MODELLING DIFFERENT LEVELS OF DETAIL OF ROADS AND INTERSECTIONS IN 3D CITY MODELS

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of

Master of Science in Geomatics for the Built Environment

by

Freek Boersma

July 2019

Freek Boersma: *Modelling different levels of detail of roads and intersections in 3D city models* (2019)

ISBN 978–94–6384–057–6

The work in this thesis was made in the:

3D geoinformation group
Department of Urbanism
Faculty of Architecture & the Built Environment
Delft University of Technology

Supervisors:	Anna Labetski, MSc
	Prof. Dr. Jantien Stoter
Co-reader:	Dr. Ir. Pirouz Nourian

ABSTRACT

In the last two decades there has been a steady rise in the gathering and use of 3D geo-information. A common way to store and use 3D data is by using 3D city models. In 3D city models, geo-information can be stored at different levels of detail. CityGML, the most commonly used data model and encoding for 3D city models, uses five levels of detail in order to model increasing geometric and semantic complexity. These levels of detail may be interpreted as a model quality measure, and as a guideline for users that need data for a certain application.

CityGML consists of several thematic modules, each with their own level of detail specification. Some of these modules have a more further developed level of detail specification than others. Recently, several authors have proposed improvements to the Transportation module. This has led to proposals for various changes, especially concerning road data. However, these proposed changes have not been encoded yet. The main critiques are the lack of a level of detail specification for linear representations of roads, no ability to model networks, no representation of intersections and general ambiguity in the level of detail specification. Many road data use cases might potentially benefit from improvements on these points.

In this thesis I attempt to improve the current level of detail specification of roads in the CityGML data model. The improvements are encoded in CityJSON, a JSON encoding of the this data model. I assess the shortcomings in the current CityGML transportation module. After, a road data needs analysis is performed on three use cases: transport modelling, navigation and road maintenance. The data needs are compared to modelling approaches of other road data standards. This has resulted in several encoded improvements. A topological structure has been added to CityJSON. This includes the addition of two new modelling classes: Nodes and Edges. This structure is general such that it can be used by other thematic modules as well. Moreover, the level of detail specification for roads has been further developed to include both the linear representations and less ambiguous areal representations. This includes a prescription on how to model intersections and roundabouts at different levels of detail. Finally, I provide a structure which enables one to link the linear and areal road data together. This link is made at the scale level of the object, which data providers may choose themselves. This way one object can be modelled in two representation types without needing a one-to-one mapping between linear road segments and areal triangulated surfaces. These concepts are then tested by creating a CityJSON road data file for all new levels of detail.

ACKNOWLEDGEMENTS

Writing a thesis is a satisfying but lonely affair. Nevertheless, I could not have done it without many others, who have contributed to my thesis in ways big or small. I would like to thank them all here. First and foremost, I would like to thank my supervisors, Anna Labetski and Jantien Stoter. I have struggled at times with the direction of my thesis, and you have helped me get back on track every time. Anna, thank you for taking so much time to guide me through the process. Your comments were always on point. I have also enjoyed your general enthusiasm for the subject. Jantien, you often approached the subject from the viewpoint of the user, which has been very helpful for me. Thank you for your input and always giving me new ideas to think about.

I would also like to thank my co-reader Pirouz Nourian for the valuable comments on the use of graphs. Also, you reminded me to keep an eye on the conceptual side of the thesis. When you work on a project for almost a year, it is sometimes hard to zoom out. Thank you for your perspectives!

Throughout the research I have done fieldwork by meeting with experts of various domains. All of them I would like to thank for their time and expertise: Stefan van Gerwen from Provincie Noord-Brabant, for giving me access to provincial data to use, and teaching me the ins and outs of Dutch governmental data models. Guus Tamminga from Sweco and TU Delft, for our discussion on how to geometrically model roads for traffic models. Hugo Ledoux from the 3D geoinformation group here at TU Delft, for discussing with me about CityJSON and my proposals for adaptation. Lastly, Dick Krijtenburg from Geonovum and Sandra Leijten from VNG for inviting me to the sessions about central object registration. These have been very informative for me personally, and have guided my research.

I must not forget to mention my fellow student Davey Oldenburg, who has helped me put together the front cover for this thesis. I really should take an InDesign course, or Illustrator, or Photoshop...

Besides Davey, I would like to thank all my fellow Geomatics students. Although we did not see each other that much the final year, I have very fond memories of us all working together in the Geolab, going to the Bouwpub, and generally just having fun. Thank you all so much for making me feel at home in Delft. Especially to my roommates I would like to say: ευχαριστω for being there. I've truly had a great time, I will miss it.

As always, a big thanks to my family. My parents, who have shown unconditional love and support throughout my life. Thank you for having me whenever I wanted to escape my self-imposed thesis exile. My brother Keje, you have always looked out for me, thank you for encouraging me to get out of the house at times to do something fun.

And finally, Iris, I want to thank you for being there for me always, and for your love and support, when I was struggling with writing my thesis, and at all other times. What might have been a difficult year for me, has turned out to be a great one.

Freek Boersma, July 2019

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

COR	Central Object Registration	1
LoD	level of detail	1
LoDs	levels of detail	1
OGC	Open Geospatial Consortium	1
XML	Extensible Markup Language	2
JSON	JavaScript Object Notation	2
NMAs	National Mapping Agencies	7
LRM	linear referencing method	8
LRS	linear referencing system	8
UML	Unified Modelling Language	11
XSD	XML schema files	11
ADE	Application Domain Extension	20
ADEs	Application Domain Extensions	11
GDF	Geographic Data Files	16
ISO	International Organization for Standardization	16
ITS	intelligent transportation systems	16
TIN	triangular irregular network	17
OSM	OpenStreetMap	17
ITF	Intersection Topology Format	19
MAP	MapData	19
BGT	Basisregistratie Grootschalige Topografie	20
IMGeo	Informatiemodel Geografie	20
NWB	Nationaal Wegenbestand	20
BAG	Basisregistratie Adressen en Gebouwen	20
BRep	boundary representation	31
cjio	CityJSON In/Out	33
GIS	geographical information system	39

1 INTRODUCTION

1.1 MOTIVATION

In the last decades there has been a shift towards 3D geo-information [Oude Elberink, 2010; Stoter et al., 2015]. Recently, in the Netherlands a projct has started to create a nation-wide 3D standard [Stoter et al., 2013]. Also, the Dutch organisation for cadastre and national mapping is working towards capturing objects in three dimensions [Kadster, 2019]. The Dutch government is considered a front runner in capturing and sharing 3D data. When talking to government officials and geo-data users, many express a desire to either create or use 3D geo-information. At this moment the Dutch government stores geo-data in several key registers. A pilot has started to initiate a move towards one overarching object-oriented register, where each object is modelled only once [Werkgroep Wegen, 2018]. The idea is to combine the step towards 3D data with this Central Object Registration (COR). Through this process, it is important to reconsider how we model objects.

3D geo-information can be stored using 3D city models. In these models, real world objects can be modelled at various level of detail (LoD). Models may have a lower LoD because of the way the data was acquired, but also because many applications do not require the most detailed model possible [Biljecki et al., 2013]. The different levels of detail (LoDs) are needed because they can be a measure of model quality, and act as a guideline for which applications can use the model [Biljecki et al., 2013]. Ultimately, 3D city models are only useful when the data can be used for all kinds of applications. From reviewing other standards, it is clear that many were developed with a use case in mind. Thus a use case based development of LoDs might lead to better applications of 3D city model data.

CityGML is arguably the most popular standard used to store and represent 3D city models at various LoDs [Biljecki et al., 2013]. It is an open standard which is maintained by the Open Geospatial Consortium (OGC). CityGML uses five LoDs (from 0 to 4) to denote objects or models with increasing geometric and semantic complexity, see Figure 1.1. Over the years since its inception, changes have been proposed to the data model of CityGML. Most of these changes concern adaptations to the building module [Löwner and Gröger, 2016]. Buildings have the most elaborate data model, especially concerning the differentiation between the different LoDs. Also, most use cases identified for 3D city models seem to use mostly building data (see for example Biljecki et al. [2015]). Therefore, it is perhaps not surprising that the debate has been mostly about buildings.

CityGML consists of a core module describing general properties that are applicable to all objects, along with different thematic modules for each separate thematic modelling class. One of these is a thematic module for transportation features. Transportation objects can be modelled as `tracks`, `roads`, `squares` and `railways`. These can be modelled using both a linear representation or an areal representation, using surfaces to model road segments [Open Geospatial Consortium, 2012b]. The LoD specification for

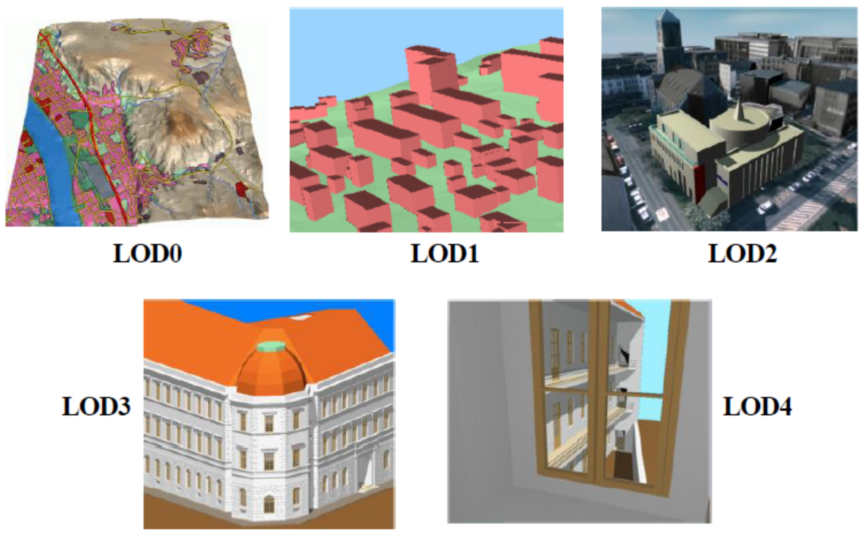


Figure 1.1: Levels of detail of CityGML [Open Geospatial Consortium, 2012b].

roads distinguishes between different LoDs, where LoD0 represents the network and LoD1 and higher represents the areal modelling (Figure 1.2. However, there is no clear difference between LoD2-4.

Proposed changes to the transportation module have been done only recently, mainly focusing on roads. Beil and Kolbe [2017] proposed an adaptation of the LoD specification for roads. Here, both the linear and areal representation have their own LoD specification, with increasing geometric and semantic complexity per representation type. Thus this will enable the possibility of also modelling networks at multiple LoDs. Also, `sections` were introduced in order to be able to segmentise roads into different parts [Beil and Kolbe, 2017]. Labetski et al. [2018] additionally propose a distinction between roads, carriageways and lanes. Also, they stress the need for the modelling of intersections as a separate class. Intersections, including roundabouts, are seen as the most complex parts of road networks, because they can have many different configurations [Quartieri et al., 2009]. The way intersections can be modelled in detail in CityGML has not yet been specified. It is also not yet clear how this can be done at the newly specified different levels of detail.

Criticism of CityGML has not only been directed at the contents of the data model, but also at the structure of the encoding. CityGML is an Extensible Markup Language (XML)-based format. As a result, CityGML files have a hierarchical structure, and thus they are verbose, sometimes hard to comprehend and hard to parse for programmers [Ledoux et al., 2019]. This has lead to the development of CityJSON, a JavaScript Object Notation (JSON) encoding and exchange format of the CityGML data model [CityJSON, 2019]. CityJSON files have a flat structure and are therefore easy to parse and compact in size compared to their CityGML counterparts.

3D city models are increasingly needed for many different applications. Recently, use cases of 3D road data have been identified (e.g.see Labetski et al. [2018]). Different use cases can have different road data needs. Some might need the road network, others detailed areal geometry. Therefore it is important that the data model takes these applications into account. From meeting with government officials and practitioners, it became clear that the current modelling of roads in 3D city models is not sufficient. 3D

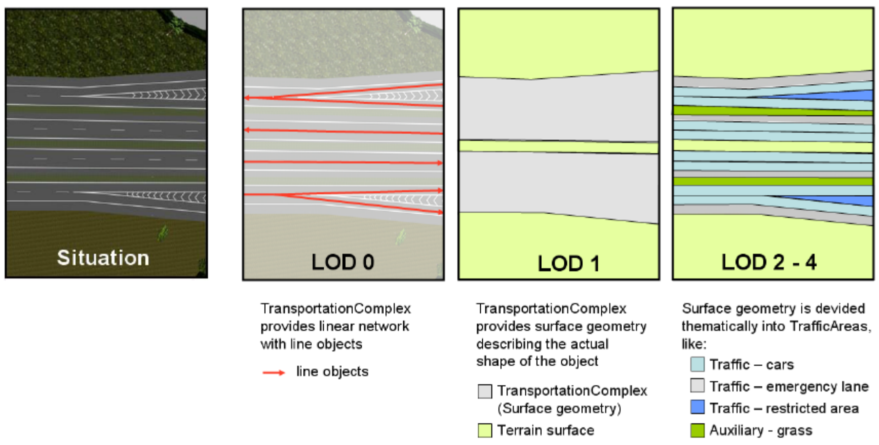

Situation
LOD 0
LOD 1
LOD 2 - 4
TransportationComplex provides linear network with line objects
line objects
TransportationComplex provides surface geometry describing the actual shape of the object
TransportationComplex (Surface geometry)
Terrain surface
Surface geometry is devided thematically into TrafficAreas, like:
Traffic – cars
Traffic – emergency lane
Traffic – restricted area
Auxiliary - grass

Figure 1.2: Road LoD specification of CityGML 2.0 [Open Geospatial Consortium, 2012b].

city models will be of more worth when users can use them for their purpose in the way that they want. The representation of roads in CityGML in particular can still be improved; the different LoDs, the dual linear-areal representation, and intersections can be improved in clarity and structure.

1.2 RESEARCH OBJECTIVES AND METHODOLOGY

The goal of this research is to be able to model roads and intersections in 3D city models at different levels of detail with a data model based on the data needs of use cases. The question I want to answer in this thesis is thus: *how can roads and intersections be modelled in 3D city models at various LoDs such that it suits user needs?* I will focus on adapting the LoD specification of the CityGML transportation module according to the data needs of different use cases. This research is deeply rooted in practice, by meeting with government officials and experts.

In order to answer the research question, the research will be conducted in several steps. The following sub-questions will serve as the building blocks of this research. Combining the answers to these questions should meet the goal stated above. Per sub-question, the methodology I will employ is described. A schematic overview of the methodology used in this research is given in Figure 1.3.

1.2.1 Use case road data needs

The first thing needed are use cases. These are used to articulate the data needs from which I will improve the data model. Thus the first sub-question is: *what are the use cases of roads and intersections in 3D city models and what are their road data needs?* Use cases are selected such that they represent diversity in representation type (linear vs. areal) as well as level of detail. As was discussed, the LoD specification of roads is especially lacking for the linear representation. Therefore I will examine two use cases which mainly use road networks. The other use case is chosen because it is considered to need both a network and an areal representation of roads. This opens up the possibility to study the need for a link between the two representation types

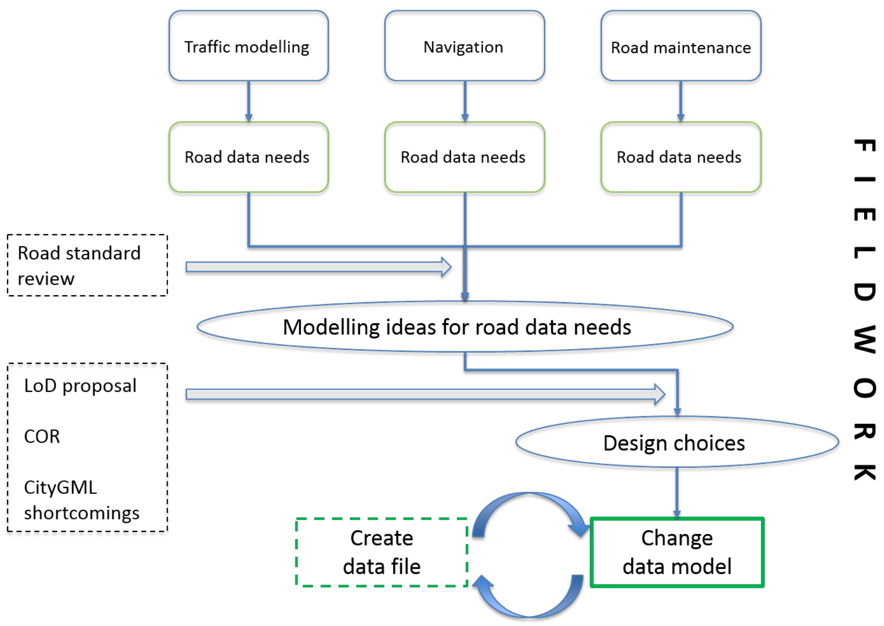


Figure 1.3: Methodology flowchart

in the data model. The data needs of the use cases are assessed through a literature review, as well as by consulting experts (see Section 1.2.5).

As described above, adaptations to the CityGML transportation module have already been proposed. In addition to the use case needs, it is also important to consider the current identified shortcomings of the CityGML data model and the encodings in both CityGML and CityJSON. From this a comprehensive view of data needs can be made.

1.2.2 Reviewing road standards for modelling choices

The second sub-question is: *what road standards exist, and how do they model the identified data needs?* Here I identify multiple road standards and their data models and check if and how they model the data needs identified earlier. In addition to road standards, road data and information models used by the Dutch government will be assessed. This will give insight in modelling choices to be made at a later stage.

1.2.3 Improving the road data model

After having assessed the data needs of the use cases, I want to know: *how can the CityGML transportation data model be improved such that it satisfies the use case data needs?* The acquired data needs have been compared to other road standards; this leads to design ideas. In this step, the design ideas will be implemented in the data model. Then, I choose the CityJSON encoding of the CityGML data model to implement changes schematically. This will lead to a detailed LoD specification of roads and intersections.

1.2.4 Creating a road data file

Finally, when the improved data model has been developed, a data file conforming to the new data model will be made. Road data from the Dutch province of Noord-Brabant will be transformed into a CityJSON road file. Making this data file will lead to a reflection on the modelling choices made. The practicality of certain modelling choices can be assessed. This might lead to a reflection on possible future additions to the road data model.

1.2.5 Fieldwork

As part of this research, meetings were held with different experts. A meeting was held with Stefan van Gerwen, from the Provincie Noord-Brabant. He has provided the dataset used in this thesis, and provided information on how governmental agencies use geographic data and link their own datasets on the maintenance of public space to the governmental information models. A second meeting was with Guus Tamminga, senior advisor mobility at Sweco. He recently completed his PhD thesis on transport and mobility modelling and contributed to the data needs analysis of the transport modelling use case. I also met with Hugo Ledoux, associate professor in the 3D geo-information group at the TU Delft. He is also the initiator of the CityJSON project, and contributed by reviewing the proposed additions to the CityJSON data model done in this thesis. Finally, I attended two meetings of the Werkgroep Wegen (Working Group Roads) of the DiSGeo project of the Dutch government. This group is assessing potential data needs for the three-dimensional representations of roads. This is done in the name of the central object registration project, see Section 2.5.3.

1.3 SCOPE

The following subjects, though relevant, will not be addressed in this thesis.

- The transportation module of CityGML does not only deal with roads, but also with railway tracks. However, this research focuses specifically on roads and therefore railways will not be considered.
- Also, it can be argued that in 3D city models, roads can or should be modelled volumetrically for some use cases. This might be important for maintenance applications concerning sewage systems or utilities, for example. However, in this research the focus will be on two(-and-a-half)-dimensional representations.
- Previous research has been done on LoD specifications. As mentioned above, Beil and Kolbe [2017] and Labetski et al. [2018] have proposed changes to the LoD specification of the CityGML transportation module. As the above already gives a good specification for roads, this research will not focus on developing a whole new LoD specification. Rather, I will build on the research already conducted and aim to add new features to it, such as an LoD description of intersections.

1.4 OVERVIEW OF RESULTS

The goal of this research was to adapt the data model of the CityGML transportation module to the data needs of several use cases. The lack of network modelling possibilities in CityGML was already determined. The use case selection was done such that the need for networks was leading. Researching the CityGML data model and the use cases resulted in a list of data model shortcomings and data needs. For all these data needs, it was checked how they were modelled in other road standards. All these road standards once started from a use case, which is very clear in the way they model the objects (and *which* objects they model). This use case driven approach is also employed here to improve the CityGML transportation module. I chose to encode the improved data model in CityJSON, a JSON encoding of the CityGML data model. CityJSON was chosen because of its ease of use, compactness, and because it is easy to parse. The choice of CityJSON combined with the researched data needs has led to an adapted data model. First, the ability to model networks is added to CityJSON. This ability is then extended with the modelling of a road network. In this, attribute changes can be modelled using graphs nodes. Furthermore, the LoD specification has been explicitly modelled, showing how roads and intersections can be modelled at different LoDs in both representation types. I have also implemented a way of linking linear and areal representations of the same road. This is not done on the smallest scale of road segments, but may be done on a more aggregated level. This way, the potentially highly segmented road network (because of attribute changes) will not interfere with the segmentation level of the areal representation of roads. Finally, CityJSON road data files were made which model a road at the different LoDs. Transforming data to CityJSON shows that some data model adaptations are quite strict, such as the linear modelling of intersections. At the same time, the data model is still quite lenient in how road-related objects can be modelled.

1.5 READING GUIDE

In Chapter 2 research related to this thesis is explored. Terminology used throughout this document is explained, and all of the relevant related subject matter is discussed. In Chapter 3, I study the data needs of three use cases: transport and traffic modelling, navigation, and road maintenance. Before that, the way CityGML and CityJSON model roads is thoroughly examined. This also exposes additional shortcomings in the current modelling approaches. The use case data needs are then compared to how they are modelled in other road standards. The final list of data needs can be found in Section 3.5. After this, Chapter 4 sets out the design choices made in order to accommodate the data needs in the CityJSON encoding of CityGML. The results of the design choices is presented by setting out what has changed and what the new LoD specification looks like. In Chapter 5 I will create a CityJSON road data file which will adhere to the newly improved data model. Chapter 6 will conclude this thesis. Here I will reflect on the use case selection, modelling choices and also the necessity of having roads in 3D city models. This chapter will conclude by setting out possible future work.

2 RELATED WORK

In this section, the related work on the subject of this research will be discussed. Previous research into road data modelling is explored, and relevant concepts are explained. First, road modelling in general is discussed, along with the role intersections have played in it up until now. After that, the related work on 3D city models, CityGML and its transportation module is reviewed. Then, other road data standards are discussed. Lastly, the explorations of road data use cases are reviewed.

2.1 MODELLING OF ROADS

2.1.1 Road representation types

National Mapping Agencies (NMAs) have been gathering topographic data for over two hundred years (see for example Kadaster [2019] and Ordinance Survey [2019]). Roads are also covered by this data. After NMAs digitalised their data at the end of the last century, road data has become available digitally [Ordinance Survey, 2019]. Small-scale topographic road data consists of polygons. Thus this is an areal representation of the real-life road features. From these polygons centre lines can be extracted. These linear road features are often used when modelling roads in higher-scale maps. From linear road features a road network representation can be formed. Government agencies, but also private parties, have constructed road networks for different types of applications, often involving navigation (see for instance Rijkswaterstaat [2017] and TomTom [2019]). The above lead to different ways to store and model road data, see sections 2.4 and 2.5.

2.1.2 Modelling networks

When roads are modelled linearly, it is often done in the form of a network. These networks can be stored as graphs. A graph is a topological data structure which models connectivity between points. The points are called *vertices* or *nodes* and the connections between these nodes are called *edges* or *links* [Singh and Sharma, 2012]. Roads modelled as networks are useful for applications that employ routing, like navigation. Although the topological connection is the main feature of a graph, the edges can be given the geometry of the corresponding road segment (for example, its centre line) such that the graph represents both the geometry and the topology of the road network. Examples of standard that model roads this way are given in Section 2.4.

There are multiple ways graphs can be stored. In essence, the information that needs to be stored is what nodes are connected to each other. One way to do this is with an *edge list*. This list contains a pair of node indices for each pair of connected nodes. The mathematical notation of an edge set

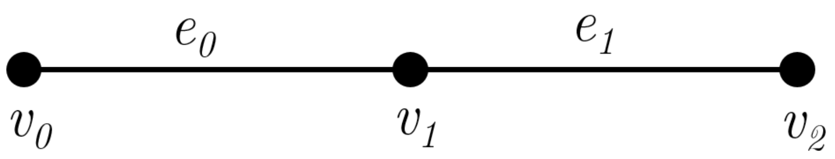


Figure 2.1: An example of a graph with three vertices and two edges.

E of the graph depicted in Figure 2.1 is $E = \{(v_0, v_1), (v_1, v_2)\}$. This is a very straightforward way to represent a graph, however it can be inefficient when checking for existence of a specific edge [West, 2000]. In that case one has to go through the entire list.

A second way to store graphs is with an *adjacency matrix*. An adjacency matrix A is an $n \times n$-matrix, where $n = |V|$, the number of nodes in the graph. The entries of A are either zero or one, where $A_{i,j} = 1$ when i is adjacent to j, and zero otherwise. This format makes checking for adjacency an easy task. The adjacency matrix for the graph in Figure 2.1 is

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

However, when a graph is sparse, meaning that the ratio between edges and nodes is low, this matrix becomes a very verbose way to store little information [West, 2000].

A third way to store graphs is to use an *adjacency list*, where each node index points to the indices of adjacent nodes. For the graph in Figure 2.1 an adjacency list looks like $[[v_1], [v_0, v_2], [v_1]]$. Adjacency lists also result in easy look-ups, but do not require big data structures for sparse graphs. When one employs graphs for solely connectivity purposes, using adjacency lists are thus a good option for storage [West, 2000]. However, as will be seen in sections 2.4 and 2.5, edges in road networks represent real world road features, with a corresponding geometry. These edges are thus explicitly stored, and just having pointers for node connectivity does not say anything about edge geometry. Therefore, graphs represented road networks are probably best modelled using edge lists. This gives way to modelling road segments as features and not just connections.

