </igml:iPoints>
  <igml:indexes>1 2</igml:indexes>
</igml:iLine>
```

- `igml:_iSurface`. This class is introduced to model the surfaces. It is modelled as a type of `gml:_GeometricPrimitive`. It has 3 subclasses: `igml:_iSurfacePrimitive` for modelling individual surface elements (polygon and triangle), `igml:_iTinPrimitive` for modelling TIN representations, and `igml:_iCompositeSurface` for modelling TINs.

- `igml:iTriangle`. An `iTriangle` (or an indexed Triangle) represents the geometry of an individual triangle. It is modelled as a type of `igml:_iSurfacePrimitive`. An `iTriangle` is specified by the references to IDs of the 3 vertices of the triangle (see snippet 5). It has an optional attribute (`vertical`) to specify if the triangle is a vertical triangle. For some applications such as flow modelling, adjacency and network analysis, it is sufficient to use a city model and its buildings as a single triangulated surface containing vertical triangles instead of using a volumetric model [83]. The `vertical` attribute helps us to identify these vertical surfaces modelled in the terrain without relying on the geometry and on-the-fly computation (which are prone to precision errors). This means that the model is more than 2.5D, but is less than 3D; the geometry is 3D, but the underlying topology remains 2D.

Snippet 5: iTriangle representation in iTINs_ADE

```
<igml:iTriangle>
  <igml:id>34</igml:id>
  <igml:vertical>true</igml:vertical>
  <igml:indexes>1 2 3</igml:indexes>
```

```
</igml:iTriangle>
```

- `igml:iTriangleList`. An `iTriangleList` (or an indexed triangle list) is a space separated list of IDs of the vertices of all the triangles. It is modelled as a type of `igml:_iSurfacePrimitive`.
- `igml:iPolygon`. An `iPolygon` (or an indexed Polygon) represents the geometry of an individual polygon. It is also modelled as a type of `igml:_iSurfacePrimitive` and has the same geometrical representation as an iTriangle. An iPolygon is specified by the references to IDs of the vertices (>3) of the polygon (see snippet 6). It also has an optional attribute `(vertical)` to specify if the polygon is a vertical surface.

Snippet 6: iPolygon representation in iTINs_ADE

```
<igml:iPolygon>
  <igml:id>14</igml:id>
  <igml:vertical>true</igml:vertical>
  <igml:indexes>3 4 5 6</igml:indexes>
</igml:iPolygon>
```

- `igml:_iCompositeSurface`. It is introduced to model disjoint, non-overlapping, topologically connected surfaces. It has two subclasses `igml:iTIN` and `igml:iPolygonSurface`.
- `igml:iTIN`. An iTIN is introduced to represent TINs as a subclass of `igml:_iCompositeSurface` (and not aggregates) because TINs represent surfaces with disjoint, non-overlapping and topologically connected triangles. Apart from the above described geometric primitives and aggregates, three new TIN representation types: `igml:iTriangulatedSurface`, `igml:iStars` and `igml:iTriStrips` are also introduced as subclasses of `igml:_TinPrimitives`. In `igml:iTIN`, the TIN vertices are represented using `igml:iMultiPoint` and the TIN surface can be represented using any of the three new surface types.

Snippet 7: iTIN representation in iTINs_ADE

```
<igml:iTIN>
  <igml:id>A24</igml:id>
  <igml:iTinPoints>
    <igml:iMultiPoint>
     ....
    </igml:iMultiPoint>
  </igml:iTinPoints>
  <igml:iTinSurface>
   .....
  <igml:iTinSurface>
</gml:iTIN>
```

- `igml:iTriangulatedSurface`. An iTriangulatedSurface stores triangles either as a collection of individual `igml:iTriangle` or as a `igml:iTriangle-List` referenced through `igml:iTrianglePatch` element (see snippet 8).

Snippet 8: iTriangulatedSurface representation in iTINs_ADE

```
<igml:iTriangulatedSurface>
  <igml:id>A24</igml:id>
  <igml:iTrianglePatch>
   <igml:iTriangle>
    ....
   </igml:iTriangle>
  </igml:iTrianglePatch>
  ....
</igml:iTriangulatedSurface>
```

- `igml:iTriStrip`. An `iTriStrip` is collection of individual triangle strips (`igml:iTStrip`) (see snippet 9). In each `iTstrip`, the first triangle is formed from first, second, and third vertex. Each subsequent triangle is formed from the next vertex in sequence, reusing the previous two vertices.

Snippet 9: iTriStrip representation in iTINs_ADE

```
<igml:iTriStrip>
  <igml:id>B54</igml:id>
  <igml:iTstrip id = "1"> 1 2 3 4 5 </igml:iTstrip>
  <igml:iTstrip id = "2"> 11 12 13 14 </igml:iTstrip>
  ....
</igml:iTriStrip>
```

- `igml:iStars`. An `iStars` is a collection of `igml:iStar` elements defined for every vertex of a triangulated surface. For every vertex, an `iStar` stores an ordered list of IDs of the vertices incident to it (see snippet below). Every triangle incident to a vertex is represented by the ID of that vertex and the IDs of two consecutive vertices in the list.

Snippet 10: iStars representation in iTINs_ADE

```
<igml:iStars>
  <igml:id>A34</igml:id>
  <igml:iStar id = "1">2 3 4 5 6 7</igml:iStar>
  <igml:iStar id = "2">3 4 7 8 9 11</igml:iStar>
  ....
</igml:iStars>
```

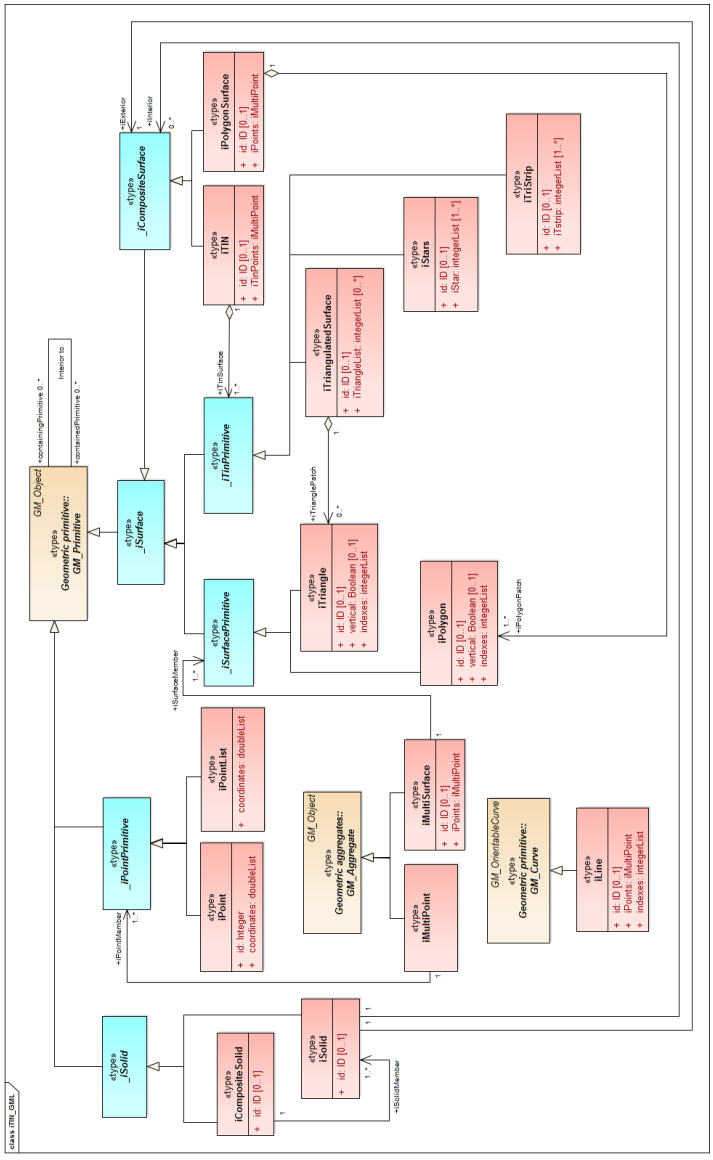


Figure 3.7: Proposed geometry types in the GML3 geometric model for modelling TINs (abstract classes are shown in blue and implementation classes are shown in red)

3.4.2 Extending CityGML for massive terrains with the proposed iTIN types

For modelling terrains as TINs, the `iTIN_GML` elements are added to CityGML using an ADE. The initial idea was to integrate these TIN representations directly in the GML model so as to use the same namespace and identifier of GML. CityGML would then inherit these geometry types automatically from the enhanced GML model. This would have eliminated the need to extend the existing CityGML classes with these new geometrical representations. However, both GML and CityGML are controlled by a formal authority: OGC. This would have changed the original standard published by the OGC.

Therefore, to show the benefits of this approach, it was developed as an ADE. We created a separate package to model the new TIN geometry types and added them to CityGML by extending the existing CityGML classes in an ADE package. Moreover, these geometry types can be easily added to the original GML/CityGML model, if approved by OGC. The ADE classes are defined in a separate file "CityGML_iTINs_ADE.xsd" with a different namespace `"https://3d.bk.tudelft.nl/schemas/iTINs_ADE"` and the `itin` identifier.

iTINRelief. The CityGML *Relief* module is extended to include the `iTIN_GML` elements for modelling terrains as TINs. In the CityGML *Relief* module, a new relief component called `iTINRelief` is introduced as a subclass `dem:TINRelief`. iTINRelief extends all the properties of the base class like name, description and LOD, etc. and has `igml:iTIN` geometrical representation (Figure 3.8). In the original `dem:TINRelief` class, the level of detail (LOD) is specified using `dem:lod` element. Here, separate geometrical representations for the relief LODs (0-4) using `lod0iTIN`, `lod1iTIN`, `lod2iTIN`, `lod3iTIN` and `lod4iTIN` are introduced. Another element called `iExtent` is also introduced to mark the extent of the TIN using `igml:iPolygon` geometry. To represent the break lines in a TIN, an element called `iBreaklines` with geometry `igml:iLine` is introduced.

Snippet 11: iTINRelief representation in iTINs_ADE

```
<cityObjectMember>
  <dem:ReliefFeature>
    <gml:name>Example iTINRelief</gml:name>
    <dem:lod>1</dem:lod>
    <dem:reliefComponent>
      <itin:iTINRelief>
        <dem:lod>1</dem:lod>
        <itin:iTINobject>
          <itin:lod1iTIN>
            <igml:iTIN>
              ....
            </igml:iTIN>
```

```
        <itin:lod1iTIN>
      </itin:iTINobject>
    </itin:iTINRelief>
  </dem:reliefComponent>
</dem:ReliefFeature>
</cityObjectMember>
```

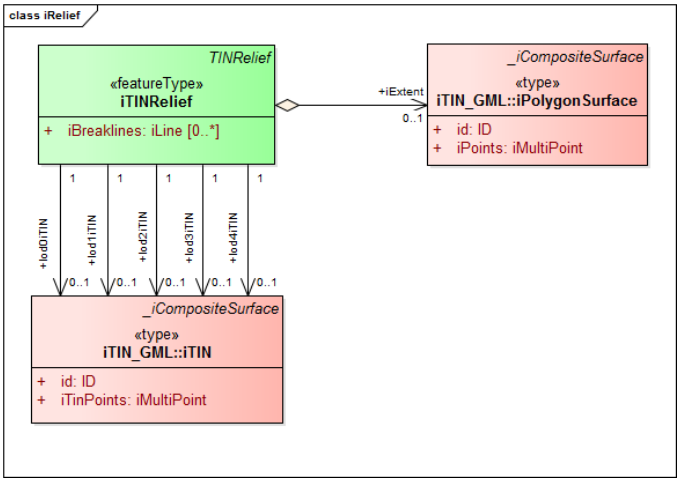


Figure 3.8: `iTINRelief` modelled in CityGML iTINs ADE using iTIN_GML

3.5 Implementation and experiments with real world datasets

3.5.1 CityGML iTINs ADE schema generation

The ShapeChange tool is used to derive the XML schema of the CityGML iTINs ADE from the UML model. Only the XML Schema for the CityGML ADE is generated and not for the whole data model as the former is already publicly available. The generated CityGML iTINs ADE schema only requires to import the existing CityGML schema. These dependencies are resolved by the ShapeChange during the transformation from UML packages to XML schema. The UML model and XML schema for the CityGML iTINs ADE are publicly available here: https://github.com/tudelft3d/CityGML_iTINs_ADE

3.5.2 Datasets used

The terrain datasets used for testing the implementation are:

1. *AHN3 TIN. AHN3 (Actueel Hoogtebestand Nederland version 3)* [2] is the national height model of the Netherlands, and contains billions of 3D points (more than 10 points/m^2) covering the whole country. AHN3 Tile #37EN/1 (size 5 km X 6.25 km) of the AHN3 point cloud is used as input for generating the TIN using LAStools [92]. The dataset is a TIN without constraints generated using the streaming paradigm of Isenburg et al. [95]. It is available in the streaming mesh format (*.sma).
2. *3DBGT. 3DBGT (3D Basisregistratie Grootschalige Topografie)* is the 3D city model of the Netherlands created using the open source software *3dfier*[1]. 3DBGT is a constrained triangulation generated from the AHN3 point cloud and the 2D BGT (large-scale 2D topographic dataset of the Netherlands) footprints [14]. 3dfier takes 2D topographic datasets and lifts every 2D polygon to the required height to make them 3D. This height information is obtained from the point cloud data. We used 3DBGT TIN of the Amsterdam area for testing. The dataset is available in OBJ format (*.obj).
3. *3DTOP10NL.* 3DTOP10NL is the 3D city model of Netherlands, which covers the whole country, including buildings, terrain, roads, canals, etc. in 1368 tiles. It is generated by adding the height information from AHN2 point cloud to the 2D topographic objects in TOP10NL [73]. The layer that is of interest in the 3DTOP10NL dataset is the `''terreinVlak_3D_LOD0''` which contains the terrain model with more than 1 billion triangles. The dataset is available in ESRI GeoDatabase format (*.gdb).

It should be noted that neither of the aforementioned datasets are the official datasets of the Netherlands. The details of the input terrain datasets along with their size in CityGML format are given in Table 3.2.

3.5.3 Prototype testing

A prototype was created to introduce new TIN representations in CityGML datasets. The prototype reads the input datasets and maps the Simple Features representation of triangles to index based structure of `igml:iTIN`. The resulting storage size of the prototype testing are given in Table 3.2 along with the achieved compression factors.

The time taken to generate data in CityGML and CityGML_iTINs_ADE formats from original test datasets (Table 3.1) is also compared to observe the

[1] https://3d.bk.tudelft.nl/opendata/3dfier/

performance of the system in handling massive terrain data. These tests were performed on a Linux server with 40 Intel Xeon E5-2650 v3 CPUs, 128 GB RAM, 3.3 GHz base clock speed and 3.6 GHz turbo boost speed. Parallel programming in Python was used to leverage the power of the multiple processors available on the Linux server. The three test datasets are available in three different formats (OBJ, SMA & GDB) and the time taken to generate output data from these datasets differs significantly. From Table 3.1, it can be seen that it takes less time to generate CityGML data from 3DTOP10NL GeoDatabase. This can be attributed to the fact that both CityGML and ESRI GeoDatabase follow Simple Features structure for representing geometry. While generating iTIN_GML geometry types from this Simple Features structure, most of the time is consumed in cleaning the vertices (removing duplicate), generating integer IDs for the vertices and assigning these indexes to the triangles for representing geometry. However, in case of other formats like OBJ & SMA which already follow simple indexing scheme, the `igml:iTriangulatedSurface` structure is generated very quickly. For `igml:iTriStrip` and `igml:iStars`, the data generation time is a bit high as it also includes the time taken to compute the neighbouring triangles/vertices (required for TIN traversal).

Tests were also performed for the storage size of quantized vertices [96]. A vertex is called quantized when we store only the difference of its coordinates from the centroid vertex (or any other vertex) and not the full vertex coordinates. The centroid vertex is the centroid of the vertices of the TIN or can also be selected randomly. Storing the difference of the coordinates from the first vertex of the TIN was also tested. However, storing quantized vertices did not change the compression factors significantly. As this was not the main objective of our study, it was not tested further.

As it can be observed from the results, the highest compression factor is achieved by using iTriStrip referencing scheme for storing TINs in place of Simple Features structure. The data structures in decreasing order of storage requirements are:

$$\text{iStars} > \text{iTriangulatedSurface} > \text{iTriStrip}$$

Although, inclusion of triangle strips (iTriStrips) provides maximum reduction in storage size, it has certain topological restrictions. We used the *TriangleStripifier* module of the *PyFFI* python package to generate triangle strips for our datasets [212]. *TriangleStripifier* is a python adaptation of the *NvTriStrip* library [182] and converts triangles into a list of strips. A triangle strip enters each triangle at one edge (known as *entry-edge*) and exits that triangle on the left or the right remaining edges (known as *exit-edges*) [230]. The triangle strip alternates between left and right exit-edges with each successive triangle until it reaches a triangle with no forwards connections [230]. For the remaining triangles, the same process is repeated until all the triangles are placed in triangle strips.

Therefore, for a single TIN, we can have a number of disconnected triangle strips storing the mesh triangles (Figure 3.9). This means there is local topological connectivity within the individual triangle strips but no overall connectivity for the entire TIN. Certain operations are thus not possible in constant time such as finding the adjacent triangles of a given triangle.

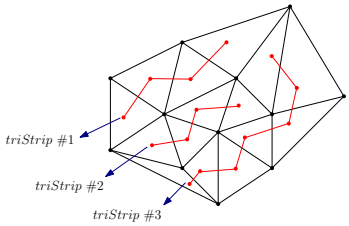


Figure 3.9: A single TIN can have a number of disconnected triangle strips. There is local connectivity within each strip (shown in red color) but no overall connectivity for the entire TIN

This is not the case with Stars. When all the Stars in a TIN are represented, each triangle is present in exactly 3 stars (its 3 vertices) and each edge is present in exactly 2 stars (its 2 vertices) [147]. There is a significant overlap in the stars from which we can derive the adjacency and incidence relationships of the triangles of a TIN [147]. For a given vertex we can easily find the incident edges or triangles using stars. Therefore, these data structures in increasing order of topology can be represented as:

$$\text{iTriStrip} < \text{iTriangulatedSurface} < \text{iStars}$$

Terrain Dataset	CityGML	iTS	iTriStrip	iStar
3DBGT (obj)	25.63 min	15.68 min	63.83 min	27.21 min
Tile #37EN/1 (sma)	52.19 min	38.87 min	93.77 min	54.32 min
3DTOP10NL (gdb)	38.54 min	121.63 min	194.31 min	166.91 min

Table 3.1: Time taken to generate CityGML & CityGML_iTINs_ADE data from test datasets

Terrain dataset	#Triangles	CityGML size	iTS		iTriStrip		iStars	
			size	CF	size	CF	size	CF
3DBGT	13,688,402	4.65 GB	337.92 MB	14.32	236.54 MB	20.1	816.13 MB	5.83
Tile #37EN/1	40,788,573	13.98 GB	952.32 MB	15.13	747.52 MB	19.22	2.28 GB	6.13
3DTOP10NL	1,156,641,666	698.74 GB	46.64 GB	14.38	37.43 GB	18.67	108.33 GB	6.45

Table 3.2: Details of the input datasets showing the number of triangles in each terrain dataset, and the storage space required for storing each dataset in CityGML and CityGML iTINs ADE format (CF = Compression Factor, iTS = iTriangulatedSurface)

3.6 Conclusion

In this chapter, I have presented an improved representation for compactly storing massive terrains in the context of 3D city models. Several TIN data structures for their storage requirements and topology were investigated, and explored how they can be implemented in CityGML for compactly storing massive TINs. Three new indexed-based geometry types (*indexed triangles, triangle strips, and stars*) were introduced for representing TINs in the GML schema and extended them to CityGML as an ADE.

This research is a stepping stone in the direction of reducing the large size of CityGML datasets while still maintaining usability for different applications. CityGML is designed for the storage and exchange of 3D city models and not for visualizing them. To visualize CityGML models over web, they are usually converted to commonly used 3D graphics formats. The CityGML iTINs ADE datasets are expected to load faster over web owing to their small file sizes and indexed-based geometry representations. These datasets can also be used for applications, other than visualisations, utilizing CityGML models, such as noise modelling, flood modelling, visibility analysis, etc. The next step is to integrate this ADE into the database to see its capabilities in handling terrain data. We have discussed this solution in Chapter 4 using the 3DCityDB[2] database.

Further, CityGML lacks the concept of LODs for terrains. The LOD of a terrain is expressed as integer attribute, `gml:lod`, with values between 0 and 4. There is no differentiation between different terrain LODs at geometrical and semantic level although its possible to model different levels of terrain [158]. To address this gap, new elements (`lod1iTIN, lod2iTIN,..., lod4iTIN`) were added in the CityGML *Relief* (and other modules) to model different LODs of the terrain. However, the proper specification to model the geometry and semantics of terrains at each LOD is still missing. To address this issue, I have discussed a proposal for modelling terrains at different LODs in Chapter 5.

[2] https://www.3dcitydb.org/

Chapter 4

Database storage for massive TINs

4.1 Introduction

In Chapter 3, I presented an improved representation for massive TIN terrains in CityGML files. In this chapter, I investigate the use of DBMS (Database Management Systems) for storing and managing massive TINs. DBMS are arguably the best tool to store and manage very large datasets (of any kind) and are already part of the toolbox of most GIS practitioners. DBMS solutions offer several advantages over file-based storage: reduced data access and storage costs, security, multi-user access, scalability, rich query language, etc. [75]. Traditionally, DBMS were used for handling administrative and other voluminous data but now they have evolved to integrate the spatial component. At present, database systems like Oracle, PostGIS, IBM DB2, Ingres, Informix, MySQL, etc. all provide some support for spatial data types, spatial indexing and other extended functionalities. Most of these database systems offer only 2D vector types (generally point, line, and polygon). Storage and analysis of spatial data can be done with SQL queries.

There exists several data structures for representing TINs in memory but only little has been done so far for their database implementations. In case of completely storing the TIN (along with the points, triangles and topology) in a database, one can fully depend on the DBMS for its management. The entire TIN is available in the database and need not be recomputed and can be queried and analysed with SQL queries. Initial implementations for storing TINs are available in both commercial and open source DBMS, Oracle Spatial `SDO_TIN` [206, 214] and PostgreSQL/PostGIS Simple Features [204], respectively. 3DCityDB, a geodatabase to store and manage 3D city models, also supports storing TINs. It is the database implementation of the CityGML schema on top of a spatial RDBMS (Relational Database Management System) such as Oracle and

PostgreSQL/PostGIS. However, it uses the SF structure (explained in Chapter 3) and this brings the same drawbacks associated with storing massive TIN terrains in CityGML (explained in Chapter 3).

The CityGML iTINs ADE described in Chapter 3 for compactly storing massive TINs addressed these problems for file-based storage. In this chapter, I discuss how the ADE with compact data structures, i.e. *indexed triangles, triangle strips* and *stars*, can be implemented in a database for efficient storage and management of massive TIN terrains. An extension to the 3DCityDB (PostgreSQL/PostGIS) database has been developed to add these data structures to the database schema. The implementation is tested with massive real-world datasets and compared with PostgreSQL/PostGIS Simple Feature implementation with respect to storage size, indexing and loading times, and topology. This research does not include implementation and testing with commercial DBMS solutions such as Oracle Spatial in its scope because the focus is on open source technologies.

4.2 Storing TINs in a DBMS: Current solutions and limitations

Only two TIN representations have been implemented so far in a DBMS namely, `SDO_TIN` in Oracle Spatial, and in PostGIS a specific TIN type which is a wrapper for the OGC Simple Features. I review here the few existing solutions.

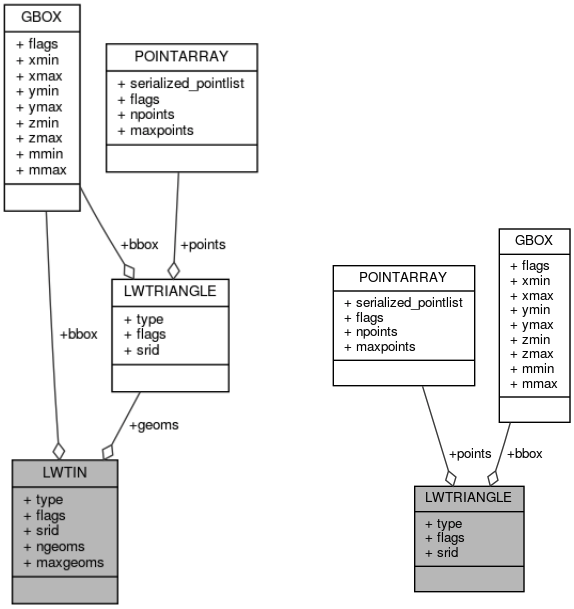
4.2.1 TINs in PostgreSQL/PostGIS

PostGIS implements the OGC Simple Features structure to store TINs. A triangle is stored as a polygon with 3 non-collinear and distinct points without any interior rings or holes. A TIN is described as a contiguous collection of such triangles (i.e. a polyhedral surface) which share common edges. In 3D, with the inclusion of z-values (height), they are referred as *TriangleZ* and *TINZ* respectively [204]. For example, the WKT (Well Known Text) for representing *TriangleZ* and *TINZ* is:
TRIANGLEZ ((0 0 0, 0 1 0, 1 1 0, 0 0 0)) and
TINZ (((0 0 0, 0 0 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 1 0, 0 0 0))).

In PostGIS, the geometry is stored with a unique ID, bounding box, and WKB (Well Known Binary) representation of OGC Geometry. At present, the support for storing *TriangleZ* and *TINZ* in the database is provided by PostGIS v2. Figure 4.1 depicts the structure of Triangle and TIN type in PostGIS.

The Simple Features structure has several limitations which are described in Chapter 3. Furthermore, indexing the data at the triangle level is problematic since the size of the spatial index (such as GiST) can become huge [147], nearly 80% of the size of a TIN. Spatial indexes are more complex than a standard

index such a B-tree [243]. Although not stated specifically, the TIN as a patch of triangles is a block in its own right, as well as all Multi geometries in the OGC SF.



(a) PostGIS TIN structure **(b)** PostGIS Triangle structure

Figure 4.1: TIN structure in PostGIS [209]

4.2.2 TINs in Oracle Spatial

Oracle has Oracle `SDO_TIN` extension to store TINs. It is similar in structure and storage scheme to the Oracle `SDO_PC` extension but has an additional column to store the TIN objects. The `SDO_PC` is meant to store point clouds in an Oracle database. It partitions the point cloud into multiple blocks which are uniquely identified by their block IDs and then spatially indexed [78].

Similarly, in `SDO_TIN`, the TIN is divided into blocks and these blocks are stored in the TIN block table `(SDO_TIN_BLK)`. Each row in the TIN block table has multiple fields which store metadata about the individual block, e.g. number of points and triangles stored, the spatial extent, a unique id for identifying the block, etc. Further, it stores two objects: a points blob (Binary Large Object)

and a triangle blob. The points blob is an array of points, storing the coordinates as doubles, together with the block id and a point id for every point `(BLK_ID, PT_ID)`. The triangle blob is an array of triangles, containing references to the three vertices for each triangle. Each vertex is identified by a pair of the block id and point id `(BLK_ID, PT_ID)`.

The problem of splitting a TIN into blocks without breaking its topology is solved here by explicitly storing the reference to each block. The indexing is performed at the block level and not at the vertex or triangle level. It does not explicitly store the topology of a TIN. Each block of a TIN is considered as a complete and connected patch.

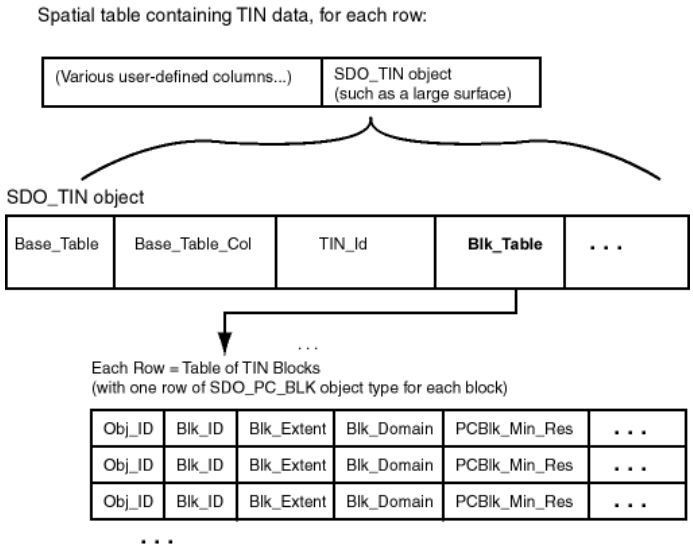


Figure 4.2: TIN storage model in Oracle [206]

4.2.3 Other academic implementations

Jones et al. [116] proposed *Implicit TIN* where only the vertices are explicitly stored, together with the definition of any linear constraints in the TIN. The triangulation is reconstructed in a query when it is required for a user operation by retrieving relevant vertices (and any constraints) and then executing the Delaunay triangulation algorithm [116]. It yields exactly the same triangles when same inputs for the triangulation are used in the exactly same order. This approach is very fast for queries on small areas of interest. However, as the

number of points that need to be triangulated increases, the longer such a query will take.

Among other implementations, the *pgTIN* extension for PostgreSQL by Ledoux [147] stores TIN as uncompressed stars. The *pgTIN* only makes use of a vertex table with each row containing a vertex ID, vertex coordinates and an array of references to the vertices in the star of the vertex. Triangles are computed on-the-fly using the explicitly stored topology. The vertices in the table are indexed using a B-tree. The main limitation of the approach is that it lacks tiling of a TIN.

Another implementation is the *multistar* approach by Pronk [211] which is built upon the aforementioned *pgTIN* prototype. It implements a tiling scheme in the star data structure with each row of database storing multiple vertices and stars. Only the spatial extent of each tile is spatially indexed. We don't implement a tiling scheme in this work, therefore multistar approach was not used.

4.3 Implementing the new TIN representations in the DBMS

Storage and management of massive TIN terrains can be tackled in different ways. One way is to compactly store them in file based systems, as shown in Chapter 3 via the CityGML iTINs ADE. Another way is to store them in database solutions, which I address in this chapter. I focus specifically on implementing in 3DCityDB the `iTINRlief` class and the three compact TIN data structures, namely *indexed triangle*, *triangle strips* and *stars* introduced in the CityGML iTINs ADE in Chapter 3. 3DCityDB version 3.3.1[1] does not provide a generic solution for handling CityGML ADE datasets.

This implementation for TIN terrains is based on the work of Agugiaro et al. [1] for extending 3DCityDB to store the CityGML Energy ADE datasets. The implementation to extend 3DCityDB (with PostgreSQL/PostGIS) to include the new classes introduced in the CityGML iTINs ADE for storing massive TINs terrains can be summarised in two main points:

- registration of the ADE via the ADE Registration module,
- mapping of ADE classes to new or existing tables and creating stored procedures for the tables.

4.3.1 Registration of the ADE

The *ADE Registration module* is meant to allow for the registration of an ADE in the 3DCityDB instance. This is based on the same methodology developed by Agugiaro et al. [1]. An ADE table *(citydb.ade)* is created in the *citydb* schema

[1] At the time of working on this project, the latest version of 3DCityDB was 3.3.1.

which stores the metadata of the registered ADE such as name, id, description, version, etc. Each registered ADE is assigned a unique prefix, e.g. *itin_* for our CityGML iTINs ADE, to avoid conflicts among the tables Agugiaro et al. [1] belonging to the other registered ADEs and standard CityGML tables.

```
-- Add entry into the table citydb.ade
INSERT INTO citydb.ade (adeid, name, description, version, db_prefix)

VALUES ("iTINsADE001","iTINs ADE", "CityGML iTINs ADE for terrains", "0.1",
"itin");
```

Upon registering an ADE in the database, the following existing 3DCityDB tables in the *citydb* schema are also updated:

- `SCHEMA` table *(citydb.schema)*: This table stores the version, namespaces and schema locations of the different CityGML classes. This table is updated to add the `iTINRelief` class.

```
-- Add entry into table citydb.schema
WITH a AS (SELECT id FROM citydb.ade WHERE db_prefix="itin")

INSERT INTO citydb.schema (ade_id, is_ade_root, citygml_version,
xml_namespace_prefix, xml_namespace_uri, xml_schema_location)

SELECT a.id, 1, "2.0", "itin",
"https://3d.bk.tudelft.nl/schemas/CityGML_iTINs_ADE",
"hhttps://3d.bk.tudelft.nl/schemas/CityGML_iTINs_ADE/
CityGML_iTINs_ADE.xsd" FROM a;
```

- `OBJECTCLASS` table *(citydb.objectclass)*: This table stores the names of all the classes of the CityGML schema. It also stores the relation of the subclasses with the parent class via the attribute `SUPERCLASS_ID` in the subclass as a foreign key to the `ID` of the parent class. This table is the central registry for all the CityGML classes. It is updated to add the new `iTINRelief` class and its relation to the parent class `_ReliefComponent` (ID 15 in the original database schema).

```
-- Add entries into table citydb.objectclass
INSERT INTO citydb.objectclass (id, superclass_id, baseclass_id,
is_ade_class, classname, tablename)

VALUES (140, 15, 3, 1, "iTINRelief", "itin_itinrelief");
```

- `SCHEMA_REFERENCING` table *(citydb.schema_referencing)*: This table is an associate table with two foreign key columns *(referenced_id, referencing_id)* to link the respective referencing and referenced schemas. The table is updated to reflect that the `iTINRelief` schema references the CityGML `Relief` schema.

```
-- Add entry into table SCHEMA_REFERENCING
-- Add for CityGML version 2.0
WITH ade AS (SELECT id FROM citydb.schema WHERE
xml_namespace_uri="https://3d.bk.tudelft.nl/schemas/CityGML_iTINs_ADE"
AND citygml_version="2.0"),
c AS (SELECT id FROM citydb.schema WHERE
xml_namespace_uri="http://schemas.opengis.net/citygml/relief/2.0")

INSERT INTO citydb.schema_referencing (referenced_id, referencing_id)
SELECT c.id,ade.id FROM ade, c;

-- Add for CityGML version 1.0
WITH ade AS (SELECT id FROM citydb.schema WHERE
xml_namespace_uri="https://3d.bk.tudelft.nl/schemas/CityGML_iTINs_ADE"
AND citygml_version="1.0"),
c AS (SELECT id FROM citydb.schema WHERE
xml_namespace_uri="http://schemas.opengis.net/citygml/relief/1.0")

INSERT INTO citydb.schema_referencing (referenced_id, referencing_id)
SELECT c.id,ade.id FROM ade, c;
```

- `SCHEMA_TO_OBJECTCLASS` table *(citydb.schema_ to_ objectclass)*: This table is also an associate table with two foreign key columns *(schema_ id, objectclass_ id)* to link the schema with the classes.

```
-- Add entry into table SCHEMA_TO_OBJECTCLASS
WITH r AS (SELECT id FROM citydb.schema WHERE
(xml_namespace_uri="https://3d.bk.tudelft.nl/schemas/CityGML_iTINs_ADE"
AND citygml_version="2.0")
OR
(xml_namespace_uri="https://3d.bk.tudelft.nl/schemas/CityGML_iTINs_ADE"
AND citygml_version="2.0")
),
s AS (SELECT id FROM citydb.objectclass WHERE id BETWEEN 140 AND 145
ORDER BY id)

INSERT INTO citydb.schema_to_objectclass (schema_id,objectclass_id)
SELECT r.id, s.id FROM s,r ORDER BY s.id;
```

4.3.2 Mapping of ADE classes to tables

The name of the tables are identical to the class name with the addition of the prefix *itin_* . The CityGML iTINs ADE class `ITINRelief` is mapped to the table *citydb.itin_ itinrelief*. The attributes of the class are mapped to the columns of the table. Similarly, tables are created for the classes: `iTriangluatedSurface` *(citydb.itin_ itriangulatedsurfaces)*, `iStars` (*citydb.itin_ istars)*, and `iTriStrips` *(citydb.itin_ itristrip)*.

In addition, to store the vertices of the TIN, a *citydb.itin_ipoints* table is created. The coordinates (x, y, z) of the vertices are stored in three separate columns in a separate table as 64-bit double precision with an ID (64-bit integers) (see snippet below). To store the indexed triangles, an *citydb.itin_itriangles* table is created based on the `iTriangle` class which stores references to the IDs of the vertices of each triangle. To store the vertex IDs and triangle IDs, 64-bit large range integers i.e. *bigint* are used because 32-bit integers limit the dataset to $2^{32}/2$ (i.e. around 2 billion) IDs. For the storage of star of a vertex, a variable length integer array is used because the link of a vertex can contain 2 to an infinite number of vertices. Similarly, to store the IDs of the triangles in each triangle strip and triangulated surface, a variable length integer array is used. A B-tree is used for indexing the tables.

The 3DCityDB is shipped with a set of stored procedures referred to as the CITYDB package *(citydb_pkg)*, which are automatically installed during the setup procedure of 3DCityDB. The stored procedures, i.e. select, insert, update and delete, for the new tables are created and added to the *citydb_pkg*.

