Concept for the generation of an ISO-compliant UML model and generation of a GML application schema for DATEX II to improve interoperability

Konzept für die Erzeugung eines ISO-konformen UML-Modells und Generierung eines GML-Applikationsschemas für DATEX II zur Verbesserung der Interoperabilität

Berichte der
Bundesanstalt für Straßenwesen

Fahrzeugtechnik Heft F 137

Concept for the generation of an ISO-compliant UML model and generation of a GML application schema for DATEX II to improve interoperability

Konzept für die Erzeugung eines ISO-konformen UML-Modells und Generierung eines GML-Applikationsschemas für DATEX II zur Verbesserung der Interoperabilität

von

Dirk Lauber
Dr. Enrico Steiger
Markus Kopka
Florian Lapolla

ISB Institut für Software-Entwicklung
und EDV-Beratung AG
Karlsruhe

Jörg Freudenstein
Dr. Josef Kaltwasser

AlbrechtConsult GmbH
Aachen

Berichte der Bundesanstalt für Straßenwesen

Fahrzeugtechnik Heft F 137

Impressum

Bericht zum Forschungsprojekt 01.0189
Konzept für die Erzeugung eines ISO-konformen UML-Modells und Generierung eines GML-Applikationsschema für DATEX II zur Verbesserung der Interoperabilität

Fachbetreuung
Holger Drees

Referat
Vernetzte Mobilität

Herausgeber
Bundesanstalt für Straßenwesen
Brüderstraße 53, D-51427 Bergisch Gladbach
Telefon: (0 22 04) 43 - 0

Redaktion
Stabsstelle Presse und Kommunikation

Druck und Verlag
Fachverlag NW in der
Carl Ed. Schünemann KG
Zweite Schlachtpforte 7, D-28195 Bremen
Telefon: (04 21) 3 69 03 - 53
Telefax: (04 21) 3 69 03 - 48
www.schuenemann-verlag.de

ISSN 0943-9307
ISBN 978-3-95606-552-1

Bergisch Gladbach, März 2021

Kurzfassung – Abstract

Konzept für die Erzeugung eines ISO-konformen UML-Modells und Generierung eines GML-Applikationsschemas für DATEX II zur Verbesserung der Interoperabilität

Der nationale Objektkatalog für das Straßen- und Verkehrswesen (OKSTRA) und die DATEX II-Spezifikation auf europäischer Ebene sind zwei essentielle Standards für den homogenisierten Austausch von statischen und dynamischen Verkehrsdaten. Im Rahmen der Harmonisierung und Referenzierbarkeit beider Standards wurde ein Konzept für die Erzeugung eines ISO-konformen GML-Applikationsschema für DATEX II erstellt, in Anlehnung an das entsprechende Vorgehen bei OKSTRA zu GML. Dabei ist eine automatisierte Vorgehensweise für die Transformation des existierenden DATEX II-UML-Modells in ein ISO 19103-konformes Modell erarbeitet worden. Als Transformationsbasis diente die DATEX II-Version 3.0. Ausgehend von dem DATEX II-Plattform Independent Model (PIM) und dem entsprechenden DATEX II-UML-Profil – beide in Enterprise Architect verfügbar – bestand die erste Aufgabe darin, Regeln für eine Transformation in ein äquivalentes GML-Applikationsschema zu erstellen. Durch die Nutzung von Automatisierungs- und Codierungsfunktionen im Enterprise Architect ist es möglich, das komplette DATEX II-Datenmodell umzuwandeln. Um ein GML-Applikationsschema aus dem ursprünglichen PIM-Modell zu erstellen, war das Open-Source-Tool ShapeChange, das sich über eine API mit Enterprise Architect (EA) verbindet, die erste Wahl. Im Zuge des Konzepts konnte allerdings ShapeChange nicht ausreichend konfiguriert werden, um ein fehlerfreies Ergebnis generieren zu können. Als Alternative wurde letztendlich die eingebaute Funktionalität des Enterprise Architect verwendet. Durch den durchgängigen Einsatz des Enterprise Architect konnten reproduzierbare Ergebnisse und ein einfach zu handhabender Mechanismus erreicht werden, so dass jede Art eines zukünftigen DATEX II-Datenmodells einfach übertragen werden kann (auch Level-B-Erweiterungen). Als Ergebnis dieses Projekts ist ein Prototyp unter Nutzung der erarbeiteten Transformationslogik implementiert worden, der ein valides GML Applikationsschema für DATEX II darstellt. In einem abschließenden Schritt ist das abgeleitete GML-Applikationsschema validiert worden, um eine ISO 19103 und ISO 19136-Konformität sicherzustellen. Im Ergebnis der prototypischen Implementierung ist aufgezeigt worden, dass eine Überführung des DATEX II-UML-Modells in ein GML-Applikationsschema möglich ist.

Concept for the generation of an ISO-compliant UML model and generation of a GML application schema for DATEX II to improve interoperability

The National Object Catalog for Roads and Transportation (OKSTRA) and the DATEX II specification at European level are two essential standards for the homogenised exchange of static and dynamic traffic data. As part of the harmonization and referencing of both standards, a concept for the generation of an ISO-compliant GML application schema for DATEX II based on the OKSTRA GML procedure has been developed. An automated procedure was implemented for the transformation of the existing DATEX II UML model into an ISO 19103 compliant model. The transformation basis was the Version 3.0 of DATEX II. Based on the DATEX II Platform Independent Model (PIM) and the corresponding DATEX II UML profile – both available in Enterprise Architect – the first task was to create rules for a transformation into an equivalent GML application schema. By using automation and coding features in the Enterprise Architect, it is possible to convert the entire DATEX II data model. To create a GML application schema from the original PIM model, the open source tool 'ShapeChange', that connects to Enterprise Architect (EA) through an API, was the first choice. In the course of the concept ShapeChange could not be configured adequately to be able to generate a faultless result. As an alternative, the built-in functionality of the Enterprise Architect was finally used. The aim of the consistent usage of the Enterprise Architect is to obtain reproducible results and an easy-to-use mechanism so that any type of future DATEX II data model can be simply transferred, including so called Level B extensions. As a result of this project, a prototype has been implemented using the elaborated transformation logic, which is a valid GML application schema for DATEX II. In a final step, the derived GML application schema has been validated to ensure compliance with ISO 19103 and ISO 19136. As a result of the prototype implementation, it has been shown that it is possible to convert the DATEX II UML model into a GML application schema.

Kurzbericht

Konzept für die Erzeugung eines ISO-konformen UML-Modells und Generierung eines GML-Applikationsschemas für DATEX II zur Verbesserung der Interoperabilität

1 Einleitung

In den Bereichen statische Straßendaten und Intelligente Verkehrssysteme (ITS) (Dynamische Verkehrsdaten) haben sich historisch zwei parallele Ansätze zur Modellierung und zum Austausch von Daten und Informationen entwickelt:

Im Straßenbetrieb ist dies der nationale Standard OKSTRA als Basis für den statischen Datenaustausch. Der internationale Standard für den Austausch von Geodaten ist die GML (Geography Markup Language) und dient als Grundlage für international standardisierte und OGC-konforme Geodienste wie GDI-DE und INSPIRE. Um diese Geodienste zu verwenden, wird das Applikationsschema OKSTRA-GML verwendet.

Auf der ITS-Seite werden die europäische Norm DATEX II (CEN/TS bzw. CEN/EN 16157) und die entsprechenden Ersetzungsmechanismen zum Austausch von verkehrs- und fahrzeitbezogenen Daten verwendet, z. B. über den deutschen Mobility Data Marketplace MDM. Bisher gibt es kein GML-Applikationsschema für DATEX II.

Die Spezifikation für die vorrangige Maßnahme B (Delegierten Verordnung (EU) 2015/962 der Kommission über die Bereitstellung von EU weiten Echtzeit-Verkehrsinformationsdiensten) im Rahmen der europäischen ITS-Richtlinie sieht nun erstmals eine gemeinsame Betrachtung von statische Straßendaten und dynamische Verkehrsdaten vor. Diese Daten sollten gemeinsam über einen nationalen Zugangspunkt (MDM in Deutschland) bereitgestellt werden. Hierfür ist die Verwendung der internationalen Standards DATEX II und INSPIRE geplant.

Da es kein GML-Applikationsschema für DATEX II gibt, ist die direkte gemeinsame Nutzung von ITS- und Straßenverkehrssystemen, sowie die Integration von DATEX II-basierten Daten in nationalen und internationalen Geodienste begrenzt und zeitaufwendig.

Projektauftrag ist die Entwicklung eines Verfahrens, mit dem ein GML-Applikationsschema auf Basis des DATEX II-UML-Modells in weitgehend automatisierten Prozessschritten erstellt werden kann. Voraussetzung dafür ist ein Regelwerk für eine Modelltransformation, mit der das vorhandene DATEX II-UML-Modell in ein ISO-19103-kompatibles UML-Modell umgewandelt werden kann. Zielstellung ist es, eine Grundlage für die Einbeziehung dieser Aspekte in den DATEX II-Spezifikationsprozess zu schaffen und so zur Verbesserung der Interoperabilität von Verkehrsdaten, statischen Straßendaten und allgemeinen Geodaten beizutragen. Die Ergebnisse des Projekts sollen den weiteren DATEX II-Spezifikationsprozess unterstützen[1].

2 Analyse des DATEX II-Metamodells

2.1 Übersicht zu DATEX II

DATEX II ist der europäische Standard für den Austausch von dynamischen Verkehrsinformationen. Die Weiterentwicklung des Standards und seines Datenmodells wird derzeit durch eine Programmunterstützung der Europäischen Union finanziert, in mehreren delegierten Verordnungen – als Ergebnis der ITS-Richtlinie 2010/40/EU – die Europäische Union fördert somit den DATEX II-Standard als ein Teil der ITS Datenkommunikation.

DATEX II ist derzeit in der Serie CEN/EN 16157 (Teile 1, 2, 3 und 7), sowie in weiteren Teilen als technische Spezifikation (Teile 4, 5, 6 CEN/TS 16157) standardisiert. Eine Migration dieser letztgenannten Teile in die europäische Norm ist im Gange.

2.2 DATEX II, Version 3.0

In Übereinstimmung mit der Veröffentlichung der ersten Teile der oben genannten EN-Serie, wurde eine neue Hauptversion von DATEX II im Frühsommer 2018 eingeführt. Die Version 3.0 von DATEX II. Diese Version enthält eine Reihe von sichtbaren, aber auch „versteckten“ Änderungen und ist nicht

[1] https://www.datex2.eu/support/tooling

mehr rückwärtskompatibel zu den Versionen zuvor (z. B. Version 2.3 und älter).

In diesem Projekt ist sich früh mit der bevorstehenden Veröffentlichung von DATEX II, Version 3.0 und die möglichen Implikationen auseinandergesetzt worden. In dieser Projektinitialphase musste eine Entscheidung zwischen der Nutzung der aktuellen DATEX II und die kommende Version getroffen werden. Aufgrund einer Reihe von technischen Vorteilen wurde letztere gewählt, d. h. die Ergebnisse dieses Projekts sind voll und ganz konform mit der DATEX II, Version 3.0. Die neue Version kommt mit einer neuen erstellten Website[2] Und ein Dokumentationsportal[3].

3 Ansatz der Transformation

Die Domain-Ontologien von OKSTRA und GML liegen relativ nahe beieinander, so dass diese bereits bei der Einführung von OKSTRA-XML (in der OKSTRA-Version 1.007) direkt als GML-Applikationsschema als Besonderheit des Geography Markup Sprache angegeben wurde GML (OGC 2004). Bei DATEX II und GML ist dies nicht der Fall. Das heißt, einerseits sind die Domain-Ontologien unterschiedlicher und andererseits gibt es aktuell kein GML-Applikationsschema für DATEX II.

2 http://datex2.eu

3 http://d2docs.ndwcloud.nu/

Im Zuge einer Harmonisierung dieser unterschiedlichen Datenmodelle ist die Generierung eines DATEX II-GML-Applikationsschema ein wichtiger erster Schritt, der sowohl Entwickler als auch Anwender im Bereich der Verkehrsdaten im Sinne einer Reduzierung unterschiedlicher, technischer Strukturen.

Die in diesem Projekt geleistete Arbeit basiert auf DATEX II, Version 3.0, standardisiert in CEN/EN 16157 Teil 1 und den folgenden Teilen. Der Vorteil dieser neuen DATEX II-Version gegenüber der Version 2.3 wird detailliert im Schussbericht erläutert.

Ausgehend von dem unabhängigen DATEX II-Plattformmodell (PIM) und dem entsprechenden DATEX II-UML-Profil – beide in Enterprise Architect verfügbar – bestand die erste Aufgabe darin, Regeln für die Umwandlung in ein (nahezu) gleichwertiges Modell (PIM) zu erstellen das ISO 19103 und ISO 19136 entspricht (siehe Bild 1).

Neben der bereits erwähnten ISO-Norm spielt die ISO-Norm 19109 „Regeln für das Applikationsschema“ eine wesentliche Rolle, die die wesentlichen Komponenten einer UML-basierten Geodaten-Modellierung angibt. Ebenso gilt die ISO 19118 die definiert wie ein UML-Diagramm kodiert ist, das ein physisches GML-Schema angibt.

Aus dem resultierenden Modell wird ein XML-Schema erstellt, genau wie in DATEX II (und auch in OKSTRA). Für das DATEX II-Modell selbst wird zu diesem Zweck ein spezielles Tool verwendet (neue Webversion verfügbar unter http://datexwebtool.azurewebsites.net/). Um ein GML-Applikationssche-

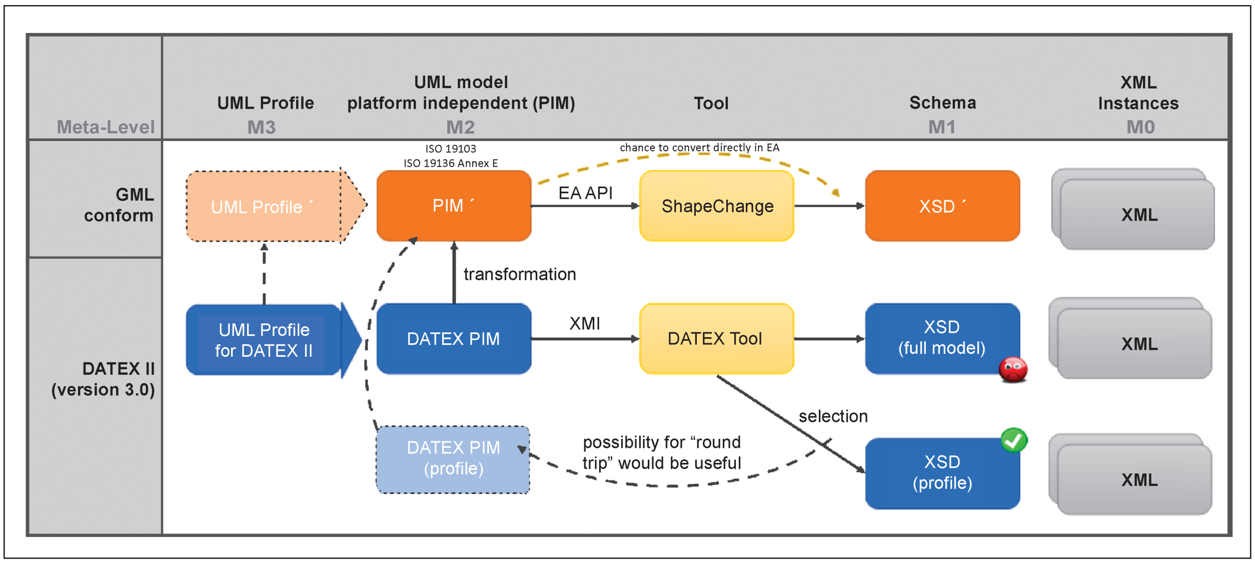


Bild 1: Unterschiedliche (Meta-)Ebenen des DATEX II- und des GML-konformen Modells und ihrer Beziehungen

ma aus dem PIM-Modell zu erstellen, steht das Open Source-Tool ShapeChange (https://shapechange.net/) zur Verfügung, dass über eine API eine Verbindung zu Enterprise Architect (EA) herstellt. Eine zweite Möglichkeit ist die direkte Verwendung von EA, da ab Version 13 eine direkte Konvertierung ohne weiteres Werkzeug möglich ist. Als Ergebnis sind beide Varianten möglich, aber die letztere unterstützt keine Anpassung für die Konvertierung. Das Projekt hat beide Methoden erprobt und die Ergebnisse verglichen.

Ein interessantes Feature ist die Generierung von DATEX II-Profilen, die ebenfalls mit dem oben genannten Tool durchgeführt wird. Um von dieser Möglichkeit zu profitieren, wäre ein „Roundtrip" vom DATEX II-Profil in ein DATEX II-UML-Modell erforderlich.

4 Generierung eines DATEX II-GML-Applikationsschemas

Im Rahmen des Proof-of-Concepts haben sich zwei Möglichkeiten der GML XSD-Schema-Generation herausgebildet. In diesem Kapitel werden die Anwendungen und die Ergebnisse vorgestellt. Zunächst wird ShapeChange als bevorzugtem Werkzeug vorgestellt. Während der Prototyp-Implementierung ist eine weitere Möglichkeit innerhalb des Enterprise Architect elaboriert worden. Eine Möglichkeit ist es die Projektergebnisse mit den entwickelten Tool Als Open Source zum Download unter https://www.datex2.eu/support/tooling bereitzustellen.

4.1 ShapeChange

Um ein GML-Applikationssschema aus dem PIM′ zu erstellen, wird das Open-Source-Tool Shape Change (https://shapechange.net/) Version 2.6.0 (11. September 2018) genutzt.

Die Hauptfunktionalität von ShapeChange ist die Generierung von Applikationsschemata, die als eine Reihe von XML-Schema-Dokumenten (XSDs) kodiert sind. Die Konvertierungsregeln von UML zu XML-Schema basieren auf den Kodierungsregeln, die in Anhang E des GML 3.2-Standards, in der GML 3.3-Norm und ISO/TS 19139:2007 – sowie einer Reihe von Erweiterungen der standardisierten Regeln festgelegt sind. Weitere Details zur Konfiguration, den Systembedingungen, der konkreten operationalen Schritte und der letztlichen Ausführung von ShapeChange sind Teil des vollständigen Berichts. Nachdem alle Regeln angewendet wurden und die Transformation durchgeführt wurde, ist es in ShapeChange nicht möglich gewesen eine gültiges IS0 19103 Schema zu erzeugen. Die Ursache für die aufgetretenen Fehler ist derzeit unbekannt.

4.2 Enterprise Architect

Um ein GML-Applikationssschema aus dem PIM′ zu erstellen, bietet der Enterprise Architect eine integrierte Funktionalität. Weitere Details zur Konfiguration, den Systembedingungen, der konkreten operationalen Schritte und der letztlichen Ausführung von Enterprise Architect sind Teil des vollständigen Berichts. Als Ergebnis des Transformationsprozesses in Enterprise Architect, wurde ein gültiges DATEX II-GML-Applikationssschema erzeugt. Die XML-Schema bildet die Relationen zwischen den Attributen und Elementen eines XML-DATEX II-Objekt ab.

5 Validierung des GML-Applikationschemas

Die Erstellung eines GML-Applikationsschemas erfolgt in zwei Schritten. Schritt eins zielt darauf ab, ein ISO 19103-konformes Modell auf der Basis des DATEX II-UML-Modells zu erstellen. Schritt zwei zielt darauf ab, das GML-Applikationsschema DATEX II zu erstellen, wie es in Kapitel 4 beschrieben ist. Nachdem die Transformationsregeln vollständig umgesetzt wurden, ist die Generierung des GML-Applikationsschemas erfolgreich mit Enterprise Architect durchgeführt.

5.1 Validierung des Datenmodells gegen ISO 19103

Beide Werkzeuge ermöglichen es ein GML-Applikationsschema zu erzeugen (XSD), und automatisch eine Validierung Gegen ISO 19103 durchzuführen. Da auch OKSTRA ISO 19103 konform ist, besteht die Möglichkeit eines Austauschs dynamischer Informationen über Straßenobjekte zwischen dem DATEX II-Datenmodell und dem OKSTRA-Modell.

5.2 Validierung des Datenmodells gegen UML

Das DATEX II-UML-Modell im EAP-Format wurde durch die Verwendung der Enterprise Architect Model Validation validiert, um das Modell auf bekannte UML-Regeln sowie alle innerhalb des Modells definierten Einschränkungen mit der Object Constraint Language (OCL) zu überprüfen. Folgende Eigenschaften des UML-Modells wurden überprüft:

- Elements, die Elemente und ihre Kinder, deren Eigenschaften (Attribute und Operationen) und Beziehungen (Vererbung),
- Diagramme, Diagramme in sich (deren Korrektheit) sowie Elemente und Anschlüsse im Diagramm,
- Pakete, Das Pakets Und alle sub-Pakete, Elemente, Anschlüsse und Diagramme.

Die Validierung wurde in Enterprise Architekt durchgeführt und die Ergebnisse im Ausgabefenster des Programms angezeigt.

5.3 Validierung des GML-Applikationsschemas

Um die Integrität und Konsistenz zu gewährleisten, wurde das erstellte DATEX II-GML-Schema validiert. Während des Validierungsprozesses wird hierfür Daher wird das zu Grunde liegende GML-Schema heruntergeladen. Für dieses Projekt wurde das Produkt oXygen verwendet. Das Ergebnis ist der Beweis, dass das erstellte Schema gültig, also frei von syntaktischen Fehlern ist. Das war in der Projektphase keineswegs immer der Fall. In der frühen Implementierungsphase gab es Probleme mit der Mapping-Software, die zu Fehlern führten. So führten zum Beispiel Zuordnungsprobleme (das Finden der richtigen Typen in den Modellstrukturen) oder Probleme mit Generalisierungen zu nicht validen XML-Strukturen.

5.4 Deutsches Baustellenprofil als Beispiel

Das deutsche DATEX II-Baustellenprofil wurde in diesem Projekt als Testbeispiel verwendet. Offiziell ist dieses Profil nur in der Version 2.3 von DATEX II verfügbar, nicht in Version 3.0. So wurde das Profil in Version 3.0 für die Nutzung dieses Projekts umgestaltet. Erweiterungen, die im 2.3-Modell (zum Beispiel zu Klassen SituationRecord, Impact und Roadworks) existierten, sind entweder ohnehin schon im 3.0-Modell enthalten oder wurden als Erweiterungen in 3.0 umgebaut.

Ein DATEX II-Profil wird in der Regel mit dem DATEX II-Tool generiert, indem Elemente (Klassen, Attribute, Literale) ausgewählt und abgewählt und verändert werden. Die Regel sollten nicht auf der Ebene des Enterprise Architect entwickelt werden (das Modell Enterprise Architect sollte immer Level A sein, optional mit Level-B-Erweiterungen, aber nicht als Profil modelliert). Jedoch wird in diesem Projekt die Umwandlung in ein GML auf Enterprise Architect-Ebene durchgeführt. Aus diesem Grund musste das deutsche Baustellenprofil direkt innerhalb von Enterprise Architect zugeschnitten werden. Bitte beachten Sie, dass es aus Sicht der Qualitätssicherung noch nicht geeignet ist, als Baustellenprofil der Version 3.0 zu verwenden – es ist ein Beispiel für die Nutzung und keine offizielle Version. Sobald das Profil der Roadworks dem EAP-Modell hinzugefügt wurde, wurde ein XML aus der transformierten GML-konformen XSD-Datei erzeugt.

6 Offene Themen

Einige Punkte wurden im Projekt identifiziert, die es zukünftig noch zu adressieren gilt:

- Klassendiagramme (Abbildungen) in Enterprise Architect

  Die aktuelle Mapping-Software konzentriert sich auf die Transformation des Modells selbst. Klassendiagramme, die in der Regel Teil eines UML-Modells (DATEX II) sind, werden nicht in das GML-Zielmodell übertragen. Aus diesem Grund kommt das daraus resultierende Enterprise Architect-Modell ohne Klassendiagramme aus. Eine Verbesserung der Mapping-Software für die Zukunft könnte sein, auch die Klassendiagramme in das resultierende GML-Modell zu übertragen.

- Reihenfolge der Elemente

  Die Reihenfolge der Elemente (z. B. Attribute, Aggregierungen) ist im DATEX II-UML-Modell durch den Wert „order“ und in der GML UML durch die SequenceNummerneigenschaft markiert. Das Eintragen dieser Sequenznummer wird noch nicht von der Mapping-Software durchgeführt, da es technische Schwierigkeiten gab, diese Eigenschaft richtig zuzuordnen. In der Folge könnte die Reihenfolge der Elemente bei der Erstellung von Schemata voneinander abweichen.

- Enterprise Architect Bug

  Die Enterprise Architect-Funktion, um ein GML-Modell in ein XML-Schema umzuwandeln, erzeugt falsche Ergebnisse, wenn die Unterklasse einer Verallgemeinerung von stereotypen „DataType“ ist. Das ist wahrscheinlich ein Fehler von Enterprise Architect. Aus diesem Grund wurde ein spezielles Vorgehen definiert, welches das Modell semantisch leicht verändert, um Objekte, die vom Stereotypen „DataType“ sein sollten, in einen „FeatureType“ abbilden zu können.

