UML-Based Approach to Developing a CityGML Application Domain Extension

Linda van den Brink,* Jantien Stoter† and Sisi Zlatanova†

*Geonovum, Amersfoort
†Delft University of Technology

Abstract
Recently a national 3D standard was established in the Netherlands as a CityGML Application Domain Extension (called IMGeo). In line with the Dutch practice of modeling geo-information, the ADE is developed using a model driven approach. The classes are designed in UML and automatically mapped to GML schema. The current OGC CityGML specification does not provide rules or guidance on correctly modeling an ADE in UML. This article fills this gap by studying how CityGML can be extended for specific applications starting from the UML diagrams. Six alternatives for modeling ADEs in UML are introduced and compared. The optimal alternative is selected and applied to obtain the national 3D standard. The approach was extensively discussed with international experts, who were members of both SIG3D and other working groups. As a consequence the approach was adopted by the SIG3D, the Special Interest Group 3D which, among other things, work on the 3D standard CityGML in cooperation with OGC. Therefore the approach contains many issues that can be generalized and reused by future domain extensions of CityGML. To further support this, the article formulates a model-driven framework to model CityGML ADEs. Open issues are described in the conclusions.

1 Introduction
In the last few years a broad discussion has started in the Netherlands on the need to develop a national standard for a 3D city and landscape model. The discussion is summarized in a number of papers (Stoter et al. 2010, 2011, 2013, Verbree et al. 2010). The main question was whether to develop a separate 3D model from scratch or to re-use and extend already existing 3D standards. The study on international and vendor specific 3D standards and formats has clearly revealed that CityGML is a good candidate for a national 3D standard (Stoter et al. 2012, Zlatanova et al. 2012). The OGC standard CityGML (OGC 2008, 2012a) is an application independent information model and exchange format for 3D city and landscape models. It maintains semantics, geometry, topology and the appearance of objects (Stadler and Kolbe 2007). Furthermore CityGML is supported by an increasing number of vendors by providing import/export functionalities and viewers (Groneman and Zlatanova 2009, Lapiere and Cote 2008, Rumor and Roccatello 2009). CityGML database implementations are also available (Vries and Zlatanova 2011). Considering all these aspects, it was decided to not generate a separate 3D model but to align the existing national 2D information model to CityGML.
The CityGML standard is meant as a generic standard for modeling topographic features. Domain specific information can be modeled in CityGML either by generic classes or by the definition of an extra formal schema based on the CityGML schema definitions. Such a schema is called a CityGML Application Domain Extension (ADE). The second approach allows definition of classes, their relationships and attributes and is recommended for applications that require a large number of new features to be defined. Therefore the recently developed national 3D standard in the Netherlands is also based on the ADE mechanism (Stoter et al. 2011, Van den Brink et al. 2012a). This ADE extends CityGML with the existing 2D national Information Model for large-scale Geo-information (called ‘IMGeo’).

IMGeo is modeled in UML (Unified Modeling Language) and contains object definitions for large-scale representations of roads, water, land use/land cover, bridges, tunnels etc. and their properties. It prescribes 2D point, curve or surface geometry for all objects. The original IMGeo classes were compatible with CityGML but there were also quite a few differences. The new version of IMGeo (version 2.0) is developed as specialization of CityGML. Using this approach 2D IMGeo data can be extended into 2.5D (i.e. as height surface representation) and 3D (as volumetric representation) according to geometric and semantic principles of CityGML.

The development of the ADE encountered two major challenges: designing ADE in UML, and automatically deriving the GML schema from the UML diagram. According to the Dutch practice of modeling geo-information, all information models must be represented with UML. However, the CityGML specification does not provide rules or guidance on modeling an ADE in UML. It describes how an ADE must be modeled in the XML schemas, which is not compatible with UML modeling. A complete description of the CityGML-IMGeo ADE (i.e. how the model was established) can be found in Van den Brink et al. (2012a, 2013).

This article presents the generic technical modeling principles of the ADE that were encountered during the establishment of the national 3D standard, i.e. how the UML models of CityGML can be extended to support concepts defined in a specific domain, e.g. noise (Stoter et al. 2008) and 3D cadastre (Stoter et al. 2013, Stoter and van Oosterom 2005), and how a GML application schema can be automatically generated from the UML model. We have followed strict rules and reached the solution proposed in this article with international teams involved in GML and CityGML modeling. Furthermore the proposed modeling approach is adopted by the SIG3D and our experiences are accepted as OGC best practice to standardize the developments of domain specific CityGML ADEs (Van den Brink et al. 2012b). Therefore the approach described here may serve as a generic approach. This article contains more detailed descriptions of the alternative approaches than the OGC best practice paper. In addition, the application to derive new ADEs (by giving the example of the new Dutch IMGeo standard) has been added.

The article is organized as follows. Section 2 describes the background of the study, i.e. the Dutch context in which the ADE is modeled, including the UML-based approach for modeling geo-information. Section 3 compares several alternatives for modeling an ADE in UML and selects the optimal modeling approach. Section 4 explains how the selected modeling approach has been applied to model the CityGML ADE “IMGeo 2.0”. Although IMGeo is country-specific, our approach to model a CityGML ADE in UML and to automatically generate a GML schema accordingly can be seen as a standard approach for designing a CityGML ADE. In order to make our approach reusable for future domain extensions of CityGML, in Section 5 we propose a model-driven framework for modeling CityGML ADEs in UML. Section 6 concludes on findings and topics for further research.
2 Explanation of the Dutch Context

The Dutch standardization organization for geo-information (Geonovum) follows a formal approach for modeling and implementing Information models. Such an approach is currently not provided by the SIG3D and OGC CityGML SWG, which is responsible for the development of the CityGML standard. Therefore, in the following section we further explain the generic UML modeling approach for geo-information in the Netherlands. Then the Information Model IMGeo is briefly presented and the most important features are highlighted. The last section describes the implications of our UML modeling approach for the IMGeo ADE.