```
-- Add the new relief cityobject to citydb.itin_itnrelief
INSERT INTO citydb.itin_itinrelief
(id, gml_id, name, lod, relief_geometry, relief_geometry_type,
objectclass_id, cityobject_id, relief_parent_id)
VALUES
(1,"GML_6bb30328-7599-4500-90ef-766fde6aa67b_iTIN", "iTIN model" ,
 1, "its001", 'iTriangulatedSurface',140, 1,1 );

-- Add points to the citydb.itin_ipoints
INSERT INTO citydb.itin_ipoints (id, x, y, z)
VALUES
(1, 458868.0, 5438362.0, 112.0),
(2, 458875.0, 5438355.0, 112.0),
(3, 458883.0, 5438362.0, 114.0);

-- Add triangle to the citydb.itin_itriangles
INSERT INTO citydb.itin_itriangles (id, itriangle_vertex_ids)
VALUES
(1, ARRAY[1,2,3]),
(2, ARRAY[1,3,5]),
(3, ARRAY[2,3,4]);

-- Add triangles to the citydb.itin_itriangulatedsurface
INSERT INTO citydb.itin_itriangulatedsurface
(id, its_id, its_triangles)
VALUES
(1,'its001', ARRAY[1,2,3,5,6,7]);

-- Add stars to the citydb.itin_istars
INSERT INTO citydb.itin_istars
(id, point_id, istar)
VALUES
```

```
(1,1, ARRAY[2 3 4 5 49 4 3 49 51 2 5 5]);

##correct this one
-- Add triangle strips to the citydb.itin_itristrips
INSERT INTO citydb.itin_itristrips
(id, itstrip_id, itristrip)
VALUES
(1,'itstrip001', ARRAY[1,2,3,4,5]);
```

4.4 Experiments with real world datasets

4.4.1 Datasets and tools used

Construction of TIN. Real-world datasets were used to test the approach for efficiently storing massive TIN terrains in a database. Two filtered subsets and a complete tile of the massive AHN3 point cloud dataset (Tile# 37EN/1) [2] are utilised as input for the construction of the TIN. The TIN is generated based on the concept of spatial streaming [97]. These modules exploit the spatial coherence of the massive point clouds to compute a streaming triangulation [97]. The TIN is created and stored without any attributes and constraints, which are possible to store. The details of the input datasets are given in Table 4.1. The tests are performed on a MacBook with 2.2 GHz Intel Core i7 and 16GB RAM with 3DCityDB version 3.3.1 and PostgreSQL version 9.5.

Dataset	#Points	#Triangles
AHN3 Subset 1	40 506 390	81 012 707
AHN3 Subset 2	220 506 390	442 022 687
AHN3 Tile	508 564 458	1 117 129 823

Table 4.1: Details of the input datasets showing the number of triangles in each input dataset

4.4.2 Populating the database

A Python prototype[2] was developed which converts the input TIN terrain datasets into the *indexed triangles*, *triangle strips*, and *stars* to populate in the newly implemented database tables. The prototype also converts the datasets to the Simple Feature structure which is simply a list of polygons and is already supported in 3DCityDB/PostgreSQL, The python prototype also provides to export the data from the database in the CityGML iTINs ADE format.

[2] https://github.com/tudelft3d/CityGML_iTINs_ADE

The TIN construction pipeline makes use of `spfinalize` and `spdelaunay2d` modules of the blast extension of Lastools. These modules exploit the spatial coherence of the massive point clouds to compute a streaming triangulation [97]. The runtime (in minutes) for the construction (and loading), and indexing of TINs in the database for *Simple Features*, *indexed triangles*, *triangle strips*, and *stars* are shown in Table 4.2. Populating the database and indexing the datasets takes a significant larger amount of time for triangles in *Simple Features*. The minimal time is taken by the *star* data structure i.e. 4 hours, followed by the *indexed triangles*. This is around 4 times less than that of the *Simple Features*. This is due to the construction of complex PostGIS GiST index which took around 14 times more time than indexing stars with a B-tree.

From Table 4.3, it is clear that GiST is larger than B-tree as it takes the maximum disk space i.e. 58.2 GB for a TIN generated from one tile of AHN3. Stars take the least space of about 13.8 GB for storing the index which is a B-tree instead of a spatial index. Storing TINs in *Simple Features* is memory expensive because of two reasons (Table 4.4): first, there are 2x triangles than vertices. Second, the bounding box of every triangle is explicitly stored.

Dataset/ Type	AHN3 S1			AHN3 S2			AHN3 Tile		
	Const.	Index	Total	Const.	Index	Total	Const.	Index	Total
SF	24.93	24.72	49.65	174.5	183.8	358.3	422.8	513.6	936
iTriangle	23.8	1.3	25.1	118.5	11.8	130.3	287.4	34	321
TriStrip	38.2	2.1	40.3	197.9	19.7	217.6	604.2	44.2	648.4
Star	13.3	2.8	16.1	73.2	15.6	88.8	212	35.8	247.8

Table 4.2: Runtime (in minutes) for the construction (and loading), and indexing of TINs in the database

Dataset/Type	AHN3 Subset 1	AHN3 Subset 2	AHN3 Tile
SF	4.2 GB	25.8 GB	58.2 GB
iTriangles	1.7 GB	13.3 GB	29.5 GB
TriStrips	2.4 GB	14.6 GB	33.2 GB
Stars	868 MB	4.3 GB	13.8 GB

Table 4.3: Size of the index in a database

Dataset/Type	AHN3 Subset 1	AHN3 Subset 2	AHN3 Tile
SF	13 GB	61.4 GB	143.5 GB
iTriangles	5.7 GB	22.9 GB	53.8 GB
TriStrips	3.1 GB	17.5 GB	47.3 GB
Stars	4.3 GB	26.4 GB	63.2 GB

Table 4.4: Size of the tables in a database

Dataset	Convex hull	Avg degree	Max degree
AHN3 Subset 1	29	5.99 (6)	321
AHN3 Subset 2	1168	5.99 (6)	238
AHN3 Tile	657	5.99 (6)	418

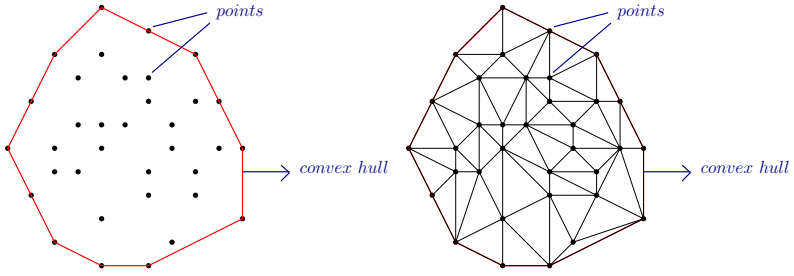
Table 4.5: Convex hull, average degree and maximum degree of a vertex in the input datasets

4.4.3 Functions implemented on the TIN terrain datasets

Apart from the import and export functionality, I also implement some functions for the analysis of TIN. Atomic functions, such as calculating slope and aspect, only exist in the raster implementation of PostGIS. These functions are most notably used in hydrology related applications. I implemented functions to calculate the slope and aspect of the triangles of a TIN. By knowing the slope and aspect of triangles of a TIN, the direction the water will runoff to can be determined for each triangle.

Other implemented functions include calculating local minima and maxima. I implemented local minima and local maximum as boolean functions which compare compare the height of a selected vertex to the height of the vertices in a triangulated surface (in case of *iTriangulateSurface*), in a single triangle strip (in case of *iTriStrips*), in its star (in case of *iStars*). Deriving the local minima are useful to define the *sinks (or depressions)* in the terrain and watershed modelling. All triangles from which the water reaches the same local minima is called a watershed.

Analysis of TIN also requires information about the neighbouring points and triangles of a vertex. It is known as the *degree* of a vertex which is expressed as the number of adjacent vertices from which edges are incident on that vertex. I implemented a function that determine the average and maximum degree of a vertex of the TIN stored in the database (see Table 4.5). The degree of a vertex provides information about the complexity of the TIN at that specific vertex.



(a) Convex hull of a set of points

(b) Convex hull of a Delaunay triangulation

Figure 4.3: Convex hull of a TIN

I also implemented a function that determine the convex hull of the triangulation stored in the database (see Table 4.5). A *convex hull* of a set of vertices is the smallest convex polygon which contains all the vertices of the given set (Figure 4.3) [251]. In a triangulation (or a TIN), the boundary or the polygon enclosing the triangulation of a set of vertices is the same as the convex hull of vertices alone.

The limitation of *indexed triangle* representation is that it lacks information about the adjacency and incidence relationships. Similarly, a single TIN can also be composed of a number of disconnected triangle strips storing the triangles (Figure 3.9). This means there is local topological connectivity within the individual triangle strips but no overall connectivity for the entire TIN. One solution can be to explicitly store this information in the database. I implemented a simple function that determines the neighbours of every triangle in a TIN (Figure 4.4). The function checks for the shared edges between the triangles to output the IDs of the neighbours of a triangle. This is useful to explicitly determine the connectivity within an entire TIN.

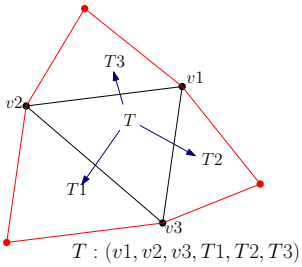


Figure 4.4: Neighbouring triangles $(T1, T2, T3)$ of a triangle T in a TIN

4.5 Conclusion

In this chapter, I have presented the database implementation of the CityGML iTINs ADE presented in Chapter 3 for compactly storing massive TIN terrains. I have reviewed the current practice of handling TINs in existing database solutions and explored how the new TIN data structures can be integrated in 3DCityDB (PostgreSQL/PostGIS).

Classes from the ADE were mapped to the tables in the database. Basic stored procedures i.e. select, insert, update and delete, for the new tables were created and added to 3DCityDB. A prototype was developed that loads the ADE datasets in the database. The tests conducted supplement our previous conclusion that the Simple Feature structure in 3DCityDB/PostGIS is not efficient for the storing massive TINs as far as memory usage, loading and indexing times, and topology is concerned.

The TIN is created and stored without any attributes and constraints, which are possible to store. Any semantic attributes can be attached to vertices, individual *itriangles*, *iTriangulatedSurface*, *iTriStrips*, and *iStars* as a list (i.e. as `ARRAY` type). The constrained edges can be stored in a separate table (say *itin_constraints*) with reference to the vertices stored in *itin_ipoints* table. Storing triangles as references to their vertices is quick, but it suffers from lack of topological information. This is not the case with Stars. The developed prototype allows to determine the neighbours of a triangle in a TIN. It also provides the functionality to export the data from the database into the CityGML iTINs ADE format.

Chapter 5

An improved LOD framework for terrains in 3D city models

This chapter is based on the paper:

Kumar K, Labetski A, Ledoux H, Stoter J. 2019. An improved LOD framework for the terrains in 3D city models. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, IV-4/W8, pp.75-82.
doi: https://doi.org/10.5194/isprs-annals-IV-4-W8-75-2019

5.1 Introduction

In Chapter 3, we discussed several problems associated with storing massive terrains in CityGML. I presented in Chapter 3 an improved representation for compactly storing massive terrains in CityGML files. In Chapter 4, I discuss a database solution for the efficient storage and management of massive CityGML TIN terrain datasets. The problem that I discuss in this chapter is the lack of specifications for modelling the LODs of a terrain in CityGML.

CityGML defines 5 LODs and supports multiple representations of the same city object in different LODs simultaneously [185]. Unlike computer graphics where point density (number of points per m^2), number of pixels, distance from the camera, etc. are used, the concept of LODs in CityGML is driven by both semantics and geometry [21]. These 5 LODs have been widely adopted by practitioners and stakeholders in 3D city modelling.

The CityGML LOD concept is well defined for buildings, bridges, tunnels, and to some extent for roads [10, 144]. However, there is no clear definition of LODs for terrain, vegetation, land use, water bodies, and generic city objects in CityGML [11, 134, 155]. Despite the popularity and general acceptance of the concept by the practitioners and stakeholders in 3D city modelling, there are still some limitations. There are many open issues in the currently accepted concept. There is no distinction between the different LODs of a terrain at the geometric and semantic level [134] (Figure 5.1). Further, it is not clear what an LOD4 representation is for features representing vegetation, land use, or water bodies [11], given that LOD4 models the indoors which is for buildings, bridges, and tunnels. In addition, CityGML 2.0 specifications state for the terrains that *"for a LOD3 scene it might be sufficient to use a regular grid in LOD2 with certain higher precision areas defined by ReliefComponents in LOD3"* [185]. It is not clear what a *"regular grid in LOD2"* is. What are those *"certain higher precision areas in LOD3"*, and how are they defined? Using the same LOD for a terrain for different applications is not justified as different applications require different features in different details. The terrain LODs defined in CityGML define a DTM (Digital Terrain Model), i.e. a 2.5D model, which is sufficient for visualisation, but for not for other applications.

Extensive research has been done to refine the LOD concept in CityGML. Currently, it is mostly concerned with buildings [21, 155, 236]. In this chapter, we focus on the terrain of a 3D city model and propose a framework for modelling terrains at different LODs in CityGML. This proposal is based on the TIN representation of the terrains (i.e. *TINRelief* in CityGML). We investigate the current status of the LOD concept for terrains in practice, and identify the shortcomings of the current concept for the terrain LODs in CityGML. As a proof of concept of

the framework, we implemented a software prototype to generate terrain models with integrated city features (e.g. buildings) at different LODs in CityGML.

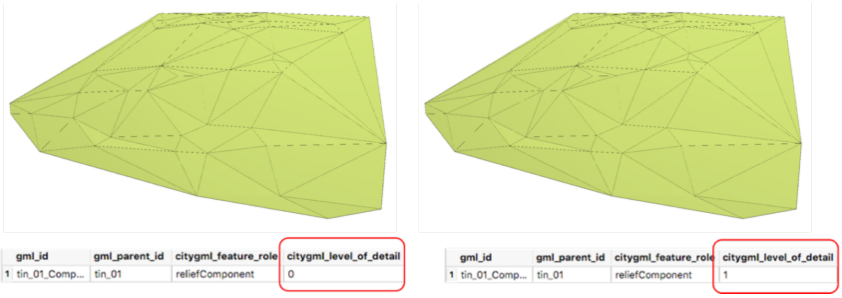


Figure 5.1: Two CityGML terrains (*TINRelief*) at different LODs, both perfectly valid according to the CityGML specifications. No distinction between the different LODs of a terrain at geometric and semantic level can be observed.

5.2 Background

5.2.1 LOD modelling for terrains in practice

The concept of LOD is an important characteristic of a 3D city model [21]. It refers to the capability of modelling multiple representations of a spatial object at different levels of data quality and complexity for different applications [48]. The concept of LODs originates from the realm of computer graphics where the focus is on balancing between complexity and performance by regulating the amount of detail utilised to represent a virtual world [157]. Terrain, specifically, has been examined extensively in the computer graphics domain, mainly in the context of view-dependent level-of-detail control [90, 153]. Some approaches focus on pre-computed levels of detail for rendering but many also focus on real-time generation based on statistics [253]. There are few definitions for the LODs and it is strictly dictated by geometry. Lindstrom et al. [153] define consecutive LODs by removing every other column and row of the next higher LOD. Hoppe [90] focuses instead on building a hierarchy based on edge collapsing and recording their inverses.

LOD in GIS is often linked to the cartographic term *scale*. Scale is often understood in different ways, ranging from *cartographic scale* denoting more detailed information on a map, *geographic scale* denoting the spatial extent of an area, *resolution* denoting the size of the smallest distinguishable part of a spatial data set, and *operational scale* which denotes the scale at which a phenomenon

operates [15]. Scale became an essential parameter of geospatial datasets and influenced the acquiring, handling, storing, and processing of the data [82]. At the same time, while there have been many studies conducted around the effect of scale change, or spatial resolution, on analysis, there has not been a formalised approach to defining a unified approach to scale or LOD in the GIS realm [82]. Furthermore, Biljecki [16] explains that while there is an association between the terms scale and LOD, with the transition from paper maps the term scale is losing its meaning, and therefore using scale in 3D city modelling should be avoided. Beyond the concept of scale, there are many sources equating LOD to *spatial resolution* in the 2D realm [32, 248], but this is also lacking a formalised definition.

5.2.2 Terrain LODs in CityGML

The *Relief* module in CityGML allows for the representation of the terrains as TINs (`TINRelief`), mass points (`MasspointRelief`), break lines (`Breakline-Relief`), or grids (`RasterRelief`). It is also possible to represent a terrain with a combination of different terrain types within a single dataset. For instance, a terrain can be modelled by a coarse grid with some areas depicted by detailed TIN or as a TIN with break lines to depict a constrained triangulation, etc. A terrain is modelled as a `ReliefFeature` which consists of one or more entities of the class `ReliefComponent`, which can be a TIN or a grid and so on. Both `ReliefFeature` and `ReliefComponent(s)` have an `dem:lod` attribute for the level of detail [185]. The LOD of a `ReliefFeature` can differ from the LOD of its `ReliefComponents` [185].

CityGML also supports 5 LODs for the terrains. However, there are no guidelines provided that differentiates between these 5 LODs at geometry and semantic level [134]. For instance, in Figure 5.1 the geometry (e.g. number of triangles) of the terrain (`TINRelief`) remains the same though the LOD changes from 0 to 1; the number of points/triangles in every LOD or any other criteria to differentiate between the terrain LODs is not prescribed. The standard has guidance for the LODs of certain modules, such as buildings, bridges, and tunnels.

There is also the concept of the Terrain Intersection Curve (TIC) in CityGML. A TIC is used to integrate 3D objects, such as buildings with the terrain model. It stores the exact position where a terrain intersects with the 3D objects to avoid the 3D objects float over or sink into the terrain. This is particularly the case if terrains and 3D objects in different LOD are combined, or if they come from different providers [185]. However, TICs are seldom used in practice, out of 31 sources of open 3D city model datasets only 1 source used TICs[1]. Further, the attribute *RelativeToTerrainType* only gives a qualitative reference

[1] https://3d.bk.tudelft.nl/opendata/opencities/

to the position of a city object with respect to the terrain *(+entirelyAboveTerrain, +substantiallyAboveTerrain, +substantiallyAboveAndBelowTerrain, +substantiallyBelowTerrain, +entirelyBelowTerrain)* and not a quantitative measure.

5.3 Our proposal for modelling terrain at different LODs in CityGML

LODs of 3D city models do not differ only by the amount of geometric data, and visual properties, but also they may differ in terms of their semantic information. One geometry based solution to differentiate between TIN terrains at different LODs can be techniques used in computer graphics to restrict the point/triangle density (number of points/triangles per m2) required for each LOD, while staying close to the original shape of the terrain. A simplified TIN will have just enough vertices/triangles to model the terrain as per the required level. However, deciding the number of points/triangles for every dataset does not seem feasible. While the number of primitives generally gives a good impression about the geometric complexity of a 3D city model, it cannot be considered as an unambiguous differentiator as is the case in computer graphics. Even in the realm of computer graphics there is no clear consensus about what constitutes a specific LOD.

In our proposal, we focus on modelling terrains with respect to the geometry and semantics of the terrain and the integration of terrain with the other city objects present in the city model. The proposal is based on the different TIN representations for modelling terrains considered in Chapter 3 (see Figure 3.2) Kumar et al. [136]. Each succeeding LOD in our framework contains more detail and complexity than the preceding LOD [21, 185]. We use the CityGML *Generics* module to introduce all the new attributes in our LODs.

5.3.1 LOD0

The terrain in CityGML is a DTM (Digital Terrain Model) and not a DSM (Digital Surface Model). LOD0 is the coarsest and most generalised representation of city objects in CityGML. For terrains (*TINRelief*), we left LOD0 as a *strict 2.5D TIN representation* (without vertical surfaces and overhangs) i.e. a simple Delaunay triangulation of the ground without man-made objects and vegetation embedded in the TIN (see Figure 5.2). This ensures that an LOD0 TIN can be readily converted and used in all GIS packages which often assume that a terrain is 2.5D. The terrain is a 2-manifold surface. We also introduced a semantic attribute `numberOfTriangles` to store the number of triangles in the TIN (see Snippet 1).

Thus, LOD0 = a strict 2.5D TIN.

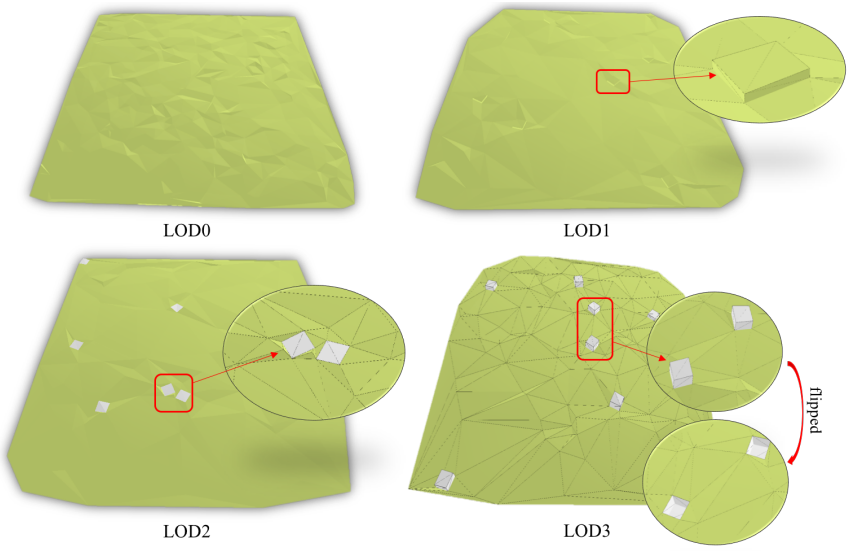


Figure 5.2: Terrain models at different LODs (0-3) according to our framework. (Source: Generated by our *Random3DTerrain* software prototype.)

```
<dem:ReliefFeature gml:id="relief_feature_01">
  <gml:name>Relief Feature</gml:name>
  <dem:lod>0</dem:lod>
  <dem:reliefComponent>
    <dem:TINRelief gml:id="tin_relief_01">
      <gml:name>TIN model</gml:name>
      <gen:intAttribute name="numberOfTriangles">
        <gen:value>89</gen:value>
      </gen:intAttribute>
      <dem:lod>0</dem:lod>
      <dem:tin>
        <gml:TriangulatedSurface>
         ....
        </gml:TriangulatedSurface>
      </dem:tin>
    </dem:TINRelief>
  </dem:reliefComponent>
</dem:ReliefFeature>
```

Snippet 1: Excerpt of the generated LOD0 terrain model in CityGML

5.3.2 LOD1

The ISO19107:2003 Spatial Schema [100] standard defines *GM_ TIN* geometry type for representing TIN models, which in theory should allow vertical triangles (surfaces) in a TIN (i.e. more than 2.5D) [136]. 3DTOP10NL terrain (TIN) is one such example dataset which has vertical walls represented as triangles [117]. Modelling it in 2.5D will result in the loss of triangles representing the vertical walls. Furthermore, a 2.5D model does not allow for overhangs, such as cliffs, naturally-formed arches and caves present in the terrain.

Therefore, we model an LOD1 terrain as *an extension to the 2.5D DTM to support the representation of vertical triangles and overhangs in the TIN i.e. a 2.5D+/2.75D model* (see Figure 5.2). The terrain is still a 2-manifold surface and the GIS software can use and edit it. We introduced an attribute `vertTrianglesID` to store the list of the IDs of these vertical triangles in the model as there is no mechanism in CityGML to flag the vertical triangles. This is important because when a model with vertical triangles is projected on a 2D surface, the vertical surfaces flatten out which distorts the geometry of the model [134]. Flagging these vertical triangles allows their removal while transforming from 3D to 2D/2.5D. Similarly, we also introduced an attribute `ovTrianglesID` to store the list of the IDs of the triangles representing the overhangs in the model. We also introduced an attribute `numberOfTriangles` to store the number of triangles in the TIN.

Thus, LOD1 = LOD0 + information about the vertical triangles and overhangs in the TIN.

```
<dem:ReliefFeature gml:id="relief_feature_01">
  <gml:name>Relief Feature</gml:name>
  <dem:lod>1</dem:lod>
  <dem:reliefComponent>
    <dem:TINRelief gml:id="tin_relief_01">
      <gml:name>TIN model</gml:name>
      <gen:intAttribute name="numberOfTriangles">
        <gen:value>89</gen:value>
      </gen:intAttribute>
      <gen:stringAttribute name="vertTrianglesID">
        <gen:value>vt1 vt2 vt3 ...</gen:value>
      </gen:stringAttribute>
      <gen:stringAttribute name="ovTrianglesID">
        <gen:value>ot1 ot2 ot3 ...</gen:value>
      </gen:stringAttribute>
      <dem:lod>1</dem:lod>
      <dem:tin>
        <gml:TriangulatedSurface>
         ....
        </gml:TriangulatedSurface>
      </dem:tin>
```

```
    </dem:TINRelief>
  </dem:reliefComponent>
</dem:ReliefFeature>
```

Snippet 2: Excerpt of the generated LOD1 terrain model in CityGML

5.3.3 LOD2

LOD0 and LOD1 terrain models can be useful in applications, such as hydrological flow modelling, natural hazard modelling, geomorphological mapping, and relief maps. However, they cannot be used in applications where information about the location of city objects with respect to the terrain is required, e.g. to determine the effect of surface features, such as buildings and vegetation on visibility analysis and viewshed calculations [123], hydrological modelling in urban environments to identify the flood risk [83], etc.

Therefore, we model an LOD2 terrain as *a semantically enriched strict 2.5D DTM* with information about the city objects integrated in the terrain. For this, we define an LOD2 terrain as a constrained Delaunay triangulation where the boundaries of the city objects, such as buildings, roads, etc. act as constraints in the triangulation (see Figure 5.2). We introduced an attribute to store the extent/boundary (`extent`) of the triangles representing the footprints of city objects in the terrain. Further attributes are added to store information about the type of city object (`cityObjectType`) and the ID of the city object (`cityObjectID`) represented by these triangles (see Snippet 3). We also introduced an attribute (`numberOfTriangles`) to store the number of triangles in the TIN.

Thus, LOD2 = LOD0 + semantic information about the city objects integrated in the terrain.

The same LOD2 terrain can integrate with both, the LOD0 building footprints and the LOD1 block model of the buildings because the building footprints remain the same for LOD0 and LOD1 buildings. It can also fit with higher LOD models of buildings provided their footprints (ground surface) remain the same. If their footprint change, then re-computation of the TIN is required.

```
<dem:ReliefFeature gml:id="relief_feature_01">
  <gml:name>Relief Feature</gml:name>
  <dem:lod>2</dem:lod>
  <dem:reliefComponent>
    <dem:TINRelief gml:id="tin_relief_01">
      <gml:name>TIN model</gml:name>
      <gen:intAttribute name="numberOfTriangles">
        <gen:value>89</gen:value>
      </gen:intAttribute>
      <gen:genericAttributeSet name="Fprint_b01">
```

```
        <gen:stringAttribute name="cityObjectType">
          <gen:value>Building</gen:value>
        </gen:stringAttribute>
        <gen:stringAttribute name="cityObjectID">
          <gen:value>b01</gen:value>
        </gen:stringAttribute>
        <gen:stringAttribute name="extent">
          <gen:value>155067.7614 466489.4299 21.27
          ....</gen:value>
        </gen:stringAttribute>
      </gen:genericAttributeSet>
      <gen:genericAttributeSet name="Fprint_b02">
        ....
      </gen:genericAttributeSet>
      <dem:lod>2</dem:lod>
      <dem:tin>
        <gml:TriangulatedSurface>
         ....
        </gml:TriangulatedSurface>
      </dem:tin>
    </dem:TINRelief>
  </dem:reliefComponent>
</dem:ReliefFeature>
```

Snippet 3: Excerpt of the generated LOD2 terrain model in CityGML

5.3.4 LOD3

Lastly, we model an LOD3 terrain as *a semantically enriched 2.5D+/2.75D extended DTM* with information about the city objects integrated in the terrain (see Snippet 4 and Figure 5.2). In short,

LOD3 = LOD1 + semantic information about the city objects integrated in the terrain.

We introduced two attribute `vertTrianglesID` and `ovTrianglesID` to store the list of the IDs of these vertical triangles and triangles representing the overhangs in the model. We also introduced an attribute `numberOfTriangles` to store the number of triangles in the TIN. Other attributes are added to store the type of city object (`cityObjectType`) and the ID of the city object (`cityObjectID`) present in the dataset.

```
<dem:ReliefFeature gml:id="relief_feature_01">
  <gml:name>Relief Feature</gml:name>
  <dem:lod>3</dem:lod>
  <dem:reliefComponent>
    <dem:TINRelief gml:id="tin_relief_01">
      <gml:name>TIN model</gml:name>
      <gen:intAttribute name="numberOfTriangles">
```

```
      <gen:value>89</gen:value>
    </gen:intAttribute>
    <gen:stringAttribute name="vertTrianglesID">
      <gen:value>vt1 vt2 vt3 ...</gen:value>
    </gen:stringAttribute>
    <gen:stringAttribute name="ovTrianglesID">
      <gen:value>ot1 ot2 ot3 ...</gen:value>
    </gen:stringAttribute>
    <gen:genericAttributeSet name="Fprint_b01">
      <gen:stringAttribute name="cityObjectType">
        <gen:value>Building</gen:value>
      </gen:stringAttribute>
      <gen:stringAttribute name="cityObjectID">
        <gen:value>b01</gen:value>
      </gen:stringAttribute>
      <gen:stringAttribute name="extent">
        <gen:value>155067.7614 466489.4299 21.27
        ....</gen:value>
      </gen:stringAttribute>
    </gen:genericAttributeSet>
    <gen:genericAttributeSet name="Fprint_b02">
      ....
    </gen:genericAttributeSet>
    <dem:lod>3</dem:lod>
    <dem:tin>
      <gml:TriangulatedSurface>
       ....
      </gml:TriangulatedSurface>
    </dem:tin>
  </dem:TINRelief>
 </dem:reliefComponent>
</dem:ReliefFeature>
```

Snippet 4: Excerpt of the generated LOD3 terrain model in CityGML

5.3.5 LOD4 = removed

We do not define an LOD4 representation for the terrains for the following reasons:

1. LOD4, in general, models the interior of city objects, such as buildings, tunnels, bridges, etc., this does not make sense in relation to terrains.

2. CityGML 3.0, the upcoming version of CityGML, plans to phase out the LOD4 representation of features [141, 156].

#	LOD	Description
1	LOD0	LOD0 = a strict 2.5D TIN representation.
2	LOD1	LOD1 = LOD0 + information about the vertical triangles and overhangs in the TIN.
3	LOD2	LOD2 = LOD0 + information about the city objects integrated in the terrain.
4	LOD3	LOD3 = LOD1 + information about the city objects integrated in the terrain.

Table 5.1: Summary of the proposed terrain LODs

5.4 Implementation

In order to test our proposed framework and show its usability, we developed a software prototype, *Random3DTIN* which generates artificial TIN terrain models at different LODs (0-3) in CityGML format (as shown in Figure 5.2). Our prototype is based on the procedural modelling engine *Random3DCity*[2] developed by Biljecki et al. [22] for generating random CityGML buildings in multiple LODs.

The new attributes for the terrain, e.g. number of triangles, ID and type of the city object, etc. are introduced as *Generic* attributes in the generated CityGML datasets (see Snippets 1, 2, 3 and 4). The software we developed, together with the sample datasets shown in Figure 5.2, is freely available in our GitHub repository: `https://github.com/tudelft3d/Random3DTIN`.

5.5 Conclusion

In Chapter 3, we outlined the problems in storing massive terrains as TINs in CityGML. One of the stated problems was the lack of a precise definition of each LOD for features such as terrain. The concept of Level of Detail in CityGML is widely used by practitioners and stakeholders in the field of 3D city modelling for representing city features with varying degrees of complexity in the geometry and semantics as per the need of a specific application. There is currently no distinction between the different LODs of a terrain/relief at the geometric and semantic level in CityGML. In this chapter, we presented our framework for modelling terrains at different LODs in CityGML. The framework that we propose is simple and compliant with the existing LOD concept in CityGML and is meant to improve the ambiguity of the current concept. It also makes

[2] `https://github.com/tudelft3d/Random3Dcity`

an explicit distinction between 2.5D and more complex representations of terrain, given that many GIS software only support 2.5D. Therefore, being able to enforce 2.5D in LOD0 and LOD3 is useful.

The methodology does not restrict the LODs to the geometric data granularity in values, such as the number of points/triangles; these values are often arbitrary or application/user specific. Rather, our approach aims to integrate the terrain with surrounding features while adding further geometric and semantic information. As a proof of concept, we also developed a software prototype (*Random3DTIN*) to generate CityGML terrain datasets with other integrated city features, such as buildings, based on our framework. The prototype is open source and a set of sample datasets is available for free, for public use, so that other researchers can benefit from the different LOD terrain datasets in their application domains. The prototype generates artificial terrain models suited for applications where having real world data is not important such as in the case of testing simulations or analysis to determine which LOD is best suited for the process. The open source 3D city modelling software *3dfier*[3] generates real world terrain models with vertical walls, overhangs, etc.The output from 3dfier can easily be adjusted to represent these terrain models according to our proposed LOD framework.

Our methodology also maintains the modular approach of CityGML by not linking the LODs of the terrain directly to the LODs of the city objects present in a dataset. This means that an LOD2 terrain is not tied to the LOD2 building model or any other LOD2 city objects. Any of the terrain LODs (i.e. LOD0, LOD1, LOD2 and LOD3) can also exist with LOD0, LOD1 or higher LOD city objects. Further, our methodology allows for the storage of the triangles and the triangulation constraints explicitly in the data structure so that same triangulation is enforced.

Compared with the aforementioned *Terrain Intersection Curve* currently present in CityGML, our proposal has several advantages. First, the TIC was designed to assist with the integration of city objects with their surrounding terrain but it is currently not applicable to all city features and only focuses on buildings and building parts, bridge, bridge parts and bridge construction elements, tunnel and tunnel parts, city furniture objects, and generic city objects [185]. This excludes transportation objects such as roads and railways, vegetation objects and water bodies, whereas our framework covers all the city objects integrated in the terrain. The knowledge of where the road interacts with the terrain can aid users in calculating more accurate calculations of road inclination. Second, the TIC only defines the geometry of the intersection and has no other semantic information that can be stored, whereas we introduced attributes with semantic information about such intersections in the TIN. 3D road networks can be utilised for optimising routing network for waste collection and transportation [237].

[3] https://github.com/tudelft3d/3dfier

Understanding the affect that road inclination and vehicle weight have can aid in optimising for minimum fuel consumption which can result in lower costs than traditional shortest route approaches [237]. Our proposal for LOD2 would enable such calculations to be done with CityGML datasets. Last, a TIC is only relevant in context with a terrain, therefore it makes more sense to store intersection information with the terrain and not individual city features.

In addition, it is necessary to examine the proposed LOD concept for the other terrain representation types. It is not reasonable to have an LOD definition that would be applicable to the entire *Relief* module and therefore there needs to be an investigation into the needs of all representation types separately. It is also possible to implement this terrain LOD framework as a CityGML ADE, both UML and XSD.

Part III

Harmonising 3D standards for the built environment

Chapter 6

The LandInfra standard and its role in solving the BIM-GIS quagmire

This chapter is based on the paper:

Kumar K, Labetski A, Ohori K A, Ledoux H, and Stoter J, 2019. The LandInfra standard and its role in solving the BIM-GIS quagmire. *Open Geospatial Data, Software and Standards*, 4(1), pp.1-16.
doi: https://doi.org/10.1186/s40965-019-0065-z

6.1 Introduction

A large number of 3D city models exists in a variety of standards and formats. Practitioners dealing with three-dimensional geoinformation about cities and infrastructure often struggle while shuttling 3D models back and forth between these different standards and formats. This forces users to convert between formats, often losing information, and sometimes even having to manually recreate whole datasets for use in different applications [258]. The BIM and 3D GIS domains are often faced with the data interoperability issues when converting 3D city models between the IFC and the CityGML standard, the two popular standards in the BIM and 3D GIS domains [3, 154, 229].

Much work has been done on the integration of IFC and CityGML [3, 52, 64, 70, 72, 98, 175]. In addition, new standards have been recently developed that integrate concepts from different standards to represent an integrated semantic 3D city model. One such standard is the OGC LandInfra (and InfraGML) which integrates concepts from CAD (Computer Aided Design), BIM, and GIS, and has overlaps with CityGML and IFC (see Section 2.2 for an overview of LandInfra and InfraGML). Despite several attempts, these standards remain disconnected, owing primarily to the differences in the underlying modelling approaches with respect to geometry, semantics, schema, level of detail, etc. [3, 70]; a situation that is referred to as the *BIM-GIS quagmire* [80].

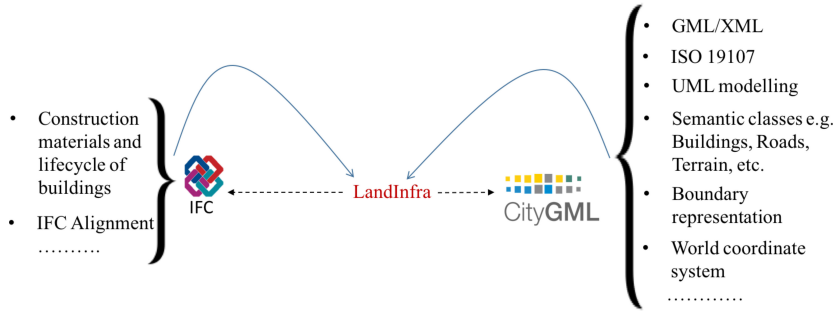


Figure 6.1: LandInfra a connecting bridge between IFC and CityGML, but is conceptually, semantically, and geometrically closer to CityGML.

In this chapter, we focus on the LandInfra standard which was designed as a 'connecting bridge' between the BIM and GIS domains (Figure 6.1). Despite the fact that LandInfra has the potential to bring the architectural, BIM, and

geospatial views onto a common footing, the standard is not well known in the BIM or GIS communities, and its applicability to the built environment applications has barely been explored. We investigate in this chapter the LandInfra standard and its potential role in solving the BIM-GIS quagmire in more detail. In order to meet this goal, we: (i) provide a review of LandInfra, its characteristics and its relation to the main open 3D GIS (CityGML) and BIM (IFC) standards; (ii) summarise what has been written in the academic literature about LandInfra and how it is used (or not) in practice; and (iii) discuss some minor issues in the data model of LandInfra, which we found through the analysis of the standard.

6.2 LandInfra (and InfraGML) in theory and practice

OGC LandInfra [188] standard was proposed as the successor to LandXML [145]. OGC InfraGML is the GML based encoding of the LandInfra data model. Section 2.2 provides an overview of LandInfra and InfraGML. LandInfra is a relatively young standard and at present it is difficult to identify any concrete examples of its usage in practice; the majority of scientific articles that mention LandInfra only describe the need to consider it in future work.

There are many papers discussing the relationship between LandInfra and the ISO 19152 Land Administration Domain Model (LADM), see for instance Cagdas et al. [39], Kalogianni et al. [118], Kara et al. [119], Lemmen et al. [151], OGC [199], Stubkjær et al. [235], van Oosterom et al. [244]. In these papers, InfraGML is cited, alongside other models such as CityGML and LandXML, in relation to harvesting the existing 3D data that is collected to open up the possibility of creating a 3D cadastral database.

There are several papers discussing the integration of LandInfra in specific use cases. Kara et al. [119] assessed the use of several different models to provide a valuation information model for property taxes. Pouliot et al. [210] compared schema matching between user needs and three geospatial standards (the CityGML UtilityNetwork ADE, InfraGML and IFC) in relation to underground utility network modelling. They were not able to come to a definitive conclusion due to contradictory results based on differences in schema matching techniques and the variation between the various levels and the number of elements when comparing one schema to another. Rajabifard et al. [213] assessed LandInfra, along with other 3D spatial information models, in terms of their capability for modelling legal interests and legal boundaries as defined in the Victorian jurisdiction in Australia. They found that the LandInfra approach for referencing IFC-based physical elements can be utilised for incorporating physical objects into the Victorian model but they would need to incorporate elements of multiple 3D spatial information models for their final model. LandInfra is also mentioned as one of the potential standards for representing data about

underground infrastructure (utilities and other subsurface features) by the OGC Underground Infrastructure Concept Development Study (CDS) to improve the underground infrastructure data interoperability [190]. LandInfra considers wet infrastructure and utilities within its scope [225], its possible alignments with the CityGML Utility Networks ADE [9] and PipelineML [189] were highlighted in the study [190].

There are several papers discussing the integration of LandInfra and InfraGML with other geospatial standards. Important work is being done in this direction by the team at *Institut Géographique National* (IGN) France for aligning CityGML and InfraGML [26, 57]. Their research investigated the acoustic process and inputs to determine which available data between CityGML and InfraGML is best suited for initial environment acoustic studies [26]. The research also raised several important points such as the lack of flexibility for extensions in the LandInfra conceptual model and the unavailability of real world InfraGML datasets in practice. Devys [57] discussed interoperability, between the RailTopoModel [242] and LandInfra, for railway infrastructure and proposed a mapping between the two models. Labetski et al. [144] analysed the usage of LandInfra as a framework for extending the definition of the LODs for roads in the transportation module of CityGML, but found that the lack of the concept of levels of detail in LandInfra made it irrelevant. Niestroj et al. [179] also propose to analyse LandInfra in the context of roads but for the purpose of determining limitations in current data standards for road assets and create recommendations towards an improved standard in order to apply SW (Semantic Web) technologies to build a prototype solution for road asset data conflation. As they continued their analysis, Niestroj et al. [178] found that the use of IFC, IFC Alignment and InfraGML should be considered, as these standards are supported by several industrial software applications. They conclude with the belief that instead of trying to develop yet another road asset information standard there should be an investigation into translation approaches to assists communication between standards [178]. Malmkvist et al. [162] utilised InfraGML and IFC Alignment for the information exchange of road asset data between the design and operation phases of a road project within different software systems.

Furthermore, attempts are being made to align IFC and LandInfra. For instance, the LandInfra `Alignment` requirement class is based on the buildingSMART IFC Alignment 1.0 standard [188]. It was developed jointly by the OGC and the buildingSMART Infrastructure IfcAlignment project team to ensure interoperability between the two standards in the future. Moreover, buildingSMART is currently working on an IFC Infrastructure extension to model the spatial and physical components of the roads, bridges, and other structures in IFC [33] so that the forthcoming IFC conceptual model for roads and railways be compatible with the LandInfra and InfraGML.

6.3 Comparative analysis between IFC, CityGML, and LandInfra

In this section, we analyse the differences and similarities between CityGML, IFC, and LandInfra. These are briefly summarised in Table 6.1. The comparison of standards is done on the basis of 16 criteria derived from the criteria described in Biljecki and Arroyo Ohori [17], Stoter et al. [232], Zlatanova et al. [258]. The first five criteria enlisted in Table 6.1, namely, `Body`, `Version`, `Users`, `Encoding`, and `Focus` describe the general information about the standards e.g. the standardising body, the current version of the standard, its main users, the type of encoding, and the main focus of the standard, respectively. The criteria `Geometry` discusses the support for different geometries types and semantics in the standards. `Topology` describes how the topological relationships between the geometries of features are stored in the data model of the standards. `Semantics` describes the differences in the modelling of feature semantics between IFC, LandInfra, and CityGML. The criteria `Metadata`, `Land use representation`, `LODs`, `Appearance`, and `Extensions` evaluates the support for metadata, land use, Levels of Detail (LODs), textures/materials and the possibility for extensions to the data model of the standards, respectively. `'Software support'` discusses the available software support for the standards which can be useful for the practitioners. The most relevant differences are analysed in more detail in this section.

1. **Geometry**. IFC uses the many geometry types defined in ISO 10303 [107], which include a variety of representation paradigms within `IfcShapeRepresentation`, such as primitive instancing, CSG, sweeps and B-rep. These paradigms can be used independently or combined with each other in a hierarchy of operations. The elements are usually modelled in local coordinate systems, which are defined by a hierarchical set of transformations based on entities that define local systems (`IfcLocalPlacement`), axes (`IfcAxis2Placement`) and 2D/3D vectors (`IfcDirection`). These systems can correspond to the levels in a decomposition structure (typically a site, project, building and individual floors), or to a series of object locations that are defined based on those below them, among other options. For example, the local placement system of a door may refer to the placement system of the corresponding wall, while that of the wall refers to the building. Global coordinates can be however obtained using the georeferencing information that is sometimes included in IFC files, such as with the latitude, longitude and elevation information in `IfcSite`.

   Meanwhile, CityGML represents elements directly in a single global coordinate system and uses only the B-rep types defined in GML 3.1.1, which represent solids, surfaces, TINs, etc. and are based on the ISO 19107 geometry model with the restriction that only planar and linear geometry types are used. LandInfra is very similar in that it also uses the ISO 19107

Table 6.1: A comparison of CityGML, IFC and LandInfra

#	Criterion	CityGML	LandInfra	IFC
1	Body	OGC	OGC	buildingSMART
2	Version	2.0.0	1.0.0	IFC4 Addendum 2
3	Users	3D city modellers	Survey engineers & BIM	BIM & AEC (Architecture, Engineering & Construction)
4	Encoding	GML	GML	Mainly STEP (Standard for the Exchange of Product model data)
5	Focus	City objects	Land and infrastructure	BIM models
6	Geometry	Subset of ISO 19107 / GML 3.1.1	ISO 19107 + more	ISO 10303
7	Topology	Shared surfaces only	Between facility parts	Openings, coverings and other connections
8	Semantics	Detailed	Not so detailed	Detailed
9	Metadata	Basic	ISO 19115 compliant	Extensively but inconsistently used
10	LODs	5 different LODs	Not supported	Not supported
11	Extensions	Generics or ADEs	Not supported	Supported
12	Appearance	Supported	Not supported	Supported
13	Software support	Low	Almost nonexistent	Medium
14	Codelists	Supported with ISO 19103	Supported with ISO 19103	Enumerations only
15	Land use	Simple types	Complex LADM types [104]	Not relevant
16	File size	Large [132]	Large	Very large

geometry model, but it defines new geometry types such as `IndexedPoint`, `PolyfaceMesh`, and `SimpleIndexedPolygon` in its conceptual model. InfraGML uses GML 3.2 for solids and surfaces, and GML 3.3 for triangles and TINs.

Despite the fact that the latter two standards use GML, there are still differences between them. For example, CityGML represents TINs as a triangulated surface (`gml:TriangulatedSurface`) with triangles specified with a Simple Features geometry (`gml:Triangle`). In the Simple Feature structure, the first vertex of every linear ring (triangle/polygon) is repeated as the last vertex of the linear ring. On the other hand, InfraGML uses GML 3.3 where a TIN is represented as a collection of `gmltin:SimpleTrianglePatch`. It is based upon the GML 3.3 SimpleTriangle, rather than the GML 3.1.1 or GML 3.2 Triangle [186] and avoids the repetition of first vertex as the last vertex in each triangle.

As another example, LandInfra defines a 'Polyface Mesh' geometry to compactly represent the boundary of a solid. A Polyface Mesh in LandInfra stores every surface (triangle/polygon) of a solid as references to the IDs of the vertices forming that surface (see InfraGML Snippet 1). The vertices are stored in a separate list with their IDs and are not repeated for every triangle like in Simple Features. CityGML supports no such geometry in the actual model. However, CityGML iTINs ADE[1] implemented new geometry types in the GML schema which are extended to existing CityGML features for compact representation of massive TINs, up to a factor of around 20 [136].

Snippet 1: Polyface Mesh spatial representation in InfraGML