7 Zusammenfassung

Die Ergebnisse des Transformations-Ansatzes mit der Implementierung des Prototyps haben gezeigt, dass technisch die Möglichkeit besteht, DATEX II-Feeds in ein gültiges GML-Applikationsschema umzuwandeln. So können mit diesem Bericht und seinen technischen Ergänzungen alle Interessierten aus einem DATEX II-UML-Modell ein GML-Applikationsschema erstellen. Für das aktuelle DATEX II-„Level A“-Modell in der Version 3.0 ist dies geschehen und durch den generischen Ansatz, können auch spezifische DATEX II-Modelle (z. B. mit Level B-Erweiterungen) oder weitere Versionen abgebildet werden. Damit ist es jetzt möglich standardisierte und OGC-konforme Geodienste wie GDI-DE und IN-SPIRE zu nutzen, ohne dass ein DATEX II-Modell manuell neugestaltet werden muss.

In diesem Bericht wurde ebenfalls das Mapping für die UML-Modellebene von DATEX II festgelegt. Die DATEX II-Profile werden jedoch auf Schema-Ebene angegeben, nicht auf UML-Ebene (es sei denn, Sie machen manuelle Änderungen, wie es für diesen Bericht mit der Veränderung des Baustellenprofils exemplarisch geschehen ist). Hier wäre ein automatisiertes Feature zur Generierung eines konsistenten UML-Modells auf Basis eines DATEX II-Profils hilfreich. Im Kapitel 3, wurde dieses Problem mit dem fehlenden „Roundtrip“-Feature adressiert. Mit einem solchen Feature wäre es noch einfacher, aus einem DATEX II-Profil ein GML-Applikationsschema zu erstellen.

Für die in diesem Bericht enthaltenen Ergebnisse wird empfohlen, sie in den verschiedenen Interessensgemeinschaften zur Verfügung zu stellen. Für DATEX II könnte dies bedeuten, die Ergebnisse auf der DATEX II-Plattform hochzuladen, so dass Nutzer und andere Interessierte selbst ein Mapping der Datenmodelle erstellen können und erste Erfahrungen damit sammeln.

Schließlich wäre es auch wünschenswert, die Ergebnisse in die europäische Normung (im Rahmen von CEN/TC 278, Arbeitsgruppe 8 „Daten im Straßenverkehr“ oder andere machbare Gruppen) einzubringen.

Content

1 Introduction

In the areas of static road data and Intelligent Transport Systems (ITS) (Dynamic Traffic Data), historically two parallel approaches to the modelling and exchange of data and information have developed:

In the field of road operation, the national standard OKSTRA is the basis for data exchange. The international standard for the exchange of geodata is the GML (Geography Markup Language) and serves as the basis for internationally standardized and OGC-compliant geoservices such as GDI-DE and INSPIRE. To use these geoservices, the OKSTRA-GML application schema is used.

On ITS side, the European standard DATEX II (CEN /TS resp. CEN/EN 16157) is used to exchange traffic information and traffic data, e.g. via the German Mobility Data Marketplace MDM. Until now, there is no GML application schema for DATEX II.

Increasingly, both ITS and roadside IT systems and services require integrated data from both domains. The specification for Priority Action B (Commission Delegated Regulation (EU) 2015/962 on the provision of EU-wide real-time traffic information services) in the framework of the European ITS Directive now provides for the first time a common consideration of road data and dynamic traffic data. These data should be provided jointly via a national access point (MDM in Germany). For this, the use of the international standards DATEX II and INSPIRE is planned.

Since there is no GML application schema for DATEX II, direct sharing in ITS and road planning and traffic systems and integration of DATEX II-based data into national and international geo-services is limited and time-consuming.

This report describes the development of a procedure with which a GML application schema can be generated on the basis of the DATEX II UML model in largely automated process steps. A necessary prerequisite for this is a set of rules for a model transformation with which the existing DATEX II UML model can be transformed into an ISO-19103-compliant UML model. The purpose of this is to establish a basis for incorporating these aspects into the DATEX II specification process, thus contributing to improving the interoperability of traffic data, road data and general geospatial data. The results are meant to support the further DATEX II specification process[1].

2 Analysis of the DATEX II metamodel

2.1 Overview on DATEX II

DATEX II is the European Standard for exchanging traffic information and traffic data. The maintenance of the standard and its large data model is currently financed by a CEF Program Support Action (PSA) of the European Union. In several Delegated Regulations – as an outcome of the ITS Directive 2010/40/EU – the European Union claims DATEX II for several kinds of ITS data communication and so fosters this standard.

DATEX II is currently standardised in the CEN/EN 16157 series (Parts 1, 2, 3 and 7) as well as in further parts still being a Technical Specification (Parts 4, 5, 6 of CEN/TS 16157). A migration of these parts into a European Norm is currently in process.

2.2 DATEX II, Version 3.0

In alignment with the publication of the first parts of the above mentioned EN-series, a new major version of DATEX II was released in early summer 2018. It is called version 3.0 of DATEX II. This version comprises a number of visible but also 'hidden' changes and is no longer backwards compatible to the versions before (i.e. 2.3 and older).

Early within this project, the news of the upcoming release of DATEX II, Version 3.0 emerged. In this situation, a decision between the current DATEX II version at this point of time and the upcoming version had to be taken. Because of a number of technical advantages, the latter one was chosen, i.e. the results of this project are fully compliant with (and depend on) DATEX II, Version 3.0.

The major changes of this new version are:

- Usage of UML 2.41 instead of UML 1.4 as before (i.a. much stricter modelling, especially in matters of tagged values and stereotypes).
- Use of a complete refurbished DATEX II UML profile (new elements, new stereotypes) – see further chapter.

[1] https://www.datex2.eu/support/tooling

- Stricter distinction between payload and exchange (exchange is no longer part of the data model).
- Introduction of multiple namespaces and modular organisation of the data model (users can choose those parts of the data model they actually need for their use case).
- Revision of large parts of the data model (according to issues collected during the version 2.x phase, including those that are non-backwards compatible model changes).
- Improvement of the DATEX II Tool (Windows version) to create schema files from the Enterprise Architect model (adaption to new version 3.0 technologies, now using XMI 2.1 as interchange format).
- New web based DATEX II Tool[2] (similar functionality to the above-mentioned Windows version, but independent from operating system and with a 'Wizard'-guidance).
- New extension rules for extending Enumerations with additional literals (as Level B).

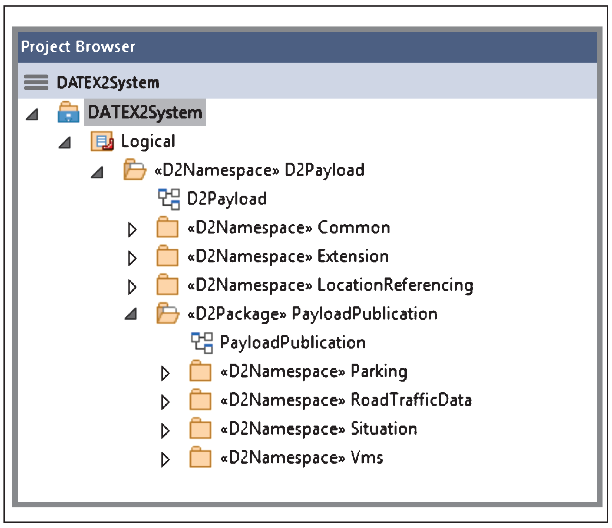
The new version comes along with a new modelled website[3] and a documentation portal[4].

Figure 1 depicts the new structure of the data model of DATEX II, Version 3.0 with its different namespaces. Note that due to diverging CEN-standardisation time schedules, the namespaces "Vms", "RoadTrafficData" and "Parking" still have to be migrated into the European Norm (Parts 4, 5 and 6) and are therefore still provided in their 2.3-datamodel version on an interim basis.

[2] http://datexwebtool.azurewebsites.net/

[3] http://datex2.eu

[4] http://d2docs.ndwcloud.nu/



Project Browser

```
DATEX2System
◢ DATEX2System
    ◢ Logical
        ◢ «D2Namespace» D2Payload
              D2Payload
            ▷ «D2Namespace» Common
            ▷ «D2Namespace» Extension
            ▷ «D2Namespace» LocationReferencing
            ◢ «D2Package» PayloadPublication
                  PayloadPublication
                ▷ «D2Namespace» Parking
                ▷ «D2Namespace» RoadTrafficData
                ▷ «D2Namespace» Situation
                ▷ «D2Namespace» Vms
```

Fig. 1: Structure of DATEX II, Version 3.0

2.3 DATEX II UML profile

The DATEX II data modelling methodology specified in CEN/EN 16157-1 uses the Unified Modelling Language (UML), version 2 as specified in ISO/IEC 19505-1:2012. More accurately the release 2.4.1 of UML 2 is used. UML provides a vast set of modelling elements that are not all used for DATEX II data modelling. In fact, DATEX II uses a fairly small UML profile, based on the following metaclasses from the Core:Basic and Core:Constructs packages specified in ISO/IEC 19505-1:2012:

- Association (stereotypes: D2Relation),
- Attribute (stereotypes: D2Attribute, D2Literal),
- Class (stereotypes: D2Class, D2Identifiable, D2VersionedIdentifiable, D2ModelRoot, ExternalClass),
- DataType (stereotypes: D2Datatype, ExternalType),
- Enumeration (stereotype: D2Enumeration),
- Generalization (some of them having the stereotype: D2LevelBExtension),
- Package (stereotypes: D2Package, D2Namespace, ExternalNamespace).

The resulting UML profile for DATEX II is depicted in figure 2.

DATEX II specifies metadata for the metaclasses in a UML profile with stereotypes which are assigned to the metaclasses, as listed above.

- Simple types that may be assigned to attributes are defined as datatypes or enumerations.
- Complex types (data structures) are defined as classes, with potential substructures connected via a composition.
- Generalization between classes is allowed, but multiple inheritance is prohibited, i.e. each class shall have either zero or one superclass.

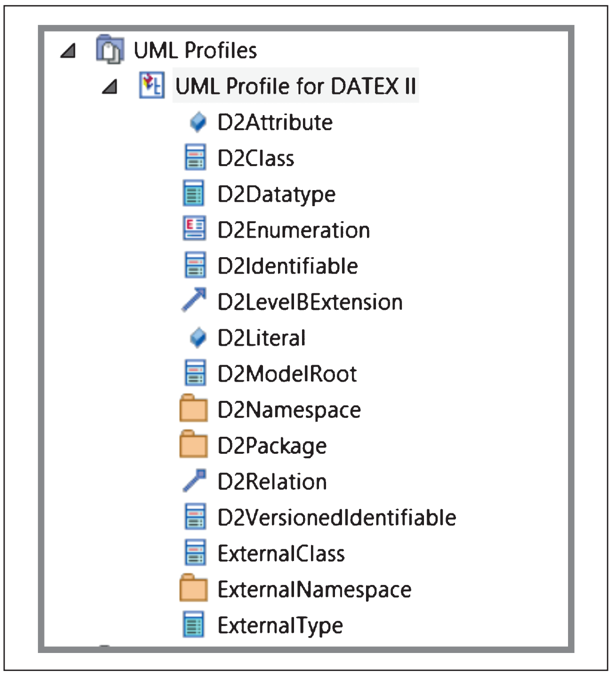

UML Profiles
UML Profile for DATEX II
D2Attribute
D2Class
D2Datatype
D2Enumeration
D2Identifiable
D2LevelBExtension
D2Literal
D2ModelRoot
D2Namespace
D2Package
D2Relation
D2VersionedIdentifiable
ExternalClass
ExternalNamespace
ExternalType

Fig. 2: UML profile for DATEX II, Version 3.0 (Enterprise Architect tab “Resources”)

- Classes, datatypes and enumerations may be structured in namespace packages, which define a namespace for their contained classes, datatypes, enumerations and sub-packages. They may also be structured simply in packages, which have no further semantics in DATEX II, they simply serve to structure DATEX II models and make them more accessible. Namespaces, classes and datatypes also have an option to refer to an external namespace, class or datatype. The stereotypes defined for this purpose contain only the metadata required to refer to the external entity.

Further to that, additional properties have been defined in DATEX II to govern the platform specific mapping to XML schema definitions. Other property sets may be added to generate an alternative platform specific model. Models that claim to comply with this specification may use these UML elements but shall comply with all provisions regarding the use of these elements.

The DATEX II stereotype-properties (also called tagged values) for the Platform Independent Model are:

- definition: a clear, comprehensible and unambiguous definition of the modelled concept.
- description: a general textual reference and explanation of an external specification that is included in a DATEX II model by using either classes with stereotype “ExternalNamespace”, “ExternalType” and/or “ExternalClass”.
- extensionName: the name of an extension if the model is extended.
- extensionVersion: the version of an extension if the model is extended.
- modelBaseVersion: the number of the major component of a version number, e.g. “n” in case of version number “n.m”.
- namespaceUri: a Universal Resource Identifier that uniquely identifes an external namespace.
- order: allows determining the order in which multiple metaclasses shall occur in a serialised structure of a container.
- profileName: the name of a profile if the model is a profile.
- profileVersion: the version of a profile if the model is a profile.
- regulatoryContext: a description of any relevant regulatory context that governs the creation or the structure of a metaclass, if applicable.
- version: the full version number of the associated metaclass.

Properties for the Platform Specific Model are:

- facets: the value of this property shall be a valid content for the restriction element of an XML schema simple type definition of a type restriction of the XML schema type that a DATEX II metaclass is mapped to.
- prefix: a prefix used to represent a namespace in an XML Schema mapping.
- rootElement: the indication to generate a top-level element in the schema with the class as type, using the property value as element name.
- schemaAttribute: if this property is set to “yes”, an attribute with simple type (datatype or enumeration) from the platform independent model will be mapped to an XML attribute, if the property is empty or has any other value, it will be mapped to an XML element.
- schemaLocation: A URL that points to an external XML schema that will be imported into the DATEX II schema.

- schemaName: if this property is non-empty, its value will be used in the mapping of the meta-class instead of the metaclass' own "name" attribute or any name following by specific naming conventions.
- schemaType: this property is used to name a XML Schema simple type that a DATEX II datatype is mapped to.
- schemaTypeInclude: a reference to an externally specified XML schema type to be used as a mapping for this metaclass.
- targetClass: for attributes referring to other classes (i.e. classes with stereotype "D2Identifiable" or "D2VersionedIdentifiable"), this tagged value depicts the name of the other class and is used to create a typed reference.

All of these tagged values used in the DATEX II UML profile can be identified in figure 3 and figure 4, which show the internal construction of the new stereotypes.

Note that especially the content of the 'definition' tagged value is handed over into the schema file(s) and gives the implementors help at hand to identify the semantic of the different elements.

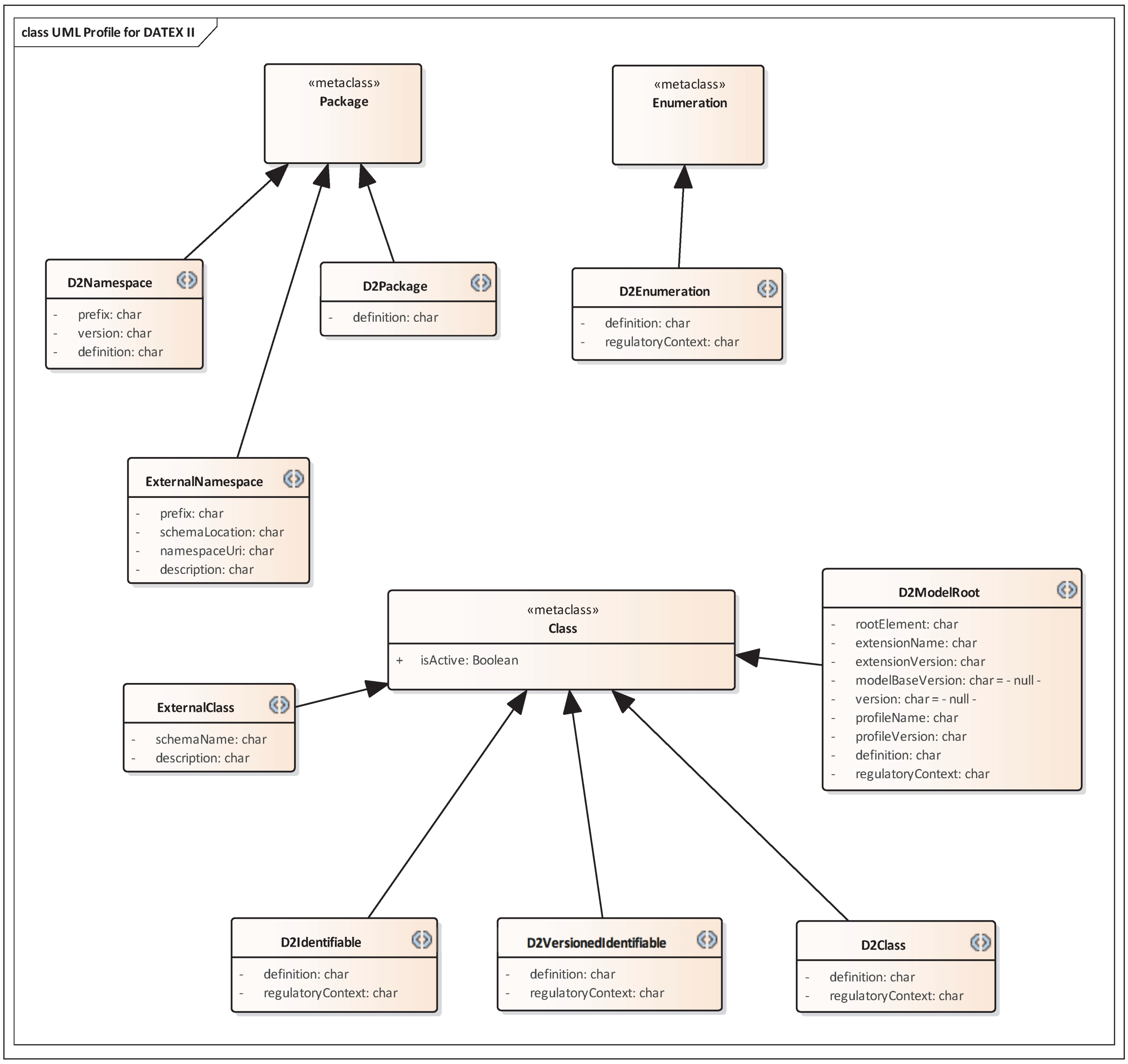


Fig. 3: Internal definition of UML profile for DATEX II, Version 3.0 (part 1)

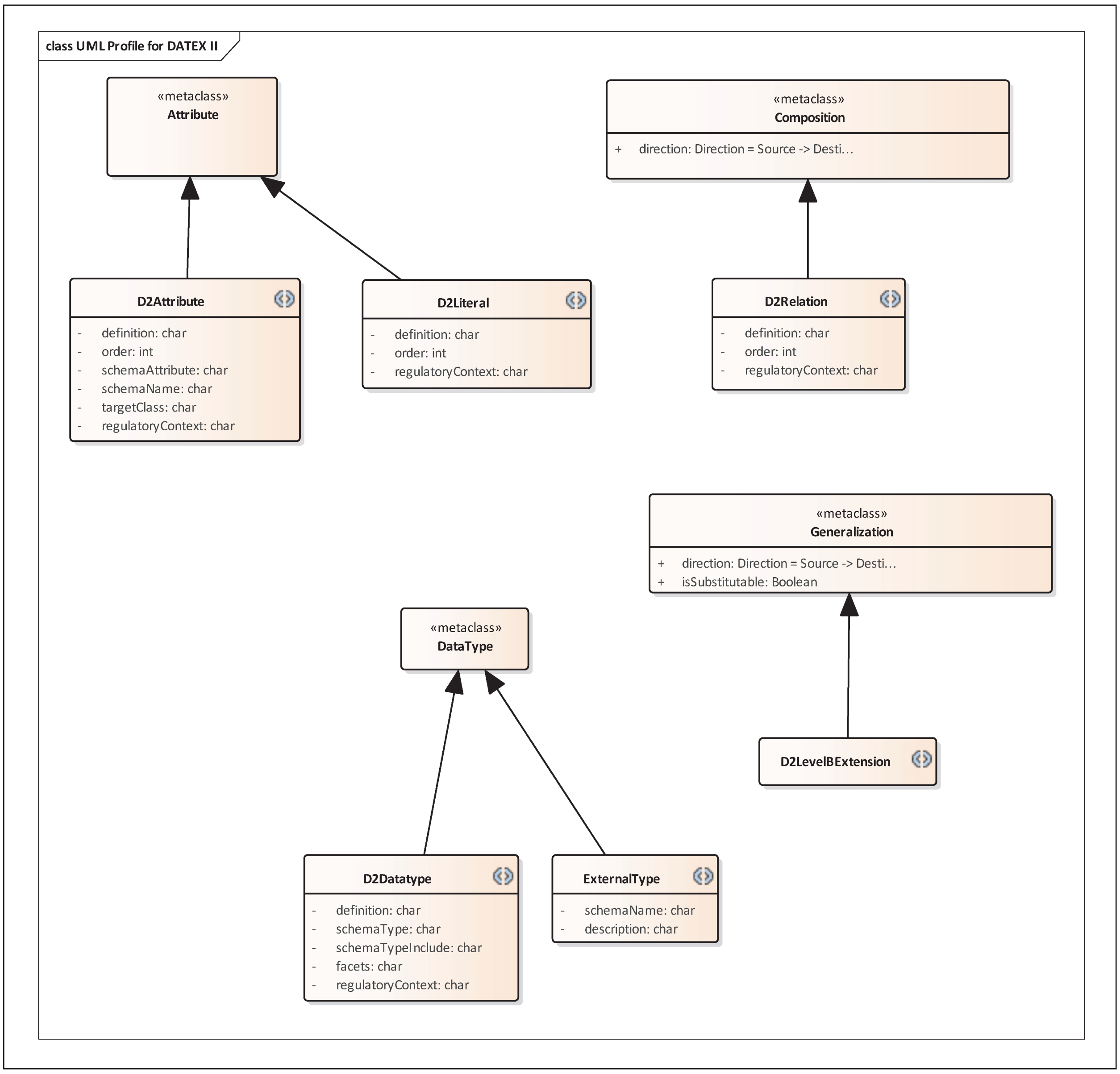


Fig. 4: Internal definition of UML profile for DATEX II, Version 3.0 (part 2)

3 Approach

The domain ontologies of OKSTRA and GML in general are very similar. Thus, with the introduction of OKSTRA-XML (in OKSTRA version 1.007) it was directly translated into a GML application schema as a special feature of the Geography Markup Language GML (OGC 2004). For DATEX II and GML this is not the case; i.e. on one hand the domain ontologies are more different and on the other hand there is no GML application schema for DATEX II.

In the course of a harmonization of these different data models the generation of a DATEX II GML application schema is an important first step, which should support both developers and users in the field of traffic engineering in the sense of a reduction of different, technical structures.

The work done in this project is based on DATEX II, Version 3.0, standardised in CEN/EN 16157 Part 1 and following parts. The advantage of this new DATEX II version in comparison to version 2.3 is explained earlier in this document.

Starting from the DATEX II Platform Independent Model (PIM) and the corresponding DATEX II UML profile – both available in Enterprise Architect – the first task was to create rules for a transformation into an (nearly) equivalent model (called PIM′) that is in accordance with ISO 19103 and ISO 19136, Annex E – see figure 5. The resulting transformation rules can be found in Annex A of this document.

In addition to the already mentioned ISO standard, the ISO standard 19109 "Rules for application schema" plays an essential role, which provides the essential components of a UML-based geodata modeling, as well as ISO 19118, which encodes a UML diagram and specifies a physical GML scheme.

From the resulting model, an XML schema will be created, just as this is done in DATEX II (as well as in OKSTRA). For the DATEX II model itself, a specific Tool is used for this purpose (new web version available at http://datexwebtool.azurewebsites.net/). To create a GML application schema from the PIM′, the open source tool ShapeChange (https://shapechange.net/) is available, which connects via an API to Enterprise Architect (EA). A second option could be the direct use of EA, as from version 13 on, a straight conversion without any further tool is supported. Probably, both variants are feasible, but the latter one does not support any customisation for the conversion. Both methods and the transformation results are compared in chapter 6.

An interesting feature is the generation of DATEX II profiles, also done with the above-mentioned Tool. By not using the complete model but only those elements and multiplicities that are needed, the resulting schema gets stricter and slimmer. Having this possibility, it would be even more interesting to use such a profile as a base for the generation of the GML application schema (denoted by the two dotted black arrows in figure 5). To benefit from this possibility, a 'round trip' from the DATEX II profile into a DATEX II UML model would be necessary. It is known, that such a feature is under development by the DATEX II PSA[5], but at this stage, it only effects the visual elements in Enterprise Architect and not the underlaying model itself.

4 Development of the transformation rules

4.1 Overview

For the transformation of the DATEX II UML model into a model, that is GML application schema conform, a set of rules has been defined. These rules are explained and denoted in Annex A – DATEX II as GML application schema – transformation rules. Technical annotations and issues are described in this annex, but some topics need a more general attention and are discussed in the following subchapters.