2.1.3 Linear referencing

Linear referencing is a technique used by institutions that work with linear features, such as transport or utility networks. It is used to reference a position or measure along these linear features [Curtin et al., 2007]. A linear referencing method (LRM) is used to identify a certain location with respect to a known point. That includes being able to identify the point, having a measurement from that point, and a direction in which to measure [Scarponcini, 2002]. A linear referencing system (LRS) uses such LRMs to store and maintain information on events and features that occur along the linear features of a network. Some applications that use LRSs are mapping accidents or incidents along a road, or recording locations of objects such as traffic signs and streetlights [Curtin et al., 2007].

Linear referencing has certain benefits. Having a location in reference to something that can be easily identified, as opposed to a location specified with coordinates, can speed up the localization process [Curtin et al., 2007]. The objects used to reference the location to are often have a real world counterpart like kilometre markers. Emergency services use locations based on their position relative to kilometre markers along the road because it is the fastest way to know where to go [Werkgroep Wegen, 2019a].

On the modelling side, another benefit of linear referencing is that it will not lead to a highly segmented linear network based on differences in attribute values. When an LRS is not implemented, modelling attribute changes in a network will lead to a new segment each time one of the attributes changes. Having an LRS as an organization means having to maintain only a relatively small dataset, which leads to reduced redundancy and less chance of errors in the data [Curtin et al., 2007]. A drawback of implementing an LRS is the fact that one would have to implement such a system over the data, whereas a node-based approach to modelling attribute changes does not require an added structure.

2.1.4 Complexity of intersections

Intersections are the parts of the road infrastructure where multiple roads meet and road users are able to make a choice in direction of further movement. For this reason, intersections have attributes other parts of the road network do not have. These attributes can include information about the number of incident roads or information about restrictions on turning movements. In linear representations, intersections also play an important role as nodes in the graph. They are critical points for road networks as traffic flows converge at these points [Quartieri et al., 2009]. For these reasons, intersections can be considered as the most complex part of road networks.

Different applications will need very different specifications for intersections. Thus there is a need to specify what types of intersections there are, and how to model these with respect to the different levels of detail (see Section 2.2) and the linear and areal representation types. Intersections can be very different from each other topologically, for example T-junctions vs. X-junctions vs. roundabouts [Şerbu et al., 2014]. How to model these different types through different LoDs and different representation in a coherent manner is still an open question.

Intersections can be classified in different ways. Şerbu et al. [2014] classify intersections by the amount of roads that connect and in what configuration. They specify between T-intersections, cross-intersections (four legs), intersections with more than four legs but not circular, Y-intersections, roundabouts and others, see Figure 2.2. Quartieri et al. [2009] do not classify intersections by the amount and configuration of roads converging but by how traffic flow is controlled. They categorise intersections as planar intersection (subdivided in linear intersections and roundabouts) without traffic lights, planar crossings with traffic lights and non-planar intersections, which basically consists of overpasses (for example most highway junctions). Thus intersections can be subdivided into different classes in different ways, depending on the supposed application. How these distinctions can be modelled in a 3D city model has up until now not been researched.

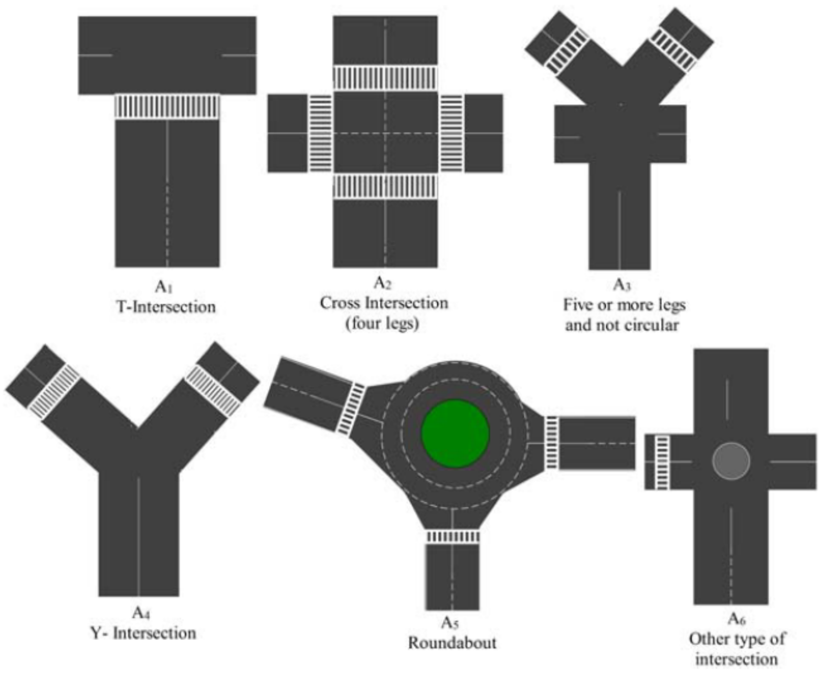


Figure 2.2: Different types of intersection categories, by Şerbu et al. [2014].

Thomson and Richardson [1999] write about the generalization of road networks. They describe the generalization process as basically removing edges from the graph one by one with respect to certain importance measures. As seen above, intersections can take many forms and different configurations. This makes intersections challenging in the generalization context, as removing edges should result in a network that is still topologically correct, and also "makes sense" with respect to the desired generalised output.

2.2 3D CITY MODELS

In the last fifteen years, there has been an increase in the construction and use of 3D city models [Baig and Rahman, 2012; Stoter et al., 2015]. These models consist of digital representations of three-dimensional urban areas and landscapes, including buildings, roads, vegetation and water. 3D (city) models were previously mostly used for visualisation purposes in the field of computer graphics [Biljecki et al., 2016]. However, 3D city models are nowadays increasingly used for many other applications, including spatial analysis, urban planning, navigation and simulation purposes, see for example Baig and Rahman [2012], Biljecki et al. [2015], Stoter et al. [2013] and Kolbe [2009].

Central to 3D city models is the concept of LoD. This concept originated in computer graphics [Luebke et al., 2003], where it represents the geometric complexity and detail of certain objects as they move relative to the viewer [Gröger and Plümer, 2012]. In 3D city models, the LoD represents the model's usability and how well it approximates the real world features [Biljecki et al., 2016]. This concept does not only deal with the geometric aspects like in computer graphics, but also with the semantic ones. Thus, an increasing LoD implies increasing geometric accuracy and semantic richness [Gröger and Plümer, 2012].

Level of detail can also be seen as an acquisition model and product specification [Biljecki et al., 2013]. When the LoDs are clearly defined, data providers and their clients can use LoDs as a way to communicate on their data needs. Furthermore, where LoD in 3D graphics just to visualization and mesh complexity, in 3D city models it is also used as a way to model multiple application requirements [Biljecki et al., 2013]. This means that LoDs are defined such that for use cases it is clear which LoD they need for their purpose. In this sense, having a higher LoD does not mean that a model should be seen as "better", as it might not fit the user's data need. Also, more detailed models are likely to be more error-prone. Having a less detailed model can thus be desirable, also given that they can be considerably smaller in storage size, and better with respect to computation performance [Labetski et al., 2017].

2.3 CITYGML

CityGML is a data model and storage format maintained by the OGC for storing and representing 3D city models [Open Geospatial Consortium, 2012b]. It was developed to have a standardised way of defining the entities, attributes and relations of 3D city models [Open Geospatial Consortium, 2012b]. The current version of CityGML is CityGML 2.0. In CityGML, the geometry and the semantics of objects are both represented in a spatio-semantic model [Kolbe, 2009]. This model is built such that it has spatio-semantic coherence, in which geometries know "what" they are and semantic entities know "where" they are [Stadler and Kolbe, 2007]. These semantics may also define hierarchical structures in the objects [Ledoux et al., 2019]. This makes CityGML very suitable for storing 3D city models. CityGML consists of a core module and thematic modules. These are used to represent the different types of objects, like buildings, roads, water and so on. These data models are conceptually defined by a series of Unified Modelling Language (UML) diagrams [Open Geospatial Consortium, 2012a]. CityGML data files are XML-based encodings of these data models. The CityGML data model is thus schematically defined by XML schema files (XSD) files. The CityGML data model can be extended by using Application Domain Extensions (ADEs). These extensions are also modelled with XSD files. They allow users to define their own objects and attributes to add to the data model [Kolbe, 2009].

2.3.1 LoD specification

CityGML differentiates between five LoDs, ranging from LoD0 to LoD4, where LoD4 is geometrically and semantically the most elaborate. Each thematic module has its own LoD specification. The LoD specification for buildings is the most elaborate and, probably, the most discussed, see for example Baig and Rahman [2012], Gröger and Plümer [2012], Biljecki et al. [2016], Labetski et al. [2017] and Löwner and Gröger [2016]. For buildings, LoD0 actually refers to a 2.5D digital terrain model. Buildings in LoD1 are block models, footprints extruded to a height value. LoD2 allows one to add more complex roof structures and buildings parts, and LoD3 should denote a complete architectural model of the external part of the buildings.

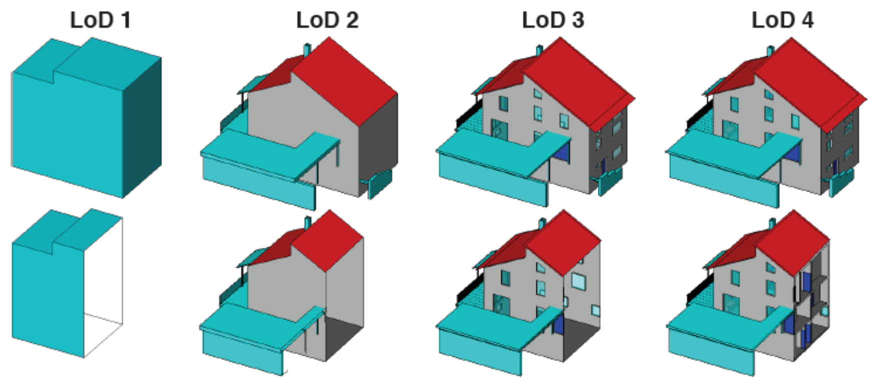


Figure 2.3: Schematic overview of CityGML building LoDs [Häfele, 2011]. LoD0 is omitted.

LoD4 is LoD3 with the internal objects added [Kolbe, 2009]. A schematic overview of LoD1-4 for buildings is shown in Figure 2.3. Buildings make up the bulk of actual 3D objects in 3D city models. Thus it can be expected that most research has been done on the LoD specification of buildings [Beil and Kolbe, 2017]. However, given that different LoDs are closely tied to its applications, other thematic models of CityGML might also benefit from a more established LoD specification.

Biljecki et al. [2013] identify certain shortcomings in the CityGML LoD specification. The amount of semantics required per LoD can be quite minimal, and there is not a normative requirement for additional semantics beyond that. At the same time, the LoDs sometimes prescribe an upper limit for LoDs, instead of a base requirement. For example, LoD2 buildings should not contain openings like windows, but it is not stated that LoD3 buildings *have to* contain openings [Biljecki et al., 2016]. There does not seem to be a uniform approach that drives the definition of LoDs [Biljecki et al., 2013]. The LoDs are generated regardless of the application. Biljecki et al. [2013] specify different factors that form properties of LoDs, like presence of certain `CityObjects`, geometric complexity, positional accuracy, depth of semantic hierarchy, and so on. They define level of detail of a 3D city model as "a quality measure of the model which has a minimum and sufficient and sensible mix of the amount of each factor for usable applications" [Biljecki et al., 2013, p. 69]. A part of this definition is that it that LoDs must provide clear constraints per LoD. This definition has been used to define a new LoD specification for buildings [Biljecki et al., 2016].

2.3.2 Representation of roads

In CityGML, roads are modelled using the `Transportation` module. In contrast to buildings, roads in CityGML are not modelled with volumes and can therefore be modelled using two-dimensional geometries [Open Geospatial Consortium, 2012b]. Given that road data is often used to find out what can be reached from where, road networks are often modelled using at most one-dimensional objects, namely nodes and lines [Beil and Kolbe, 2017]. For other applications, such as serious games, driving simulation and autonomous driving cars, areal representations might be used. In research devoted to areal road models, little attention was paid to the actual geometrically correct modelling of real-world roads [Nguyen et al., 2016]. CityGML 2.0 combines the above in its LoD specification for roads [Open Geospatial Consortium, 2012b], where LoD0 represents a line representation and LoD1 and higher represents the areal modelling. In Figure 1.2 a schematic overview of the different LoDs is given. As seen in the figure, there are no specified differences between LoD2-4. The only difference between LoD1 and LoD2 is that the road is subdivided into different thematic road parts. Also, this LoD specification implies that the areal representation is more detailed than the linear network representation. While geometrically this is correct, an elaborate network might contain much more useful information than just a sequence of LoD1 surfaces. Furthermore, having just one LoD for linear representations makes it difficult to specify between networks of different level of detail. Lastly, the linear representation is made with a GeometricComplex, consisting of linestrings. Nodes are not explicitly present in the linear model, therefore it does not constitute a full network. Thus, there is no way to model intersections linearly [Open Geospatial Consortium, 2012b].

Labetski et al. [2018] reviewed different road standards and concluded that the LoD specification for roads in CityGML is not well-established. They also concluded that in the other road standards there is a focus on linear road representations. Beil and Kolbe [2017] also reviewed road standards and CityGML 2.0, and provided a starting point for an improvement of the transportation module. Among other proposed changes, they propose adaptations to the LoD specification. In their proposal, the linear representation is not restricted to just LoD0, but it can also be modelled up to LoD3 with increasing complexity, similar to the areal representation. This is schematically represented in Figure 2.4. At LoD1, the areal representation covers the entire road, and at LoD2 it is divided up in different `TrafficAreas`. The corresponding LoD network follows this segmentation. They also remove LoD4 from the model, as this is only used to add inside features of objects and this does not apply to roads.

Beil and Kolbe [2017] also introduce the concept of a road `section`. Now in CityGML it is possible to model a whole road network with just one `Road` object. Sections make it possible to subdivide `Road` elements into smaller pieces (for example, so that no sections cross other sections, but just touch). Then sections with a shared attribute (e.g. street name) can be aggregated to retrieve the whole Road element [Beil and Kolbe, 2017].

After reviewing the proposed changes of Beil and Kolbe [2017] and consulting with government agencies for a needs analysis, Labetski et al. [2018] proposed more changes to the CityGML transportation module. One proposed change is to distinguish between roads, carriageways and lanes at LoD3, shown schematically in Figure 2.5). Here, a road represents the whole area which permits traffic flow, a carriageway the part with the same traffic direction and the lanes the individual driving lanes. They also agree to omit LoD4 from the transportation module. This results is a more distinct LoD specification than seen in Figure 1.2.

Labetski et al. [2018] propose four different LoDs for road networks. Given that LoD0 is originally the only LoD which models linear representations of roads, the four network LoDs are named LoD0.0 through LoD0.3. LoD0.0 is the original LoD0: linear roads modelled with solely lines. From LoD0.1, nodes are introduced such that a network can be formed. From LoD0.1 to LoD0.3, the complexity of the network increasing similarly as to that proposed in the areal representation [Labetski et al., 2018], see Figure 2.7. The idea is to increase the network complexity akin to the areal representation in

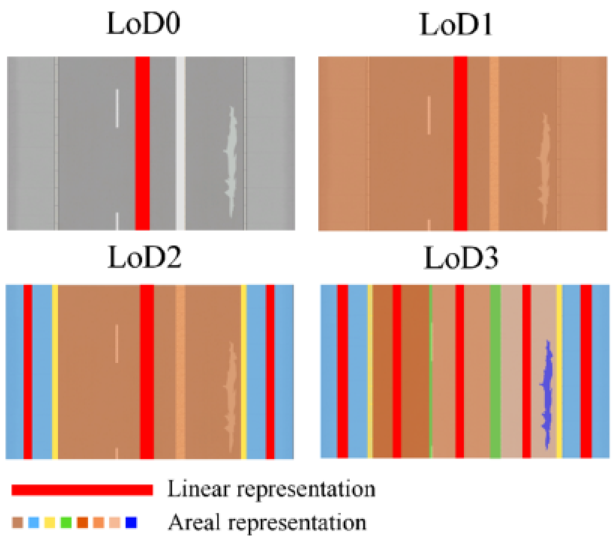
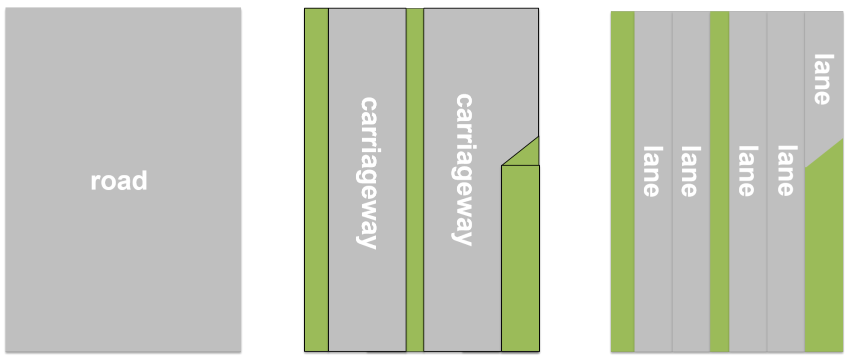


Figure 2.4: Improvement of LoD specification by Beil and Kolbe [2017].



(a) LoD1 (b) LoD2 (c) LoD3

Figure 2.5: Roads, carriageways and lanes as proposed by Labetski et al. [2018].

the same corresponding LoD. This means that, for instance, in LoD0.3 each driving lane would be modelled with its own line. From now on, when referring to the LoD specification of linear road representation, I will write LoD0.0 – LoD0.3. Moreover, given that most linear road data applications use network data, LoD0.0 is not really considered in this research. Thus, often I will refer only to LoD1 – LoD3 and LoD0.1 – LoD0.3.

The proposal to split up geometries according to carriageways and lanes at higher LoDs needs a discussion on how to define these concepts [Van Gerwen, 2019]. When a dual carriageway highway is modelled, the definition of both carriageways and lanes are clear. The presence of a median strip creates an obvious boundary between the carriageways. However, a road consisting of two lanes – one for each driving direction, like in Figure 2.6 – does not have two separated carriageways. It is not clear whether the proposed LoD2 carriageway specification focuses on having separate surfaces for driving directions, or whether these actually need to be physically separated.

Another proposed change to the data model by Labetski et al. [2018] is that intersections are explicitly modelled as a separate class. Tamminga [2019b] further stresses that the roads in CityGML lack attribute options for

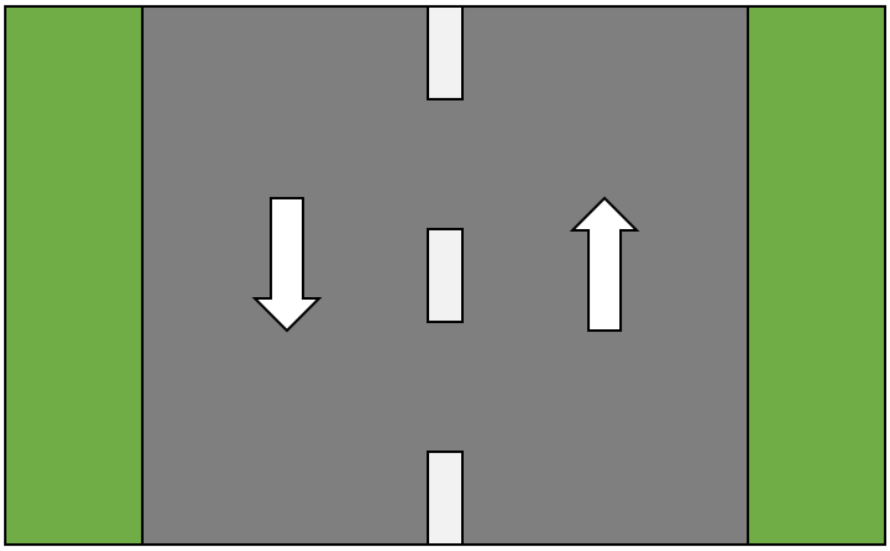


Figure 2.6: Modelling ambiguity: can the lanes also be considered carriageways?

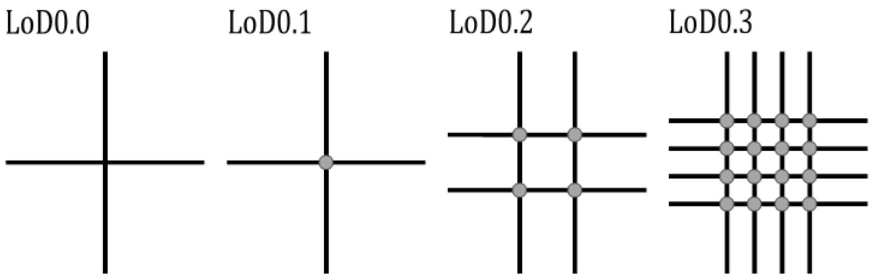


Figure 2.7: Differentiation of network LoDs [Labetski et al., 2018].

lanes. Also, he identifies the need to link the linear and areal representations of the same object to each other. This should be easier with the proposed LoD specifications for both linear and areal road representation types by Labetski et al. [2018]. It is not yet researched how these proposed changes should be implemented or encoded in CityGML.

2.3.3 CityJSON

CityJSON is a data format for encoding (a subset of) the CityGML 2.0 data model using JavaScript Object Notation [Ledoux et al., 2019]. CityGML is an XML-based format and can therefore be verbose and complex to work with. CityJSON flattens out the hierarchical structure of CityGML as much as possible. By this, CityJSON aims to be much more compact and easier to work with for programmers. The current version of CityJSON is 1.0 [CityJSON, 2019]. The entire CityGML data model has been encoded in CityJSON, except for LoD4. The reason for this is that up until now LoD4 datasets have been very rare [Ledoux et al., 2019]. All other LoDs have been encoded in CityJSON, however sometimes in a different structure, to ensure the 'flatness' of the model. CityJSON also allows extensions of the model. These extensions are similar to the concept of ADEs in CityGML.

The (transportation) data model of CityGML and its encoding in CityJSON are extensively discussed further in Section 3.1.

2.4 OVERVIEW OF ROAD STANDARDS

In this section, current road standards are discussed in order to see how they represent geometry, topology, semantics, LoD and intersections. This section is partly based on Beil and Kolbe [2017] and Labetski et al. [2018]. An overview of the standards is shown in Table 2.1.

2.4.1 Geographic Data Files

Geographic Data Files (GDF) is an International Organization for Standardization (ISO) standard which provides a data model and an exchange format for structured road network data [ISO, 2011]. It also provides other geographic information needed for applications in intelligent transportation systems (ITS) [Essen and Hiestermann, 2005]. ITS are systems where new technologies are used which result in smart and efficient usage of multimodal transportation networks [Boxill and Yu, 2000]. Transportation objects in GDF are modelled in three categories, from level 0 to level 2; these levels do not correspond to CityGML LoDs. At all three levels, it is essentially a combination of linking road segments to points that represent intersections. At level 0, roads are represented by straight line segments, with nodes at both ends of each line segment. At level 1, `Road Elements` are ordered lists of level 0 segments which constitute a road between two junctions. `Junctions` are defined as the place where three or more `Road Elements` meet. At level 2, the road network is composed of `Roads` and `Intersections`. Here, `Roads` are constructed from level 1 `Road Elements`, but it can also combine different carriageways. Intersections can be T- or X-junctions, or more of complex types like roundabouts or clover-leaf designs. GDF objects can have many attributes assigned to them, these differ across the different levels [ISO, 2011].

2.4.2 OpenDRIVE

OpenDRIVE is an open data format, in which road network data is stored in an XML-based format. It was developed for use in driving simulations, where the OpenDRIVE road network data interacts with the other objects which are visualised in the simulation [Dupuis et al., 2010]. In OpenDRIVE, the road network is modelled using reference lines. These consist of sequences of geometric primitives. OpenDRIVE does not have a notion of LoD. Details regarding the number of lanes are added as attributes. Road crossings in OpenDRIVE can be modelled in two different ways. Two roads can be labelled as each other's predecessor and successor (called *standard linkage*), or can be made into a `junction`. `Junctions` link in-coming roads to out-going roads through connecting paths. `Junctions` are used when the linkage between two roads can be ambiguous. `Junctions` then consist of a connection matrix which indicates the possibilities of entering a road from another [Dupuis, 2015].

2.4.3 LandInfra

LandInfra is an OGC standard, short for Land and Infrastructure Conceptual Model Standard. LandInfra defines concepts for land and civil engineering infrastructure applications. This model is mainly used to model data for exchanging road designs [Scarponcini, 2016]. The base class in LandInfra is `Facility`, which represents an infrastructure facility. `Facilities` can be simple or very complex. The `Road` features are parts of a `Facility` that are single segments of roads that are continuous, non-overlapping and non-branching, although they may contain intersections with other roads. A `Road` feature consists of multiple `RoadElement` features, which specify the parts of the `Road` segment. This can for instance be pavement, curb, shoulder, etc. Roads can be represented by triangular irregular network (TIN) surfaces. LandInfra does not have a separate class for intersections. It also does not have a concept of level of detail [Open Geospatial Consortium, 2016].

LandInfra uses alignment, which is a positioning element used for linear referencing. It consists of geometric segments, mostly 2D linestrings, along which some form of distance can be measured using a linear referencing method. This can be useful for maintenance operations, for instance. Road features can have any number of `Alignments`. Typically, centre lines of roads have an `Alignment`. Dual carriageway roads can consist of different `Road` features and may thus have different centre lines and corresponding alignments [Open Geospatial Consortium, 2016].