```
....
<spatialRepresentation>
  <geometry>
    <!-- Polyface Mesh geometry -->
    <PolyfaceMesh gml:id="polyfaceMesh01">
      <IndexedPointList>
        <IndexedPoint>
          <index>1</index>
          <coordinates>0 0 0</coordinates>
        </IndexedPoint>
        <IndexedPoint>
          <index>2</index>
          <coordinates>1 0 0</coordinates>
        </IndexedPoint>
        <IndexedPoint>
          <index>3</index>
          <coordinates>1 1 0</coordinates>
```

[1] https://github.com/tudelft3d/CityGML_iTINs_ADE

```
        </IndexedPoint>
      </IndexedPointList>
      <SimpleIndexedPolygonList>
        <SimpleIndexedPolygon gml:id="indexedPolygon01">
          <pointIndex>1 2 3</pointIndex>
        </SimpleIndexedPolygon>
      </SimpleIndexedPolygonList>
    </PolyfaceMesh>
  </geometry>
</spatialRepresentation>
....
```

Furthermore, LandInfra supports the concept of an `Alignment` for linear construction works, such as roads and rails, which is similar to `IfcAlignment` [188]. The simplest geometry representation for an alignment is a 2D straight line, but an alignment can consist of multiple segments which are connected, i.e. from the end of one to the start of the next. Since there is no requirement that a segment should be tangentially continuous with the next one, the transition from one segment to the other can be jerky when using straight lines for representing these segments. However, it is often recommended to smooth out the transitions using a circular curve, clothoid, or another spiral for design and construction, which are supported by the LandInfra `Alignment` class, similar to `IfcAlignment` [188]. These geometry types, which are taken from the OGC Abstract Specification Topic 1 (Feature Geometry in LandInfra), and are not supported in CityGML.

2. **Topology**. Topology in BIM usually refers to hierarchical geometric representations like CSG or Half-space intersection models. However, IFC also contains several topological relationships in a GIS sense. Elements are expected to be connected to their openings (`IfcOpeningElement`) and coverings (`IfcCovering`), and there are also various connections between related elements defined using the connectivity relationship `IfcRelConnects`, such as with ports (`IfcPort`) and the structural members of an element (`IfcStructuralMember`).

   CityGML uses the concept of *XLinks* provided by GML to store only once a surface shared by two objects. For example, if a wall (`bldg:WallSurface`) is shared by two different buildings, then its ID can be referenced by the other building using *XLinks*. This mechanism is however not mandatory [185] and a CityGML dataset can contain repetition of multiple identical geometries [20]. No other topological relationship e.g. adjacency or incidence can be explicitly stored in the model.

   LandInfra conceptual model uses the same concept of XLinks for sharing of surfaces among features. It is also possible to link all the facility parts

(lif:FacilityPart) to the facility (lif:Facility) they belong to using Xlinks (see InfraGML Snippet 2). Further, relationship between different facility parts can be specified using XLinks.

Snippet 2: Topology relationship between the facility parts in InfraGML

```
....
  <!-- A Facility in InfraGML with two facility parts -->
  <lif:Facility gml:id="Facility_f1">
    <lif:part xlink:href="#fp_1"/>
    <lif:part xlink:href="#fp2_2"/>
  </lif:Facility>

  ....
  <!-- A facility part in InfraGML -->
  <lif:FacilityPart gml:id="fp_1">
    <!-- Relationship between facility parts in InfraGML -->
    <lif:relationship>
      <lif:FacilityPartRelationship gml:id="fpr_1">
        <lif:relationship>connected</lif:relationship>
        <lif:description>fp1 connected to fp2</lif:description>
        <lif:facilityPart xlink:href="#fp2"/>
      </lif:FacilityPartRelationship>
    </lif:relationship>
  <lif:FacilityPart>
  ....
....
```

3. **Semantics**. There are clear differences in the modelling of feature semantics between IFC, LandInfra and CityGML. For example, a building in CityGML can be subdivided into semantic surfaces such as roofs, walls, doors, and windows. In IFC, it would instead be subdivided into the elements used in its construction, such as slabs, columns and beams, as well as fittings like windows, stairs and doors. Neither of these are possible in LandInfra.

   All three standards exhibit coherence between the semantics and the geometry of the objects they model. For instance, in CityGML, if the hierarchical decompositions of semantics and geometry depict the same structure, then they are considered coherent [185, 231]. For example, a building represented as a `gml:CompositeSolid` can be decomposed into two building parts, each of which is a `gml:Solid`. This is similar to IFC, since many building elements (i.e. `IfcElement` and its subtypes) have a concrete semantic meaning in theory, such as the subtypes of `IfcBuildingElement`: `IfcCovering`, `IfcBeam`, `IfcColumn`, `IfcCurtainWall`, `IfcDoor`, `IfcMember`, `IfcRailing`, `IfcRamp`, `IfcRampFlight`, `IfcWall`, `IfcSlab`, `IfcStairFlight`, `IfcWindow`, `IfcStair`, `IfcRoof`, `IfcPile`, `IfcFooting`, and `IfcPlate`. How-

ever, these are inconsistently used in practice, and software often just exports generic types like `IfcBuildingElementProxy` instead.

LandInfra also exhibits coherence between semantics and geometry of features in its data model. In Figure 6.2, a `Facility` (represented as a `gml:MultiSolid` in InfraGML) is decomposed into two `FacilityPart(s)`. A facility part can either be a building, a road or a railway feature. If the two `FacilityPart(s)` are the same, e.g. buildings with `gml:Solid` geometry, then the hierarchical decomposition of geometry is structured similarly to CityGML. However, there is no (or partial) coherence if the facility parts are different with different geometry, e.g. a building with `gml:Solid` geometry and a road with `gml:MultiSurface` geometry.

Additionally, even as there are many similarities between the LandInfra feature types and the CityGML thematic classes, they are not always grouped in the same way and also have different names for the same concepts. For example, LandInfra separates `Roads` and `Railways` while CityGML groups the two in the `Transportation` thematic module.

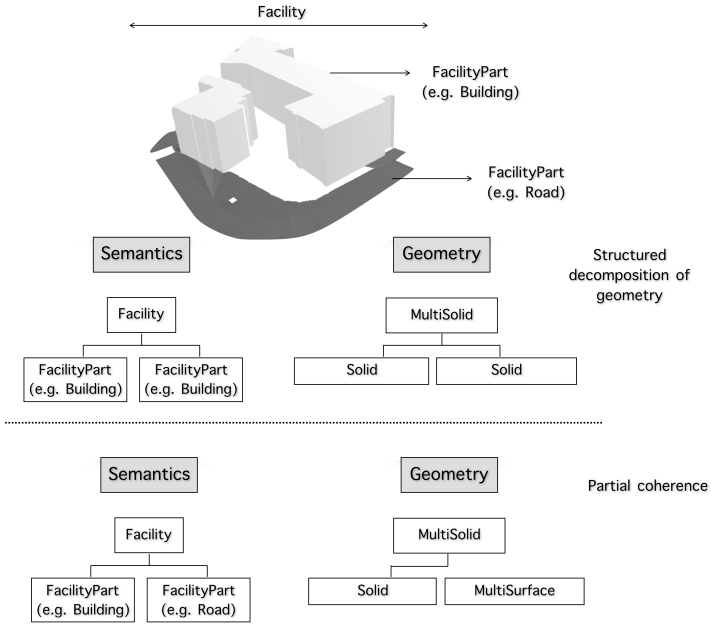


Figure 6.2: Geometric-semantic coherence in the LandInfra `Facility` class

4. **Metadata**. Metadata is extensively used in IFC, but it is not used in a consistent manner by different software. For basic information, the IFC standard provides specific entries for the header of an IFC file (`FILE_DESCRIPTION`, `FILE_NAME`, `FILE_SCHEMA`), as well as specific entities in the body of the file, such as `IfcOrganization`, `IfcPerson` and `IfcPostalAddress`. Additional information is usually added through a reference (e.g. `IfcDocumentInformation`) to an external document. The reference captures metadata of the external file (e.g. document IDs, author, description, purpose and timestamps), and the metadata of the IFC file is contained in the external document. For instance, the latter process is often done to add scheduling and construction information in IFC files.

   In CityGML, there is very basic support for metadata using `gml:metaDataProperty` inherited from GML3 and is not ISO 19115 compliant [108]. GML does not provide an information model for metadata, instead a mechanism to include or reference metadata is provided [184]. A 3D Metadata ADE[2] was recently developed focusing on adding metadata related to 3D city models in CityGML [143]. It incorporates ISO 19115 metadata elements and several other elements related to 3D city models such as LODs, feature count, and metadata related to CityGML thematic models. LandInfra has ISO 19115 compliant metadata to describe the geospatial dataset and sensor observations (see InfraGML Snippet 3).

Snippet 3: Metadata about the dataset in InfraGML

```
<LandInfraDataset gml:id="GML_e8e7963f-718c-40fb-8253f">
  <datasetID>
    <ID>
      <identifier>GML_e8e7963f-718c-40fb-8253f</identifier>
      <scope>OGC LandInfraSWG</scope>
    </ID>
  </datasetID>
  <name>Land Infra Dataset</li:name>
  <description>LandInfra Dataset of terrain</description>
  <dateTime>2018-10-04T16:52:59</dateTime>
  <datasetVersion>1.0</li:datasetVersion>
  <application>Generated by CityGML2InfraGML utility</application>
  <author>TU Delft</author>
  <infraVersion>1.0</infraVersion>
  <language>English</language>
  <defaultCRS xlink:href="EPSG:28992"/>
  ....
</LandInfraDataset>
```

[2] https://github.com/tudelft3d/3D_Metadata_ADE

5. **LODs (Levels of Detail)**. CityGML supports 5 different LODs, from LOD0 to LOD4 for multi-representation of 3D city objects. In CityGML, the concept of LODs is very well established for buildings and bridges. For instance, LOD0 for a building is a 2D footprint, LOD1 is a block model generated by extruding the footprint, LOD2 is an upgraded LOD1 model with roof structure and semantically differentiated boundary surfaces, LOD3 are architecturally detailed models, and LOD4 models contain information about the interior of an object (see Figure 6.3). Biljecki et al. [23] proposed an improved LOD specification for buildings, however, it is not a part of the current CityGML specifications.

   IFC files usually contain building models only in very high detail. Since BIM focuses on information about the design and construction of building sites, it thus usually has very geometrically complex and semantically rich information about the buildings [3]. However, they can also contain 2D architectural floor plans as well as the usual 3D building models in one file [3]. However, regarding BIM models in general, there is the concept of the *level of development* (also abbreviated as LOD), which represents a model in the typical stages that it goes through. These include everything from its conception, detailed design, construction and the as-built model for facilities management. As the model gets progressively more detailed in these stages, the concept is indirectly related to the level of detail in it.

   LODs are not supported in LandInfra.

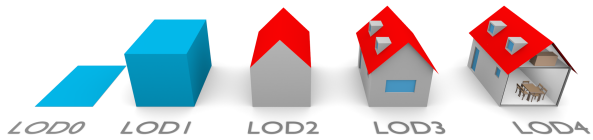


Figure 6.3: A building represented in LOD0 to LOD4 in CityGML (Source: Biljecki et al. [18])

6. **Extensions**. It is possible to extend the CityGML model using `Generic` city objects or ADEs [25]. Extensions and Generics are not supported in LandInfra. IFC models can be extended using property sets, proxy elements, and by defining new entities or types [249]. Several researchers have defined their own IFC extensions using these "de facto" methods, e.g. IFC extension to estimate the construction cost of buildings by Zhiliang et al. [257], IFC extension to incorporate information of RFID tags attached to building elements in IFC by Motamedi et al. [169] and so on.
7. **Appearance (Textures & Materials)**. IFC has wide support for appearance in two ways: for design and construction purposes through `IfcMate-`

`rial` (e.g. to know its mechanical or fire resistance capabilities stored as `IfcMaterialProperties`), and for visualisation purposes through `IfcMaterialDefinitionRepresentation`. CityGML draws concepts from both X3D [105, 109] and COLLADA [8] for material and texture information of city objects [185], LandInfra does not support textures nor materials.

8. **Software Usage & Support**. LandInfra conceptual model was accepted as an OGC standard in 2016. Its GML encoding (InfraGML 1.0) became a standard later in 2017. In spite of a stable release, there is no software support available, that I know of, to parse, visualize, and use InfraGML.

   On the other hand, a number of software packages and libraries are available which can be useful for practitioners and researchers dealing with CityGML. Most of the software are recent and well-maintained. For instance, citygml4j[3] is an open source Java library for reading/writing CityGML datasets; 3D City Database [255] is a database solution to store, and manage CityGML models on top of a standard spatial relational database (PostGIS and Oracle); azul[4] and FZK[5] are popular CityGML viewers for macOS and Windows, respectively and so on. CityGMLwiki[dot]org [41] provides a list of available software for CityGML.

   IFC has the widest usage and software support among these standards, but the standard is implemented inconsistently by different software, which limits compatibility in practice. In particular, most BIM and building design software can export from its native (internal) formats to IFC, but there is a degree of information loss while doing so. Importing from IFC is known to be even more problematic, as arguably less effort has been put into this process. Recently, a GeoBIM software compatibility benchmark[6] was funded to assess all of these issues in more detail. Some of the well known open-source projects for IFC include IfcOpenShell[7], BiMserver[8], etc. IFCwiki[dot]org [93] provides a list of available software for IFC.

9. **Codelists**. While both CityGML and LandInfra define their code lists in accordance with ISO 19103 — Geographic information — Conceptual schema language [110], neither follows a standard in naming conventions which makes mapping between similar code lists impossible. This means that there may be significant overlap between two code lists and thus unnecessary duplication. There is a further risk of a specific terminology being utilised twice but with differing definition or meaning. The issue of

[3] https://github.com/citygml4j/citygml4j
[4] https://github.com/tudelft3d/azul
[5] https://www.iai.kit.edu/1302.php
[6] https://3d.bk.tudelft.nl/projects/geobim-benchmark/
[7] http://www.ifcopenshell.org
[8] http://www.bimserver.org

standardising code lists and enumerations is described further in the work of Stubkjær et al. [235].

Meanwhile, IFC only has support for enumerations, but the standard does contain a lot of them (207 in IFC4 Addendum 2), and they have similar extensibility to codelists because they contain specific definitions for user-defined and undefined types.

10. **Land use representation**. CityGML only represents the division of the Earth's surface according to specific land use e.g. residential, industrial, and so on. LandInfra uses ISO 19152:2012 Land Administration Domain Model (LADM) [104] which offers a rich conceptual scheme for recording of interests in land including above and below the ground surface, ownership, rights, restrictions, and so on [188]. Since IFC focuses on specific building sites, land use is typically not a concern.

6.4 Minor practical issues with LandInfra and InfraGML

During the course of our evaluation of LandInfra, we encountered some issues which we believe are of concern when using the standard in real world applications.

1. In Section 7.9.4 of the LandInfra specifications document, it is mentioned that "LandLayer is an abstract class". However, it is implemented as a concrete class in InfraGML and can be instantiated (see Snippet 9).

Snippet 9: Instantiated LandLayer in InfraGML

```
<feature>
  <lilf:LandLayer gml:id="ll1">
    <gml:description>A land layer land feature that
    is not a SolidLayer SurfacesLayer or LinearLayer
    </gml:description>
    <gml:name>OtherLandLayer</gml:name>
    <lilf:state>existing</lilf:state>
    <lilf:landLayerID>
      <lilf:ID>
        <identifier>OLL1</identifier>
        <scope>OGC LandInfraSWG</scope>
      </lilf:ID>
    </lilf:landLayerID>
  </lilf:LandLayer>
</feature>
```

2. InfraGML is developed as a multi-part standard and each part has a separate XML schema file (XSD). When trying to validate an InfraGML dataset containing more than one type of requirement class (such as Facilities with

LandSurface) against one XSD, validation errors are reported. A 'wrapper schema' (like in CityGML) which links the XSDs of all the parts of InfraGML is missing.

3. LandInfra `Document` class contains 'information in permanent form approved by one or more signature' [188]. It is derived from LandInfra `Feature` class and therefore has an optional spatial representation `(spatialRepresentation)`. It is not clear what the spatial representation of a document could be: that described in the document, the spatial location where the document is located, or other possible interpretations.

4. Given that the facilities, such as `Roads`, inherit the `spatialRepresentation` property from the LandInfra `Feature` this means that the representation type of facilities is not restricted by type, but instead can utilise any geometry type that is supported by `spatialRepresentation`. This can lead to unrealistic representation types that validate under the current schema, such as a road modelled as a point.

Snippet 10: Unrealistic representation of a road as a point in InfraGML

```
<feature>
  <lifr:Road gml:id="r1">
    <lifr:element>
      <lifr:RoadElement gml:id="re1">
        <spatialRepresentation>
          <SpatialRepresentation>
            <geometry>
              <gml:Point gml:id="p2">
                <gml:pos>105 230</gml:pos>
              </gml:Point>
            </geometry>
          </SpatialRepresentation>
        </spatialRepresentation>
      </lifr:RoadElement>
    </lifr:element>
  </lifr:Road>
</feature>
```

5. The LandInfra standard includes future proposed classes directly in the current UML schema which can be confusing to interpret and can potentially give the impression of a false claim of completeness. For example, the `Facility` requirement class has 8 proposed future classes, namely, `Bridge`, `Drainage`, `Environmental`, `Site`, `Tunnel`, `Wastewater`, `WaterDistribution` and `OtherFacility` (see Figure 6.4). Additionally it seems unrealistic to propose future classes mixed directly with current classes without having tested and received feedback for the current classes. This may discourage contribution from potential collaborators by providing a false sense of finality of future versions of LandInfra.

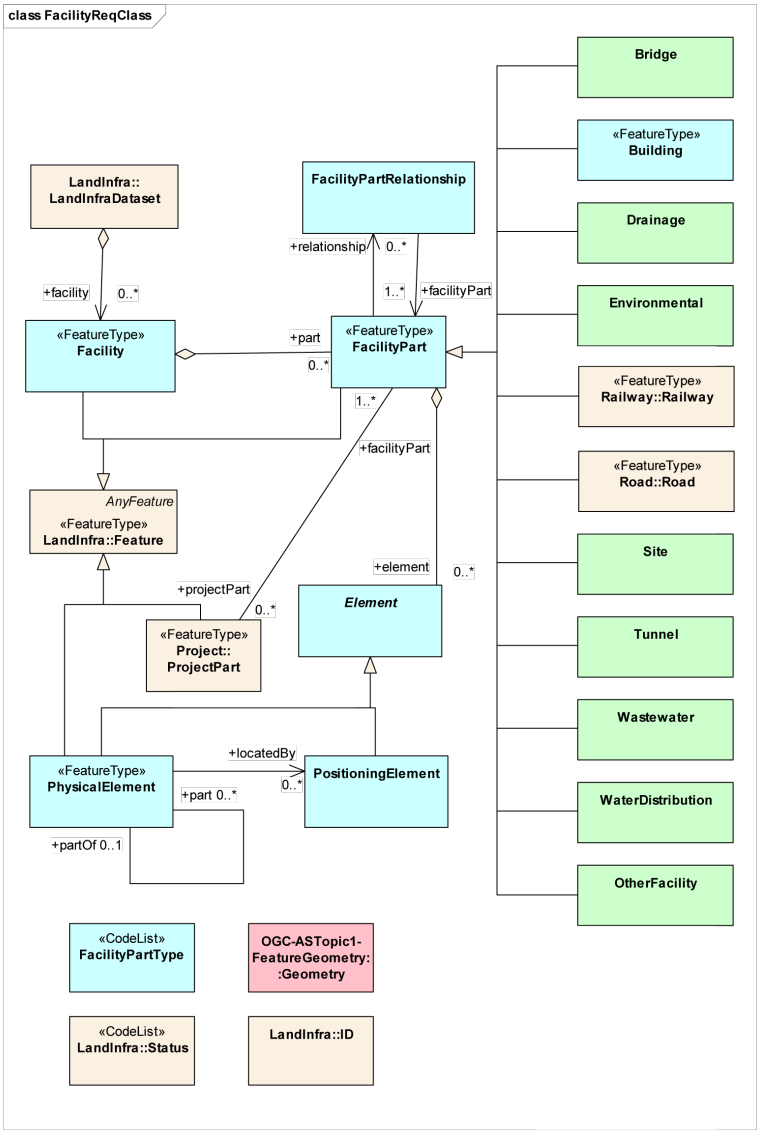


Figure 6.4: LandInfra `Facility` requirement class demonstrating future proposed classes for `FacilityPart`, (Figure 10 from OGC [188])

6.5 Conclusions and future work

This chapter presents our investigation on how the development of new multi-disciplinary standards such as the LandInfra can contribute to the convergence of interoperability issues between different 3D domains, specifically BIM and GIS. We provided a detailed comparison of the LandInfra conceptual model with those of CityGML and IFC. The comparison aimed to discuss the substantial overlaps of LandInfra standard with CityGML and IFC, and its potential to bring the GIS and BIM community on a common footing. As an open standard at the junction between BIM and GIS, LandInfra is uniquely situated to act as a "connecting bridge" for BIM-GIS integration. It has good support for metadata, land division, and the ISO 19107 geometry types are all positive aspects of the standard for potential GIS and BIM integration, as is the involvement of buildingSMART and practitioners in its development.

However, at this moment, it is hard to claim that LandInfra is the answer to many BIM-GIS integration problems. First, as it stands, LandInfra is clearly a standard that is much closer to the standards of 3D GIS than to how objects are modelled in BIM world, which in practice will greatly limit its interoperability with BIM formats like IFC. Second, the data model of LandInfra cannot be extended or modified, which limits its use in practice especially since it aims as bridging different communities. By comparison, the core data model of CityGML can be extended in a structured manner with the ADE mechanism (as explained in Chapter 2). Third, the current lack of example InfraGML datasets makes extensive testing and validation difficult, if not impossible. The fact that there is no software packages to read/write, edit, or manipulate LandInfra makes it rather difficult to convince practitioners to convert their datasets to it. Without open implementations and greater concern for the implementability of the standard, there is a danger that the standard will languish and be unused, as has been the case with a few OGC and other open standards. To address this issue, we have worked on a solution i.e. a LandInfra ADE for CityGML, which will encourage the adoption of LandInfra's features (next Chapter of this thesis).

Chapter 7

Harmonising the OGC standards for the built environment

This chapter is based on the paper:

Kumar K, Labetski A, Ohori K A, Ledoux H, and Stoter J, 2019. Harmonising the OGC Standards for the Built Environment: A CityGML Extension for LandInfra. *ISPRS International Journal of Geo-Information*, 8(6), p.246.
doi: https://doi.org/10.3390/ijgi8060246

7.1 Introduction

In Chapter 6, we review LandInfra, and its relation to CityGML and IFC. Currently, LandInfra has no software support yet and is barely used in practice, which means that the advantages of LandInfra (and InfraGML) are not yet being used. The fact that a standard has been tested and implemented in code is a positive feature of the standardization approach and will help increase the usability of the standard. In this Chapter, we focus on the harmonisation of LandInfra and CityGML in order to encourage the adoption of LandInfra's features. We have developed and implemented the *Infra ADE*—an Application Domain Extension for CityGML that integrates LandInfra's concepts into CityGML, which we describe in this chapter, including the steps that we have taken and the decisions we have made to develop the ADE. The idea behind the integration is to take the best of both worlds (i.e. CityGML and LandInfra) and have more information than CityGML can represent for specific applications or use cases such as urban environment analysis, subsurface modelling, etc. We provide in the following a complete mapping between LandInfra and CityGML, where we identify the matching classes and attributes in the two data models, as well as the LandInfra classes and attributes that do not have a semantic equivalent in CityGML but are useful for the built environment applications, e.g. the material of a building, and the life cycle phase of a building. We also discuss a few use cases for the built environment of the CityGML Infra ADE to bring the benefits of our ADE in practice. As a proof of concept of the Infra ADE, we have also implemented two software prototypes to convert datasets from InfraGML to CityGML (and our ADE) and vice versa, as well as a prototype to ensure that the InfraGML files are valid. It should be noticed that we do not attempt to convert to a harmonised data model (see El-Mekawy et al. [71] as an example) since the resulting files would not be useful in practice: software packages do not have support for such data models.

We chose to map LandInfra with CityGML, and not with IFC, because of the following reasons:

1. The scope of IFC is traditionally limited to modelling buildings and it does not support other city features such as terrain and landuse, which are already available in CityGML. Features like roads and bridges in IFC are slated for future release[1]. LandInfra is much closer to CityGML with respect to geometry, semantic classes, and UML model and, has an GML/XML based encoding (InfraGML) similar to CityGML (Figure 6.1 in Chapter 6). Further, IFC uses sweep volumes, CSG (Constructive Solid Geometry), and b-rep (boundary representations) as geometric representations, whereas CityGML and LandInfra support only b-rep for geometry.

[1] At the time of writing this thesis, features like roads and bridges were slated for future release in the IFC standard.

Mapping IFC and LandInfra would require transforming sweeping volumes and/or CSG geometry of objects to boundary representation.

2. IFC does not have world coordinates, it has a local coordinate system (`IfcLocalPlacement`) to describe the location of an object. It does not refer to the location of an object in the real world as LandInfra and CityGML do. For example, the local placement system of a door may refer to the placement system of the corresponding wall, while the wall refers to the building. However, in CityGML & InfraGML all the objects are defined with their absolute coordinates in a regional or world coordinate system. Coordinates are required to be transformed from regional or world to local coordinate system when mapping from CityGML/LandInfra to IFC [222].

3. There is no official mechanism to extend IFC whereas CityGML has Generics and ADEs (see Chapter 2). Weise et al. [249] lists three methods to extend IFC, namely: defining new entities or types, using proxy elements, and using the property sets or types. Several researchers have defined their own IFC extensions using these "de facto" methods, e.g. IFC extension to estimate the construction cost of buildings by Zhiliang et al. [257], extension to incorporate information of RFID tags attached to building elements in IFC by Rajabifard et al. [213] and so on.

7.2 Methodology for mapping LandInfra and CityGML

LandInfra and CityGML have significant similarities and differences, which we have discussed in detail in the previous chapter. After comparing the two standards and analysing the individual correspondences of the classes, attributes and other concepts in the data model of LandInfra to their equivalent ones in CityGML, we found that they fit into five different categories as mentioned below. To avoid any confusion in the names of the classes, the LandInfra and CityGML classes are appended with prefixes `LI` and `CG` respectively, in this research.

1. Classes (and their attributes) which can be directly mapped from LandInfra to CityGML, e.g. `LI::LandSurface` with `CG::TINRelief`, `LI::Road` with `CG::Road`, and so on.

2. LandInfra classes that require a specific attribute value to determine their corresponding matching classes in CityGML e.g. `LI::LandElement` can be mapped to `CG::PlantCover`, `CG::WaterBody` or `CG::TINRelief` based on the value of its attribute `elementType`. The values for `elementType` attribute are defined in the LandInfra codelist `LandElementType`.

3. LandInfra classes (and their attributes) that do not have a semantically equivalent class in CityGML e.g. classes such as `LI::_LandLayer` and `LI::-Facility`.

4. Classes (and their attributes) that define relationships among features e.g. how different LandInfra features (`LI::Feature(s)`) are related to each other via `LI::FeatureAssociation`, which are not present in CityGML.
5. LandInfra constraints such as data types, enumerations, and codelists which do not have a semantic equivalent in CityGML.

Based on these categories, we have developed a complete mapping from LandInfra to CityGML, which is presented in Table 7.1. Notice that many classes in LandInfra do not have clear correspondences in CityGML (option 3, listed above), which is largely because LandInfra is much more detailed than CityGML with respect to the semantic information and relationships of land and infrastructure features. Loss of information while converting InfraGML datasets to CityGML is thus inevitable without an extension to the CityGML data model.

Table 7.1: Mapping between the LandInfra conceptual model and CityGML. For simplicity, we only show here the mapping for all the LandInfra main requirement classes (marked with M) and a few other LandInfra classes as examples.

#	LandInfra	CityGML	Description
1	`LandInfraDataset` (M)	`CityModel`	Aggregations of features with optional metadata.
2	`Feature`	`_CityObject`	The base classes for all the features. Both are derived from GML class `_Feature`, but `LI::Feature` is a concrete class whereas `CG::_CityObject` is abstract; they are only conceptually similar.
3	`Document`	—	No matching semantic class available.
4	`SurveyMark`	—	No matching semantic class available.
5	`Project` (M)	—	No matching semantic class available.
6	`Survey` (M)	—	No matching semantic class available.
7	`Alignment` (M)	—	No matching semantic class available.
8	`FacilityPart`	—	No matching semantic class available.

continued on next page

Table 7.1 — continued from previous page

#	LandInfra	CityGML	Description
9	`Facility` (M)	`_Site`	Both include buildings and other civil engineering structures, such as tunnels and bridges, but `LI::Facility` also defines runways, pipelines and water systems, which are not present in CityGML. However, `LI::Facility` is a concrete class whereas *`CG::_Site`* is abstract; they are only conceptually similar.
10	`Building`	`Building`	In LandInfra it is a subclass of `LI::FacilityPart`, whereas in CityGML it is a part of the `CG::Building` module.
11	`LandFeature` (M)	—	No matching semantic class available.
12	`LandElement`	`WaterBody`/ `PlantCover`/ `TINRelief`	The correspondence depends on the value of a `LI::LandElement`'s `elementType`, which can be `waterBody`, `vegetation` or `landForm`, as defined in the codelist `LandElementType`.
13	`LandSurface`	`TINRelief`	`LI::LandSurface` models terrain as a TIN, much like `CG::TINRelief`.
14	`_LandLayer`	—	No matching semantic class available.
15	`Condominium-Building` (M)	—	No matching semantic class available.
16	`Road` (M)	`Road`	In the `Transportation` module in CityGML and in the `Facility` requirement class in LandInfra. CityGML focuses on polygon representations with the possibility to include lines as well, while LandInfra focuses on lines and surfaces (i.e. TINs).

continued on next page

Table 7.1 — continued from previous page

#	LandInfra	CityGML	Description
17	`Railway` (M)	`Railway`	In the `Transportation` module in CityGML and in the `Facility` requirement class in LandInfra. LandInfra provides many more components that can be modelled.
18	`LandDivision` (M)	`LandUse`	`CG::LandUse` represents divisions according to specific land uses, whereas `LI::LandDivision` has richer semantics and is further divided according to political, judicial, or executive views, ownership, rights, and so on.

7.3 The CityGML Infra ADE

In order to support LandInfra concepts in CityGML, we have developed the Infra ADE, which is able to store and manage LandInfra/InfraGML datasets in CityGML with full compatibility. We implemented the ADE and provide the UML model, the XSD schema, the documentation and a prototype software, which are all publicly available in the GitHub repository: https://github.com/tudelft3d/city2InfraGML.

Depending on the five cases of the classification in our analysis, we have built the Infra ADE for CityGML by:

1. adding the missing LandInfra attributes to the CityGML classes that matched LandInfra classes (Cases 1 and 2);
2. adding new types that represent the LandInfra classes that do not have matching CityGML classes (Case 3 and 4);
3. adding support for the LandInfra geometry types, data types and codelists (Case 5).

These solutions are individually presented in the following subsections. Note that in order to avoid any conflict with the existing CityGML elements, the new Infra ADE elements are defined in a different namespace https://3d.bk.tudelft.nl/schemas/infraADE with an identifier `'infra'`.

7.3.1 Extending the CityGML classes that match LandInfra classes

Following CityGML classes are extended to match the LandInfra classes in our ADE:

- **CityModel.** `CG::CityModel` is extended to include the `LI::LandInfraDataset`'s classes and attributes (Figure 7.1):
  - the ID and the scope of the dataset (`infra:datasetID`),
  - the metadata about the dataset (e.g. `infra:name`, `infra:dateTime` and `infra:author`),
  - the associations between the features in the dataset (`infra:featureAssociation`),
  - the information about the survey(s) done (`infra:survey`),
  - and the collective information about the features belonging to a particular type or authority in the dataset (`infra:set`).

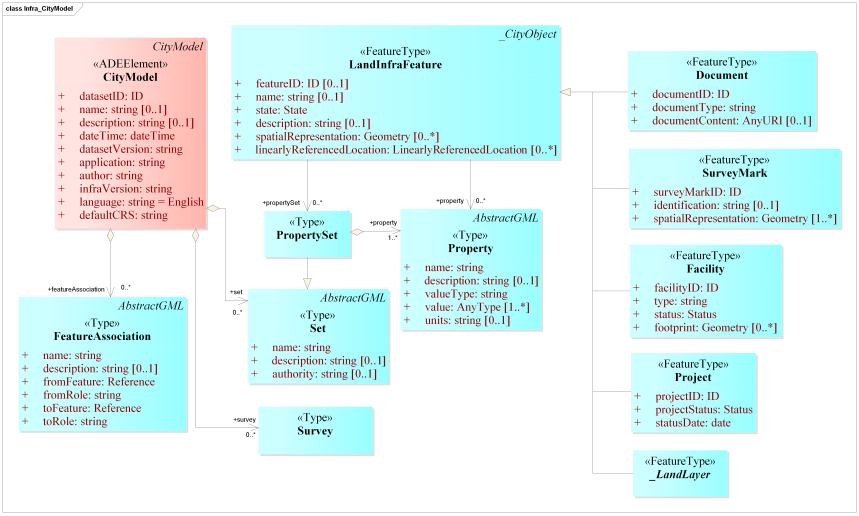


Figure 7.1: The UML excerpt depicts extended CityGML class `CG::CityModel` (marked with stereotype «ADEElement») to include attributes from the LandInfra class `LI::LandInfraDataset`. The LandInfra classes such as `Facility`, `Project`, `LandFeature`, `Document`, `SurveyMark`, and `_LandLayer`, which do not have a corresponding match in CityGML are introduced in the Infra ADE.

- **Railway.** `CG::Railway` is extended to include the `LI::Railway`'s classes and attributes (Figure 7.2):

- the ID and the scope of the railway features in the dataset (`infra:railwayID`),
- the attribute to indicate if the railway feature is existing or proposed (`infra:railwayState`),
- and the status to indicate where the railway feature is within its life cycle (`infra:railwayStatus`),
- the railway elements such as switches, rails, etc. present in the dataset (`infra:railwayElement`),
- the specifications of the cant (also called superelevation) of the railway tracks present in the dataset (`infra:cantSpecification`),
- the alignments (positioning elements) used to define the geometry of the railway tracks (`infra:railwayAlignment`).

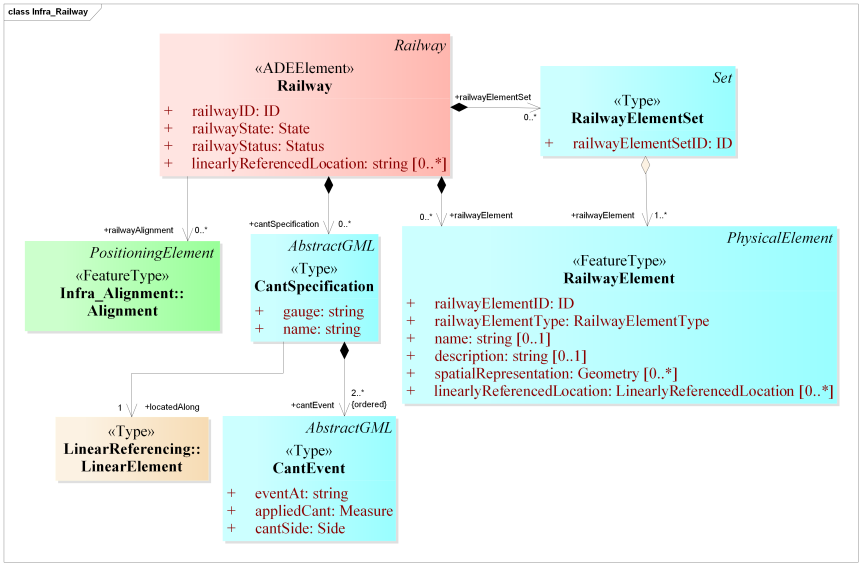


Figure 7.2: The UML excerpt depicts existing CityGML class `CG::Railway` (marked with stereotype «ADEElement») extended to include attributes and classes from LandInfra `LI::Railway` class in the CityGML Infra ADE. Other classes (taken from LandInfra) such as `CrossSection`, `Alignment`, etc. are also introduced.

- **Road.** `CG::Road` is extended to include the `LI::Road`'s classes and attributes (Figure 7.3). It stores:
  - the ID and the scope of the road features present in the dataset (`infra:roadID`),
  - the estimated width of the road (`infra:approximateWidth`),

- the material of the road (`infra:material`),
- the attribute to indicate if the road feature is existing or proposed (`infra:roadState`),
- and the status to indicate where the road feature is within its life cycle (`infra:roadStatus`).
- the road elements such as pavements, side walks, etc. present in the dataset (`infra:roadElement`),
- the alignments (positioning elements) used to define the geometry of the roads (`infra:roadAlignment`),
- alternative ways for representing a road from design perspective such as 3D StringLines (`infra:stringLine` aka profile views, longitudinal breaklines, long sections), and 3D surfaces (`infra:surface`), or as well as collections of these (`infra:stringLineSet` or `infra:surfaceSet`) [188],
- the 2D cross section views cut across the road at a particular location along the length of the road (`infra:roadCrossSection`).

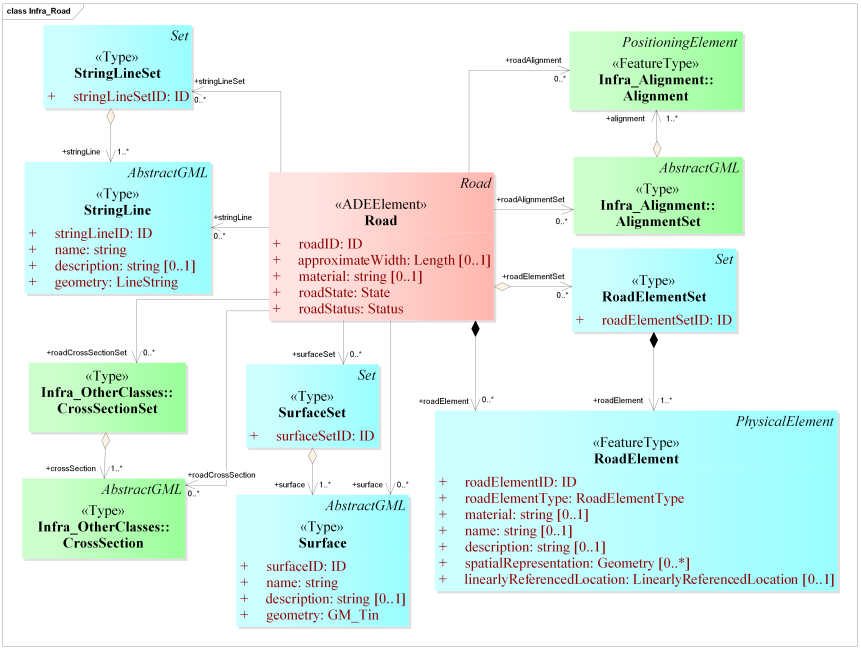


Figure 7.3: The UML excerpt depicts existing CityGML class `CG::Road` (marked with stereotype «ADEElement») extended to include attributes and classes from LandInfra `LI::Road` class in the CityGML Infra ADE.

- **LandUse.** `CG::LandUse` is extended to include the `LI::LandDivision`'s classes and attributes (Figure 7.4):
  - the ID and the scope of the land division features present in the dataset,
  - the type of land division. It can be public (`infra:administrativeDivision`) or private (`infra:easement` or `infra:propertyUnit`) in nature,
  - the statement document which specifies which establishment or acquisition of the land (`infra:documentation`),
  - the ownership rights of properties (`infra:ownership`),
  - the cadastral parcels present in the dataset (`infra:landParcel`),
  - the spatial units to define the geometry (shape and location) of the land parcels, easements, and other administrative divisions (`infra:SpatialUnit`),
  - and the bounding elements to specify the boundary of the spatial units (`infra:boundingElement`).
- **TINRelief.** `CG::TINRelief` is extended to include the `LI::LandSurface`'s attributes:
  - the ID and the scope of the land surface features present in the dataset (`infra:landSurfaceID`),
  - the material of the land surface (`infra:material`),
  - the spatial representation of the land surface as TINs (similar to `TINRelief`),
  - and the attribute to indicate if the feature is existing or proposed (`infra:state`).

  A LandInfra `LI::LandElement` feature can be a terrain, water body, or vegetation depending upon the value of its attribute `elementType`. Therefore, the CityGML classes `CG::WaterBody`, and *CG::_VegetationObject* (`CG::PlantCover` and `CG::SolitaryVegetationObject`) are extended to include the attributes of the LandInfra `LandElement` class, such as ID (`infra:landElementID`), type of land element (`infra:landElementType`), material (`infra:material`), state (`infra:state`), property (`infra:property`), and sets of associated properties (`infra:propertySet`), and so on.

  Similarly, CityGML `CG::Building` and `CG::BuildingPart` are extended to include LandInfra `LI::Building`'s (from `LI::FacilityPart`) attributes such has ID, state and status.

7.3.2 Adding new feature types for non-matching LandInfra classes

A new feature type `LandInfraFeature` is introduced as a subclass of *CG::_CityObject* to represent the LandInfra's `LI::Feature` class. Since `LI::Feature`

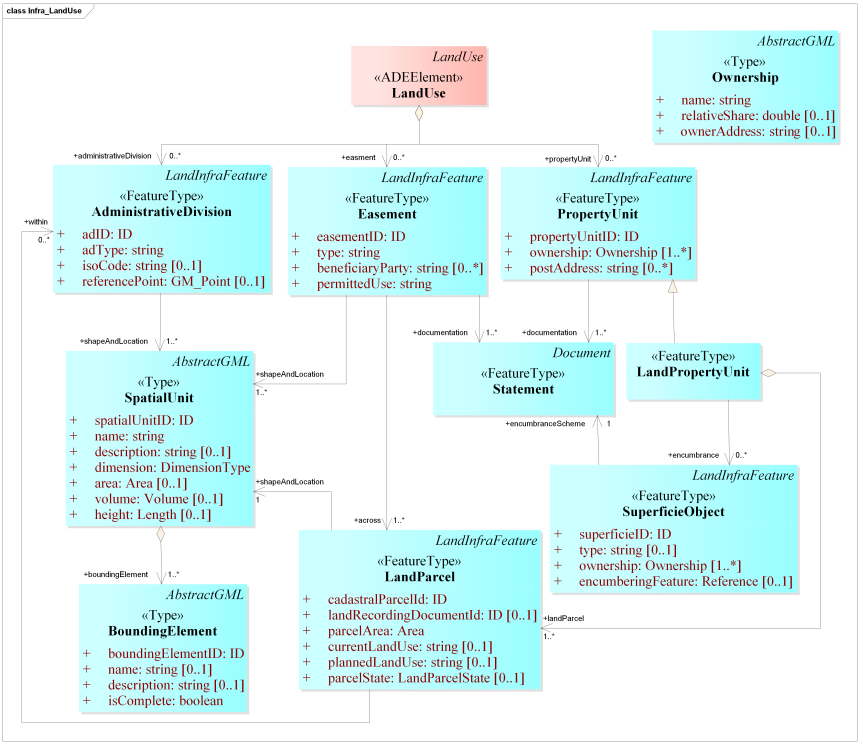


Figure 7.4: The UML excerpt depicts existing CityGML `CG::LandUse` (marked with stereotype «ADEElement») extended to include attributes and classes from LandInfra `LI::LandDivision`, in the CityGML Infra ADE.

is a concrete class and *CG::_CityObject* is an abstract class, we introduced `LandInfraFeature` as a concrete class in our ADE (Figure 7.1).

LandInfra `LI::Document` is introduced to store documents with information about the datasets, e.g. statements, condominium schemes, etc. (Figure 7.1). Similarly, LandInfra's `LI::SurveyMark` is defined to store points on the surface of the Earth which are stable during surveying operations.

LandInfra `LI::Facility` and `LI::FacilityPart` are introduced as the subclasses of `LandInfraFeature` to represent the infrastructure facilities in the Infra ADE. Further, to store the activities related to the improvement of facilities, such as design and/or construction, we introduced the LandInfra `LI::Project` and `LI::ProjectyPart` feature types in the ADE (Figure 7.1).

LandInfra's `LI::Alignment` and its associated classes are introduced in the ADE to provide a linear referencing system for locating the features, e.g. an alignment for the centreline of a road, alignment for rails, etc. (Figure 7.5). An alignment can be represented as:

- a simple 2D line string (`infra:lineString2DRepresentation`),
- a horizontal alignment (`infra:Alignment2DHorizontal`),
- a horizontal alignment with an accompanying 2D vertical long section taken along the horizontal alignment (`infra:Alignment2DVertical`),
- or a 3D line string (`infra:lineString3DRepresentation`).

LandInfra `LI::CondominiumBuilding`, `LI::CondominiumBuildingPart` and other associated classes are introduced in the Infra ADE. The LandInfra abstract class *LI::_ LandLayer* is introduced as it is to represent the layers underneath the land surface (Figure 7.6). They can be defined in three ways: as a 3D polyface mesh solid (`infra:SolidLayer`), as a collection of surface layers (`infra:SurfaceLayer`), or as a series of 2D vertical cross sections (`infra:LinearLayer`). Lastly, the LandInfra `LI::Survey` is introduced in the ADE to model information related to the acquisition of geometry and semantic properties of features.

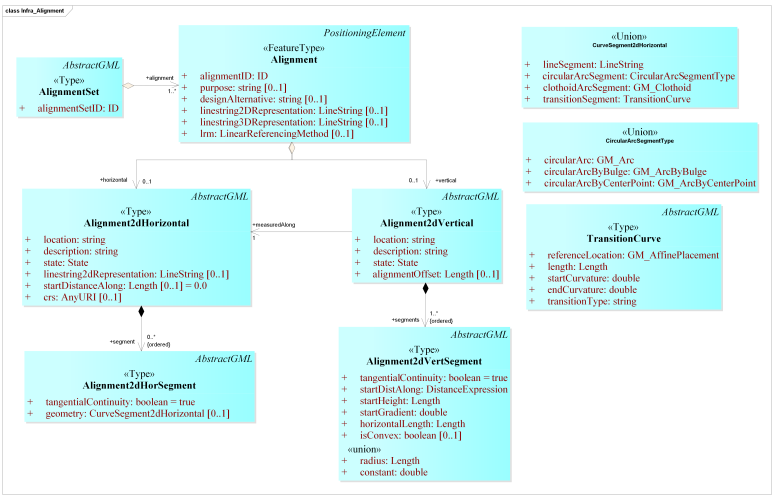


Figure 7.5: The UML excerpt depicts the `Infra_Alignment` introduced in the Infra ADE for linear referencing of the features.

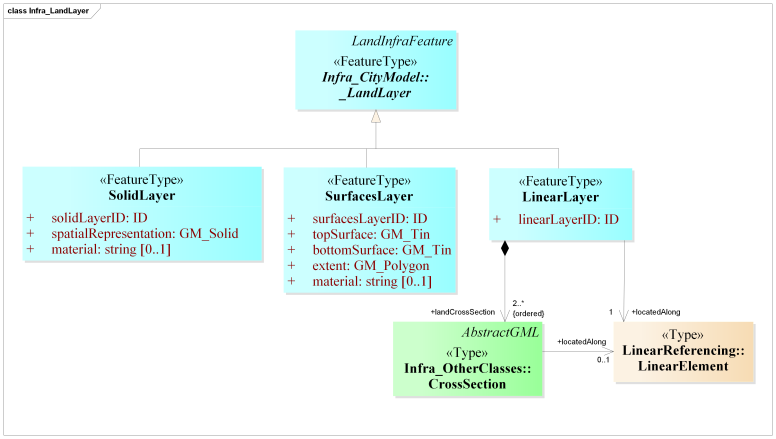


Figure 7.6: The UML excerpt depicts new LandInfra feature ***_LandLayer*** introduced in the CityGML Infra ADE to represent the layers underneath the land surface.

7.3.3 New geometry types, data types and codelists

Three new LandInfra-specific geometry types are introduced in the CityGML Infra ADE, namely `IndexedPoint`, `SimpleIndexedPolygon`, and `PolyfaceMesh`. These geometry types do not exist in the ISO 19107 and in GML. Given that existing software (FME [221], FZK viewer [126], etc.) would require additional implementation to support and visualize these new geometry types, we made their implementation optional in the ADE (in case there is support in the future). It can be difficult to extend software support for each and every ADE that defines new geometry types. Therefore, while converting from InfraGML to CityGML Infra ADE using our software prototype, these geometry types are for now converted to the existing OGC Simple Feature structure supported in GML.

Further, one new LandInfra data type, `ID` is introduced. The `ID` data type is defined to uniquely identify the features within the scope of the dataset. It has an attribute `identifier` which is a user defined ID unique within the dataset or globally unique with the inclusion of `scope` attribute (see snippet below).