Note: To simplify description, the term "GML model" is used here to describe the UML PIM′ which may serve as a base for the GML application schema.

[5] Program Support Action of the European Union, that develops and maintains DATEX II.

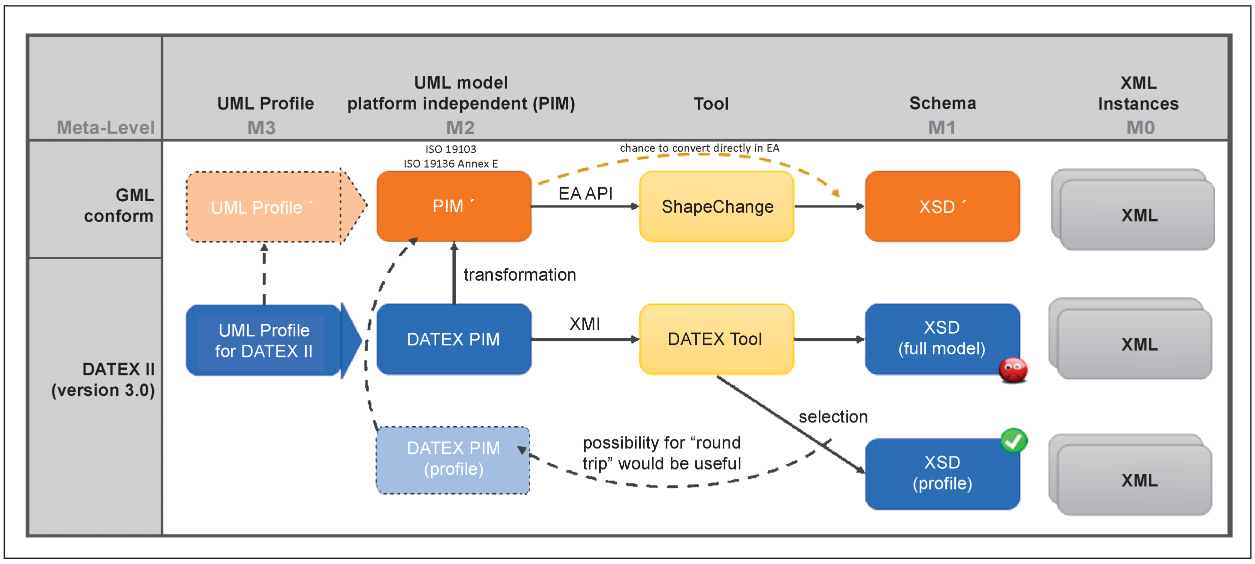


Fig. 5: Different (meta-)levels of the DATEX II and the GML conform model(s) and their relationships/transformation needs

4.2 DATEX II extensions

4.2.1 Level B and Level C extensions in DATEX II

DATEX II offers the ability to add a limited amount of application specific business logic to a model without losing backwards compatibility. Here, DATEX II differs between level B extensions and level C extensions. The levels' names indicate a compatibility hierarchy, where compatibility means system level interoperability. Level A means maximum compatibility, so both interacting systems use an identical model (the core model). Level B extensions require backward compatibility with an existing model, e.g. for systems exchanging XML messages a valid instance of an extended model shall always be also a valid instance for the core model. Level C extensions are extensions which cannot seek any type of system level interoperability.

From a technical point of view, Level B extensions in DATEX II are denoted with the stereotype "LevelBExtension". It is mapped to an ordinary generalisation. Nevertheless, the technical usage is a bit different from a generalisation, because the extension will always be present, whenever an instance of the extended class is created. The following figure 6 shows an extension of the Roadworks class.

In DATEX II, each class has got a generic extension element by default that points to a generic type filled with "any" (not shown in the code below). This ensures backwards compatibility in case someone uses an extended model, but the client system is not aware of that extension.

In case an extension is specified, the pointer to the generic type is replaced by a specific one. In the corresponding schema files to the extension in figure 6, a generic element "_roadworksExtension" points to the specific "_RoadworksExtensionType" (both yellow) which leads to the extended content (green) see (figure 7).

4.2.2 Mapping to GML

Since all extension types (Level B and Level C) must fulfil the same DATEX II conformance rules as the core model, they can be mapped to GML by the transformation rules presented in this document. figure 8 shows the problem, which occurs when mapping DATEX II level B Extensions to GML.

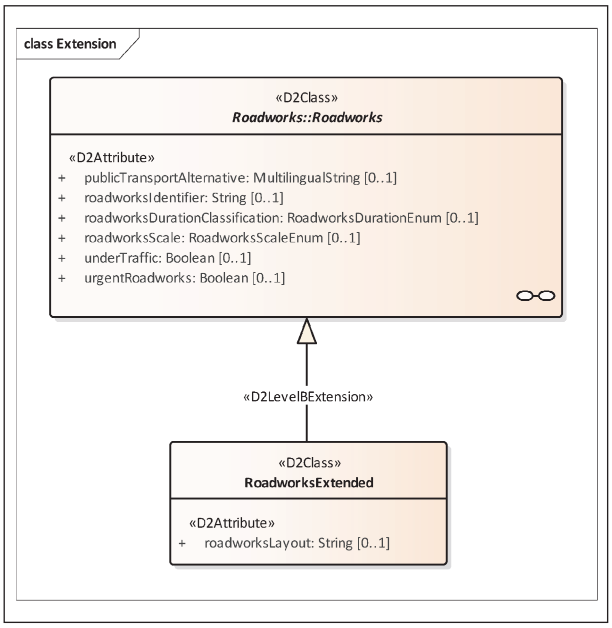
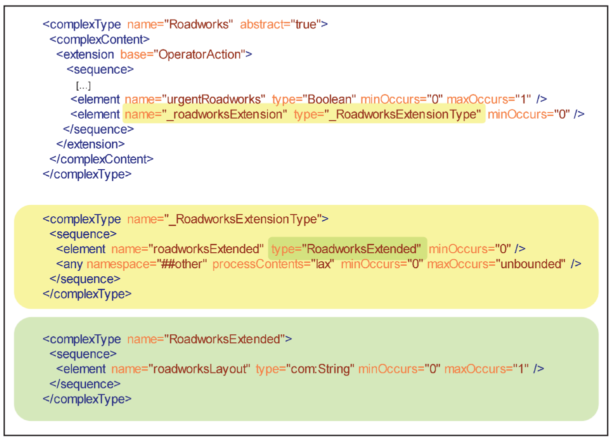

class Extension
«D2Class»
Roadworks::Roadworks
«D2Attribute»
+ publicTransportAlternative: MultilingualString [0..1]
+ roadworksIdentifier: String [0..1]
+ roadworksDurationClassification: RoadworksDurationEnum [0..1]
+ roadworksScale: RoadworksScaleEnum [0..1]
+ underTraffic: Boolean [0..1]
+ urgentRoadworks: Boolean [0..1]
«D2LevelBExtension»
«D2Class»
RoadworksExtended
«D2Attribute»
+ roadworksLayout: String [0..1]

Fig. 6: Extension of the Roadworks class



```
<complexType name="Roadworks" abstract="true">
  <complexContent>
    <extension base="OperatorAction">
      <sequence>
        [...]
        <element name="urgentRoadworks" type="Boolean" minOccurs="0" maxOccurs="1" />
        <element name="_roadworksExtension" type="_RoadworksExtensionType" minOccurs="0" />
      </sequence>
    </extension>
  </complexContent>
</complexType>

<complexType name="_RoadworksExtensionType">
  <sequence>
    <element name="roadworksExtended" type="RoadworksExtended" minOccurs="0" />
    <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>

<complexType name="RoadworksExtended">
  <sequence>
    <element name="roadworksLayout" type="com:String" minOccurs="0" maxOccurs="1" />
  </sequence>
</complexType>
```

Fig. 7: Code snippet – Extension of the Roadworks class

Since GML does not support backwards compatible Extensibility in the same way as DATEX II, the cross validation of instances conforming to level B extended models to the core model schema is lost.

Figure 9 and figure 10 show a DATEX II level A model and a corresponding level B extension. Figure 10 depicts how the D2Class Generic Publication of the level A model is extended by another attribute called additionalAttribute. Both DATEX II models can be mapped to GML by the transformation rules presented in this document. However, there is no way to map the DATEX II stereotype LevelBExtension of a generalisation to any corresponding extensibility feature in GML. Generalisations with this DATEX II stereotype assigned are therefore mapped to a Generalisation without any stereotype.

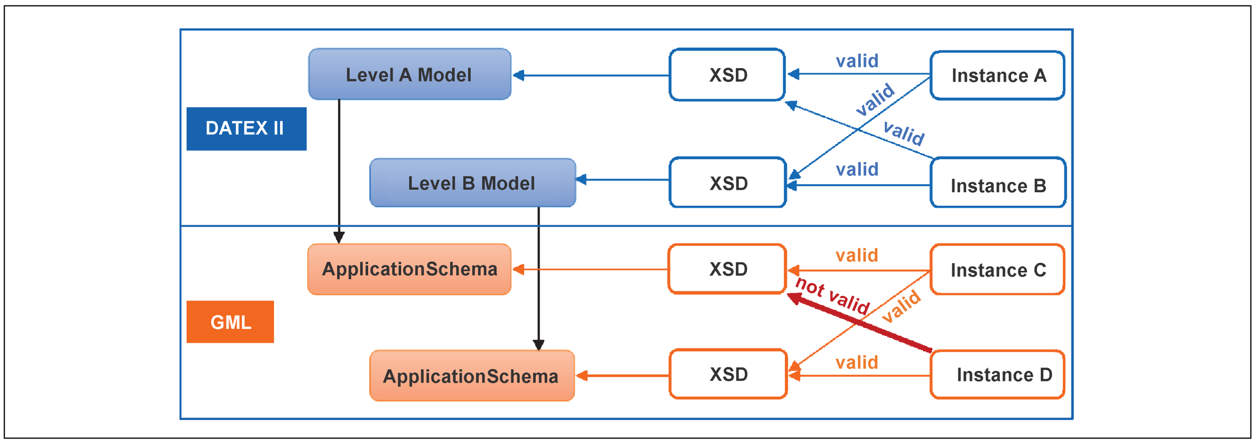


Fig. 8: Backwards compatibility problem of level B extension mapping

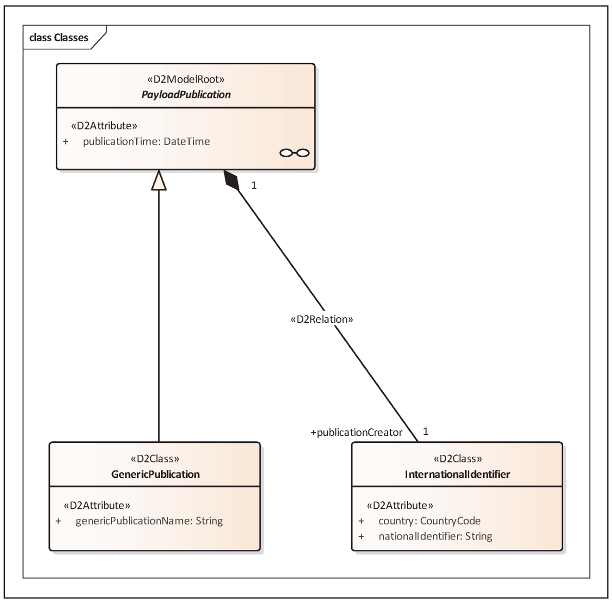


Fig. 9: DATEX II level A model

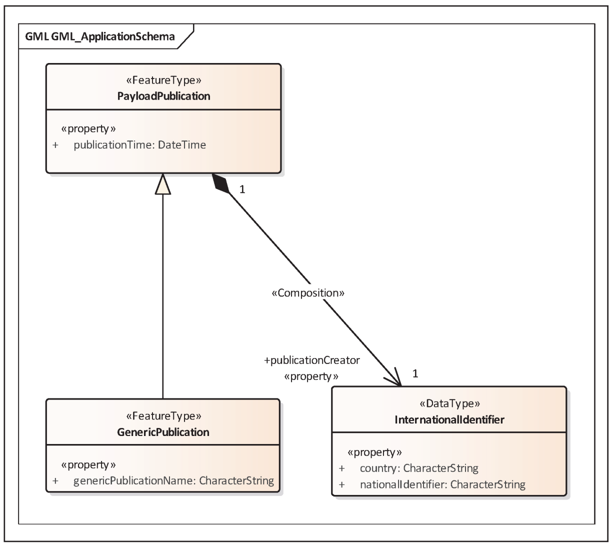


Fig. 11: GML application schema without extension

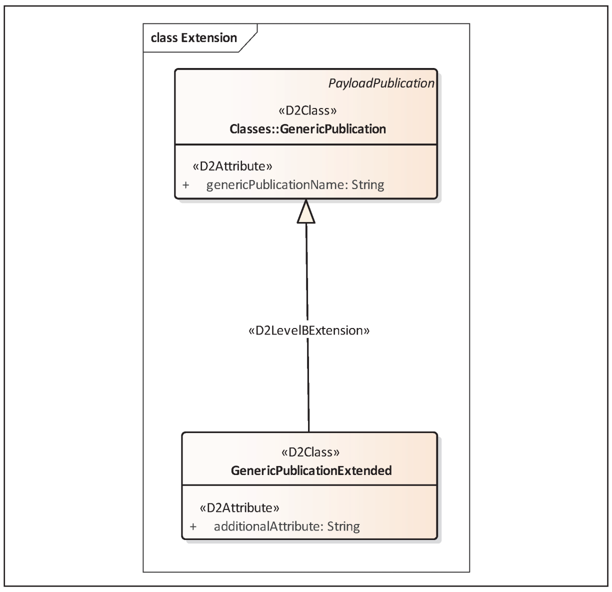

class Extension
PayloadPublication
«D2Class»
Classes::GenericPublication
«D2Attribute»
+ genericPublicationName: String
«D2LevelBExtension»
«D2Class»
GenericPublicationExtended
«D2Attribute»
+ additionalAttribute: String

Fig. 10: DATEX II level B extension

Figure 11 and figure 12 show the transformation results of these two models. It is obvious, that the specialisation of GenericPublication in the extended GML application schema (figure 12) has no mark-up that makes it distinguishable from any other specialisation in the model. The nature of the level B extension has been lost in the transformation, i.e. the new application schema is like a new “Level A” model in DATEX II.

As figure 8 shows, the next step is to create the XML schema documents from all four UML models. This is the step where the distinction between the DATEX II extension feature and the modification in the GML application schemas becomes clearly visible. The code extracts in figures 13, 14, 15 ,16 show, that the new class GenericPublication Extended in DATEX II is not created as an XSD extension of the superclass, but as a new class which is incorporated into the extension element of

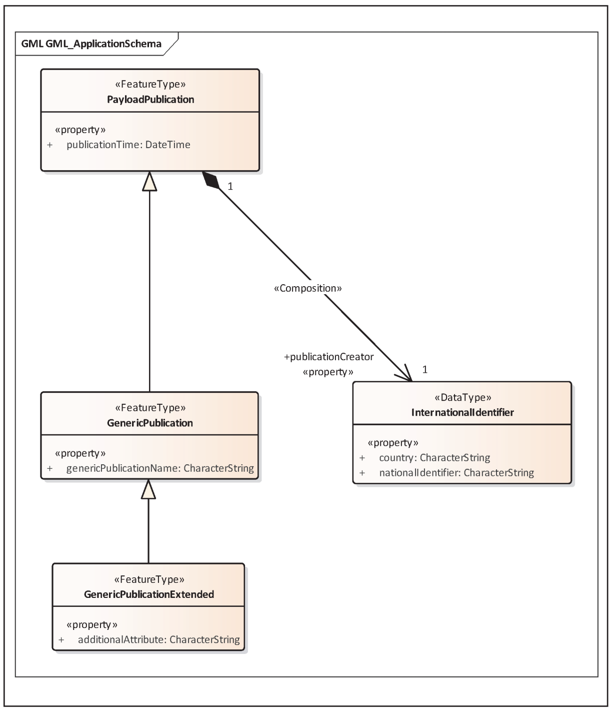

GML GML_ApplicationSchema
«FeatureType»
PayloadPublication
«property»
+ publicationTime: DateTime
1
«Composition»
+publicationCreator
«property»
1
«FeatureType»
GenericPublication
«property»
+ genericPublicationName: CharacterString
«DataType»
InternationalIdentifier
«property»
+ country: CharacterString
+ nationalIdentifier: CharacterString
«FeatureType»
GenericPublicationExtended
«property»
+ additionalAttribute: CharacterString

Fig. 12: GML application schema with extension

class GenericPublication (every complex type in DATEX II has such an extension element by default), see figure 14 to figure 16. Compared to this, the GenericPublicationExtendedType (figure 17) in the new GML application schema is – as expected – just an extension of the GenericPublicationType.

If one now takes a valid instance of the level B extended DATEX II schema and validate this against the original level A schema, the instance is still valid, although the content of the extension is of course not processed (see figure 18). If one takes a valid instance of the GML application schema produced by mapping the level B extended model and try to validate this against the GML application schema produced by mapping the original model, the validation fails since the new attribute additional Attribute appearing in this instance is not valid according to the content model described by the original schema. Figures 17 and 18 show the two different validation results.

```
<xs:complexType name="GenericPublication">
  <xs:complexContent>
    <xs:extension base="com:PayloadPublication">
      <xs:sequence>
        <xs:element name="genericPublicationName" type="com:String" minOccurs="1" maxOccurs="1" />
        <xs:element name="_genericPublicationExtension" type="com:_GenericPublicationExtensionType" minOccurs="0" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Fig. 13: Code snippet – XML schema document code extract – DATEX II class GenericPublication

```
<xs:complexType name="_GenericPublicationExtensionType">
  <xs:sequence>
    <xs:element name="genericPublicationExtended" type="ext:GenericPublicationExtended" minOccurs="0" />
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
```

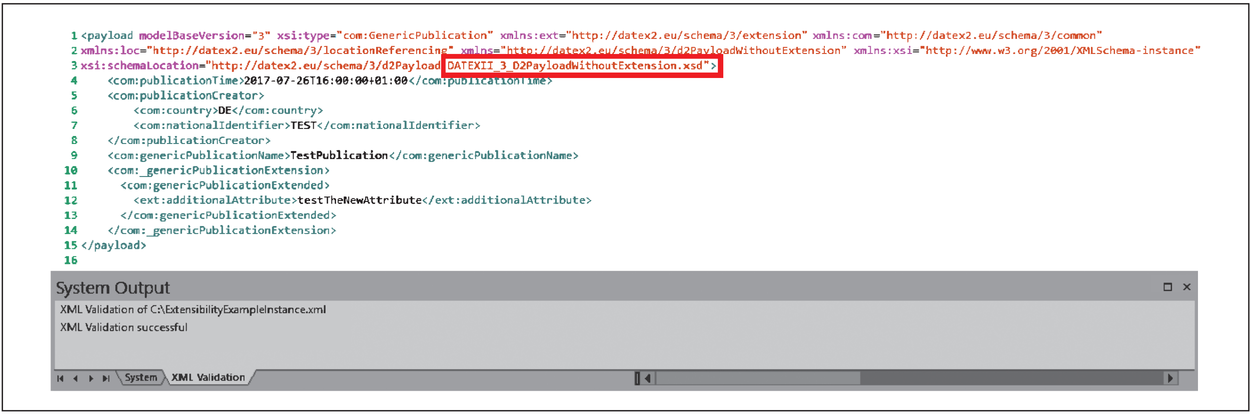
Fig. 14: Code snippet – XML schema document code extract – DATEX II GenericPublicationExtensionType

```
 <xs:complexType name="GenericPublicationExtended">
   <xs:sequence>
     <xs:element name="additionalAttribute" type="com:String" minOccurs="1" maxOccurs="1" />
   </xs:sequence>
 </xs:complexType>
</xs:schema>
```

Fig. 15: Code snippet – XML schema document code extract – DATEX II class GenericPublicationExtended

```
<xs:complexType name="GenericPublicationExtendedType">
    <xs:complexContent>
        <xs:extension base="exa:GenericPublicationType">
            <xs:sequence>
                <xs:element name="additionalAttribute" type="xs:string">
                    <xs:annotation>
                        <xs:documentation>For Test.</xs:documentation>
                    </xs:annotation>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Fig. 16: Code snippet – XML schema document code extract – GML GenericPublicationExtendedType

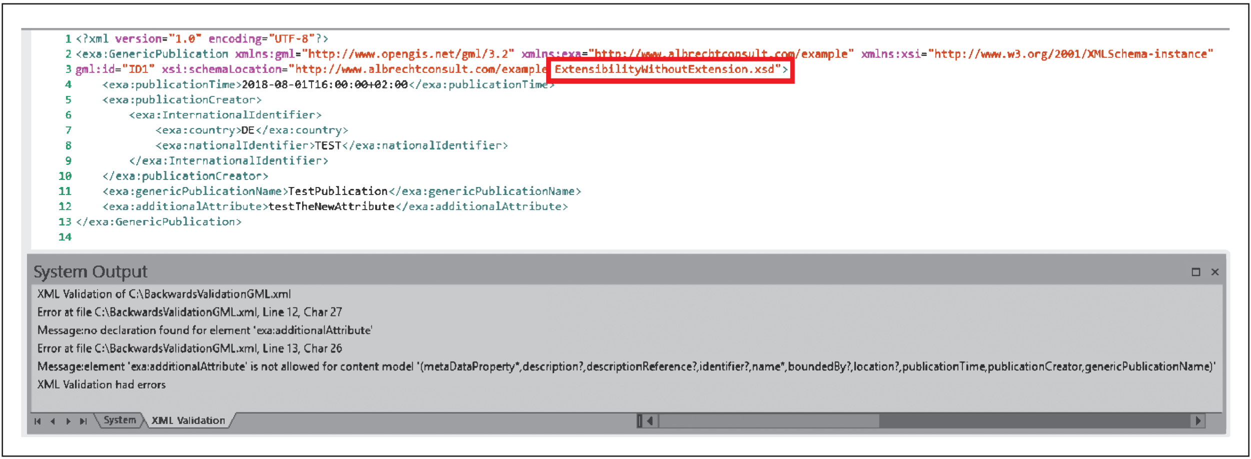


```
<payload modelBaseVersion="3" xsi:type="com:GenericPublication" xmlns:ext="http://datex2.eu/schema/3/extension" xmlns:com="http://datex2.eu/schema/3/common"
xmlns:loc="http://datex2.eu/schema/3/locationReferencing" xmlns="http://datex2.eu/schema/3/d2PayloadWithoutExtension" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://datex2.eu/schema/3/d2Payload DATEXII_3_D2PayloadWithoutExtension.xsd">
    <com:publicationTime>2017-07-26T16:00:00+01:00</com:publicationTime>
    <com:publicationCreator>
        <com:country>DE</com:country>
        <com:nationalIdentifier>TEST</com:nationalIdentifier>
    </com:publicationCreator>
    <com:genericPublicationName>TestPublication</com:genericPublicationName>
    <com:_genericPublicationExtension>
      <com:genericPublicationExtended>
        <ext:additionalAttribute>testTheNewAttribute</ext:additionalAttribute>
      </com:genericPublicationExtended>
    </com:_genericPublicationExtension>
</payload>
```

System Output

```
XML Validation of C:\ExtensibilityExampleInstance.xml
XML Validation successful
```

Fig. 17: Code snippet – validation of instance of level B extended DATEX II schema against original level A model



```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <exa:GenericPublication xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:exa="http://www.albrechtconsult.com/example" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 3 gml:id="ID1" xsi:schemaLocation="http://www.albrechtconsult.com/example ExtensibilityWithoutExtension.xsd">
 4     <exa:publicationTime>2018-08-01T16:00:00+02:00</exa:publicationTime>
 5     <exa:publicationCreator>
 6         <exa:InternationalIdentifier>
 7             <exa:country>DE</exa:country>
 8             <exa:nationalIdentifier>TEST</exa:nationalIdentifier>
 9         </exa:InternationalIdentifier>
10     </exa:publicationCreator>
11     <exa:genericPublicationName>TestPublication</exa:genericPublicationName>
12     <exa:additionalAttribute>testTheNewAttribute</exa:additionalAttribute>
13 </exa:GenericPublication>
14
```

System Output

```
XML Validation of C:\BackwardsValidationGML.xml
Error at file C:\BackwardsValidationGML.xml, Line 12, Char 27
Message:no declaration found for element 'exa:additionalAttribute'
Error at file C:\BackwardsValidationGML.xml, Line 13, Char 26
Message:element 'exa:additionalAttribute' is not allowed for content model '(metaDataProperty*,description?,descriptionReference?,identifier?,name*,boundedBy?,location?,publicationTime,publicationCreator,genericPublicationName)'
XML Validation had errors
```

Fig. 18: Code snippet – validation of instance of extended GML application schema against original model

4.3 MultilingualStrings

MultilingualStrings in DATEX II offer the ability, to supply a string together with a language information, and, if there is need for it, to supply the same string in more than one language. This is realised by an unlimited sequence of language and corresponding string for each multilingual string entry (independent of this construction, the attribute itself of type MultilingualString might have a multiplicity larger than one).