2.4.4 RoadXML

RoadXML is a road data standard that was originally designed for driving simulation purposes. It consists of four different layers of information. The topological layer represents the network and location. The logical layer represents the significance of the road elements in the environment. The physical layer represents the physical properties of a road element, like the road surface. Lastly, the visual layer represents the element's geometry and 3D representation. The network is built up from different `Subnetworks`. These `Subnetworks` consist of `Tracks` and `Intersections`, and these can be enhanced with the different layers of data [Ducloux, 2016].

A `Subnetwork` has a `Profile` which consists of `Lanes` and `LaneBorders`. `Intersections` are used to connect `RoadElements` in a `Subnetwork`. `Subnetworks` are joined together using `Subnetworkjunctions`. These can only connect two different `Tracks`, not `Intersections`. RoadXML does not have a level of detail concept [Ducloux, 2016].

2.4.5 OpenStreetMap

OpenStreetMap (OSM) is a free online map that anyone can edit. Along with the map itself, the data gathered from individual contributions from volunteers is considered the main output [OSM, 2019]. OSM employs a topological data structure in which physical ground features are modelled. The data structure consists of four data primitives: `Nodes`, `Ways`, `Relations` and `Tags`. `Nodes` are points with coordinates stored in WGS84. They are either used to represent point map features, or used to represent `Ways`. `Ways` are ordered lists of `Nodes`. `Ways` can represent lines features or, if the `Nodes` form a closed loop, polygons. `Relations` model logical or geographical relationships between objects. It links `Nodes` and `Ways` with certain roles.

Standard	**Geometry**	**Topology**	**Semantics / Attributes**	**LoD concept**	**Intersections**
GDF	Linear	Node and link at Level 1, Road and Intersection at Level 2	Attribute catalogue, for RoadElements and Junctions	Three levels	Nodes at level 1, complex intersections at level 2
OpenDRIVE	One ref. line per road	Predecessors and successors, no nodes	Everything in relation to the ref. line	Not modelled	Connecting lines in the Junction area
LandInfra	Areal, 3D	No	Alignment through a linear reference system	Not modelled	No
RoadXML	Linear	Network	Attributes can be added to lanes	Not modelled	Intersections of Subnetworks
OSM	Linear, cartographically as surfaces	Nodes and Ways result in a network	Tag catalogue, Relations between Nodes and Ways	Not modelled	(Complex) intersections modelled with (multiple) nodes
ITF	Lane information as offsets from the Intersection middle point	Lane centre line modeled as sequence of nodes away from the ConflictArea	Attribute catalogue for intersections, LaneSet is an attribute. LaneSet has its own attribute catalogue	Not modelled	Intersection as ConflictArea, with explicitly modelled turning lanes

Table 2.1: Overview of road standards.

Tags are key-value pairs which are always attached to a map feature. They describe attributes or types of `Nodes`, `Ways` and `Relations`. The possible values of `Tags` are agreed upon by the OSM community and the prescribed use is documented on the Wiki [OSM, 2019].

Roads are modelled using `Ways`. A road has a `Tag` with key "Highway", and the road type is determined by the value of the `Tag`. The list of possible values of the "Highway" key is elaborate. The value also determines the cartographic appearance of the object. Thus where roads appear on the map as two-dimensional objects, they are modelled by non-closed `Ways` [OSM, 2019]. Intersections are not separately modelled, but represented by a `Node` linking multiple `Ways`. Highways do have a possible `Junction` tag, with which a roundabout can be modelled. Here, roundabouts are modelled as one way streets with the tag `Junction=Roundabout` added. All other attribute information is also added as Tags, for example number of lanes, driving direction, etc. [OSM, 2019]. OSM does not have different modelling LoDs.

2.4.6 Intersection Topology Format

Innovations in smart mobility lead to possibilities for communication between smart traffic lights and vehicles. This communication can be used to increase flow of traffic and, for example, give priorities to certain emergency vehicles. Further development can also lead to implementations for automated driving. The Intersection Topology Format (ITF) is a Dutch standard which gives information on the configuration of intersections [CROW, 2018]. It is mostly based on the MapData (MAP) message [Southwest Research Institute, 2018]. The MAP message is a way for vehicles and smart traffic controllers to communicate. The message includes complex intersection descriptions. ITF can be seen as an extension of the MAP message, with added elements which are relevant in the Netherlands.

MAP messages contain information on one or more intersections. Each intersection contains one reference point which represents the intersection (also called the conflict area). Each intersection has a `LaneSet`, which contains the properties of all the lanes of an intersection, like allowed movements to the other lanes. Each lane is part of an approach, which can be considered as the whole road incident to the intersection. An intersection lane is either an `ingressApproach` lane (incoming) or an `egressApproach` lane (outgoing). Each lane has a `nodeList`, a sequence of point values (stored as offset from the reference point) used in order to build the centre line of the lane. Nodes are always sequenced away from the conflict area. Thus the first node of an ingress lane is positioned at the stop line, while the first node of an egress lane is positioned at the end of the conflict area. The way vehicles move through the conflict area is modelled by stating which egress lanes can be reached from an ingress lane, and that trajectory is modelled by a sequence of nodes. ITF does not have an LoD specification, its only LoD is very detailed [CROW, 2018].

2.5 ROAD DATA IN THE NETHERLANDS

Governmental agencies in the Netherlands provide road data in different data models, information models and key registers. The most important are outlined below.

2.5.1 BGT / IMGeo

The Basisregistratie Grootschalige Topografie (BGT) is the Dutch key register for large-scale topography [Van den Brink et al., 2013]. The BGT information model is Informatiemodel Geografie (IMGeo). Part of IMGeo is mandatory for all BGT objects. The other part of IMGeo can additionally be used in order to add many desired attributes to objects if needed. The aim of the BGT is to describe all objects in the country in 2D, giving a two-dimensional planar partition. Thus, almost all objects represented in BGT are areal. The roads are modelled using two classes, `Wegdeel` and `Ondersteunend wegdeel` (analogous to `TrafficArea` and `AuxiliaryTrafficArea` from CityGML). These can then be given a function, e.g. lane of highway or bicycle path, and an attribute about its physical appearance. The BGT does not have centre lines and does not give any information about connectivity of roads. Intersections are thus also not specified. The BGT is expected to expand to 3D geo-information at a later stage. IMGeo is modelled as an Application Domain Extension (ADE) of CityGML [Van den Brink et al., 2013].

2.5.2 NWB

The Nationaal Wegenbestand (NWB) is the Dutch national road register for public roads that are maintained by any of the layers of government. All streets, footpaths, bicyclepaths or dirt roads are present, as long as they have a name or a number [Rijkswaterstaat, 2013]. The NWB is a network representation of the road network of these roads. It was developed as a base register on which other internal databases can be built. Like GDF, it consists of links and nodes (`Wegvak` and `Junctie` respectively). Roads are split into a new `Wegvak` when for example it crosses local government boundaries or changes name (becomes another road). Every `Wegvak` is bookended by two `Juncties`. When driving lanes are physically split, they will each have their own `Wegvak`. The direction of traffic is specified for each `Wegvak`. The `Juncties` function as intersections in the NWB [Rijkswaterstaat, 2017].

The basic structure of the NWB is thus very basic. The data model defines many possible attributes for the `Wegvakken` and `Juncties`. The NWB also contains information about so called "hectometerpalen", marker posts which are present every 100 meters along most numbered roads in the Netherlands. Rijkswaterstaat and other Dutch governmental agencies gather data with respect to these posts. Therefore the NWB supports linear referencing in order to support integration of these datasets with the NWB [Rijkswaterstaat, 2013].

2.5.3 Future: central object registration

The Netherlands has a system of different key registers, where information is gathered once so it can be used many times. There are differences in definitions and time accuracy between the different key registers, and also with respect to other data models [Werkgroep Wegen, 2018]. The Dutch government has therefore started a pilot to explore the possibility of one coherent object register, in which the real-life objects can also be represented in 3D. In this scenario, the modelled objects are seen as the base, instead of the demands of the different key registers. For example, buildings are represented the earlier mentioned BGT, but also in the key register Basisregistratie Adressen en Gebouwen (BAG), for buildings and addresses.

	Areal			Linear		
	LoD1	LoD2	LoD3	LoD0.1	LoD0.2	LoD0.3
Road repair			x			
De-icing roads		x	x		x	x
Disaster management			x			x
Surface heat monitoring		x	x			
Air quality monitoring	x	x	x			x
Visibility analysis		x	x			
Noise mapping	x	x	x			x
Traffic light configuration			x			x
Traffic simulations		x	x		x	x
Routing / navigation				x	x	x
Autonomous driving			x			x

Table 2.2: Road data LoDs needed by potential applications of 3D city models. LoD0.0 is omitted, given that networks almost always needs nodes.

This leads to having the same object represented twice in key registers, sometimes with slightly different definitions. Meanwhile the purpose of having these registers is that information only has to be collected once.

One working group was formed which focuses strictly on roads [Werkgroep Wegen, 2018]. It identifies two types of users, one which is primarily interested in topography (areal representation) and one which is primarily interested in topology (lines and nodes). The working group stresses that these two groups can have need for the other type of representation, and therefore adds that an integral topography-topology approach is needed for the coherent object register. The exploration takes current Dutch key registers and other governmental data models as a starting point [Werkgroep Wegen, 2018], which were described above.

2.6 3D CITY MODEL ROAD DATA USE CASES

With the growth of the amount of 3D geo-information available, the potential in using that data for various applications also increases [Stoter et al., 2013]. Ross [2010] divides 3D use cases into three categories: 1) applications using solely geometry, 2) applications based on both geometry and semantics, and 3) applications based on ADEs and external data. Biljecki et al. [2015] note that certain use cases will be a combination of these categories, especially when geometries are semantically enriched with external data.

Biljecki et al. [2015] have made an extensive state of the art review of applications of 3D city models. Many of these applications are not related to roads. Earlier it was established that many road data standards are focused on linear representation [Labetski et al., 2018]. Given that 3D city models mostly model roads areally, it is perhaps to be expected that road data use cases do not employ 3D city models for their application. Still, Labetski et al. [2018] list potential applications for 3D road data, adapted from Beil [2017]. They also consulted with governmental agencies. From this, several

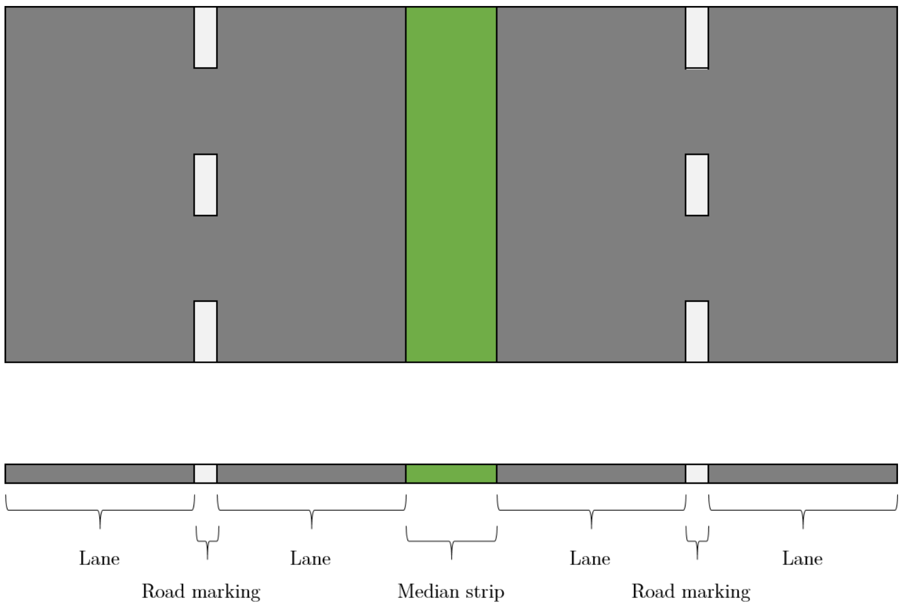


Figure 2.8: An overview of a road with its cross section. The thematic division below resembles the proposed LoD3 in Labetski et al. [2018].

applications of road data for use cases like road maintenance and traffic modelling were identified.

The previously mentioned COR working group [Werkgroep Wegen, 2018] identified different themes as being users of road data. These themes are, among others, mobility (traffic and (public) transport, logistics, hazardous materials), parking, maintenance, water management, air quality and noise, order and safety, information and communication. As part of their research into the central object registration, for these themes they explored which representation types of roads were needed for different applications [Werkgroep Wegen, 2019b]. An areal representation of roads is suitable for road maintenance, while a network is needed for routing. However, for many applications they identify a need for a combination of both representation types. These use cases include information systems for emergency vehicles, de-icing of roads, noise analysis, air quality analysis and traffic modelling [Werkgroep Wegen, 2019a].

An overview of potential use cases for road data in 3D city models identified in the explorations above is given in Table 2.2. For each use case, the presumed needed LoD(s) of the roads is given. The needed LoDs marked in this table are not definite. For example, noise mapping is said to use LoD1, LoD2, LoD3 and LoD0.3. All three areal LoDs are given, because modelling noise propagation could be done with models in all three LoDs. However, using a higher areal LoD will probably result in a more accurate result (assuming the data geometric data is also more accurate). For example, when a road is modelled with high geometric and semantic level of detail, as in Figure 2.8, the location of the cause of the noise can be more accurately determined. LoD0.3 is also mentioned, because adding a high LoD road network can also aid in locating where the most heavy traffic passes. Thus, the choice of LoD may depend on what data is available, and also the scale of the simulation one wants to run. One might want to use a low LoD dataset to map noise propagation for a large area.

In addition the working group identified some use cases that could benefit from a volumetric road representation. These mostly concern road, intersection and bridge design and maintenance, in addition to modelling service networks [Werkgroep Wegen, 2019a]. As mentioned in Section 1.3, a volumetric approach to modelling roads is not further explored in this thesis.

3 ROAD DATA NEEDS ANALYSIS

In this chapter I will try to answer the first two sub-questions outlined in Section 1.2:

- What are use cases of roads and intersections in 3D city models and their road data needs?
- How do other road standards model these data needs?

In the previous chapter, several shortcomings in how roads and intersections are currently modelled in CityGML were set out. The LoD specification for roads is not sufficient for various applications [Labetski et al., 2018]. The LoDs of CityGML should be defined such that they serve as a model for a certain set of use cases [Biljecki et al., 2013]. In Section 2.6 a summary has been given into use cases for road data in 3D city models, partly answering the first sub-question. It followed that while there are applications which require network representations at various levels of detail, it is currently not possible to model this in CityGML. Given that the areal LoDs are already more developed, I have decided to focus mainly on use cases which use road networks. In total I will check the data needs of three different, broadly defined, use cases: transport and traffic modelling, car navigation and road maintenance. As we will see, the latter use case relies on both representation types. This use case has been selected to try to incorporate the idea of a central object registration, mentioned in Section 2.5.3.

The data needs of the use cases are extracted through literature review. After, these data needs are compared to how other road standards (set out in Section 2.4) model these data needs. This will answer the second sub-question. The comparison will lead to modelling ideas which may be used for implementing the data needs in Chapter 4. The data need analysis and standard comparison are done per use case in Section 3.2 through Section 3.4. The data needs are also compared to what is already modelled in the CityGML transportation module data model. To be able to do that, first the CityGML transportation data model and the data encodings of CityGML and CityJSON are explored below in Section 3.1.

3.1 CITYGML DATA MODEL

In this section the CityGML transportation data model will be examined. Other modules that contain mobility-related objects are also checked. The goal is to understand how roads and related objects are modelled, and to identify current shortcomings in modelling these objects. Moreover, the encoding of these objects in CityJSON is assessed and compared with how it is encoded in CityGML. Finally I compare the extension possibilities of both encodings. The goals of this section are: to understand how roads and related objects are modelled, to identify current shortcomings in modelling these objects (which may be already known), and I want to compare the use case data needs to what is now possible in the CityGML data model.

3.1.1 Transportation module

CityGML is partitioned into different modules. It has a Core module, in which the base class of the CityObject is defined. The UML diagram of the Core is found in Appendix A. City models in CityGML consist of a collection of these CityObjects. All objects are modelled as subclasses of the abstract CityObject type. CityObjects have as attributes "class", "function" and "usage". Next to the core module, CityGML has thematic modules that represent the subclasses of the CityObjects. These subclasses represent buildings, water, vegetation, transportation, and so on. As the thematic classes are subclasses of the CityObject, thematic objects will inherit the attributes of the CityObject type [Kolbe, 2009].

One of the thematic modules of CityGML is transportation. Its UML diagram is shown in Figure 3.1. This module enables one to model infrastructural objects like roads, squares and railway lines. The main class in this module is the TransportationComplex. A TransportationComplex is an aggregation of TrafficAreas and AuxiliaryTrafficAreas. TrafficAreas are representations of those objects that are used for hosting traffic. Thus they represent driving lanes, pavements, cycle lanes etc. AuxiliaryTrafficAreas describe the other elements in roads like kerbstones and patches of vegetation between two carriageways [Open Geospatial Consortium, 2012b]. A schematic overview of this division is shown in Figure 3.2.

TransportationComplexes can be thematically divided into subclasses Railway, Road, Square and Track. It is a subclass of CityObject, and therefore inherits the attributes "class", "function" and "usage". Here, "class" denotes the classification of the objects. "Function" describes the purpose of the objects (for example highway, airport, bicycle lane), while "usage" might be used if the actual usage of the object differs from the original intended use. Both "function" and "usage" may be used multiple times per feature. TrafficArea and AuxiliaryTrafficArea both also have the attribute "surfaceMaterial", in which the surface of the modelled object is described. The OGC prescribes enumerated code lists which state possible values for the three attributes [Open Geospatial Consortium, 2012b].

The LoD specification for transportation objects states that LoD0 has a linear representation of the objects. Here, the TransportationComplex is modelled with a GeometricComplex. At LoD1, the TransportationComplex is represented by one MultiSurface, embedded in three-dimensional space. From LoD2 and higher, the TransportationComplexes become aggregates of the TrafficAreas and AuxiliaryTrafficAreas. The latter two are then again modelled by single MultiSurfaces [Open Geospatial Consortium, 2012b]. An example of a road object in CityGML is given in Figure 3.3.

3.1.2 Bridges and tunnels

Road networks also include bridges and tunnels. In CityGML, these objects each have their own thematic module, which are quite elaborate. The Bridge module is similar to the building module in LoD specification. LoD1 is a footprint extruded to a certain height, where LoD2 and LoD3 have increasing geometrical and semantic detail. LoD4 includes interior structures. As Bridge is a subclass of CityObject, it also has attributes "class",

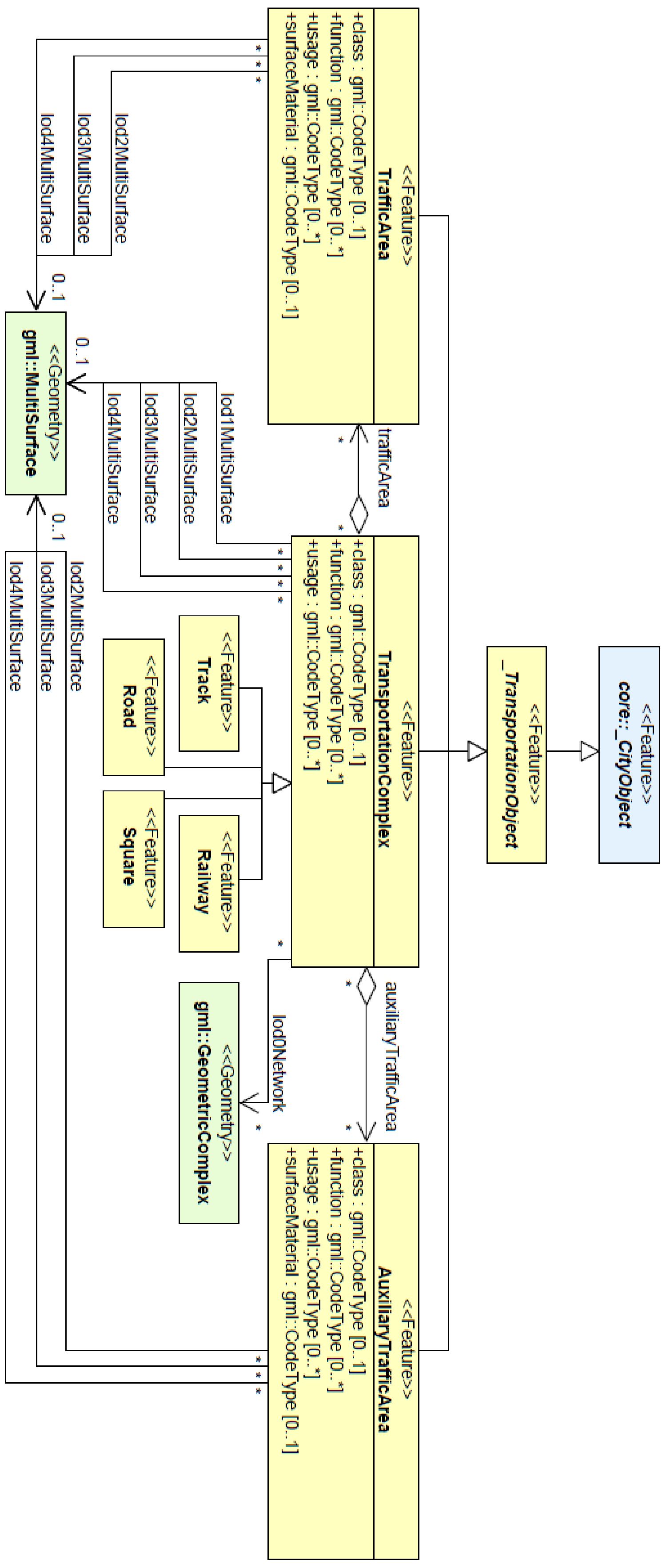


Figure 3.1: UML diagram of the Transportation module of CityGML [Open Geospatial Consortium, 2012b].

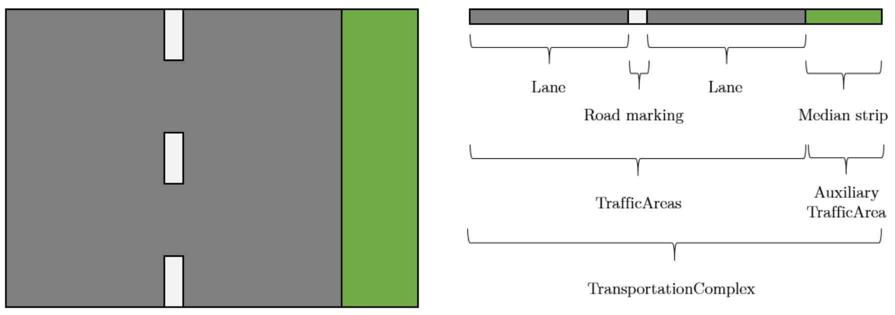


Figure 3.2: Subdivision of a road in the CityGML transportation module.

```xml
<core:cityObjectMember>
 <tran:TrafficArea gml:id="_A299D47AD4E6D2BB7E0532B0B5B0AE93E">
  <core:creationDate>2014-02-13</core:creationDate>
  <tran:class>local carriageway</tran:class>
  <tran:surfaceMaterial>surfaced pavement</tran:surfaceMaterial>
  <tran:lod2MultiSurface>
   <gml:MultiSurface srsName="EPSG:7415" srsDimension="3">
    <gml:surfaceMember>
     <gml:Polygon>
      <gml:exterior>
       <gml:LinearRing>
        <gml:posList>94273.344 463812.831 0.6688626441047193
         94260.472 463809.828 0.583103089885288 94272.374
         463807.149 0.6979061812650841 94273.344 463812.831
         0.6688626441047193</gml:posList>
       </gml:LinearRing>
      </gml:exterior>
     </gml:Polygon>
    </gml:surfaceMember>
    <gml:surfaceMember>
     <gml:Polygon>
      <gml:exterior>
       <gml:LinearRing>
        <gml:posList>94261.624 463815.323 0.5593409338157872
         94260.472 463809.828 0.583103089885288 94273.344
         463812.831 0.6688626441047193 94261.624 463815.323
         0.5593409338157872</gml:posList>
       </gml:LinearRing>
      </gml:exterior>
     </gml:Polygon>
    </gml:surfaceMember>
   </gml:MultiSurface>
  </tran:lod2MultiSurface>
 </tran:TrafficArea>
</core:cityObjectMember>
```

Figure 3.3: Example of an encoding of a road object in CityGML.

"function" and "usage". The attribute "function" may describe whether the bridge is used for railway transport, pedestrians etc. These values are again given in code lists maintained by the OGC. From this attribute the type of transportation usage can be inferred. At LoD2 and higher, the traffic area of a bridge may be modelled with an `OuterFloorSurface`. However, these surfaces do not contain any more information which pertains to transportation infrastructure [Open Geospatial Consortium, 2012b].

The thematic module for tunnels is also somewhat similar to that of buildings. A `Tunnel` can be subdivided into `TunnelParts`. Again, LoD1 is a block model, with an extruded footprint, LoD2 and LoD3 are more detailed versions of the exterior, and LoD4 contains the interior. One could argue that the interior is the most important part of a tunnel when modelling. However, for consistency reasons it was decided that the interior is only modelled at LoD4, without representations at other LoDs. This can be done by using the class `HollowSpace`. Here, `FloorSurface` may be used to model the transportation infrastructure object at the "bottom" of the inside of the tunnel. This corresponds to the road surface. Again, the attribute "function" inherited from the superclass `CityObject` can say something about the traffic mode that uses the tunnel [Open Geospatial Consortium, 2012b].

3.1.3 CityFurniture

Traffic lights or bus stops are examples of objects that can be of influence on transportation. These immovable objects can be modelled using the thematic module `CityFurniture`. The model only describes that a `CityFurniture` has a geometry at an LoD, and that it inherits the attributes "class", "function" and "usage". The code list provided by the OGC is extensive, but will not provide details on, for example, how a traffic light works, or how it is connected to the road. This information would have to be provided externally [Open Geospatial Consortium, 2012b].