```
<li:ID>
    <li:identifier>GML_e8e7963f-718c-40fb-8253-753f2d468f0f</li:identifier>
    <li:scope>OGC LandInfraSWG</li:scope>
</li:ID>
```

We also defined 18 new codelists that were taken from LandInfra, which are summarised in Table 7.2. They are implemented as simple dictionaries according to the CityGML specifications and can be further extended. It is interesting to note that the codelist used to identify the type of easement (`EasementType`) in the LandInfra requirement class `LandDivision` is missing in the LandInfra specifications. It is therefore not included in the codelists defined for the ADE. We also defined two enumerations taken from LandInfra: `Side` and `StringDirectionType`.

7.4 Implementation and Testing

7.4.1 Software prototypes

In order to test the ADE and show its usability, we have developed an open source prototype that automatically converts datasets from InfraGML to CityGML (with our Infra ADE), and vice versa. For the conversion from CityGML to InfraGML, since InfraGML does not offer the possibility to extend its core model, only the classes and attributes that can be mapped are converted.

The software we have developed, together with sample datasets, is freely available in the GitHub repository: `https://github.com/tudelft3d/city2InfraGML`. It is composed of two Python scripts:

1. *citygml2infragml.py* for converting original CityGML models to InfraGML.
2. *infragml2citygml.py* for converting InfraGML models to CityGML (with Infra ADE).

In addition, we developed a validator that checks an InfraGML dataset against the schema[2]. It can be combined with val3dity[3] to validate the geometry of the 3D primitives according to the international standard ISO 19107 [146, 149]. For the validator, we introduced an additional wrapper schema for specifically validating different LandInfra features (e.g. terrain, facilities, roads, etc.) within a single dataset.

7.4.2 Experiments and validation

We tested our software with various real world datasets in the Netherlands (Figure 7.7) and validated the results by checking them against the schema using the validator. The examples of XML in the following sections are taken from the real-world datasets that we have created.

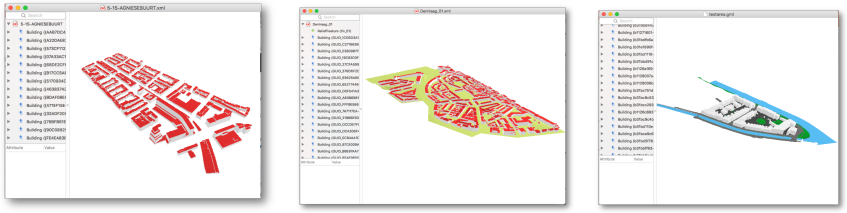
[2] `https://github.com/tudelft3d/city2InfraGML`
[3] `https://github.com/tudelft3d/val3dity`

Table 7.2: Description of codelists included in the Infra ADE

#	Codelist	Description
1	State	Whether an object is existing or proposed.
2	Status	Life cycle stage of an object, e.g. planned, under construction, etc.
3	ProfessionalType	Position of the person in charge of the land development and infrastructure project e.g. draftsman, engineer, surveyor, etc.
4	FacilityPartType	e.g. building, road, etc.
5	LandElementType	e.g. vegetation, terrain, etc.
6	RoadElementType	e.g. pavement, gutter, etc.
7	RailwayElementType	e.g. switch, rail, etc.
8	LandParcelState	e.g. main parcel or carrier parcel, etc.
9	SurveyMonumentType	e.g. boundary marker, observation point of geodetic significance, etc.
10	StatementType	Statement that is signed to establish the interest in the land.
11	SigningRole	Role a signing party plays, e.g. owner, buyer, seller, etc.
12	CondominiumUseType	e.g. residential, office, etc.
13	BuildingPartType	e.g. the main part to which the postal address refers to, or a secondary part like the basement of a shop, etc.
14	StringType	Geometric string representation of bounding element `BEString`.
15	DimensionType	Dimension of a spatial unit. Depending on the value, a spatial unit can include attributes such as area, volume and height.
16	ImplicitSurfaceType	Top or bottom surface of a spatial unit for the bounding element `BEImplicitFace`.
17	SurveyType	e.g. compiled, computed or actually surveyed, etc.
18	SurveyResultType	e.g. point, point cloud, image, etc.

The tested datasets were the following:

1. From CityGML to InfraGML:
   a) CityGML 2.0.0 LOD2 city model of an area in Rotterdam, the Netherlands (Source: http://rotterdamopendata.nl/dataset/rotterdam-3d-bestanden).
   b) CityGML 1.0.0 LOD2 models of building and terrain of an area in the Hague, the Netherlands (Source: https://data.overheid.nl/data/dataset/48265-3d-lod2-stadsmodel-2010-den-haag-citygml).
   c) CityGML 2.0.0 LOD1 city model of an area of Delft, the Netherlands generated using 3dfier[4] (Source: https://3d.bk.tudelft.nl/opendata/3dfier/).
2. From InfraGML to CityGML:
   a) Sample InfraGML 1.0.0 datasets of land surfaces, facilities and roads.
   b) There are no publicly available real-world InfraGML datasets. Therefore, we used the 3D city models of an area in Delft, Rotterdam, and the Hague, the Netherlands generated by converting original CityGML LOD1 and LOD2 models to InfraGML using *citygml2-infragml.py*. These generated InfraGML datasets are publicly available in our aforementioned GitHub repository.



(a) Rotterdam **(b)** Den Haag **(c)** Delft

Figure 7.7: CityGML datasets used for testing the prototypes, visualised in azul.

7.5 Use cases for the CityGML Infra ADE

Some use cases for LandInfra are included in the official documentation [188] of the standard: road alignments, surveying, conversions between LandXML and InfraGML, storage of terrain data, land division, and representation of railway

[4] https://github.com/tudelft3d/3dfier

features. However, these are described only at a superficial level, vaguely explaining broad cases where the standard could be used for (rather than how) and omitting any technical details. Moreover, it is a relatively new standard, and at present, there are not any concrete examples of its usage in real world applications. Blanchet et al. [26] investigated whether CityGML or InfraGML is best suited for initial environment acoustic studies, but the research was limited to a conceptual study of the LandInfra standard, and real world InfraGML datasets were not available.

That being said, we believe that there is potential for LandInfra in many areas. For instance, buildings are currently the main focus of the integration of BIM and GIS [3], while other features, e.g. terrain, vegetation, roads, water bodies, bridges, etc. are often ignored. This is something that can change with LandInfra, since it covers all the aforementioned city features and provides extensive semantic information for land and infrastructure features. As a way to contribute to this discussion, we present here a few additional potential use cases where LandInfra and CityGML Infra ADE can be useful in practice:

1. **Subsurface modelling**

   Data about the geological subsurface such as type of soil, its porosity and depth, bedrock layers, etc. provides important information about the conditions of the ground. This data is of crucial interest for projects which involve shallow or deep digging of the ground, such as building construction, excavation, etc. [259]. By including such information in the design stage, risks of accidents can be better handled and costs can be reduced significantly. The abundance of aboveground 3D city models often overlooks the fact that the cities and their infrastructures are not lying on a "flying carpet". We need holistic modelling of cities in 3D with integrated subsurface information.

   GIS standards (such as the OGC standard CityGML) or BIM standards (such as IFC) are not designed to work with real world subsurface data originating from the 3D geological models [259], even if some work has been done to model such information in integrated models [76, 238], as well as in IFC [59]. For instance, CityGML originally does not model real-world subsurface data originating from the geological models [259], which is useful for many applications, such as infrastructural works that require excavations and soil studies. Since LandInfra has support for modelling topography (terrain) and subsurface information in its requirement class `LandFeature`, our Infra ADE enables the modelling of surface features (such as buildings, roads, etc.) with subsurface information in an integrated framework (see snippet 1 below for implementation in an Infra ADE dataset). The subsurface layers can be represented in three ways in the Infra ADE: as TINs, 3D polyface mesh solids, or vertical 2D cross sections. Each subsurface

layer can have an additional attribute `material` to specify the material of the layer. This integrated framework will not only benefit the planning and design process for surface and subsurface structure construction, but also make transparent the risk management.

Snippet 1: CityGML Infra ADE with an LOD1 Building and a subsurface layer

```
<CityModel>
  <!-- CityGML LOD1 Building with Infra ADE attributes -->
  <cityObjectMember>
    <bldg:Building gml:id="building01">
      <!-- CityGML attributes -->
      <gml:name>CityGML Infra ADE LOD1 Building </gml:name>
      <creationDate>2016-11-24</creationDate>
      <bldg:function>1004</bldg:function>
      <bldg:measuredHeight uom="m">4.12</bldg:measuredHeight>
      <bldg:lod1Solid/>
      <bldg:address/>
      <!-- Infra ADE attributes -->
      <infra:buildingID>
        <infra:ID>
          <infra:identifier>building01</infra:identifier>
        </infra:ID>
      </infra:buildingID>
      <infra:buildingState>existing</infra:buildingState>
      <infra:buildingStatus>constructed</infra:buildingStatus>
    </bldg:Building>
  </cityObjectMember>
  <!-- CityGML Infra ADE subsurface layer (SolidLayer) -->
  <cityObjectMember>
    <infra:SolidLayer gml:id="layer01">
      <infra:state>existing</infra:state>
      <infra:spatialRepresentation xlink:href="#pmesh1"/>
      <infra:solidLayerID>
        <infra:ID>
          <infra:identifier>laye01</infra:identifier>
        </infra:ID>
      </infra:solidLayerID>
      <infra:material>clay</infra:material>
    </infra:SolidLayer>
  </cityObjectMember>
</CityModel>
```

2. **3D cadastre**

   The land administration organizations in different countries, such as the Netherlands, Norway, Germany, Australia, have investigated a 3D approach to digitally manage information about the ownership rights of properties/units within building complexes, see e.g. Stoter et al. [233]. Digital management of property interests in 3D mainly requires legal information (ownership, boundaries, is it public/private?) and physical information

(location, semantics, and 3D geometry) about the property [4]. BIM can provide highly detailed 3D physical information about the buildings. However, IFC currently lacks a standardised way to internally represent the legal information of a building site encompassing many properties, such as condominium boundaries, which is the core of land administration information [4]. CityGML can provide physical information about the buildings and other surrounding features such as terrain, roads, tunnels, but the representation of legal extents and rights is not explicitly covered in the standard.

There has been significant amount of research over the past decade on the integration of legal information with 3D physical models for the management of property rights. For instance, Rönsdorff et al. [219] proposed the CityGML LADM ADE (Land Administration Domain Model) to represent the legal ownership of buildings and their parts in CityGML in accordance with ISO 19152-LADM standard [104]. Similarly, Atazadeh et al. [4] proposed an extension to IFC to manage legal information about the buildings in 3D. However, most of the available land administration research with IFC and CityGML is centred around buildings.

LandInfra is more than LADM. LandInfra addresses land development in the context of activities concerning civil engineering infrastructure facilities [188]. This is achieved by modelling what is needed to account for such activities, including defining the legal entities, their boundaries, as well as identification of the signing parties [188]. *LandDivision* is one of the requirement classes of LandInfra. As mentioned in the LandInfra specifications, *the scope of LandInfra does not include land recording and database storage*. The LandInfra Standard addresses only a subset of LADM [188]. The integration of LandInfra with CityGML in our Infra ADE further enables modelling administrative divisions, cadastral information and ownership rights of condominiums, and subsurface infrastructure such as underground tunnels (Snippet 2).

Snippet 2: CityGML LandUse extended with divisions in Infra ADE

```
<cityObjectMember>
<!-- CityGML LandUse -->
  <luse:LandUse gml:id="luse01">
  <!-- Infra AdministrativeDivision -->
    <infra:administrativeDivision>
      <infra:AdministrativeDivision gml:id="admin01">
        <infra:state>existing</infra:state>
        <infra:adID>
          <infra:ID>
            <infra:identifier>admin01</infra:identifier>
          </infra:ID>
        </infra:adID>
```

```
        <infra:adType>Municipality</infra:adType>
        <infra:shapeAndLocation>
          <infra:SpatialUnit gml:id="su01">
          <infra:spatialUnitID>
            <infra:ID>
              <infra:identifier>su01</infra:identifier>
            </infra:ID>
          </infra:spatialUnitID>
          <infra:dimension>3D</infra:dimension>
          <infra:boundingElement>
          .... <!-- Any of the Bounding elements -->
          </infra:boundingElement>
          </infra:SpatialUnit>
        </infra:shapeAndLocation>
      </infra:AdministrativeDivision>
    </infra:administrativeDivision>
  </luse:LandUse>
</cityObjectMember>
```

3. **Urban facility management**

   Currently, most of the research related to facility management is confined to buildings. For instance, Kim et al. [124] implemented the CityGML Indoor ADE to implement indoor space and indoor facility management applications for buildings. Similarly, the CityGML CAFM (Computer Aided Facility Management) ADE was developed by Moshrefzadeh et al. [168] to integrate detailed geometric and semantic information on the outer shell of the buildings for applications like cleaning management, and cost planning and management.

   In LandInfra, a facility includes buildings and other infrastructure, such as roads, railways, runways, waste water system, bridge, utilities (pipelines), etc. [188]. The integration of LandInfra with CityGML in our Infra ADE enables effective management of all the aforementioned facilities in CityGML. Each facility, whether a building or a road, has a life cycle, including planning, design, construction, maintenance, operation, and termination phases. Furthermore, a facility may be broken down into parts (`FacilityPart`), e.g. a shopping mall may include buildings, roads, site, drainage, water distribution and waste water [188].

   Any activity such as the design or construction related to a facility (or its parts) is managed through projects (`Project/ProjectPart`). The CityGML Infra ADE dataset can include any number of projects to store the status of the facility project (`projectStatus`) and the date on which the status value is valid (`statusDate`) to make the dataset more manageable (Snippet 3).

Snippet 3: CityGML Infra ADE with project data for facility management

```
<CityModel>
  <!-- CityGML Infra ADE facility with two parts: Building & Road-->
  <cityObjectMember>
    <infra:Facility gml:id="facility01">
      <infra:state>existing</infra:state>
      <infra:facilityID>
        <infra:ID>
          <infra:identifier>facility1</infra:identifier>
        </infra:ID>
      </infra:facilityID>
      <infra:type>shopping mall</infra:type>
      <infra:status>under constructed</infra:status>
      <infra:part xlink:href="#Building01"/>
      <infra:part xlink:href="#Road01"/>
    </infra:Facility>
  </cityObjectMember>
  <!-- CityGML Building -->
  <cityObjectMember>
    <bldg:Building gml:id="Building01">
    ....
    </bldg:Building>
  </cityObjectMember>
  <!-- CityGML Road -->
  <cityObjectMember>
    <tran:Road gml:id="Road01">
    ....
    </tran:Road>
  </cityObjectMember>
  <!-- Infra ADE Project for the facility-->
  <cityObjectMember>
    <infra:Project gml:id="Project01">
      <infra:state>existing</infra:state>
      <infra:projectID>
        <infra:ID>
          <infra:identifier>Project01</infra:identifier>
        </infra:ID>
      </infra:projectID>
      <infra:projectStatus>under construction</infra:projectStatus>
      <infra:statusDate>2019-01-01</infra:statusDate>
      <!-- Infra ADE ProjectPart for the facility part Building -->
      <infra:projectPart>
        <infra:ProjectPart gml:id="ProjectPart1">
          <infra:state>existing</infra:state>
          <infra:projectPartID>
            <infra:ID>
              <infra:identifier>ProjectPart1</infra:identifier>
            </infra:ID>
          </infra:projectPartID>
          <infra:status>constructed</infra:status>
          <infra:statusDate>2019-01-01</infra:statusDate>
          <infra:facilityPart xlink:href="Building01"/>
        </infra:ProjectPart>
```

```
      </infra:projectPart>
      <!-- Infra ADE ProjectPart for the facility part Road -->
      <infra:projectPart>
        <infra:ProjectPart gml:id="ProjectPart2">
          <infra:state>existing</infra:state>
          <infra:projectPartID>
            <infra:ID>
              <infra:identifier>ProjectPart2</infra:identifier>
            </infra:ID>
          </infra:projectPartID>
          <infra:status>under construction</infra:status>
          <infra:statusDate>2019-01-01</infra:statusDate>
          <infra:facilityPart xlink:href="Road01"/>
        </infra:ProjectPart>
      </infra:projectPart>
    </tran:Road>
  </cityObjectMember>
</CityModel>
```

4. **Surveying**

   Biljecki et al. [24] highlighted that the choices made by the practitioners when acquiring and processing data for generating 3D city models are rarely documented in a dataset, mostly because there is no standardised way of storing it. Survey data can be used as a reliable data source at all the stages of the life cycle of a building or other features. Designers of architectural and design projects, armed with accurate site data, can work with reduced overall commercial risk, and with greater certainty. GIS standards (such as the OGC standard CityGML) or BIM standards (such as IFC) do not offer a mechanism to store such data in a structured way. LandInfra has a requirement class `Survey` outlining the specifications to store such data. It is based on the OGC standard SensorML (Sensor Model Language) [187]. The integration of LandInfra with CityGML in our Infra ADE enables effective management of survey metadata in CityGML, e.g. survey type and its purpose, surveyor information, and so on. Further, it is possible to store information about the survey observation points, accuracy information, equipments or sensors used, and the results.

Snippet 4: CityGML Infra ADE with Survey data

```
<CityModel>
  <infra:survey>
    <infra:Survey gml:id="Survey01">
      <infra:surveyID>
        <infra:ID>
          <infra:identifier>Survey01</infra:identifier>
        </infra:ID>
      </infra:surveyID>
      <infra:name>SurveyData</infra:name>
```

```
      <infra:description>Survey dataset</infra:description>
      <infra:type>computed</infra:type>
      <!-- Field notes taken during the survey -->
      <infra:fieldNote>
        <infra:Document gml:id="SurveyDoc01">
          <infra:state>existing</infra:state>
          <infra:documentID>
            <infra:ID>
              <infra:identifier>SurveyDoc01</infra:identifier>
            </infra:ID>
          </infra:documentID>
          <infra:documentType>Open Data</infra:documentType>
          <infra:documentContent>doc.pdf</infra:documentContent>
        </infra:Document>
      </infra:fieldNote>
      <infra:setup>
      .... <!-- Survey setup data -->
      </infra:setup>
      <infra:equipment>
      .... <!-- Equipments used for surveying -->
      </infra:equipment>
      <infra:surveyResult>
      .... <!-- Results of the survey here -->
      </infra:surveyResult>
    </infra:Survey>
  </infra:survey>
</CityModel>
```

5. **Asset management**

   Asset management can be summarised as a systematic approach to the process of maintaining, upgrading and operating physical assets in a cost-effective way for both short- and long-term planning [165]. For municipalities and regional governments asset management is a crucial element of day-to-day operations. Within asset management, the maintenance of roads and transportation networks is key to keeping traffic moving smoothly and safely on a daily basis. Road maintenance includes activities such as smoothness control/de-icing, repairs, closures, milestone maintenance and traffic city furniture replacement [144].

   While CityGML has support for LODs, it only supports line representation at LOD0 and polygon representation at LOD1-LOD4. InfraGML supports four representations for roads: solid, faceted (triangular) surfaces, lines running longitudinally and 2D views cut perpendicular to a road's centreline. It is also possible to model the cross-section of a road, which is valuable for repair projects. Furthermore, CityGML has support for railways in its `Transportation` module, but it has little support and almost no documentation. It is also unclear how to exactly model a railway in CityGML, as it is also mentioned as being a part of the `Tunnel` and

`Bridge` modules. With InfraGML, there is a dedicated class for railways and this has support for 3D railway elements and track geometry including superelevation (cant). The Snippet 5 summarises the potential additional elements from LandInfra that can enhance CityGML data for road asset management.

6. **Urban environmental analysis**

The added value of integrating CityGML and LandInfra in our Infra ADE can also be seen in urban applications such as estimating the level of noise exposure on buildings, or how much solar irradiation a building will receive. Unlike CityGML, LandInfra explicitly models the materials of road surfaces and terrain, geometry and semantics of railways, type of road elements such as pavements, hard shoulders, soft shoulders, etc., construction materials of buildings, and information about the observation/measurement points to name a few. Such information is useful for environmental applications such as urban noise and flood mapping. We have added all these elements in the Infra ADE to supplement environmental analysis using CityGML.

Snippet 5: CityGML Infra ADE with project data for road asset management

```
<CityModel>
  <cityObjectMember>
    <tran:Road gml:id="road_01">
      <gml:name>Main Street</gml:name>
      <creationDate>2016-11-24</core:creationDate>
      <tran:class>350000</tran:class>
      <tran:function>1</tran:function>
      <tran:usage>2</tran:usage>
      <tran:lod2MultiSurface/> <!-- Geometry Here -->
      <infra:roadID>
        <infra:ID>
          <infra:identifier>road_01</infra:identifier>
        </infra:ID>
      </infra:roadID>
      <infra:approximateWidth uom="m">5</infra:approximateWidth>
      <infra:roadElement xlink:href="#re1"/>
      <infra:roadAlignment xlink:href="#Alignment1"/>
      <infra:roadElement>
        <infra:RoadElement>
          <gml:description>6.5 cm asphalt top </gml:description>
          <gml:name>top pavement layer</gml:name>
          <infra:state>existing</infra:state>
          <infra:spatialRepresentation>
            .... <!-- Spatial Representation Here -->
          </infra:spatialRepresentation>
          <infra:roadElementID>
            <infra:ID>
              <infra:identifier>pavement1</infra:identifier>
            </infra:ID>
```

```
        </infra:roadElementID>
        <infra:roadElementType>Pavement</infra:roadElementType>
        <infra:material>asphalt</infra:material>
      </infra:RoadElement>
    </infra:roadElement>
  </tran:Road>
 </cityObjectMember>
</CityModel>
```

7.6 Conclusions and future work

LandInfra is a more powerful standard than CityGML in some areas, as it has a much more detailed representation for land and infrastructure features. However, it currently has essentially no support in software, and even the academic papers, that we have mentioned in Chapter 6, which touch upon the theoretical potential of LandInfra, do not use it in practice. The Infra ADE for CityGML developed in this the chapter provides a way to change this situation by embedding LandInfra's features in CityGML. This way, we can use the best of both standards, and we can also ensure that the resulting datasets can be used in practice by the software packages already supporting CityGML. There is support available for the CityGML extension mechanism such as parsers, validators, DBMS, and so on. One example is the latest version of *3DCityDB* (3D City Database 4.0.0), which offers support to store CityGML files having ADEs in a database. A CityGML ADE is handled like a 'plugin' and the 3DCityDB core database schema is extended dynamically with new tables based on the schemas of the ADE [254, 255]. Similarly, *citygml4j* [174], a Java API for CityGML, supports reading and writing CityGML ADE datasets. Further, it is also possible to visualize the ADE datasets with new city objects (with GML geometries) and semantic attributes using *FME* [221] and *FZK viewer* [126]. As discussed before, these software would require additional implementation to support and visualize any new geometry types (other than GML) introduced in an ADE dataset.

In order to develop the Infra ADE, we have performed a detailed analysis of the individual classes, attributes and relations in CityGML and LandInfra, and created a mapping from LandInfra to CityGML. We have mapped LandInfra to CityGML (and not vice versa) because CityGML provides mechanisms to extend its data model with new feature types and attributes using Generic city objects or ADEs, whereas similar extensions are not supported in the LandInfra standard. Moreover, CityGML has the concept of LODs, which is widely used in practice and missing in LandInfra. To provide a proof-of-concept of our mapping, I have developed two open source software prototypes for converting CityGML (and Infra ADE) datasets to InfraGML and vice versa. Since LandInfra is a relatively new standard and there are no datasets available for it, the developed prototypes can help practitioners to generate valid real-world sample InfraGML

datasets, which can then lead to the real-world applications that are currently missing from the standard.

Part IV

3D data modelling and discovery for the built environment applications

Chapter 8

ISO-19115 compliant metadata for 3D data discovery and management

This chapter is based on the paper:

Labetski A, Kumar K, Ledoux H, and Stoter J, 2018. A metadata ADE for CityGML. *Open Geospatial Data, Software and Standards*, 3(1), pp.16. doi: https://doi.org/10.1186/s40965-018-0057-4

8.1 Introduction

With more and more 3D city models being created and used, special emphasis has to be laid on their *discovery and dissemination*, i.e. the users can discover relevant 3D city models for a specific application. The metadata can play a key role in the retrieval and dissemination of the 3D city models. The boundary between data and metadata can be a fuzzy one but one distinction that we use in this thesis is: metadata is structured to some degree and this structuring is what converts "raw information into actionable metadata" [60, 216]. The use of metadata to describe and document 3D datasets has several other advantages. First, it can ensure that data creators and data users from different 3D domains can understand and communicate about data requirements and its usability [164]. Second, the presence of metadata is as crucial for achieving transparency as the presence of bibliography in an academic print publication [129].

In GIS, the international metadata standard ISO 19115 is relatively mature, several GIS software implement it, and is used by practitioners. Its use is however mostly restrained to 2D datasets (both raster and vector), 3D datasets very rarely have metadata information stored. One cause is probably because, as highlighted by Danko [47], the specifications of ISO 19115 do not cover several aspects specific to 3D datasets. Furthermore, CityGML, the international standard for storing 3D city models [185], offers no mechanisms to store metadata in a structured way. Practitioners often need to define their own definitions for CityGML metadata and create their own methodology for storing it [91, 247]. CityGML files would most benefit from having (structured) metadata explicitly stored: in practice the size and the complexity of a given CityGML file are way larger and higher than a 2D dataset of the same region. This means that parsing an unknown CityGML dataset to extract information is no easy task for practitioners. Currently, it can seen that many people write their own parsers and a great deal of time is lost on simply analysing a dataset to understand what it contains; examples are the bounding box, the year of creation, the levels of detail in the dataset, etc. Also, because they are too big, many real-world datasets can simply not be opened by text editors, let alone be visualised by a GIS viewer. While several datasets are tiled into sub-parts, e.g. the openly available dataset of Berlin[1], they are still very large in size, often >1GB for a single tile excluding texture information. Metadata is also necessary in the development of CityGML extensions i.e. ADEs (see Chapter 2) in order to track and manage the diversity of data sources and data qualities [181]. Having metadata attached to the CityGML file would help in assessing the fitness-for-purpose of data for use within a specific application.

In this chapter, we examine the metadata needs specific to 3D geospatial datasets

[1] http://www.businesslocationcenter.de/en/downloadportal

and present an ISO 19115 compliant solution to add metadata to the 3D city models represented in CityGML. We built upon the work of Dietze et al. [60] and propose a set of metadata categories, which are ISO 19115 compliant, that ought to be stored. We developed a CityGML extension, the 3D Metadata ADE, so that these can be easily included in the data model of the CityGML in a structured manner. Furthermore, the ADE allows to store metadata for existing city objects of CityGML, as well as for other ADEs. That is, one could define specific metadata elements in her ADE (let say for energy or noise) and use the metadata ADE to store these specific metadata; we believe to be the first to allow an ADE to use (be used by) another ADEs.

8.2 Background

8.2.1 ISO 19115

While there exist multiple metadata standards, such as the American Content Standard for Digital Geospatial Metadata[2] (CSGDM) and the European INSPIRE Metadata Directive[3], their usage is being phased out in favour of ISO 19115 due to its international focus [47]. ISO 19115 is the metadata standard specifically for *geographic information* developed by the ISO. It (latest revision is from 2014) defines mandatory, conditional and optional metadata attributes such as dataset title, responsible party and conditions of use [108]. The standard also provides guidance for the minimum set of metadata attributes required to serve most metadata applications, these are: data discovery, determining data fitness for use, data access, data transfer, and use of digital data and services [108]. It is composed of 13 packages, individual packages may be used alone to provide separate components of metadata to meet specific use case requirements [108].

The previous version of ISO 19115 (2003 and including later revisions) did not provide any encoding and an XML encoding was specified in ISO 19139 [102]. With the 2014 version of ISO 19115 there was a re-definition of the standard by splitting it into 3 parts: Part1 - *Fundamentals*, Part 2 - *Extensions for imagery and gridded data*, and Part 3 - *XML schema implementation of metadata fundamentals* [108]. Part 3 was published in 2016 and defines XML schemas for encoding Parts 1 and 2, effectively superseding ISO 19139 [112].

The first function of metadata is data discovery, Table 8.1 outlines the *'Metadata for the discovery of geographic datasets and series'* as defined by ISO [108]. The discovery list of attributes matches the attributes that were previously part of the metadata *core* concept present in previous versions of ISO 19115 and

[2] http://www.fgdc.gov/metadata/csdgm
[3] http://www.inspire.ec.europa.eu/metadata

are therefore the attributes most commonly associated with the standard and featured in the metadata packages of many GISs. This is why the metadata proposal in this chapter also focuses on meeting the requirements necessary for discovering 3D city model datasets.

#	Metadata element	Obligation	Comment
1	Metadata reference information	Optional	Unique identifier for the metadata.
2	Resource title	Mandatory	Title by which the resource is known.
3	Resource reference data	Optional	A date which is used to help identify the resource.
4	Resource identifier	Optional	Unique identifier for the resource.
5	Resource point of contact	Optional	Name of the person, position, or organisation responsible for the resource.
6	Geographic location	Conditional*	Geographic description of coordinates (latitude/longitude) which describes the location of the resource.
7	Resource language	Conditional	The language and character set used in the resource.
8	Resource topic category	Conditional	A selection of the 20 elements in the MD_TopicCategory enumeration which describe the topic of the resource.
9	Spatial resolution	Optional	The nominal scale and/or/spatial resolution of the resource.
10	Resource type	Conditional	A resource code identifying the type of resource.
11	Resource abstract	Mandatory	A brief description of the content of the resource.
12	Extent information for the dataset (additional)	Optional	The temporal or vertical extent of the resource.
13	Resource lineage	Optional	A description of the source(s) and production process(es) used in producing the resource.
14	Resource on-line Link	Optional	Link (URL) in the metadata for the resource.
15	Keywords	Optional	Words or phrases describing the resource to be indexed and searched.
16	Constraints on resource access and use	Optional	Restrictions on the access and use of the resources.
17	Metadata date stamp	Mandatory	Reference date(s) for the metadata, especially creation.
18	Metadata point of contact	Mandatory	The party responsible for the metadata.

*Conditional means that certain elements become mandatory based on the values of other elements.

Table 8.1: ISO 19115-1:2014 - Table F.1: *Metadata for the discovery of geographic datasets and series* [108].

8.2.2 3D Geospatial Metadata Needs

Dietze et al. [60] examined the applicability of ISO 19115 for 3D city models and found that while there exist several attributes that are important there are further attributes that are missing, the most prominent being the level of detail and semantic object classes (e.g. buildings, roads, etc.). Additionally, Biljecki et al. [24] found that the modelling choices made by practitioners when acquiring, processing, and utilising 3D city models are rarely documented in the metadata of a dataset, often because there is no way to store this information. This information is necessarily not only for dataset discovery but also to ensure interoperability between various 3D city models [256]. A lack of metadata in 3D city models means it is more difficult to integrate them in 3D spatial data infrastructures (SDIs) where metadata is an important base [176].

8.2.3 CityGML and metadata support

CityGML version 2.0.0 has very limited support for metadata and of the limited number of elements that are supported, i.e. name, description, bounding box, and coordinate system, are not stored explicitly as metadata, thus making the integration of CityGML within current resource discovery databases difficult. In practice, such as in the example below [185], the metadata related elements are normally present near the top of the file, almost directly after the `<CityModel>` tag, or sometimes at the very end before the closing of the `<CityModel>` tag.

Snippet 1: Limited metadata support in CityGML

```
<CityModel>
  <gml:name>Simple 3D city model LOD0 without Appearance</gml:name>
  <gml:boundedBy>
    <gml:Envelope srsDimension="3"
    srsName="urn:ogc:def:crs,crs:EPSG::25832,crs:EPSG::5783">
      <gml:lowerCorner>458868.0 5438343.0 112.0</gml:lowerCorner>
      <gml:upperCorner>458892.0 5438362.0 117.0</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
  ....
</CityModel>
```

CityGML inherits the metadata property (which can be information about the author/creator of the dataset, lineage of the dataset, reference system, etc.) from GML but this only hosts very basic attributes and is often not implemented in practice. For example, the `<gml:metaDataProperty>` tag can be utilised to define the specifications necessary in the usage of a local coordinate system (Appendix G.9 in OGC [185]).

As an alternative approach, some CityGML users tend to write adhoc metadata elements as comments in XML, such as the following snippet from a 3D city model of Montreal[4]:

Snippet 2: Adhoc metadata as comments in CityGML