To map the datatype MultilingualString into GML, it could be possible to rebuild the above-mentioned structure "1:1". This would result in a separate (overwhelmed?) class structure for every attribute of this datatype. As a second option, it could be possible to specify each MultilingualString-attribute in GML with unbounded multiplicity and thus create an instance of the attribute for each language. But this option gets into trouble when already having the original DATEX II attribute with a multiplicity larger than 1.

As GML has got no concept for using multiple languages and having the above-mentioned disadvantages, it seems convenient to operate GML with only one language. Thus, a MultilingualString is mapped to a simple Character String with the same multiplicity as the original DATEX II attribute.

This solution holds for the transformation on UML model level. For the transformation on XML-instance level (which is actually not done within this project), different options exist on how to handle data in MultilingualString attributes:

- Option 1: Multiple pairs of languages and the corresponding strings are concatenated in a single string. Here a uniform form must be chosen such that the different languages can be differentiated from each other. One possibility would be to use a separator. However, since every character is allowed in strings, this separator can occur in the string to be transferred itself. Therefore, another rule must be defined that allows the separator to be used in the text.

Example: “#” can be used as a separator. If you want to transfer the instance of an attribute of the type MultilingualString, you have to concatenate the languages and the corresponding string as follows: “EN: this is an example # DE: Dies ist ein Beispiel”. If the case occurs that a “#” is used in the string, a second “#” must be used to indicate that the separator is not meant here. An example would be “EN: this is an example for the use of ## # DE: dies ist ein Beispiel für die Verwendung von ##”.

- Option 2: Since GML does not provide an option for multilingualism, GML Application Scheme are apparently intended to be monolingual. Therefore, you could also create a priority list to select one of the available languages. If, for example, an instance of an attribute is available in English and German and you set German as first priority and English as second, only the German variant is used for the transformation of the instances. A possibility to select the appropriate language in some user interface might be also an option.

Note that the above-mentioned options are on the level of an instance conversion, not on the level of UML conversion. For this reason, this report does not further sort out these options.

4.4 Summary of mapping issues

Setting up the transformation rules (see Annex A) brought up the following issues and interesting points. Not surprisingly, it is not possible to generate a bidirectional mapping and thus creating complete equivalent models. Nevertheless, the restrictions are of lower significance as they do not adulterate the results.

- It is important to understand that “DataType” is a stereotype used widely in the GML model (corresponding to DATEX II classes) but has nothing to do with the DATEX II datatypes or the D2DataType stereotype. It‘s just an unfortunate coexistence of wording.
- Since GML does not support facets, the facet property of the DATEX II “String” cannot be mapped. This leads to Character Strings of unlimited length in instances of the GML application schema, which is acceptable. Some further specific datatypes are affected as well.
- For some DATEX II generic datatypes (e.g. Base64Binary or Language), no GML equivalent exists, but at least in EA it is possible to directly use the XML simple types instead.
- For the DATEX II generic datatypes Versioned Reference and MultilingualString it is necessary to map them to a string, i.e. references are simplified to String-Level, and Strings can only be expressed in one language in the GML-model, not in multiple languages (these two DATEX II datatypes are defined in a customised XSD-Snippet, which results in the restrictions).
- The DATEX II specific datatypes cannot be mapped to the GML model. Rather, the corresponding attributes using such a type are directly mapped to the underlying generic type. In essence, this means that an additional level of encapsulation is eliminated in the GML model, but this causes loss of information of the type itself and its underlaying definition. Anyway, due to the construction of most specific datatypes and their very specific usage, this can be accepted (see also example in section 2.2 of Annex A).
- DATEX II classes with stereotype D2Identifiable are mapped to UML classes with stereotype FeatureType. In DATEX II, you usually have only one root element (with stereotype D2ModelRoot). By this construction, now all D2Identifiables are mapped into root elements in the GML model. Basically, this is no disadvantage but rather an improvement.
- Extensions and External model elements cannot be mapped to the GML model.

5 Generation of an ISO 19103 compliant model

To achieve the goal of creating a GML application schema consists of two steps. Step one aims to create an ISO 19103 compliant model based on the DATEX II UML model. Step two aims to create the DATEX II GML application schema as described in chapter 6.

Both steps should be done in an automated and repeatable way. The usage of the Java programming language is preferred since Java is a widely used state-of-the-art technology.

Based on the following facts the Automation API of the Sparx Systems Enterprise Architect was used:

- The feature description covers our needs to transform and create a new UML model.
- ShapeChange uses the API, too.
- A Java API is available.

The API issued by Sparx Systems is described as follows:

"Enterprise Architect has a formidable set of built-in features for working with models, but it also provides a range of environments for accessing and manipulating the contents of a repository programmatically. This is an extremely powerfully facility that gives you unlimited ability to query and manipulate models, add to the Enterprise Architect user interface, generate reports, and even create support for new modeling languages. The Automation Interface gives you access to the Object Model, which is an easy to use and well defined set of objects with properties and methods that can be used to query and manipulate the repository and its contents, shielding the programmer from having to know the underlying repository data structures."

5.1 Execution options and requirements

The execution of the Automation API must be done out of a Java program. Therefore, a runnable Java program must be built.

Technical requirements:

- Eclipse IDE,
- Maven,
- Java 8,
- Enterprise Architect v7.5 or later is required,
- Copy SSJavaCom.dll located in *<EA installation folder>/Java API* to *<Windows folder>/System32* (on a 32-bit machine) or to *<Windows folder>/SysWOW64* (on a 64-bit machine). On a 64-bit machine, verify that you are using a 32-bit version of Java (e.g. using the command "java -version") since Enterprise Architect is a 32-bit application.

To build the *DatexIIForGml* Java program, the eaapi.jar must be installed in the local maven repository because it is not available in a public repository. To install the eaapi.jar in your local maven repository, use following Maven goal:

```
mvn install:install-file -Dfile=eaapi.jar
-DgroupId=org.sparx -DartifactId=eaapi
-Dversion=14.0.1423 -Dpackaging=jar
```

The main method is part of the main class DatexIIForGml. Therefore, you have to use this class in your Eclipse Run Configuration to run a rule transformation. The name of the input file must be specified in the command line. Be aware, that the Java program doesn't support the handling of file names with whitespaces. For example, if you want to process file DATEX_II_PIM.EAP use following command line parameter:

```
-eapFile /home/datex/ DATEX_II_PIM.eap
```

5.2 Operational steps

To create a new ISO 19103 compliant model, following steps and input files are necessary. (Prerequisite is a functioning installation of Sparx Systems Enterprise Architect).

Input file:

- The DATEX II UML model in EAP format.

Output file:

- A new generated ISO 19103 compliant UML model in EAP format (Datex_II_ISO_19103_Compliant.EAP).

Automated steps:

- Read input EA project and create input repository,
- create a new Enterprise Architect repository (output repository),
- apply transformation rules to input repository and store the results in the output repository,
- write output repository result to a newly created Enterprise Architect project.

5.2.1 Note to the Sparx Systems Automation API

As to-date all transformation rules could be applied and no significant restrictions by the API are known.

```
if(inElem.GetStereotype().equals("D2Identifiable") {
   Element elem = outputRepo.GetElements().AddNew(prefix+inElem.GetName(),
"Class");
   elem.SetStereotype("GML::FeatureType");
   elem.SetNotes(elements.GetTaggedValues().GetByName("definition").GetValue());
   elem.Update();
   outputRepo.Update();
   outputRepo.GetElements().Refresh();
   // Apply Transformation Rule 4.4
   Element rule = output.GetElements().GetByName(prefix + inElem.GetName());

   TaggedValue tagNoPropertyType =
rule.GetTaggedValues().GetByName("noPropertyType");
   tagNoPropertyType.SetValue("true");
   tagNoPropertyType.Update();

   outputPackage.Refresh();
}
```

Fig. 19: Code snippet – example of a transformation rule

Performance speed is acceptable and the processing of an UML model happens within a few minutes.

Unfortunately, the Java API description (Javadoc) is of no good quality. Therefore, a lot of trial and error is sometimes needed or the developer depends on good examples from the internet. An initial slow development speed is the result, but it improves after a while as you get to know the API principles better during implementation.

5.2.2 Transformation snippet

Figure 19 contains a small snippet to illustrate how a typical transformation rule, as documented in Annex A, is implemented via the Automation API.

5.3 Sample configuration

There are no configurations outside the Java program that need to be done.

6 Generation of the DATEX II GML application schema

During the proof-of-concept study, two possibilities of GML XSD schema generation have emerged. This chapter introduces the method of use and the results. ShapeChange was used initially as the preferred tool. During the prototype implementation, another possibility within Enterprise Architect was discovered. In the future it is also possible to provide the project results with the developed tool as open source for download under https://www.datex2.eu/support/tooling by the Federal Highway Research Institute.

6.1 ShapeChange

In order to create a GML application schema from the PIM′, the open source tool ShapeChange (https://shapechange.net/) version 2.6.0 (11. September 2018) is examined. ShapeChange is a Java tool that takes application schemas constructed according to ISO 19109 from a UML model and derives implementation representations.

The source code of ShapeChange is available under the GNU General Public License. Therefore ShapeChange is highly customizable if there are any needs to change or enhance functionality. The tool is used out-of-the-box and can if necessary be customized.

The main functionality of ShapeChange is to generate XML Schema representations of application schemas, encoded as a set of XML Schema documents (XSDs). The conversion rules from UML to XML Schema are based on the encoding rules specified in Annex E of the GML 3.2 standard, in the GML 3.3 standard and ISO/TS 19139:2007 – as well as a series of extensions to the standardized rules.

The most commonly used target representation is XML Schema and the following standardized encoding rules are supported:

- GML 3.2 encoding rule for GML application schema,
- GML 3.3 extensions,
- ISO/TS 19139 encoding rule,
- INSPIRE encoding rule.

ShapeChange provides these basic encoding rules, but it is possible to enhance a basic encoding rule by configuration at any times. Even new rules can be implemented and added to a specific rule configuration.

ShapeChange directly accesses Enterprise Architect models via their Java API, it can read XMI 1.0 and directly access GSIP model databases. Enterprise Architect 7.5 or later is required.

6.1.1 Execution options and requirements

ShapeChange execution options:

- command line,
- with or without dialog,

- invoke from Enterprise Architect.

Technical requirements:

- Java 1.8 or later,
- other open source libraries (i.e. for XSL transformations),
- Enterprise Architect v7.5 or later is required,
- Copy SSJavaCom.dll located in *<EA installation folder>/Java API* to *<Windows folder>/System32* (on a 32-bit machine) or to *<Windows folder>/SysWOW64* (on a 64-bit machine). On a 64-bit machine, verify that you are using a 32-bit version of Java (e.g. using the command "java -version") since Enterprise Architect is a 32-bit application.

6.1.2 Operational steps

To create a new GML application schema, following steps and input files are necessary. (Prerequisite is a functioning installation of ShapeChange and Enterprise Architect).

Input files:

- A valid PIM′ in EAP format,
- a suitable ShapeChange configuration file (i.e. datex.cfg).

Open a windows shell and execute:

```
java -jar ShapeChange-2.6.0.jar
-Dfile.encoding=UTF-8 -c datex.cfg
```

After the execution is complete, you will find:

- log messages (contains warnings and errors),
- the GML application schema.

6.1.3 Sample configuration

The sample configuration file for ShapeChange is shown in figure 20. Some parts have been shortened for readability.

6.1.4 Transformation result

After all the rules have been applied and the transformation has been done in an ISO 19103 compliant way, ShapeChange failed to generate a valid schema. The cause of the emerging errors are unknown at this time. An example of two error messages can be found in figure 21.

6.2 Enterprise Architect

In order to create a GML application schema from the PIM′, the Enterprise Architect provides a built-in functionality.

```
<ShapeChangeConfiguration>
<input>
  <parameter name="inputModelType" value="EA7"/>
    <parameter name="inputFile" value="datex.EAP"/>
    <parameter name="publicOnly" value="true"/>
    <parameter name="checkingConstraints" value="enabled"/>
    <parameter name="sortedSchemaOutput" value="true"/>
    <xi:include
href="http://sh.net/resources/config/StandardAliases.xml"/> </input>
<log>
  <parameter name="reportLevel" value="INFO"/>
  <parameter name="logFile" value="logs/datex.xml"/>
</log>
<targets>
  <TargetXmlSchema class="XmlSchema" mode="enabled">
    <targetParameter name="outputDirectory" value="XML"/>
    <targetParameter name="defaultEncodingRule" value="iso19136_2007"/>
    <targetParameter name="sortedOutput" value="true"/>
    <xi:include href="http://sh.net/resources/config/StandardRules.xml"/>
    <xi:include
href="http://sh.net/resources/config/StandardNamespaces.xml"/>
    <xi:include
href="http://sh.net/resources/config/StandardMapEntries.xml"/>
 </TargetXmlSchema>
</targets>
</ShapeChangeConfiguration>
```

Fig. 20: Code snippet – example of a configuration file

```
<Error message="No type can be provided for the property 'com-
AxleWeight.com-axleWeight'.">
<Detail message="Context: Property 'Model::D2Payload::Common::com-
Classes::com-Vehicle::com-AxleWeight::com-axleWeight'"/>
</Error>
<Error message="No type can be provided for the property 'com-
AxleWeight.com-maximumPermittedAxleWeight'.">
<Detail message="Context: Property 'Model::D2Payload::Common::com-
Classes::com-Vehicle::com-AxleWeight::com-maximumPermittedAxleWeight'"/>
</Error>
```

Fig. 21: Code snippet – occurred errors during transformation process

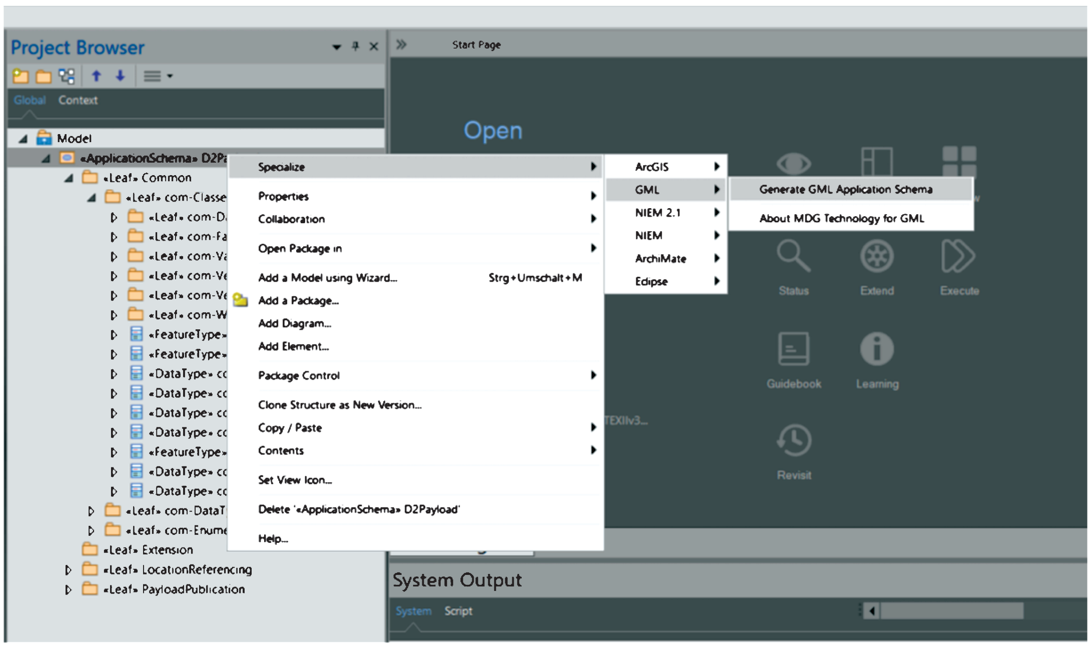


Fig. 22: Generated application schema

6.2.1 Execution options and requirements

Technical requirements:

- Enterprise Architect v7.5 or later is required.

6.2.2 Operational steps

To create a new GML application schema, following steps and input files are necessary. (prerequisite is an operative Enterprise Architect installation).

Input files:

- A valid PIM′ in EAP format.

The following steps must be executed:

- Open EAP file in Enterprise Architect,
- unfold model element,
- right-click (context menu) on element with stereotype application schema,
- specialize (see screenshot),
- GML (see screenshot),
- generate GML application schema (see figure 22),
- a new window opens up, select Application Schema of the model,
- select output file name,
- generate.

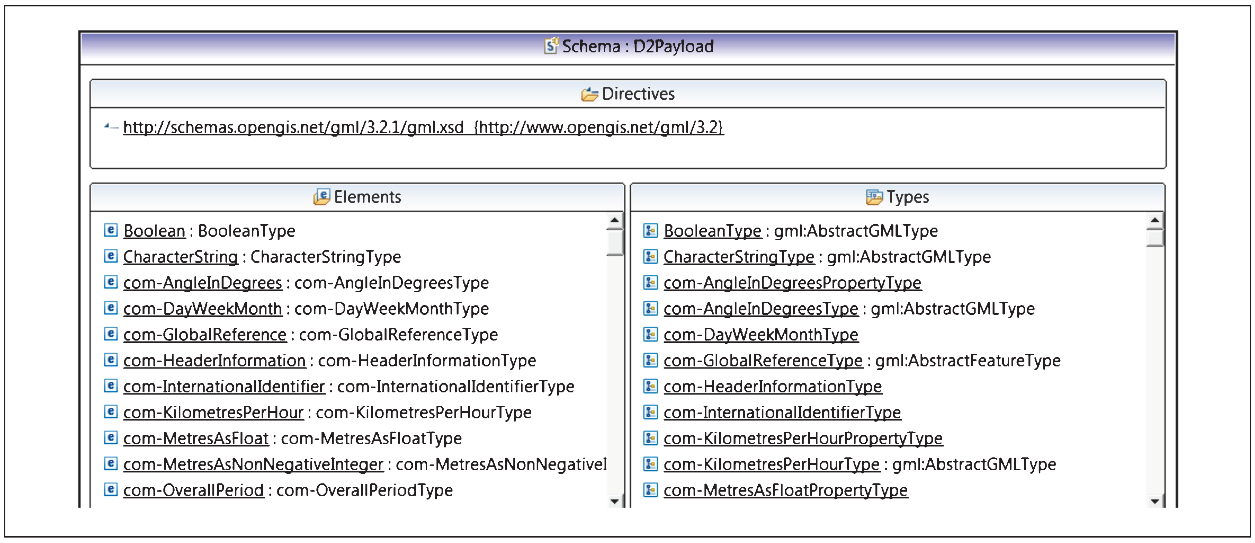
6.2.3 Sample Configuration

There is no need for an extra configuration for this task as it is a built-in function.

6.3 GML XSD schema as transformation result

An XSD is a formal specification of how an XML document can be formed. The W3C XML Schema 1.0 Recommendation defines an XML schema language. Its salient characteristics are defined in XML 1.0 (ISO 8879). As a result of the transformation process, a valid DATEX II GML application schema was recieved. The XML schema represents the interrelationship between the attributes and elements of an XML DATEX II object (see figure 23).

In the sample extraction in figure 24, the general structure and the structural elements of an XSD schema can be analysed. In this example the example structure of an AlertCLocation element can be observed.



Schema : D2Payload

Directives

http://schemas.opengis.net/gml/3.2.1/gml.xsd {http://www.opengis.net/gml/3.2}

Elements

- Boolean : BooleanType
- CharacterString : CharacterStringType
- com-AngleInDegrees : com-AngleInDegreesType
- com-DayWeekMonth : com-DayWeekMonthType
- com-GlobalReference : com-GlobalReferenceType
- com-HeaderInformation : com-HeaderInformationType
- com-InternationalIdentifier : com-InternationalIdentifierType
- com-KilometresPerHour : com-KilometresPerHourType
- com-MetresAsFloat : com-MetresAsFloatType
- com-MetresAsNonNegativeInteger : com-MetresAsNonNegativel
- com-OverallPeriod : com-OverallPeriodType

Types

- BooleanType : gml:AbstractGMLType
- CharacterStringType : gml:AbstractGMLType
- com-AngleInDegreesPropertyType
- com-AngleInDegreesType : gml:AbstractGMLType
- com-DayWeekMonthType
- com-GlobalReferenceType : gml:AbstractFeatureType
- com-HeaderInformationType
- com-InternationalIdentifierType
- com-KilometresPerHourPropertyType
- com-KilometresPerHourType : gml:AbstractGMLType
- com-MetresAsFloatPropertyType

Fig. 23: Elements and types of the DATEX II GML application schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generated by Enterprise Architect 14.0.1423 ( Build: 1423 )-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" element-
FormDefault="qualified" targetNamespace="D2Payload" xmlns:d2="D2Payload"
xmlns:gml="http://www.opengis.net/gml/3.2">
<xs:import namespace="http://www.opengis.net/gml/3.2" schemaLoca-
tion="http://schemas.opengis.net/gml/3.2.1/gml.xsd"/>
 <xs:element name="com-GenericPublication" type="d2:com-
GenericPublicationType" substitutionGroup="d2:com-PayloadPublication">
  <xs:annotation>
   <xs:documentation>A publication used to make level B extensions at the
publication level.</xs:documentation>
  </xs:annotation>
 </xs:element>
<xs:element name="loc-AlertCLocationCode" type="d2:loc-
AlertCLocationCodeType" substitutionGroup="gml:AbstractGML">
 <xs:annotation>
  <xs:documentation>A positive integer number (between 1 and 63 487) which
uniquely identifies a pre-defined Alert C location defined within an Alert-C
table.</xs:documentation>
 </xs:annotation>
</xs:element>
<xs:complexType name="loc-AlertCLocationType">
</xs:complexType>
[…]
</xs:schema>
```

Fig. 24: Code snippet – example structure of an AlertCLocation element

7 Validation of the GML application schema

The creation of a GML application schema is done in two steps. Step one aims to create an ISO 19103 compliant model based on the DATEX II UML model as described in chapter 5. Step two aims to create the DATEX II GML application schema as described in chapter 6.

After the transformation rules were completely implemented, the generation of the GML application schema was successfully carried out with Enterprise Architect.

7.1 Validation of the data model against ISO 19103

Both tools allow to transform into GML application schema (XSD), which automatically contains a validation against ISO 19103. Since OKSTRA is also fullfilling ISO 19103 requirements, an exchange of dynamic information about roadside objects between the created DATEX II data model and the OKSTRA model is therefore possible.

7.1.1 Validation using ShapeChange

ShapeChange provides after each run a XML result file. As already mentioned in chapter 6.1.4, there was no successful validation result. A report file with warnings and errors can be found in figure 25. For better readability, the result file has been shortened. The symbols **** mark deletions due to repetitions.