3.1.4 Extending CityGML: Application Domain Extensions

Despite the availability of the thematic modules, in practice it is often necessary to store extra attributes which are not represented in CityGML, or objects that do not belong to any of the thematic classes already defined. There are two ways to extend CityGML in order to incorporate these extra features [Kolbe, 2009]. First, one can use the thematic extension module `Generics`. This enables one to add `GenericAttributes` to the `CityObject` core. These are name-value pairs which can be freely chosen and may consist of many data types (string, integer, double, etc.). Because they are defined as attributes of the `CityObject`, all thematic subclasses also inherit these attributes. In `Generics` one can also make `GenericCityObjects`, where the class, function and usage can be defined by the user. Geometries can be assigned to an LoD of choice.

Using generics has some downsides. CityGML is an XML-based standard and thus uses XML schema files (XSD files) to define the structure of the files. When one uses generics, they go outside the scope of the XSD files. Thus, it may for instance impossible to validate a certain CityGML file, or a parser may skip the added `Generics`. When added domain specific information is well structured, CityGML can also be extended by using ADEs. ADEs specify the extension of the CityGML data model in an XSD file and in a

UML diagram. These can include adding new properties to existing classes or defining new feature classes [Open Geospatial Consortium, 2012b].

Biljecki et al. [2018] have made an overview of CityGML ADEs and the developments surrounding them. ADEs have been part of CityGML since the beginning of the project. It has received much focus, and many ADEs have been developed for CityGML. Developement of ADEs has been application driven, and has stemmed from the need for specific semantic enrichment of the modelled objects. ADEs have enabled the CityGML data model to remain relatively "light" [Biljecki et al., 2018]. Another way to look at it is that ADEs add complexity to the data model. This can also be seen as a negative. A related downside of ADEs, is that often ADEs cannot be fully interpreted by software [Biljecki et al., 2018]. The added schema with its verbose XML structure means that existing software is not easily modified to incorporate parsing of the ADE.

3.1.5 Proposed additions to Transportation module

Adaptations to the Transportation module of CityGML have been proposed. This has been set out in Section 2.3.2. Here I will set out how this is supposedly modelled in the CityGML data model. Beil and Kolbe [2017] propose to change the LoD specification of the module. They propose to have only four LoDs (0-3) instead of five, for both the linear and areal representation of the transportation objects. Labetski et al. [2018] worked this out further. At LoD0 only the lowest LoD linear representation is present. LoD1 contains the same linear representation as LoD1, but also including an areal representation for the whole `TransportationComplex`. LoD2 has separate `TrafficAreas` for the different driving directions (carriageways) and its corresponding centre lines make up the lines. LoD3 has separate `TrafficAreas` for each driving lane and again the centre lines as the linear representation. Labetski et al. [2018] also introduce a new feature class `Section`. An aggregation of `Sections` can make up one of the four thematic subclasses `Road`, `Rail`, `Track` and `Square`. The idea of `Sections` is that a road can be modelled as one `TransportationComplex` while still detailing the changing attributes inside the road itself. Also, this makes possible that a section can be part of multiple roads.

Labetski et al. [2018] propose more changes. Whereas the linear representation in CityGML is modelled by using a GeometricComplex, how to do this is not specified. Also, it does not limit one to using just lines and points. Therefore it is proposed to explicitly model the linear representation by using just lines and points. This also makes the use of nodes explicit, which might be needed for applications which rely on a graph representation. They propose to make the use of nodes explicit from LoD1 upwards. They also propose to add stop lines, which indicate where a car should stop at an intersection. Intersections also should be modelled as a separate feature class, which describes its geometry. It can then be linked to its corresponding `TrafficAreas`, `AuxiliaryTrafficAreas`, stop lines and `CityFurniture` by using XLinks.

3.1.6 CityJSON encoding of CityGML data model

In Section 2.3.3, CityJSON was briefly introduced. CityJSON is a JSON-based exchange format for the CityGML data model [Ledoux et al., 2019]. A CityJSON file contains a CityJSON object, which represents a 3D city

```
{
  "type": "CityJSON",
  "version": "1.0",
  "CityObjects": {
    "id1": {
      "type": ...,
      "attributes": {
        ...
      },
      "geometry": [{
        "type": ...,
        "lod": ...,
        "boundaries": ...
      }]

    },
  },
  "vertices": [
    ...
  ]
}
```

Figure 3.4: Schematic overview of CityJSON structure

model. This 3D city model is a collection of features which are defined in the CityGML data model. The CityJSON object in the file has one member: "CityObjects". This member contains a collection of key value pairs, one pair for each object modelled, for a schematic overview see Figure 3.4. Here, the key is the id of the object, and the value contains the information of the `CityObject`. A CityJSON object also contains the member "Vertices", which is an array of coordinates of all vertices in the model. In the `CityObjects`, the vertices are not explicitly stored, but pointers are used where they refer to the index of the vertex in the "Vertices" array. This is in contrast to CityGML, where vertices are always explicitly stored [Ledoux et al., 2019].

The above is mandatory in each CityJSON file. A CityJSON file can additionally have members representing the metadata, the transform information or extensions [Ledoux et al., 2019].

CityGML has quite a hierarchical structure [Open Geospatial Consortium, 2012b]. The aim of the CityJSON encoding is to flatten out the data structure [Ledoux et al., 2019]. CityJSON differentiates between first-level and second-level `CityObjects`. Second-level objects are those that need to have a "parent" to exist. `CityObjects` have a member "Geometry", of which the value is an array containing zero or more geometry objects. If there is more than one geometry object, it is used to model the same object at different LoDs. `CityObjects` may have a member "Attributes", containing the attributes code listed by the CityGML data model.

Because of the flattened out structure of CityJSON, the class `Transportation Complex` is not present [CityJSON, 2019]. `CityObjects` are directly considered to be a `Road`, `Railway` or `TransportSquare`. `Track` is omitted as it can be modelled as a `Road` with a certain attribute. These three classes can subsequently be modelled with `TrafficAreas` and `AuxiliaryTrafficAreas`. This is done by using semantic surface objects. These semantic surfaces are the method with which CityJSON assigns the semantics to the different surfaces that model `CityObjects` [CityJSON, 2019]. A geometry object can have a member "Semantics", whose values are the properties "Surfaces" and "Values". "Surfaces" contains the different semantic types occurring in that geometry, and "Values" contains an array which has for each boundary representation (BRep) surface a pointer to which semantic surface belongs to

```
"ma_rue": {
  "type": "Road",
  "geometry": [{
    "type": "MultiSurface",
    "lod": 2,
    "boundaries": [
      [[0, 3, 2, 1, 4]], [[4, 5, 6, 666, 12]], [[0, 1, 5]], [[20, 21, 75]]
    ],
    "semantics": {
      "surfaces": [
        {
          "type": "TrafficArea",
          "surfaceMaterial": ["asphalt"],
          "function": "road"
        },
        {
          "type": "AuxiliaryTrafficArea",
          "function": "green areas"
        },
        {
          "type": "TrafficArea",
          "surfaceMaterial": ["dirt"],
          "function": "road"
        }
      ],
      "values": [0, 1, null, 2]
    }
  }]
}
```

Figure 3.5: Example of an areal Road object in CityJSON, adapted from CityJSON [2019].

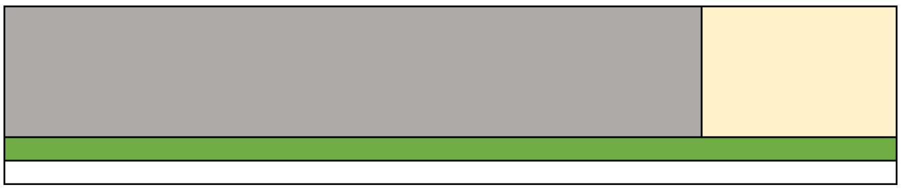


Figure 3.6: Figure showing a road with different surfaces modelled as semantic surface objects, corresponding to Figure 3.5.

it [Ledoux et al., 2019]. This modelling technique is efficient when surfaces are triangulated. In that case, many small surfaces that make up a MultiSurface or CompositeSurface will have the same semantics. By defining the semantic type once and using pointers to the surface array, this will results in less redundant data [CityJSON, 2019].

Roads are modelled with either a MultiSurface or a CompositeSurface. The semantics that make up the main geometry (`TrafficArea` or `AuxiliaryTrafficArea`, for example) are then thus specified per sub-surface. An example of what a road `CityObject` would look like in a CityJSON data file is given in Figure 3.5. A very schematic overview of what that road object could look like is given in Figure 3.6.

Note that the `CityObject` member "Attributes" relates to the whole `CityObject`, while additional attributes can also be defined in the semantic surface, as in Figure 3.5. This makes it possible to retain some semantic hierarchy in a flattened structure.

LoD4 is currently not implemented in CityJSON. This means that the inside of a Tunnel cannot be modelled yet.

```
<bldg:Building>
 <bldg:lod2Solid>
  <gml:surfaceMember>
   <gml:OrientableSurface orientation="-">
    <gml:baseSurface xlink:href="#wallSurface4711"/>
   </gml:OrientableSurface>
  </gml:surfaceMember>
  ...
 </bldg:lod2Solid>
</bldg:Building>
```

Figure 3.7: A surface of a `Building` object is modelled using an XLink [Open Geospatial Consortium, 2012b]. It refers to the object ID of another polygon, defined elsewhere in the data file.

3.1.7 Topology in CityGML and CityJSON

For many applications of 3D city models, having a topologically correct model is important [Kolbe, 2009]. In CityGML, it is possible to construct a topological relationship using XLinks. With XLinks, one object references another object in the header of an object. The XLink type specifies what type of relationship the objects have. A small example is shown in Figure 3.7. In practice, this is seldom used [Ledoux, 2019].

CityJSON does not currently provide a standardised way to reference other objects. In some instances, with second-level CityObjects, a parent-child relationship is specified. CityJSON does make it easier to construct topologically correct 3D city models. In CityGML, every geometry has its own list of coordinates. This can lead to topological errors. In CityJSON, vertices need to be stored only once, and can be reused by different geometries [Ledoux et al., 2019]. However, there is no method for constructing a graph from distinct lines and points. There are `CityObject` classes that permit these geometric primitives, but a method to link these has not yet been developed [CityJSON, 2019].

3.1.8 Extending CityJSON

In contrast to CityGML, one can add attributes or features to CityJSON files without documenting this in schemas [Ledoux et al., 2019]. The CityJSON validator, part of CityJSON In/Out (cjio), a Python command line interface for CityJSON files, will not consider the file invalid, but it will return a warning. This is part of the "schema-less" philosophy of CityJSON. However, these additions to the data model may also be documented formally, such that it will not trigger a warning. Where CityGML can be extended using ADEs, CityJSON uses Extensions to document how the data model is altered. This is done by using JSON Schemas. These schemas also double as a tool for validating your extensions. A CityJSON extension file is a separate JSON file of which the most important members are "extraAttributes", "extraCityObjects" and "extraRootProperties". In "extraAttributes", the schema should specify for which class the attributes are meant, the name of the attribute (which should start with a "+") and the possible values and data types of the attribute object [CityJSON, 2019].

In "extraCityObject", new `CityObject` types can be defined. All `CityObjects` are defined in the schemas of CityJSON, and because CityJSON does not do inheritance, if one wants to extend a `CityObject`, the schema of that `CityObject` has to be copied into the schema of the new one. Also, it needs to be stated explicitly whether it is a first-level or second-level `CityObject` [CityJSON, 2019].

In "extraRootProperties" information can be stored at the root of the CityJSON document. If one has data which pertains to the whole city model (for example, the city model covers one neighbourhood and you want to add neighbourhood statistics), here this data can be added [CityJSON, 2019]. A further elaboration on the way these schemas are structured is found in Section 4.2.

3.1.9 Shortcomings in data model and encodings

In this section I have reviewed how roads and related objects are explicitly modelled using the CityGML and CityJSON encodings of the CityGML data model. This was done to get familiar with the standard, and to be able to compare the use case data needs – that will be identified in the next chapter – to the way they are currently modelled in the CityGML data model. This review has also resulted in some identified shortcomings in modelling roads and their related objects, summarised below.

- Roads use bridges and tunnels. There is currently no explicit way to model a link between a road surface and the corresponding bridge and tunnel surfaces.
- It was established that the LoD specification for roads is not sufficient. The improved LoD specification according to Beil and Kolbe [2017] and Labetski et al. [2018] is not yet incorporated in the data model.
- It is not yet possible to model graphs. The concept of nodes and links is not present in the data model. CityJSON developers have expressed the desire to implement this at a later stage [CityJSON, 2019].
- The road data model lacks a concept of road segments, or sections. These can be used to aggregate short pieces of road into larger road objects in different ways.
- It is currently not possible to model intersection stop lines in the data model.

When improving the data model in Chapter 4, these shortcomings can also be considered, along with the use case data needs identified below.

3.2 USE CASE: TRANSPORT AND TRAFFIC MODELS

In transportation and traffic models real world objects are modelled which interact to resemble and simulate as closely as possible the transportation and traffic processes [Tamminga, 2019b]. These models abstract objects like people, trip destinations like houses and addresses, and different types of activities. These activities cause a demand for transport services as people have activities at different locations. The key to traffic modelling is to balance this demand with the supply of transportation services in these locations. The result should be a model which shows how the activities model throughout the transport infrastructure [Tamminga, 2019b]. Transport and traffic models exist for all modes of transport, however here I will mostly consider those that concern cars.

Transport models are identified as a use case of 3D city models [Biljecki et al., 2015], but also as needing two-dimensional data [Labetski et al., 2018] and road networks [Werkgroep Wegen, 2019b; de Dios Ortúzar and Willumsen, 2011]. Research done by Tamminga [2019b] contains an analysis of road data needs of transport models. This section is partly based on this analysis, and also on a meeting I had with the author [Tamminga, 2019a]. de Dios Ortúzar and Willumsen [2011] also list data needs for road network attributes.

3.2.1 Road data needs

Transport and traffic models can exist at various levels of detail [de Dios Ortúzar and Willumsen, 2011]. They can roughly be divided into two LoD groups: microscopic models, and meso- and macroscopic models [Haubrich et al., 2014; Tamminga, 2019b]. Microscopic models consider individual traffic units, and every individual's behavior is modelled separately. Meso- and macroscopic models aggregate traffic units and behavior of group to various degrees. Both groups of models mostly use graph-like structures to model traffic. Microscopic models mostly need networks where every lane has its own edge. For meso- and macroscopic models an edge per driving direction will suffice [Tamminga, 2019b].

The transportation network represents the supply of the transport model. It represents what the transportation system offers to satisfy movement needs of people that make trips [de Dios Ortúzar and Willumsen, 2011]. This is normally modelled with a directed graph, with nodes and directional edges linking the nodes. However, many models use undirected graphs, which deduce directionality from attributes [de Dios Ortúzar and Willumsen, 2011]. A data requirement for a edge-and-node model as described above is that it needs nodes at intersections, or when lanes merge or split. For intersections, this requires a representation of each turning lane, including the location of the stop lines. Each lane entering a junction has to have a connecting edge to each of the exit lanes that are reachable from that lane. In a meso- or macroscopic simulation, the network still needs to be topologically valid [Tamminga, 2019b]. At junctions, connecting edges still need to be put for each exit edge reachable. An example of what this could look like is given in Figure 3.8. In these models, characteristics of lanes might become attributes of edges (for example, the number of lanes per carriageway, or junction connector attributes in intersection nodes). It is also important to be able to determine routes between different points, but this is addressed in the next section, Section 3.3.

One can think of some applications where an areal representation may be needed. For example, parallel parking next to roads. Also, some intersections have a "stopping field" bicycle lane behind the vehicle stop line. Also, some bicycle lanes are not strictly separated from vehicle lanes, but share it (when there is a discontinuous lane marker). This raises the question whether the areal representation needs to be incorporated [Tamminga, 2019a].

Related to the above is the concept of the cross section. A vehicle in a lane needs to know whether it can change lanes. Therefore it needs to know if there are other lanes surrounding it, and also whether it is allowed to change lanes (what type of lane markers are separating the lanes?) [Tamminga,

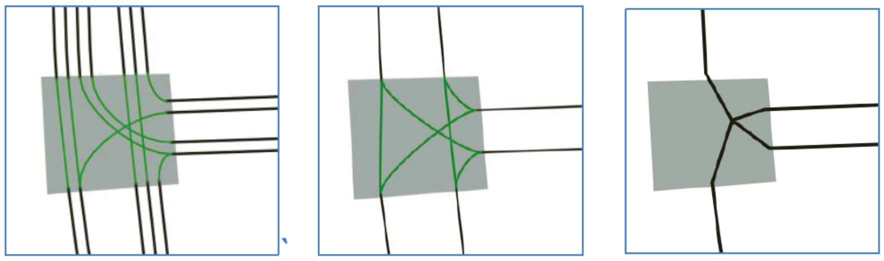


Figure 3.8: Modelling turning lanes of junctions at different levels of detail, by Tamminga [2019b].

2019b]. For this, vehicles need to know what the cross section of the road is. Cross section information is present at an LoD3 areal representation. If this information is not explicitly stored in the linear model, the linear model needs to be connected to the areal model.

Other physical road properties that are useful in transport models are speed bumps, width of the lane(s) and curvature of the road. Of course, also the maximum driving speed allowed on a road is important. These properties have an impact on the speed of drivers and therefore on the flow of traffic. Finally, at intersections and points where lanes merge or split, it is important to know which lane has the right of way [Tamminga, 2019b].

The data needs specified above are listed in Table 3.1, along with whether they are already modelled in the CityGML data model, and how they are modelled in other road standards.

3.2.2 Data needs in CityGML & CityJSON

As mentioned before, CityGML currently only supports a linear representation of roads at LoD0, and that representation does not contain nodes. However, the need for the road network in the micro- and mesoscopic models do correspond to the LoD0.3 and LoD0.2 of the proposed extended CityGML transportation module [Labetski et al., 2018]. Because this linear LoD specification is not worked out yet, many data need attributes also are not modelled yet in CityGML, like lane width or curvature of the road.

Some attributes, although not modelled now in CityGML, could potentially be deduced from the areal representation. As said, cross section information is present at an LoD3 areal representation. If speed bumps or stop lines are added to the areal representation, they could also be deduced as attributes.

3.2.3 Data needs in other road standards

It is checked how the identified road data needs are modelled in other road standards. The full comparison can be found in Table 3.1. In this section I elaborate on the most relevant modelling ideas. Lane information is often added as an attribute to the road objects. This includes the width and height of the lane, but also cross section information. Sometimes this is stored as an array, with an entry per lane. Another way this can be modelled is by defining these attributes as an offset from a reference line or point. Road standards often have either a linear or an areal representation, thus a link between the two was not found in the road standards. Therefore geometric attributes are always stored in linear features. Splitting or merging lanes are

Data need	**CityGML & CityJSON**	**GDF**	**OpenDRIVE**	**LandInfra**	**RoadXML**	**OSM**	**ITF**
Network for micro and meso	LoD specification proposed	Lanes not modelled separately	Lanes specified in relation to ref. line	Not modelled	No, lanes are attributes of the Profile	Lanes modelled separately when separated	Every lane modelled separately
Lane information	Not modelled	Attributes of Road-Element	Very detailed lane information	Not modelled	Extensive in Profile	# lanes as attr., lane info as sub-attr.	Attributes in Lane-Set attributes
Stop lines	Proposed for areal representation	Relationship between intersection and roads	Not modelled	Not modelled	Deduced from stop sign	Nodes have attribute highway = stop	Node on ConflictArea border
Merge / split lanes	Not modelled	Modelled at lower level with node	In lane information	Not modelled	Deduced from Profile	# lanes as attribute	Split / merge node
Explicit turning lanes	Proposed for linear representation	Not modelled	Yes	Not modelled	Not modelled	Turning lanes only as an attribute	Yes, modelled in the ConflictArea
Cross section	Can be deduced from high LoD areal rep.	Not modelled	Defined in Road Lanes Record w.r.t. ref. line	RoadElements have elaborate Cross Sections	Extensive in Profile	Can be derived from lane information	Not modelled
Speed bumps	Not modelled	Not modelled	Not modelled	Not modelled	Deduced from traffic sign	Nodes can have traffic_calming = speed bump	Not modelled
Curvature	Not modelled	Not modelled	Ref. line may be curve	Not modelled	Lines can be modelled as curves	Not modelled	Not modelled
Giving way	Not modelled	Relationship between intersection and roads	Not modelled	Not modelled	Deduced from traffic sign	Highway = priority is a tag	Not modelled
Link between rep. types	Not modelled	Not modelled	Not modelled	Not modelled	Not modelled	Not modelled	Not modelled

Table 3.1: Transport model data needs in other road standards.

sometimes modelled with nodes (when lanes have their own linear feature) or as a change in the lane attribute information.

The proposed LoD specification for networks by Beil and Kolbe [2017] makes room for the explicit modelling of turning lanes. How intersection should explicitly be modelled has not yet been specified. OpenStreetMap uses nodes to model intersections. Lanes are not modelled separately, but are attributes of ways. Turning restrictions in the nodes are used to determine which way one can turn. ITF and OpenDRIVE both explicitly model turning lanes inside an intersection area. Such an area is a polygon, where the links within determine what the intersection looks like and how it behaves. Stop lines are often modelled using nodes, while way giving information is handled differently per standard. GDF models a relationship between the intersection and the incident roads, while the OSM data model admits a priority attribute to ways.

3.3 USE CASE: NAVIGATION

Automotive navigation is one of the most well-known and widely used applications of geographic data. Navigation relies heavily on the geometric and semantic modeling of spatial data [Zhang and Ai, 2015]. With navigation, one uses this data to determine how to get from a certain location to another location. Navigation systems help driver to plan trips, create routes and aid them trying to follow that route by giving instructions [Egenhofer, 1993]. The objects modelled in these systems contain roads and intersections, but also landmarks along roads and information on signage [Egenhofer, 1993]. The topological structure of the road network itself can be represented by a graph [Thomson and Richardson, 1999], where edges represent the roads between pairs of vertices. Nodes here can function as intersections, a lane merge/split, a dead-end of a road, or perhaps places of other interest. By inserting a node in a road segment where an attribute changes, nodes can be used to model these changes such that every road segment has unique attributes over its whole geometry [Thomson and Richardson, 1999].

3.3.1 Road data needs

Vehicle navigation systems have a front-end component, where the user can interact with the system and it will give directions, and a back-end component which does the routing based on different kind of preference settings [Mapbox, 2019]. For the road data needs I am mostly interested in the latter. Mapbox [2019] and Claussen et al. [1989] both mostly identify the same data needs in road geometry and attributes, outlined below.

As navigation systems have to tell you how to conduct your route, the first main requirement of the navigation system is that it has knowledge of the topology of the road network. Thus, the road network needs to be represented as a graph. In order to identify the location one wants to reach, one often enters the destination address into the navigation system. Therefore, it is important to have the street name as an attribute of the road segments. To differentiate between different roads with the same name, the municipality can also be considered as an attribute. Also, the navigation system needs to know what roads are meant for, or accessible to, cars. Therefore, the modes of transport allowed on a road segments can also be an attribute.

In routing, often one can choose between different path options. For instance, one can choose the shortest route or the predicted fastest route. Apart from maximum speed, another classification that influences these routes is the road classification. Administrations often classify roads in a hierarchy, with terms like motorway, highway, local road etc. Routing algorithms can make use of these classifications to estimate what road segments might yield the fastest travel time.

Finally, in routing it is important to know which way you are allowed to go. Therefore, you need to know the driving direction of the road (whether it is one way or two way for instance). At intersections, any possible turning restrictions need to be modelled. That is, if I am not allowed to turn left at an intersection, the navigation system should be aware of that. This is also a important for transportation models [de Dios Ortúzar and Willumsen, 2011]. Furthermore, roundabouts are somewhere between an intersection and a road. The characteristics of roundabouts are unique. For navigation they need an explicit way of modelling.

As said above, navigation systems sometimes use landmarks for user orientation purposes. For this, the CityGML data model could be useful, because the surroundings are also modelled and could be extracted from it. Also, road signs are sometimes present in navigation systems.

Tavares et al. [2009] conducted research in implementing a 3D geographical information system (GIS) for the collection of municipal waste. Here, they used the third dimension to extract the gradient of the road. This gradient was then use in routing algorithms. Waste collection trucks are heavy and will also carry the waste. Thus the gradient has a big effect on the cost-effectiveness of the route choice.

3.3.2 Data needs in CityGML & CityJSON

As already mentioned in Section 3.2, a graph structure for CityGML roads has been proposed, but it is not developed yet. Currently, at LoD0 the roads are modelled by using just lines, without any connectivity. In CityGML there is the option to use Xlinks to make connections between objects. In practice however, this is seldom used, and also not worked out well. Also in CityJSON the possibility to create networks is not supported as of yet.

`TransportationComplexes` are sub-classes of the root class `CityObject`. From here it inherits the attribute "gml:name". This attribute can be used to model street names. However, it does not have fixed attributes for street names or municipality names. CityJSON does not support inheritance, therefore the only thing that can be used for identification is the object id. However, because a road with a fixed name may consists of separately modelled parts, this is not suitable for a road name.

The modes of transport that can use a road can be modelled using the Usage attribute that `TransportationComplexes` inherit. The OGC provides a code list of usages [Open Geospatial Consortium, 2012b], but one can add these themselves. Again, in CityJSON inheritance does not exist. Therefore this road usage can be added as an attribute. Driving direction, turn restrictions, roundabouts, slopes and road classifications are not modelled. Road sign is a value in the CityGML `CityFurniture` code list.