2.1 Model Driven Approach

Formal representation of conceptual models for geo-information applying UML is seen as an important prerequisite of the Dutch Spatial Data Infrastructure (SDI). UML is one of the most used modeling languages by standardization bodies dealing with geo-information. Using a UML class diagram, geo-information objects can be formally described with their properties, relationships and semantic meaning. A good understanding of the meaning of objects is especially important when different organizations reuse each other’s information. Although not as elaborated as some modeling languages focused on semantics (such as OWL, RDF, etc.), UML provides sufficient means to record the meaning of objects.

In the Netherlands, the Base Model Geo-Information (NEN 3610:2011) forms a common base for domain specific information models. This national standard establishes a standard modeling method based on ISO 19101 and contains a generic semantic UML model with definitions of the most common, shared concepts in the geo-domain such as Road, Water, etc. Since NEN 3610 was first published in 2005, many domain specific information models have been developed. These domain models, such as the information model for spatial planning (IMRO) or the information model water (IMWA) extend the classes defined in NEN 3610 with properties and more specific classes. IMGeo is also one of these information models. The semantic geo-standards in the Netherlands can be viewed as a pyramid of information models (see Figure 1), which forms a common language across organizations containing definitions of concepts from different domains. Because these are all based on the shared definitions in NEN 3610, exchange of information across organizations, based on different information models in the pyramid, is possible.

In the Netherlands, a Model Driven Approach (MDA) is applied for modeling concepts and their implementation in different domains (Gaševic et al. 2006). A key point of this approach is that either the conceptual information models are independent of their technical implementation(s) or they are platform-independent (Hespanha et al. 2008, OMG 2003). This means that the technical implementations (for data storage or data exchange) are automatically created from the UML schemas of the domain models. For data exchange based on these models, Geography Markup Language (GML) is used. The technical implementations (in this case GML application schemas) are not designed and maintained separately, but are automatically derived from the UML models using the Java tool ShapeChange (Portele 2008). ShapeChange implements the UML to GML encoding rules described in ISO 19136:2007, ISO/TS 19139:2007, ISO 19118 rev 1, ISO/TS 19103:2005, and ISO 19109:2005.

2.2 Information Model Geography (IMGeo)

The Dutch Information Model Geography (IMGeo) describes how object-based, large-scale (1:1000–1:2000) topographic features must be defined to make the national exchange of this
information possible. This large-scale topographic map is created and maintained by the municipalities and several other public organizations, and financially supported by local governments and private companies. Version 1.0 of IMGeo was published in 2007 (IMGeo 2007). Version 2.0 was completed at the end of 2011 and published in February 2012. IMGeo 2.0 has a mandatory core, see Figure 2 and Appendix A. The mandatory core model contains object definitions for large-scale representations of roads, water, land use, land cover, bridges, tunnels, etc. The optional part of IMGeo allows further division of these objects into parts suitable for maintenance, and contains definitions for all kinds of city furniture and other non-mandatory classes.

Data providers such as municipalities, organizations responsible for the road, water and railway infrastructure, etc. are required by law to provide their objects that fall under the definitions of the IMGeo 2.0 core to a national ‘basic registry’ (Basisregistratie Grootschalige Topografie, BGT) where they are available for reuse.

All IMGeo object types have 2D geometry for which ISO 19107 geometry types are used. Topological rules are part of the standard, but are not modeled by ISO topological types. The most notable rule is that the complete set of polygon-objects at the surface level (height level 0) in the mandatory core must together form a complete coverage of the Netherlands without gaps or overlap.

IMGeo classes have a limited number of attributes. Most of the classes have one or two attributes to further classify the object or to indicate function. Code lists are used to provide allowed values. In addition, all classes share attributes for identification and versioning, for a reference to the data provider, for the object’s status (planned, existing, or historic) and an indication whether a possible error in the data is under investigation. In addition, all measurements are stored with metadata such as information on the accuracy. The IMGeo model also includes rules for visualization, e.g. the colour of lines and areas, type and thickness of lines, etc. These are based on a Dutch standard for web cartography. Three visualization themes are provided: one vector-based representation where large-scale topography is leading, and two where it is used as a backdrop map for other themes.
2.3 Implications of the Dutch UML Approach for the CityGML ADE

As mentioned above, it was unclear how exactly the CityGML UML diagrams can be extended. Although it is not explicitly specified in the CityGML specifications, there are examples of CityGML ADE modeling in UML (CityGML ADE 2012). Most of these ADEs extend CityGML with one or more new feature types, while our IMGeo ADE should (also) extend existing CityGML feature types with domain specific properties. In the CityGML standard, the informative Annex G shows one example of an ADE using UML diagrams: the Noise ADE. In this example, new classes are added, but also properties are added directly to the CityGML Building class. These extra properties are visually distinguishable from the Building properties defined in CityGML by using a different colour and addition of a namespace prefix ‘noise:’ to the property names.