```
<?xml version="1.0" encoding="UTF-8"?>
<ShapeChangeResult xmlns="http://www.interactive-
instruments.de/ShapeChange/Result" xmlns:r="http://www.interactive-
instruments.de/ShapeChange/Result" config="datex.xml" end="Thu Nov 29
22:19:07 CET 2018" start="Thu Nov 29 22:17:38 CET 2018" version="2.6.0">
 <Messages>
  <Info message="---------- Semantic validation of ShapeChange configura-
tion: START ----------"/>
  <Info message="---------- Semantic validation of ShapeChange configura-
tion: COMPLETE ----------"/>
  <Info message="Connecting to C:\Program Files (x86)\ShapeChange-
2.6.0\test.EAP"/>
  <Info message="Connected to C:\Program Files (x86)\ShapeChange-
2.6.0\test.EAP"/>
  <Info message="Starting reading C:\Program Files (x86)\ShapeChange-
2.6.0\test.EAP"/>
  <Warning message="Stereotype &lt;&lt;GML::property&gt;&gt; not support-
ed for UML model elements of type 'AssociationEnd'.">
  <Detail message="Context: Property 'Model::D2Payload::Common::com-
Classes::com-GlobalReference::externalPublisher'"/>
  </Warning>
****
  <Error message="No type can be provided for the property 'vms-
VmsTextDisplayCharacteristics.vms-textPositionY'.">
   <Detail message="Context: Property 'Mod-
el::D2Payload::PayloadPublication::Vms::vms-Classes::vms-VmsRelated::vms-
VmsTextDisplayCharacteristics::vms-textPositionY'"/>
  </Error>
  <Info message="Executed target class
'de.interactive_instruments.ShapeChange.Target.XmlSchema.XmlSchema' for
input ID: 'INPUT'.&#10;--------------------------------------------------
"/>
  </Messages>
 <Results>
  <Result href="file:/C:/Program%20Files%20(x86)/ShapeChange-
2.6.0/XML/INPUT/D2Payload.xsd" scope="D2Payload" target="XML Sche-
ma">D2Payload.xsd</Result>
 </Results>
</ShapeChangeResult>
```

Fig. 25: Code snippet – example structure of an occurred transformation error in ShapeChange

7.1.2 Validation built-in function in Enterprise Architect

Enterprise Architect provides after each run a protocol file. As mentioned in chapter 6.3 a successful generation was achieved. In the following report file (figure 26), all processed parts of the input file are listed. For better readability, the result file has been shortened. The four asterisks written in bold stand for entry repetition.

7.2 Validation of the data model against UML

The DATEX II UML model in EAP format has been validated by using the Enterprise Architect Model Validation to check the model against known UML rules as well as any constraints defined within the model, using the Object Constraint Language (OCL). The following properties of the UML model have been validated:

- Elements, the elements and its children, features (attributes and operations) and relationships (connectors).
- Diagrams, the diagrams themselves (for correctness) as well as any elements and connectors within the diagram.
- Packages, the packages and all sub packages, elements, connectors and diagrams.

The validation was performed in Enterprise Architect and the results displayed in the Output window.

7.3 Validation of the GML application schema

To guarantee the integrity and usability, the created DATEX II GML schema was validated. Because of the import-statement listed below, the underlaying GML schema is downloaded during the validation process:

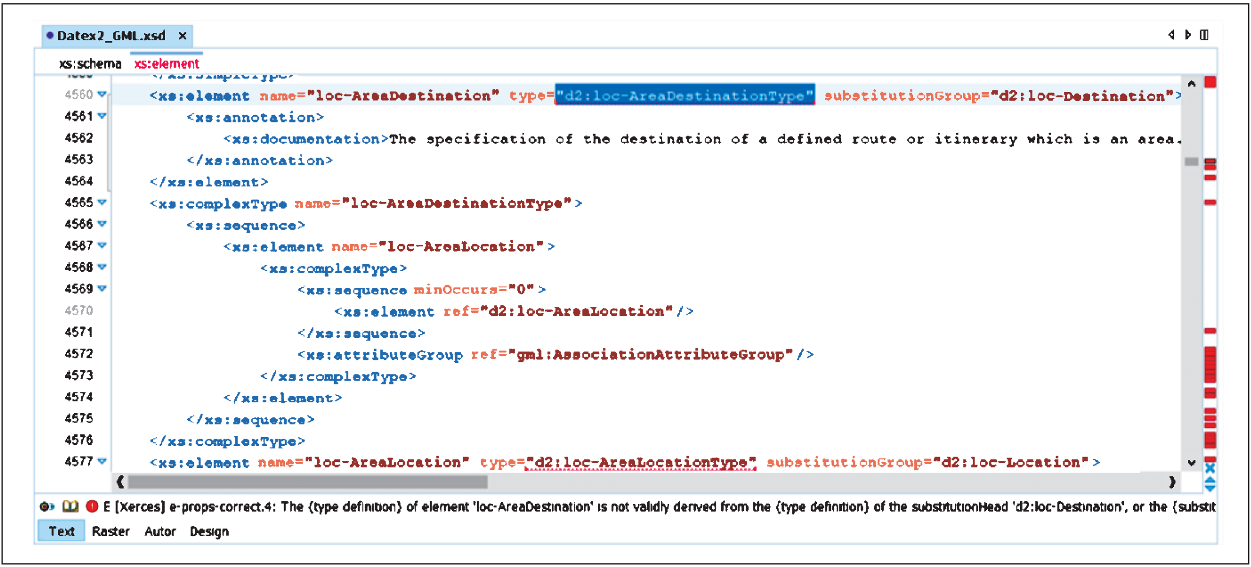
```
<xs:import namespace="http://www.opengis.net/gml/3.2" schemaLoca-
tion="http://schemas.opengis.net/gml/3.2.1/gml.xsd"/>,
```

This takes some additional time, and furthermore, not all XML editors are capable of this functionality (for this project, the produkt oXygen[6] was used). The result is the proof that the created schema is valid, i.e. free of syntactical errors. This was by no means always the case during the project phase. During the early implementation phase, problems arise with the mapping software that led to errors.

[6] https://www.oxygenxml.com/

```
Validating Package Details :
 * D2Payload
 * vms-VmsTablePublication
 * vms-VmsPublication
 * vms-Enumerations
 * vms-VmsUnit
****
 * com-Fault
 * com-DataValue
 * com-Classes
 * Common
Loading Package Contents
Generating Schema for Package : D2Payload
  Mapping Package Contents :
   * com-GenericPublication
   * com-GlobalReference
   * com-GroupOfVehiclesInvolved
   * com-HazardousMaterials
****
   * vms-VmsDynamicCharacteristics
   * vms-VmsSetting
   * vms-VmsUnit
  Adding External Schema References
Saving to File : C:\Users\Desktop\D2Payload.xsd .. OK
Generation Complete
```

Fig. 26: Code snippet – validation result



```
<xs:element name="loc-AreaDestination" type="d2:loc-AreaDestinationType" substitutionGroup="d2:loc-Destination">
    <xs:annotation>
        <xs:documentation>The specification of the destination of a defined route or itinerary which is an area.
    </xs:annotation>
</xs:element>
<xs:complexType name="loc-AreaDestinationType">
    <xs:sequence>
        <xs:element name="loc-AreaLocation">
            <xs:complexType>
                <xs:sequence minOccurs="0">
                    <xs:element ref="d2:loc-AreaLocation"/>
                </xs:sequence>
                <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>
<xs:element name="loc-AreaLocation" type="d2:loc-AreaLocationType" substitutionGroup="d2:loc-Location">
```

```
E [Xerces] e-props-correct.4: The {type definition} of element 'loc-AreaDestination' is not validly derived from the {type definition} of the substitutionHead 'd2:loc-Destination', or the {substit
```

Fig. 27: Elements and types of the DATEX II GML application schema and occurred errors during the vaildation

For example, binding problems (finding the correct types in the model structures) or problems with Generalisations led to useless, expanded XML structures.

One issue to mention here specifically is, that generalisations with the subclass of type "DataType" are not mapped correctly in the schema when using the built-in Enterprise Architect mapping function. The schema got broken in this case (see figure 27).

From the GML standard, it should be allowed to construct a generalisation between two DataTypes. To limit the use of resources on this problem, the project team decided to introduce an additional mapping rule and to change the stereotypes in these cases to "FeatureType" – see Rule 7-3 in Annex A – DATEX II as GML application schema – Transformation rules.

In the final validation step, an example test case was set up in order to determine whether our previously generated GML application schema satisfies the requirements and works correctly. The process of developing test cases helps to find problems in the requirements or design of our defined transformation approach.

7.4 Example German Roadworks Profile

The German DATEX II profile for roadworks[7] was used as a test case in this project. Officially, this profile is only available in Version 2.3 of DATEX II, not in Version 3.0.

Thus, the profile was remodelled in Version 3.0 for the use of this project. Extensions that existed in the 2.3 model (for example on classes SituationRecord, Impact and Roadworks) are either already included in the 3.0 model anyway or have been reengineered as extensions in 3.0.

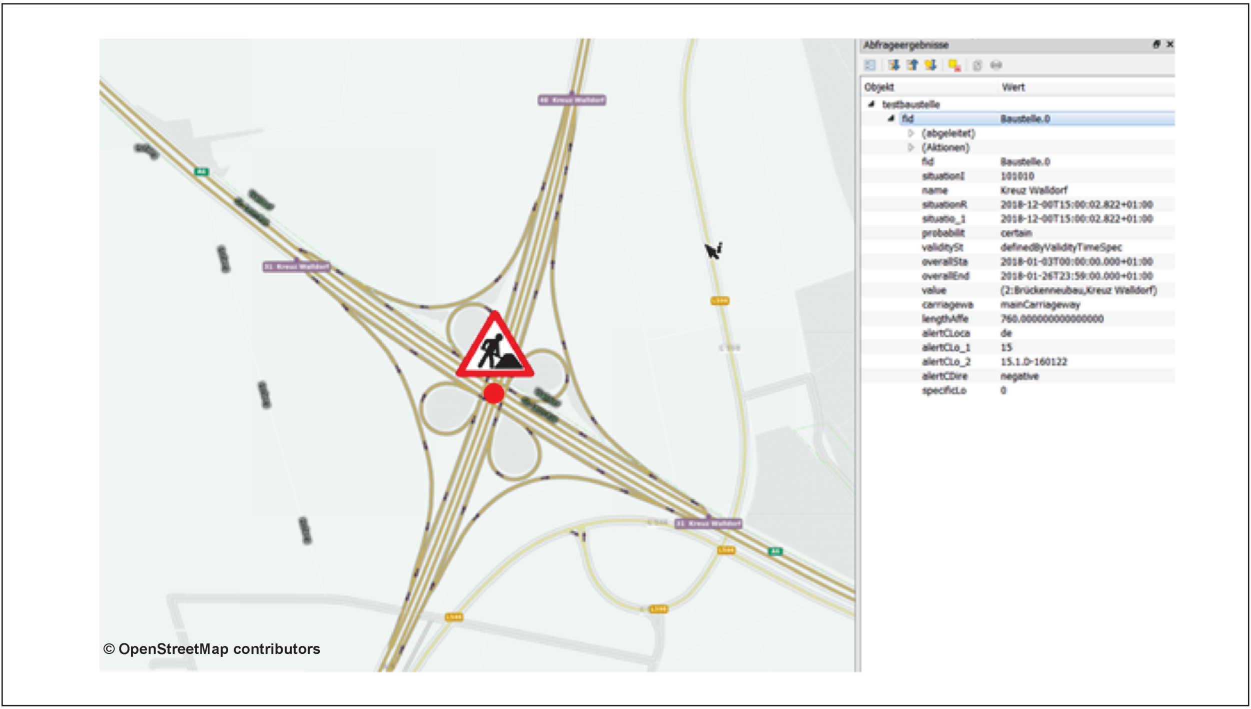
A DATEX II profile is usually generated with the DATEX II Tool by selecting and deselecting elements (classes, attributes, literals) and modifying multiplications. It is usually not done at the level of Enterprise Architect (the Enterprise Architect model should always be Level A, optional with Level B extensions, but not being profiled as such). But in this project, the transformation into GML is done on Enterprise Architect level. For this reason and this special example, the above policy had to be ignored. Thus, the German roadworks profile was tailored directly within Enterprise Architect, e.g. all unused elements were deleted from the model (classes, attributes, literals) and multiplicities where needed adapted. Therefore, the EAP model is by no means Level A compliant, because elements are missing. Please note, that from the point of view of quality assurance, it is not yet suitable to be published as a roadworks profile of version 3.0 – it is just an example of use and not an official version.

Once the roadworks profile has been added to the EAP model as depicted in chapter 3, figure 5, a XML from the transformed GML conform XSD

[7] https://www.mdm-portal.de/service/hilfe/datex-ii-profile.html

```xml
<?xml version="1.0" encoding="UTF-8"?>
<d2:sit-SituationPublication xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:gml="http://www.opengis.net/gml/3.2"
 xmlns:d2="D2Payload"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="D2Payload Datex2_GML.xsd">
  <d2:com-defaultLanguage>de</d2:com-defaultLanguage>
  <d2:publicationCreator>
    <d2:com-InternationalIdentifier>
      <d2:com-country>de</d2:com-country>
      <d2:com-nationalIdentifier>DE-MDM-Baustelleninformationssystem des Bundes und der Länder</d2:com-nationalIdentifier>
    </d2:com-InternationalIdentifier>
  </d2:publicationCreator>
  <d2:com-publicationTime>2018-12-04T18:13:51.0</d2:com-publicationTime>

  <d2:sit-Situation>
    <d2:sit-Situation gml:id="x3484d88a-4139-4d68-96b4-638aabfa2fec">
      <d2:com-HeaderInformation>
        <d2:com-HeaderInformation>
          <d2:com-informationStatus>test</d2:com-informationStatus>
        </d2:com-HeaderInformation>
      </d2:com-HeaderInformation>
      <d2:informationManager>
        <d2:com-InternationalIdentifier>
          <d2:com-country>de</d2:com-country>
          <d2:com-nationalIdentifier>DE-MDM-Baustelleninformationssystem des Bundes und der Länder</d2:com-nationalIdentifier>
        </d2:com-InternationalIdentifier>
      </d2:informationManager>

      <d2:sit-SituationRecord>
        <d2:sit-ConstructionWorks gml:id="x59864aae-cb27-49ae-856b-dabda8112903">
          <d2:com-Validity>
            <d2:com-Validity>
              <d2:com-validityStatus>active</d2:com-validityStatus>
              <d2:validityTimeSpecification>
                <d2:com-OverallPeriod>
                  <d2:com-overallStartTime>2018-12-05T13:00:00.0</d2:com-overallStartTime>
                </d2:com-OverallPeriod>
              </d2:validityTimeSpecification>
            </d2:com-Validity>
          </d2:com-Validity>
          <d2:loc-LocationReference>
            <d2:loc-SingleRoadLinearLocation>
              <d2:coordinatesForDisplay>
                <gml:Point srsName="EPSG:4326">
                  <gml:pos>8.6165497 49.2742921</gml:pos>
                </gml:Point>
              </d2:coordinatesForDisplay>
              <d2:loc-AlertCLinear>
                <d2:loc-AlertCMethod2Linear>
                  <d2:loc-alertCLocationCountryCode>D</d2:loc-alertCLocationCountryCode>
                  <d2:loc-alertCLocationTableNumber>1</d2:loc-alertCLocationTableNumber>
                  <d2:loc-alertCLocationTableVersion>17.0</d2:loc-alertCLocationTableVersion>
                  <d2:loc-AlertCDirection>
                    <d2:loc-AlertCDirection>
                      <d2:loc-alertCDirectionCoded>negative</d2:loc-alertCDirectionCoded>
                      <d2:loc-alertCAffectedDirection>aligned</d2:loc-alertCAffectedDirection>
                    </d2:loc-AlertCDirection>
                  </d2:loc-AlertCDirection>
                  <d2:loc-AlertCMethod2PrimaryPointLocation>
                    <d2:loc-AlertCMethod2PrimaryPointLocation>
                      <d2:loc-AlertCLocation>
                        <d2:loc-AlertCLocation>
                          <d2:loc-alertCLocationName>Walldorf</d2:loc-alertCLocationName>
                          <d2:loc-specificLocation>11600</d2:loc-specificLocation>
                        </d2:loc-AlertCLocation>
                      </d2:loc-AlertCLocation>
                    </d2:loc-AlertCMethod2PrimaryPointLocation>
                  </d2:loc-AlertCMethod2PrimaryPointLocation>
                  <d2:loc-AlertCMethod2SecondaryPointLocation>
                    <d2:loc-AlertCMethod2SecondaryPointLocation>
                      <d2:loc-AlertCLocation>
                        <d2:loc-AlertCLocation>
                          <d2:loc-alertCLocationName>Lußhardt</d2:loc-alertCLocationName>
                          <d2:loc-specificLocation>56712</d2:loc-specificLocation>
                        </d2:loc-AlertCLocation>
                      </d2:loc-AlertCLocation>
                    </d2:loc-AlertCMethod2SecondaryPointLocation>
                  </d2:loc-AlertCMethod2SecondaryPointLocation>
                </d2:loc-AlertCMethod2Linear>
              </d2:loc-AlertCLinear>
            </d2:loc-SingleRoadLinearLocation>
          </d2:loc-LocationReference>
          <d2:generalPublicComment>
            <d2:sit-Comment>
              <d2:sit-comment>Brückenausbau</d2:sit-comment>
            </d2:sit-Comment>
          </d2:generalPublicComment>
          <d2:sit-situationRecordCreationTime>2018-12-04T18:13:51.0</d2:sit-situationRecordCreationTime>
          <d2:sit-situationRecordVersionTime>2018-12-04T18:13:51.0</d2:sit-situationRecordVersionTime>
          <d2:sit-probabilityOfOccurrence>certain</d2:sit-probabilityOfOccurrence>
          <d2:sit-Subjects>
            <d2:sit-Subjects>
              <d2:sit-subjectTypeOfWorks>bridge</d2:sit-subjectTypeOfWorks>
            </d2:sit-Subjects>
          </d2:sit-Subjects>
          <d2:sit-constructionWorkType>constructionWork</d2:sit-constructionWorkType>
        </d2:sit-ConstructionWorks>
      </d2:sit-SituationRecord>

      <d2:sit-situationVersionTime>2018-12-04T18:13:51.0</d2:sit-situationVersionTime>
    </d2:sit-Situation>
  </d2:sit-Situation>