```
<!-- File Written With RhinoCity Software CopyRight Rhinoterraain 2012 -->
```

8.3 Methodology

Our methodology for including the metadata elements required for the discovery of 3D city models included:

1. an analysis of ISO 19115 for appropriate elements,
2. a literature review of 3D city models and their applications to determine necessary elements not currently supported by ISO 19115,
3. a study of the CityGML schema to understand which elements are missing,
4. an analysis of which elements are required based on the hierarchy levels of CityGML, i.e. city model level, thematic module level and feature level,
5. interview software developers about what they like to know to help them in dealing with large files

We utilised Table 8.1 to guide the ISO elements that we have included in the CityGML Metadata ADE, and maintained the same obligation level (i.e. mandatory, conditional or optional). Elements that were not relevant to CityGML were excluded and certain elements were modified to be explicitly 3D. Our inclusions, modifications and exclusions are summarised in Table 8.2.

We decided to use the discovery table in particular and not the full suite of categories for the following reasons:

1. The increasing size of 3D city models tends to be one of the largest deterrents preventing their discovery and usage. Parsing a massive unknown 3D dataset to extract information is not an easy task and a great deal of time is lost while simply analysing a dataset to understand what it contains.
2. Many of the attributes listed in Table 8.1 can already be easily derived from existing models.
3. Duval et al. [66] argues that metadata architects should utilise base schemas to facilitate the interoperability of various metadata packages thus ensuring higher usability with various applications. Table 8.1 has most of the elements that are likely to be found in many metadata packages of many GISs.

[4] http://donnees.ville.montreal.qc.ca/dataset/maquette-numerique-batiments-citygml-lod2-avec-textures

Table 8.2: Our inclusion of ISO 19115 metadata elements for data discovery.

#	Metadata element	Inclusion	Comment
1	Metadata reference information	Included	-
2	Resource title	Included	-
3	Resource reference data	Included	-
4	Resource identifier	Included	-
5	Resource point of contact	Included	-
6	Geographic location	Modified	Coordinate representation is supported in the `Extent` element and therefore geographic location was restricted to a string representation.
7	Resource language	Included	-
8	Resource topic category	Included	-
9	Spatial resolution	Excluded	This category is supported for rasters in the *Relief* module but is not applicable at the city model level.
10	Resource type	Included	-
11	Resource abstract	Included	-
12	Extent information for the dataset (additional)	Modified	This was renamed to `Extent` and follows the `Extent` package in ISO 19115. Modifications include removing *Vertical Extent* as a separate category and instead making *Geographic Extent* explicitly 3D. *Geographic Extent* is given a mandatory obligation and *Temporal Extent* is optional.
13	Resource lineage	Included	-
14	Resource on-line Link	Included	-
15	Keywords	Included	-
16	Constraints on resource access and use	Included	-
17	Metadata date stamp	Included	-
18	Metadata point of contact	Included	-

Building on the elements identified in the literature review, particularly in Dietze et al. [60], as well as assessing the CityGML schemas, we identified several elements that needed to be added to the ISO 19115 elements. These are summarised in Table 8.3. Note that the need to store the datasets used in acquisition and reconstruction as well as the model generation method [60, 256] is supported by the ISO element `Lineage` as described in the next section.

Most 3D city models are generated with a mixture of different methodologies

and data sources and therefore top level metadata is not sufficient, instead there is a need for feature level metadata [60]. For CityGML this means metadata at the city model level, the thematic module level and the feature level. Note that there is often confusion between feature level metadata and feature attributes, attributes contain information about the *feature* while metadata contains information about the feature *data*. Attributes are already supported by CityGML: each feature has the attributes *class*, *function*, and *usage*. While feature-level metadata is currently not supported and is necessary for instances of lineage. The following section describes all of the above metadata values in greater detail and explains the hierarchy level at which it is implemented.

Table 8.3: Additional metadata elements for 3D city models we include in our ADE.

#	Metadata element	Description	Source
1	Levels Of Detail (LODs)	This includes the LODs present in the city model and each thematic module with unique and aggregate counts for each. We support the improved specifications of buildings as developed by Biljecki et al. [24]	Biljecki et al. [24], Dietze et al. [60]
2	Semantic Surfaces	The presence or absence of semantic surfaces in objects, e.g. roofs, walls, etc.	Dietze et al. [60]
3	Textures/Materials	The presence or absence of textures and/or materials in a city model which are representation of object surface characteristics	
4	XLinks	The presence or absence of *XLinks* in a city model, these are used to share geometry elements between features.	
5	External References	The presence or absence of external references, these are a reference of a 3D object to its corresponding object in an external data set	
6	Thematic Modules	A list of all thematic modules present in a city model, e.g. `Building`, `Transportation`, etc.	
7	ADEs	A list of the ADEs utilised in the city model and their corresponding metadata	

8.4 The 3D Metadata ADE

The objective of developing this ADE is to store and manage the metadata associated with a 3D city model in the CityGML format. All resources (UML, XSD, and documentation) related to the 3D Metadata ADE are available on our public GitHub repository: https://github.com/tudelft3d/3D_Metadata_ADE

To avoid any conflict with the existing CityGML classes and attributes, the new

3D metadata classes and attributes are defined in a different namespace with identifier 'md'. Our metadata ADE has three main classes: (i) MDcitymodel (ii) _MetadataCityfeatures and (iii) _MetadataHelperClasses.

MDcitymodel is the core class of this ADE which stores the metadata about a CityGML dataset (Figure 8.1). It includes the following attributes (see snippet below for detailed attributes and their values):

- ID of the metadata dataset (md:metadataIdentifier)
- ID of the city model (md:citymodelIdentifier)
- ISO 19115 metadata elements as mentioned in Table 8.1 and 8.2 (md:ISOmetadata)
- metadata about the ADEs present in the dataset (md:ADEmetadata)
- attribute indicating which thematic modules are present in the dataset (md:thematicModules)
- attribute indicating if textures are present in the dataset (md:textures)
- attribute indicating if materials are present in the dataset (md:materials)
- attribute indicating if XLinks are present in the dataset (md:xLinks)
- attribute indicating if external references are present in the dataset (md:externalReferences)
- metadata about the city features present in the dataset e.g. total number of buildings, building parts, levels of details, etc.(md:MDcityfeatures)
- levels of detail present in the dataset e.g. (md:LevelsOfDetail)

The XML snippets in this chapter are taken from the sample data[5] created to implement the proposed 3D Metadata ADE.

Snippet 3: Metadata in the CityGML ADE

```
<md:MDcitymodel>
  <!-- Identifiers of the city model and its metadata -->
  <md:metadataIdentifier>MD_38b566d8-7ea3</md:metadataIdentifier>
  <md:citymodelIdentifier>GML_38bb4326-7ea3</md:citymodelIdentifier>
  <!-- ISO metadata categories -->
  <md:ISOmetadata>
    <md:ISOidentifier>
      <md:datasetTitle>3D Building Model</md:datasetTitle>
      <md:datasetReferenceDate>2018-08-23</md:datasetReferenceDate>
        .......
    </md:ISOidentifier>
  </md:ISOmetadata>
  <!-- Thematic modules in the dataset -->
  <md:thematicModules>
    <md:presentThematicModules>Building</md:presentThematicModules>
  </md:thematicModules>
  <!-- Textures, materials, references, and Xlinks -->
```

[5] https://github.com/tudelft3d/3D_Metadata_ADE/tree/master/Code/citygmldatasets

```
  <md:textures>absent</md:textures>
  <md:materials>absent</md:materials>
  <md:xLinks>present</md:xLinks>
  <md:externalReferences>absent</md:externalReferences>
  <!-- ADE in the dataset -->
  <md:ADEmetadata>
    <md:ADEidentifier>
      <md:adeName>iTINs ADE</md:adeName>
      <md:adeVersion>0.1</md:adeVersion>
      <md:namespace>http://tudelft.nl/iTINs_ADE</md:namespace>
      ....
    </md:ADEidentifier>
  </md:ADEmetadata>
  <!-- City features in the dataset -->
  <md:MDcityfeatures>
    ....
  </md:MDcityfeatures>
  <!-- LODs in the dataset -->
  <md:LevelsOfDetail>
    ....
  </md:LevelsOfDetail>
</md:MDcitymodel>
```

`_MDcityfeature` is an abstract class which defines the metadata classes for different CityGML thematic modules. It defines the attributes information about different city features present in the dataset such as:

- the type of city feature (Building, Vegetation, etc.) (`md:featureType`)
- total number of a specific type of city feature (`md:uniqueFeatureCount`)
- total number of a specific type of city feature if it exists in more than 1 level of detail (`md:aggregateFeatureCount`)
- levels of detail of that specific city feature (`md:LevelsOfDetail`)
- lineage of that specific city feature (`md:featureLineage`)

Apart from the aforementioned metadata elements, `_MDcityfeature` has specialised subclasses (e.g. `md:MDbuilding`, `md:MDbridge`, etc.) (Figure 8.1).

- `md:MDbuilding`. It defines attributes to store metadata about the buildings present in the dataset such as:
  - number of building parts (`md:buildingParts`)
  - number of building installations (`md:buildingInstallations`)

Snippet 4: Metadata about the buildings in the dataset

```
<md:MDcityfeatures>
  <md:MDbuilding>
    <md:featureType>Building</md:featureType>
```

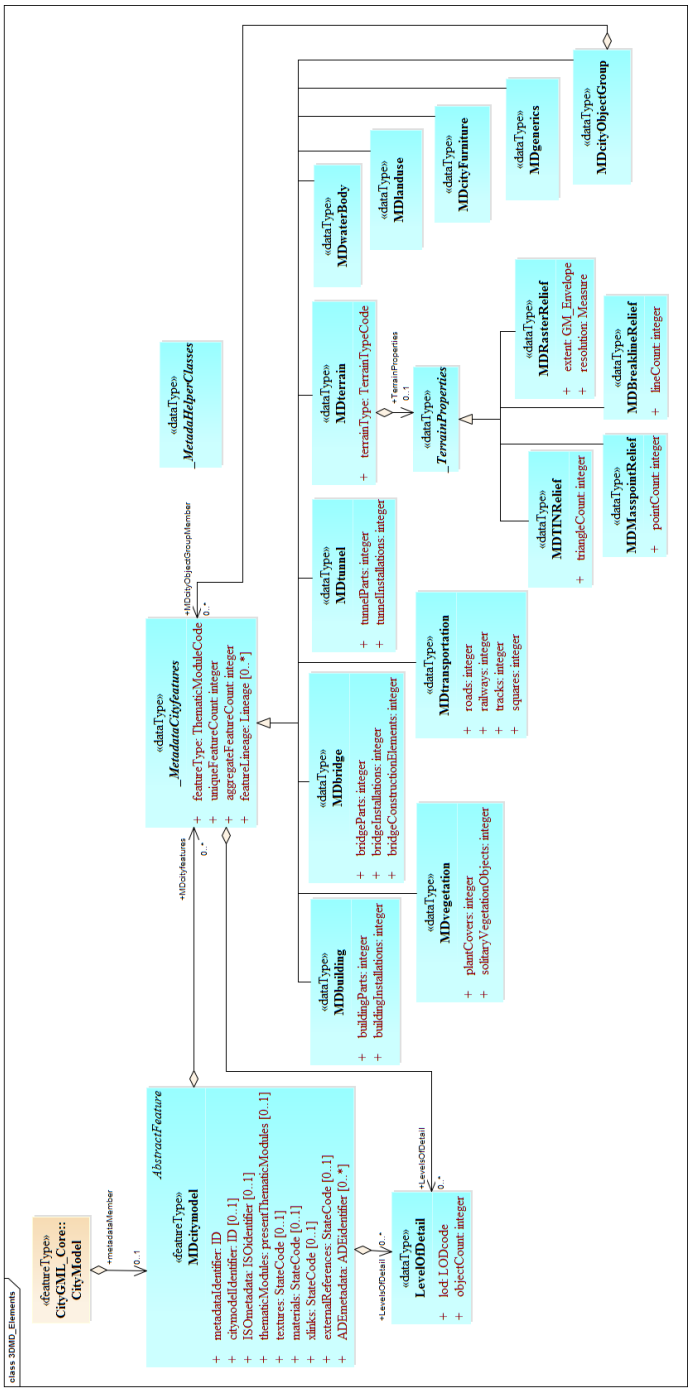


Figure 8.1: UML model of the CityGML 3D Metadata ADE depicting the proposed metadata objects for storing metadata related to 3D city models (`_MetadataObjects`) (shown coloured in blue)