Data need	**CityGML & CityJSON**	**GDF**	**OpenDRIVE**	**LandInfra**	**RoadXML**	**OSM**	**ITF**
Graph structure	Proposed, currently not modelled	Yes	No nodes, but graph through adjacency	No	Yes, startNode to endNode	Nodes and Ways provide graph	Sequence of nodes
Address information	Could be deduced from Buildings nearby	Attribute of RoadElement	Not modelled	Objects can have address as attribute	Not modelled	Address tag for all objects	Not modelled
Modes of transport allowed	Usage in CityGML, not present in CityJSON	Attribute	Bicycle lanes, pedestrian lanes, no restrictions	Not modelled	Lanes have VehicleType attribute	As a tag for all road objects	Not modelled
Driving direction	Not modelled	Attribute of RoadElement	Attribute of Roads and Lanes	Not modelled	Attribute CirculationWay	Boolean tag for being one way road	IngressApproach or EgressApproach
Turn restrictions	Not modelled	Attribute	Turning lanes explicitly modelled	Not modelled	Only on traffic sign information	Can be explicitly modelled for every node	Determined by lack of turning lane
Roundabouts	Not modelled	Yes, modelled as a circular road	Group of Junctions, not related further	A Road subclass	As LaneType attribute	Tag junction = roundabout possible	Not modelled
Road classification	Not modelled	Attribute	RoadType attribute provides some classification	Yes, motorway, carriageway, highway, etc.	Road attribute PriorityLevel	Highway key has hierarchical values	Not modelled
Slope	Can be deduced from areal geometry	Attribute	Can be deduced from ref. line	May be deduced from geometry	Not modelled	Include tag possible for ways	Can be deduced from Node altitude
Road signs	Can be modelled as CityFurniture	Linear referenced along network	Not modelled	Not modelled	Extensive information	Their use modelled in Nodes	Not modelled

Table 3.2: Navigation data needs in other road standards.

3.3.3 Data needs in other road standards

The comparison of the navigation data needs with how they are modelled in other road standards can be found in Table 3.2. Road standards that employ graph data structures model these in different ways. A common method is to specify a start and end node for each linear road segment. OpenDRIVE does not use nodes, but specify adjacency of road segments. ITF only models a sequence of nodes. Note that this is only possible for road segments, and intersections cannot be modelled in this way. Strictly speaking the latter two modelling methods are not graphs, since they only represent either nodes or edges. However it does model the connectivity.

Road classification, driving direction and transportation modes allowed are always attributes of road segments. The latter is sometimes split into a road type (road, bicycle path, pedestrian path) and vehicle restrictions (no trucks allowed) for easy identification of different types of infrastructure. Turn restrictions, also identified in the previous section, can be modelled either as an attribute of an intersection or by explicitly modelling turning lanes. Roundabouts are often modelled as a separate class, or have an attribute identifying them as such. Sometimes they are also simply modelled as a one way circular road with many junctions. Slope is almost never modelled, but can often be deduced from three-dimensional geometry. If information on road signs and addresses is modelled, it is probably linearly referenced to the road data. In OpenStreetMap, road signs can be modelled as nodes as well.

3.4 USE CASE: ROAD MAINTENANCE

Road maintenance is carried out by the administration that is responsible for the road. Spielmann and Scholz [2005] identify road marking, road lighting, weed control and de-icing of roads as parts of road maintenance. Road de-icing happens after it has snowed, or when it freezes. Soetens and Tijink [2019] conducted a study into the data needs for automatic dispensing of salt by salt trucks in the Netherlands. At this moment, salt truck routes are determined beforehand, and the truck drivers manually handle the salt dispenser. They can determine how much salt is dispensed, and how wide. The goal of the research is to work towards a system that automatically makes the routes and determines how much salt (and how wide) is dispensed based on road characteristics along the route. This automation has two benefits. First, it is considered to be safer, as the truck driver does not have to check the route and handle the salt dispenser simultaneously. Secondly, it is better for the infrastructure and the environment, as there is less chance of dispensing too much salt on the roads. This can lead to defects in infrastructure and surrounding buildings [Soetens and Tijink, 2019].

Ozbek et al. [2010] have made an inventory of variables that influence the need of maintenance of bridges. These include variables like cost, which are unrelated to this research. However, they also include variables based on geometric and semantic attributes of the bridge, like road deck type.

Road maintenance and road de-icing in particular have been identified as a road use cases needing both a network representation and an areal representation [Werkgroep Wegen, 2019a]. Thus by checking the road data needs for this use case, I want to check how the two representation types can co-exist and perhaps be linked. The following road data needs are based on an attribute catalogue determined by Soetens and Tijink [2019] and the variable listing by Ozbek et al. [2010].

3.4.1 Road data needs

As with the other use cases mentioned in this research, routing is important for road de-icing [Soetens and Tijink, 2019]. Thus again the road network needs to be modelled as a graph. Another important data need is lane geometry. Trucks are wide vehicles, thus the lane width but also height is important. Lane height is the maximum height a vehicle can have such that it can safely drive on that lane. This height may be limited by bridges, for example. This lane geometry includes the road surface material. This greatly influences the way ice forms on the road. Ozbek et al. [2010] also identify surface material and thickness as a factor in maintenance.

For road de-icing, it is also important to know whether the road segment is part of a bridge or tunnel. This influences the amount of salt that should be dispensed. Furthermore, the surroundings of the road are important. When housing is close to the road, this should limit the amount of salt that is being dispensed. Also obstacles like traffic islands, and related road parts like bus stops and passing lanes should be taken into account [Soetens and Tijink, 2019].

An attributes that is important is who is responsible for the maintenance of the road. Most of the time this is one of the layers of government, be it local of national. Road classification is again important, as 'more important' roads will probably have a higher priority [Soetens and Tijink, 2019; Ozbek et al., 2010].

The height of the road itself is also something to take into account. High altitude can lead to lower temperatures, and thus it might influence de-icing algorithms [Soetens and Tijink, 2019].

3.4.2 Data needs in CityGML & CityJSON

Lane width can be determined from the areal geometry of the lane. Lane height could theoretically also be extracted, when a road passes underneath a bridge or some other object. Realistically this would be very challenging. This is not an attribute that is modelled in CityGML. The altitude of the road segments are modelled in CityGML; after all it is a data model for 3D city models. Obstacles, bus stops, and surrounding housing can all be extracted from surrounding objects in the 3D city model.

In CityGML, there is an attribute "surfaceMaterial", which again refers to a code list. This attribute is also modelled in CityJSON. Who is responsible for the maintenance of the road is not a modelled attribute, this could be added however.

3.4.3 Data needs in other road standards

The comparison with other road standards can be found in Table 3.3. Many of the data needs identified in this section are modelled as attributes. Road surface material, lane information, road administrator, and whether it is part of a bridge or tunnel can all be modelled as attributes of road segments. When roads are modelled in three dimensions, the altitude can be deduced.

Data need	**CityGML & CityJSON**	**GDF**	**OpenDRIVE**	**LandInfra**	**RoadXML**	**OSM**	**ITF**
Road surface material	SurfaceMaterial attribute for areal representation	Attribute of RoadElement	Attribute of lanes	Attribute of RoadElement	Surface materials are modelled with Ground type	Ways have surface tag	Not modelled
Lane width	Can be extracted from geometry	Attribute of RoadElement	Attribute as offset from reference line	Road has ApproximateWidth attribute	Information in Lane in Profile	Lane specific key after # lanes is stated	Attribute of the lanes
Lane height	Not modelled	Attribute of RoadElement	Not modelled	Not modelled	Part of lane attribute VehicleType	Lane specific key after # lanes is stated	Attribute of the nodes
Administrator	Not modelled	Attribute of RoadElement	Not modelled	Not modelled	Not modelled	Not modelled	Not modelled
Road altitude	Inherently part of its geometry	May be modelled	In relation to ref. line	Can be deduced from geometry	Lines are modelled in 3D	Elevation tag for Nodes, not for Ways	Ref. point of the intersection may contain z-value
Road obstacles	Can be modelled with AuxiliaryTrafficArea	May be modelled by an area	Not modelled	Not modelled	Can be modelled adjacent to road	Traffic_calming = island tag for Areas	Not modelled
Bus stops / passing lanes	CityFurniture or Transportation-Complex	Bus stop modelled as a Stop Point	Lanes with Type attributes	Not modelled	Not modelled	Highway = bus_stop for Nodes, lanes otherwise	Not modelled
Part of bridge or tunnel	Could be connected to OuterFloorSurface	Attribute	Attribute	Bridges and tunnels separately modelled	Profile has attributes Bridge and Tunnel	Separate Tunnel and Bridge keys	Not modelled
Proximity of housing	Can be deduced in 3D city model	Distance to Addressed area can be determined	Not modelled	May be deduced from distance to building objects	Not modelled	Can be deduced from houses in the data	Not modelled

Table 3.3: Road maintenance data needs in other road standards.

3.5 USE CASE DATA NEEDS SUMMARY

After analyzing the shortcoming in CityGML and CityJSON and checking the data needs of the three use cases above, a list of data needs can be assembled. Some of the data needs are already modelled in the CityGML data model. The others are outlined below. The data needs are grouped in categories. This categorization of data needs in then used in the next chapter to specify the design choices for improving the data model.

LoD specification. Use the improved LoD specification by Beil and Kolbe [2017] and Labetski et al. [2018] as a starting point for improving the data model. This includes the road-carriageway-lane differentiation for respectively LoD(0.)1, LoD(0.)2 and LoD(0.)3.

Graph structure. Implement a graph structure such that LoD0.1 up to LoD0.3 can be modelled as a network. This includes having a node when lanes merge or split.

Attributes. Attributes that need to be added to roads are: allowed vehicle types, road classification, driving direction, administrator, maximum speed. Other attributes that have to do with its geographic configuration are lane width, lane height, cross section, slope, surface material, road curvature. Also road features that might have an impact on driving can be included, like speed bumps.

Road segments and linking representation types. Introduce the concept of road segments such that small parts can be aggregated into bigger road objects. Related to this is the ability to link linear and areal representations.

Intersections. Intersections and roundabouts need explicit modelling, including specific turning lanes, turn restrictions, way giving information and stop lines.

Connecting to other modules. Link `Road` surfaces with `Bridge` surfaces when a road is on a bridge. The same applies to roads in tunnels. Establish whether a link can be made with relevant roadside information like addresses, road signs and traffic lights.

In the next chapter, the importance of these data needs will be discussed. After, I will try to improve the CityGML data model such that these data needs are satisfied.

4 IMPROVING THE DATA MODEL

In this chapter I want to answer the question: how can the CityGML transportation data model be improved such that it satisfies the use case data needs? In Chapter 3 the road data needs of three use cases were determined. It was checked how these needs are currently modelled in the data model and how they are modelled in other road standards. Now choices have to be made on whether, and if so, how, these data needs will be modelled.

In Section 3.1 the CityGML data model and the encodings in both CityGML and CityJSON were explained. Besides wanting to improve the data model, I want to encode these changes in a schema. This way data files adhering to the improved data model can be created and validated. In this thesis I choose to encode the data model in CityJSON. The flattened structure that the JSON encoding provides makes it easier to implement changes [Ledoux et al., 2019]. Furthermore, CityJSON is developmentally in an early stage. At the time of writing, the first version has just been released. The CityJSON developing community highly encourages users to propose changes and additions. The need for a network structure was also identified CityJSON [2019]. The above motivated me to use CityJSON as the CityGML data model encoding.

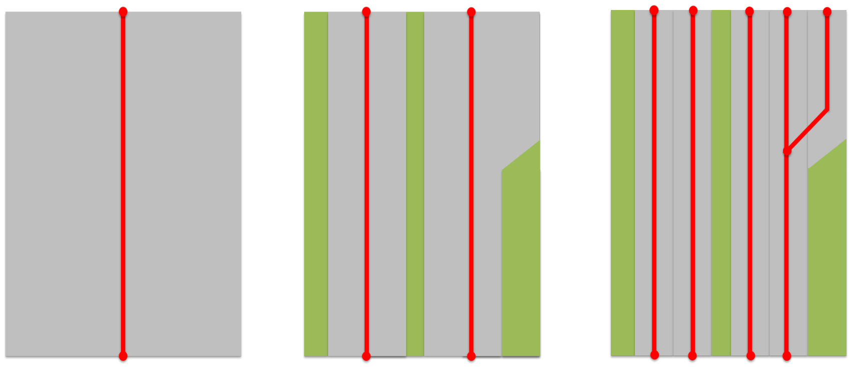
In Section 4.1 the design choices for improving the data model are presented and motivated. Section 4.2 contains a technical exploration of how JSON files (and by extension, CityJSON files) can be modelled using JSON schemas. This knowledge is then used to explain and adapt the current CityJSON schemas. Finally, Section 4.3 contains the results of the improved data model as an LoD specification. Here it is explicitly shown how roads and intersections can be modelled at different LoDs. In Chapter 5, this new LoD specification will then be used to create a CityJSON road file that adheres to this improved data model.

4.1 DATA MODEL DESIGN CHOICES

In this section I will discuss the considerations made towards the data needs identified in the previous chapter. I will discuss whether they should be addressed in the improved model and if so, how this will be encoded in CityJSON.

4.1.1 LoD specification

The starting point of this research was the research already conducted on enhanced LoD specifications of the CityGML transportation module [Beil and Kolbe, 2017; Labetski et al., 2018]. Thus my proposal is to model roads in a linear and areal representation according to the LoD concept set out in Section 2.3.2. This means that at LoD(0.)1 the road is modelled as one piece, at LoD(0.)2 I specify by carriageway, or driving direction and at LoD(0.)3



(a) LoD(0.)1 (b) LoD(0.)2 (c) LoD(0.)3

Figure 4.1: Overview of implementation of road LoDs.

each lane is modelled separately. An overview is given in Figure 4.1. A benefit of this LoD specification is that data providers can choose at what LoD they offer their data. This could be dependent on the accuracy of their data gathering techniques, for example.

Zhang and Ai [2015] identify modelling issues for dual carriageways in OpenStreetMap. Users are often not consistent in the way they model this. This is also important for modelling this LoD specification. Regarding carriageways and lanes, there are different ways to model this. At LoD0.2, one could always make an edge for each driving direction possible (two edges for a two-way street), or one could only do that when the two driving directions are physically separated, as in carriageways. Sometimes these two situations happen sequentially, how does one model this consistently? This point was already discussed in Section 2.3.2 and Figure 2.6. Here, I make the choice to allow the user to model "downwards". For example, when a narrow road is bi-directional and doesn't have separate lanes, it is allowed to model this at LoD0.3 with only one edge. This edge then does need to have information on the directionality and the number of lanes.

4.1.2 Graph structure

From the use cases analysis it is clear a topological graph structure is needed. As seen in Section 2.1.2, there are different ways to store graphs: edge lists, adjacency lists and adjacency matrices. In terms of storage and efficiency, adjacency lists seems the best choice. However, I would like to implement this graph structure in CityJSON. Therefore the possibilities of modelling a graph in CityJSON should be taken into account. As seen in Section 3.1, in CityJSON geometries are stored as arrays of pointers to vertices. When roads are considered to be edges represented by (Multi)Linestrings, then each of these geometries represents an edge. Thus there already is something akin to an edge list: CityJSON facilitates an edge list graph-like structure because of the pointers to the vertices list.

These edges also need to be linked together. There are different ways to do this. The OpenDRIVE standard refers to a predecessor and a successor edge [Dupuis, 2015]. However, other standards opt for a structure where the edges are connected using nodes. In this data structure, nodes point to the incident edges and edges point to the incident vertices. The main reason to use nodes seems to be because this is an object which can be used for attribute changes in road segments. As discussed in Chapter 2, attribute changes in networks can be modelled by starting a new road segment (by adding a node), or through linear referencing. The main benefit of using nodes of these attribute changes, is that it doesn't require one to have an added structure of a linear referencing system. Therefore it seems that modelling attribute changes with nodes is more beneficial. Also it seems that when CityJSON road data files will be used not only for storage, but also because one wants to perform an application on it, then additional systems will be used anyway. Thus it is decided to pick a node-based model but to accommodate the external addition of a linear referencing system.

An important benefit of having a graph structure is that networks can be used by more classes than just `Transportation`. The class `WaterBody` might for instance use the graph structure to model a network of waterways. This possibility of multiple use of graphs is important for the implementation. It leads to having graphs as a base structure in the data model. This is achieved by adding new `CityObject` types: `Nodes` and `Edges`. These new classes can then be extended for the required class, in our case roads. In CityGML this might be modelled by inheritance. Here I will extend the new classes by using a CityJSON extension, see Section 4.2. These extended classes will be called `RoadNode` and `RoadEdge`.

4.1.3 Modelling attributes

Many data needs that arose from the need analysis can be added as attributes to geometry. Some of these are only useful for one of the two representations, others can be useful for both. Also, some attributes concern the physical road configuration, which might be deduced from the geometry itself.

The concept of Application Domain Extensions in CityGML has made it possible to keep the CityGML data model itself relatively brief. Thus the CityGML modules facilitate the use of the model by many different applications. As the name indicates, ADEs are made to enhance the data model specifically for a certain application. In the previous chapter, I have identified the data needs of three use cases. If I add all these to the improved road data model, this will not correspond to the original idea of CityGML. Therefore, the idea is to extract geometrically and semantically from the data needs what should be added to the geometry, such that the data model facilitates the use of many other use cases.

The question that remains is: what attributes can be considered general enough to warrant a place in the data model? Attributes collected from the data needs assessment that do not concern geometric aspects are: allowed vehicle types, road classification, driving direction, administrator and maximum speed. Attributes that have to do with its geographic configuration are road/lane width, road/lane height, cross section, slope, surface material, road curvature. At first I planned to add all assessed attributes to a CityJSON extension. However, like ADEs, extensions are mostly meant to be used to facilitate one single use case. Related, the COR working group wants to move to a core set of attributes which are mandatory for every road object [Werkgroep Wegen, 2019a]. During a meeting it was decided that the attributes classification, vehicle restriction, maximum speed and driving direction should always be present, given that these attributes are used for many applications. I have decided to follow this approach. These attributes are mostly used in road networks. Therefore I add them as attributes of `RoadEdges` in the data model.

The driving direction will be modelled in relation to the start and end node. Thus there will be three possible values: from start to end node, from end to start node, and both ways. Vehicle restriction can often be deduced from the road classification attribute (like highway, or bicycle path). This is not always these. Therefore I also add an attribute `modes`, which may additionally specify what modes are allowed.

I have chosen not to model the attributes related to geometry. Most of these attributes can be deduced from areal geometry, when this is modelled. "SurfaceMaterial" is already an attribute of the roads in CityGML. Road / lane width, slope or restricted height can be added to the linear representation in an extension if one wants to model this. Again, given that CityJSON is "schema-less", this is not necessary. The attributes can also be added to the objects without becoming invalid. Cross sections are more complicated. Some use cases need road cross section information to determine whether one can change lanes. This information can become quite specific, as seen in Section 2.4. In theory, this information can be deduced from LoD3. This might be hard to implement practically. Further research needs to be done in how to model cross section information in CityGML.

4.1.4 Road segments and linking representation types

Another data need that arose was the ability to make a connection between the linear and areal representation. In Section 3.2 it was concluded that none of the considered road standards model an explicit link. The main benefits of linking a linear representation of a road segment with its areal representation is that one can combine the data in analyses, and that they can be kept consistent [Werkgroep Wegen, 2019a]. Linking of representation types is also beneficial when one wants to combine the road network with an analysis of other objects in the 3D city models, as the areal representation directly borders neighbouring features [Biljecki et al., 2015]. However, linking representation types has several drawbacks. It would only make sense if the two representations linked have the same respective level of detail in their representation. Otherwise the geometries would maybe not intersect. This would mean that the two datasets should align in their LoD modelling. A bigger drawback is that when the linear network is modelled as a graph with attribute changing nodes, this might result in a highly fragmented network. Making a one-to-one mapping between a highly fragmented network and an areal representation is theoretically possible, but practically will result in an also highly fragmented areal representation nobody might want. Linked representation types can be useful however. As mentioned above, when one wants to have lane information concerning their geometry, like lane width or information about the cross section of the road. When these representations are not linked, this has to be become an attribute of the `RoadEdges`. Another option in an application would be to link the two geometries on the fly, provided that the two geometries intersect.

CityJSON is an object-oriented data format. Data files consist of a sequence `CityObjects` which together make up (a part of) a 3D city model. This object-oriented approach makes that CityJSON has the ability to link representation types already built in. Each `CityObject` can have an array of geometry objects. These are meant to store different LoDs of the same object in one file. Therefore when we have two representation types, they can thus both be modelled with two different geometry objects in one `CityObject`. However, as said above, this might cause issues because a network could be

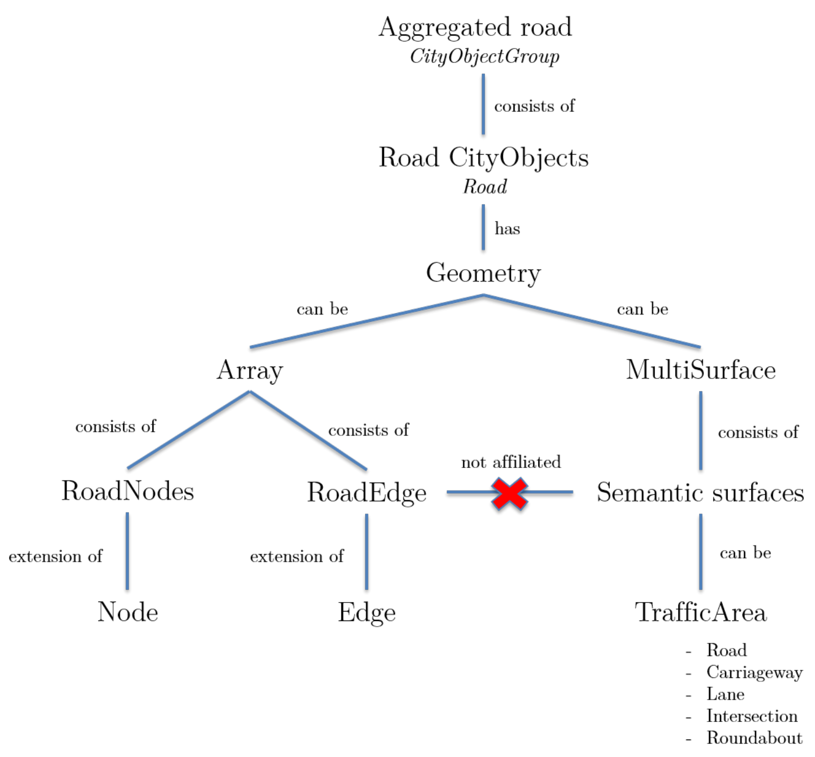


Figure 4.2: Schematic overview of how road objects are modelled.

segmented in small parts while road surfaces might be modelled as large areas. This would lead to having to split up both representation types in exactly the same manner, which would be a lot of work.

The same considerations arise in the discussion on the COR, as discussed in Section 2.5.3. The idea of the COR is that the object is central. It serves as the base of object modelling in governmental spatial data. The question then becomes: what is the scale of an object. For buildings, this is more intuitive. However, almost all roads in a country are interconnected, and it is not always clear where one begins and another one ends. The street name could be leading in determining what an object is. Intersections then however are parts of multiple roads. Combining this with using nodes for modelling attribute changes, it becomes difficult to determine what the object is. It can be a road segment with constant attributes. However this would seem an arbitrary base to define an object as the main governmental modelling object.

In Chapter 3 it was shown that it has been proposed to model sections [Beil and Kolbe, 2017], or road segments. These sections are smaller parts of roads which can be aggregated to form "proper" roads. This notion can be used as an answer to the object scale question. An object can be made up of smaller sub-objects, which each have different characteristics. In this way, the user can determine themselves what the scale of the object is. In the context of the dual representation modelling in CityJSON, the question is how to implement this notion of sections, or segments.

At first I tried to implement sections, and to model the areal and linear sections one-to-one. I only added `RoadNode` as a new object class. I changed the name of object class `Road` to `RoadSegment`. These road segments could then have multiple geometry values, for both the linear and areal repre-

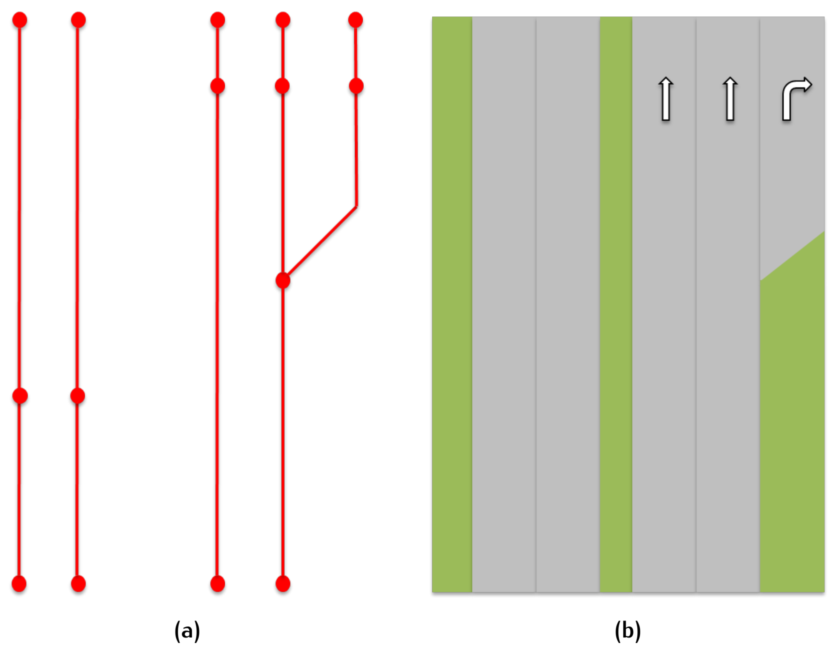


Figure 4.3: One Road object modelled in both representation types. Note that they are segmented differently.

sentations. The RoadSegments could then a aggregated to roads with an extension of the CityObjectGroup (which I called Road). In this scenario, the areal and linear representations are segmented identically. However, in Chapter 3 it was shown that the areal representation has less needs for different attributes. Most applications that need attributes, need a linear network. Thus, it is assumed that areal representations of roads can often be modelled in "bigger" segments. In addition, areal representations already have a notion of segments in the semantic surface objects. The CityObjects are divided in different areas with different properties (see Figure 3.6).