Although this results in a visually understandable diagram, from a modeling perspective these properties are in fact part of the CityGML Building class, and not part of the ADE. In UML there is no concept that allows a class having a number of properties in one package, while other properties of the same class reside in another package. When generating a GML application schema from these diagrams using the ISO 19118 rules, the properties would be added to the CityGML namespace and not the ADE namespace. Therefore we had to look for another option, which is studied further in the remainder of this article.
3 Extending CityGML UML Diagrams with Application Specific Concepts

To understand how the problem of modeling ADEs in UML can be addressed, we first further explain the technical issues. Then we present and evaluate different alternatives for extending CityGML UML diagrams and finally select the best alternative. For clarity some basic UML terms used throughout the text are introduced first. See also Figure 3.

A set of objects with shared characteristics (attributes, associations and behavior) is modeled as a class. A class has a name, attributes, relations with other classes and operations. Classes can be abstract or concrete. Abstract classes (represented in UML diagrams with their class name in italics) are used to model super categories with characteristics shared by several subclasses (specialization). The subclasses inherit all characteristics of the superclass. Individual objects (object instances) are always members of a concrete class (NEN3610:2011).

The properties of a class and its relations to other classes are modeled as attributes or associations. Attributes have a name and a value, and their cardinality indicates how many times they may occur (zero, one or many times). Both ends of an association have a cardinality and a name.
Stereotypes are meta properties of classes, attributes and associations, used to semantically extend UML for a specific domain. In diagrams they are shown between angled brackets at the element they apply to. Another way to extend UML is by using tagged values that can be used to add properties to classes, attributes and associations. Often tagged values are used to add properties relevant for automatic code generation (ISO19103). Tagged values are usually not shown in diagrams.

Code lists and enumerations are special types of classes, which both give a list of allowed values for a specific class attribute. In the case of an enumeration the list may not be extended, while for a code list this is allowed.

Finally, hook elements are not a standard UML concept, but a concept from CityGML and XML modeling. Basically this is an abstract, extensible class property, which can be re-used for many classes with different definitions. Hook elements are explained further in the next section.

3.1 Detailed Technical Explanation of the Problem

The main challenge that we had to face was that ‘hook elements’, the CityGML approach for adding properties in XML Schema as described in OGC (2008, s. 10.13.1), do not have an equivalent in UML modeling. Consequently, we also had to design an encoding rule that establishes a mapping between the method for adding properties to CityGML classes in a UML ADE and the way this is encoded in an XML Schema according to the CityGML standard. It should be noted that CityGML 1.0 (OGC 2008) was the valid CityGML version at the moment of modeling our CityGML ADE. However, developments on this issue were still ongoing when CityGML 2.0 was established (2012) and therefore the problems of modeling an ADE in UML remain in the new version of CityGML.

The reason the CityGML approach for adding properties in XML Schema does not have an equivalent in UML modeling can be explained as follows. For creating XML schemata based on UML models we follow ISO 19118 encoding rules. According to these rules, UML classes are represented by a global element in the XML schema and UML class attributes are represented by local elements in the global element’s complex type. UML class specialization is represented by the complex type of a subclass extending the complex type of its super class. In addition, the corresponding global XML element of a UML subclass is added to the substitution group of the element representing the UML super class. In this way it may appear anywhere in the content model where the super class element is allowed. As noted in OGC (2008, s. 10.11.1), a disadvantage of this approach is that two different subclasses, each with their own extra attributes with regard to their common super class, cannot be used together. Either one or the other must always be chosen, and thus attributes from both subclasses cannot be combined. In addition, applications must have knowledge of the XML Schema to recognize the subclass as a more specific form of a super class from the CityGML standard.

Because of this, CityGML has a different approach for extending classes with additional properties. Basically, the approach chosen in CityGML is a form of an attribute substitution. Each CityGML class has a ‘hook’ in the form of an abstract, global XML child element which is referenced from each CityGML class complex type and which can be replaced an arbitrary number of times with ADE property definitions. In the ADE XML Schema, such a property is defined by extending the ‘hook’ element’s complex type and by placing the property element in the ‘hook’ element’s substitution group (OGC 2008, s. 10.11.1). This is the same technique as is normally used for mapping UML class specialization to XML Schema, as described above.
While this is perfectly normal in XML Schema, UML has no concept of sub-classing properties and there is nothing in UML or in the ISO 19100 series that is similar to global properties and/or substitution of properties. Therefore, a way of representing this in UML had to be found during the modeling of the IMGeo ADE.

Several approaches to model application specific concepts of an ADE in UML have been considered and intensively discussed within the SIG3D modeling subgroup and in e-mail discussions by the authors, SIG3D members and other participants. The alternatives for modeling ADEs in UML class diagrams, which featured in these discussions, are introduced and evaluated in the next sections.

3.2 Alternatives for Modeling ADEs in UML

Alternative 1: Use the extension possibilities of GenericCityObject and _genericAttribute (OGC 2008, s. 10.10)

Advantage: No extra modeling work is required.

Disadvantage: As noted in the CityGML standard (OGC 2008, s. 10.11), the extension would not be formally defined with names, definitions, and types, and it would therefore be impossible to validate data that uses such extensions.

Alternative 2: Add properties in the CityGML classes directly in the CityGML package instead of in an ADE package (discussed in the SIG3D). The added properties would be marked as ADE extensions in the UML model using a stereotype or tagged value.

Advantage: No subclass definition is necessary.