</d2:sit-SituationPublication>
```

Fig. 28: Code snippet – a test roadwork site in Germany



Objekt	Wert
testbaustelle	
fid	Baustelle.0
(abgeleitet)	
(Aktionen)	
fid	Baustelle.0
situationI	101010
name	Kreuz Walldorf
situationR	2018-12-00T15:00:02.822+01:00
situatio_1	2018-12-00T15:00:02.822+01:00
probabilit	certain
validitySt	definedByValidityTimeSpec
overallSta	2018-01-03T00:00:00.000+01:00
overallEnd	2018-01-26T23:59:00.000+01:00
value	(2:Brückenneubau,Kreuz Walldorf)
carriagewa	mainCarriageway
lengthAffe	760.000000000000000
alertCLoca	de
alertCLo_1	15
alertCLo_2	15.1.0-160122
alertCDire	negative
specificLo	0

Fig. 29: Construction site visualized in a GIS (using prototype GML application schema DATEX II, Version 3.0)

file was generated. The example in figure 28 shows a test roadwork site in Germany and how it is structurally modelled as a GML compliant schema.

Furthermore, the shown test construction (figure 28) site is visualized as a geographical feature on a map, since a valid GML application schema has been created. This allows to represent geographical features and objects as a GML according to the Open Geospatial Consortium (OGC). Figure 29 shows the exemplarily modelled construction site visualized on a map. In the attribute data of the GML point feature the previous elements of the XMI file can be observed.

8 Open issues

Some points have been identified that need to be addressed and might still require some effort:

- Class diagrams (figures) in Enterprise Architect

  The current mapping software focusses on the transformation of the model itself. Class diagrams, which are usually part of a (DATEX II) UML model, are not transferred into the GML target model. For this reason, the resulting Enterprise Architect model comes without any class diagrams. If you want to have a visual impression of the result, you have to create these diagrams yourself.

  By using the mapping software, Enterprise Architect creates new elements in each pass, i.e. each element will get a different internal id each time you create a new mapping (usually, each element keeps its unique id during the lifetime of the DATEX II UML model, unless the element itself is not deleted due to structural changes). For this reason, it is not possible to simple transfer class diagrams – once created – to a new version of the mapping, because the elements in the class diagram are not recognized any longer (such a diagram will be empty). In other words: Every time you use the mapping software, diagrams need to be rebuilt by hand.

  An improvement of the mapping software for the future could be, to transfer also the class diagrams into the resulting GML model.

- Order of elements

  The order of elements (e.g. attributes, aggregations) is determined in the DATEX II UML model by the tagged value “order” and in the GML UML by the sequenceNumber property (at least it has got this effect). Filling this sequence Number is not yet done by the mapping software, as there have been technical difficulties to assign this property. Due to limited resources, this problem has not been resolved so far.

As a consequence, the order of elements might diverge when creating schemas out of the DATEX II and the GML model, which is acceptable at this stage.

- Enterprise Architect bug

  The Enterprise Architect feature to convert a GML model into an XML schema does not create proper results, if the subclass of a generalisation is of stereotype “DataType”. This is probably a bug of Enterprise Architect (see Rule A.7-3). For this reason, a special treatment was defined, which slightly changes the model semantics, because under these circumstances, objects which should be of stereotype “DataType” are mapped into a “FeatureType”. This change is acceptable, but it should be kept in mind, that it is not necessary from a methodologic point of view, but only because of this probable Enterprise Architect bug.

9 Conclusions

The results of the transformation approach with the implementation of the prototype have demonstrated the technical feasibility to convert DATEX II feeds into a valid GML application schema. Thus, with this report and its technical supplements, all interested persons are able to create a GML application schema out of a DATEX II UML model. For the current DATEX II “Level A” model in version 3.0 this already has been done exemplarily, but as the approach is generic, also customised DATEX II models (for example with Level B extensions) or further versions can be mapped.

Thus, it is possible now, to use standardised and OGC-compliant geoservices such as GDI-DE and IN-SPIRE without the need of manual redesigning a DATEX II model.

This report specifies the mapping for the UML model level of DATEX II. However, DATEX II profiles are specified on schema level, not on UML level (unless you do manual modifications and manipulations as it was done exemplarily for this report with the roadworks extension). Here, an automated feature of generating a consistent UML model on the base of a DATEX II profile would be helpful. In chapter 3, this problem was focussed and named with the missing “round trip” feature. Having such a feature, it would be even more easy to create a GML application schema out of a DATEX II profiles (on a visual level, the “round trip feature” already has been developed by the DATEX II PSA, but this does not fully satisfy the requirement raised here). Creating a direct tool to convert DATEX II XML to GML on a schema level would also enable the possibility to offer data convertion services.

Another feasible approach would be, to generate the mapping from DATEX II to GML directly on the level of XML instances, i.e., this procedure would ignore the UML model level as far as possible. Nevertheless, the necessary steps should show similarity to the specifications defined here. Most probably, once again a set of rules would need be defined to map all possible XML constructs. A first thought on this topic has been done for MultilingualStrings in this report, see chapter 4.3. An advantage of this approach would be, that the grade of automation even could be raised: Having clients communicating by DATEX II, a scenario could be to add this new kind of instance converter to directly produce GML compliant content for every message they send (or receive). These thoughts require some other programming paradigm, which would be no longer embedded in the Enterprise Architect system but could need some integration in client systems of customers. Of course, as a first step, also stand-alone instance converters could be an option.

For the specifications made in this report, it is recommended to publish them in the different stakeholder communities to prove their validity and robustness as well as their significance and importance.

For DATEX II, this could mean to upload the results on the DATEX II platform, so that users and other interested people can produce a mapping of the data models theirself and collect some first experiences with it.

But also for the GML-related communities, it is recommended to bring in some example mappings so that people can get a feeling on the commonalities of the two worlds and ease exchange between them. Open source projects (like doing filtering[8]

[8] Filtering of content is not supported in DATEX II, so further data processing with Open Source tools can be valuable.

with database scenarios, for example using a MongoDB) or the portablity into the GeoJSON world are fields, which could be taken into consideration for further testing and outreach work.

Finally, bringing the results into European standardisation (in the context of CEN/TC 278, Working Group 8 “Road traffic data” or other feasible groups) would be appreciated, too. A standard can raise importance and impact of this mapping as well as of the different involved modelling principles.

Figures

Annex A – DATEX II GML application schema – transformation rules

This document contains the instructions for transforming a DATEX II v3 UML model into a UML model that serves as a base for the transformation into a GML application schema.

A.1 Packages

See EN 16157-1 Section 7.9 Requirements on packages and GML V3.2.1 E.2.1.1.1 General (application schema, packages).

The GML application schema must be represented by a UML package with the stereotype ApplicationSchema. This package should contain all elements of the UML model. Except for the top-level package with stereotype ApplicationSchema, all other packages have no stereotype. Enterprise Architect provides the two stereotypes ApplicationSchema and Leaf for UML packages. Here the stereotype leaf shall be selected for the mapping of all but the top-level package. Hence, the top level D2Namespace package (in the current DATEX II v3 Level A model this is the D2Payload package) has to be mapped to an ApplicationSchema package of the same name.

There are no matching stereotypes for subsequent D2Namespace DATEX II packages and ExternalNamespace DATEX II packages in GML. To map subsequent D2Namespace DATEX II packages, a UML package with stereotype Leaf must be added. A naming convention is defined in chapter A.8 Namespaces for the content of this DATEX II namespace packages, so that there can be no naming conflicts with the content of other D2Namespace packages.

D2Package DATEX II packages are also mapped to UML packages with stereotype Leaf.

ExternalNamespace DATEX II packages: see chapter A.9 External model elements

Rule 1-1: For the top-level DATEX II package with stereotype D2Namespace (e.g. D2Payload in current DATEX II v3 Level A), create a UML package with stereotype ApplicationSchema and copy the name of the DATEX II package to the created UML package.

Rule 1-2: For every other DATEX II package with stereotype D2Namespace or D2Package create a UML package with stereotype Leaf and copy the name of the DATEX II package to the created UML package. For all content of such a D2namespace package, apply the naming rule specified in section 8.

Rule 1-3: Transfer the definition from the definition property of the DATEX II package to the documentation property of the created UML package (see **Note 7** below).

Rule 1-4: For the top-level DATEX II package with stereotype D2Namespace transfer the prefix from the prefix property of the DATEX II package to the tagged value xmlns of the created UML package.

Rule 1-5: For the top-level DATEX II package with stereotype D2Namespace transfer the version from the version property of the DATEX II package to the tagged value version of the created UML package.

Rule 1-6: Fill the xsdDocument property of the created GML ApplicationSchema package with the name of the created GML ApplicationSchema package with the prefix ".xsd" (e.g. If the DATEX II top-level package is called "D2Payload", the xsdDocument property of the created GML ApplicationSchema package would have the value "D2Payload.xsd". For every other package the xsdDocument property must be empty.

Rule 1-7: For every created package, transfer the version number from the package settings of the DATEX II package to the package settings of the created GML package.

Note 7: Enterprise Architect does not implement the documentation property. Here, the definition must be copied into the "Notes" field in the properties of the UML class.

A.2 Data types

See EN 16157-1 7.5 Requirements on datatypes and GML V3.2.1 E.2.1.1.5.

In DATEX II, datatypes are modelled by a UML class with stereotype D2Datatype and are grouped into generic and specific datatypes. They are called DATEX II generic datatypes or DATEX II specific datatypes in this document.

A.2.1 Generic data types

GML provides predefined basic types to which the DATEX II generic datatypes must be mapped - see table 1.

The DATEX II stereotype D2Datatype can be mapped to the GML stereotype Type. For each mappable DATEX II generic datatype a new UML class is created, to which the GML stereotype Type is assigned. The name of the DATEX II datatype cannot be adopted, but the name of the associated GML basic type must be used (see table 1). The content of the definition property of a DATEX II datatype can be transferred to the tagged value documentation of a Type GML class (see **Note 1**). Furthermore, the schemaType property of a DATEX II datatype can be transferred to the tagged value xmlSchemaType of a Type GML class.

Note 1: Enterprise Architect does not know the tagged value documentation. Here the definition must be copied into the "Notes" field of the class properties.

Since GML does not support facets, the facet property of the DATEX II "String" cannot be mapped. This leads to Character Strings of unlimited length in instances of the GML applications schema, which is acceptable.

For the DATEX II datatypes NonNegativeInteger, Url, Base64Binary and Language there are no corresponding GML basic types. As a fall-back, they are mapped to GML basic types (see table 1). When using Enterprise Architect for generating a GML ApplicationSchema, however, it is possible to use the XSD type directly rather than the corresponding GML basic type as an attribute type (see table 1). This does not work when using ShapeChange to generate a GML ApplicationSchema. The DATEX II datatypes MultilingualString, Reference and VersionedReference cannot be mapped as GML does not allow including XML code fragments. As a fall-back, they are mapped to the XSD string type.

Included schemas

In DATEX II, it is possible to include XML schema fragments in the UML model to specify special datatypes (by usage of tagged value "schemaTypeInclude"). This is done for the D2Datatypes Reference, VersionedReference and MultilingualString.

In general, this is also possible for datatypes created within a Level B extension, but for the time, no such usage is known.

Mapping these fragments automatically to GML is possible, but however a bit tricky, as the general approach of the mapping is a UML based mapping, and there is no correspondent feature for a GML-like UML model. For this reason, it becomes more feasible to ease the mapping of the three above-mentioned datatypes:

As a fall-back, Reference, VersionedReference and MultilingualString are mapped to the XSD string type.

Rule 2-1 (for DATEX II datatypes 1-10): Create a new UML class with Stereotype Type and use the name of the associated GML basic type (see table 1) as name of the created UML class.

Exception: For a "LongString", do not create the GML basic type CharacterString a second time, but use the "CharacterString" as a mapping for "LongString" anyway.

Rule 2-2 (for DATEX II datatypes 1-10): Copy the definition of the DATEX II datatype from the definition property into the tagged value documentation (see **Note 1**) of the created UML class.

Rule 2-3 (for DATEX II datatypes 1-10): Copy the XML schema type of the DATEX II datatype from the schemaType property to the tagged value xmlSchemaType of the created UML class.

Rule 2-4 (for DATEX II datatypes 11-15): If one of the DATEX II datatypes NonNegativeInteger, Url, Base64Binary and Language is to be assigned to an attribute, either use the corresponding XML schema type or the corresponding GML basic type as type of the attribute. For the DATEX II datatypes Reference, VersionedReference and MultilingualString either use the GML basic type CharacterString or the XML schema type string as type of the attribute (see **Note 2** and table 1).

Rule 2-5 (for DATEX II datatypes 1-10): Set the value of the noPropertyType tagged value to true (see **Note 6**).

Note 2: Using the corresponding XML schema type as type of an attribute works when generating a GML ApplicationSchema with Enterprise Architect but is not described in GML V3.2.1 and does not work for ShapeChange. Here the matching GML basic type must be used as type of the attribute.

Number	DATEX II datatype[9]	GML basic type[10]	XSD type
1	String	CharacterString	string
2	LongString	CharacterString	string
3	Boolean	Boolean	boolean
4	Float	Real	double
5	Integer	Integer	integer
6	DateTime	DateTime	dateTime
7	Double	Real	double
8	Date	Date	date
9	Time	Time	time
10	Decimal	Decimal	decimal
11	NonNegativeInteger	Integer	nonNegativeInteger/integer
12	Url	CharacterString	anyURI/string
13	Base64Binary	CharacterString	base64Binary/string
14	Language	CharacterString	language/string
15	Reference	CharacterString	string
16	VersionedReference	CharacterString	string
17	MultilingualString	CharacterString	string

Tab. 1: Mapping of DATEX II generic datatypes to GML Types

A.2.2 Specific data types

Each of the DATEX II specific datatypes in the current DATEX II v3 Level A model has a superclass which is one of the DATEX II generic datatypes. This is not possible in GML. Therefore, the DATEX II

[9] CEN EN 16157-1 Intelligent transport systems – DATEX II data exchange specifications for traffic management and information – part 1: Context and framework

[10] OGC 07-036: OpenGIS® Geography Markup Language (GML) Encoding Standard

specific datatypes are mapped to the DATEX II generic datatype of the superclass of the DATEX II specific datatype.

The definition tagged value of a DATEX II specific datatype cannot be mapped in general. To ensure that the definition of the specific DATEX II datatypes is not lost, it is added to the definition of the DATEX II Attributes to be mapped (see Attributes). Some definitions of the DATEX II Attributes already equal the definition of its specific DATEX II Datatype. If this is the case, the definition must not be added.