```
    <md:uniqueFeatureCount>2</md:uniqueFeatureCount>
    <md:aggregateFeatureCount>2</md:aggregateFeatureCount>
    <md:LevelsOfDetail>
      <md:LevelOfDetail>
        <md:lod>2</md:lod>
        <md:objectCount>2</md:objectCount>
      </md:LevelOfDetail>
    </md:LevelsOfDetail>
    <md:buildingParts>2</md:buildingParts>
    <md:buildingInstallations>0</md:buildingInstallations>
  </md:MDbuilding>
</md:MDcityfeatures>
```

- `md:MDbridge`. It defines attributes to store metadata about the bridges present in the dataset such as:
  - number of bridge parts (`md:bridgeParts`)
  - number of bridge installations (`md:bridgeInstallations`)
  - number of bridge construction elements (`md:bridgeConstructionElements`)
- `md:MDtunnel`. It defines attributes to store metadata about the tunnels present in the dataset such as:
  - number of tunnel parts (`md:tunnelParts`)
  - number of tunnel installations (`md:tunnelInstallations`)
- `md:MDtransportation`. It defines attributes to store metadata about the transportation features such as:
  - number of roads (`md:roads`)
  - number of railways (`md:railways`)
  - number of tracks (`md:tracks`)
  - number of squares (`md:squares`)
- `md:MDvegetation`. It defines attributes to store metadata about the vegetation features present in the dataset suc as:
  - number of plant covers (`md:plantCovers`)
  - number of solitary vegetation objects (`md:solitaryVegetationObjects`)
- `md:MDterrain`. It defines attributes to store metadata about the terrain model present in the dataset such as:
  - type of terrain representation (TIN, raster, etc.) (`md:terrainType`)
  - levels of detail of the terrain (`md:LevelsOfDetail`)

Depending on the type of terrain, it is possible to store additional properties (`md:TerrainProperties`) such as the number of triangles (`md:triangleCount`) in case of a TIN or the spatial resolution (`md:resolution`) in case of a raster.

- Similarly, it has `md:MDwaterBody`, `md:MDlanduse`, `md:MDcityFurniture`, `md:MDcityObjectGroup`, and `md:MDgenerics` to store metadata about water bodies, landuse, city object groups, and generic city objects and attributes.

`_MetadataHelperClasses`. It defines the supporting classes and attributes required by `MDcitymodel` (see Figure 8.2). It includes:

- `ISOidentifier`. It defines metadata elements for a city model according to ISO 19115 Table F.1: *Metadata for the discovery of geographic datasets and series* explained in Table 8.1 and 8.2.
- `ADEidentifier`. It defines attributes to store metadata about the ADEs present in the dataset such as: the name of the ADE and its version, URI of the UML and XML schema and any other available documentation.
  - name of the ADE (`md:adeName`)
  - version of the ADE (`md:adeVersion`)
  - namespace of the ADE (`md:namespace`)
  - status of the ADE (`md:status`)
  - authority responsible for the ADE (`md:authority`)
  - short summary about the ADE (`md:summary`)
  - link to its XML schema (`md:xmlSchema`)
  - link to its UML model (`md:umlModel`)
  - link to any additional documentation (`md:documentation>`)
- `LevelOfDetail`. It stores which LODs are present in a city model (`md:lod`) and their count `md:objectCount`). The LoDs are defined in a separate enumeration list (`LODCode`).
- `_Contact`. It stores information (such as name, address, role, etc.) about the person (`md:IndividualContact`) or organization (`md:OrganizationalContact`) reponsible for the dataset.
- `Lineage`. It is possible to store two things with `md:Lineage`:(1) metadata about the data sources (`md:source`) and production steps (`processStep`) used in the generation of the whole dataset, (2) metadata about individual city features e.g. if a dataset has two buildings A & B created by different organizations/authorities using different methods and data (see snippet below).

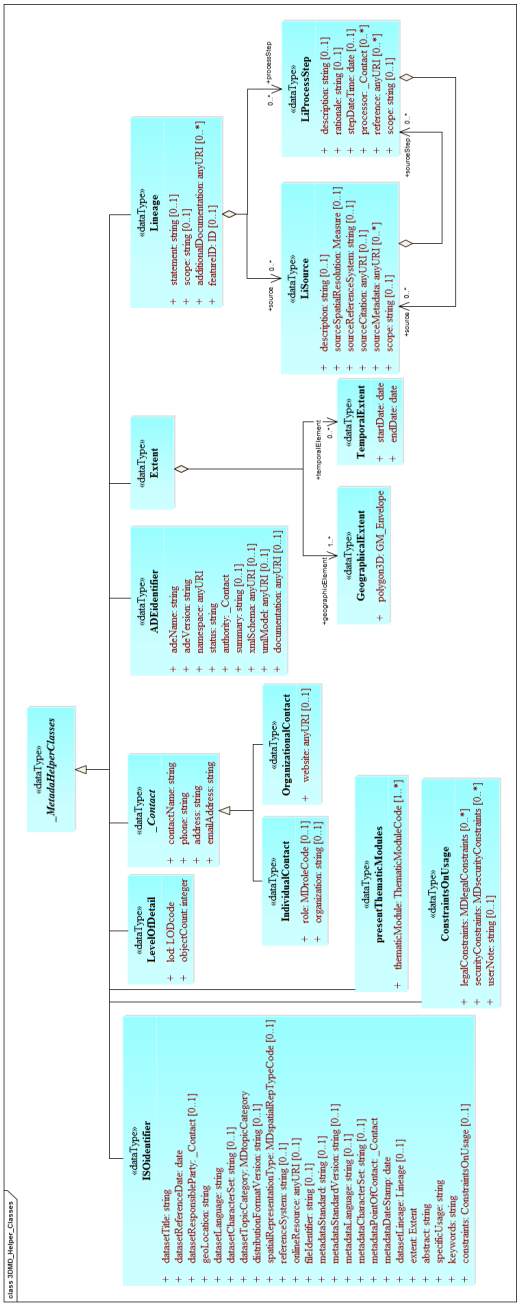


Figure 8.2: UML model of the CityGML 3D Metadata ADE depicting the proposed supporting classes (`_MetadataHelperClasses`) (shown coloured in blue)

Codelists & Enumerations. We defined 5 codelists (taken from ISO 19115) and 4 enumerations (Table 8.4, Figure 8.3). These codelists are implemented as simple dictionaries according to CityGML specifications and can be further extended.

Table 8.4: Description of codelists proposed for the 3D Metadata ADE

Codelist	Description
MDtopicCategory	ISO 19115 codelist of themes (such as environment, atmosphere, climatology) for classification of datasets.
MDroleCode	ISO 19115 codelist of the functions performed by the person responsible for the dataset.
MDlegalConstraints	ISO 19115 codelist of restrictions and legal prerequisites for accessing and using the dataset or metadata.
MDsecurityConstraints	ISO 19115 codelist of the restrictions imposed on the data or metadata for national security or similar security concerns.
MDspatialRepType	ISO 19115 codelist of the methods (such as raster or vector) used to represent the geoinformation present in the dataset.
Enumeration	**Description**
ThematicModelCode	Enumeration of different thematic models present in CityGML such as Building, Vegetation, etc.
TerrainTypeCode	Enumeration of different terrain types present in CityGML such as TINRlief, RasterRelief, etc.
LODcode	Enumeration of the CityGML LODs (0-4). We also included the LODs proposed by Biljecki et al. [24]
stateCode	Enumeration with values to identify if a feature is present or absent.

8.4.1 Extendability

The Metadata ADE is modularly designed for future possible extensions to store metadata related to other domains and applications. It can be extended by other ADEs to incorporate domain-specific data needs. Figure 8.4 presents such an example for the existing CityGML Noise ADE. A new class `MDnoiseBuilding` is created by extending the Metadata ADE `MDbuilding` class. `MDnoiseBuilding` has a new attribute to store information such as the number of buildings enriched with Noise ADE attributes (`numberOfNoiseBuildings`). Similarly, `MDnoiseRoads` is a subclass of `MDtransportation` with a new attribute to store the number of noise road segments present in the data (`numberOfNoiseSegments`).

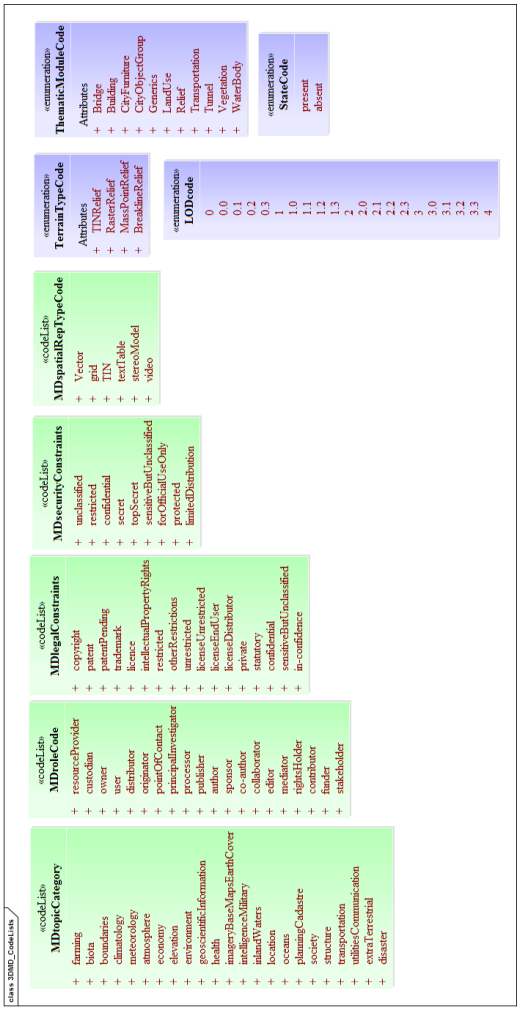


Figure 8.3: Implemented ISO codelists (shown coloured in green) and proposed new codelists (shown coloured in purple) in the CityGML 3D Metadata ADE.

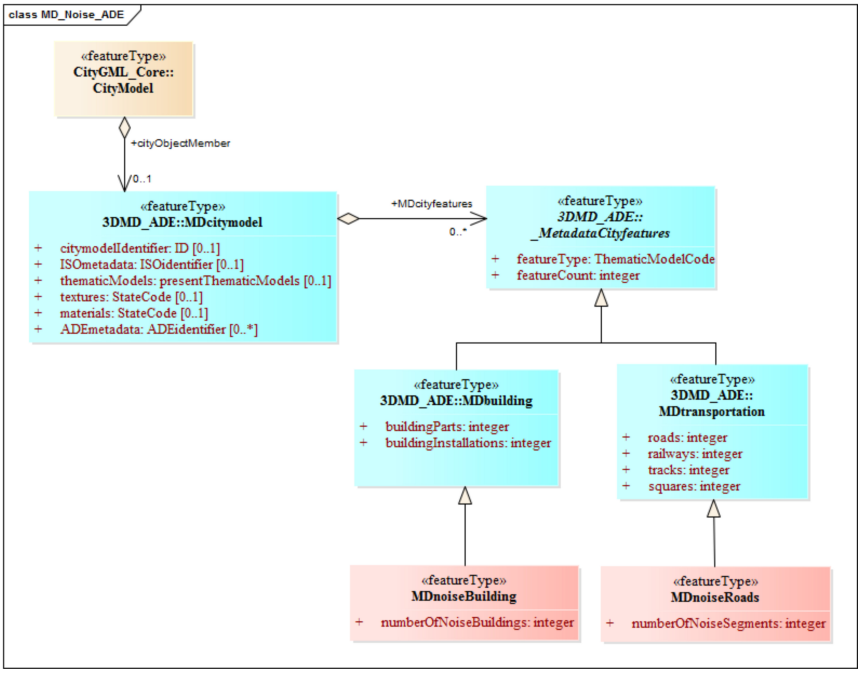


Figure 8.4: Excerpt of the UML diagram of the 3D Metadata ADE, depicting how to extend it to incorporate the metadata related to other ADEs (such as Noise ADE)

8.5 Automatic metadata generation

Metadata generation, to populate the metadata extension with data, does not need to be a painful task and while the seemingly time-consuming and uninteresting nature of the topic is often seen as a deterrent [74], it is more often hindered by a lack of definition of metadata for 3D city models in particular. Many of the values for the categories discussed in this paper can be easily accessed during the city model generation process and therefore having a solidly defined metadata ADE is advantageous to guide data creators.

To further ensure the usability of this work, we offer a Python software[6] that derives categories such as the levels of detail present, thematic models, extent, etc. It parses a dataset looking to see if other metadata information is present

[6] https://github.com/tudelft3d/3D_Metadata_ADE/tree/master/Code

and has default values to indicate which values are missing. Due to the large file size of most CityGML files, we chose to generate the metadata as a separate file which ensures faster parsing but users can write to the original file if they wish to. The automation script already ensures that the output conforms to the definition of the Metadata ADE.

8.6 Conclusions

In this chapter, we addressed the lack of metadata in 3D city models in general and CityGML in particular for the discovery of 3D datasets which are often too large to be parsed easily. We proposed a CityGML ADE that is both ISO 19115 compliant and incorporates further elements as required by users of 3D city models. Given the open nature of CityGML and its collaborative evolution process, this ADE could serve as the model for the next version of CityGML, version 3.0. We modelled the ADE to reuse CityGML elements as much as possible to realise an easy transition from ADE to a core module of the CityGML standard. We developed a script that automatically generates metadata for the CityGML datasets. It also addresses a major barrier to dataset discovery which is a lack of user-friendly interfaces [203]. Further, the metadata ADE can be extended by other ADEs such as Noise ADE, Energy ADE, etc. to incorporate application-specific metadata.

Furthermore, as is argued by many metadata practitioners, including Ellul et al. [74] and Olfat et al. [203], creating metadata after dataset generation requires a considerable amount of effort and the availability of information may be reduced which leads to incomplete metadata. They argue that metadata generation should be a process that is run parallel to data generation. Having a well-defined Metadata ADE can aid data-creators by providing neat guidelines to follow. It can be considered as a first step towards a further discussion on the metadata needs of future 3D SDI, particularly of 3D city models.

Our schema for the metadata in CityGML is also implemented in CityJSON[7]. CityJSON is a JSON-based encoding of a subset of the OGC CityGML data model (version 2.0.0) for storing 3D city models [150]. CityJSON offers an alternative to the GML encoding of CityGML. Our metadata categories and elements are available in the core of the recent version of CityJSON, version 1.0.1.

[7] http://www.cityjson.org

Chapter 9

A harmonized data model for noise simulation in the EU

This chapter is based on the paper:

Kumar K, Ledoux H, Schmidt R, Verheij T, and Stoter J, 2019. A Harmonized Data Model for Noise Simulation in the EU. *ISPRS International Journal of Geo-Information*, 9(2), p.121.
doi: https://doi.org/10.3390/ijgi9020121

9.1 Introduction

In previous chapters, we discussed how the 3D city models can be enriched with application specific information for use in different applications. The use of such 3D city models can greatly improve environmental analysis in 3D. However, there is a dearth of such enriched 3D city models for use in applications. In this chapter, I focus on the application of 3D city models for urban noise simulations. Monitoring and mapping of urban noise is an active area of research which is drawing substantial public attention.

The European Union (EU) has formulated the *2002/49/EC Environmental Noise Directive (END)* as a common management plan to deal with urban noise [61]. The Directive requires the EU Member States to determine the exposure of an individual to environmental noise through strategic noise mapping and to make action plans to reduce noise, where necessary [61]. Different EU Member states have developed their own noise assessment methods and guidelines for estimating noise at local, regional, and national scales, e.g. RMW (Reken en meetvoorschrift Verkeerslawaai) in the Netherlands, NMPB (Nouvelle Méthode du Prévision de Bruit) in France, CRTN (Calculation of Road Traffic Noise) in UK, etc. Noise simulation studies for estimating the noise levels utilises input data (spatial and non-spatial data) about the noise sources, noise measures (barriers and screens), and the built environment in computer implementations of these noise assessment methods [51]. As explained in Kephalopoulos et al. [122], King et al. [125], Murphy and King [173], these assessment methods along with the input data extracted from the national registers and databases, and other open and commercial data, differ in several aspects, such as: (1) the method used to assign receivers on the façades of the buildings; (2) completeness, accuracy, and reliability of spatial data used; (3) approach followed to calculate the number of inhabitants in a building; (4) use of default data instead of actual real world data such as maximum speed value for the speed of the vehicles, and so on. The heterogeneity in these methods and utilised input data makes it difficult to obtain comparable results across the EU [125, 173, 180].

To address this problem, a Common framework for NOise aSSessment methOdS (CNOSSOS-EU) was developed by the European Commission, in co-operation with the EU Member States to enable a consistent and accurate reporting of strategic noise mapping by the Member States, and thus to fulfil their obligations under the END [121]. In 2015, an update to the END Annex II was published, it requires all the EU Member States to use CNOSSOS-EU from 31 December 2018 onwards [62]. The core methodological framework of CNOSSOS for the EU noise mapping was developed during the first phase of the CNOSSOS-EU process (2009-2012) and is described in Kephalopoulos et al. [121]. The second phase of the CNOSSOS-EU process is the implementation phase. It was expected to cover the development of guidelines for the practical implementation of CNOSSOS

such as schema and database design for inputs and outputs of CNOSSOS-EU, use of common standards, etc. A proof of concept version of the initial ideas and features of the guidelines was established in the form of a website but it was not made public and no follow ups have been done so far [121]. Further, Kephalopoulos et al. [121] lists only the outline of the proposed guidelines for CNOSSOS-EU.

In this chapter, I focus on one of the crucial challenges of the implementation phase of the CNOSSOS-EU process, i.e. structuring of input and output data of noise simulation in a standardized format in order to be able to compare the outcomes of different noise studies across EU and to be able to better structure and exchange data between different stakeholders. An important work in this direction is the development of the CityGML Noise ADE (Application Domain Extension) by the Special Interest Group SIG 3D [185]. CityGML is an international 3D GIS standard established by the international standardization organization for geoinformation: Open Geospatial Consortium [185]. The CityGML Noise ADE extends the existing CityGML schema by adding new classes and objects relevant for noise simulation. Many cities in the state of North-Rhine Westphalia, Germany have modelled their noise data based on the CityGML Noise ADE [45]. However, it is set in a German context based on the German regulations for modelling noise caused by roads and railways only. Furthermore, it does not model industrial and aircraft noise (refer to the Section 9.2.3 for the limitations of the current Noise ADE).

I describe and implement in this chapter a harmonised CityGML-based input/output data model for the CNOSSOS-based urban noise simulation for the EU. I review the existing noise assessment methods and provide an extensive inventory of the noise assessment methods and guidelines used in the third round of the strategic noise mapping in the EU (Table 9.1). Further, I discuss the current status of the guidelines for data harmonisation for different noise simulation studies using CNOSSOS-EU. Finally, I present my framework for harmonising input/output data model in CityGML for urban noise simulation. In this approach, the existing CityGML Noise ADE is extended and restructured with new classes, attributes and other concepts in accordance with the CNOSSOS-EU. Further, a real world dataset for an area in the Netherlands is generated to model the data required for simulating urban noise using CNOSSOS-EU. This model could serve as a reference for future developments, applications, and validation of the CNOSSOS-EU methodological framework.

9.2 Need for a harmonised noise model: Current status and challenges

9.2.1 CNOSSOS-EU: Where does it stand?

CNOSSOS-EU represents a harmonized method to assess noise levels from the main sources of noise (road traffic, railway traffic, aircraft and industries) across the EU and should replace national models for the next round of strategic noise mapping (2021/2022) [245]. Finland has already used the CNOSSOS-EU model in the 2017 END strategic noise mapping [127, 128]. Some of the other MS have also started the implementation of the CNOSSOS-EU framework and research has been going on to determine its adaptability. Vergoed and van Leeuwen [245] examined the applicability of CNOSSOS for legal purposes such as limiting the maximum emission of a road, a railway and an industry, and controlling the maximum noise levels on façades of buildings in the Netherlands. Wszołek et al. [252] compared ISO 9613-2 and CNOSSOS for industrial noise assessment of a large industrial plant in Poland. Bertellino et al. [13] determined the compatibility of the old approach (XPS 31-133) with the upcoming one (CNOSSOS-EU Road) for calculating road noise levels for the city of Trento, Italy. Furthermore, Switzerland has updated its road traffic noise emission model (sonROAD18) based on CNOSSOS-EU [89]. CNOSSOS has also been implemented in noise simulation software such as Geomilieu [58], MithraSIG [81], IMMI [250], Predictor-LimA [31], CadnaA [49], etc. In addition, new noise mapping systems such as DYNAMAP [12] can be easily interfaced with CNOSSOS.

At present, CNOSSOS is hardly being used for many reasons e.g. it is still under development, it needs software implementations and data guidelines, Q&A needs to be done, it is new and still in the process of being compared and analysed, and so on. The CNOSSOS-EU framework would help in increasing the consistency among the action plans adopted by the EU MS on the basis of the results of the noise mapping and also would allow a better evaluation of the effectiveness of the action plans and the development of a basis for community measures by the commission to reduce noise emitted by the major noise sources. This would also allow the EU member states to concentrate more on the reliable implementation of common tools and guidelines for input data, and further development and maintenance, thus optimising their efforts instead of coping with different assessment noise methods used for different purposes and with different capabilities and range of applicability, which is a highly demanding task in terms of both resources and time.

9.2.2 Present guidelines for data harmonisation for noise simulations using the CNOSSOS-EU

Apart from the software implementations and validation experiments for CNOSSOS (mentioned in Section 9.2.1), very little has been done for the practical guidelines outlining the specifications for the data (spatial/non-spatial), schema design, standards, etc. for CNOSSOS. The CNOSSOS documentation [121] lists only the outline of the proposed guidelines for it. Designing a common underlying schema for the data to be used in CNOSSOS-based noise simulation can aid in obtaining comparable outcomes across the EU. Some of the data-related aspects mentioned (and not implemented) in the documentation include [121]:

- Data schema design:
  - inputs and outputs for CNOSSOS-EU,
  - data specification tables and schema diagram,
  - an INSPIRE-compliant, open and extensible standard,
  - rules and guidance on how additional objects and attributes may be added to the schema,
  - a common data format which allows interfacing with data providers, other data owners and cross-border project liaison.
- GIS and data set specifications
  - GIS and END requirements,
  - terminologies and technical specifications of the GIS data and software,
  - GIS layers, scale, and accuracy: data model, data dictionary, data validation, reference system, metadata, GML specification.

It was proposed that these aforementioned CNOSSOS-EU guidelines would be developed as an interactive web-based tool, which links to the specific aspects of the technical description [121]. There have been no follow-ups so far. Independent studies have been done by various researchers to model the data for use with CNOSSOS. For instance, Shilton et al. [227] discussed how existing datasets for roads can be used while migrating from interim methods (NMPB-Routes-96) to CNOSSOS-EU. Maijala and Kontkanen [161] investigated the sensitivity of the CNOSSOS-EU framework with respect to the changes in meteorological conditions in Nordic countries. However, no semantic model exists for modelling the data for CNOSSOS based noise simulations.

9.2.3 Existing CityGML Noise ADE

An END compliant CityGML noise ADE is documented in OGC [185] and is in accordance with the German regulations for noise assessment. The CityGML

Noise ADE, extends the existing CityGML schema by adding new classes and attributes relevant to noise mapping. Many cities in the state of North-Rhine Westphalia, Germany have implemented their 3D noise models based on the CityGML Noise ADE [43, 44, 45]. Three existing CityGML modules *(Transportation, Building, CityFurniture)* were extended to include noise related data in CityGML.

Transportation module contains new classes e.g. *NoiseRoadSegment* and *NoiseRailwaySegment* which are the segments of roads and railway lines. These objects have their own set of attributes e.g. traffic flow, speed limits, surface type, etc. The geometry of these objects is derived from their respective parent classes: *Road* and *Railway*. It also contains a new feature type *Train* with attributes like train type, its speed, etc. to store the information about individual trains. Similarly, the *Building* module was extended to include noise attributes related to existing buildings e.g. reflection from buildings, noise levels observed during the day and night, number of people living in the buildings, etc. Similarly, the *CityFurniture* module contains new class *NoiseCityFurnitureSegment* with new attributes for noise. All these noise attributes are derived based on the regulations issued by the Federal Government of Germany in accordance with END.

The current Noise ADE has some limitations, documented in Kumar et al. [135]. For instance, it only represents noise data arising from road traffic and railways and does not support industrial noise. Trams are also not included in the model. Further, no distinction is made between the speeds of different type of vehicles e.g. motor cycles, light vehicles and heavy vehicles. In addition, the noise ADE is set in the German context i.e. the specific attributes and object types result from the German regulations for noise emmision calculations (see Annex H.1 [185]).

9.3 Methodology

The methodology in this research is the literature review and the development of a data model for data harmonisation for noise simulations. I carried out a systematic screening of the scientific literature, reports, and official data of the EU Member States submitted to the Central Data Repository (CDR)[1] of the European Environment Agency (EEA) for strategic noise mapping. As a part of this work, I have also been in contact with the stakeholders in this project, such as the dutch municipalities (Hague and Rotterdam) and the companies (DGMR B.V., the Netherlands) to determine their requirements for input data to perform noise simulations. I put together an extensive inventory of the meth-

[1] http://cdr.eionet.europa.eu/

ods and guidelines for noise assessment used in the Round 3 of strategic noise mapping in the EU (Section 9.4). I briefly discuss how the disparity among these assessment methods and the input data can influence the results of noise simulations of different European countries. Furthermore, the need of having a common European noise assessment method and a common input database for noise simulations is also highlighted.

Based on the findings from the review and discussions with the noise modelling experts, I focus here on the development of an harmonised input/output data model for noise simulations (Section 9.5). The data model is based on CNOSSOS-EU because the 2015 update to the Annex II requires all the EU Member States MS to use CNOSSOS-EU from 31 December 2018 onwards for strategic noise mapping. I explored the CNOSSOS-EU guidelines and the following different data sources and models to understand the data requirements for noise simulations:

- *Geluidsregister*[2]: the national noise register of the Netherlands,
- *Dutch Information Model (IM) Geluid*[3]: the upcoming information and data exchange model for the noise data in the Netherlands,
- *CityGML Noise ADE*: A CityGML extension to model data accroding to the German regulations for noise assessment (explained in Section 9.2.3),
- *INSPIRE Directive*[4]: the spatial data infrastructure for the EU environmental policies and activities which may have an impact on the environment.

CityGML is selected as the best available solution to implement our data model for modelling the data of noise simulations. First, CityGML is the GML-based, open, semantic 3D data model with the possibility to extend its data model with application specific information aka ADEs. Second, it has already been used in Germany for modelling noise related data i.e. the CityGML Noise ADE. The existing CityGML Noise ADE is studied to understand which elements are already present and which elements are missing in this data model and restructure it by introducing new elements (objects and attributes) so that it aligns with CNOSSOS-EU, IM Geluid Model, INSPIRE, etc.

9.4 An inventory of the noise assessment methods and guidelines in the EU

Previously, the Article 6 and Annex II of the END recommended NMPB-Routes-96 [65], RMR (Reken- en Meetvoorschrift Railverkeerslawaai) [54], ECAC.CEAC

[2] https://www.rijkswaterstaat.nl/kaarten/geluidregister.aspx
[3] https://github.com/Geonovum/IMG
[4] https://inspire.ec.europa.eu/

Doc 29 (2nd Edition, 1997) [68], and ISO 9613-2 [99] as the interim computation methods for calculating road traffic noise, railway noise, aircraft noise and industrial noise.

The END requires EU member states to produce strategic noise maps every five years. The first round of the strategic noise mapping was done in 2007, second round followed in 2012, and the third round in 2017. There are comparability issues between the outcomes of the these three rounds owing to different reporting requirements and structure, differences in the input data used, a lack of common assessment method, missing/incomplete data of noise exposure, and data quality differences.

We report on the latest round (Round 3) of the strategic noise mapping. A variety of noise assessment methods were used in the third round. We provide, in Table 9.1, an extensive inventory of the noise assessment methods and guidelines followed in the EU Member States for the Round #3 of strategic noise mapping in 2017. In total, 30 different noise assessment methods and guidelines were counted. The French method NMPB was the most widely used road traffic noise assessment method with 17 out of 28 MS using it. Ireland used the UK CRTN method while CNOSSOS-EU was only used by Finland. Dutch RMR was applied by 16 out of 28 MS for calculating noise from railways. The computation methods for the aircraft and industrial noise, ECAC.CEAC Doc 29 and ISO 9613-2 were used by 18 and 14 MS, respectively.

While preparing this inventory, some problems were also encountered. For instance, not all the EU Member States submitted their data for evaluation to the CDR such as Romania, and Cyprus. Some Member States had incomplete data such as Ireland, Greece, and UK. The Electronic Noise Data Reporting Mechanism [77] requires the EU Member States to submit metadata to the CDR along with the main data files. This metadata contains information about the authority responsible for the data, spatial reference system and extent of the data, noise assessment methods and guidelines used, and so on. However, not all the Member States submitted this metadata.

The heterogeneity in the assessment methods and utilised input data makes it difficult to obtain comparable results across the EU. These assessment methods differ in several aspects, such as the type (e.g. a single point or a line source of emission), position, and the height of the source, approach used to calculate the noise from a source, spectral bands used, and so on. Nijland and Van Wee [180] provide a detailed comparison of the existing noise assessment methods used in different EU Member States. For instance, roads are treated as a collection of incoherent point sources in NMPB and Nord 2000, but CRTN treats roads as a line source [173]. In general, the noise propagation calculations also differ in these methods. For instance, these methods differ in their noise propagation

parameters such as noise absorption by air and ground, inclination of ground, multiple reflections from nearby façades, noise barriers and other obstacles.For instance, CRTN does not take atmospheric attenuation into account, which is not the case in NMPB, and Nord 2000. Another reason for the differences can be attributed to the databases and data registers used in association with the national methods e.g. differences in the classification and speed of vehicles for the road traffic noise, classification of trains for the railway noise, etc. [152]. Stoter et al. [234] described an approach to automatically reconstruct input data for noise simulations in order to improve the efficiency and reliability, and consistency of different noise studies. For the quantitative analysis, the authors compared the differences between the simulation outcomes using the data created by their approach and the existing semi-automatic approach, for a study area in the Netherlands on about 1000 observation points. The differences in the noise levels ranged from -2.17 dB to 1.83 dB. The disparity in the use of different methods, input databases, and software can have a considerable impact on the overall calculations. However, some differences are just due to the local materials (like in case of the trains) and cannot be solved by having uniformity in method and input data. A common EU noise assessment method along with a harmonised EU database of input data is essential to achieve results of sufficient quality.

9.5 Our data model for noise simulations: The eNoise ADE

Our eNoise ADE (extended Noise ADE) is realised as a UML model, and XSD schema. All resources (UML, XSD, and documentation) related to the eNoise ADE are available on our public GitHub repository: `https://github.com/tudelft3d/eNoiseModel` The data model covers the noise arising from the roads, the railways, and the industries. It does not model the aircraft noise in this research. The classes in the model are divided into 2 categories:

- Existing classes which are updated (Section 9.5.1)
- New classes which are introduced (Section 9.5.2)

To avoid any ambiguity, the attributes and any other classes in the ADE are separated based on their source, namely, original CityGML, existing Noise ADE, geluidregister, Dutch IM Geluid, INSPIRE, and CNOSSOS-EU. Further, identifiers *'noise'* and *'enoise'* are used to distinguish between the attributes of the existing Noise ADE and our eNoise ADE. CityGML and GML identifiers remains the same as in the actual data model.

9.5.1 Existing classes which are updated

CityModel The CityGML *CityModel* class originally stores only the name (*gml:name*) and ID (*gml:id*) of the 3D city model present in the dataset. The *CityModel* class in our ADE is extended to store the metadata attributes related to the eNoise ADE such as:

- name of the ADE (*enoise:adeName*)
- version of the ADE (*enoise:adeVersion*)
- namespace of the ADE (*enoise:adeNamespace*)
- authority responsible for the ADE (*enoise:adeAuthority*)
- link to the XML schema of the ADE (*enoise:adeXMLschema*)
- link to the UML model of the ADE (*enoise:adeUML*)
- authority responsible for the dataset (*enoise:dataAuthority*)
- assessment methods or guidelines used for the noise from the roads (*enoise:roadNoise*), the railways (*enoise:railwayNoise*), and the industries (*enoise:industrialNoise*)
- name (*enoise:noiseSoftwareName*), version (*enoise:noiseSoftwareVersion*), and organisation/company (*enoise:noiseSoftwareAuthority*) of the software used for noise simulation.
- link to any additional documentation (*enoise:documentation*)

These attributes are ISO 19115 compliant, and based on the CityGML Metadata ADE developed by Labetski et al. [143].

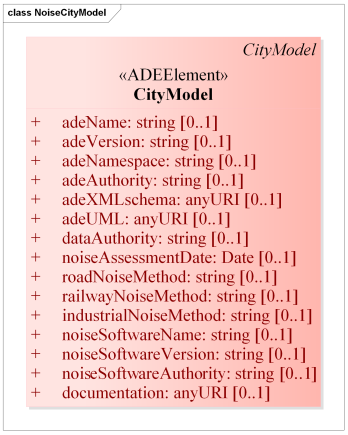


Figure 9.1: The UML depicts the CityGML *CityModel* class extended to store the metadata attributes related to the eNoise ADE

Building The CityGML *Building* class inherits the attributes and relationships from the abstract class *_AbstractBuilding*. The *_AbstractBuilding* already has some attributes to store the semantics of the building model in the dataset such as the class of the building, its function (e.g. residential, public, or industry), its usage, year of construction, year of demolition, type of its roof, its measured height, and the number of the storeys above and below ground and their heights (as shown in yellow in Figure 9.3).

The existing Noise ADE enriched the *_AbstractBuilding* class with noise related attributes (as shown in orange in Figure 9.3) such as:

- reflection from the building (*noise:buildingReflection*) and reflection correction in dB (*noise:buildingReflectionCorrection*)
- noise levels observed during the day, in the evening, and at night (*noise:LDenMax, noise:LDenMin, noise:LNightMax, noise:LNightMin, noise:LDenEq, noise:LNightEq*)
- number of inhabitants in the building (*noise:buildingHabitants*)
- number of apartments in the building (*noise:buildingAppartments*)
- a list of emission points (*noise:buildingImmissionPoints*)

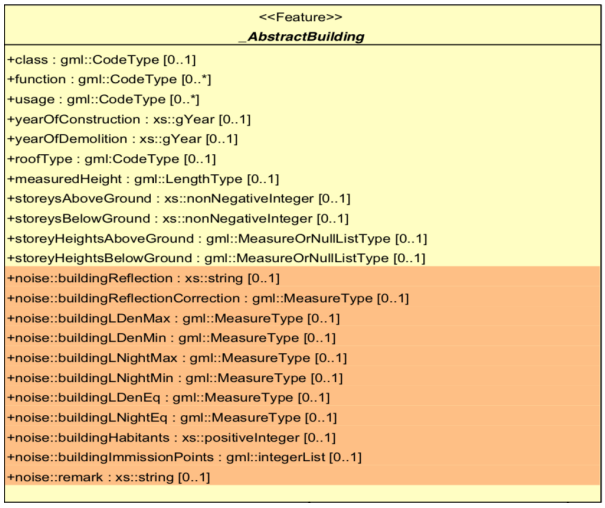


Figure 9.2: Original attributes in the the UML model of the CityGML *_AbstractBuilding* (shown in yellow) and extended attributes in the Noise ADE (shown in orange)

However, the term *Building* is not referenced directly in the context of the exposure assessments required by the Annex VI of the END. Instead the term

dwelling is used which can be described as a *self-contained unit of accommodation* [94]. A building may contain zero, one or more individual dwellings. For instance, a residential building can contain one or more individual dwellings. To comply with the END, CNOSSOS-EU and INSPIRE, a new city object *Dwelling* is introduced in the eNoise ADE with the following attributes:

- ID of the dwelling (*gml:id*)
- address of the dwelling (*core:address*)
- number of floors in the dwelling (*enoise:numberOfFloors*)
- number of inhabitants in the dwelling (*enoise:numberOfInhabitants*)
- floor area of the dwelling (*enoise:floorArea*)
- noise levels for each individual dwelling (*enoise:LDenMax, enoise:LDenMin, enoise:LNightMax, enoise:LNightMin, enoise:LDenEq, enoise:LNightEq-, enoise:L24Max, enoise:L24Min*)
- octave band for each time period (day, evening and night) (*enoise:octaveBandDay, enoise:octaveBandEvening, enoise:octaveBandNight*)
- number of receivers placed on the building (*enoise:numberOfReceivers*)
- information about the receivers placed on the dwelling (*enoise:receiver*)

The eNoise ADE includes all the aforementioned attributes for buildings from the Noise ADE. Apart from that, the following attributes are added to the *_AbstractBuilding* class in the eNoise ADE in accordance with the *INSPIRE* and the *CNOSSOS-EU*:

- octave band for each time period (day, evening and night) (*enoise:octaveBandDay, enoise:octaveBandEvening, enoise:octaveBandNight*)
- calculated noise levels for the whole building for 24 hours (*enoise:buildingL24Max, enoise:buildingL24Min*)
- most exposed façade (*enoise:mostExposedFacade*)
- quiet façade (*enoise:quietFacade*)
- base area of the building (*enoise:baseArea*)
- volume of the building (*enoise:volume*)
- absolute ground height of the building (*enoise:groundHeight*)
- number of receivers placed on the building (*enoise:numberOfReceivers*)
- information about the receivers placed on the building i.e. ID of the façade on which the receiver is placed (*enoise:receiverFacade*), location of the receiver (*enoise:receiverLocation*), and height above the ground (*enoise:-receiverHeight*)
- number of dwellings in the building (*enoise:numberOfDwellings*)
- information about the dwellings (*dwelling*)

Relief (Terrain) GIS practitioners often model terrain (relief) as TINs (Triangulated Irregular Networks) or grids. Additionally, CityGML allows to model a

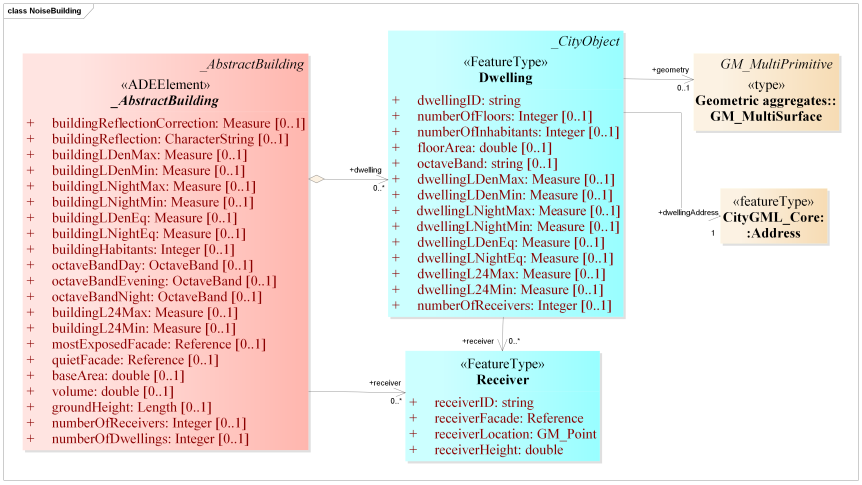


Figure 9.3: The UML model of the *_AbstractBuilding* (shown in Red) extended to include noise related attributes and receiver information. A new feature type *Dwelling* (shown in blue) is introduced to describe the self-contained unit of accommodation within the buildings

terrain as a regular grid, or a TIN, or as break lines, or as a collection of points [185]. Apart from the aforementioned representations, it is also possible to model a terrain with 3D lines (which can also be contour lines). One advantage can be to store only those 3D lines that significantly contribute to the terrain, thereby reducing the volume of input data required for the simulation and hence, increasing the computation performance of the simulation. Therefore, *enoise:LineRelief* is added as one more terrain representation type in CityGML representing such 3D lines. This is also in accordance with the *Dutch IM Geluid* model. The idea is to capture height variations with as few elements (elevation points or lines) as possible.

Land use The CityGML *LandUse* class is not only used to describe the specific land use of an area, but also the physical characteristics of the area, e.g. wetlands, grasslands, forests, sand, etc.[185]. The noise reflection or absorption are characteristics properties of a land area. Therefore, the CityGML *LandUse* class is extended to include an attribute (*enoise:absorptionFraction*) to model the degree of noise absorption by the land area. This is also in accordance with the *Dutch IM Geluid* and *CNOSSOS-EU*. The value of the proposed attribute are defined in an enumeration (*enoise:AbsorptionFractionValues*) with values *0 (hard), 1 (soft), 0.5 (middle/medium)*.

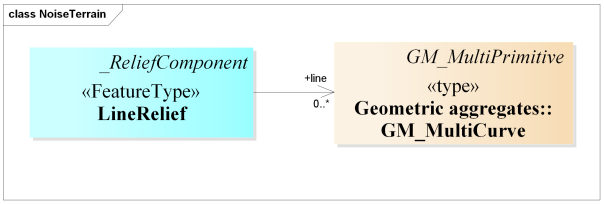


Figure 9.4: The UML model for the *LineRelief* representation added to the CityGML *Relief* module to represent the terrain as height lines in CityGML

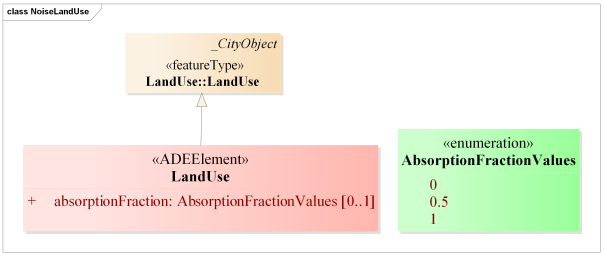


Figure 9.5: The UML model depicts the CityGML *LandUse* class extended to model the noise absorption property of the land area

Road The CityGML *Road* class originally has only a few attributes to store the ID *(gml:id)*, class *(tran:class)*, function *(tran:function)*, and usage *(tran:usage)* of the road. The *Road* class inherits these aforementioned attributes from the CityGML class *TransportationComplex*. The existing Noise ADE added a new object type *noise:NoiseRoadSegment* to represent the individual segments of a road with special attributes for noise calculation such as:

- average hourly traffic flow for the day, evening and at night on the road segment (*noise:mDay, noise:mEvening, noise:mNight*)
- average hourly traffic flow for 16 hours of the day on the road segment i.e. day and evening summarised (*noise:mDay16*)
- heavy vehicle percentage in% for the day, evening and at night on the road segment (*noise:pDay, noise:pEvening, noise:pNight*)
- heavy vehicle percentage in% for 16 hours of the day on the road segment i.e. day and evening summarised (*noise:pDay16*)
- average daily traffic flow on the road segment (*noise:dtv*)
- speed limit in km/h for the passenger cars for the day, evening and at night on the road segment (*noise:speedDayPkw, speedEveningPkw, speed-*

NightPkw)
- speed limit in km/h for the heavy vehicles for the day, evening and at night on the road segment (*noise:speedDayLkw, speedEveningLkw, speedNightLkw*)
- material of the surface of the road segment (*noise:roadSurfaceMaterial*)
- correction of noise emission of the according road segment surface material in dB (*noise:roadSurfaceCorrection*)
- width of a cross-section of the road segment in metres (*noise:distanceCarriageway*)
- width of the road segment in metres (*noise:distanceD*)
- attribute to indicate if the road segment is a bridge (*noise:bridge*) or a tunnel (*noise:tunnel*)
- slope correction for the road segment (*noise:roadGradientPercent*)
- lineage of the data (*noise:lineage*)
- geometry of the road segment (*noise:lodOBaseLine*)

To comply with the CNOSSOS-EU, the road segments are updated with the following attributes:

- height of the source (*enoise:sourceHeight*)
- removed the *noise:distanceCarriageway* attribute and renamed the *noise:distanceD* as *enoise:roadWidth* to reflect the width of the road segment in metres
- age of the road surface (*enoise:ageOfRoad*)
- attribute to identify if the road segment is a crossing (*enoise:crossing*) or a roundabout (*enoise:roundabout*)
- speed of the vehicles during the day, in the evening and at night. We divided the vehicles into five categories as proposed by the CNOSSOS-EU: *light motor vehicles, medium heavy motor vehicles, heavy vehicles, powered two wheelers (motorcycles and moped), and electric motor vehicles* [121] and report on their speeds during the day (*enoise:speedDay*), in the evening (*enoise:speedEvening*), and at night (*enoise:speedNight*). CNOSSOS includes an open class for the new vehicles to be developed in the future. This class could cover electric or hybrid vehicles or any other futuristic vehicle [121]. We included only the prevalent electric motor vehicles category in our model. We did not include the speed related attributes of the current Noise ADE.
- average yearly traffic flow per vehicle category per time period i.e. for the day, evening and at night for the road segment (*enoise:mDay, enoise:mEvening, enoise:mNight*). The traffic flow attributes of the current Noise ADE did not take vehicle category into account.
- octave band for each time period (day, evening and night) in which the emissions are recorded (*enoise:octaveBandDay, enoise:octaveBandEvening, enoise:octaveBandNight*)

- noise emission during the day (*enoise:emissionDay*), in the evening (*enoise-:emissionEvening*), and at night (*enoise:emissionNight*)

Railway Railway represents routes that are utilised by rail vehicles such as trams or trains. The CityGML *Railway* class inherits the same attributes as the CityGML *Road* from the CityGML *TransportationComplex*. The existing Noise ADE added a new object type *noise:NoiseRailwaySegment* to represent the individual segments of a railway track with special attributes for noise calculation such as:

- type of the surface material of the segment of the rail track (*noise:railway-SurfaceMaterial*) and its correction in dB (*noise:railwaySurfaceCorrection*)
- attribute to indicate if the rail segment is a bridge (*noise:bridge*) or a crossing (*noise:tunnel*)
- curve radius of the rail segment in metres (*noise:curveRadius*)
- additional correction of noise emission if required (*noise:additionalCorrec-tionSegment*)
- geometry of the rail segment (*noise:lodOBaseLine*)

Generally, the most relevant elements influencing the railway noise emission are railhead roughness, rail pad stiffness, track base, rail joints, and the radius of curvature of the rail track [121]. The railhead roughness and the track decay rate are the two acoustically important parameters according to the ISO 3095 [106]. These are not covered in the existing CityGML Noise ADE. To comply with the CNOSSOS-EU and ISO 3095, the railway segments are updated with the following attributes:

- indicator for the railhead roughness (*enoise:railRoughness*) such as well maintained and very smooth, normally maintained, and so on, and its value (usually in microns) (*enoise:railRoughnessValue*). The values of the attribute *enoise:railRoughness* are defined in an enumeration (*enoise:rail-RoughnessValues*).
- indicator for the rail pad stiffness (*enoise:railpadStiffness*) such as soft, hard, medium and its value (*enoise:railpadStiffnessValue*). The values of the attribute *enoise:railpadStiffness* are defined in an enumeration (*enoise-:railpadStiffnessValues*).
- type of the base of the rail segment (track) (*enoise:trackBase*) e.g. ballast, slab track, etc. The values of this attribute are defined in an enumeration (*enoise:trackBaseValues*).
- type (*enoise:curveType*) of the curve of a railway segment i.e. straight, low curve, medium curve, and high curve. The values of this attribute are taken from an enumeration (*enoise:CurveTypeValues*).
- noise reduction measures in the rail track (*enoise:noiseMeasure*) such as

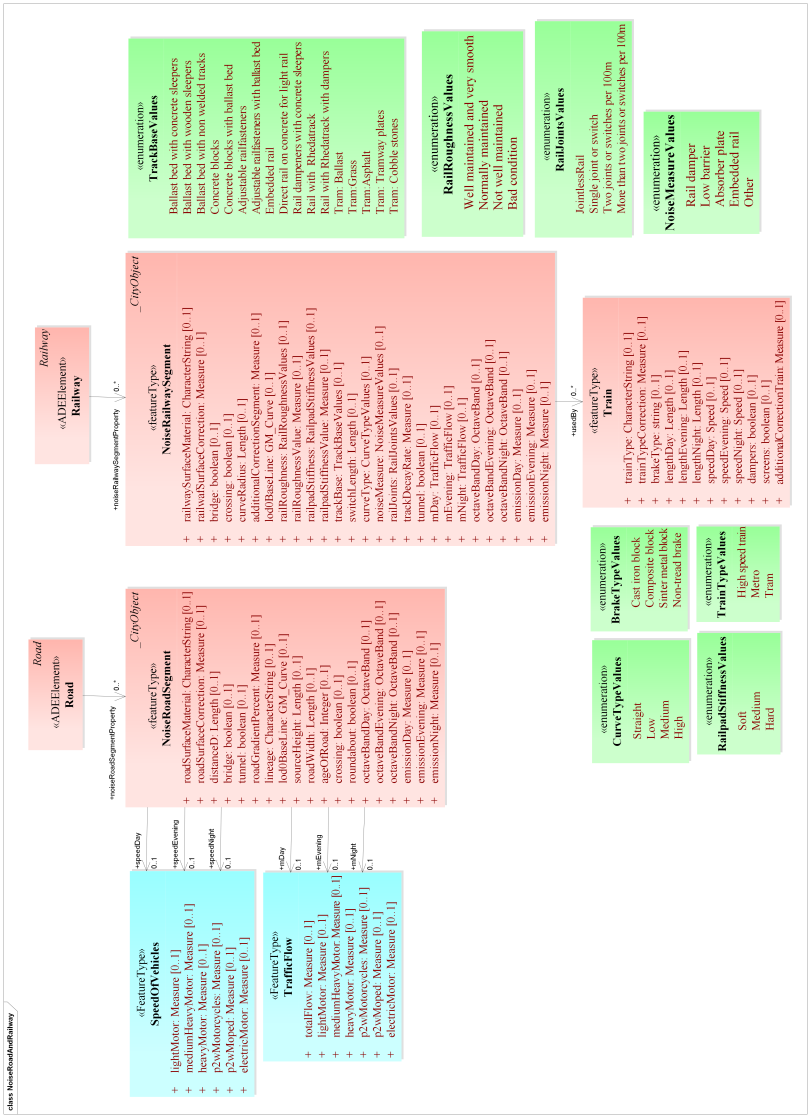


Figure 9.6: UML model of the CityGML *Road* and *Railway* (shown in Red) extended to include noise related attributes

dampers, barriers, and so on. The values of this attribute are defined in an enumeration (*enoise:noiseMeasureValues*).
- disconnections (switches or joints) on the rail segment (*enoise:railJoints*)
- length of the switch of a rail segment (*enoise:switchLength*)
- track decay rate in dB/m (*enoise:trackDecayRate*). A high decay rate usually indicate low noise and can be obtained for instance by the use of stiff pads between the rail track and the sleepers.
- attribute to identify if the railway segment is in a tunnel (*enoise:tunnel*)
- octave band for each time period (day, evening and night) in which the emissions are recorded (*enoise:octaveBandDay, enoise:octaveBandEvening, enoise:octaveBandNight*)
- average yearly traffic flow over the railway segment during the day, evening, and at night (*enoise:mDay, enoise:mEvening, enoise:mNight*).
- noise emission from the railway segment during the day (*enoise:emissionDay*), in the evening (*enoise:emissionEvening*), and at night (*enoise:emissionNight*)

In the CNOSSOS-EU source model for the rail, the directivity of a source is part of the source description. Perpendicular to the source there is no correction, but if the source is seen at an angle the emission is lower. This is expressed in the directivity correction. It is a source attribute, but it can only be evaluated during the calculation itself and it is not included in the ADE.

Train *Train* feature type is already present in the existing CityGML Noise ADE with the following attributes:

- type of train *(noise:trainType)*
- noise emission correction (in dB) as per the type of the train (*noise:trainTypeCorrection*)
- portion (in %) of wagons with wheel disc brake for the day, evening and at night (*noise:brakePortionDay, noise:brakePortionEvening, noise:brakePortionNight*)
- total length of each train for the day, evening and at night in metres (*noise:lengthDay, lengthEvening, lengthNight*)
- speed of the train during the day (*noise:speedDay*), in the evening (*noise:speedEvening*), and at night (*noise:speedNight*)
- additional correction of noise emission if required (*noise:additionalCorrectionTrain*)

The *Train* feature type is updated with the following attributes in the ADE:

- type of train *(enoise:trainType)*. We introduced an enumeration *TrainTypeValues* for the different types of trains (such as high speed trains, trams, etc.)

- type of brakes in the train (*enoise:brakeType*). We introduced an enumeration *enoise:BrakeTypeValues* for the different types of brakes in a train.
- attribute to indicate the presence of noise reduction measures such as (*enoise:dampers*) and screens (*enoise:screens*)

9.5.2 New classes in the model

Industry To model the schema for the industrial noise, a concrete class *IndustrialNoiseSource* is defined to represent the industrial noise sources. Since at the moment, CNOSSOS-EU is used for strategic noise models the directivity information of the noise sources is not used and thus not described in the ADE. The class *IndustrialNoiseSource* has the following attributes:

- type of the industrial noise source i.e. point/line/surface (*enoise:type*)
- geometry of the noise source (*enoise:geometry*)
- height of the source from the ground (*enoise:heightAboveGround*)
- operating hours (day, evening, night on a yearly averaged basis) (*enoise:operatingHoursDay*, *enoise:operatingHoursEvening*, *enoise:operatingHoursNight*)
- emitted noise level by the source (*enoise:emissionDay*, *enoise:emissionEvening*, *enoise:emissionNight*)
- octave band for each time period (day, evening and night) in which the emissions are recorded (*enoise:octaveBandDay, enoise:octaveBandEvening, enoise:octaveBandNight*)

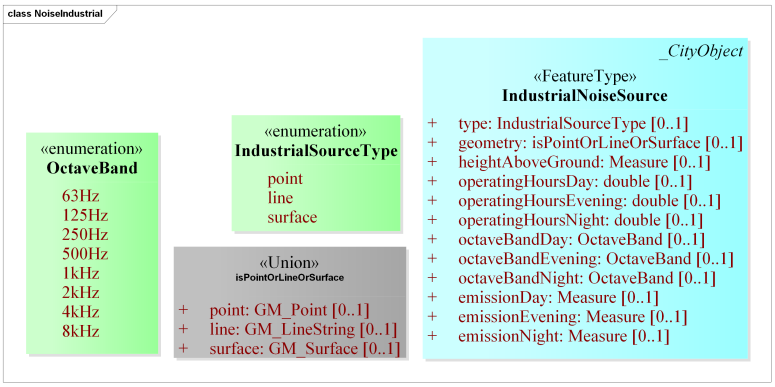


Figure 9.7: UML depicts the different types of industrial noise sources introduced in CityGML

Reference Points The reference points are legally prescribed points where the maximum levels of noise are determined by a legal decision and are monitored as such. These are modelled based on the *geluidregister* and *IM Geluid*. A class (*enoise:ReferencePoint*) is implemented as as a part of the CityGML *CityFurniture* module to represent such points. We store the geometry of these points ((*enoise:geometry*), the actual and the maximum noise value at these points (*enoise:noiseValue* and *enoise:maxNoiseValue*), and the height of these points above the ground (*enoise:heightAboveGround*).

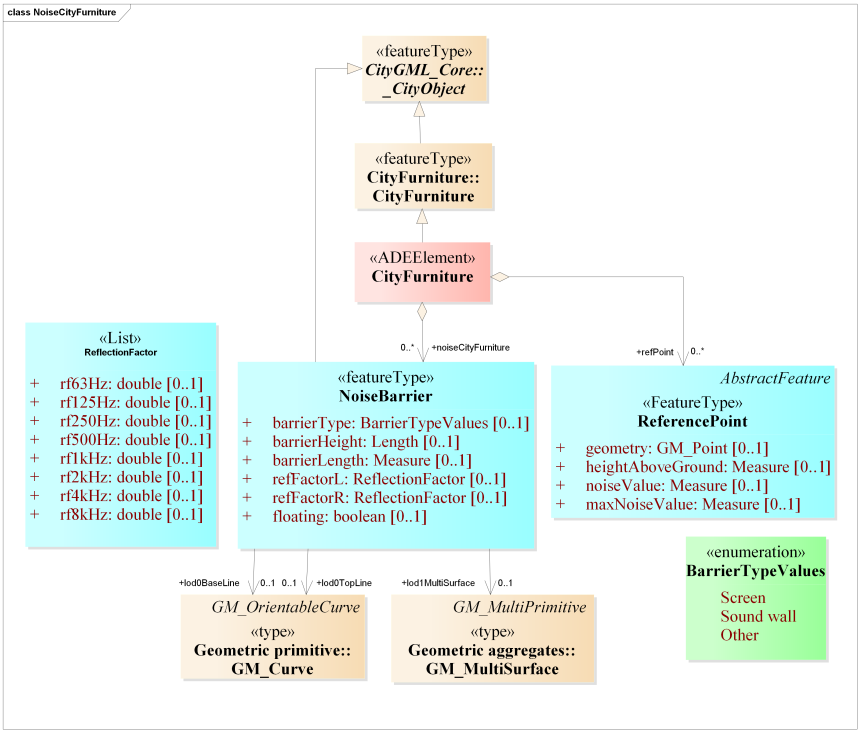


Figure 9.8: UML model for the noise barriers and reference points modelled in CityGML

Noise barriers The noise barrier feature is defined in a codelist in the *CityFurniture* module in the existing CityGML Noise ADE. Further, the geometry of noise barriers is defined as an LOD0 line representation in the existing Noise ADE, whereas there is no LOD0 representation of city furniture features in the

CityFurniture module. Further, it is not clear whether this line representation stores the top or bottom of the noise barriers. We define it as a concrete class in CityGML with the following attributes based on the *geluidregister*, *IM Geluid*, and the existing Noise ADE:

- type of the noise barrier (*enoise:barrierType*). We made an enumeration *BarrierTypeValues* to store the types of the noise barriers.
- height of the barrier in relation to the ground (*enoise:barrierHeight*)
- length of the barrier (*enoise:barrierLength*)
- reflection factor of the barrier on its left and right sides in the octave band (*enoise:refFactorL, enoise:refFactorR*)
- attribute *enoise:floating* to check the position of the barrier (value 0 = noise barrier on a slope and 1 = noise barrier on a viaduct)
- geometry of the barrier as curves/lines (*enoise:lod0TopLine* representing the top of the noise barriers and *enoise:lod0BaseLine* representing the base of the noise barriers) and surfaces (*enoise:lod1MultiSurface*).

9.6 Datasets used and Implementation

9.6.1 Datasets used

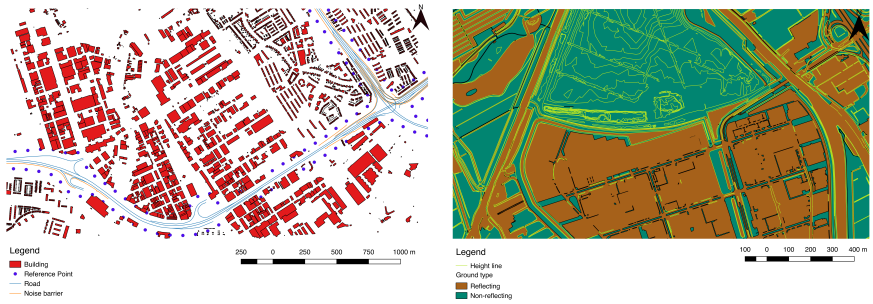
The following input data were used for testing the usability of the ADE:

- *Noise sources*, i.e. data about the roads and road segments, noise barriers and the reference points placed along side the roads in an area in the Netherlands. The reference points are always 50m away from the road, 4m high, and 100m apart [215]. This data was taken from the *geluidregister*[5], the national noise register of the Netherlands.

- *Other built-environment data*, i.e. data about the buildings, terrain, and ground types with noise reflection/absorption factors for an area in the Netherlands. This data was created as a part of the ongoing project *"Automated reconstruction of 3D input data for noise studies"*[6] in collaboration with Rijkswaterstaat, RIVM, Kadaster, and the 3D geoinformation research group, TU Delft.

Figure 9.9 depicts the input data used used in this research.

[5] https://www.rijkswaterstaat.nl/kaarten/geluidregister.aspx
[6] https://3d.bk.tudelft.nl/opendata/noise3d/en.html



(a) Buildings, roads, noise barriers, and reference points

(b) Ground types and height lines

Figure 9.9: Input data used for testing the ADE.

9.6.2 Implementation

An open source Python prototype is developed that automatically structures the aforementioned input datasets (described in Section 9.6.1) to comply with our CityGML eNoise ADE format for data storage, 3D noise analysis, and visualisation. Figure 9.10 depicts the generated 3D city model in our CityGML eNoise ADE.

Buildings

The dataset contains 2D footprints for the buildings with height information. These footprints are extruded as solids in 3D using the given height, thus yielding LOD1 building models.

Roads and reference points

The dataset contains road segments represented as lines with information about the width of the road, height of the source, traffic flow, etc. The road segments are also modelled as lines in the ADE. The reference points placed along the length of the roads are also stored with their geometry, height, and the actual and maximum noise value observed at these points. Snippet 1 depicts the generated noise related input attributes and their values for the road segments in the CityGML eNoise ADE.

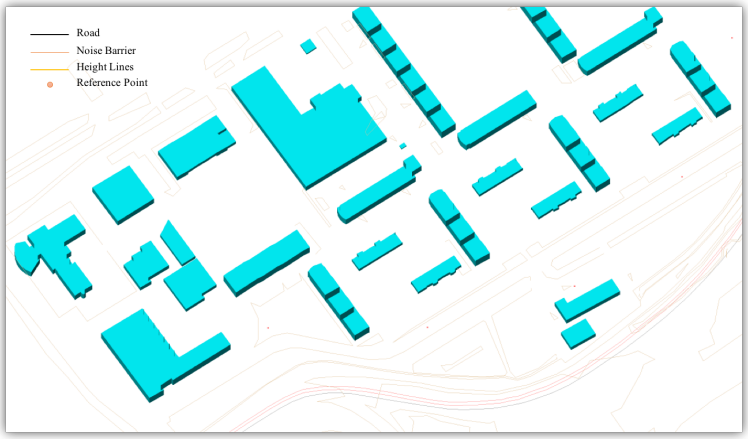


Figure 9.10: 3D city model of the study area in our CityGML eNoise ADE depicting buildings, roads, noise barriers, and height lines. Roads and noise barriers are modelled as 3D lines; Buildings are modelled as 3D solids; Terrain is modelled as 3D height lines

```
<CityModel>
  <cityObjectMember>
  <!-- CityGML Road -->
    <trn:Road gml:id="road01">
      <!-- Noise Road Segment -->
      <enoise:noiseRoadSegmentProperty>
        <enoise:noiseRoadSegment gml:id="GML_36595">
          <enoise:roadSurfaceMaterial>214</enoise:roadSurfaceMaterial>
          <enoise:sourceHeight uom="m">0.75</enoise:sourceHeight>
          <enoise:roadWidth uom="m">5.0</enoise:roadWidth>
          <!-- Traffic flow -->
          <enoise:mDay>
            <enoise:lightMotor>1037.0</enoise:lightMotor>
            <enoise:mediumHeavyMotor>22.0</enoise:mediumHeavyMotor>
            <enoise:heavyMotor>30.0</enoise:heavyMotor>
            <enoise:electricMotor>10.0</enoise:electricMotor>
          </enoise:mDay>
          <enoise:mEvening>
            ...
          </enoise:mEvening>
          <enoise:mNight>
            ....
          </enoise:mNight>
          <!-- Speed of vehicles -->
          <enoise:speedDay>
```

```
          <enoise:lightMotor uom="kmph">100</enoise:lightMotor>
          <enoise:mediumHeavyMotor uom="kmph">80</enoise:mediumHeavyMotor>
          <enoise:heavyMotor uom="kmph">80</enoise:heavyMotor>
          <enoise:electricMotor uom="kmph">40.0</enoise:electricMotor>
        </enoise:speedDay>
        <enoise:speedEvening>
          <enoise:lightMotor uom="kmph">100</enoise:lightMotor>
          <enoise:mediumHeavyMotor uom="kmph">80</enoise:mediumHeavyMotor>
          <enoise:heavyMotor uom="kmph">80</enoise:heavyMotor>
          <enoise:electricMotor uom="kmph">40.0</enoise:electricMotor>
        </enoise:speedEvening>
        <enoise:speedNight>
          <enoise:lightMotor uom="kmph">100</enoise:lightMotor>
          <enoise:mediumHeavyMotor uom="kmph">80</enoise:mediumHeavyMotor>
          <enoise:heavyMotor uom="kmph">80</enoise:heavyMotor>
          <enoise:electricMotor uom="kmph">40.0</enoise:electricMotor>
        </enoise:speedNight>
        <!-- Geometry -->
        <enoise:lod0BaseLine>
          <gml:LineString>
            ....
          </gml:LineString>
        </enoise:lod0BaseLine>
      </enoise:noiseRoadSegment>
    </enoise:noiseRoadSegmentProperty>
    ....
  </trn:Road>
 </cityObjectMember>
```

Snippet 1: CityGML *Road* extended with the input noise attributes in the eNoise ADE

Noise barriers

The dataset contains noise barriers represented as lines with information about its type, height, length, reflection factors, etc. The noise barriers can be stored in two ways in our model: as 3D surfaces extruded using the height information (see Figure 9.11) or as 3D lines representing the top and the bottom of the barriers (see Figure 9.10). Snippet 2 depicts the noise barriers in the CityGML eNoise ADE.

```
<CityModel>
  <cityObjectMember>
  <!-- CityGML Noise Barrier -->
    <frn:CityFurniture gml:id="cf01">
    <enoise:noiseCityFurniture>
      <enoise:NoiseBarrier gml:id="GML_1041">
        <enoise:barrierType>Screen</enoise:barrierType>
        <enoise:barrierLength uom="m">20.84</enoise:barrierLength>
```

```
        <enoise:barrierHeight uom="m">7.0</enoise:barrierHeight>
        <!-- Right Reflection factors -->
        <enoise:refFactorR>
          <enoise:ReflectionFactor>
            <enoise:rf63Hz>0.2</enoise:rf63Hz>
            <enoise:rf125Hz>0.2</enoise:rf125Hz>
            <enoise:rf500Hz>0.2</enoise:rf500Hz>
            <enoise:rf1kHz>0.2</enoise:rf1kHz>
            <enoise:rf2kHz>0.2</enoise:rf2kHz>
            <enoise:rf4kHz>0.2</enoise:rf4kHz>
            <enoise:rf8kHz>0.2</enoise:rf8kHz>
          </enoise:ReflectionFactor>
        </enoise:refFactorR>
        <enoise:floating>1</enoise:floating>
        <!-- Geometry of the barrier -->
        <enoise:lod0BaseLine>
          <gml:LineString>
            <gml:posList srsDimension="3"> ... </gml:posList>
          </gml:LineString>
        </enoise:lod0BaseLine>
        <enoise:lod0TopLine>
          <gml:LineString>
            <gml:posList srsDimension="3"> ... </gml:posList>
          </gml:LineString>
        </enoise:lod0TopLine>
        <!-- Left Reflection factors -->
        <enoise:refFactorL>
          <enoise:ReflectionFactor>
            <enoise:rf63Hz>0.2</enoise:rf63Hz>
            <enoise:rf125Hz>0.2</enoise:rf125Hz>
            <enoise:rf500Hz>0.2</enoise:rf500Hz>
            <enoise:rf1kHz>0.2</enoise:rf1kHz>
            <enoise:rf2kHz>0.2</enoise:rf2kHz>
            <enoise:rf4kHz>0.2</enoise:rf4kHz>
            <enoise:rf8kHz>0.2</enoise:rf8kHz>
          </enoise:ReflectionFactor>
        </enoise:refFactorL>
      </enoise:NoiseBarrier>
    </enoise:noiseCityFurniture>
    </frn:CityFurniture>
  </cityObjectMember>
</CityModel>
```

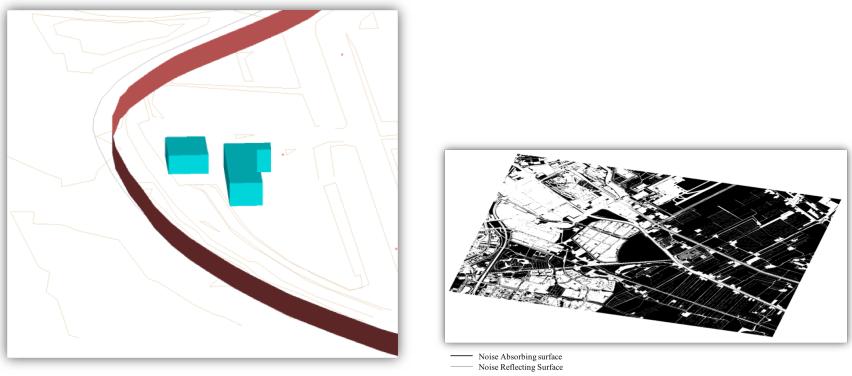
Snippet 2: Noise barriers modelled as 3D surfaces in the eNoise ADE

Terrain

The input data contains terrain as height lines (which are not isolines or contour lines or breaklines) [240]. We modelled these as 3D lines i.e. the CityGML *LineRelief*.

Ground type

The noise absorption property of the ground surface is modelled as the characteristic property of the land and is represented using the CityGML *LandUse* class (see Figure 9.11).



(a) Noise barriers as 3D surfaces

(b) CityGML *LandUse* for ground noise absorption/reflections

Figure 9.11: Noise barriers and ground types in the generated 3D city modelE.

Simulation output

The output noise levels generated by the noise simulations can also be stored in the ADE. The noise levels are calculated based on the emission and location of the noise source and a 3D model of the environment (i.e. buildings, roads, noise barriers) that is used to determine the noise propagation. Geomilieu is a noise simulation software developed by the DGMR (the Hague, the Netherlands) for computing the noise levels, arising from road traffic, rails, industries and aircraft, in accordance with the noise assessment methods [58]. Geomilieu also implements CNOSSOS-EU for calculating the noise from the roads, industrial and railways. We prepared the data to be ran in Geomilieu for calculating the noise levels. Snippet 3 depicts the CityGML ADE *Building* enriched with the results of the noise simulation.