Disadvantage: There are two disadvantages: (a) It is in conflict with UML. Packages are namespaces and reflect governance. The CityGML packages are controlled by the CityGML SWG, an ADE package by some other authority. That authority cannot edit the CityGML packages; (b) It does not meet the requirements of the modular specification standard of OGC (OGC 2009) for very similar reasons. In particular, see Section 7.2.2.

Alternative 3: Add properties in a subclass in the ADE package but suppress this subclass from the generated XML Schema

Advantage: This approach does not violate UML, ISO 19100 series, and OGC rules.

Disadvantage: It is confusing to introduce an ADE subtype of a CityGML type although the ADE hooks provide a means to avoid subtyping; and also because in UML a subclass inherits all methods and attributes from its super class, but in this case this is not intended. The class would be marked as an ADE extension in the UML model using a stereotype or tagged value. A stereotype is preferred because it makes clear from the UML diagrams that the ADE subtype is not mapped to an XML Schema component. Tagged values are not always (and usually not) shown in the graphical notation. However, this could be viewed as violating the GML encoding rule that stereotypes are used for conceptual aspects and tagged values for encoding-related aspects.

Alternative 4: Define the ADE hook ‘_GenericApplicationPropertyOf . . .’ as a class associated with the CityGML class and add properties as subclass of the ADE hook class.

Advantage: This way of modeling provides a clear distinction between the concept of subtyping a CityGML class and extending a CityGML class with properties; any confusion is avoided.

Disadvantage: It is less clear than subtyping the CityGML class, which is a well-known way of modeling. Furthermore it is not in line with the ISO 19109 General Feature Model, where there is no concept of attribute substitution.
**Alternative 5:** Define the ADE hook ‘_GenericApplicationPropertyOf . . .’ as a class associated to the CityGML class and add properties in an abstract superclass.

**Advantage:** This avoids the problem of using a subclass while not intending inheritance of properties.

**Disadvantage:** The generalization relationship would have to be added to the CityGML class which violates basic UML and XML namespace governance rules, i.e. an ADE cannot make changes to the CityGML package which is controlled by the CityGML SWG at OGC.

**Alternative 6:** Define a general type ADEPropertyType with name, definition, type, and extendsType and maintain the added types outside the UML in a registry.

**Advantage:** This avoids violating UML rules, the General Feature Model (ISO 19109) and other rules from the ISO19100 series.

**Disadvantage:** It is necessary to maintain added feature properties outside the UML. The ADE extension cannot be completely modeled in UML. Extension with extra classes would be part of the UML model, but extension with properties would not. In addition, the feature properties that are modeled outside UML would still have to be included somehow in the generated XML Schema.

3.3 Conclusion on the Alternatives: Best Approach

After comparing the advantages and disadvantages of the above alternatives, Alternative 3 has been selected as the best option for the IMGeo ADE. Both the authors and the SIG3D decided to implement this option. This approach defines the to-be-added properties in subclasses in the ADE package but suppresses these subclasses from the generated XML Schema. There are several reasons why we have chosen this approach.

Firstly, conceptually IMGeo is an extension of CityGML and therefore defining the IMGeo classes as subclasses of CityGML classes and adding the extra properties to these subclasses is appropriate. Another aspect in favor of this alternative is that the use of sub classing is understandable by people with basic knowledge of UML class diagrams. This is an important requirement of the IMGeo UML model. In addition, this approach conforms to relevant rules of UML, the ISO 19100 series and OGC unlike most of the alternatives described previously. Finally this approach is the most in line with the current geo-information modeling approach in the Netherlands.

The fact that in the XML Schema implementation the subclasses are omitted, is seen as a technical implementation choice to allow the combining of properties from different ADEs. While this is a valid reason on the technical level, it is not taken to mean that in the conceptual UML model sub classing should also be avoided.

4 Modeling IMGeo as CityGML ADE

This section presents the procedure that was followed to model IMGeo as an ADE of CityGML in UML using the selected approach. In the following sections we will elaborate on the modeling of classes, subclasses, code lists, geometry and topology. Finally we explain how the XML Schema can be automatically generated.

4.1 Modeling IMGeo Classes as Subclasses of CityGML Classes

Since CityGML 1.0 (and also 2.0) is only available as XML Schema, the first step was to recreate the UML model in the modeling tool Enterprise Architect, based on OGC (2008). In the
next step all IMGeo classes were modeled as subclasses of CityGML classes. Using the selected modeling approach, these subclasses get the same class name as the CityGML class they are extending. The stereotype <<ADEElement>> is assigned to the subclasses and the specialization relation is marked with a stereotype <<ADE>>. Having a stereotype also on the specialization relation, as well as the names of these stereotypes were proposed by members of the SIG3D during the discussion on this issue. The stereotypes mark these classes as special subtypes that only add properties to the CityGML class, and accordingly no XML component for these classes will be created in the XML Schema. For documentation purposes, a Dutch translation of the class name is added as an alias. For all CityGML classes, relevant for IMGeo, a subclass is created, adding at least a 2D geometry property to all classes.

Figure 4 shows an example in the IMGeo ADE of a subclass TunnelPart that contains additional properties compared to the equivalent CityGML class (2D geometry and LOD0 geometry properties). The yellow classes are classes from the CityGML Tunnel package. The <<ADEElement>> TunnelPart is a class defined in the IMGeo ADE package as a subclass of CityGML TunnelPart class. The Dutch alias is shown between brackets on the class diagram.

Applying this inheritance structure gives the domain-specific information model the same structure as defined by the CityGML model, see Appendix A.