Example: The DATEX II specific datatype “KilometresPerHour”, derived from the DATEX II generic datatype “Float”, has the definition “A measure of speed defined in kilometres per hour”. Now look at the DATEX II attribute “speedValue” (located in the DATEX II class “SpeedValue”). The definition of “speedValue” states “A measure of speed defined in kilometres per hour”. Therefore, the semantic meaning of the DATEX II datatype “KilometresPerHour” can be understood clearly. The DATEX II attribute “speedValue” has the DATEX II datatype “KilometresPerHour” assigned, whereby the superclass of “KilometresPerHour” is the DATEX II datatype Float. According to the table 1, the DATEX II datatype Float is mapped to the GML basic type Real. In essence, the GML attribute "speedValue" would be of type Real (see also Rule 5.2).

Since GML does not support facets, the facet property of a DATEX II specific datatype cannot be mapped. However, in the current DATEX II v3 Level A model, facets only apply to a limited number of DATEX II specific datatypes (which are usually derived from String or (NonNegative)Integer). Ignoring facets allows a larger lexical value space for the corresponding attributes in instances of the GML applications schema, which is acceptable.

Rule 2-6: Map each of the DATEX II specific datatypes to the corresponding GML basic type of the DATEX II generic datatype it is derived from.

A.2.3 ExternalType

ExternalType DATEX II Datatypes: See chapter A.9 External model elements

A.3 Enumerations

See EN 16157-1 Section 7.6 Requirements on enumerations and 7.8 Requirements on literals as well as GML V3.2.1 E.2.4.8 UML classes (enumerations).

DATEX II enumerations with stereotype D2Enumeration, which are called DATEX II enumerations in this document, are equivalent to UML classes with stereotype Enumeration in GML. The definition described by the definition property of a DATEX II enumeration can be mapped to the documentation property (see **Note 4**) of the created UML class with stereotype Enumeration. Each D2Literal, which is called DATEX II literal in this document, is mapped to a UML attribute of the corresponding UML class stereotyped Enumeration. The definition is also copied from the definition property of a DATEX II literal to the tagged value documentation (see **Note 4**) of the created UML attribute. The value of the order property of a DATEX II literal can be copied to the sequenceNumber property of the created UML attribute.

Note 3: GML V3.2.1 E.2.1.1.3 stipulates that UML classes with stereotype Enumeration should have no order of attributes. By testing in Enterprise Architect, however, it can be observed that copying the order property of a DATEX II literal to the sequenceNumber property produces an ordered list of the literals.

Note 4: Enterprise Architect does not know the documentation property. Here, the definition of a DATEX II enumeration or a DATEX II literal must be copied into the "Notes" field of the created UML class or attribute.

Rule 3-1: Create a new UML class with stereotype Enumeration with the name of the original DATEX II enumeration modified by the rules in chapter A.9 External model elements.

Rule 3-2: Copy the definition of the DATEX II enumeration from the definition property to the documentation property of the created UML class.

Rule 3-3: Add a UML attribute in the created UML class for each DATEX II literal in the corresponding DATEX II enumeration and copy the name of the DATEX II literal to the created attribute.

Rule 3-4: Copy the definition of each literal from the definition property of a DATEX II literal to the documentation property of the created attribute (see **Note 4**).

Rule 3-5: Copy the value of the order property of each DATEX II literal to the sequenceNumber property of the created attribute (see **Note 3**).

A.4 Classes

See EN 16157-1 Section 7.4 Requirements on classes and GML V3.2.1 E.2.1.1.2 Classes.

DATEX II classes with stereotype D2Class can be mapped to UML classes with stereotype DataType. The definition of the D2Class DATEX II class can be copied to the documentation property (see **Note 5**).

Note 5: Enterprise Architect does not know the documentation property. Here, the definition must be copied into the "Notes" field of the UML class.

DATEX II classes with stereotype D2Identifiable can be mapped to UML classes with stereotype FeatureType. In GML UML classes with stereotype FeatureType are derived directly or indirectly from the gml:AbstractFeatureType. This in turn is derived from the gml:AbstractGMLType, which has the attribute gml:id. Therefore, UML classes with the FeatureType stereotype also have the attribute gml:id (see GML V3.2.1 E.2.4.6). This allows to transfer the attribute id of a DATEX II class with stereotype D2Identifiable. However, in GML there is no additional attribute for the DATEX II attribute version of the D2VersionedIdentifiable DATEX II. You could add an attribute to the respective UML class with stereotype FeatureType, to map the version attribute of a D2VersionedIdentifiable DATEX II class. Otherwise the version property of D2VersionedIdentifiable DATEX II class cannot be mapped to the GML application schema. Instances could concatenate the value of the two DATEX II XML attributes, at the price of losing the possibility to have unversioned references to versioned identifiables.

DATEX II classes with stereotype D2ModelRoot are also mapped to UML classes with stereotype FeatureType. However, only the definition property of the D2ModelRoot DATEX II class be transferred to the GML application schema since the other DATEX II properties of a D2ModelRoot DATEX II class are not supported in GML.

In GML a generalisation relationship is only allowed between either two FeatureType classes or two DataType classes (see GML V3.2.1 E.2.1.1.2 Classes). Since the DATEX II stereotypes D2ModelRoot, D2Identifiable and D2VersionedIdentifiable are mapped to the GML stereotype FeatureType, D2Class DATEX II classes with a superclass which has one of these stereotypes assigned must be mapped to a UML class with stereotype FeatureType.

ExternalClass DATEX II classes: See chapter A.9 External model elements.

The DATEX II classes GmlLineString, GmlLinearRing, GmlPolygon and GmlMultiPolygon were adopted from GML, therefore they can be mapped on the data model level (see Rule 4-5). Here the DATEX II class GmlLineString equals LineString in GML (see 10.4.4 LineStringType, LineString), GmlPolygon equals Polygon (see 10.5.4 PolygonType, Polygon) and GmlLinearRing equals LinearRing (see 10.5.8 LinearRingType, LinearRing). The GML class MultiPolygon is obsolete since GML V3.0 and can be replaced by the GML class MultiSurface (11.3.4.1 MultiSurfaceType, MultiSurface). If a DATEX II class has one of these GML classes as a component a D2Attribute is added to the DATEX II class. As the name of the added D2Attribute the “name” meta attribute of the D2Relation is used. If the "name" meta attribute of the D2Relation is empty, the default assumption is a name constructed from the name of the connected memberEnd UML Class, with its first character turned to lower case. The type of the added D2Attribute is the corresponding name of the GML class (LineString, LinearRing, Polygon, MultiSurface) with the prefix “gml:” added to it.

The DATEX II class PointCoordinates equals the GML class Point. Therefore, it can be mapped in the same way as described above.

Rule 4-1: Add a UML class and assign the stereotype DataType for D2Class DATEX II classes and the stereotype FeatureType for D2Identifiable, D2VersionedIdentifiable and D2ModelRoot DATEX II classes. If the DATEX II class to be mapped specialises a class which has one of the DATEX II stereotypes D2ModelRoot, D2Identifiable or D2VersionedIdentifiable assigned, assign the stereotype FeatureType to the created UML class. The DATEX II classes GmlLineString, GmlLinearRing, GmlPolygon and GmlMultiPolygon have been adopted from GML and therefore no UML class is created for these D2Class DATEX II classes. Same applies when mapping the DATEX II class PointCoordinates.

Rule 4-2: Change the name of the DATEX II class according to the rules in chapter A.8 Namespaces and use the changed name as name of the created UML class.

Rule 4-3: Transfer the definition from the definition property of the DATEX II class to the tagged value documentation of the created UML class (see **Note 5**).

Rule 4-4: Set the value of the noPropertyType tagged value to true (see **Note 6**).

Rule 4-5 (data model level: classes GmlLineString, GmlLinearRing, GmlPolygon, GmlMultiPolygon and PointCoordinates)**:** Create a new GML property in the GML class. Set the name of the created property to the “name” meta attribute of the D2Relation. If the "name" metaattribute of the D2Relation is empty, use the name of the connected memberEnd UML Class, with its first character turned to lower case. Set the type of the created GML property to the name of the corresponding GML class (LineString, LinearRing, Polygon, MultiSurface) with the added prefix “gml:”. Same applies when mapping the DATEX II class PointCoordinates. Here the GML class Point is used with the prefix “gml:”. The multiplicity of the new GML property must be set to the Multiplicity property of the non-composite end of the of the D2Relation.

Example:

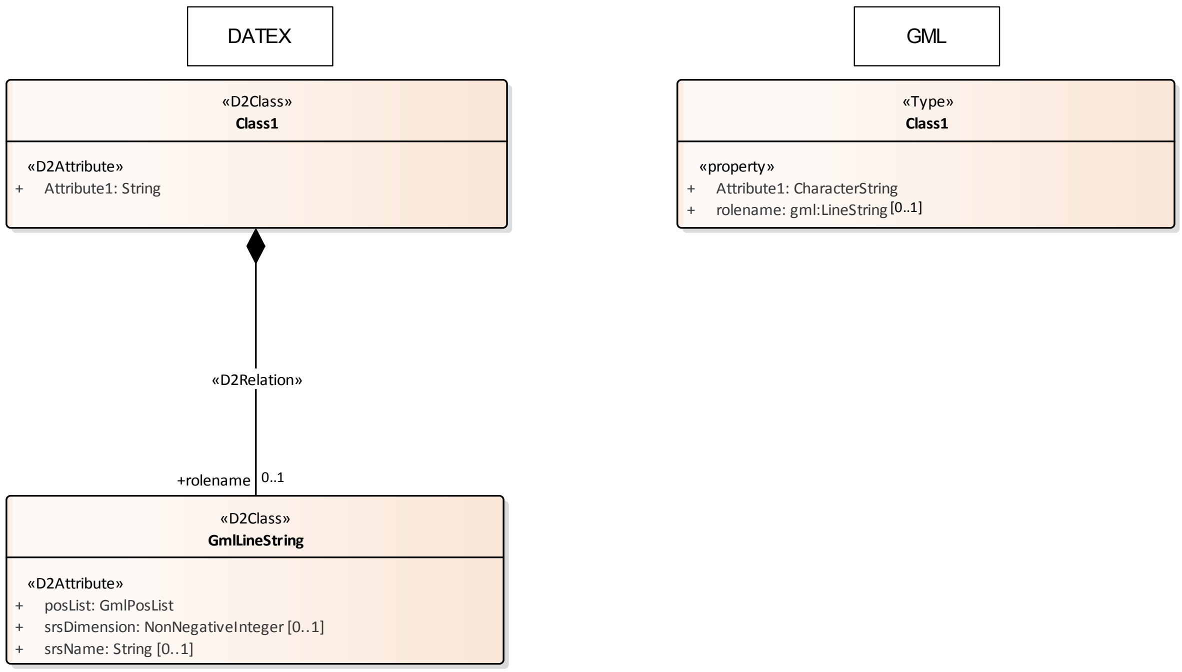


Fig. 30: Example mapping of the DATEX II class GmlLineString to GML

Note 6: If the tagged value noPropertyType is not set to true, a named complex type is created for these classes with the suffix "PropertyType" (see GML V3.2.1 E.2.4.5 and E.2.4.6). This is not required in DATEX II.

A.5 Attributes

See EN 16157-1 Section 7.3 Requirements on attributes and GML V3.2.1 E.2.1.13 Attributes.

Attributes of DATEX II classes with stereotype D2Attribute, which are called DATEX II attributes in this document, can be mapped by applying the following rules. A DATEX II attribute is equivalent to a UML attribute in GML. Note that the datatype of an DATEX II attribute must be mapped according to the rules in chapter 2 Approach.

Rule 5-1: For each DATEX II attribute, create a new GML Property in the GML class corresponding to the DATEX II class containing the DATEX II attribute. Change its name to the name of the original attribute modified by the rules in chapter A.8 Namespaces.

Rule 5-2: As type of the created GML Property, set the corresponding GML type mapped for the corresponding D2Datatype (see section 2) or D2Enumeration (see Section 3).

Rule 5-3: Transfer the definition from the definition property of the DATEX II Attribute to the tagged value documentation of the created UML Attribute (see **Note 8**). If the datatype of the DATEX II Attribute is one of the specific DATEX II datatypes add the definition of the specific DATEX II datatype to the Definition of the DATEX II Attribute to be mapped.

Example: The DATEX II Attributes “speed” of the DATEX II Class “Mobility” has the specific DATEX II Datatype “KilometresPerHour”. When mapping this DATEX II Attribute add the Definition of “KilometresPerHour” to the Definition of “speed”. The Definition of “KilometresPerHour” is “A measure of speed defined in kilometres per hour.” and the Definition of “speed” is “Speed of the mobile entity.”. The Definition of the mapped Attribute would then be “Speed of the mobile entity. (KilometresPerHour: “A measure of speed defined in kilometres per hour.”)”

Rule 5-4: Copy the value of the order property of the DATEX II attribute into the sequenceNumber property of the created GML Property.

Rule 5-5: Copy the value of the multiplicity property of the DATEX II attribute into the multiplicity property of the created GML Property.

Note 7: Enterprise Architect does not know the documentation property. Here, the definition must be copied into the "Notes" field of the UML Attribute.

A.6 Associations

See EN 16157-1 Section 7.2 Requirements on associations, and GML V3.2.1 E.2.1.1.4 Associations and association ends.

A D2Relation DATEX II Association, which is called DATEX II relation in this document, can be mapped to a UML association. For the composite end of the DATEX II relation the aggregation kind of the same end of the created UML association needs to be set to “composite” and the Navigability property needs to be set to “Unspecified”. For the other non-composite end of the created UML association the aggregation kind needs to be set to “none”, and the Navigability property needs to be set to “Navigable”. The value of the Multiplicity property of the non-composite end of the DATEX II relation must be transferred to the Multiplicity property of the navigable end of the generated UML association. In addition, a GML property must be added to the navigable end of the generated UML association whose name corresponds to the role name of the association end of the DATEX II relation. If there is no role name, the name of the created property is set to the name of the class at the navigable end of the association with the first letter turned to lower case. The value of the order property of the DATEX II relation can be copied to the sequenceNumber property of the created GML property. However, the value of the order property of the DATEX II relation cannot be transferred directly, because GML requires that the value must be unique for all attributes and association ends of a class. To ensure this, the values of the sequenceNumber property of this GML property must be greater than the values of the attributes of the source class. GML does not provide any properties for the association itself, so the definition of a DATEX II relation cannot be mapped accordingly.

Rule 6-1: Add For a new DATEX II relation create a UML association connecting. Connect the same classes, which are already mapped according to the two DATEX II classes connected by the D2Relation by the rules in section 4, as the DATEX II relation to be mapped.

Rule 6-2: Transfer the aggregation kind of both ends of the DATEX II relation to the aggregation property of both ends of the created UML association. It is important, that the “composite” end of the DATEX II relation is mapped to the same end of the created UML Association. The aggregation kind of the non-composite end is set to “none”.

Rule 6-3: In the Association properties, set the value of the Navigability property of the composite end to Unspecified and on the non-composite end to “Navigable”.

Rule 6-4: Copy the value of the Multiplicity property of the non-composite end of the D2Relation to the Multiplicity property of the non-composite end of the created UML Association.

Rule 6-5: Add a GML property to the non-composite end of the created UML Association. If the DATEX II associations’ member property on the “part” side (non-composite side) has an explicit name (role name), copy this to the name of the GML property, modified by the rules in chapter A.8Namespaces. If not, take a name constructed from the name of the class on the con-composite part of the D2Relation with its first character turned to lower case.

Rule 6-6: Copy the value of the order property of the DATEX II relation to the property sequenceNumber of the created GML property and add the number of attributes of the source class to this value.

A.7 Generalisations

See EN 16157-1 Section 7.7 Requirements on generalisations, and GML V3.2.1 E.2.1.1.2 Classes.

DATEX II generalisations without a stereotype can be adopted without any changes. Generalisation relationships have no stereotype in GML. DATEX II generalisations with stereotype D2LevelBExtension can be mapped just the same way, see chapter 4.2.

Rule 7-1: Add a generalisation relationship connecting the classes mapped to the two DATEX II classes connected by the generalisation relationship by the rules in section 4. Exception see Rule 7.2.

Rule 7-2: Do not create the generalisation in Rule 7-1, if in the GML model subclass and superclass are identical (i.e. no self-referencing generalisation).

Note 9: Above situation (Rule 7-2) can occur, if different DATEX II DataTypes (with a generalisation in between) are mapped to the same GML class.

Note 8: The Enterprise Architect feature to convert a GML model into an XML schema does not create proper results, if the subclass of a generalisation is of stereotype "DataType" (probably a bug). For this reason, the following rule was introduced. The rule changes both, the stereotypes of the subclass and of the superclass, because in GML, only generalisation between the same type of classes are allowed.

Rule 7-3: For each generalisation, determine, if the subclass in the GML model is of stereotype "DataType". If so, change the stereotype of the subclass and the superclass both into "FeatureType".

This rule was introduced due to a probable Bug in Enterprise Architect (see Note 8), not due to methodical considerations.

A.8 Namespaces

Since D2Namespace DATEX II packages cannot be mapped directly, a naming convention must ensure that there are no conflicts when mapping the D2Namespace packages. Each D2Namespace DATEX II package contains a prefix property, describing a prefix for the namespace of this package. Each element of a D2Namespace DATEX II is mapped to a new name by using the prefix of the D2Namespace DATEX II package followed by the character "-".

Example: There is a D2Package DATEX II package Classes in both the D2Namespace DATEX II packages Common and Parking. There would therefore be a conflict here. The namespaces of the two packages are "com" for the D2Namespace DATEX II package Common and "par" for the D2Namespace DATEX II package Parking. So, the D2Package Classes from the package Common would map to com-Classes and the package Classes from the package Parking would map to par-Classes.