```
<!-- CityGML Building -->
<bldg:Building gml:id="building01">
  <bldg:lod1Solid>
    .... <!-- Geometry here.. -->
```

```
  </bldg:lod1Solid>
  <bldg:address>
    .... <!-- Address here.. -->
  </bldg:address>
  <!-- roof type from the codelist -->
  <bldg:roofType>1000<bldg:roofType>
  <bldg:measuredHeight uom="m">8.60</enoise:measuredHeight>
  <bldg:numberOfStoreysAboveGround>2</enoise:numberOfStoreysAboveGround>
  <!--eNoise ADE attributes -->
  <enoise:groundAbsoluteHeight uom="m">-1.73</enoise:groundAbsoluteHeight>
  <enoise:buildingRefCorrection uom="dB">4.12</enoise:buildingRefCorrection>
  <enoise:buildingLDenMax uom="dB">85.12</enoise:buildingLDenMax>
  <enoise:buildingLDenMin uom="dB">44.12</enoise:buildingLDenMin>
  <enoise:buildingLDenEq uom="dB">58.12</enoise:buildingLDenEq>
  <enoise:buildingLNightMax uom="dB">56.12</enoise:buildingLNightMax>
  <enoise:buildingLNightMin uom="dB">39.12</enoise:buildingLNightMin>
  <enoise:buildingLNightEq uom="dB">43.12</enoise:buildingLNightEq>
  <enoise:buildingL24Max uom="dB">88.64</enoise:buildingL24Max>
  <enoise:buildingL24Min uom="dB">35.76</enoise:buildingL24Min>
  <enoise:numberOfHabitants>22</enoise:numberOfHabitants>
  <enoise:numberOfReceivers>5</enoise:numberOfReceivers>
  ....
</bldg:Building>
```

Snippet 3: CityGML *Building* extended with the output noise attributes in the eNoise ADE

9.7 Conclusion

In this chapter, I have described the research about the implementation of an harmonised input/output data model for urban noise simulations in the EU. Further, I have provided an inventory of the noise assessment methods and guidelines followed in the EU member states for the third round of the strategic noise mapping in 2017 (Table 9.1). I have discussed the challenges encountered while preparing this inventory and described how the heterogeneity in the assessment methods and input data used, can make it difficult to obtain comparable noise levels across the EU.

Further, it is discussed how the adoption of the CNOSSOS-EU framework can help in increasing the consistency among the results of the noise simulation studies across the EU. Finally, a CityGML-based harmonised data model is presented for modelling the inputs and outputs of noise simulations based on CNOSSOS-EU. In addition, a dataset in the ADE format is generated for an area in the Netherlands to model the input/output data for noise simulation. Evaluating the proposed data model against the requirements mentioned in Section 9.2.2, this model:

- stores the input/output data for noise simulation,
- has a UML, and an XSD (XML schema),
- is open, extensible and INSPIRE compliant. Further additions can be made to the model by following the rules for extending the CityGML schema,
- is GML-compliant and combines geometry and semantics of noise relevant city objects in one data model,
- has metadata for noise ADE datasets,
- can be used for validating the noise ADE datasets.

EU Members States are at a turning point and the opportunity to set up a common data model for input/output data of noise simulation using CNOSSOS-EU should be seized before the next round of strategic noise mapping. Members States should be ready to implement the next mapping round, testing the new framework. This research is a stepping stone in the direction of standardising the input data for noise simulations to obtain comparable outcomes of different noise simulations using CNOSSOS across the EU.

Table 9.1: Noise assessment methods and guidelines followed in the EU member states for the Round #3 of strategic noise mapping (2017) (Note: "–" means that data is not available) (Src: http://cdr.eionet.europa.eu/nl/eu/noise/)

#	EU MS	Road Noise	Rail Noise	Industrial Noise	Aircraft Noise
1	Austria	RVS 04.02.11 [79]	ONR 305011 [7]	ÖNORM ISO 9613-2 [6]	ÖAL Guideline No 24-1 [5]
2	Belgium	NMPB-Routes-2008 (Wallonia), RMV [53] (Flanders)	RMR (Wallonia & Flanders)	ISO 9613-2 (Wallonia & Flanders)	Integrated Noise Model (INM) version 7.0b [28]
3	Bulgaria	NMPB-Routes-96 [40]	RMR	ISO 9613-2	ECAC.CEAC Doc 29 - 2nd Edition
4	Croatia	NMPB-Routes-96	RMR	ISO9613-2	ECAC.CEAC Doc 29 - 3rd Edition [69]
5	Cyprus	–	–	–	–
6	Czech Republic	NMPB-Routes-96	RMR	ISO 9613-2	–
7	Denmark	Nord2000 [115]	Nord2000	DanishEPA 5/1993 [46]	DANSIM [208]
8	Estonia	NMPB-Routes-96	RMR	ISO 9613-2	–
9	Finland	CNOSSOS-EU	CNOSSOS-EU	CNOSSOS-EU	ECAC.CEAC Doc 29 - 3rd Edition
10	France	NMPB-Routes-2008	NMPB-Routes-2008	ISO 9613-2	ECAC.CEAC Doc 29 - 3rd Edition
11	Greece	NMPB-Routes-96	RMR	–	ECAC.CEAC Doc 29 - 2nd Edition

12	Germany	VBUS (Vorläufige Berechnungsmethode für den Umgebungslärm an Straßen) [37]	VBUSch (Vorläufige Berechnungsmethode für den Umgebungslärm an Schienenwegen) [38]	VBUI (Vorläufige Berechnungsmethode für den Umgebungslärm durch Industrie und Gewerbe) [36]	VBUF (Vorläufige Berechnungsmethode für den Umgebungslärm an Flugplätzen) [35]
13	Hungary	UT2.1-302 [131]	MSZ-07-2904:1990 [171]	Hungarian national computation method according to KvVM Decree 25/2004 (XII. 20.) on the Required Form and Content of Strategic Noise Maps Used for the Evaluation and Management of Environmental Noise and MSZ 15036:2002 [170]	Hungarian national computation method according to joint KHVM-KTM Decree 18/1997. (X. 11.) on Detailed Technical Rules of Designation, Management and Termination of Noise-Protective Zones in the Vicinity of Airports
14	Ireland	UK CRTN 1988	RMR	–	ECAC.CEAC Doc 29 - 3rd Edition and Integrated Noise Model (INM) Version 7.0d
15	Italy	NMPB-Routes-96	RMR	–	ECAC.CEAC Doc 29 - 3rd Edition and Integrated Noise Model (INM) Version 7.0d
16	Latvia	NMPB-Routes-96	RMR	ISO 9613-2	ECAC.CEAC Doc 29 - 3rd Edition

17	Lithuania	NMPB-Routes-96	RMR	ISO 9613-2	ECAC.CEAC Doc 29 - 3rd Edition
18	Luxembourg	NMPB-Routes-96	RMR	Courrier d'information de la Commission concernant la cartographie stratégique du bruit des sites d'activité' industrielle au sein de l'agglomération de la Ville de Luxembourg et environs [142]	ECAC.CEAC Doc 29 - 3rd Edition
19	Malta	NMPB-Routes-96	–	ISO 9613-2	ECAC.CEAC Doc 29 - 3rd Edition
20	Netherlands	RMV	RMR	Handleiding meten en rekenen Industrielawaai (HMRI) [166]	ECAC.CEAC Doc 29 - 3rd Edition
21	Poland	NMPB-Routes-96	RMR	ISO 9613-2	ECAC.CEAC Doc 29 - 3rd Edition
22	Portugal	NMPB-Routes-96	RMR	ISO 9613-2	ECAC.CEAC Doc 29 - 3rd Edition, Integrated Noise Model (INM) Version 7.0d, and AIR 1845 [220]
23	Romania	–	–	–	–
24	Slovakia	NMPB-Routes-96	Shall03 [167]	ISO 9613-2	ECAC.CEAC Doc 29 - 3rd Edition
25	Slovenia	NMPB-Routes-96	RMR	ISO 9613-2	ECAC.CEAC Doc 29 - 3rd Edition

26	Spain	NMPB- Routes-96	RMR	ISO 9613-2	ECAC.CEAC Doc 29 - 2nd Edition
27	Sweden	Nordic Pred. Method for Road Traffic Noise (RTN) [177]	Nordic Pred. Method for Train Noise (NMT) [217]	Environmental Noise from Industrial Plants. General Prediction Method [130]	ECAC.CEAC Doc 29 - 3rd Edition and Integrated Noise Model (INM) Version 7.0d
28	UK	CRTN (Calculation of Road Traffic Noise) [55]	CRN (Calculation of Railway Noise) [56]	–	Integrated Noise Model (INM) Version 7.0d

Part V

Conclusions and future work

Chapter 10

Conclusions and future prospects

10.1 Key findings and contributions

Cities and local governments have been increasingly adopting 3D city models in recent years. The increasing applications of 3D city models clearly suggests that they offer additional insight and allow new applications when using conventional 2D maps [19]. Further, the possibility to enrich these city models with additional application-specific information, such as new city objects or attributes, increases their usability for environmental analysis. Such semantically enriched 3D city models go beyond visualisation and have the potential to become hubs of integrated information for spatial analysis.

During the course of this thesis, I have observed that GIS practitioners involved in the development and use of 3D city models of large cities often encounter several issues, such as massive size of these 3D city models, insufficient or missing specifications, 3D data interoperability issues, lack of metadata, and inconsistencies in the 3D city models coming from heterogeneous sources. In addition, a number of standards and formats (XML, JSON, Graphics formats, etc.) are available to store 3D city models, which differ in their underlying data model for data representation. This hinders the reuse of once collected 3D data. Substantial information can be altered or lost when converting 3D city models between different formats. The unavailability of one-to-one mapping of city objects and attributes between different standards e.g. IFC and CityGML, also complicates the process of automatic conversion between different formats. The integration of highly detailed and differently structured BIM IFC models with 3D GIS CityGML models needs further attention in order for 3D city models to serve as 'digital twins' of reality and provide information for a wide variety of applications. Furthermore, unclear, too flexible, or missing specifications in a standard also pose a deal of challenge when modelling city objects in 3D, such as lack

of distinguishable LOD representation for terrain (LOD0-4), landuse (LOD0-4), vegetation (LOD2-4), roads (LOD2-4) in CityGML.

Based on the aforementioned points, it may seem as a lot still needs to be done. In this thesis, I have tried to touch upon these points and present a solution direction. The main research question is answered by sub-diving it into multiple research questions. In this section I answer these research questions, with references to the chapters and contributions.

Q1. How to efficiently store and manage massive terrains in the context of 3D city models (in flat files and database systems)?

Through extensive review of the data model of CityGML and practical implementations, I have found out several problems associated with storing massive TIN terrains in CityGML [134]. I have discussed these problems in detail in Chapter 3 of this thesis. One such problem that I have covered in Chapter 3 is how to deal with the massive size of TIN terrain models in the context of 3D city modelling. CityGML datasets can become very large because of the redundancy in the underlying data structure, which greatly hinders web-based rendering, exchange, and use of data in applications. This problem has been highlighted by many researchers but has not been investigated before to a great extent.

I have shown in this thesis that CityGML, using Simple Features for modelling the geometry of 3D city objects, is not efficient for storing massive terrains in the context of 3D city models. I investigated a number of data structures for the compact representation of TINs and explored how they can be integrated in the data model of CityGML. Since CityGML is an XML-based data model, introducing highly compressed data structures would require more preprocessing and later on extensive decoding for comprehensibility. Therefore, I have only considered solutions that fit in the data model of CityGML. In Chapter 3, I have introduced the selected compact data structures *(indexed triangle, triangle strips, stars)* in the data model of CityGML as an ADE for representing TINs and did tests to compare their performance against "plain" CityGML datasets. I have also developed a prototype that automatically converts CityGML terrain datasets to the ADE implementation. In Chapter 4, I have implemented these data structures in 3DCityDB to efficiently store massive TIN terrains in a database.

Q2. How can we model a terrain at different levels of detail in a 3D city model?

I have found that, even within the same standard (CityGML 2.0), there is a lack of precise definition of LODs for features such as landuse and terrain. In Chapter 5, I have proposed an LOD framework for terrains represented as TINs in CityGML. The proposed LOD framework is simple and compliant with the existing LOD concept in CityGML and is meant to improve the ambiguity of

the current concept. It is based on examining real-world datasets, standards, documentations, and discussions with GIS practitioners. It does not restrict the LODs to the geometric data granularity in values, such as the number of points/triangles; these values are often arbitrary or application/user specific. Rather, the approach aimed to integrate the terrain with surrounding features while adding further geometric and semantic information.

The research also takes into account the work being done in the upcoming version of CityGML (version 3.0), such as phasing out the LOD4 representation of features [141, 156]. I have also developed an experimental prototype to realise the proposed terrain LOD specifications. The prototype generates artificial CityGML terrain models at different LODs and was released as open source, so that other researchers can benefit from the different LOD terrain datasets in their application domains. The approach is based on the procedural modelling engine *Random3DCity* developed by Biljecki et al. [22] for generating random CityGML buildings in multiple LODs. In addition, sample datasets produced by our prototype have been released as open data.

Further experiments with real-world datasets are required for understanding how different terrain LODs can affect the accuracy of a spatial analysis. Modelling data at finer LODs comes at a higher cost, and whether spatial analyses can take advantage of this finer detail is an open question. However, an increase in complexity (i.e. using a higher LOD) can come at the expense of usability, interoperability, and maintenance.

Q3. How can the development of new multi-disciplinary standards such as LandInfra and InfraGML contribute to the re-use of data amongst different application domains such as BIM and 3D GIS?

The BIM and 3D GIS domains are often faced with the data interoperability issues when converting 3D city models between the IFC and the CityGML standard. The integration of IFC and CityGML can avoid unnecessary efforts in redundant modelling with focus on reusing the available data. For instance, models of buildings are produced in both standards for different applications, such as design and construction in BIM, and spatial analysis in GIS. With BIM and GIS integration, more detailed 3D city models can be built by reusing the BIM data.

Much work has already been done on this integration by developing data converters for IFC and CityGML. In addition, new standards have been recently developed that integrate concepts from different standards to represent an integrated semantic 3D city model. In Chapter 6, I explored one such standard, OGC LandInfra and its GML encoding InfraGML, to see its potential to bring the architectural, BIM, and geospatial views onto a common footing. It was

designed as a 'connecting bridge' between the BIM and GIS domains. I have compared LandInfra with CityGML and IFC to analyse the differences and similarities between the standards. I have also highlighted some issues that were encountered during the evaluation of LandInfra.

Q4. How to harmonise the already existing (i.e. CityGML/IFC) and the newly developed open standards (i.e. LandIfra/InfraGML) for semantically rich 3D city models?

LandInfra is a relatively new OGC open standard and is a more powerful standard than CityGML in some areas, as it has a much more detailed representation for land and infrastructure features. CityGML is widely used in the 3D GIS community, has software support, and an official mechanism to extend its data model via Generics and ADE. Therefore, in order to show the applications of LandInfra and encourage its adoption, I have developed an harmonised mapping between CityGML and LandIfra and implemented it as a CityGML ADE (Chapter 7). The idea behind the integration is to take the best of both worlds (i.e. CityGML and LandInfra) and have more information than CityGML can represent for specific applications or use cases such as urban environment analysis, subsurface modelling, etc. Furthermore, LandInfra has no concrete real-world datasets. To address this issue, I have developed prototypes can help practitioners to generate valid real-world sample datasets for use in different applications.

Q5. How to effectively model the metadata associated with 3D city models for 3D data discovery and dissemination?

Since more and more 3D city models are being created and used, metadata can play an effective role in the management, retrieval, and dissemination of these models. However, after extensive review, it was inferred that CityGML lacks precise specifications for modelling metadata of 3D city models. Documenting the metadata of 3D city models has several benefits. First, it can ensure that creators and users of these models from different 3D domains can understand and communicate about data requirements and its usability. Second, the presence of metadata is crucial for *3D data discovery*. With the current lack of precise definitions for metadata, practitioners often need to define their own definitions for CityGML metadata and create their own methodology for storing it. In Chapter 8, I have examined the metadata needs specific to 3D city models and presented an ISO 19115 compliant solution to add metadata to the 3D city models represented in CityGML. CityGML files would most benefit from having an explicitly stored (structured) metadata.

I also developed a prototype that automatically generates the metadata for CityGML datasets according to the proposed definition, such as the levels of detail present, thematic models, extent, etc. It parses a dataset to check if other

metadata information is present and has default values to indicate which values are missing. Having metadata attached to the CityGML file will help in assessing the fitness-for-purpose of data for use within an application.

Q6. How to develop a semantically enriched data model for integrating data from different sources for urban noise simulation?

The 3D city models supplemented with application specific information can greatly aid environmental simulations. Urban noise simulation is one such application where the use of 3D city models has gained popularity. CityGML documentation already provides specifications for modelling urban noise in 3D as an ADE. However, I have identified in Chapter 9 that current CityGML Noise ADE has limitations. For instance, it is set in a German context based on the German regulations for modelling noise caused by roads and railways only. I have presented in this thesis the development of a harmonised semantic 3D city model based on CityGML for use in urban noise simulations in EU. Different EU member states employ different noise assessment methods, which differ significantly in their data requirements and methodology for estimating noise levels (Table 9.1). The heterogeneity in these methods and utilised input data makes it difficult to obtain comparable results across the EU. To address this problem, CNOSSOS-EU was developed by the European Commission to enable a consistent and accurate reporting of strategic noise mapping by the EU Member States. I have extended and restructured the existing CityGML Noise ADE with new classes, attributes and other concepts in accordance with the CNOSSOS-EU. In addition, a dataset in the ADE format is generated for an area in the Netherlands to model the input/output data for noise simulation using CNOSSOS-EU.

10.2 Reflection and future prospects

10.2.1 Reflection

The ability to replicate the physical world in a 3D model has become a valuable asset for a wide range of urban applications. The current challenges stand in the way of efficiently utilising 3D city models. A lot of 3D city models are available at different LODs and time periods for use in different applications. It is therefore important to store these 3D city models in a standardized manner with semantics and metadata for use in different applications.

The recently accepted OGC community standards 3D Tiles [200] and I3S (Indexed 3D Scene Layers) [201] are able to represent semantic 3D city models as well. These standards were not covered in the time frame of this thesis. However, I explored glTF (one of the base formats of 3D Tiles), which also stores

the geometry of a 3D city model in binary format. I believe that these are quite powerful data models and can go a long way in compactly storing and visualising 3D city models. 3D Tiles and I3S encode both geometry as well as semantics with focus on web-based streaming and visualization of large 3D datasets. Both support binary encoding of triangle geometry which is perfect match for representing and storing massive TINs. However, these 3D geometries are not based on ISO 19107. There are no complex ISO 19107 geometry types such as solids, multisolids, etc.

Requirements for 3D data interoperability have been partly fulfilled by commercial vendors and open source initiatives, which try to facilitate seamless integration via import/export capabilities from one data format to another. However, such support for standards like LandInfra/InfraGML do not exist yet. In this context, the fact that a standard has been tested and implemented in code is a positive feature of the standardization approach, which increases the usability of the standard. In fact, in 2018, there was a discussion in the OGC mailing list regarding the importance of having a reference software implementation for the OGC standards. With this research, I have tried to get the GIS community to acknowledge the missing software support for existing standards and aimed at providing one. Having a reference implementation is invaluable to see the standard in action, both from a functional viewpoint (e.g. whether it is aligned with needs of the users), as well as from a technical standpoint (e.g. system architecture, implementability, performance). Without open implementations and greater concern for the implementability of the standard, there is a danger that the standard will languish and be unused.

Another concept which does not sound thrilling but is crucial for 3D data discovery is metadata. The CityGML standard currently does not offer a mechanism to store metadata in a structured way. However, other OGC standards such as LandInfra has ISO 19115 compliant metadata in its data model. Given the size and complexity of most CityGML datasets, having access to metadata would assist users in quickly understanding the nature of a dataset and its fitness-for-purpose. The approach explored in this thesis for adding metadata to CityGML files is based on the well-developed CityGML ADE mechanism. The major advantage of developing the metadata ADE is that creating an ADE does not need a formal approval by any standardisation body. It is a good way to depict working proof of concepts for adding application specific objects and attributes missing in CityGML. Due to the large file size of most CityGML files, the prototype generates metadata as a separate file which ensures faster parsing but users can write to the original file if they wish to. Given the open nature of CityGML and its collaborative evolution process, this ADE could serve as a template for adding metadata in the next version of CityGML (version 3.0). The same schema for metadata is also implemented in CityJSON. However, I have translated manually the schema for metadata from CityGML ADE XSD

to JSON format for CityJSON. Similarly, a separate extension or translation is required for a different standard represented in a different format. There is a need for a universally accepted data model for representing metadata which can be reused for various standards for 3D city modelling. Having a well-defined data model for metadata can aid data-creators by providing neat guidelines to follow. It can be considered as a first step towards a further discussion on the metadata needs of future 3D SDI.

10.2.2 Future prospects

Further investigation of compact TIN representations In Chapter 3, I investigated three data structures, namely *indexed triangle*, *triangle strips* and *stars* and explored how they can incorporated in the data model of CityGML for an improved representation of TIN terrains. However, there are many other data structures for TIN representation such as *half-edge or DCEL* [163, 172], *SQuad* [86], *grouper* [159], *Laced Ring (LR)* [87], *zipper* [88], and *tripod* [228], which I did not cover in this thesis. These data structures are also capable of reducing the storage requirements for a TIN. They can be useful for streaming and visualization of large TINs. An investigation into their implementation in CityGML and/or 3DCityDB can be insightful to determine their usability for different 3D city modelling applications. Another interesting investigation can be to look into the standards which offer binary representation of triangle geometry such as 3D Tiles and I3S.

Further investigation into missing LOD definitions In Chapter 5, I have proposed an LOD framework for terrains modelled as TINs in CityGML. It is compliant with the existing LOD concept in CityGML, and is meant to improve the ambiguity of the current concept for terrains. However, the focus in our research was on CityGML *TINRelief* representation for terrains. There are three other terrain representations in CityGML, namely *RasterRelief* for grids, *BreaklineRelief* for breaklines, and *MassPointRelief* for mass points. My research did not investigate the LODs for these terrain representations. It is also interesting to investigate if it is reasonable to have an LOD definition that would be applicable to the entire *Relief* module or separate LOD definitions for each terrain representation type. Furthermore, there are other city objects with no or unclear specifications which can be investigated. For instance, landuse and vegetation objects have no LOD specifications. Similarly, for roads, CityGML has LOD0 and LOD1 representation but there is no distinction between L0D2 to LOD4.

Possibilities with LandInfra and InfraGML In Chapter 6 and Chapter 7, I explored LandInfra (and InfraGML), designed as a 'connecting bridge' between the BIM and 3D GIS domains. LandInfra is a relatively young standard and

does not have concrete real-world datasets, and no software support in the geospatial community. We do not want LandInfra (and InfraGML), a connecting link between BIM and GIS, to suffer because of low support and no reference implementation. While many academic research has demonstrated IFC-CityGML transformations, nothing has been done so far for IFC-LandInfra/InfraGML and CityGML-LandInfra/InfraGML transformations. In my research work, I developed a utility that converts CityGML datasets to InfraGML to help practitioners generate valid real-world sample InfraGML datasets. However, support tools such as parsers, validators, visualisers, DBMS support, APIs, and so on are still lacking. It would be interesting to see how the standards evolve if such support is available. Furthermore, LandInfra does not have the concept of LODs and ADEs in its data model, which can be explored. Interoperability of LandInfra with IFC and other standards is also an open area of research.

Potential recommendations for BIM-GIS integration Many research works have demonstrated successful IFC-CityGML transformations. However, they have mostly developed their own specific solution with only their use cases or data available at hand. There is a need to define one uniform and standardised transformation for IFC-CityGML since at present there are several different interpretations available for how to best convert an IFC dataset to CityGML. Identifying real-world use cases can aid the process of defining the guidelines for this standardisation [3]. In addition, supplementing these use cases with complete real-world datasets can help in better understanding the disparities between the BIM-GIS standards.

Metadata for ADEs The current metadata ADE for CityGML includes basic attributes to store the metadata about the ADEs present in the dataset such as: the name of the ADE and its version, URI of the UML and XML schema and any other available documentation. However, metadata about the new city objects introduced by an ADE is not stored e.g. Energy ADE, eNoise ADE, etc. Future work regarding metadata for CityGML datasets can include determining how metadata ADE can extended for the application-specific metadata.

Opportunities with CityJSON The problem of storing massive datasets in CityGML is evident in this thesis. CityGML was not designed for visualizing 3D city models on the web. Visualizing CityGML in a web browser requires to follow another pathway of separating the geometric information from the semantic part and transforming it to commonly used 3D graphics formats for visualization. On the other hand, CityJSON is designed to be compact and friendly for web and mobile development. CityJSON already offer 6X more compression for storing 3D city models than their CityGML equivalent [150]. The geometry representation that is followed

in CityJSON is similar to *'indexed triangle'*. An investigation into the integration of *triangle strips* and *stars* in CityJSON can be interesting for an improved representation of TIN terrains. CityJSON is relatively young in the 3D world (published in 2019) but has extensive open source software support such as parsers, validators, and visualizers (desktop and web-based). The database support for CityJSON now exists in 3DCityDB relational schema. Integration of CityJSON schema with a NoSQL (such as MongoDB[1]) or a graph database (such as Neo4j[2]) can be of interest to the open source 3D community.

[1] https://www.mongodb.com
[2] https://neo4j.com

Bibliography

[1] G. Agugiaro, J. Benner, P. Cipriano, and R. Nouvel. The energy application domain extension for CityGML: enhancing interoperability for urban energy simulations. *Open Geospatial Data, Software and Standards*, 3(1), 2018.

[2] AHN3. Actueel Hoogtebestand Nederland version 3. Retrieved from https://www.pdok.nl/nl/ahn3-downloads, 2015.

[3] K. Arroyo Ohori, A. Diakité, T. Krijnen, H. Ledoux, and J. Stoter. 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, 2018.

[4] B. Atazadeh, M. Kalantari, A. Rajabifard, S. Ho, and T. Ngo. Building Information Modelling for high-rise land administration. *Transactions in GIS*, 21(1):91–113, 2017.

[5] Austrian Standards. ÖAL Richtlinie Nr. 24 Blatt 1: 2008 - Noise protection zones in the vicinity of airports - planning and calculation bases, 2008.

[6] Austrian Standards. ÖNORM ISO 9613-2: 2008 - Acoustics - Attenuation of sound during propagation outdoors - Part 2: General method of calculation, 2008.

[7] Austrian Standards. ONR 305011: 2009 - Determination of noise immission caused by rail traffic - Railway traffic, shunting and cargo handling operations, 2009.

[8] M. Barnes and E. L. Finch. COLLADA - Digital Asset Schema Release 1.5.0 Specification, 2008.

[9] T. Becker, C. Nagel, and T. H. Kolbe. Integrated 3d modeling of multi-utility networks and their interdependencies for critical infrastructure analysis. In *Advances in 3D Geo-Information Sciences*, pages 1–20. Springer, 2011.

[10] C. Beil, T. H. Kolbe, and . CityGML and the streets of New York-a proposal for detailed street space modelling. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-4/W5: 9–16, 2017.

[11] J. Benner, A. Geiger, G. Gröger, K.-H. Häfele, and M.-O. Löwner. Enhanced LOD concepts for virtual 3D city models. *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences*, II-2/w1: 51–61, 2013.

[12] R. Benocci, P. Bellucci, L. Peruzzi, A. Bisceglie, F. Angelini, C. Confalonieri, and G. Zambon. Dynamic noise mapping in the suburban area of rome (italy). *Environments*, 6(7):79, 2019.

[13] F. Bertellino, F. Gerola, M. Clementel, P. Scaramuzza, and M. Nardelli. Noise mapping of agglomerations: a comparison of interim standards vs. new cnossos method in a real case study. In *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, volume 253, pages 3639–3649. Institute of Noise Control Engineering, 2016.

[14] BGT. Basisregistratie Grootschalige Topografie. Retrieved from https://www.pdok.nl/nl/producten/pdok-downloads/download-basisregistratie-grootschalige-topografie, 2016.

[15] L. Bian. Multiscale nature of spatial data is scaling up environmental models. In 'Scale in Remote Sensing and GIS'. (Eds MF Goodchild, DA Quattrochi) pp. 13–25, 1997.

[16] F. Biljecki. *Level of detail in 3D city models.* PhD thesis, Delft University of Technology, Delft, the Netherlands, May 2017.

[17] F. Biljecki and K. Arroyo Ohori. Automatic semantic-preserving conversion between OBJ and CityGML. In *Eurographics Workshop on Urban Data Modelling and Visualisation*, volume 2015, pages 25–30, 2015.

[18] F. Biljecki, H. Ledoux, J. Stoter, and J. Zhao. Formalisation of the level of detail in 3D city modelling. *Computers, Environment and Urban Systems*, 48:1–15, 2014.

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

[20] F. Biljecki, H. Ledoux, X. Du, J. Stoter, K. H. Soon, and V. Khoo. The most common geometric and semantic errors in citygml datasets. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4(2W1):13–22, 2016.

[21] F. Biljecki, H. Ledoux, and J. Stoter. An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59: 25–37, 2016. doi: 10.1016/j.compenvurbsys.2016.04.005.

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

[23] F. Biljecki, H. Ledoux, and J. Stoter. An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59: 25–37, 2016.

[24] F. Biljecki, H. Ledoux, J. Stoter, and G. Vosselman. The variants of an LOD of a 3D building model and their influence on spatial analyses. *ISPRS Journal of Photogrammetry and Remote Sensing*, 116:42–54, 2016.

[25] F. Biljecki, K. Kumar, and C. Nagel. CityGML Application Domain Extension (ADE): overview of developments. *Open Geospatial Data, Software and Standards*, 2018. URL https://doi.org/10.1186/s40965-018-0055-6.

[26] C. Blanchet, C. Castaing, M. Beaufils, and D. Emmanuel. GeoBIM (MINnD) use case on an infrastructure acoustic study: feedback on the use of CityGML and InfraGML. https://portal.opengeospatial.org/files/?artifact_id=75554, 2017. Accessed 28 March 2019.

[27] D. K. Blandford, G. E. Blelloch, D. E. Cardoze, and C. Kadow. Compact representations of simplicial meshes in two and three dimensions. *International Journal of Computational Geometry and Applications*, 15(1):3–24, 2005.

[28] E. R. Boeker, E. Dinges, B. He, G. Fleming, C. J. Roof, P. J. Gerbi, A. S. Rapoza, J. Hermann, et al. Integrated Noise Model (INM) version 7.0 technical manual. Technical report, United States. Federal Aviation Administration. Office of Environment and Energy, 2008.

[29] J.-D. Boissonnat, O. Devillers, S. Pion, M. Teillaud, and M. Yvinec. Triangulations in CGAL. *Computational Geometry—Theory and Applications*, 22:5–19, 2002.

[30] L. Brink, J. Stoter, and S. Zlatanova. UML-based approach to developing a CityGML Application Domain Extension. *Transactions in GIS*, 17(6): 920–942, 2013.

[31] Brüel and Kjaer . Predictor-LimA software, 2018.

[32] R. Budiarto, P. Isawasan, and M. A. Aziz. Transformation of spatial data format for interoperability between gis applications. In *2009 Sixth International Conference on Computer Graphics, Imaging and Visualization*, pages 536–539. IEEE, 2009.

[33] buildingSMART. buildingSMART for Infrastructure. http://www.buildingsmart-tech.org/infrastructure/projects, 2015. Accessed 30 March 2019 .

[34] buildingSMART. Industry Foundation Classes 4.0.2.1 Version 4.0 - Addendum 2 - Technical Corrigendum 1. https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/, 2017. Accessed April 10 March 2019.

[35] Bundesministerium für Umwelt, Naturschutz und Reaktorsicherheit. VBUF - Vorläufige Berechnungsmethode für den Umgebungslärm an Flugplätzen, 2006.

[36] Bundesministerium für Umwelt, Naturschutz und Reaktorsicherheit. VBUI - Vorläufige Berechnungsmethode für den Umgebungslärm durch Industrie und Gewerbe, 2006.

[37] Bundesministerium für Umwelt, Naturschutz und Reaktorsicherheit. VBUS - Vorläufige Berechnungsmethode für den Umgebungslärm an Straßen, 2006.

[38] Bundesministerium für Umwelt, Naturschutz und Reaktorsicherheit. VBUSch - Vorläufige Berechnungsmethode für den Umgebungslärm an Schienenwegen , 2006.

[39] V. Cagdas, A. Kara, P. van Oosterom, C. Lemmen, Ü. Işıkdağ, R. Kathmann, and E. Stubkjær. An initial design of ISO 19152: 2012 LADM based valuation and taxation data model. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 145–154, 2016.

[40] CERTU, CSTB, LCPC, and SETRA. Bruit des infrastructures routières: méthode de calcul incluant les effets météorologiques, version expérimentale NMPB-Routes-96, 1997.

[41] CityGMLwiki[dot]org. CityGML Wiki. http://www.citygmlwiki.org, 2017. Accessed 30 January 2019.

[42] T. J. Cova and M. F. Goodchild. Extending geographical representation to include fields of spatial objects. *International Journal of geographical information science*, 16(6):509–532, 2002.

[43] A. Czerwinski, T. H. Kolbe, L. Plümer, and E. Stöcker-Meier. Interoperability and accuracy requirements for EU environmental noise mapping. In *International Symposium InterCarto–InterGIS 12, Berlin, Germany.*, 2006.

[44] A. Czerwinski, T. H. Kolbe, L. Plümer, and E. Stöcker-Meier. Spatial data infrastructure techniques for flexible noise mapping strategies. In *20th International Conference on Environmental Informatics-Managing Environmental Knowledge, Graz, Austria*, pages 99–106, 2006.

[45] A. Czerwinski, S. Sandmann, E. Stöcker-Meier, and L. Plümer. Sustainable SDI for EU noise mapping in NRW-best practice for INSPIRE. *IJSDIR*, 2:90–111, 2007.

[46] Danish Environmental Protection Agency. EPA Guideline No. 5/1993 - Calculation of External Noise from Enterprises, 1993.

[47] D. M. Danko. Geospatial metadata. In W. Kresse and D. M. Danko, editors, *Springer Handbook of Geographic Information*, pages 359–391. Springer Science & Business Media, 2010.

[48] E. Danovaro, L. De Floriani, P. Magillo, E. Puppo, and D. Sobrero. Level-of-detail for data analysis and exploration: a historical overview and some new perspectives. *Computers & Graphics*, 30(3):334–344, 2006.

[49] DataKustik GmbH. CadnaA, 2017.

[50] M. De Berg, M. Van Kreveld, M. Overmars, and O. C. Schwarzkopf. *Computational geometry*. Springer, 2000.

[51] H. de Kluijver and J. Stoter. Noise mapping and GIS: optimising quality and efficiency of noise effect studies. *Computers, Environment and Urban Systems*, 27(1):85–102, 2003.

[52] R. de Laat and L. van Berlo. Integration of BIM and GIS: The development of the CityGML GeoBIM extension. In *Advances in 3D geo-information sciences*, pages 211–225. Springer, 2011.

[53] De minister van Volkshuisvesting, Ruimtelijke Ordening en Milieubeheer. Reken- en meetvoorschrift Verkeerslawaai (RMV), 2002.

[54] De minister van Volkshuisvesting, Ruimtelijke Ordening en Milieubeheer. Reken- en Meetvoorschrift Railverkeerslawaai (RMR), 2012.

[55] Department of Transport Welsh Office. Calculation of Road Traffic Noise (CRTN), 1988.

[56] Department of Transport Welsh Office. Calculation of Railway Noise (CRN), 1995.

[57] E. Devys. Rail infrastructure: RailTopoModel and LandInfra interoperability. http://rtm.uic.org/wp-content/uploads/2018/06/10-ISN-18.063-10-Rail-infra-RailTopoModel-LandInfra-Interoperability-220518.pptx, 2018. 9th RailTopoModel Conference. Accessed 28 March 2019.

[58] DGMR. Geomilieu - berekenen en analyseren van milieubelasting, 2018.

[59] A. Diakité and J. Stoter. Eindrapport scoping studie voor integratie geotop en bim: Als input voor de ontwikkeling van basis registratie ondergrond. Technical report, Delft University of Technology, jun 2017.

[60] L. Dietze, U. Nonn, and A. Zipf. Metadata for 3D city models analysis of the applicability of the ISO 19115 standard and possibilities for further amendments. In *Proceedings of the 10th AGILE International Conference on Geographic Information Science*, pages 1–9, 2007.

[61] E. Directive. Directive 2002/49/EC of the European parliament and the Council of 25 June 2002 relating to the assessment and management of environmental noise. *Official Journal of the European Communities, L*, 189(18.07), 2002.

[62] E. Directive. Commission Directive (EU) 2015/996 of 19 May 2015 establishing common noise assessment methods according to Directive 2002/49/EC of the European Parliament and of the Council, 2015.

[63] S. Donkers, H. Ledoux, J. Zhao, and J. Stoter. Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings. *Transactions in GIS*, 20(4):547–569, 2016.

[64] S. Donkers, H. Ledoux, J. Zhao, and J. Stoter. Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings. *Transactions in GIS*, 20(4):547–569, 2016.

[65] G. Dutilleux, J. Defrance, D. Ecotière, B. Gauvreau, M. Bérengier, F. Besnard, and E. L. Duc. NMPB-ROUTES-2008: the revision of the French method for road traffic noise prediction. *Acta Acustica united with Acustica*, 96(3):452–462, 2010.

[66] E. Duval, W. Hodgins, S. Sutton, and S. L. Weibel. Metadata principles and practicalities. *D-lib Magazine*, 8(4):1082–9873, 2002.

[67] N. Dyn, D. Levin, and S. Rippa. Data dependent triangulations for piecewise linear interpolation. *IMA journal of numerical analysis*, 10(1):137–154, 1990.

[68] ECAC. ECAC.CEAC Doc 29 - 2nd Edition - Report on Standard Method of Computing Noise Contours around Civil Airports, 1997.

[69] ECAC. ECAC.CEAC Doc 29 - 3rd Edition - Report on Standard Method of Computing Noise Contours around Civil Airports, 2016.

[70] M. El-Mekawy. *Integrating BIM and GIS for 3D city modelling: The case of IFC and CityGML*. PhD thesis, KTH, 2010.

[71] M. El-Mekawy, A. Östman, and K. Shahzad. Towards interoperating CityGML and IFC building models: A unified model based approach. *Advances in 3D Geo-Information Sciences*, pages 73–93, 2011.

[72] M. El-Mekawy, A. Östman, and I. Hijazi. A unified building model for 3D urban GIS. *ISPRS International Journal of Geo-Information*, 1(2): 120–145, 2012.

[73] S. O. Elberink, J. Stoter, H. Ledoux, and T. Commandeur. Generation and dissemination of a national virtual 3D city and landscape model for the Netherlands. *Photogrammetric engineering & remote sensing*, 79(2): 147–158, 2013.

[74] C. Ellul, N. Tamash, F. Xian, J. Stuiver, and P. Rickles. Using Free and Open Source GIS to Automatically Create Standards-Based Spatial Metadata in Academia. *OSGeo Journal*, 13(1):51–59, 2014.

[75] R. Elmasri and S. Navathe. *Fundamentals of database systems*. Pearson, 2017.

[76] L. Emgård and S. Zlatanova. Design of an integrated 3D information model. In V. Coors, M. Rumor, E. Fendel, and S. Zlatanova, editors, *Urban and Regional Data Management — UDMS Annual 2007*. Taylor & Francis, 2008.

[77] European Environment Agency. Electronic Noise Data Reporting Mechanism: A handbook for delivery of data in accordance with Directive 2002/49/EC. Technical report, 2012.