To accommodate the central object concept, it is chosen to model the Road in CityJSON as either the areal geometry consisting of multiple semantic surfaces, or as an array of RoadEdges. These RoadEdges are the edges of the graph with which the road network is modelled. This way, the same Road object can be modelled in two different representation types which are partitioned in different ways. Thus the implementation of the road segments, or sections, is done differently for the two representation types. This gives some flexibility to the dual modelling. A schematic overview of how roads are modelled can be found in Figure 4.2. What this would look like can be seen in Figure 4.3. It is chosen not to explicitly model links between the road segments and the semantic surfaces. This would negate the flexibility mentioned above. If one wants to explicitly link the two segments together, one can always use GIS software to join them together. In order to model bigger aggregated of road objects, the aggregation class CityObjectGroup can be used.

4.1.5 Intersections

From the use case analysis, there was a need to explicitly model intersections' turning lanes, roundabouts, stop lines, way giving properties among other things. All these needs relate to the linear representation. Tamminga

[2019b] recommends to model turning lanes explicitly for microscopic transport models, and per driving direction for mesoscopic models. This corresponds to our LoD0.3 and LoD0.2, respectively, see Figure 4.4. By putting the node that starts the turning lane at the location of the stop line, these can be modelled. Given that LoD0.1 modelled whole roads by a single `RoadEdge`, I choose to model intersections at this LoD with a single node. Furthermore, nodes are used at LoD0.3 to represent splitting or merging lanes, as in Figure 4.3a.

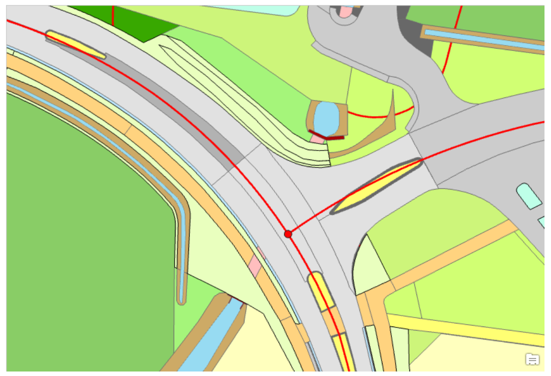
Roundabouts follow a similar configuration. At LoD0.1 they are modelled with a single node. At LoD0.2, they are modelled with roundabout lane types. The nodes become the points where the road meets the roundabout. Roundabout lanes always have a one way driving direction. In LoD0.3, roundabout are modelled the same as in LoD0.2. Only roundabouts that have multiple lanes going around the centre (common in the Netherlands), will be modelled differently, see Figure 4.4.

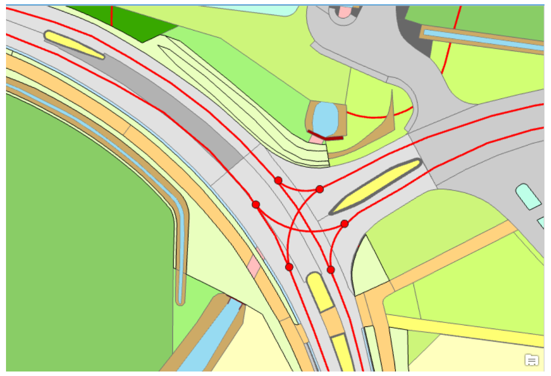
Thus, `RoadEdges` can have three different edge types: "Road" for regular stretches of road, "Connecting" for turning lanes, and "Roundabout" for roundabout lanes. The latter two always have a single driving direction. For routing, one needs to be able to model intersection turning restrictions. In LoD0.2 and LoD0.3 these are explicitly modelled by the turning lanes. In order to be able to model turning restrictions in LoD0.1 as well, I add a `RoadNode` attribute "turns". This is a two-dimensional array which specifies which incident `RoadEdge` can be reached from other `RoadEdges`. This can be considered an adjacency matrix as in Section 2.1.2, only now for edges instead of nodes. The indices of the matrix refer to the position of the edge in the `edges` list in the `RoadNode`.

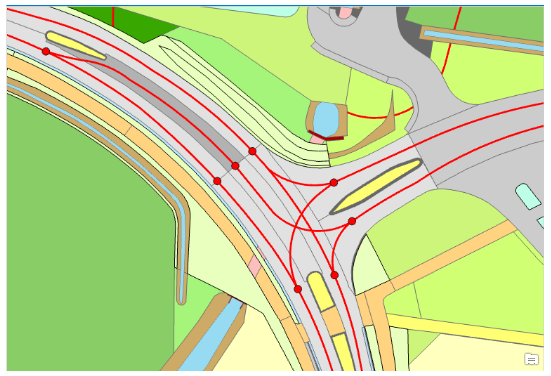
The use cases did not contain many areal intersection data needs, except that it is sometimes nice to know that there is one (for salt dispensing for example). The exact areal configuration of the intersection seems to matter less. Therefore I choose to add "Intersection" as a `Road` type, like "Carriageway" and "Lane" will be added for LoD2 and LoD3. The decision is made to model intersections as single surfaces. Therefore it may be best not to link network and areal representations for intersections. There will not be a one-to-one mapping with all the turning lanes. It might also not be very useful for any application. Instead, it is agreed to have the option to assign an id to an intersection, such that every linear `RoadEdge` road segment or turning lane can be linked to a certain intersection. This was already proposed by Labetski et al. [2018], by using XLinks in the CityGML encoding. This also makes it easier to link traffic lights, modelled with `CityFurniture`, to an entire intersection (see next section). This also opens up the possibility to generalise interchanges. At an interchange, roads pass each other at different grade level, but have exit and enter lanes to and from the other road. At the highest level of detail this is not considered an intersection, but just many lanes splitting and merging. By letting all these segments have a link to the same intersection id, this interchange can be easily generalised to a node that *is* an intersection at a lower LoD. It sometimes seems arbitrary whether two neighbouring intersection are the same or separate. Therefore it is allowed for objects to link to more than one intersection.

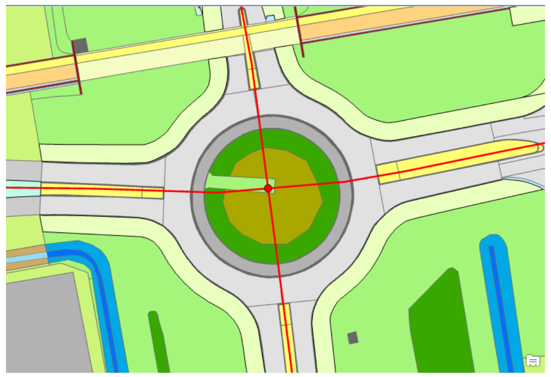
4.1.6 Connecting to other modules

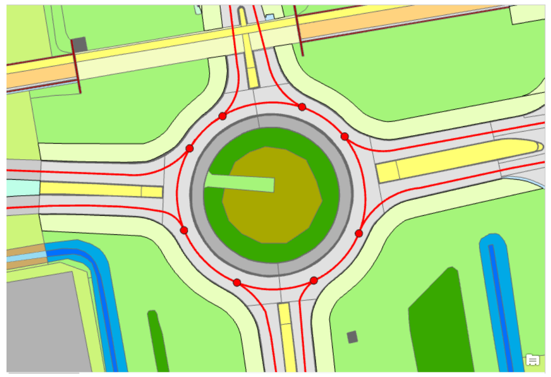
A data need that arose was to model a link between roads, and bridges and tunnels. For linear roads, this is not especially necessary. `RoadEdges` could


(a) LoDo.1 intersection


(b) LoDo.2 intersection


(c) LoDo.3 intersection


(d) LoDo.1 intersection


(e) LoDo.2 intersection

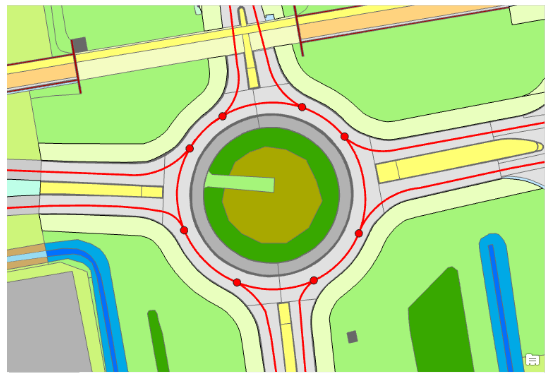

(f) LoDo.3 intersection

Figure 4.4: Linear modelling of intersections and roundabouts.

be enhanced with an attribute which specifies that it is part of a bridge or tunnel. In theory this could also be done for areal roads. However, this would mean that both the road and the bridge or tunnel would be modelled. This would lead to having overlapping geometries, as the road surface and the road surface of the bridge or road would be touching. However, I also do not want to for instance model the entire road in the Bridge module, or vice versa. Therefore I choose to be able to link a road surface to a bridge or tunnel surface. This "link" property can be added in all three modules. In CityJSON, the tunnel surface on which one drives is not modelled, given that it belongs to LoD4. Thus this link is not yet implemented for tunnels.

As said in the previous section, traffic lights modelled with `CityFurniture` can be linked to intersections through the new intersection id. How this will be modelled exactly is outside the scope of this research.

4.2 IMPLEMENTATION OF DESIGN CHOICES

In the previous section, choices have been made on how to translate the data needs identified in Chapter 3 into modelling concepts in CityJSON. In this section, it is explained how these are implemented technically. The CityJSON file structure is defined by JSON schemas. How these work is outlined in Section 4.2.1. After, I describe what has been changed in the schema in order to accommodate the data needs. The full set of updated schemas can be found on `https://github.com/fhb1990/cityjson`, which is forked from the original CityJSON repository.

4.2.1 JSON schema

What a CityJSON file should look like is determined by the CityJSON specifications [CityJSON, 2019]. These specifications are modelled in JSON schemas. JSON schema is a tool for validating JSON data file structure [Droettboom, 2019]. Thus the CityJSON schemas serve to assess the validity of a CityJSON data file.

JSON files are built out of objects (a sequence of key-value pairs), arrays, numbers, strings, booleans and a null value. From these data structures, complex structured data can be represented. When one wants to standardise certain data formats, it is important to know how exactly the data will be represented. This is what JSON schemas do for JSON files [Droettboom, 2019]. As JSON schemas themselves are JSON files, it has limitations. It can only declare the way the JSON data should be structured. It cannot parse the data and check whether it is semantically correct. Therefore, validation of files should be extended with a semantic validation, outside of the syntactic schema validation, see Section 4.2.4.

Objects are the most important data structure in JSON schemas. They map keys to values. The keys must always be strings. A key-value pair is referred to as a "property". When the type is required to be an "object", the structure of its properties can be set, as seen in Figure 4.5. These properties can again have value type "object". This will create a hierarchical object structure, which is often seen in JSON files.

Objects can have a key "additionalProperties", which controls whether one is allowed to add properties to an object that are not modelled in the schema. This can be set as "false", or an object which specifies what data structure these added properties are allowed to have. Similarly, the key "re-

```
{
  "type": "object",
  "properties": {
    "number": { "type": "number" },
    "street_name": { "type": "string" },
    "street_type": {
      "type": "string",
      "enum": ["Street", "Avenue", "Boulevard"]
    }
  }
}
```

Figure 4.5: Schema example for JSON objects, from Droettboom [2019].

```
|-- appearance.schema.json
|-- cityjson.schema.json
|-- cityobjects.schema.json
|-- geomprimitives.schema.json
|-- geomtemplates.schema.json
|-- metadata.schema.json
|-- /extensions
    |-- noise.json
```

Figure 4.6: CityJSON schema structure, from CityJSON [2019].

quiredProperties" can also be added. Normally, an empty object is always valid. However, the value of "requiredProperties" may determine what keywords should be present in the object in the data file [Droettboom, 2019].

JSON schema also has commands for combining schemas. This is extensively used in CityJSON. By using the keywords "allOf", "anyOf" and "oneOf", a value can be validated against all, at least one or exactly one specified sub-schema respectively. It is also customary to add a "definitions" keyword at the beginning of a schema. This will include certain data type templates which can be re-used in the schema [Droettboom, 2019]. Finally, an important element is "$ref". This can be used to create a recursive schema that refers to itself. Using "$ref" in combination with the keywords "allOf", "anyOf" and "oneOf" will give the tools to make powerful schemas without much duplication [Droettboom, 2019].

4.2.2 CityJSON schema structure

CityJSON files are validated against the file `cityjson.schema.json`. This file refers to other schema files. The schema files and how they are structured is shown in Figure 4.6. These files are referred to as the CityJSON core. Outside of the core are the extensions, covered in Section 3.1. The file `cityjson.schema.json` validates the `CityObjects` to the schema `cityobjects.schema.json`, which in turn validates the different geometry types to `geomprimitives.schema.json`. In a CityJSON file, the extensions used are declared. The extension files need to be in the folder extensions in the schema folder. The CityJSON file is then also validated against the extension. The extensions themselves can be validated against `extension.schema.json`.

4.2.3 Changes to CityJSON schemas

Here I outline how the data needs will be modelled in CityJSON. Some changes will be made in the core schemas of CityJSON. The other changes

```
"Node": {
  "allOf": [
    { "$ref": "#/_AbstractCityObject" },
    {
      "properties": {
        "attributes": {
          "properties": {
            "edges": {
              "type": "array",
              "items": {
                "type": "string"
                },
              "description": "array of Edge CityObject IDs
              which are incident to the node"
            },
          }
        },
        "geometry": {
          "type": "array",
          "items": {
            "$ref":"geomprimitives.schema.json#/MultiPoint"
          }
        }
      },
      "required": ["geometry"]
    }
  ]
}
```

Figure 4.7: Addition of object class Node in `cityobjects.schema.json`.

will be realised in an extension called `RoadExt.json`. Why a change is modelled in one or the other is motivated below.

As observed earlier, roads are not the only class which could benefit from a graph structure. For this reason, nodes and edges are modelled as new general classes `Node` and `Edge` in the core. They are modelled by MultiPoint and MultiLinestring geometries repspectively. These new `CityObjects` are directly added in `cityobjects.schema.json`. Nodes have an array attribute that specifies which `CityObjects` are the edges that are incident to it. Edges have both a "startNode" and "endNode" attribute, which specify from and to which `Node` the edge goes. The additions in the schema for `Nodes` and `Edges` can be found in Figure 4.7 and Figure 4.8 respectively.

A graph structure has been defined which can be applied throughout the data model. In order to use this structure for roads, new `CityObjects` are defined in the extension, under "extraCityObjects". The new objects defined are `+RoadNode` and `+RoadEdge`, which are `Nodes` and `Edges` enriched with road attributes. The `RoadNode` can have four different types: "Intersection", "Roundabout", "Attribute" and "LaneSplit". In what LoD which attribute can be modelled is specified in Section 4.3. The `RoadEdge` can be of the type: "Road", "Connecting" or "Roundabout". `RoadNodes` can have an attribute "turns", which specifies what roads can be reached from where at an intersection node. This is represented by a two-dimensional array with zeroes and ones. A one at position i, j means that that one can drive from edge i to edge j, where i and j represent the index in the "edges" array in the `Node`.

The linear road segments `RoadEdges` can have many attributes. Earlier, I identified the needs for allowed modes of transport, street name, maximum speed, administrator and driving direction. Because of the schemaless principle of CityJSON, these can just be added as attributes in data files. However, they are modelled in the extension for completeness. As the class Road is validated against `_AbstractTransportationComplex`, it gets the at-

```
"Edge": {
  "allOf": [
    { "$ref": "#/_AbstractCityObject" },
    {
      "properties": {
        "attributes": {
          "properties": {
            "startNode": {
              "type": "string",
              "description": "CityObject ID of startNode"
            },
            "endNode": {
              "type": "string",
              "description": "CityObject ID of endNode"
            },
          }
        },
        "geometry": {
          "type": "array",
          "items": {
            "$ref": "geomprimitives.schema.json#/MultiLineString"}
        }
      }
    }
  ]
}
```

Figure 4.8: Addition of object class Edge in `cityobjects.schema.json`.

tribute "surfaceMaterial", but the `RoadEdge` does not inherit this. This is because the class `Node` only inherits from the `_AbstractCityObject`. As a consequence, in the linear representation, the surface material still needs to be added as an attribute. From `_AbstractCityObject` both representations do get "class". Thus road classification does not need a separate attribute.

Attributes for the areal representation are modelled differently. This is because an areal `Road` object is modelled with the aforementioned semantic surfaces. The attributes have to be added per semantic surface. This can again be done schema-less. To document this in the schema, it requires some change in how the semantic surfaces are modelled. Road semantic surfaces are now either of the type `TrafficAreas` or `AuxiliaryTrafficAreas`. Additional attributes or categorisation is not modelled. For the LoD specification proposed by Labetski et al. [2018], a further specification to "Road", "Carriageway" and "Lane" is needed. From the data needs "Intersection" and "Roundabout" should also be added. To keep the CityGML data model as intact as possible, the `TrafficArea` and `AuxiliaryTrafficArea` are kept. Thus we model Road, Carriageway, Lane, Intersection and Roundabout as subclasses of TrafficArea. I do this by adding attribute "roadType", which can have one of the five values mentioned above when the semantic surface is a `TrafficArea`. To the semantic road surfaces I propose to add an attribute to link it to an "OuterFloorSurface" of a bridge. This way, the road can be "laid over" the bridge with the same surface. As said above, this is not possible yet for tunnels, as LoD4 is not yet modelled in CityJSON.

In order to link the representation types, I have added an extra possibility to model geometries in `Roads` with an array of strings. These strings then refer to the `CityObject` id's of the `RoadEdges` that correspond to the object. Thus the geometry of a `Road` object is either a MultiSurface, or an array with `RoadEdges`. To each `RoadNode`, `RoadEdge` and `Road`, I add the ability to add an "intersectionID". This is also modelled as an attribute. This might be useful for generalization purposes, for instance. This way, traffic light information can also be linked to intersections.

4.2.4 Schema validation and semantic validation

The CityJSON schemas serve not only as a guide for how to model the data. It is also used to validate the syntax of the data files. The files can be validated with a JSON schema validator. However, these are not compatible with extensions. The software `cjio` has been developed to handle, adapt and validate CityJSON files [Ledoux et al., 2019]. Data files can also be checked against CityJSON extensions with `cjio`.

JSON schemas are not suitable for semantic validation. Although one can set limitations on the values an attribute value might attain, it cannot be checked whether these values make sense with respect to other attributes or features. The software `cjio` does also perform semantic validation. However, this is at this moment of course just for the regular CityJSON release. The road extension syntax developed from this thesis can be checked according to the file `extension.schema.json`. This has been done, and the `RoadExt.json` extension is valid. However, the added semantics cannot yet be validated. Examples of added semantic constraints are:

- LoD constraints. A semantic surface can only be a carriageway in LoD2, and a lane in LoD3. Turning lanes may only be represented in LoD0.2 and LoD0.3. A node may only have as NodeType "laneSplit" in LoD0.3. These types of restrictions cannot be made in JSON schema.
- `Nodes` refer to `CityObject` id's of incident `RoadEdges`, and vice versa. It needs to be checked whether these id's exist in the data file.
- In an LoD0.1 intersection RoadNode, the number of rows and columns of the two-dimensional array value of the "turns" attribute must correspond to the number of incident RoadEdges.

Among others, these semantic constraints need to be either incorporated in `cjio`, or externally validated. This is outside the scope of this research, and is recommended as future research.

4.3 UPDATED LOD SPECIFICATION

The identified road data needs have been processed in the updated CityJSON schemas. Here the improved LoD specification of roads in CityJSON is summarised in tables. In Table 4.1 the modelled attributes for the linear representation is shown. When combined with the geometric overview in Figure 4.4, we can get a complete overview of how I propose to model roads linearly in the CityGML data model. As can be seen, most attributes are modelled at all linear LoDs, except for some related to topology. Modelling turn restrictions with "turns" is meant only for LoD0.1, but is allowed in LoD0.2 and LoD0.3 when "modelling down", as discussed before.

The areal LoD specification is an extension of the proposals by Beil and Kolbe [2017] and Labetski et al. [2018]. The attributes modelled are set out in Table 4.2. Road objects can be modelled by a `TrafficArea` at LoD1, where the whole area is of the type `Road`. "Intersections" and "Roundabout" are already their own "roadType" in LoD1. In LoD2, different carriageways are modelled separately, both as `TrafficArea` semantic surfaces with type "Carriageway". From LoD2, `AuxiliaryTrafficArea` is used to represent the non-drivable parts of the road. Intersections and roundabouts are still modelled with their own roadType. In LoD3, each driving lane gets its own

				Linear	
Object	Attribute	Value	LoD0.1	LoD0.2	LoD0.3
RoadNode	edges		x	x	x
	roadNodeType	Intersection	x	x	x
		Roundabout	x	x	x
		LaneSplit			x
		Attribute	x	x	x
	turns		x	(x)	(x)
	intersectionID		x	x	x
RoadEdge	startNode		x	x	x
	endNode		x	x	x
	edgeType	Road	x	x	x
		Connecting		x	x
		Roundabout		x	x
	class		x	x	x
	function		x	x	x
	direction	toEnd, toStart, both	x	x	x
	maxSpeed		x	x	x
	administrator		x	x	x
	streetName		x	x	x
	intersectionID		x	x	x
	surfaceMaterial		x	x	x
	modes		x	x	x

Table 4.1: Attributes modelled per linear LoD.

				Areal	
Object	Attribute	Value	LoD1	LoD2	LoD3
Road	roadType	Road	x		
		Carriageway		x	
		Lane			x
		Intersection	x	x	x
		Roundabout	x	x	x
	class		x	x	x
	function		x	x	x
	intersectionID		x	x	x
	streetName		x	x	x
	bridge		x	x	x
	administrator		x	x	x

Table 4.2: Attributes modelled per areal LoD.

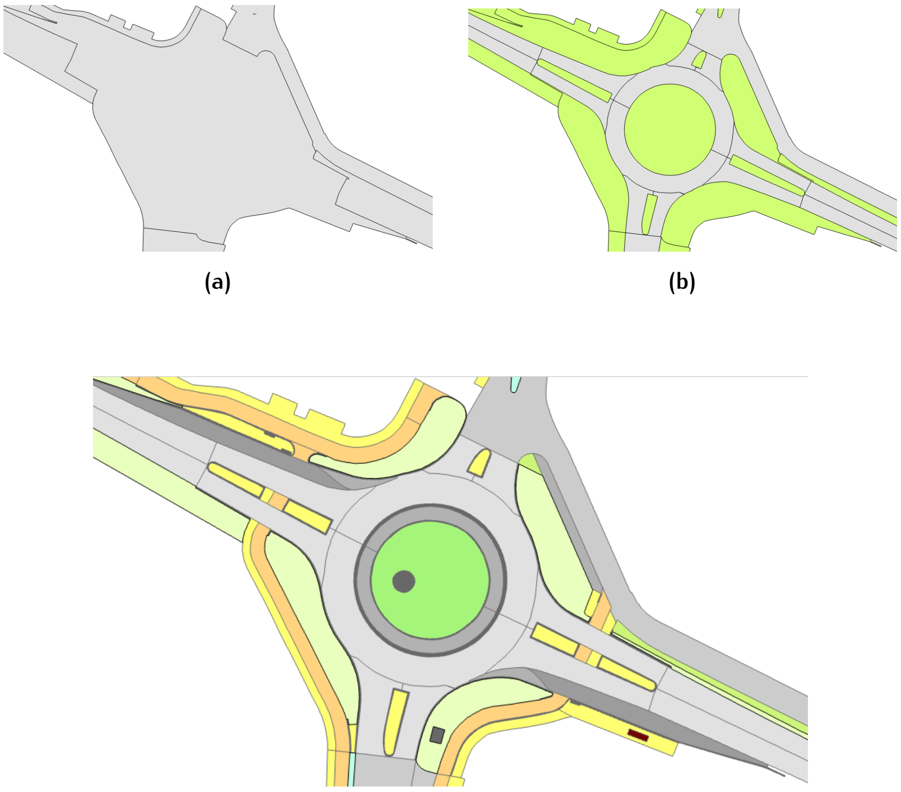

(a) (b) (c)

Figure 4.9: A roundabout at (a) LoD1, (b) LoD2 and (c) LoD3.

TrafficArea, with type "Lane". Again, "Intersections" and "Roundabouts" have their own "roadType". The areal modelling of intersections and roundabouts as TrafficAreas thus does not necessarily change through LoD1 to LoD3, as the driving lanes are not modelled – although an intersection might be more segmented in LoD3. Some parts of an intersection might become AuxiliaryTrafficAreas from LoD2 upwards (for example, a grassy area in the centre of a roundabout, see Figure 4.9).

The use of AuxiliaryTrafficArea in CityGML is not well specified. In the documentation, it is used from LoD2 onwards, with increasing geometric and semantic complexity [Open Geospatial Consortium, 2012b]. However, I have followed Beil and Kolbe [2017] and Labetski et al. [2018] in removing LoD4 from the road specification. Thus there are only two levels where AuxiliaryTrafficAreas are modelled. My suggestion is to model these parts of the road at LoD2 as simply AuxiliaryTrafficAreas, and provide further segmentation at LoD3 in the categories one desires. The existing code list for AuxiliaryTrafficArea prescribes values such as "Kerbstone", "Ditch", "Traffic Island". These values can then be attained at LoD3.

When solely the areal representation is modelled, without linking with linear road objects, the modeller can make a choice in object scale. One can model each surface as a separate object. In that case, each Road object has only one semantic surface. These might be very small polygons, for example in the case of triangulation. A rather more elegant solution would be to set the object at a bigger scale, and model smaller surfaces as semantic surfaces belonging to that object. This is however not necessary. The modeller can make their own choice in deciding the object scale.