To identify equivalent concepts that can be modeled via this sub classing method, first a conceptual mapping was made between CityGML and IMGeo classes. These mappings compared the concepts at the semantic level, i.e. independent of the LOD at which the concept appears in CityGML. The CityGML Levels of Detail (LOD) concept is used to model objects with different accuracy for different purposes between LOD0 (terrain) to LOD4 (interior); where LOD1-LOD3 also represent volumetric properties. Some classes only get spatial representations at higher LODs in CityGML and because we also wanted to take these concepts into account, we compared the IMGeo and CityGML classes independent of the LOD at which it appears.

Obviously, a 1-to-1 mapping to an equivalent CityGML class could not be found for every IMGeo class. Two solutions are possible. The first option, which is preferred and therefore applied as much as possible, remodels the IMGeo concept so that an equivalent CityGML class can be found. For IMGeo this is for example used for Vegetation that models any vegetation-related concept (in IMGeo 1.0 divided over several classes) and AuxiliaryTrafficArea meant for road segments that are not used for traffic, such as verges (in IMGeo modeled under the classes Road or Land Use).

If it is not possible to remodel the concept into a CityGML class, CityGML is extended with a new class, as a subclass of one of the CityGML classes. In this case, because a whole new class is added and not just properties to an existing CityGML class, the hook mechanism is not used; instead the class is modeled as a subclass of a CityGML class with stereotype <<featureType>> and not suppressed from the XML Schema. Figure 5 shows an example in the IMGeo ADE of a class, which is not available in CityGML but needed in IMGeo. The class ‘OverigeConstructie’ (OtherConstruction) is a class to represent man-made constructions other than buildings, bridges and tunnels. Examples are water management constructs such as pumping plants, locks, and weirs but also wharfs, fences, loose-standing walls, high-tension line towers, wind turbines, and so on. It is modeled as a <<featureType>> subclass of the CityGML class _Site (with a Dutch class name) that is not suppressed from the XML Schema. Note that this class does not get the <<ADEElement>> stereotype; this is only used for classes that model extension properties on an existing CityGML class and that need to be suppressed from the XML Schema. The class has its own properties that are modeled like CityGML classes, with implicit geometry on different LODs as well as 2D and 3D geometry up to LOD3.
In 2D IMGeo, the parts of larger built objects like buildings, tunnels and bridges are modeled, but the whole object is not. For the 2D application of IMGeo this is not necessary. When mapping these classes (e.g. TunnelPart) to CityGML, we selected the concept in CityGML that was the closest match semantically, not to the abstract superclass (e.g. Tunnel), even though this could make the model more flexible. When creating 3D IMGeo data from a 2D dataset, a problem occurs because in CityGML the whole object must be modeled as well as the parts, while in the 2D IMGeo data the geometry of the whole objects (buildings, bridges) is not present. This is a gap in IMGeo which remains to be addressed.
Additional properties in an ADE must have a globally unique name. To cope with this restriction in the ADE approach, all additional properties in the IMGeo ADE have a property name containing the class name. In our approach this was accomplished manually in the UML model, but it is feasible to generate globally unique names for properties automatically when the XML Schema is inferred from the UML model, for example by appending the class name.

4.2 Code Lists in the ADE

CityGML provides code lists to allow predefined values for the CityGML attributes. However, the CityGML-IMGeo ADE makes use of national classification code lists instead of the CityGML code lists, because the national lists are specifically suited to the Dutch context and contain a definition for each concept, approved by the Dutch organizations involved. Other reasons for not using the CityGML code lists are that IMGeo favors Dutch language code lists and that the CityGML standard does not provide definitions for the code list values, which makes it hard to decide which value to use.
The current version of CityGML (2.0), which was not yet published at the time of establishing IMGeo, does allow extension and replacement of code lists. However for the IMGeo ADE there is no need to map the Dutch code lists to the CityGML code lists, as these are non-normative and software does not check on code list values nor process them in specific ways.

Both CityGML and GML do not provide a normative way to structure code lists. Prominent choices are GML dictionary and Simple Knowledge Organization System (SKOS; W3C 2009a).

GML dictionaries can be used to collect sets of definitions or references to definitions (OGC 2007). In GML, these can be used, for example, to define coordinate reference systems or units of measure. The GML dictionary model implements a simple nested hierarchy of definitions, but is not intended to represent complex interrelating sets of definitions such as taxonomies, thesauri or ontologies. A definition in a GML dictionary has an identifier, and possibly one or more names and inline descriptions or links to descriptions.

GML 3.3 states that “Definition and Dictionary encoding is part of the GML schema as a stop-gap, pending the availability of a suitable general purpose dictionary model” (OGC 2012b). Now that new standards have matured (in particular RDF, OWL and SKOS) the use of GML dictionaries is deprecated in GML 3.3 for generic definitions and code lists. The best practice, emerged from the semantic web community, is now to use URIs for referring to items in vocabularies. The Resource Description Framework model (RDF) (W3C 2004) is in line with this best practice.

SKOS is based on RDF and contains a common model for vocabularies, thesauri, and taxonomies. It is more lightweight than Web Ontology Language (OWL) (W3C 2009b), which is a formal knowledge representation language. Items in a SKOS vocabulary are called ‘concepts’ and can have several labels as well as broader, narrower, and non-hierarchical, associative relations with other concepts. Concepts can be part of concept schemes (vocabularies) and can be grouped in collections.

GML dictionary was considered but not selected, because these are deprecated in GML 3.3, while SKOS adoption is growing in the geo community. SKOS was therefore selected.