[78] D. C. Finnegan and M. Smith. Managing lidar topography using oracle and open source geospatial software. *Proceedings GeoWeb*, 2010.

[79] Forschungsgesellschaft StraSSe - Schiene - Verkehr. 04.02.11 lärmschutz, 2006.

[80] Geoff Zeiss. BIM + geospatial interoperability would avoid another CAD + GIS quagmire. https://www.geospatialworld.net/blogs/bim-geospatial-interoperability-would-avoid-another-cad-gis-quagmire/, 2019.

[81] Geomod. MithraSIG: acoustic simulation software, 2018.

[82] M. F. Goodchild. Scale in gis: An overview. *Geomorphology*, 130(1-2):5–9, 2011.

[83] B. Gorte and J. Lesparre. Representation and reconconstruction of triangular irregular networks with vertical walls. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3826:15–19, 2012.

[84] G. Gröger and M.-O. Löwner. Proposal for a new LoD concept for CityGML 3.0. Technical report, CityGML OGC Work Package 03, 2016.

[85] G. Gröger and L. Plümer. How to get 3-D for the price of 2-D? topology and consistency of 3-D urban GIS. *Geoinformatica*, 9(2):139–158, 2005.

[86] T. Gurung, D. Laney, P. Lindstrom, and J. Rossignac. SQuad: Compact representation for triangle meshes. In *Computer Graphics Forum*, volume 30, pages 355–364. Wiley Online Library, 2011.

[87] T. Gurung, M. Luffel, P. Lindstrom, and J. Rossignac. *LR: compact connectivity representation for triangle meshes*, volume 30. ACM, 2011.

[88] T. Gurung, M. Luffel, P. Lindstrom, and J. Rossignac. Zipper: A compact connectivity data structure for triangle meshes. *Computer-Aided Design*, 45(2):262–269, 2013.

[89] K. Heutschi, B. Locher, and M. Gerber. sonROAD18: Swiss implementation of the CNOSSOS-EU road traffic noise emission model. *Acta Acustica united with Acustica*, 104(4):697–706, 2018.

[90] H. Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proceedings Visualization'98 (Cat. No. 98CB36276)*, pages 35–42. IEEE, 1998.

[91] F. Huang. CityGML-based 3D GIS visualization in the cloud environment. *Computer Modelling and New Technologies*, (18):91–98, 2014.

[92] C. Hug, P. Krzystek, and W. Fuchs. Advanced lidar data processing with LasTools. In *XXth ISPRS Congress*, pages 12–23, 2004.

[93] IFCwiki[dot]org. IFC Wiki. http://www.ifcwiki.org, 2018. Accessed 28 March 2019.

[94] INSPIRE TWG Buildings. D2.8.III.2 INSPIRE Data Specification on Buildings - Technical Guidelines, 2013.

[95] M. Isenburg, P. Lindstrom, S. Gumhold, and J. Snoeyink. Streaming Formats for Geometric Data Sets. Retrieved from http://www.cs.unc.edu/~isenburg/research/sm/download/streaming_formats.pdf, 2005.

[96] M. Isenburg, P. Lindstrom, and J. Snoeyink. Lossless compression of predicted floating-point geometry. *Computer-Aided Design*, 37(8):869–877, 2005.

[97] M. Isenburg, Y. Liu, J. Shewchuk, and J. Snoeyink. Streaming computation of Delaunay triangulations. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 1049–1056. ACM, 2006.

[98] U. Isikdag and S. Zlatanova. Towards defining a framework for automatic generation of buildings in CityGML using building Information Models. In *3D geo-information sciences*, pages 79–96. Springer, 2009.

[99] ISO. ISO 9613-2:1996 - Acoustics – Attenuation of sound during propagation outdoors – Part 2: General method of calculation, 1996.

[100] ISO. 19107: 2003(E) Geographic information: Spatial schema, 2003.

[101] ISO. ISO 19136:2007 Geographic information – Geography Markup Language (GML). 2007.

[102] ISO. Geographic information - Metadata - XML schema implementation (ISO/TS 19139:2007,IDT), Nov. 2009.

[103] ISO. ISO 19118:2011 Geographic information – Encoding. Geneva: Standard. 2011.

[104] ISO. ISO 19152:2012 Geographic information – Land Administration Domain Model (LADM), 2012.

[105] ISO. ISO/IEC 19775-1:2013 Information technology – Computer graphics, image processing and environmental data representation – Extensible 3D (X3D) – Part 1: Architecture and base components, 2013.

[106] ISO. ISO 3095:2013 –Acoustics – Railway applications – Measurement of noise emitted by railbound vehicles, 2013.

[107] ISO. *ISO 10303-105:2014 Industrial automation systems and integration – Product data representation and exchange.* International Organization for Standardization, Aug. 2014.

[108] ISO. ISO 19115-1:2014 Geographic information – Metadata – Part 1: Fundamentals, Feb. 2014.

[109] ISO. ISO/IEC 19775-2:2015 Information technology – Computer graphics, image processing and environmental data representation – Extensible 3D (X3D) – Part 2: Scene access interface (SAI), 2015.

[110] ISO. ISO 19103:2015 Geographic information – Conceptual schema language, 2015.

[111] ISO. ISO 19109:2015 Geographic information – Rules for application schema. 2015.

[112] ISO. Geographic information — Metadata — Part 3: XML schema implementation for fundamental concepts (ISO 19115-3:2016), Aug. 2016.

[113] ISO. ISO 16739-1:2018 Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries – Part 1: Data schema, 2018.

[114] ISO. ISO 16739-1:2018 Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries – Part 1: Data schema, 2018.

[115] H. G. Jonasson and S. Storeheier. Nord 2000. New Nordic Prediction Method for Road Traffic Noise, 2001.

[116] C. B. Jones, D. B. Kidner, and J. M. Ware. The implicit triangulated irregular network and multiscale spatial databases. *The Computer Journal*, 37(1):43–57, 1994.

[117] Kadaster. 3DTOP10NL. http://arcg.is/1GKYy7E, 2015. Accessed 05 May 2019.

[118] E. Kalogianni, E. Dimopoulou, W. Quak, and P. van Oosterom. LADM and INTERLIS as a perfect match for 3D cadastre. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 23–26, 2017.

[119] A. Kara, V. Çağdaş, C. Lemmen, Ü. Işikdağ, P. van Oosterom, and E. Stubkjær. Supporting fiscal aspect of land administration through a LADM-based valuation information model. In *19th Annual World Bank Conference on Land and Poverty 2018: Proceedings: Land Governance in an Interconnected World, Washington, USA.*, 2018.

[120] K. Kavisha, H. Ledoux, R. Schmidt, T. Verheij, and J. Stoter. A harmonized data model for noise simulation in the EU. *ISPRS International Journal of Geo-Information*, 9(2), 2020.

[121] S. Kephalopoulos, M. Paviotti, and F. Anfosso-Lédée. Common noise assessment methods in Europe (CNOSSOS-EU), 2012.

[122] S. Kephalopoulos, M. Paviotti, F. Anfosso-Lédée, D. Van Maercke, S. Shilton, and N. Jones. Advances in the development of common noise assessment methods in Europe: The CNOSSOS-EU framework for strategic environmental noise mapping. *Science of the Total Environment*, 482: 400–410, 2014.

[123] D. B. Kidner, J. M. Ware, A. J. Sparkes, and C. B. Jones. Multiscale terrain and topographic modelling with the implicit TIN. *Transactions in GIS*, 4(4):361–378, 2000.

[124] Y. Kim, H. Kang, and J. Lee. Developing CityGML indoor ADE to manage indoor facilities. In *Innovations in 3D Geo-information sciences*, pages 243–265. Springer, 2014.

[125] E. A. King, E. Murphy, and H. J. Rice. Implementation of the EU environmental noise directive: Lessons from the first phase of strategic noise mapping and action planning in Ireland. *Journal of environmental management*, 92(3):756–764, 2011.

[126] KIT. FZK Viewer 5.1. https://www.iai.kit.edu/1302.php, 2019. Accessed 30 April 2019 .

[127] J. Kokkonen. CNOSSOS-EU noise model implementation in Finland and experience of it in 3rd END round.

[128] J. Kokkonen, O. Kontkanen, and P. P. Maijala. CNOSSOS-EU noise model implementation in Finland. In *Baltic-Nordic Acoustic Meeting*, pages 19–22, 2016.

[129] D. Koller, B. Frischer, and G. Humphreys. Research challenges for digital archives of 3D cultural heritage models. *Journal on Computing and Cultural Heritage (JOCCH)*, 2(3):7, 2009.

[130] J. Kragh, B. Andersen, and J. Jakobsen. Environmental noise from industrial plants. general prediction method. Technical report, 1982.

[131] KTI Rt. UT 2.1-302 Road Traffic Noise Calculation, 2000.

[132] K. Kumar, H. Ledoux, and J. Stoter. Comparative analysis of data structures for storing massive TINs in a DBMS. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLI-B2:123–130, 2016.

[133] K. Kumar, H. Ledoux, and J. Stoter. A CityGML extension for handling very large TINs. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4:137, 2016.

[134] K. Kumar, H. Ledoux, and J. Stoter. A CityGML extension for handling very large TINs. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W1:137–143, 2016.

[135] K. Kumar, H. Ledoux, T. J. F. Commandeur, and J. E. Stoter. Modelling urban noise in CityGML ADE: Case of the Netherlands. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-4/W5:73–81, 2017. doi: 10.5194/isprs-annals-IV-4-W5-73-2017. URL https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/IV-4-W5/73/2017/.

[136] K. Kumar, H. Ledoux, and J. Stoter. Compactly representing massive terrain models as TINs in CityGML. *Transaction in GIS*, 22(5):1152–1178, 2018. URL https://doi.org/10.1111/tgis.12456.

[137] K. Kumar, A. Labetski, H. Ledoux, and J. Stoter. An improved LOD framework for the terrains in 3D city models. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4, 2019.

[138] K. Kumar, A. Labetski, K. A. Ohori, H. Ledoux, and J. Stoter. Harmonising the OGC standards for the built environment: A CityGML extension for LandInfra. *ISPRS International Journal of Geo-Information*, 8(6):246, 2019.

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

[140] M. Kumler. An intensive comparison of triangulated irregular networks (tins) and digital elevation models (dems). *Cartographica*, 31(2):1–99, 1994.

[141] T. Kutzner, K. Chaturvedi, and T. H. Kolbe. Citygml 3.0: New functions open up new applications. *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, pages 1–19, 2020.

[142] La Ministre de l'Environnement Luxembourg. Courrier d'information de la Commission concernant la cartographie stratégique du bruit des sites d?activité industrielle au sein de l'agglomération de la Ville de Luxembourg et environs, 2018.

[143] A. Labetski, K. Kumar, H. Ledoux, and J. Stoter. A metadata ADE for CityGML. *Open Geospatial Data, Software and Standards*, 3(16), 2018. doi: https://doi.org/10.1186/s40965-018-0057-4.

[144] A. Labetski, S. van Gerwen, G. Tamminga, H. Ledoux, and J. Stoter. A proposal for an improved transportation model in CityGML. In *13th 3D GeoInfo Conference 2018*, volume XLII-4 of *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 89–96. ISPRS, oct 2018. doi: https://doi.org/10.5194/isprs-archives-XLII-4-W10-89-2018.

[145] LandXML[dot]org. LandXML - 1.2. http://www.landxml.org/About.aspx, 2016. Accessed 30 March 2019.

[146] H. Ledoux. On the validation of solids represented with the international standards for geographic information. *Computer-Aided Civil and Infrastructure Engineering*, 28(9):693–706, 2013. ISSN 1467-8667. doi: http://dx.doi.org/10.1111/mice.12043.

[147] H. Ledoux. Storing and analysing massive tins in a dbms with a star-based data structure. Technical Report 2015.01, 3D geoinformation, Delft

University of Technology, Delft, the Netherlands., 2015. URL https://3d.bk.tudelft.nl/pdfs/15_tech_pgtin.pdf.

[148] H. Ledoux. Representation: Fields. In *The International Encyclopedia of Geography*, pages 1–15. John Wiley & Sons, Ltd, 2017.

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

[150] H. Ledoux, K. A. Ohori, K. Kumar, B. Dukai, A. Labetski, and S. Vitalis. CityJSON: A compact and easy-to-use encoding of the CityGML data model. *Open Geospatial Data, Software and Standards*, 4(1):4, 2019.

[151] C. Lemmen, P. van Oosterom, M. Kalantari, E.-M. Unger, C. H. Teo, and K. de Zeeuw. Further standardization in land administration. In *Proceedings of the 2017 World Bank Conference on Land and Poverty: Responsible Land Governance–Towards an Evidence-Based Approach, The World Bank, Washington, DC, USA*, pages 20–24, 2017.

[152] G. Licitra. *Noise mapping in the EU: models and procedures*. CRC Press, 2012.

[153] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. L. Faust, and G. Turner. Real-time, continuous level of detail rendering of height fields. Technical report, Georgia Institute of Technology, 1996.

[154] X. Liu, X. Wang, G. Wright, J. Cheng, X. Li, and R. Liu. 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, 2017.

[155] M.-O. Löwner, J. Benner, G. Gröger, and K.-H. Häfele. New concepts for structuring 3D city models - An extended level of detail concept for CityGML buildings. In *Computational Science and Its Applications–ICCSA 2013*, pages 466–480. Springer, 2013.

[156] M.-O. Löwner, G. Gröger, J. Benner, F. Biljecki, and C. Nagel. Proposal for a new lod and multi-representation concept for citygml. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4, 2016.

[157] D. Luebke, B. Watson, J. D. Cohen, M. Reddy, and A. Varshney. *Level of Detail for 3D Graphics*. Elsevier Science Inc., New York, NY, USA, 2002. ISBN 1558608389.

[158] D. P. Luebke. *Level of detail for 3D graphics*. Morgan Kaufmann, 2003.

[159] M. Luffel, T. Gurung, P. Lindstrom, and J. Rossignac. Grouper: A compact, streamable triangle mesh data structure. *Visualization and Computer Graphics, IEEE Transactions on*, 20(1):84–98, 2014.

[160] J. G. Lyon. *GIS for Water Resource and Watershed Management*. CRC Press, 2003.

[161] P. Maijala and O. Kontkanen. CNOSSOS-EU sensitivity to meteorological and to some road initial value changes. In *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, volume 253, pages 6245–6256. Institute of Noise Control Engineering, 2016.

[162] M. Malmkvist, P. Axelsson, L. Wikström, O. Bergman, A. Nilsson, S. Granberg, J. Jensen, E. Häggström, J. Sigfrid, and K. Karlsson. Alignment deployment. implementation report. Verification IFC Alignment and InfraGML. Technical report, Nordic project team, BuildingSMART, 2017.

[163] M. Mäntylä. *An Introduction to Solid Modeling*. Computer Science Press, Inc., New York, NY, USA, 1987. ISBN 0-88175-108-1.

[164] E. Maravelakis, A. Konstantaras, A. Kritsotaki, D. Angelakis, and M. Xinogalos. Analysing user needs for a unified 3D metadata recording and exploitation of cultural heritage monuments system. In *International Symposium on Visual Computing*, pages 138–147. Springer, 2013.

[165] S. McNeil, M. Tischer, and A. DeBlasio. Asset management: What is the fuss? *Transportation Research Record: Journal of the Transportation Research Board*, (1729):21–25, 2000.

[166] Minister van Volkshuisvesting, Ruimtelijke Ordening en Milieubeheer. Handleiding meten en rekenen Industrielawaai, 1991.

[167] U. Moehler, U. J. Kurze, M. Liepert, and H. Onnich. The new german prediction model for railway noise "schall 03 2006": an alternative method for the harmonised calculation method proposed in the eu directive on environmental noise. *Acta Acustica united with Acustica*, 94(4):548–552, 2008.

[168] M. Moshrefzadeh, A. Donaubauer, and T. H. Kolbe. A citygml-based façade information model for computer aided facility management. In *Bridging Scales-Skalenübergreifende Nah-und Fernerkundungsmethoden, 35. Wissenschaftlich-Technische Jahrestagung der DGPF*, 2015.

[169] A. Motamedi, M. M. Soltani, S. Setayeshgar, and A. Hammad. Extending ifc to incorporate information of rfid tags attached to building elements. *Advanced Engineering Informatics*, 30(1):39–53, 2016.

[170] MSZT/MB 327. MSZ 15036:2002 - Outdoor sound propagation, 2002.

[171] MSZT/MCS 302. MSZ-07-2904:1990 - Method of calculation of traffic noise. Noise caused by railway traffic, 1990.

[172] D. E. Muller and F. P. Preparata. Finding the intersection of two convex polyhedra. *Theoretical Computer Science*, 7:217–236, 1978.

[173] E. Murphy and E. A. King. Strategic environmental noise mapping: Methodological issues concerning the implementation of the EU Environmental Noise Directive and their policy implications. *Environment international*, 36(3):290–298, 2010.

[174] C. Nagel. citygml4j: the open Source Java API for CityGML. https://github.com/citygml4j/citygml4j, 2015. Accessed 30 April 2019 .

[175] C. Nagel, A. Stadler, and T. Kolbe. Conversion of IFC to CityGML. In *Meeting of the OGC 3DIM Working Group at OGC TC/PC Meeting, Paris (Frankreich)*, 2007.

[176] S. Neubauer and A. Zipf. Suggestions for extending the OGC styled layer descriptor (SLD) specification into 3D–towards visualization rules for 3D city models. *Applied Sciences*, page 133, 2007.

[177] H. L. Nielsen. *Road traffic noise: Nordic prediction method.* The Nordic Council of Ministers, 1997.

[178] M. G. Niestroj, D. A. McMeekin, and P. Helmholz. Overview of standards towards road asset information exchange. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42(4), 2018.

[179] M. G. Niestroj, D. A. McMeekin, P. Helmholz, and M. Kuhn. A proposal to use semantic web technologies for improved road network information exchange. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4(4), 2018.

[180] H. Nijland and G. Van Wee. Traffic noise in Europe: A comparison of calculation methods, noise indices and noise standards for road and railroad traffic in Europe. *Transport Reviews*, 25(5):591–612, 2005.

[181] R. Nouvel, J.-M. Bahu, R. Kaden, J. Kaempf, P. Cipriano, M. Lauster, and E. Casper. Development of the CityGML Application Domain Extension energy for urban energy simulation. *Proceedings of Building Simulation 2015*, 2015.

[182] NVIDIA. NvTriStrip Library. Retrieved from http://www.nvidia.com/object/nvtristrip_library.html, 2004.

[183] OGC. LandGML interoperability experiment. http://www.opengeospatial.org/projects/initiatives/landgmlie, 2004. Accessed 30 March 2019.

[184] OGC. OGCÂő Geography Markup Language (GML) Implementation Specifications Version 3.1.1 Doc. No. 03-105r1. http://www.opengis.net/spec/GML/3.1.1, 2004.

[185] OGC. OGC City Geography Markup Language (CityGML) Encoding Standard 2.0.0. Doc. No. 12-019. https://portal.opengeospatial.org/files/?artifact_id=47842, 2012.

[186] OGC. OGC® Geography Markup Language (GML) — Extended schemas and encoding rules Version 3.3 Document No. 10-129r1., 2012.

[187] OGC. OGC® SensorML: Model and XML Encoding Standard. Doc No. OGC 12-000, 2014.

[188] OGC. *OGC®Land and Infrastructure Conceptual Model Standard. Doc. No. 15-111r1*. OGC, 2016.

[189] OGC. *PipelineML Conceptual and Encoding Model Standard*. OGC, 2016.

[190] OGC. *OGC Underground Infrastructure Concept Study Engineering Report. Document No. OGC 17-048*. OGC, 2017.

[191] OGC. OGC InfraGML 1.0: Part 0 – LandInfra Core - Encoding Standard. Document No. 16-100r2, 2017.

[192] OGC. OGC InfraGML 1.0: Part 1 – LandInfra Land Features - Encoding Standard. Document No.16-101r2 , 2017.

[193] OGC. OGC InfraGML 1.0: Part 2 – LandInfra Facilities and Projects - Encoding Standard. Document No. 16-102r2, 2017.

[194] OGC. OGC InfraGML 1.0: Part 3 – Alignments - Encoding Standard. Document No. 16-103r2, 2017.

[195] OGC. OGC InfraGML 1.0: Part 4 – LandInfra Roads - Encoding Standard. Document No. 16-104r2, 2017.

[196] OGC. OGC InfraGML 1.0: Part 5 – Railways - Encoding Standard. Document No. 16-105r2, 2017.

[197] OGC. OGC InfraGML 1.0: Part 6 – LandInfra Survey - Encoding Standard. Document No. 16-106r2, 2017.

[198] OGC. OGC InfraGML 1.0: Part 7 – LandInfra Land Division - Encoding Standard. Document No. 16-107r2, 2017.

[199] OGC. OGC White Paper on Land Administration. https://docs.opengeospatial.org/wp/18-008r1/18-008r1.html, 2019. Accessed 02 April 2019.

[200] OGC. *OGC®3D Tiles Specification 1.0 Doc. No. 18-053r2*. OGC, 2019.

[201] OGC. *OGC®Indexed 3d Scene Layer (I3S) and Scene Layer Package Format Specification Doc. No. 17-014r7*. OGC, 2020.

[202] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*, volume 501. John Wiley & Sons, 2009.

[203] H. Olfat, S. Kalantari Soltanieh, A. Rajabifard, H. Senot, and I. P. Williamson. Spatial metadata automation: A key to spatially enabling platform. *International Journal of Spatial Data Infrastructures Research*, 7, 2012.

[204] Open Geospatial Consortium. Opengis® implementation specification for geographic information-simple feature access-part 1: common architecture. *Document reference 06-103r4*, 2011.

[205] Open Geospatial Consortium. Modeling an application domain extension of CityGML in UML (OGC Best Practice). *Document reference 12-066*, 2014.

[206] Oracle. Oracle spatial developer's guide –TIN-related object types. https://docs.oracle.com/database/121/SPATL/tin-related-object-types.htm, 2019. Accessed 25 March 2020.

[207] F. Penninga. *3D topography: a simplicial complex-based solution in a spatial DBMS*. PhD thesis, TU Delft, Delft University of Technology, 2008.

[208] B. Plovsing and C. Svane. DANSIM- Danish Airport Noise Simulation Model: basic principles, experience, and improvements. *Inter-noise 90*, pages 425–428, 1990.

[209] PostGIS. PostGIS LWTIN and LWTRIANGLE. https://docs.oracle.com/cd/B28359_01/appdev.111/b28400/sdo_objrelschema.htm#SPATL503, 2019. Accessed 07 Aug 2019.

[210] J. Pouliot, S. Larrivée, C. Ellul, and A. Boudhaim. Exploring schema matching to compare geospatial standards: application to underground utility networks. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42, 2018.

[211] M. Pronk. Storing massive tins in a dbms: A comparison and a prototype implementation of the multistar approach, 2015.

[212] PyFFI. Package PyFFI: Class TriangleStripifier. Retrieved from http://pyffi.sourceforge.net/apidocs/, 2011.

[213] A. Rajabifard, B. Atazadeh, and M. Kalantari. A critical evaluation of 3D spatial information models for managing legal arrangements of multi-owned developments in Victoria, Australia. *International Journal of Geographical Information Science*, 32(10):2098–2122, 2018.

[214] S. Ravada, B. M. Kazar, and R. Kothuri. Query processing in 3D spatial databases: Experiences with oracle spatial 11g. In *3D Geo-Information Sciences*, pages 153–173. Springer, 2009.

[215] Rijkswaterstraat, Ministerie van Infrastructuur en Milieu. Gebruikershandleiding geluidregister hoofdwegennet, 2014.

[216] J. Riley. Understanding Metadata - What is metadata, and what is it for? National Information Standards Organization (NISO) Primer Series, 2017.

[217] M. Ringheim and H. L. Nielsen. *Railway Traffic Noise – The Nordic Prediction Method*. Number 524. The Nordic Council of Ministers, 1997.

[218] S. Rippa. Minimal roughness property of the Delaunay triangulation. *Computer Aided Geometric Design*, 7(6):489–497, 1990.

[219] C. Rönsdorff, D. Wilson, and J. Stoter. Integration of land administration domain model with CityGML for 3D Cadastre. In *Proceedings 4th International Workshop on 3D Cadastres, 9-11 November 2014, Dubai, United Arab Emirates*. International Federation of Surveyors (FIG), 2014.

[220] SAE International. Air 1845 - Procedure for the calculation of airplane noise in the vicinity of airports, 2012.

[221] Safe Software. FME 2019.0. https://www.safe.com, 2019. Accessed 15 May 2019 .

[222] N. Salheb. *Automatic Conversion of CityGML to IFC*. PhD thesis, Delft University of Technology, 2019.

[223] P. Scarponcini. InfraGML proposal (13-121). https://portal.opengeospatial.org/files/?artifact_id=56299, 2013. Accessed 30 March 2019.

[224] P. Scarponcini. OGC as-is LandXML-1.2 conceptual model. Preliminary draft. Version 0.3. https://portal.opengeospatial.org/files/?artifact_id=54380, 2013. Accessed 30 March 2019.

[225] P. Scarponcini. OGC LandInfra / InfraGML standards for infrastructure. https://www.opengeospatial.org/projects/initiatives/undergroundcds, 2017. OGC Underground Infrastructure Mapping and Modeling Workshop 2017. Accessed 28 Match 2019.

[226] J. R. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. *Applied Computational Geometry: Towards Geometric Engineering*, 1148:203–222, 1996. ISSN 00983004. doi: 10.1007/BFb0014497. URL http://www.cs.cmu.edu/{~}quake/triangle.html.

[227] S. J. Shilton, F. Anfosso Lédée, and H. Van Leeuwen. Conversion of existing road source data to use CNOSSOS-EU. In *EuroNoise 2015 - Conference Proceedings*, 2015.

[228] J. Snoeyink and B. Speckmann. Tripod: a minimalist data structure for embedded triangulations. *Computational Graph Theory and Combinatorics*, 1999.

[229] Y. Song, X. Wang, Y. Tan, P. Wu, M. Sutrisna, J. Cheng, and K. Hampson. Trends and opportunities of bim-gis integration in the architecture, engineering and construction industry: a review from a spatio-temporal statistical perspective. *ISPRS International Journal of Geo-Information*, 6(12):397, 2017.

[230] B. Speckmann and J. Snoeyink. Easy triangle strips for TIN terrain models. *International journal of geographical information science*, 15(4):379–386, 2001.

[231] A. Stadler and T. H. Kolbe. Spatio-semantic coherence in the integration of 3D city models. In *Proceedings of the 5th International Symposium on Spatial Data Quality, Enschede*, 2007.

[232] J. Stoter, G. Vosselman, J. Goos, S. Zlatanova, E. Verbree, R. Klooster, and M. Reuvers. Towards a national 3D spatial data infrastructure: case of the Netherlands. *Photogrammetrie-Fernerkundung-Geoinformation*, 2011 (6):405–420, 2011.

[233] J. Stoter, H. Ploeger, R. Roes, E. van der Riet, F. Biljecki, and H. Ledoux. First 3D Cadastral Registration of Multi-level Ownerships Rights in the Netherlands. In *5th International FIG Workshop on 3D Cadastres*, pages 491–504, Athens, Greece, 2016.

[234] J. Stoter, R. Peters, T. Commandeur, B. Dukai, K. Kumar, and H. Ledoux. Automated reconstruction of 3d input data for noise simulation. *Computers, Environment and Urban Systems*, 80:101424, 2020.

[235] E. Stubkjær, J. M. Paasch, V. Cagdas, P. v. Oosterom, S. Simmons, J. Paulsson, and C. Lemmen. International code list management–the case of land administration. In *The 7th Land Administration Domain Model Workshop*, page 21. FIG-International Federation of Surveyors, 2018.

[236] L. Tang, L. Li, S. Ying, and Y. Lei. A full level-of-detail specification for 3D building models combining indoor and outdoor scenes. *ISPRS International Journal of Geo-Information*, 7(11):419, 2018.

[237] G. Tavares, Z. Zsigraiova, V. Semiao, and M. d. G. Carvalho. Optimisation of MSW collection routes for minimum fuel consumption using 3D GIS modelling. *Waste Management*, 29(3):1176–1185, 2009.

[238] W. Tegtmeier, S. Zlatanova, P. van Oosterom, and H. Hack. 3D-GEM: Geo-technical extension towards an integrated 3D information model for infrastructural development. *Computers & Geosciences*, 64:126–135, 2014.

[239] R. Tse and C. Gold. TIN meets CAD-extending the TIN concept in GIS. *Future Generation Computer Systems*, 20(7):1171–1184, 2004.

[240] TU Delft. 3d input data for noise studies (experimental). https://3d.bk.tudelft.nl/opendata/noise3d/en.html, 2019.

[241] TU Delft. Cities/regions around the world with open datasets. https://3d.bk.tudelft.nl/opendata/opencities/, 2019.

[242] UIC. RailTopoModel. http://www.railtopomodel.org, 2016. Accessed 28 March 2019.

[243] P. van Oosterom. Spatial access methods. *Geographical information systems*, 1:385–400, 1999.

[244] P. van Oosterom, C. Lemmen, R. Thompson, K. Janečka, S. Zlatanova, and M. Kalantari. Cadastral Information Modelling. In P. van Oosterom, editor, *Best Practices 3D Cadastres: extended version*, FIG publication, pages 95–132. Copenhagen: International Federation of Surveyors (FIG), 2018. ISBN 978-87-92853-64-6.

[245] T. Vergoed and H. J. van Leeuwen. Evaluation and validation of the CNOSSOS calculation method in the netherlands. In *EuroNoise 2018 - Conference Proceedings*, 2018.

[246] VTP. ITF Format - Virtual Terrain Project. http://vterrain.org/Implementation/Formats/ITF.html, 2012.

[247] Y. Wan and F. Bian. A Extended Web Feature Service Based Web 3D GIS Architecture. In *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, pages 5947–5950. IEEE, 2007.

[248] P. Wate, S. Srivastav, S. Saran, and Y. K. Murthy. Formulation of hierarchical framework for 3d-gis data acquisition techniques in context of level-of-detail (lod). In *2013 IEEE Second International Conference on Image Information Processing (ICIIP-2013)*, pages 154–159. IEEE, 2013.

[249] M. Weise, T. Liebich, and J. Wix. Integrating use case definitions for ifc developments. *eWork and eBusiness in Architecture and Construction. London: Taylor & Francis Group*, pages 637–645, 2009.

[250] Wölfel Group. IMMI: Precise and efficient software for noise prediction and dispersion of air pollutants, 2018.

[251] M. F. Worboys and M. Duckham. *GIS: a computing perspective*. CRC press, 2004.

[252] T. Wszołek, B. Stępień, and D. Mleczko. Comparison of iso 9613-2 and cnossos-eu methods in noise modelling of a large industrial plant. In *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, volume 259, pages 7807–7810. Institute of Noise Control Engineering, 2019.

[253] J. C. Xia, J. El-Sana, and A. Varshney. Adaptive real-time level-of-detail based rendering for polygonal models. *IEEE Transactions on Visualization and Computer graphics*, 3(2):171–183, 1997.

[254] Z. Yao and T. H. Kolbe. Dynamically extending spatial databases to support CityGML application domain extensions using graph transformations. In *Kulturelles Erbe erfassen und bewahren-Von der Dokumentation zum virtuellen Rundgang, 37. Wissenschaftlich-Technische Jahrestagung der DGPF*, pages 316–331, 2017.

[255] Z. Yao, C. Nagel, F. Kunde, G. Hudra, P. Willkomm, A. Donaubauer, T. Adolphi, and T. H. Kolbe. 3DCityDB — a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML. *Open Geospatial Data, Software and Standards*, 3(2), 2018.

[256] A. Zamyadi, J. Pouliot, and Y. Bédard. Towards Precise Metadata-set for Discovering 3D Geospatial Models in Geo-portals. In *8th 3D GeoInfo Conference*, 2013.

[257] M. Zhiliang, W. Zhenhua, S. Wu, and L. Zhe. Application and extension of the ifc standard in construction cost estimating for tendering in china. *Automation in Construction*, 20(2):196–204, 2011.

[258] S. Zlatanova, J. Stoter, and U. Isikdag. Standards for exchange and storage of 3D information: Challenges and opportunities for emergency response. In *Proceedings of the 4th International Conference on Cartography & GIS, Volume 2, Albena, June 2012, pp. 17-28*. International Cartographic Association, 2012.

[259] F. Zobl, K. Chmelina, R. Faber, J. Kooijman, R. Marschallinger, and J. Stoter. Multidimensional aspects of GeoBIM data: new standards

needed. In *Mathematical Geosciences at the Crossroads of Theory and Practice, Proceedings of the IAMG2011 conference*, pages 1–14, 2011.

Summary

A 3D city model is a digital representation of the spatial features in an urban environment. Buildings, terrain, vegetation, water bodies, etc. all form an integral part of a 3D city model. The possibility to enrich these city models with additional application-specific information, whether new semantics or geometry, further increases their usability. However, in practice, the applications of 3D city models are mainly focused on buildings. The majority of standards available for representing 3D city models, such as IFC and CityGML, have well-defined specifications for modelling buildings, but often none for other city features. In addition, there are several other issues associated with the development and use of 3D city models of large cities, such as massive size of 3D city models, interoperability issues for 3D data from heterogeneous sources, harmonisation of different 3D standards, etc.

In this thesis, I investigate how to better model these massive and semantically enriched 3D city models, and I focus on their use in different applications. I make five contributions. First, I explain how CityGML, the international standard for semantic 3D city modelling, is not efficient for storing massive TIN terrains, and present an improved solution to compactly store massive terrains in CityGML. Second, I describe how to model terrains at different LODs in CityGML, since the current CityGML data model lacks the specifications for modelling different terrain LODs at geometric and semantic level. Third, I explain how CityGML lacks precise specifications for modelling metadata of 3D city models and present an ISO 19115 compliant solution to add metadata. Fourth, I describe in this thesis how the development of the new standards LandInfra and InfraGML and their integration with the existing popular standards (IFC and CityGML) can contribute to the BIM and 3D GIS interoperability and bring the two domains to a common footing. Fifth, I demonstrate my approach for the development of a harmonised semantic 3D city model based on CityGML for use in urban noise simulations. In addition, I have developed open source prototypes to help practitioners with the use of 3D city models. In this way, I also contribute to the open source community for 3D city modelling.

The thesis proposes additional research for future work. For example, since this research focuses specifically on the LODs of terrain models, it would be

worthwhile to extend the research to explore the LOD concept for other urban features such as vegetation and landuse. Furthermore, LandInfra is a relatively young standard with low community support. This too requires more attention. Tools such as parsers, validators, visualisers, DBMS support, APIs, and so on are still lacking for LandInfra (and InfraGML). It would be interesting to see how the standards evolve and whether it can be applied in practice when such support is available. Interoperability of LandInfra with IFC and other standards is also an area that requires further investigation.

Samenvatting

Een 3D-stadsmodel is een digitale representatie van de ruimtelijke kenmerken in een stedelijke omgeving. Gebouwen, terrein, vegetatie, landgebruik, enz. zijn allemaal integrale onderdelen van een 3D-stadsmodel. De mogelijkheid om deze stadsmodellen te verrijken met toepassingsspecifieke informatie, of het nu gaat om aanvullende semantiek of geometrie, vergroot de bruikbaarheid van een 3D-stadsmodel nog verder. In de praktijk zijn de toepassingen van 3D-stadsmodellen echter vooral gefocussed op gebouwen. Zo hebben de meeste beschikbare standaarden voor het weergeven van 3D-stadsmodellen, zoals IFC en CityGML, duidelijk gedefinieerde specificaties voor het modelleren van gebouwen. Maar voor andere type objecten ontbreken uitgewerkte specificaties. Daarnaast zijn er andere problemen die verband houden met het genereren en het gebruik van 3D-stadsmodellen, zoals de enorme data-volumes om 3D-stadsmodellen uit te wisselen en te gebruiken, interoperabiliteitsproblemen voor 3D-gegevens uit heterogene bronnen, harmonisatie van verschillende 3D-standaarden, enz.

In dit proefschrift onderzoek ik hoe deze massieve en semantisch verrijkte 3D-stadsmodellen beter kunnen worden gemodelleerd voor gebruik in verschillende toepassingen. Ik lever vijf bijdragen. Eerst leg ik uit hoe CityGML, de populaire standaard voor semantische 3D-stadsmodellering, niet efficiënt is voor het opslaan van TIN-terreinen, die vaak heel veel data-volume omvatten, en stel ik een verbeterde oplossing voor om terreinmodellen compact op te slaan in CityGML. Ten tweede beschrijf ik hoe je terreinen op verschillende LODs in CityGML kunt modelleren. Het huidige CityGML-datamodel mist de specificaties voor het modelleren van verschillende terrein-LODs zowel geometrisch als semantisch. Ten derde leg ik uit hoe CityGML nauwkeurige specificaties mist voor het modelleren van metadata van 3D-stadsmodellen en presenteren we een ISO 19115-compatibele oplossing om metadata toe te voegen aan CityGML-modellen. Ten vierde beschrijf ik in dit proefschrift hoe integratie van nieuwe standaarden, zoals LandInfra en InfraGML, met de bestaande populaire BIM en Geo standaarden (IFC en CityGML) kan bijdragen aan de BIM en 3D GIS-interoperabiliteit door deze twee domeinen op een gemeenschappelijke basis te brengen. Tenslotte presenteer ik mijn voorstel voor een geharmoniseerd semantisch 3D-stadsmodel op basis van CityGML voor gebruik in stedelijke geluidssimulaties. Voor al deze onderwerpen heb ik open source prototypes

ontwikkeld om professionals in de praktijk te helpen bij het gebruik van 3D stadsmodellen. Op deze manier draag ik ook bij aan de open source gemeenschap voor 3D stadsmodellering.

Het proefschrift stelt aan het einde nader onderzoek voor. Omdat mijn onderzoek zich bijvoorbeeld specifiek richt op de LODs van terreinmodellen, zou het de moeite waard zijn om het onderzoek uit te breiden naar het LOD-concept voor andere stedelijke kenmerken zoals vegetatie en landgebruik. Bovendien is LandInfra een relatief jonge standaard met nog weinig praktijk ondersteuning. Ook dit zou meer aandacht behoeven. Daarnaast ontbreken er nog tools zoals parsers, validators, visualisers, DBMS-ondersteuning, API's, enzovoort om goed om te gaan met LandInfra (and InfraGML). Het zou interessant zijn om te zien hoe de standaarden evolueren en in de praktijk kunnen worden toegepast als dergelijke ondersteuning beschikbaar is. Interoperabiliteit van LandInfra met IFC en andere standaarden is ook een gebied dat nader onderzoek behoeft.

Glossary of terms

3D4EM 3D for Environmental Modelling

3DBGT 3D Basisregistratie Grootschalige Topografie

3DCityDB 3D City Database

ADE Application Domain Extension

AHN3 Actueel Hoogtebestand Nederland version 3

BIM Building Information Modelling

B-rep Boundary Representation

CAFM Computer Aided Facility Management

CDR Central Data Repository

CDS Concept Development Study

CDT Constrained Delaunay Triangulation

CityGML City Geographic Markup Language

CNOSSOS Common framework for NOise aSSessment methOdS

COLLADA COLLAborative Design Activity

CRTN Calculation of Road Traffic Noise

CSG Constructive Solid Geometry

DBMS Database Management Systems

DT Delaunay Triangulation

DTM Digital Terrain Model

EEA European Environment Agency

END Environmental Noise Directive

EU European Union

gbXML Green Building XML

GIS Geographical information Systems

glTF GL Transmission Format

GML3 Geography Markup Language version 3

I3S Indexed 3D Scene Layers

IFC Industry Foundation Classes

IGN Institut GeÌĄographique National

ISO International Standards Organisation

JSON JavaScript Object Notation

LADM Land Administration Domain Model

LandInfra Land and Infrastructure

LOD Level Of Detail

NMPB Nouvelle Méthode du Prévision de Bruit

NWO Netherlands Organisation for Scientific Research

OGC Open Geospatial Consortium

RDBMS Relational Database Management System

RMR Reken- en Meetvoorschrift Railverkeerslawaai

RMW Reken en meetvoorschrift Verkeerslawaai

SDI Spatial Data Infrastructure

SensorML Sensor Model Language

SF Simple Feature

SW Semantic Web

SWG Standards Working Group

TIC Terrain Intersection Curve

TIN Triangulated Irregular Network

UML Unified Modelling Language

WKB Well Known Binary

XSD XML Schema Definition

Curriculum Vitæ

Kavisha (1992) was born in India. She obtained her B.Tech. degree in Information Technology in 2013 from Banasthali University, Rajasthan, and an M.Tech. degree in RS & GIS with specialization in Geoinformatics from IIRS, Dehradun in 2015.

From 2015 to 2019, Kavisha conducted her PhD research on *Modelling and managing massive 3D data of the built environment*, funded by the Netherlands Organisation for Scientific Research (*NWO*) and supervised by Hugo Ledoux and Jantien Stoter at TU Delft.

Journal Publications

A Harmonized Data Model for Noise Simulation in the EU. Kavisha Kumar, Hugo Ledoux, Richard Schmidt, Theo Verheij, and Jantien Stoter. *ISPRS International Journal of Geo-Information* 9(2), 2020, p.121, doi: https://doi.org/10.3390/ijgi9020121

The LandInfra standard and its role in solving the BIM-GIS quagmire. Kavisha Kumar, Anna Labetski, Ken Arroyo Ohori, Hugo Ledoux, and Jantien Stoter. *Open Geospatial Data, Software and Standards* 4(1), 2019, p.1 , doi: https://doi.org/10.1186/s40965-019-0065-z

Harmonising the OGC Standards for the Built Environment: A CityGML Extension for LandInfra. Kavisha Kumar, Anna Labetski, Ken Arroyo Ohori, Hugo Ledoux, and Jantien Stoter. *ISPRS International Journal of Geo-Information* 8(6), 2019, p.246, doi: https://doi.org/10.3390/ijgi8060246

CityJSON: a compact and easy-to-use encoding of the CityGML data model. Hugo Ledoux, Ken Arroyo Ohori, Kavisha Kumar, Balázs Dukai, Anna Labetski and Stelios Vitalis. *Open Geospatial Data, Software and Standards* 4(4), 2019, doi: https://doi.org/10.1186/s40965-019-0064-0

A metadata ADE for CityGML. Anna Labetski, Kavisha Kumar, Hugo Ledoux, and Jantien Stoter. *Open Geospatial Data, Software and Standards* 3(1), 2019, p.16, doi: https://doi.org/10.1186/s40965-018-0057-4

Compact representation of massive terrains represented as TINs in 3D city models. Kavisha Kumar, Hugo Ledoux, and Jantien Stoter. *Transaction in GIS* 22(5), 2018, p.1152, doi: https://doi.org/10.1111/tgis.12456

CityGML Application Domain Extension (ADE): overview of developments. Filip Biljecki, Kavisha Kumar, and Claus Nagel. *Open Geospatial Data, Software and Standards* 3(13), 2018, doi: https://doi.org/10.1186/s40965-018-0055-6

Conference Publications

An improved LOD framework for the terrains in 3D city models. Kavisha Kumar, Anna Labetski, Hugo Ledoux, and Jantien Stoter. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* IV-4/W8, 2019, p.75, doi: https://doi.org/10.5194/isprs-annals-IV-4-W8-75-2019

A survey on the adoption of GIS data and standards in urban application domains. Kavisha Kumar, Anna Labetski, Giorgio Agugiaro, and Jantien Stoter. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* XLII-4/W15, 2019, p.41, doi: https://doi.org/10.5194/isprs-archives-XLII-4-W15-41-2019

Dynamic 3D visualization of floods: Case of the Netherlands. Kavisha Kumar, Hugo Ledoux, and Jantien Stoter. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* XLII-4/W10, 2018, p.83, doi: https://doi.org/10.5194/isprs-archives-XLII-4-W10-83-2018

Modelling urban noise in CityGML ADE: Case of the Netherlands. Kavisha Kumar, Tom Commandeur, Hugo Ledoux, and Jantien Stoter. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* IV-4/W5, 2017, p.73, doi: https://doi.org/10.5194/isprs-annals-IV-4-W5-73-2017

A CityGML extension for handling very large TINS. Kavisha Kumar, Hugo Ledoux, and Jantien Stoter. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* IV-2/W1, 2016, p.137, doi: https://doi.org/10.5194/isprs-annals-IV-2-W1-137-2016

Comparative analysis of data structures for storing massive TINs in a DBMS. Kavisha Kumar, Hugo Ledoux, and Jantien Stoter. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* XLI-B2, 2016, p.123, doi: https://doi.org/10.5194/isprs-archives-XLI-B2-123-2016

Book Chapters

Modeling Cities and Landscapes in 3D with CityGML. Ken Arroyo Ohori, Filip Biljecki, Kavisha Kumar, and Jantien Stoter. *In: Borrmann A., König M., Koch C., Beetz J. (eds) Building Information Modeling* Springer, Cham, 2018, doi: https://doi.org/10.1007/978-3-319-92862-3_11


9 789463 663168