5 PROOF OF CONCEPT: CREATING CITYJSON ROAD DATA FILES

The updated CityJSON data model and the new LoD specification for roads are the main results in this thesis. In this chapter, as a proof of concept, I will transform a dataset into a CityJSON road data file. This serves multiple purposes. First, a data file is useful for demonstrating the new model. But more importantly, it can be used to identify potential shortcomings of the data modelling choices. Some modelling choices can seem theoretically smart, but in practice might be very difficult to use. Thus, creating a data file from another dataset will reveal certain discussion points on the usability of the data model.

5.1 DESCRIPTION OF THE DATASET

For creating the data file, a dataset maintained by the Provincie Noord-Brabant is used. The dataset are shapefiles with areal and centre line data of provincial road N640. The source data is from the BGT/IMGEO (see Section 2.5), and the data is enriched with additional attributes and classes used for maintenance of public space. The modelling catalogue can be found in Deijzen et al. [2014].

Both the areal and linear dataset are very detailed. The extensive catalogue makes that small areal features, like kerbs, are modelled as their own objects. The areal dataset can be considered LoD3, based on the geometric and semantic scale, and the fact that the polygons have three-dimensional vertices. The linear dataset is also detailed, but can be considered a combination of LoDs. It consists of centre lines of all lanes, and is thus LoD0.3. However, there is also a centre line for the entire road, which is LoD0.1. Furthermore, there is no line adjacency data in the dataset; no nodes or adjacency lists are present. In a sense this data can thus even be considered LoD0.0. In Figure 5.1 both layers are shown overlapping.

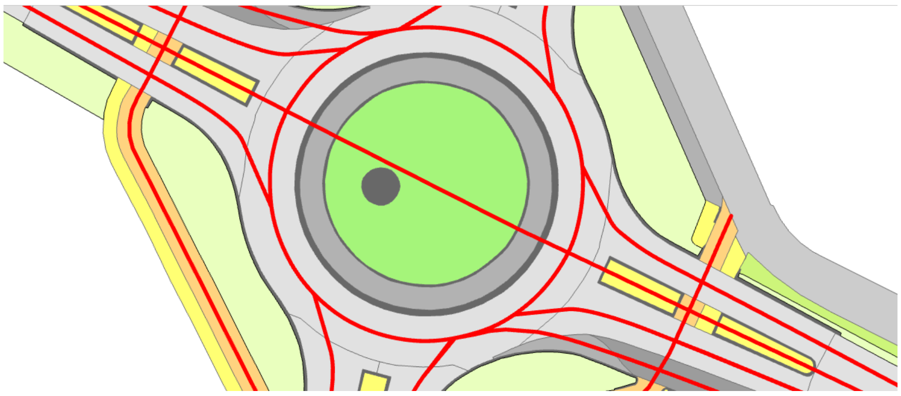


Figure 5.1: Linear and areal data of the N640 road in Noord-Brabant.

My goal is to model one stretch of road at all developed levels of detail. In order to cover all elementary modelling concepts, I have chosen a stretch between an intersection and a roundabout, between Kruisstraat and Hoeven. The street has one lane per driving direction, and at the intersection some separated turning lanes (see also Figure 5.3a). This makes an interesting case, given that it links to the carriageway discussion from Section 2.3.2. The dataset also includes polygons and centre lines for bicycle lanes, also visible in Figure 5.1. For this case I have decided to not consider these and focus on the main roads. It is however interesting to research how these can be incorporated, and is recommended future research.

5.2 TRANSFORMING THE DATA

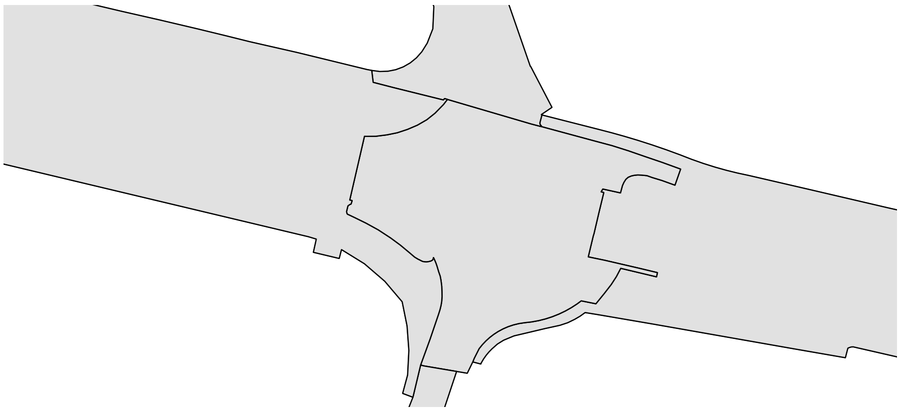
In this section it is explained what steps have been taken to transform the data into CityJSON road format. I use both QGIS and Python to create the data file. QGIS was used to preprocess the data, and to do some spatial analysis. After, the data was written to CityJSON with Python. In the program, also some additional linking of data was done. The process, along with the difficulties, is outlined below. I want to have a CityJSON file at every level of detail, and one where I link both representation types at LoD(0.)1. First I considered the areal representations, then the linear representations, and after that the linking of both. Note that the purpose of the transformation was not to create a perfect dataset. The transformation was done to reflect on the modelling choices made, and to see what happens when one wants to model a single dataset at various levels of detail. The resulting data files and the Python programs used can be found at `https://github.com/fhb1990/cityjson`.

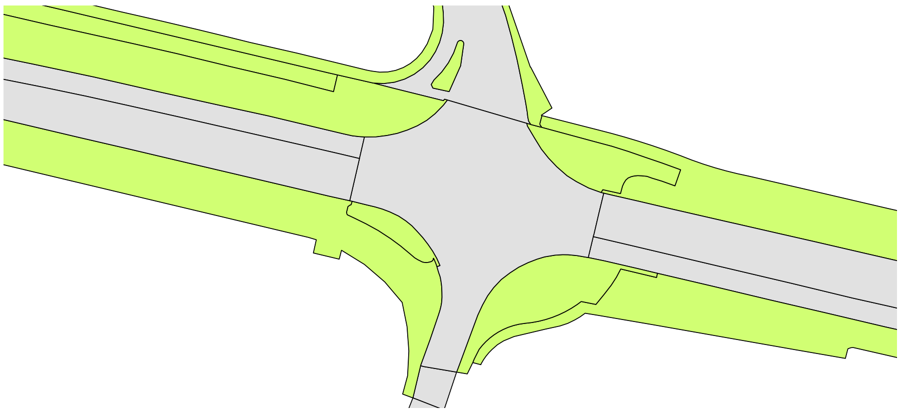
5.2.1 Areal modelling

The areal dataset contains polygons that are directly adjacent to the road. The first step in data processing was to remove these. After, the features that I wanted to consider in the transformation were selected. Besides the roundabout, the intersection, and the stretch of road between the two, the other "ends" of the roundabout and intersections were also selected, but only for a short distance. Afterwards, polygons that I consider part of an intersection were given an "intersectionID". Thus there are only two id's. These were used later to map the value to the CityJSON attribute.

Creating the LoD3 file was rather straightforward. Figure 5.2c shows the dataset without any processing; the layer symbology came with the dataset. As said, the dataset already subdivides the road into driving lanes. All features that I considered to be driving lanes (no footpaths and bicycle lanes) were written as a `TrafficArea` with roadType "Lane", except when it had an "intersectionID"; then it was either written as roadType "Intersection" or "Roundabout". The Dutch object description was mapped to "function" and the administrator information to "administrator". Other attribute values were not available. All other features present were written as `AuxiliaryTrafficArea`, with the Dutch object description mapped to "function".

I decided to model each feature from the dataset as its own object. Thus each `CityObject` in the data file has only one semantic surface. All features had their own object ID. The only attribute I could have used to link sur-


(a) LoD1


(b) LoD2

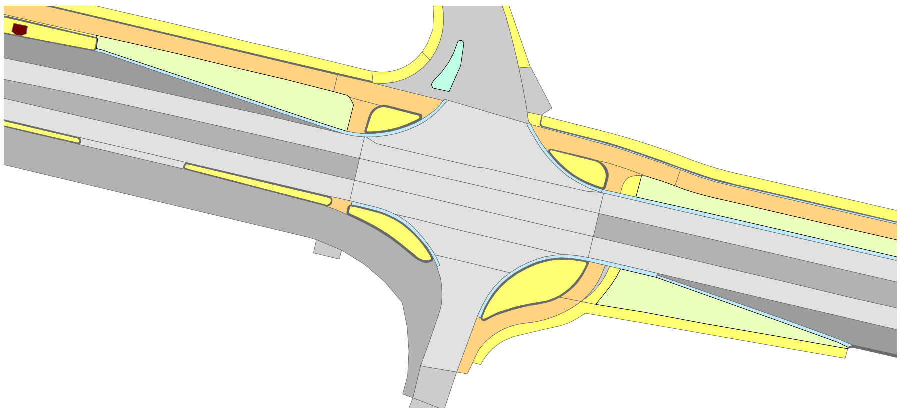

(c) LoD3

Figure 5.2: (a) & (b) Derived areal levels of detail after merging in the (c) original

faces together was the road number N640, but this was equal for almost the entire dataset. Therefore I chose to model each feature as its own object. In Section 5.2.3 I try to incorporate more semantic surfaces in one object.

Creating the LoD1 and LoD2 data files proved to be more difficult. This required the merging of features, such that the geometry aligns to the new LoD. For two adjacent driving lanes this is relatively easy. Still, adjacent lane features might have very different "endpoints" along the road, in which case merging result in exotically shaped polygons. For LoD1, I merged all features across a road; see Figure 5.2a. One could alter the vertices of the polygon boundaries in order to change the shape, but I chose to leave it as is.

The same issue occurs when merging `AuxiliaryTrafficAreas`. As outlined in the previous chapter, I decided to model these at LoD2 without any further specification, or semantic information. Thus, I randomly merged features that were at the side of the road (see Figure 5.2b), and wrote these as `AuxiliaryTrafficAreas` without any further attributes. The `TrafficAreas` were written with roadType "Carriageway" at LoD2 and "Road" at LoD1, as prescribed by the adapted data model. Also, when merging, I merged by the previously added "intersectionID", such that the intersection and roundabout could be written as separate objects with their own "roadType" at both LoD1 and LoD2.

One of the benefits of CityJSON is that is reuses vertices. However, the current implementation of my Python writer program does not reuse vertices. This is allowed by the CityJSON specifications, but this can be improved at a later stage.

5.2.2 Linear modelling

When writing a linear representation CityJSON road file, we need to write two types of objects: `RoadNodes` and `RoadEdges`. Given that the linear data only consists of edges, the nodes have to be generated. The linear data is specified per lane, but it was still work to create the LoD0.3 data file. This mostly has to do with the geometry of my proposed data model. The intersections at LoD0.2 and LoD0.3 have precise modelling prescriptions. The linear road data has some turning lanes, but not many. Thus these needed to be added manually. Some lines needed to be split in two. Also, not all endpoints of the linear data matched, which is crucial for implementing the topological data structure. Thus, I also had to snap many lines together. It needed only a small adjustment to obtain the LoD0.2 lines from these. Creating the LoD0.1 lines was easy, given that it consisted of only a few lines.

With QGIS I generated a point layer for start and end points of the lines, which contained the incident line features id. For LoD0.3, the result is shown in Figure 5.3. This happened for every line, which resulted in overlapping points (for instance: four when four lines meet in a vertex). Later, in the writer, it is checked whether there already is a vertex around that location, and if so, it adds its incident edge information to the other node. This information is crucial for the network structure. Again, I manually assigned intersectionID's, this time to turning lanes and the nodes that belong to intersections and roundabouts. When the node's `roadNodeType` had to be either "attribute" or "laneSplit", I also added this manually as an attribute value. The `edgeType` can be determined from the "intersectionID", otherwise it is just "Road".

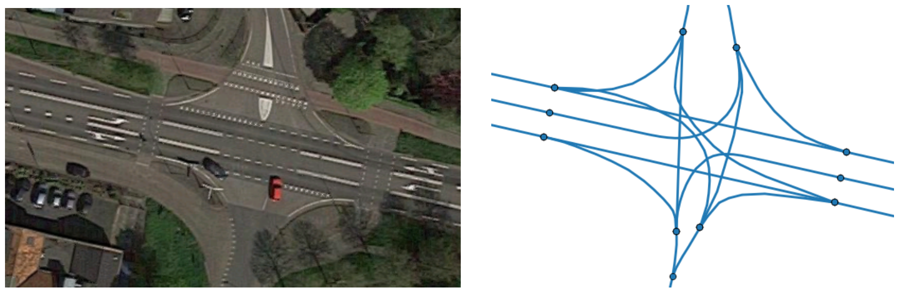


Figure 5.3: LoD0.3 representation of the intersection in the dataset.

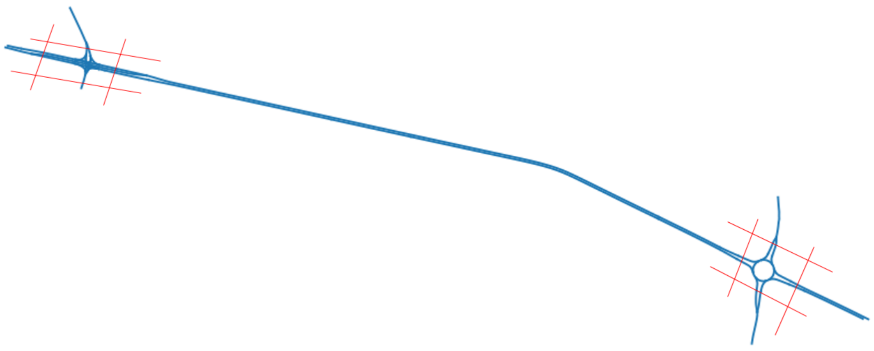


Figure 5.4: Division of the road data into various objects.

While the centre line data of the lane had attributes such as maximum speed and administrator, the LoD0.1 centre line data had no attributes which I could map. Other attributes like street names or driving directions were not present in the data.

When writing the CityJSON file, first the nodes were handled. As described above, duplicate points were aggregated such that all incident edge id's appeared in the "edges" attribute of the `RoadNodes`. After, the edges were written to the file. To obtain the start and end node of the edges, for each edge the program goes through all nodes to search for its own id in the attribute "edges". This is very inefficient, but given that the dataset was relatively small, it did not matter in this case. When this writer is used for larger datasets, the incident node id's should be added as attributes in pre-processing through spatial analysis.

5.2.3 Linking representation types

One of the results of the previous chapter was to provide a method to link a linear road representation to an areal representation. To do this, an object needs to be defined first. We need to know on what basis we aggregate edges and surfaces.

As a test case, I decided to manually create objects. I used LoD0.1 and LoD1 data to create a data file. I defined the intersection, the roundabout, the stretch of road between the two, and the other outgoing roads of the intersections as the objects(see Figure 5.4). The object id's were added manually, as it concerns a small amount of features.

After defining the object id's, the process was a combination of the previous two sections. First the nodes and lines were written as in the previous section. Then, the `Road` objects were written to file. The difference with before is that now surfaces with the same object id will become semantic surfaces in the same object, as desired. The writer program was slightly altered to achieve this. After writing the `RoadNode`, `RoadEdge` and `Road` objects, for each `Road` object the corresponding `RoadEdges` were written in an array in the "geometry" property.

5.3 REFLECTION ON MODELLING CHOICES

The aim of this chapter is to reflect on the new LoD specification by populating the improved data model. It can be concluded that it is not always easy to create a data file adhering to the LoD specification. Creating the LoD3 file was straightforward. The input dataset was already LoD3-like, and thus was easy to transform. Creating lower LoD datasets was actually less easy. I had to merge features, which led to some strange feature shapes. The process of merging polygons to create lower LoDs could be automated. This gets more in the domain of map generalisation. One can also ask whether it is desirable to create lower LoD datasets from higher LoD input data.

LoD0.2 and LoD0.3 require exact modelling of the turning lanes at intersections. When an input dataset does not have these turning lanes, they need to be added. This will cease to be a problem when datasets will be created with the improved data model in mind. However, for transforming data to the CityJSON encoding having these explicit turning lanes might be much to ask. The dataset used here already had some turning lanes, but still it took quite some preprocessing to get it right according to the LoD specification.

In Chapter 2 and Chapter 4 I talked about the possibility of "modelling down": using a lower LoD type to model an object. This also appeared when using this dataset. In Figure 5.3b, the road coming from the south is bi-directional, but has no lane specification. Because of this, the road is modelled in the input dataset with a single edge. Another option would have been to use two edges, one for each driving direction. This is both allowed. But given that it is already modelled like this, I chose to accommodate this. At the same time, users want to know what is in what LoD. Leaving too much room for own interpretation might lead to ambiguity in what an object is supposed to be at a certain LoD.

Linking the two representation types also leads to the question: what is the object? I have determined this manually for a small dataset. Ultimately it is something the data provider should decide on. Also users can determine objects themselves if they want to aggregate certain features. It is probably better to assign object ids based on spatial analysis, or on an attribute. In the linking done above, the linear representation objects did not end exactly at the boundaries of the corresponding polygons. The question is whether this is necessary. If desirable, one could split geometries based on the other representation type. However, when making a new datafile this should be less of a problem, and objects can be created such that they correspond spatially as well as semantically.

Working with this specific dataset brought up some additional questions. Merging `AuxiliaryTrafficAreas` when creating an LoD2 file resulted in strange-looking geometries. The role of `AuxiliaryTrafficAreas` in LoD2 is still not well-defined. Perhaps this is also the case for LoD3. One models nothing, the other model everything. The use of `AuxiliaryTrafficArea` might need reconsideration. It is also not clear what to do with parallel lanes, bicycle paths and footpaths. One can consider them part of the same `Road` object. In that case, at what LoD do I start modelling them, LoD2 or LoD3? Here I modelled them as `AuxiliaryTrafficAreas`, mostly for simplicity's sake. However, a further specification seems appropriate.

Creating the data files has given some insight in what the ideal CityJSON road dataset should look like. Ideally it contains both a linear and areal representation, with corresponding LoD. The objects are classified such that it is easy to determine whether something is an intersection, roundabout, road, or something auxiliary. Turning lanes are explicitly modelled, and endpoints of lines meet in the exact same point. It would be nice if features had an object id which specifies to what object a feature belongs. This would result in an easy linking between the representation types.

6 CONCLUSION, DISCUSSION AND FUTURE WORK

6.1 CONCLUSION

3D city models are becoming increasingly prevalent. As three-dimensional data becomes more common, the usage and potential applications of these models increases. As we have observed, government agencies are also moving towards capturing and storing their data in 3D. They are enthused about the potential of 3D applications. The different levels of detail of 3D city models allow objects to be modelled at multiple geometric and semantic scales. Different applications demand different LoDs. In CityGML, the most widely used data format and data model for 3D city models, there are five LoDs. This research was guided by the notion that the LoD specification of the transportation module is not well developed. As the development and importance of three-dimensional geographic data and city models is dependent on the use cases it enables, the aim of this research was to answer the question: how should roads and intersections be modelled at different levels of detail in 3D city models based on use case needs?

In meetings with the Dutch working group for Central Object Registration, it was made clear that the Dutch government wants to move to one central register where real-world objects are modelled only once. In this register, the object takes centre stage, and is the modelling unit. At this moment, objects are modelled in different key registers, which leads to problems with linking the data, and with temporal accuracy. 3D city models are also object-based. The CityJSON encoding of the CityGML data model accentuates that by having the `CityObjects` as the main modelling unit. Because of the COR intention and the object-central modelling of CityJSON, in addition to CityJSON user-friendliness, it was decided to model the roads and intersections in the CityJSON encoding of the CityGML data model. Some elementary changes were made in the CityJSON core. All other improvements were modelled with a CityJSON extension.

In short, the methodology in this thesis was that a) road data needs were extracted from use cases, b) these needs were compared to modelling choices for these data needs in other road standards, and then c) modelled in CityJSON. From reviewing other road standards, it was clear that these standards were all developed with one, or a couple of use cases in mind. In this research, in the same vein, I tried to select use cases that were diverse in application but still all had data needs that CityGML and CityJSON currently lack: the modelling of road networks. The use cases researched were transport modelling, automotive navigation and road maintenance. The latter use case was chosen because it also combines the two representation types.

From the use case data needs analysis, it became clear that many use cases rely on a graph structure. The road standard review showed different ways in which these structures were actually modelled. Choosing to adapt the data model in CityJSON allowed me to model graphs in a flat structure. New CityObject types `Node` and `Edge` were added to the CityJSON core such that networks can be topologically modelled. Furthermore, these graphs are not limited to road networks, and can be used for other classes as well. I have extended these new classes to RoadNodes and RoadEdges. With these new objects road networks can be modelled in 3D city models. This is one of the main results of this thesis, as this is something that up to now has not been done before. This opens many new possibilities for potential applications.

Apart from having a graph structure, many data needs concerned the way these networks were modelled through different LoDs. For example, how to allow for routing information at different linear LoDs, or how intersections should be explicitly modelled. The previously proposed LoD specification for the linear representation was combined with an assessment of how other road standards model roads and intersections in networks. This has led to a new LoD specification for the linear representations of roads, as seen in Section 4.3. This gives a clear description of how nodes and edges should be combined to form roads, intersections and roundabouts at various levels of detail.

The use cases assessed mostly pertained to modelling linear networks. However, data needs related to areal representations were also acquired. The data needs for areal representations are mostly attributes that can be added to the geometry itself. Thus, from the assessment of the three use cases listed, a further specification for the areal LoDs was not needed. The areal LoD specification in Section 4.3 is therefore mostly a worked out model of the previously proposed LoD improvement by Beil and Kolbe [2017] and Labetski et al. [2018]. There was a data need to specify intersections and roundabouts as objects separate from the roads, but no need for a further elaboration of the configuration for higher areal LoDs. As seen, intersections can have complex configurations and can differ among themselves. While at LoD0.3 the intersection configuration with all its turning lanes is completely modelled, my proposal for all areal LoDs is to simply model intersections and roundabouts as surfaces, without any further specification. The majority of data model improvements were thus made in the modelling of the linear network. These data needs were implemented in the schemas defining the CityJSON data structure.

A data need that arose was the ability to link the linear and areal geometries together. This corresponds to the idea of having central object registration. The decision was made to model changing attributes in networks by using a node. In this way, the edges of road networks have their own set of attributes. This can however lead to a highly fragmented network when many attributes are modelled. When one wants to link areal and linear representation under the guise of one object, and the modelling features are quite segmented, it will be very difficult to consistently model a one-to-one mapping between the two. The concept of having road segments in the data model has been proposed before. The way this is modelled in this thesis is not by mapping the smallest modelling objects to each other, but by making this possible on a more aggregated level. As previously stated, `Nodes` and `Edges` become separate `CityObjects` such that they can be used by multiple modelling classes. In the CityJSON extension, these are enhanced into `RoadNodes` and `RoadEdges`. These constitute the linear road network. The areal geometry is already modelled using semantic surfaces. A Road object can contain many of these surfaces within one geometric object. Together they are the geometric value of a Road object. The decision was made to allow `Road` objects to also have, as a geometry, an array of `RoadEdges`. In this way, there does not need to be a new `Road` object each time an attribute changes. Thus there can be a mapping between the two representation types, despite different segmentations. The data provider decides what the scale of the Road object is at which the different segmentations of the same `Road` object (linear road segments and semantic surfaces) are aggregated.

Transforming data into CityJSON road files has shown that as an encoding, CityJSON is easy to work with. Transforming areal data to the CityGML data model is rather straight forward, as it is mostly a case of attribute mapping. However, linear data needs to adhere to the new LoD specification, which can be geometrically strict at high LoD intersections, for example. The linear dataset needed quite some preprocessing before the mapping could take place. However, in order to obtain a graph structure such that routing is possible, strict modelling choices are perhaps necessary.

How do we model roads and intersections across various levels of detail in 3D city models? Previous proposals by Beil and Kolbe [2017] and Labetski et al. [2018] have been combined with a new data needs assessment. This has resulted in a new improved CityGML road data model, with a corresponding LoD specification. For LoD2 and LoD3 it is specified how the areal geometry and semantics should be modelled, whereas this was not clear in the current CityGML transport module specification. Furthermore, a new LoD0 specification has been developed at three levels with a prescribed approach describing how to model these. Moreover, the research has resulted in a CityJSON extension which provides the schema for modelling the new LoD specification. In order to implement this, a method has been developed to model graphs in CityJSON. Finally, changes to the CityJSON core have been made to accommodate the central object modelling of different representation types into one Road object.

While the goal was to further enhance the LoD specification, I have tried to still allow for a lot of flexibility for the user. For example, I want to leave room for users to implement their own linear referencing system when they do not want to use a node-based attribute system. Also, navigation could be done with LoD0.1 or LoD0.3, depending on the amount of attributes one has. The goal is not to over-fit the data model with attributes, but allow users and data providers a framework where they can implement their own data.

6.2 DISCUSSION

In this research, the CityGML road data model has been improved based on data needs arising from different use cases. Thus the result is heavily dependent on use cases chosen. We saw that other road standards often had an origin in applications. This is also true for the changes in the data model made here. The use cases were chosen because of the previous research into an expanded LoD specification of roads. Given that the areal representation was already more specified, use cases which relied more on a network representation were chosen for the data needs analysis. Thus, the improvements to the data model mostly concern the LoD specification of the network.

The focus on using the linear representation of roads might lead to a one-sided view on what is necessary for road modelling. The use cases, transport modelling and navigation, had overlapping data needs, especially in attributes needed. This might lead to the preconception that some attributes are necessary for modelling roads (in 3D city models or otherwise).

However, in this research I have tried not to over-fit the data model with attributes. In government meetings about the COR, the same discussion was held. It can be difficult, and sometimes arbitrary, to determine what attributes are necessary to have in your model.