The code lists are maintained in the UML model and XML structured code lists can be generated from the UML using a ShapeChange customization which allows generation of SKOS-encoded code lists from UML classes with a <<code list>> stereotype. The disadvantage of maintaining the code lists in the UML model is that the latter needs to be updated in case the code lists need an update. For IMGeo the code lists are considered as part of the standard and allowed to change only when a new IMGeo version is published.

The SKOS code lists will be published in a national, publicly available registry, which also contains the IMGeo XML Schema. Each code list and code list value is accessible via its own URL. Code list validation can be accomplished using standard XML techniques such as Schematron constraints (ISO/IEC 19757-3:2006(E)). Further work is needed to assess how the IMGeo code lists are best represented and structured in SKOS. Open questions are whether each code list is encoded as a SKOS concept scheme or a collection, how to construct valid URIs for all code list values and whether the code values are stored in one SKOS file for all code lists, one per code list, or one per code list value.

Figure 6 shows the IMGeo code list for classification of WaterBody in UML (attribute class) and a fragment of how this could be encoded in SKOS as a concept scheme, including each code list value’s definition. The UML class has a stereotype <<code list>>. The first five values are marked with a stereotype <<BGT>>, which means these are part of the mandatory core of IMGeo. The others are optional further classifications. In the SKOS fragment the first value from the UML code list is shown. Its unique URI is declared in the rdf:about attribute.
The code list value itself can be found as a SKOS prefLabel, and its definition as a SKOS definition. The inScheme property declares the concept as a member of the concept scheme Type-Water for WaterBody.class.

4.3 Geometry and Topology in the IMGeo ADE

For the use of geometry types and the LOD concept in the IMGeo ADE, we formulated the following guidelines:

- The higher LODs will always be derived from IMGeo data. Therefore all LODs have the same x, y accuracy as the 2D IMGeo data. This assures consistency between all LODs starting from the 2D geometry. Which LOD is used depends entirely on the required level of detail in the third dimension rather than the positional accuracy as mentioned in CityGML specifications.
- Extra attributes defining geometry types are added to the subclasses to support the full range of geometries: 2D geometry for all subclasses (not modeled in CityGML) and the LOD0 geometry if not present in the equivalent CityGML super class. With the LOD0 representations the footprint and the exact location of the footprint in the terrain is known. Consequently 3D representations of those objects (LOD1 and higher) can easily be placed on the terrain assuring that they do not float in the air or disappear in the

Figure 6  UML code list for Waterbody, attribute typeWater

The code list value itself can be found as a SKOS prefLabel, and its definition as a SKOS definition. The inScheme property declares the concept as a member of the concept scheme Type-Water for WaterBody.class.
ground. In addition this approach supports 2D, 2.5D and 3D representations of objects in an integrated manner. For some classes other geometry types were used than modeled in CityGML. In our approach Water and Road objects are represented by surface geometry at LOD0 and Railway is represented by the curve geometry at LOD0, while CityGML represents those classes with the GeometricComplex at LOD to accommodate networks. The changes were made to be compliant with geometry types in the previous version of IMGeo.

- Special attention was given to the link between buildings and terrain. Two approaches can be followed:
  - The buildings always have horizontal foundations in the terrain and can sink, but the above and below surface geometries are modeled separately. The 2D representations of the buildings represent the building geometries at surface level, which can straightforwardly be extended into 2.5D. This modeling makes the Terrain Intersection Curves (TIC) as a solution proposed in CityGML redundant. This approach was used in CityGML-IMGeo.
  - The buildings always have horizontal foundation and may have underground parts (Zlatanova et al. 1996). This means that the building can sink under the surface and the TIC for buildings has to be maintained to ensure the consistency with the terrain surface.

- For applying the CityGML Digital Terrain Model three different approaches can be followed:
  - Terrain is represented by a surface by regular or irregular grid. The topographic objects are not integrated in the terrain (the current concept of CityGML)
  - Terrain is represented by constrained TINs in which the boundaries of objects form the constraints. This approach ensures the consistency between objects and the terrain surface. The terrain is still present in the model.
  - Terrain is not represented as a separate object, i.e. all objects on the surface level (such as roads, land use, etc.) incorporate the terrain curvature in their representation (as in Emgård and Zlatanova 2008). This approach is used in CityGML-IMGeo and supported in CityGML.

For topology the following principles were copied from IMGeo 1.0. Although not expressed in the formal model, the IMGeo standard contains a general rule that the 2D objects at the surface level must form a topological structure of the surface of the Netherlands without gaps or overlaps. Since the 2D IMGeo is a topologically correct model, the 2.5D surface (LOD0) should support this as well: all objects at the surface level have a representation at LOD0 (= 2.5D surface) which together form a 2.5D topological structure for those objects located at the surface level. Objects that are located above and below the surface can also be placed in the third dimensional space with their LOD0 2.5D representation. An important requirement here is the connection to the 2.5D DTM that represents the surface. This may require adding new 2D boundaries for adding more variance in 3D or extra 2.5D surfaces to the structure at the surface to avoid gaps (see Stoter et al. 2011 for an example).

4.4 Generating XML Schema from the UML ADE

The Java tool ShapeChange is used to generate an XML Schema (GML application schema) from the ADE defined in UML. As mentioned before, ShapeChange implements the UML to GML encoding rules described in ISO 19136, ISO 10118, and ISO 19109. ShapeChange was
only used to generate the XML Schema for the IMGeo ADE, not to generate the XML Schemata for CityGML. These schemata are already publicly available and the generated ADE schema only needs to correctly import the CityGML Schemata, which ShapeChange does based on dependencies between UML packages. The CityGML 2.0 XML Schema as published by OGC is not generated from the UML model and this would not be possible without changing the current UML model, because it does not adhere to several aspects of the relevant encoding rules. It is, however, beyond the scope of the article to discuss this.