Furthermore, it is also debatable to what extent one wants to encode the possible values of an attribute. Take for instance the attribute driving direction. Every information model has different ways of modelling such an attribute. My proposal is to have three possible values for the three possibilities ("toEnd", "toStart", or "both"). The question is whether it is necessary to prescribe these values. Users might want to store their data in CityGML / CityJSON, but use their own information model values. When I created the CityJSON road data files, I for instance mapped the object description attribute to "function". The latter has a code list which prescribes values, but this way the actual values from the original information model can be transferred. This accommodates data providers to store their own data using the CityGML data model. Accommodating data providers' attribute values can also be done by creating a CityJSON extension.

The choice was made to model attribute changes in networks using nodes. The choice for node-based attribute modelling can be defended by the way the representation types are linked, and also because using a linear referencing system instead is still supported. The network might become highly segmented, but this will not influence the segmentation of the areal representation. Furthermore, if one wants to prevent a high level of segmentation, an LRS can still be implemented. For example, the NWB uses linear referencing for many road attributes. There are plans for the NWB to be part of the future road central object registration of the Netherlands. Thus, this is an extra incentive to be able to accommodate LRSs in the data format. The way this can be integrated with the proposed CityJSON encoding is something that is still open to research.

The new LoD specification leads to a discussion on the meaning of LoDs. A higher level of detail in a 3D city model means that both the geometry and the semantics related to the geometries are more detailed. Different authors place different emphasis on whether the LoD differentiation should focus more on the geometry or the semantics. In the areal road LoD specification, the LoDs are geometry-led, but the semantics change along with it. In a high areal LoD, having a "lane" semantic surface only makes sense when that surface actually exists. The geometry has to be divided into different parts in order for the semantics to follow. In a certain way, the linear LoD specification is similar in this aspect. The LoD specification set out in Section 4.3 is very geometrical in its approach, and the semantics also follows suit. However, in a low level of detail, the network can still be quite semantically rich. In networks, many things that cannot be deduced from its geometry, can be added as attributes (e.g. number of lanes, cross section, etc.). An example of this is OpenStreetMap, where roads are often represented by just one line, but have many key-value pairs attached to it, which makes it usable for many applications. This again leads to the discussion on how desirable it is to have many attributes modelled. Do I want to be able to do complex network computations with LoD0.1? If yes, should this information be added to the data model, or can this be modelled as an extension? Low LoD road models can be useful when detailed geometric information is not necessary. This might be the case when working with a dataset with a very large extent. This might lead to a case where the lack of detailed geometry has to be compensated by a high semantic detail in order to perform the needed computations, but still have a manageable dataset. This can lead to a superfluous idea of LoDs. Thus, it can be discussed whether all of these strict LoDs are necessary in a 3D city model. However, the strength in having the detailed LoD specification is perhaps also in having people decide what LoD they want to use for the data they already have. Furthermore, it is of course important to have a standardised way of knowing what the data requirements are for certain datasets, so users know what to expect.

The above mirrors how Biljecki et al. [2013] see LoDs: as a quality measure of a model based on different factors. They eventually let go of the five LoDs of CityGML, in order to permit an LoD specification based on more factors than first exterior complexity and eventually interior complexity. However, in this research I have stuck to the LoD proposals by Beil and Kolbe [2017] and Labetski et al. [2018]. Mostly because the data needs and assessed use cases seemed to fit with the LoDs presented in these proposals.

The node-based attribute changes in road networks also lead to a parallel idea in level of detail. When many attributes changes are modelled, a network can become highly fragmented. This can be interpreted as having a high level of detail. However, this could be done at LoD0.1. In this sense, the LoD specification of networks becomes semantic-led: as the semantic detail increases, the network becomes more segmented, and thus becomes more detailed. This is a separate notion of LoD, which can be considered *along* the road, as opposed to the current LoD specification, which mostly considers the detail *across* the road. Earlier I established that in the *across* linear LoD specification, geometric and semantic LoD are not necessarily related. The same can thus be said about the *along* LoD when using the node-based attribute change method. Naturally, this notion changes when one uses a linear referencing method instead.

An important question that impacts the relevance of this study, is whether we need 3D road data at all. It can be argued that 3D data was not really needed in the use cases assessed in this research. Then again, some use cases can benefit, like the waste collection use case which incorporated slope deducted from 3D data. Although this could also be an attribute of a 2D dataset. I would argue that the real strength of having the road data in a 3D city model is that it adds to the benefit of being able to combine a road network with spatial analysis of the surroundings around the road. Other use cases like noise modelling definitely benefit from using 3D city models, as all objects surrounding roads that are necessary for the analysis are already present. This use case was not chosen for this thesis as it has already been researched at TU Delft. Also, keeping the central object registration in mind, and the tendency of governments to move towards 3D geographic data, it makes sense to have these objects modelled in a 3D city model, since this is also 3D and object based.

The central object concept has led to the implementation, in this research, of linking the two representation types, but not at the smallest scale of semantic surfaces and RoadEdges. A question is whether this mapping of two representations of objects is otherwise desirable. 3D city models are object-based models. Thus, conceptually linking the types seems like good practice. However, when doing spatial analysis, the linking may also be done on the fly by doing a spatial join. This does require that the network is geometrically modelled overlapping with its areal counterpart. Having the linkage modelled may lead to issues in creating the data such that the objects line up correctly. In the COR meetings this was also addressed as a potential problem. Given that they would like to have a link between the representation types, there is skepticism around the use of node-based attribute change modelling. My proposal might be a solution for them, as it permits bigger objects, but still with a possibly highly segmented network.

Eventually, it comes down to accommodating the user and data providers. 3D city models can never be one size fits all. Not even five sizes fit all. The five specified LoDs of CityGML help in providing quality measures for both users and providers. Nevertheless, these levels will never be perfect for any application. There needs to be a balance between meaningless LoDs – as the Transportation module specification was before – and highly specified and restricted LoD specifications. In every modelling choice made in this research I have tried to keep that balance. However, from creating the data files it did turn out that it is still very dependent on the dataset whether the LoDs fit well with the data. Then again, having well-specified LoDs makes it easier for new data providers to create consistent datasets among themselves.

It was decided to encode the improved data model using the CityJSON encoding of the CityGML data model. The development of CityJSON is still in its early stages. Thus, this research also serves as proof of how CityJSON can be extended for certain use cases. However, the characteristics of CityJSON have also influenced certain design choices. For example, the object-based nature of CityJSON helped in modelling the linking of the representation types. Moreover, because CityJSON uses `Road` as a separate CityObject class, the changes made to the data model are actually restricted to the roads only. In CityGML, roads are part of the `Transportation` module. Furthermore, the semantic surfaces used in modelling the areal representation differ from the way CityGML stores surfaces. The semantic surfaces made it easy to have a `Road` object composed of different smaller parts. Therefore it made it easier to link this with the corresponding road segments that are part of the same object. When one wants to store a data file adhering to the newly improved data model in CityGML, they would need a new way of implementing these concepts. Topology of networks can theoretically be implemented using XLinks. However, because CityGML does not store vertices like CityJSON, this might lead to too much redundant data.

Choosing to encode the improved data model in CityJSON does not only have technical consequences. CityGML is widely accepted as the main standard for exchanging 3D city models. Modelling my proposed changes in CityGML might thus lead to more impact. However, I have noticed that users and government officials are interested in the JSON encoding of the CityGML data model. It seems like users are not overly attached to the CityGML encoding itself. They care more for how they can eventually use and store the information. Also, there exists software to convert CityGML to CityJSON and vice versa [CityJSON, 2019]. Thus, it matters less which encoding is used. Though I must add that the improvements I propose are not yet processed in this software.

As was said, the data model changes were based on a needs analysis of three different use cases. This, by definition, makes the additions limited. A benefit of using CityJSON is that it is still in development, and its development is an ongoing open source project. This makes it easier to propose changes and implement them. This is also why it is interesting to focus future research on what other changes can be done, not only to the Road data model, but also to the structure of the entire JSON encoding of the data model. This might lead to new changes that will partly contradict with my final proposal. In this way the development of CityJSON can be advanced such that it might become a widely-used alternative encoding of CityGML.

6.3 FUTURE RESEARCH

Conducting this research has led to the insight that gaining knowledge also means gaining many new questions. The following subjects can be researched following this thesis.

- In this research I focused on roads which are mostly used by cars. Further thought can be put into checking whether a further specification is needed for bicycle lanes, footpaths, or pedestrian areas. The CityGML Transportation module also considers railways and squares. Can the concept devised in this thesis also be applied to these object classes?
- In this research I have only considered having objects of the same LoD in one data file. Modelling the same object in different LoDs is already possible. How do we model neighbouring road objects in different LoDs? Can a `RoadNode` be incident to `RoadEdges` of various LoD? What would this imply for the data model?
- In the discussion above I observed that having a segmented road network leads to a second type of LoD: not *along* the road but *across*. Should the CityGML concept of LoD take this into account. Previous authors have proposed multi-dimensional concepts of LoD (see for instance Biljecki et al. [2013]). This might be something to consider for roads as well.
- This research mostly focused on improving the linear LoD specification of roads in the CityGML data model. However, also a further specification of areal LoDs might be needed for several use cases. A further use case analysis based on areal representation might be useful. Also, my implementation of `AuxiliaryTrafficArea` is still somewhat vague for practical use, as turned out in Chapter 5. This might benefit from a further improved specification. One could also think of getting rid of both `TrafficArea` and `AuxiliaryTrafficArea` completely. I decided to keep these, in order to keep in line with the current CityGML data model and encoding. Which leads me to...
- At the time of writing, CityGML 3.0 is in development. This might entail changes to the `Transportation` module. If that is the case, a potential topic to research is whether the improved CityJSON encoding has to be adapted, and if how this can be done.
- My modelling choices were often made with the thought of accommodating many different data users and providers in the same data model. I implemented a node-based approach to attribute changes, but also mentioned that, if one wants, they can additionally implement a linear referencing structure. How this might be done in CityGML or CityJSON is still to be researched.
- A topological graph structure has been implemented for linear road networks. It is however not clear whether this structure is especially suited for routing purposes. If one wants to do perform routing with

CityJSON road data, is an additional structure needed? Perhaps an adjacency matrix or list can be extracted from the road data, which helps routing performance.

- During the data need analysis, the need for having road cross section information came up. This information can be quite complicated for roads with various lanes. It was out of scope for this thesis to implement a way to use cross section data in CityJSON. However, an extension can be devised which works out this concept.
- The software `cjio` currently provides a way to validate CityJSON files both syntactically and semantically. However, by changing the data model, the semantic validation process needs to be updated. Some examples of validation points are given in Section 4.2.4. A full overview of validation checks is needed. Afterwards this needs to be modelled, either in `cjio`, or in a separate program.
- In Chapter 5, I transformed road data to CityJSON road data at all LoDs. The CityJSON writer Python program is however only made to use the specific dataset I used with the specific preprocessing I did. This writer can be generalised and further developed into a writer that takes arguments, and perhaps does geometric / topologic data processing inside the program.

BIBLIOGRAPHY

Baig, S. U. and Rahman, A. A. (2012). Generalization and visualization of 3D building models in CityGML. In *Pouliot J., Daniel S., Hubert F., Zamyadi A. (eds) Progress and New Trends in 3D Geoinformation Sciences. Lecture Notes in Geoinformation and Cartography.*, pages 63–77. Springer Berlin Heidelberg.

Beil, C. (2017). Detaillierte repräsentation des straßenraums in 3d-stadtmodellen. Master's thesis, Technische Universität München.

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

Biljecki, F., Kumar, K., and Nagel, C. (2018). CityGML application domain extension (ADE): overview of developments. *Open Geospatial Data, Software and Standards*, 3(1):13.

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

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

Biljecki, F., Zhao, J., Stoter, J., and Ledoux, H. (2013). Revisiting the concept of level of detail in 3D city modelling. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-2/W1:63–74.

Boxill, S. A. and Yu, L. (2000). An evaluation of traffic simulation models for supporting ITS. *Houston, TX: Development Centre for Transportation Training and Research, Texas Southern University*.

CityJSON (2019). CityJSON 1.0 Documentation. https://www.cityjson.org/. Accessed: 2019-05-02.

Claussen, H., Lichtner, W., Heres, L., Lahaije, P., and Siebold, J. (1989). GDF, a proposed standard for digital road maps to be used in car navigation systems. In *Conference Record of papers presented at the First Vehicle Navigation and Information Systems Conference (VNIS '89)*. IEEE.

CROW (2018). Intersection Topology Format (ITF). Topology Guidelines version 2.1. Technical report, CROW.

Curtin, K. M., Nicoara, G., and Arifin, R. R. (2007). A comprehensive process for linear referencing. *URISA Journal*, 19(2):23–32.

de Dios Ortúzar, J. and Willumsen, L. G. (2011). *Modelling transport*. John Wiley & Sons, 4th edition.

Deijzen, W. v., Steinvoort, N., Buul, K. v., Reyngoud, N., Looijen, W., and Gerwen, S. v. (2014). *Objectenwoordenboek IMGEO+ Provincie Noord-Brabant*. Provincie Noord-Brabant.

Droettboom, M. (2019). *Understanding JSON schema. Release 7.0.* Space Telescope Science Institute.

Ducloux, P. (2016). RoadXML format specification 2.4.1. Technical report, RoadXML.

Dupuis, M. (2015). OpenDRIVE format specification, Rev. 14. Technical report, VIRES Simulationstechnologie GmbH.

Dupuis, M., Strobl, M., and Grezlikowski, H. (2010). OpenDRIVE 2010 and beyond–status and future of the de facto standard for the description of road networks. In *Proc. of the Driving Simulation Conference Europe*, pages 231–242.

Egenhofer, M. J. (1993). What's special about spatial? Database requirements for vehicle navigation in geographic space. In *SIGMOD '93 Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 398–402.

Essen, R. v. and Hiestermann, V. (2005). "X-GDF" — the ISO model of geographic information for ITS. In *ISPRS Workshop on Service and Application of Spatial Data Infrastructure, XXXVI (4/W6)*.

Gröger, G. and Plümer, L. (2012). CityGML – interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 71:12–33.

Haubrich, T., Seele, S., Herpers, R., Müller, M. E., and Becker, P. (2014). A semantic road network model for traffic simulations in virtual environments: Generation and integration. In *2014 IEEE 7th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*, pages 43–50. IEEE.

Häfele, K.-H. (2011). CityGML model of the FJK-haus.

ISO (2011). *Intelligent transport systems - Geographic Data Files (GDF) - GDF5.0 (ISO 14825:2011)*.

Kadaster (2019). Geschiedenis van de topografie. https://www.kadaster.nl/web/kadaster.nl/over-ons/het-kadaster/geschiedenis/topografie. Accessed: 2019-06-21.

Kadster (2019). Meerjarenbeleidsplan: speerpunten. https://www.kadaster.nl/over-ons/beleid/meerjarenbeleidsplan/speerpunten. Accessed: 2019-06-29.

Kolbe, T. H. (2009). Representing and exchanging 3D city models with CityGML. In *Lee J., Zlatanova S. (eds) 3D Geo-Information Sciences. Lecture Notes in Geoinformation and Cartography*, pages 15–31. Springer Berlin Heidelberg.

Labetski, A., Gerwen, S. v., Tamminga, G., Ledoux, H., and Stoter, J. (2018). A proposal for an improved transportation model in CityGML. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-4/W10:89–96.

Labetski, A., Ledoux, H., and Stoter, J. (2017). *Generalising 3D Buildings from LoD2 to LoD1*, chapter 92. University of Manchester, United Kingdom.

Ledoux, H. (2019). Personal communication. Date: 2019-04-29.

Ledoux, H., Ohori, K. A., Kumar, K., Dukai, B., Labetski, A., and Vitalis, S. (2019). CityJSON: A compact and easy-to-use encoding of the CityGML data model. *Open Geospatial Data, Software and Standards*, 4(1):4.

Luebke, D., Reddy, M., Cohen, J. D., Varshney, A., Watson, B., and Huebner, R. (2003). *Level of detail for 3D graphics*. Morgan Kaufmann Publishers.

Löwner, M.-O. and Gröger, G. (2016). Evaluation criteria for recent LoD proposals for CityGML buildings. *Photogrammetrie - Fernerkundung - Geoinformation*, 2016(1):31–43.

Mapbox (2019). Mapping guides: mapping for navigation. https://labs.mapbox.com/mapping/mapping-for-navigation/. Accessed: 2019-03-20.

Nguyen, H. H., Daniel, M., and Desbenoit, B. (2016). Realistic urban road network modelling from GIS data. In Tourre, V. and Biljecki, F., editors, *Eurographics Workshop on Urban Data Modelling and Visualisation*, pages 9–15.

Open Geospatial Consortium (2012a). *CityGML UML diagrams*.

Open Geospatial Consortium (2012b). *OGC City Geography Markup Language (CityGML) Encoding Standard*.

Open Geospatial Consortium (2016). *OGC Land and Infrastructure Conceptual Model Standard (LandInfra)*.

Ordinance Survey (2019). Our history. https://www.ordnancesurvey.co.uk/about/overview/history.html. Accessed: 2019-06-21.

OSM (2019). OpenStreetMap Wiki. https://wiki.openstreetmap.org/wiki/Main_Page. Accessed: 2019-03-12.

Oude Elberink, S. (2010). *Acquisition of 3D topography: automated 3D road and building reconstruction using airborne laser scanner data and topographic maps*. PhD thesis, Universiteit Twente, Enschede.

Ozbek, M. E., de la Garza, J. M., and Triantis, K. (2010). Data and modeling issues faced during the efficiency measurement of road maintenance using data envelopment analysis. *Journal of Infrastructure Systems*, 16(1):21–30.

Quartieri, J., Mastorakis, N., Guarnaccia, C., Troisi, A., D'Ambrosio, S., and Iannone, G. (2009). Road intersections noise impact on urban environment quality. In Bulucea, C., Mladenov, V., Pop, E., Leba, M., and Mastorakis, N., editors, *Recent Advances in Applied and Theoretical Mechanics*, pages 162–171. World Scientific and Engineering Academy and Society, WSEAS Press.

Rijkswaterstaat (2013). *Handleiding Nationaal Wegenbestand - NWB*.

Rijkswaterstaat (2017). *Nationaal Wegenbestand Achtergrondinformatie*.

Ross, L. (2010). *Virtual 3D City Models in Urban Land Management*. PhD thesis, Technischen Universität Berlin.

Scarponcini, P. (2002). Generalized model for linear referencing in transportation. *GeoInformatica*, 6(1):35–55.

Scarponcini, P. (2016). *OGC Project Document 16-000r1: LandInfra Executive Summary*.

Singh, H. and Sharma, R. (2012). Role of adjacency matrix & adjacency list in graph theory. *International Journal of Computers & Technology*, 3(1):179–183.

Soetens, M. and Tijink, G. (2019). De volgende stap in gladheidsbestrijding. Project HAS Hogeschool.

Southwest Research Institute (2018). *Basic infrastructure message development and standards support for connected vehicles applications. The MAP Message – Beyond Intersections. White Paper*.

Spielmann, M. and Scholz, R. (2005). Life cycle inventories of transport services: Background data for freight transport. *The International Journal of Life Cycle Assessment*, 10(1):85–94.

Stadler, A. and Kolbe, T. H. (2007). Spatio-semantic coherence in the integration of 3D city models. In Stein, A., editor, *Proceedings of the 5th International ISPRS Symposium on Spatial Data Quality ISSDQ 2007 in Enschede, The Netherlands, 13-15 June 2007*, ISPRS Archives. ISPRS.

Stoter, J., Beetz, J., Ledoux, H., Reuvers, M., Klooster, R., Janssen, P., Penninga, F., Zlatanova, S., and van den Brink, L. (2013). Implementation of a national 3D standard: Case of the Netherlands. In *Pouliot J., Daniel S., Hubert F., Zamyadi A. (eds) Progress and New Trends in 3D Geoinformation Sciences. Lecture Notes in Geoinformation and Cartography.*, pages 277–298. Springer Berlin Heidelberg.

Stoter, J., Roensdorf, C., Home, R., Capstick, D., Streilein, A., Kellenberger, T., Bayers, E., Kane, P., Dorsch, J., Woźniak, P., Lysell, G., Lithen, T., Bucher, B., Paparoditis, N., and Ilves, R. (2015). 3d modelling with national coverage: Bridging the gap between research and practice. In *Breunig M., Al-Doori M., Butwilowski E., Kuper P., Benner J., Haefele K. (eds) 3D Geoinformation Science. Lecture Notes in Geoinformation and Cartography*, pages 207–225. Springer International Publishing.

Tamminga, G. (2019a). Personal communication. Date: 2019-02-26.

Tamminga, G. (2019b). *A novel design of the Transportation Context of OpenTrafficSim*. PhD thesis, Delft University of Technology.

Tavares, G., Zsigraiova, Z., Semiao, V., and Carvalho, M. (2009). Optimisation of MSW collection routes for minimum fuel consumption using 3D GIS modelling. *Waste Management*, 29(3):1176–1185.

Thomson, R. C. and Richardson, D. E. (1999). The 'good continuation' principle of perceptual organization applied to the generalization of road networks. In *Proceedings of the ICA, Ottawa, Canada, Session 47B*.

TomTom (2019). TomTom Maps - Map Data. https://www.tomtommaps.com/mapdata/. Accessed: 2019-06-21.

Van den Brink, L., Krijtenburg, D., Van Eekelen, H., and Maessen, H. (2013). Basisregistratie Grootschalige Topografie: Gegevenscatalogus BGT 1.0. Technical report, Ministerie van Infrastructuur en Milieu.

Van Gerwen, S. (2019). Personal communication. Date: 2019-02-14.

Werkgroep Wegen (2018). *Programma Doorontwikkeling in Samenhang: Rapport Werkgroep Wegen 1.0*.

Werkgroep Wegen (2019a). Personal communication. Date: 2019-05-07.

Werkgroep Wegen (2019b). Centrale objectregistratie: werkgroep wegen. vervolg verkenning.

West, D. B. (2000). *Introduction to Graph Theory*. Prentice Hall, 2nd edition.

Zhang, X. and Ai, T. (2015). How to model roads in OpenStreetMap? a method for evaluating the fitness-for-use of the network for navigation. In *Advances in Geographic Information Science*, pages 143–162. Springer International Publishing.

Şerbu, C., Opruța, D., and Socaciu, L. (2014). Ranking the types of intersections for assessing the safety of pedestrians using TOPSIS method. *Leonardo Electronic Journal of Practices and Technologies*, 13(25):242–253.

A CITYGML 2.0 CORE MODULE UML DIAGRAM

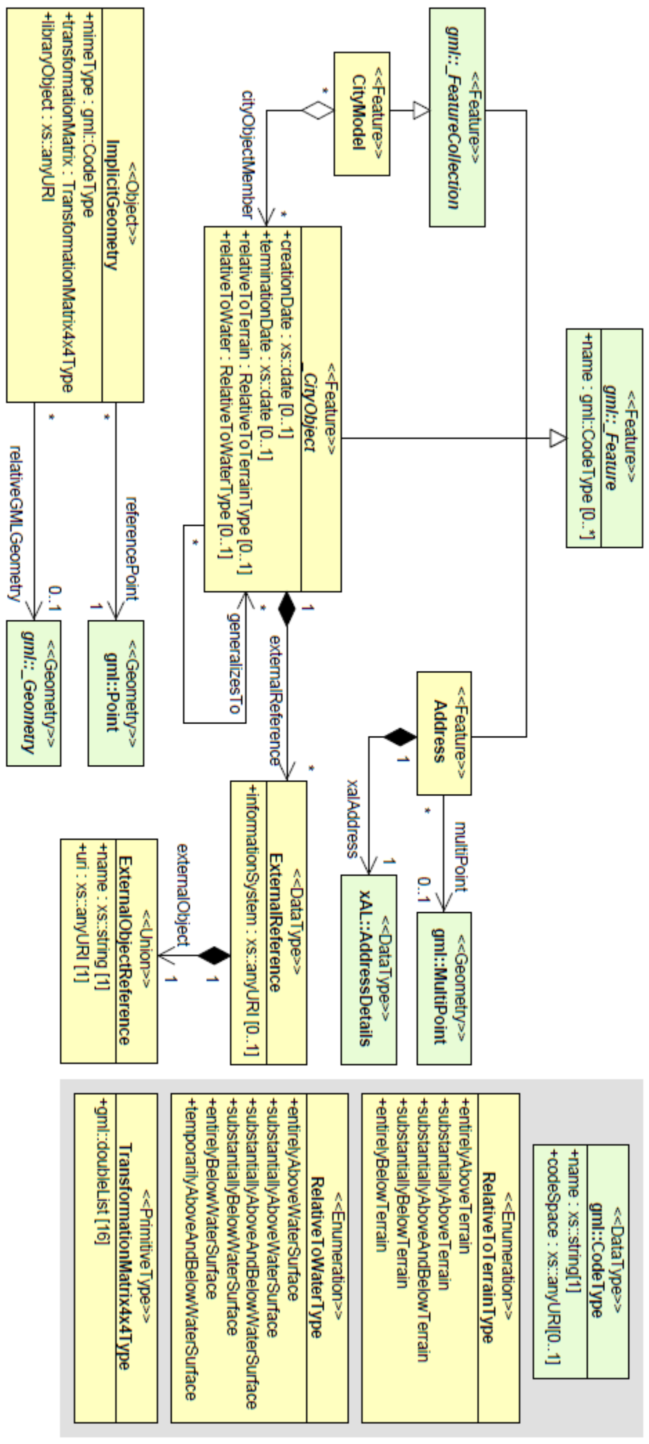


Figure A.1: UML diagram of the Core module of CityGML [Open Geospatial Consortium, 2012b].

COLOPHON

This document was typeset using LaTeX. The document layout was generated using the `arsclassica` package by Lorenzo Pantieri, which is an adaption of the original `classicthesis` package from André Miede.