ShapeChange was modified to add a custom encoding rule for classes with the "<<ADEElement>>" stereotype. These classes are suppressed from the GML Application schema, while their properties are added to the ADE namespace as substitutes for the CityGML "_GenericApplicationPropertyOf<Featuretypename>" hooks as described in CityGML 2.0, section 10.13.1.

The generated schema fragment below shows that the IMGeo ADE extension class of TunnelPart (see Figure 4) is suppressed from the XML Schema, while the extra properties (only one example shown below) are implemented according to the CityGML extension hook mechanism. The IMGeo class OverigeConstructie (see Figure 5) on the other hand, is a newly added class, and not suppressed from the XML Schema.

```
<element name="geometrie2dTunneldeel" substitutionGroup="tun:_GenericApplicationPropertyOfTunnelPart" type="gml:SurfacePropertyType"/>
...  
<element abstract="true" name="OverigeConstructie" substitutionGroup="cit:_Site" type="imgeo:OverigeConstructieType"/>
<complexType abstract="true" name="OverigeConstructieType">
<complexContent>
<extension base="cit:AbstractSiteType">
<sequence>
  <element maxOccurs="unbounded" name="plaatsbepalingspuntOverigeConstructie" type="imgeo:PlaatsbepalingspuntPropertyType"/>
  ...
</sequence>
</extension>
</complexContent>
</complexType>
```

Fragment of the generated GML application schema

4.5 Creation of IMGeo 2.0 Data

The organizations responsible for IMGeo data are now in the process of creating IMGeo 2.0 compliant data based on existing large scale data. IMGeo 2.0 compliant test data has been generated to show how the model works when applied to data, see Figure 7. The viewer used is FZK Viewer.

5 Model-driven Framework for Developing CityGML ADE

The approach to modelling an ADE of CityGML in UML as presented in this article contains many points that can be generalized and reused by future domain extensions of CityGML.
Therefore, in this section we propose a generic model-driven framework for developing a CityGML ADE in UML based on our experiences. It should be noted that this approach is not only CityGML specific. For example the German standard XPlanGML also supports the ADE mechanism. Therefore XPlanGML ADEs can be developed in the same way (Benner et al. 2012).

The steps to be followed when developing an ADE can be summarized as follows:

1. **Select a formal modeling language, e.g. UML to represent all the classes.** In the Netherlands UML is widely used for modeling in the geo-domain. The Basic Schema for Geo-Information (NEN3610:2011) has been modeled in UML since 2005, and many other information models in the geo domain have followed suit. Reasons for selecting UML include:
   a. Its visual modeling approach makes the model suitable for communication with stakeholders (users, software developers);
   b. It is possible to generate XML schema and documentation directly and automatically from the model, i.e. it supports the MDA approach;
   c. It is a formal language which can unambiguously express the structure and rules of an information model;
   d. It is an international standard; and
   e. It is used extensively in ISO 19xxx standards which are relevant to the GIS domain.

In this step, all object types are created as UML classes in UML class diagrams with the appropriate properties and relations. A UML model of CityGML is also needed. During our work on the CityGML ADE IMGeo, the CityGML classes also had to be

![Figure 7 Visualization of CityGML-IMGeo encoded data: CityGML LOD2](image)
created, because a UML model in which these classes were already fully defined, and which could be imported in the UML tool of choice, was not available. The SIG3D group is currently completing the CityGML part of our UML model and plans to make it publicly available so that future ADEs can build on this work.

2. **Define correspondences between semantic classes of the application and CityGML generic classes.** In this step, a first study is conducted on the semantic correspondence between the classes from the application domain and CityGML. The mapping is not yet formally defined, but only intended to list all the correspondences, mainly on the class level. In our approach we have created only informal drawings and text in this step. For all classes in the application model, an equivalent class in CityGML was searched based on its name, description, geometry representations and properties. More details on schema mapping can be found in Letho (2007). Based on this exercise, the relevant CityGML classes were identified. Figure 8 shows the mapping of two other Dutch information models to CityGML.

3. **Decide which subclasses should be extended.** In the current approach we use specialization relations to define this correspondence. We distinguish two types of correspondence. Either the class is semantically the same as the corresponding CityGML class, and only adds properties, or the class is semantically a subclass of the corresponding CityGML class. In both cases specialization is used, but in the first case the specialization relation is marked with a stereotype <<ADE>>, the application specific class is marked with a stereotype <<ADEElement>>, and its class name is the same as the corresponding CityGML class name. The UML class in this case is only a placeholder for additional properties that are attached to the ‘hooks’ CityGML provides for extension of its classes. In the second case, the specialization relation receives no stereotype, the application specific class receives the stereotype <<FeatureType>> (ISO19xxx) and the class name is different from the corresponding CityGML class name. When classes in the source domain model do not correspond exactly with CityGML classes, e.g. some instances of the class correspond with a CityGML class but other instances do not, then the classes in the domain model should be harmonized with CityGML.

4. **Define application code lists if necessary.** Code lists provide additional semantic richness in CityGML. CityGML has non-normative code lists for most properties that add semantic detail. As of CityGML 2.0 it is possible to replace these code lists with others. Reasons for using our own code lists include the need to provide non-English code lists and a need for clear definitions of each code list value (not provided in CityGML). To make use of national classification code lists (which will be so specific in most cases that those national lists are preferred over the CityGML code lists) mapping tables between the code lists/code list values should be provided.

5. **Optional: define a geometry representation for each class for each applicable level of detail.** This step is optional because often only non-geometric additional properties are defined in ADE-classes, the geometric representations provided by the CityGML base class being sufficient. In the case of IMGeo, additional geometry representations are required. It was already defined which 2D geometry types must be used for each class. Which levels of detail apply for which classes, and which geometry representations they have in those LODs, should be decided based on user requirements, rather than the positional accuracy as mentioned in CityGML specifications. That is, for each class it should be decided whether it should be represented with full solids or if a 2.5D representation is sufficient. Extra attributes defining geometry types could be needed to support the full range of geometries for every class: 2D geometry (not modeled in CityGML) and the LOD0
Planologisch gebied (planning area)

Figure 8 Examples of mapping between domain models and CityGML: (a) Information model for spatial planning; and (b) Information model for cultural and historic objects
geometry if not present in CityGML (as footprints). Special attention should be given to the link between buildings and terrain. Finally, terrain object as given in CityGML should be carefully considered (Van den Brink 2012a).

6. **Make a decision on which LOD should be topologically correct.** The most notable rule in IMGeo is that the complete set of polygon-objects at surface level (height level 0) in the mandatory core must together form a complete coverage of the Netherlands without gaps or overlap in 2D. If the 2D model contains a topological structure, the 2.5D surface (LOD0) should support this as well: all objects at surface level have a representation at LOD0 (= 2.5D surface). Objects that are located above and below the surface can also be placed in the third dimensional space with their LOD0 2.5D representation. An important requirement here is the connection to the 2.5D DTM that represents the surface. This may require adding new 2D boundaries.

6 Conclusions and Further Research

This article presents a model driven approach to generate a CityGML ADE starting from UML schema. Several alternatives were proposed, investigated and discussed within a group of international experts (SIG3D members and others). The most beneficial approach to model a CityGML ADE in UML was applied for the CityGML ADE IMGeo (i.e. the Dutch national 3D standard on large-scale geo-information). The main principle of the selected approach is that all classes of IMGeo are modeled as subclasses of CityGML classes. If these subclasses only add properties to existing CityGML classes, they get the same class name as the CityGML class they are extending. The stereotype <<ADEElement>> marks these classes as subtypes that only add properties to the CityGML class, and accordingly no XML component for these classes will be created in the XML Schema. Following the design approach for CityGML ADE IMGeo, we have established a model-driven framework for developing CityGML ADE.

In the development of the CityGML ADE IMGeo 2.0 a number of topics are identified that require further research. Firstly, more research is needed to understand how this model works in practice including the consequences of this new modeling method for IMGeo when used for both 2D and 3D datasets, e.g. how to preserve the links between the different LODs and how to upgrade 2D LOD to higher LODs. Secondly, knowledge is required on the ability to use the CityGML ADE IMGeo UML model to generate working database schemes. Thirdly, more research is required on how to handle every type of correspondence between CityGML and domain classes, for example how to deal with overlapping classes that are neither super- or subclasses of a CityGML class. Finally, more research is needed concerning the creation and management of CityGML-IMGeo data. Which methods can be used to generate CityGML-IMGeo data? How should this data be validated and maintained? How can 2.5D topology be created and maintained?

These open issues are currently being studied in a follow-up project of the 3D pilot NL. The first phase finished in June, 2011 and has been reported in Stoter et al. (2011). Since October 2011 over 100 organizations (over 300 persons) are contributing to the six activities of the 3D Pilot NL. The activities related to learning more about the UML modeling approach for ADEs are the generation of 3D IMGeo example data (several levels of detail and several classes) and the design and implementation of a 3D validator that tests whether both the semantics and the geometry of the data are compliant with the standard.

In conclusion, this is the first study on extending the UML diagrams of CityGML for specific domains. Since the OGC CityGML specification does not provide rules or guidance on
correctly modeling an ADE in UML, and this article reflects the combined efforts of international experts, we believe that this article may serve as best practice for future ADEs to be modeled in UML.

References


CityGML 2012 CityGMLADEs. WWW document, http://www.citygmlwiki.org/index.php/CityGMLADEs


© 2013 John Wiley & Sons Ltd Transactions in GIS, 2013, 17(6)


Van den Brink L, Stoter J, and Zlatanova S 2012a Modeling an application domain extension of CityGML in UML. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 38-4/C26: 11–4


© 2013 John Wiley & Sons Ltd

*Transactions in GIS*, 2013, 17(6)
Appendix A Overview of main classes in CityGML ADE for IMGeo

CityGML Core
- CityObject
  - Building
    - BuildingPart
  - Transportation
    - TrafficArea
  - OverigeConstructie
    - Railw
      - Way
    - TunnelPart
  - LandUse
    - FunctioneelGebied
    - RegistratiefGebied

Transportation::TrafficArea
- Transportation::TransportationObject
  - AuxiliaryTrafficArea
  - Railway

CityGML Core::CityObject
- Bridge
  - BridgeConstructionElement

Transportation::Overbruggingsdeel
- BridgeConstructionElement

Vegetation::VegetationObject
- PlantCover

LandUse::OngeclassificeerdObject
- RegistratiefGebied
- FunctioneelGebied

© 2013 John Wiley & Sons Ltd