Note that both names above would not be allowed in a DATEX II model according to the DATEX II naming convention in clause 6.3 of EN 16157-1, but they are valid values for NCName as specified in Annex E of GML V3.2.1.

Rule 8-1: Prefix every name for classes, datatypes (only the DATEX II specific datatypes, see Tab. 1: Mapping of DATEX II generic datatypes to GML Types

A.2.2Specific data types), enumerations and packages contained in a D2Namespace below the top-level package with the corresponding namespace prefix, followed by the character “-”.

A.9 External model elements

See EN 16157-1 Section 7.4 Requirements on classes, Section 7.5 Requirements on datatypes, 7.9 Requirements on packages.

There is no possibility to map external model elements from DATEX II to GML. This applies to the DATEX II stereotypes ExternalClass for classes, ExternalType for data types and ExternalNamespace for packages.

A.10 Extensions

See EN 16157-1 Section 8 Extension rules.

GML does not support an extension mechanism as DATEX II does. Nevertheless, it is possible to map DATEX II extensions to GML, so that extension elements become an ordinary part of the GML model. As a consequence, there will be no backwards compatibility in GML as it is known from DATEX II.

Thus, DATEX II Extension are mapped according to the rules denoted in this document, i.e. there are no specific rules for extensions necessary.

For further details on this topic see chapter 4.2.

A.11 Credentials

[1] CEN EN 16157-1 Intelligent transport systems – DATEX II data exchange specifications for traffic management and information – part 1: context and framework

[2] OGC 07-036: OpenGIS® Geography Markup Language (GML) Encoding Standard

Annex B – Mapping of the DATEX II model

The following figures are some exemplary views on the mapping from the DATEX II UML model to the GML UML model. For each figure of the DATEX II, version 3.0 model (in light red colour), the corresponding GML model figure (in light yellow colour) follows.

B.1 Generic data types

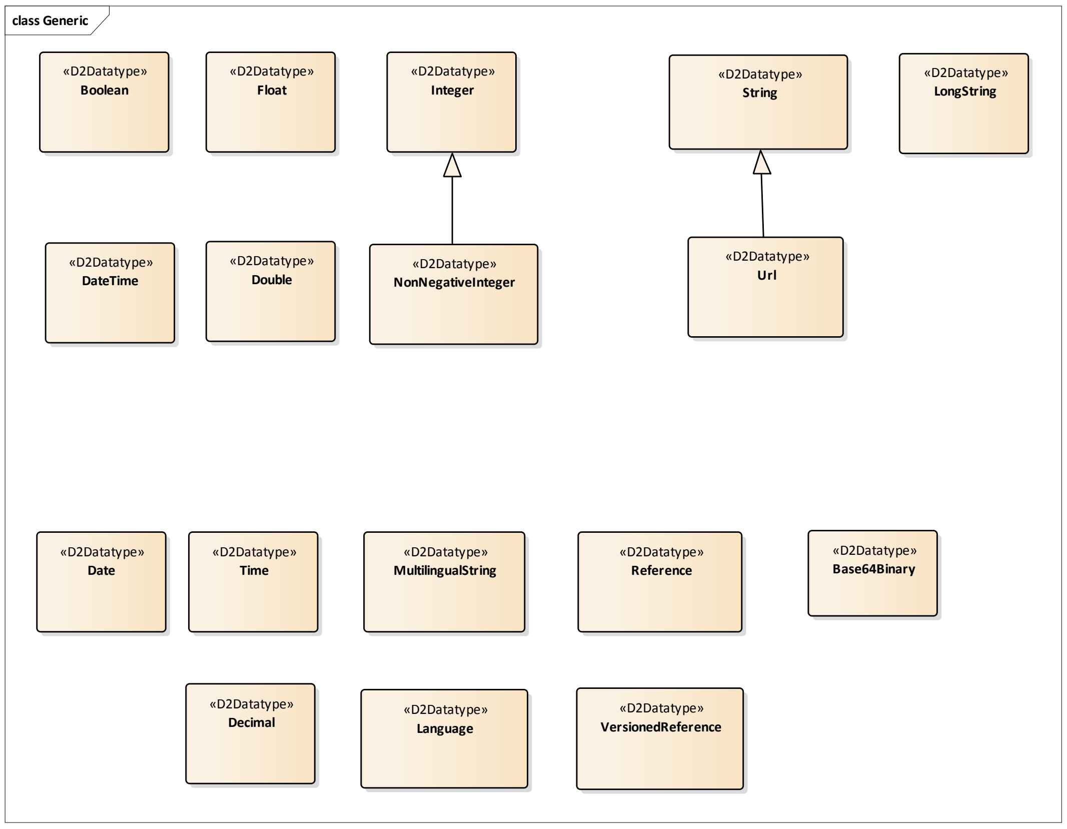


Fig. 31: DATEX II – generic data types

Note that not all Generic data types of DATEX II have an equivalent in the GML mapping, as only basic types are used there. For those that do not have a specific equivalent, one of the basic types is used instead.

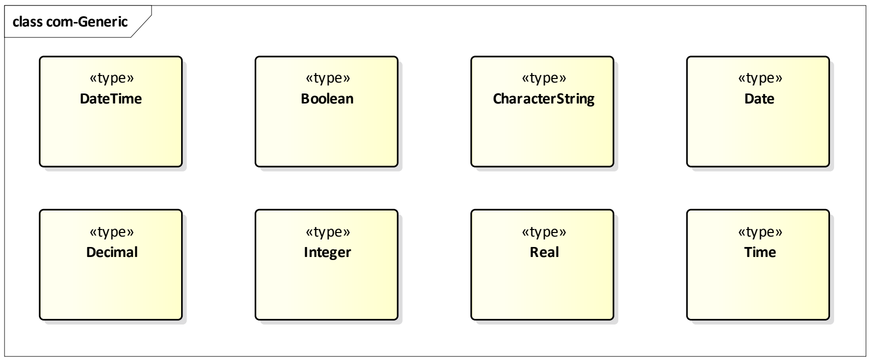


Fig. 32: GML – basic data types

B.2 Specific data types

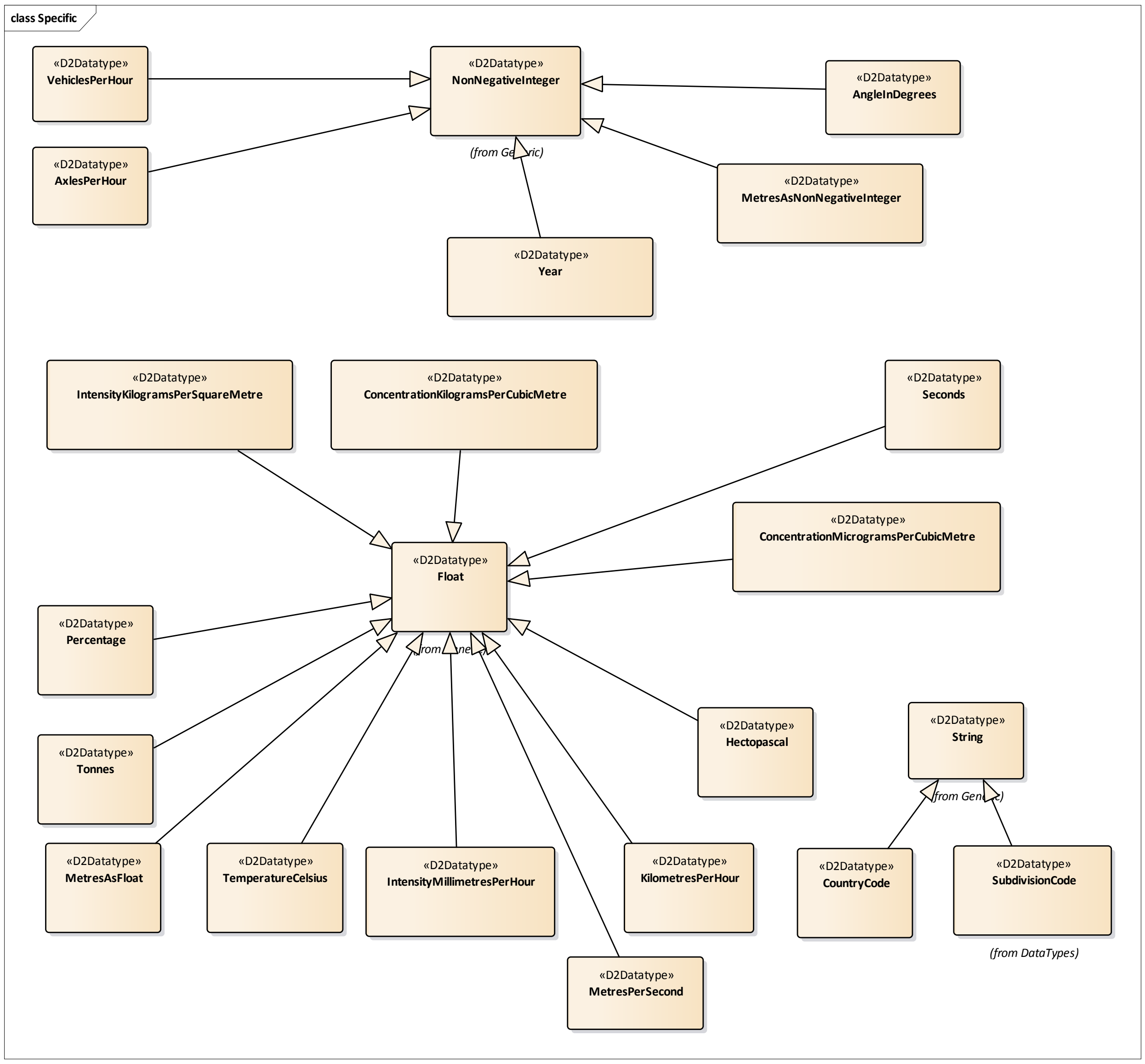
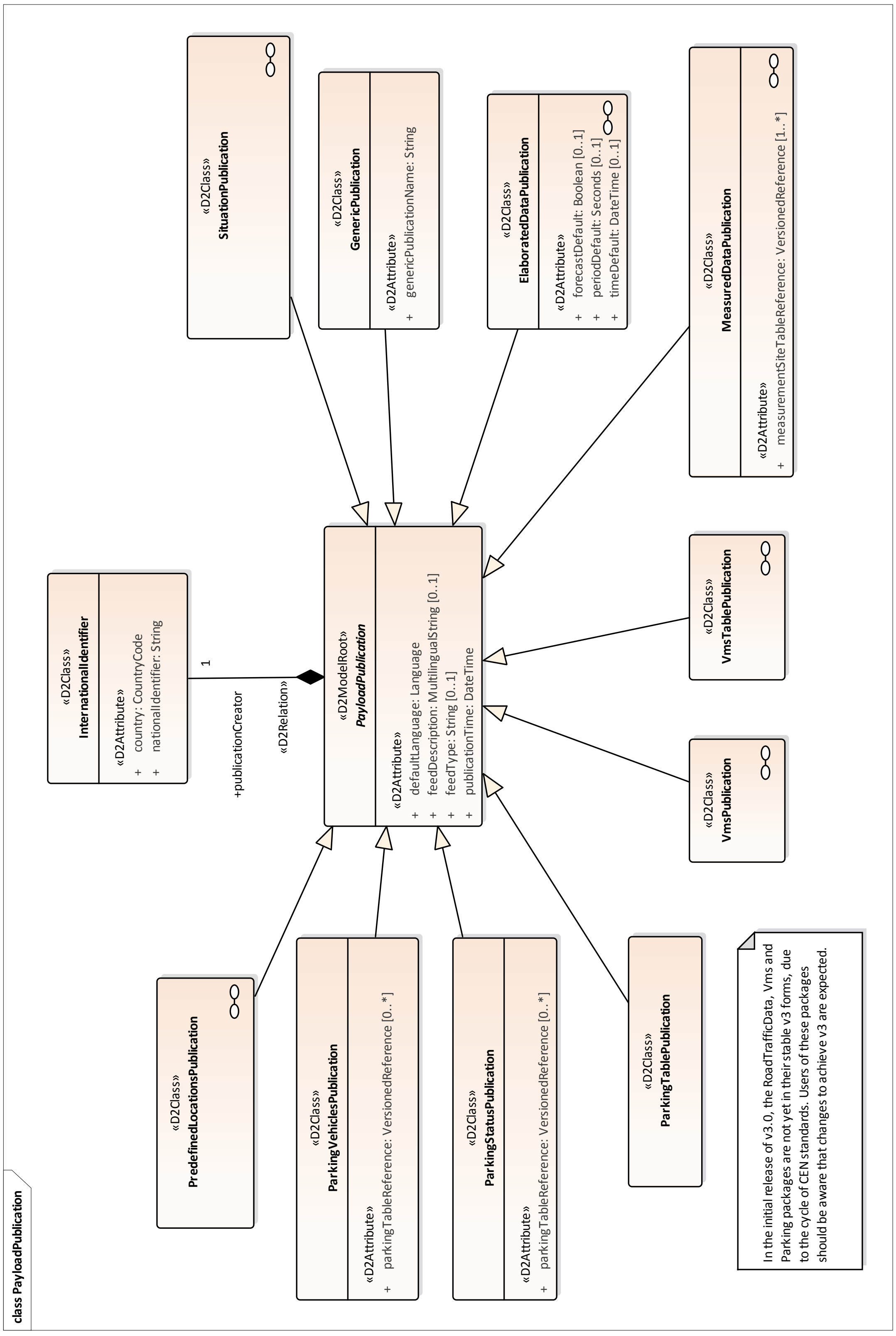

class Specific
«D2Datatype» VehiclesPerHour
«D2Datatype» NonNegativeInteger
(from Generic)
«D2Datatype» AngleInDegrees
«D2Datatype» AxlesPerHour
«D2Datatype» MetresAsNonNegativeInteger
«D2Datatype» Year
«D2Datatype» IntensityKilogramsPerSquareMetre
«D2Datatype» ConcentrationKilogramsPerCubicMetre
«D2Datatype» Seconds
«D2Datatype» ConcentrationMicrogramsPerCubicMetre
«D2Datatype» Float
(from Generic)
«D2Datatype» Percentage
«D2Datatype» Tonnes
«D2Datatype» Hectopascal
«D2Datatype» String
(from Generic)
«D2Datatype» MetresAsFloat
«D2Datatype» TemperatureCelsius
«D2Datatype» IntensityMillimetresPerHour
«D2Datatype» KilometresPerHour
«D2Datatype» CountryCode
«D2Datatype» SubdivisionCode
(from DataTypes)
«D2Datatype» MetresPerSecond

Fig. 33: DATEX II – specific data types

Note that there is no equivalent figure for the GML specific data types, as specific data types are not mapped to GML (see rules section in A.2). In GML, only the basic data types (see Fig. 32) are present. All attributes, that are of a specific data type in DATEX II are directly mapped to the corresponding basic data type in GML.

B.3 Payload publication


class PayloadPublication
«D2Class» SituationPublication
«D2Class» GenericPublication
«D2Attribute» + genericPublicationName: String
«D2Class» ElaboratedDataPublication
«D2Attribute» + forecastDefault: Boolean [0..1] + periodDefault: Seconds [0..1] + timeDefault: DateTime [0..1]
«D2Class» MeasuredDataPublication
«D2Attribute» + measurementSiteTableReference: VersionedReference [1..*]
«D2Class» InternationalIdentifier
«D2Attribute» + country: CountryCode + nationalIdentifier: String
1
+publicationCreator
«D2Relation»
«D2ModelRoot» PayloadPublication
«D2Attribute» + defaultLanguage: Language + feedDescription: MultilingualString [0..1] + feedType: String [0..1] + publicationTime: DateTime
«D2Class» VmsTablePublication
«D2Class» VmsPublication
«D2Class» PredefinedLocationsPublication
«D2Class» ParkingVehiclesPublication
«D2Attribute» + parkingTableReference: VersionedReference [0..*]
«D2Class» ParkingStatusPublication
«D2Attribute» + parkingTableReference: VersionedReference [0..*]
«D2Class» ParkingTablePublication

In the initial release of v3.0, the RoadTrafficData, Vms and Parking packages are not yet in their stable v3 forms, due to the cycle of CEN standards. Users of these packages should be aware that changes to achieve v3 are expected.

Fig. 34: DATEX II – payload publication

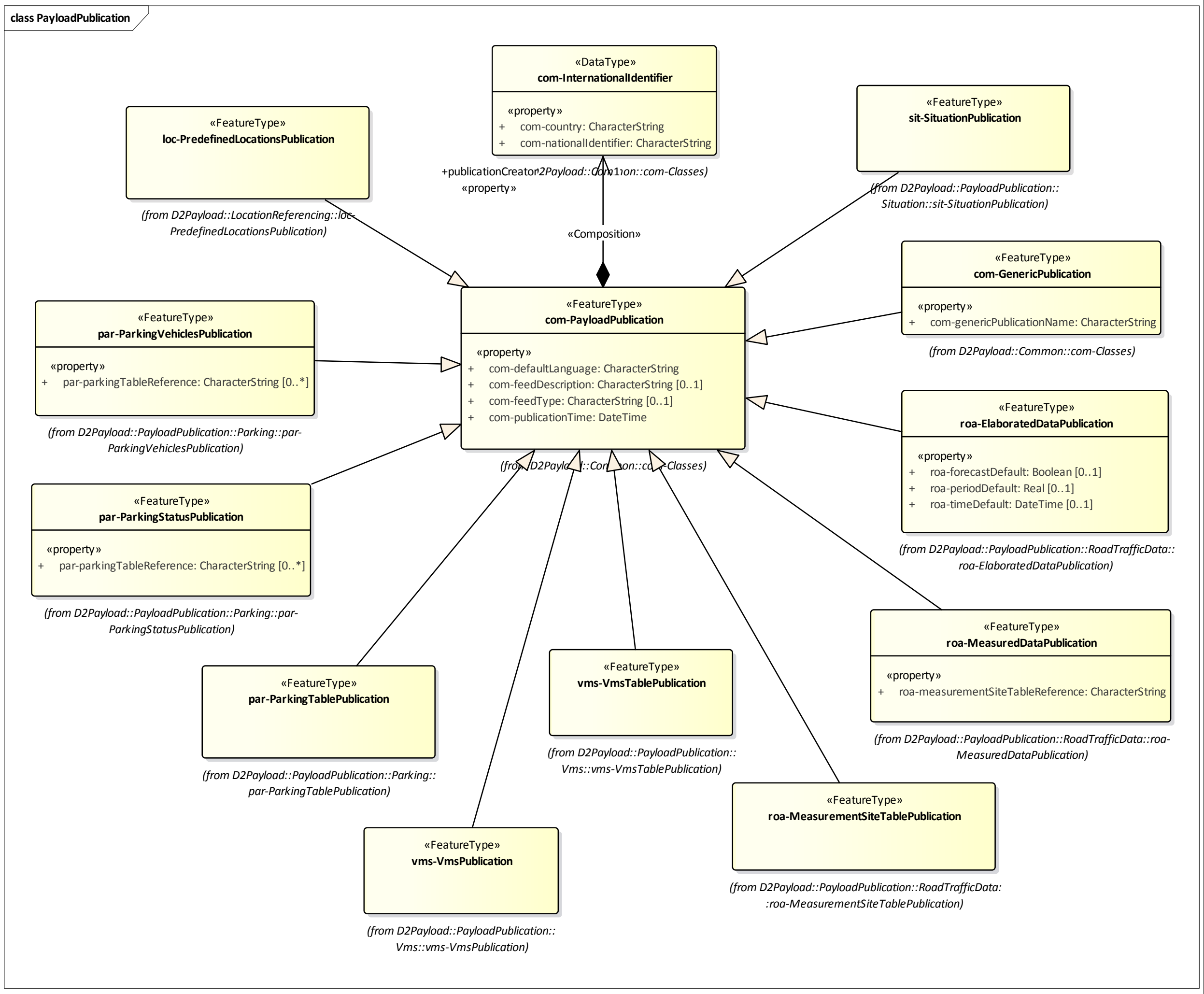

class PayloadPublication
«DataType»
com-InternationalIdentifier
«property»
+ com-country: CharacterString
+ com-nationalIdentifier: CharacterString
+publicationCreator
«property»
(from D2Payload::Common::com-Classes)
1
«FeatureType»
loc-PredefinedLocationsPublication
(from D2Payload::LocationReferencing::loc-PredefinedLocationsPublication)
«FeatureType»
sit-SituationPublication
(from D2Payload::PayloadPublication::Situation::sit-SituationPublication)
«Composition»
«FeatureType»
com-GenericPublication
«property»
+ com-genericPublicationName: CharacterString
(from D2Payload::Common::com-Classes)
«FeatureType»
par-ParkingVehiclesPublication
«property»
+ par-parkingTableReference: CharacterString [0..*]
(from D2Payload::PayloadPublication::Parking::par-ParkingVehiclesPublication)
«FeatureType»
com-PayloadPublication
«property»
+ com-defaultLanguage: CharacterString
+ com-feedDescription: CharacterString [0..1]
+ com-feedType: CharacterString [0..1]
+ com-publicationTime: DateTime
(from D2Payload::Common::com-Classes)
«FeatureType»
roa-ElaboratedDataPublication
«property»
+ roa-forecastDefault: Boolean [0..1]
+ roa-periodDefault: Real [0..1]
+ roa-timeDefault: DateTime [0..1]
(from D2Payload::PayloadPublication::RoadTrafficData::roa-ElaboratedDataPublication)
«FeatureType»
par-ParkingStatusPublication
«property»
+ par-parkingTableReference: CharacterString [0..*]
(from D2Payload::PayloadPublication::Parking::par-ParkingStatusPublication)
«FeatureType»
roa-MeasuredDataPublication
«property»
+ roa-measurementSiteTableReference: CharacterString
(from D2Payload::PayloadPublication::RoadTrafficData::roa-MeasuredDataPublication)
«FeatureType»
par-ParkingTablePublication
(from D2Payload::PayloadPublication::Parking::par-ParkingTablePublication)
«FeatureType»
vms-VmsTablePublication
(from D2Payload::PayloadPublication::Vms::vms-VmsTablePublication)
«FeatureType»
roa-MeasurementSiteTablePublication
(from D2Payload::PayloadPublication::RoadTrafficData::roa-MeasurementSiteTablePublication)
«FeatureType»
vms-VmsPublication
(from D2Payload::PayloadPublication::Vms::vms-VmsPublication)

Fig. 35: GML – payload publication

B.4 Location reference

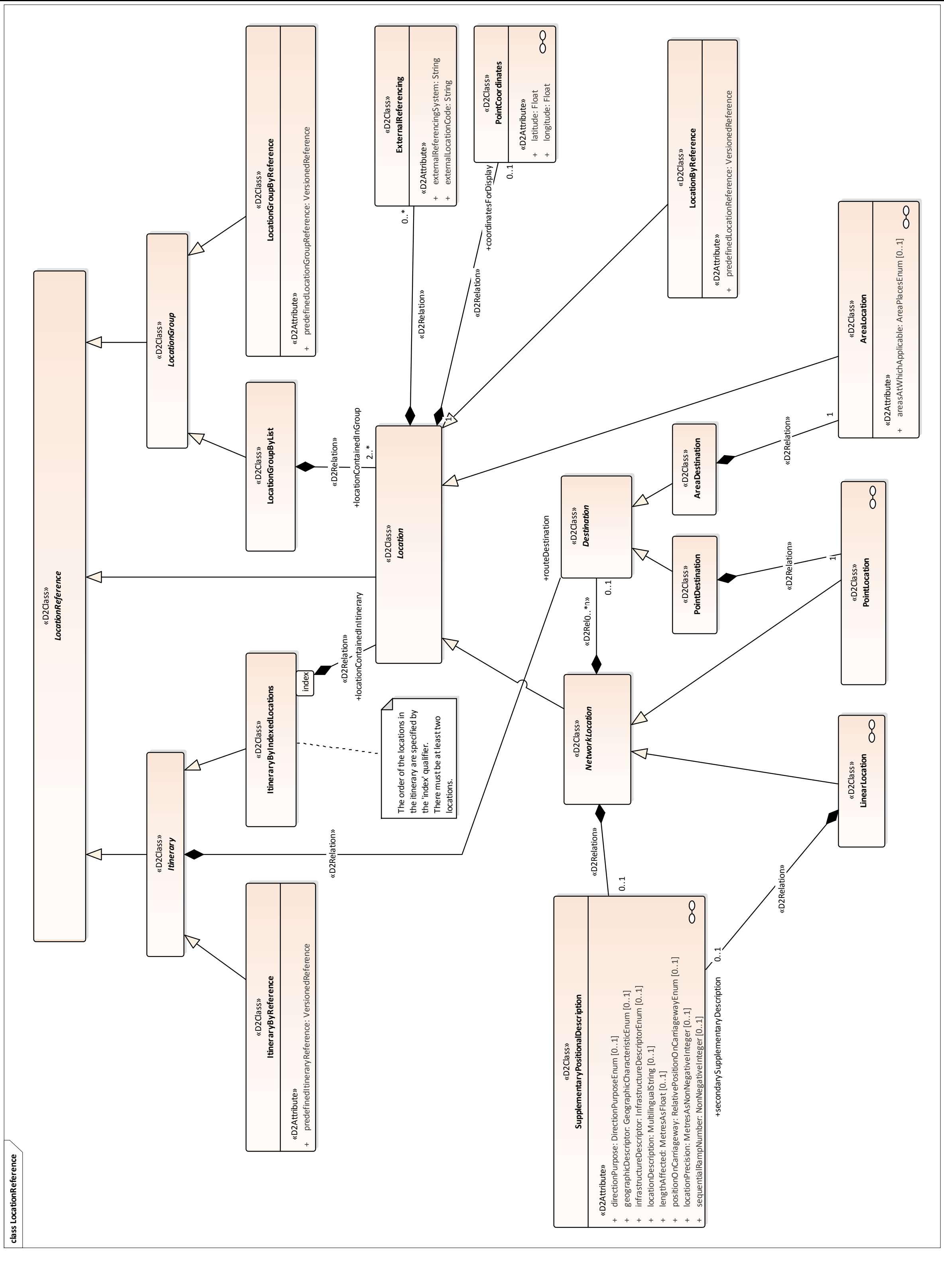

class LocationReference
«D2Class» LocationReference
«D2Class» LocationGroup
«D2Class» LocationGroupByReference
«D2Attribute» + predefinedLocationGroupReference: VersionedReference
«D2Class» LocationGroupByList
«D2Relation»
+locationContainedInGroup
2..*
«D2Class» ExternalReferencing
«D2Attribute» + externalReferencingSystem: String + externalLocationCode: String
0..*
«D2Relation»
«D2Class» PointCoordinates
«D2Attribute» + latitude: Float + longitude: Float
0..1
+coordinatesForDisplay
«D2Relation»
«D2Class» LocationByReference
«D2Attribute» + predefinedLocationReference: VersionedReference
1
«D2Class» Location
«D2Class» AreaLocation
«D2Attribute» + areasAtWhichApplicable: AreaPlacesEnum [0..1]
1
«D2Relation»
«D2Class» AreaDestination
«D2Class» Destination
+routeDestination
«D2Class» PointDestination
«D2Relation»
1
«D2Class» PointLocation
0..1
«D2Rel0..*n»
«D2Class» Itinerary
«D2Class» ItineraryByIndexedLocations
index
«D2Relation»
+locationContainedInItinerary
The order of the locations in the itinerary are specified by the 'index' qualifier. There must be at least two locations.
«D2Relation»
«D2Class» ItineraryByReference
«D2Attribute» + predefinedItineraryReference: VersionedReference
«D2Class» NetworkLocation
«D2Class» LinearLocation
«D2Relation»
0..1
«D2Class» SupplementaryPositionalDescription
«D2Attribute»
+ directionPurpose: DirectionPurposeEnum [0..1]
+ geographicDescriptor: GeographicCharacteristicEnum [0..1]
+ infrastructureDescriptor: InfrastructureDescriptorEnum [0..1]
+ locationDescription: MultilingualString [0..1]
+ lengthAffected: MetresAsFloat [0..1]
+ positionOnCarriageway: RelativePositionOnCarriagewayEnum [0..1]
+ locationPrecision: MetresAsNonNegativeInteger [0..1]
+ sequentialRampNumber: NonNegativeInteger [0..1]
0..1
+secondarySupplementaryDescription
«D2Relation»

Fig. 36: DATEX II – location reference

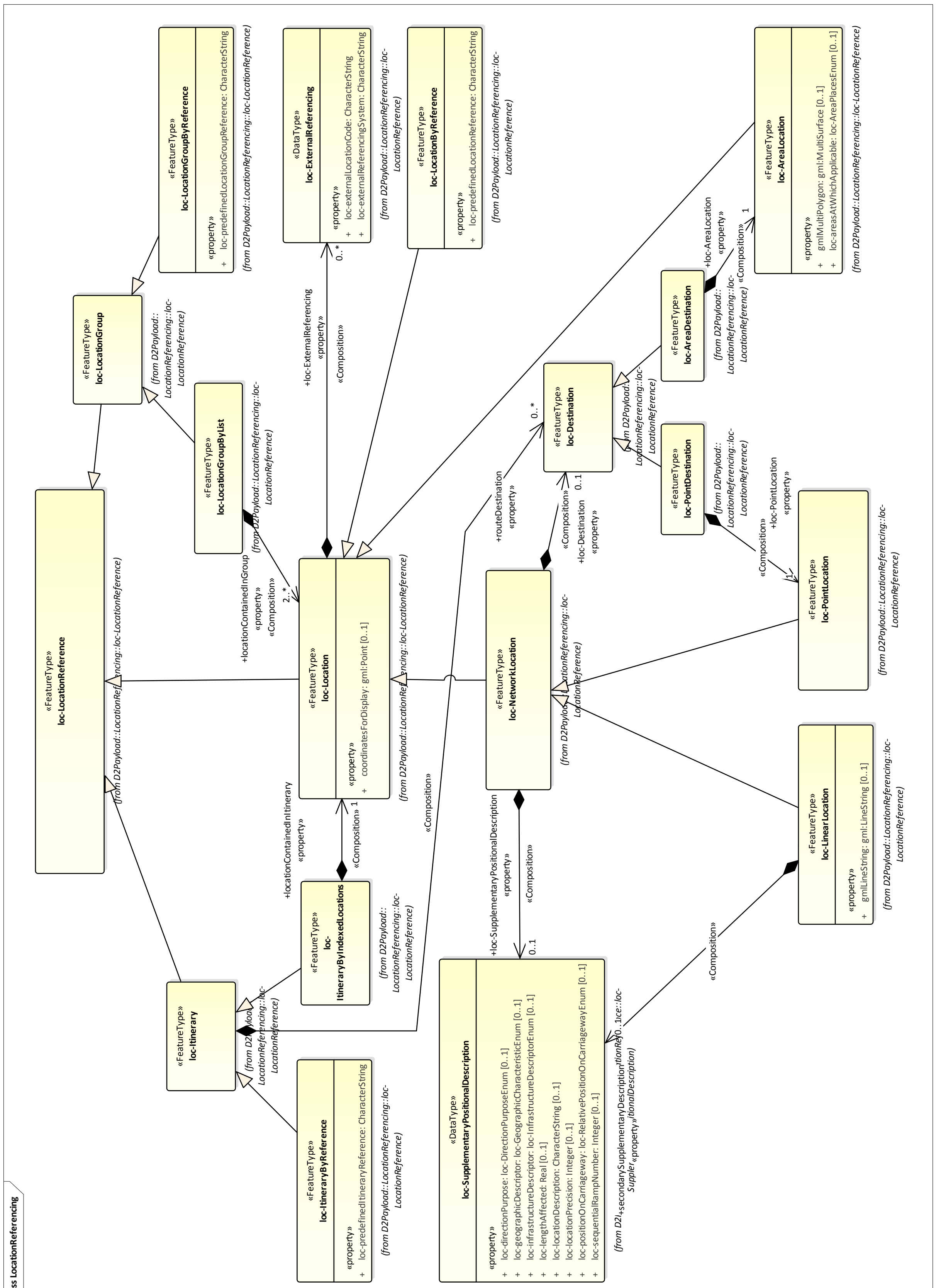

class LocationReferencing
«FeatureType» loc-LocationGroupByReference
«property» + loc-predefinedLocationGroupReference: CharacterString
(from D2Payload::LocationReferencing::loc-LocationReference)
«DataType» loc-ExternalReferencing
«property» + loc-externalLocationCode: CharacterString + loc-externalReferencingSystem: CharacterString
«FeatureType» loc-LocationByReference
«property» + loc-predefinedLocationReference: CharacterString
«FeatureType» loc-AreaLocation
«property» + gml:MultiPolygon: gml:MultiSurface [0..1] + loc-areasAtWhichApplicable: loc-AreaPlacesEnum [0..1]
«FeatureType» loc-LocationGroup
«FeatureType» loc-LocationGroupByList
«FeatureType» loc-AreaDestination
«FeatureType» loc-Destination
«FeatureType» loc-PointDestination
«FeatureType» loc-PointLocation
«FeatureType» loc-LocationReference
«FeatureType» loc-Location
«property» + coordinatesForDisplay: gml:Point [0..1]
«FeatureType» loc-NetworkLocation
«FeatureType» loc-LinearLocation
«property» + gmlLineString: gml:LineString [0..1]
«FeatureType» loc-ItineraryByIndexedLocations
«FeatureType» loc-Itinerary
«FeatureType» loc-ItineraryByReference
«property» + loc-predefinedItineraryReference: CharacterString
«DataType» loc-SupplementaryPositionalDescription
«property» + loc-directionPurpose: loc-DirectionPurposeEnum [0..1] + loc-geographicDescriptor: loc-GeographicCharacteristicEnum [0..1] + loc-infrastructureDescriptor: loc-InfrastructureDescriptorEnum [0..1] + loc-lengthAffected: Real [0..1] + loc-locationDescription: CharacterString [0..1] + loc-locationPrecision: Integer [0..1] + loc-positionOnCarriageway: loc-RelativePositionOnCarriagewayEnum [0..1] + loc-sequentialRampNumber: Integer [0..1]
+loc-ExternalReferencing «property» «Composition» 0..*
+locationContainedInGroup «property» «Composition» 2..*
+routeDestination «property» 0..*
«Composition» +loc-Destination «property» 0..1
+loc-AreaLocation «property» «Composition» 1
«Composition» +loc-PointLocation «property» 1
+locationContainedInItinerary «property» «Composition» 1
«Composition»
+loc-SupplementaryPositionalDescription «property» «Composition» 0..1
+secondarySupplementaryDescription «property» 0..1
(from D2Payload::LocationReferencing::loc-LocationReference)

Fig. 37: GML – location reference

B.5 Linear location

Note that by design the DATEX II class "GMLLineString" is replaced by an attribute of type "gml:LineString" in the class LinearLocation.

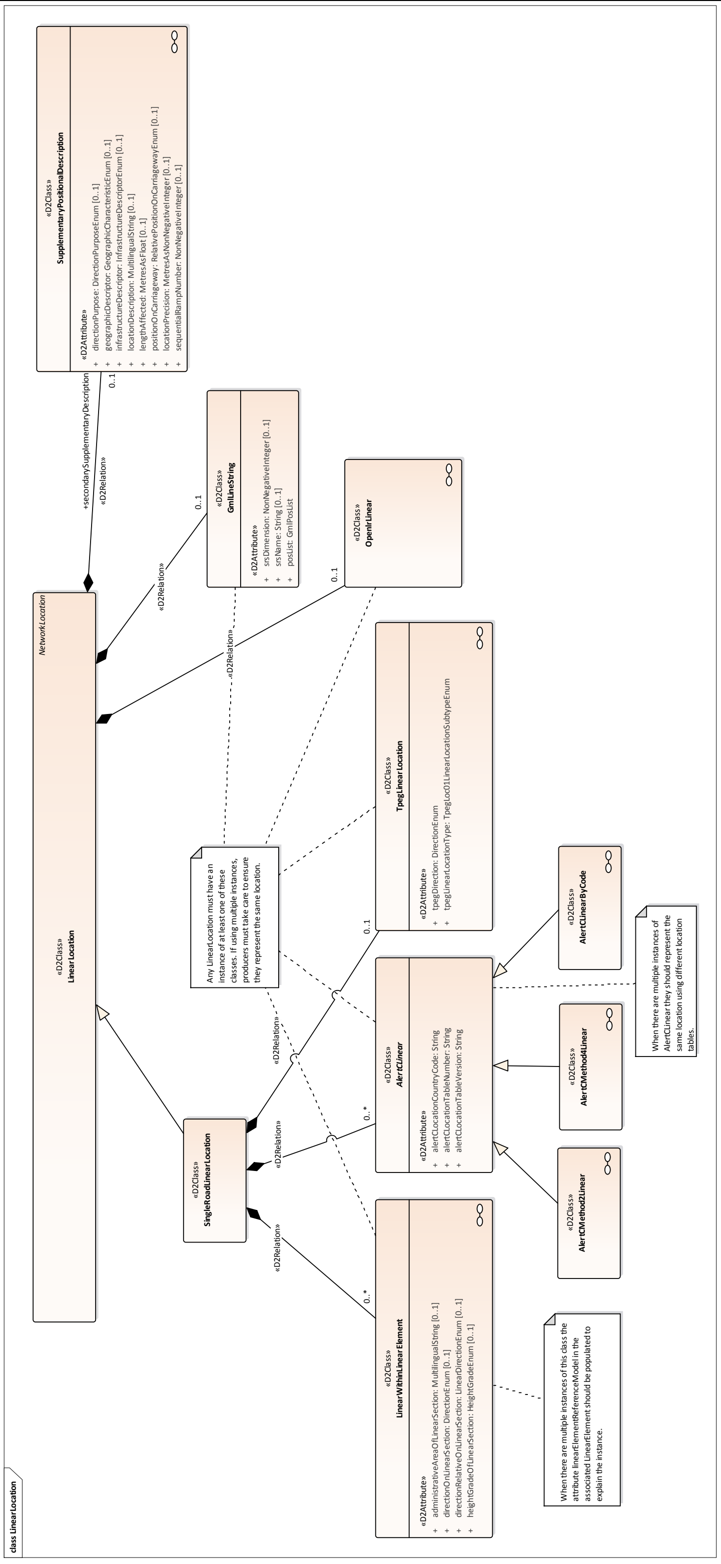

class LinearLocation
«D2Class» SupplementaryPositionalDescription
«D2Attribute»
+ directionPurpose: DirectionPurposeEnum [0..1]
+ geographicDescriptor: GeographicCharacteristicEnum [0..1]
+ infrastructureDescriptor: InfrastructureDescriptorEnum [0..1]
+ locationDescription: MultilingualString [0..1]
+ lengthAffected: MetresAsFloat [0..1]
+ positionOnCarriageway: RelativePositionOnCarriagewayEnum [0..1]
+ locationPrecision: MetresAsNonNegativeInteger [0..1]
+ sequentialRampNumber: NonNegativeInteger [0..1]
«secondarySupplementaryDescription
«D2Relation»
0..1
«D2Class» GmlLineString
«D2Attribute»
+ srsDimension: NonNegativeInteger [0..1]
+ srsName: String [0..1]
+ posList: GmlPosList
«D2Relation»
0..1
«D2Class» OpenlrLinear
0..1
NetworkLocation
«D2Class» LinearLocation
«D2Class» TpegLinearLocation
«D2Attribute»
+ tpegDirection: DirectionEnum
+ tpegLinearLocationType: TpegLoc01LinearLocationSubtypeEnum
Any LinearLocation must have an instance of at least one of these classes. If using multiple instances, producers must take care to ensure they represent the same location.
«D2Relation»
0..1
«D2Class» AlertCLinear
«D2Attribute»
+ alertCLocationCountryCode: String
+ alertCLocationTableNumber: String
+ alertCLocationTableVersion: String
«D2Class» AlertCLinearByCode
«D2Class» AlertCMethod4Linear
«D2Class» AlertCMethod2Linear
When there are multiple instances of AlertCLinear they should represent the same location using different location tables.
«D2Class» SingleRoadLinearLocation
«D2Relation»
0..*
«D2Relation»
0..*
«D2Class» LinearWithinLinearElement
«D2Attribute»
+ administrativeAreaOfLinearSection: MultilingualString [0..1]
+ directionOnLinearSection: DirectionEnum [0..1]
+ directionRelativeOnLinearSection: LinearDirectionEnum [0..1]
+ heightGradeOfLinearSection: HeightGradeEnum [0..1]
When there are multiple instances of this class the attribute linearElementReferenceModel in the associated LinearElement should be populated to explain the instance.

Fig. 38: DATEX II – linear location

Note that in the GML version, the DATEX II class "GmlLineString" has been mapped into an attribute of the same name (in class loc-LinearLocation):

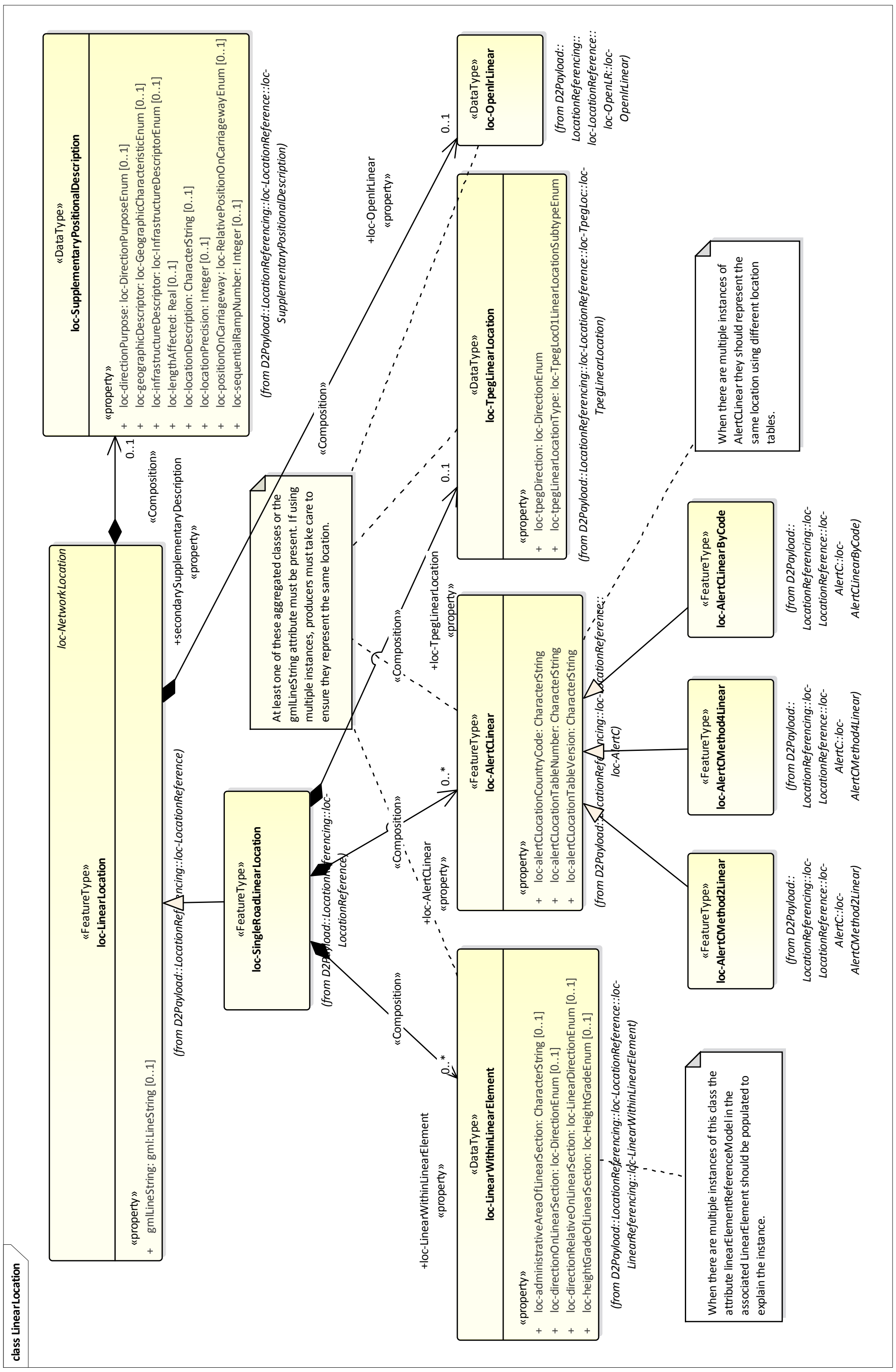

class LinearLocation
loc-NetworkLocation
«FeatureType»
loc-LinearLocation
«property»
+ gmlLineString: gml:LineString [0..1]
(from D2Payload::LocationReferencing::loc-LocationReference)
«DataType»
loc-SupplementaryPositionalDescription
«property»
+ loc-directionPurpose: loc-DirectionPurposeEnum [0..1]
+ loc-geographicDescriptor: loc-GeographicCharacteristicEnum [0..1]
+ loc-infrastructureDescriptor: loc-InfrastructureDescriptorEnum [0..1]
+ loc-lengthAffected: Real [0..1]
+ loc-locationDescription: CharacterString [0..1]
+ loc-locationPrecision: Integer [0..1]
+ loc-positionOnCarriageway: loc-RelativePositionOnCarriagewayEnum [0..1]
+ loc-sequentialRampNumber: Integer [0..1]
(from D2Payload::LocationReferencing::loc-LocationReference::loc-SupplementaryPositionalDescription)
+secondarySupplementaryDescription
«property»
«Composition»
0..1
«FeatureType»
loc-SingleRoadLinearLocation
(from D2Payload::LocationReferencing::loc-LocationReference)
At least one of these aggregated classes or the gmlLineString attribute must be present. If using multiple instances, producers must take care to ensure they represent the same location.
«Composition»
+loc-OpenlrLinear
«property»
0..1
«DataType»
loc-OpenlrLinear
(from D2Payload::LocationReferencing::loc-LocationReference::loc-OpenLR::loc-OpenlrLinear)
«Composition»
+loc-TpegLinearLocation
«property»
0..1
«DataType»
loc-TpegLinearLocation
«property»
+ loc-tpegDirection: loc-DirectionEnum
+ loc-tpegLinearLocationType: loc-TpegLoc01LinearLocationSubtypeEnum
(from D2Payload::LocationReferencing::loc-LocationReference::loc-TpegLoc::loc-TpegLinearLocation)
«Composition»
+loc-AlertCLinear
«property»
0..*
«FeatureType»
loc-AlertCLinear
«property»
+ loc-alertCLocationCountryCode: CharacterString
+ loc-alertCLocationTableNumber: CharacterString
+ loc-alertCLocationTableVersion: CharacterString
(from D2Payload::LocationReferencing::loc-LocationReference::loc-AlertC)
When there are multiple instances of AlertCLinear they should represent the same location using different location tables.
«FeatureType»
loc-AlertCLinearByCode
(from D2Payload::LocationReferencing::loc-LocationReference::loc-AlertC::loc-AlertCLinearByCode)
«FeatureType»
loc-AlertCMethod4Linear
(from D2Payload::LocationReferencing::loc-LocationReference::loc-AlertC::loc-AlertCMethod4Linear)
«FeatureType»
loc-AlertCMethod2Linear
(from D2Payload::LocationReferencing::loc-LocationReference::loc-AlertC::loc-AlertCMethod2Linear)
«Composition»
+loc-LinearWithinLinearElement
«property»
0..*
«DataType»
loc-LinearWithinLinearElement
«property»
+ loc-administrativeAreaOfLinearSection: CharacterString [0..1]
+ loc-directionOnLinearSection: loc-DirectionEnum [0..1]
+ loc-directionRelativeOnLinearSection: loc-LinearDirectionEnum [0..1]
+ loc-heightGradeOfLinearSection: loc-HeightGradeEnum [0..1]
(from D2Payload::LocationReferencing::loc-LocationReference::loc-LinearReferencing::loc-LinearWithinLinearElement)
When there are multiple instances of this class the attribute linearElementReferenceModel in the associated LinearElement should be populated to explain the instance.

Fig. 39: GML – linear location

B.6 Area location

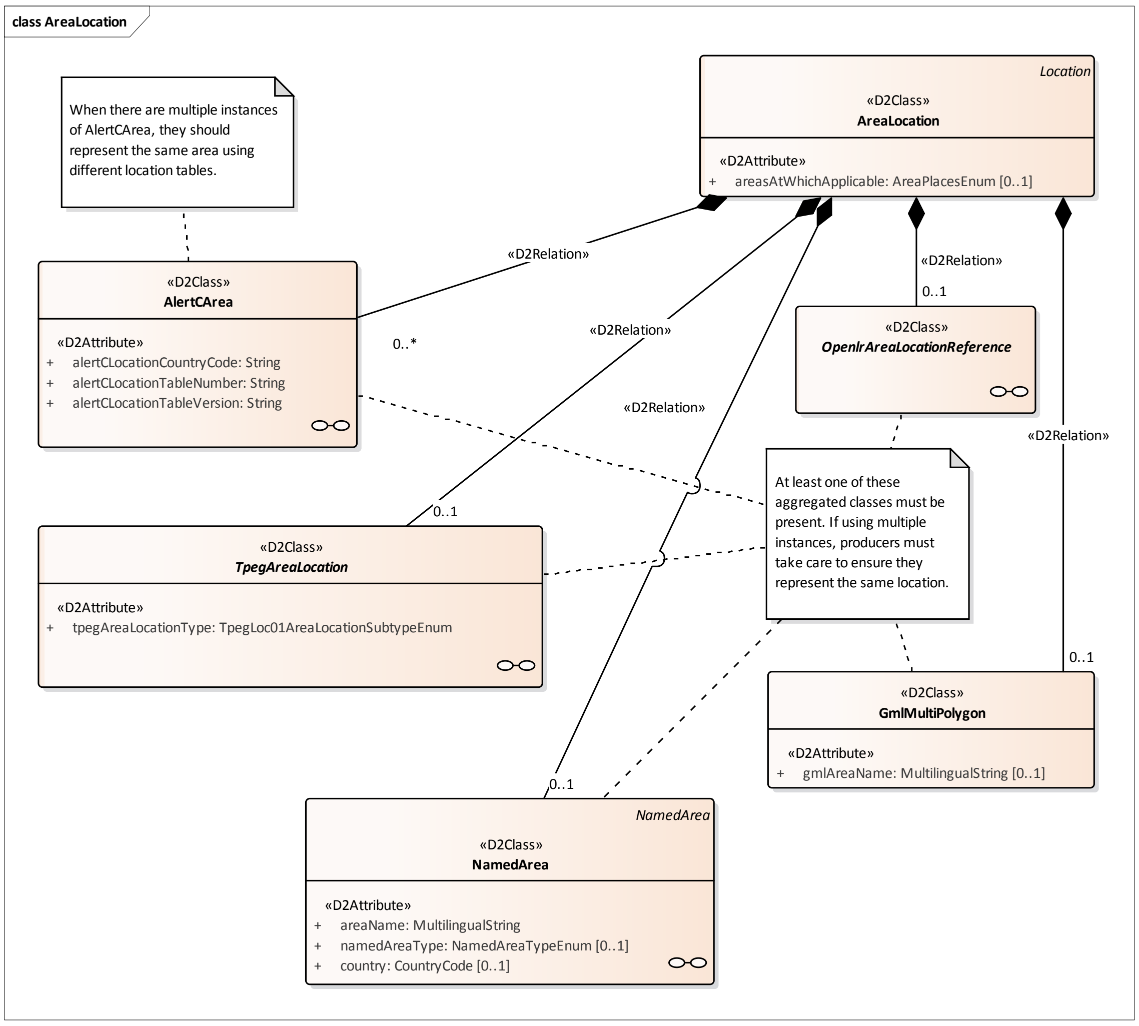

class AreaLocation
When there are multiple instances of AlertCArea, they should represent the same area using different location tables.
Location
«D2Class»
AreaLocation
«D2Attribute»
+ areasAtWhichApplicable: AreaPlacesEnum [0..1]
«D2Relation»
«D2Class»
AlertCArea
«D2Attribute»
+ alertCLocationCountryCode: String
+ alertCLocationTableNumber: String
+ alertCLocationTableVersion: String
0..*
«D2Relation»
«D2Relation»
«D2Relation»
0..1
«D2Class»
OpenlrAreaLocationReference
«D2Relation»
At least one of these aggregated classes must be present. If using multiple instances, producers must take care to ensure they represent the same location.
0..1
«D2Class»
TpegAreaLocation
«D2Attribute»
+ tpegAreaLocationType: TpegLoc01AreaLocationSubtypeEnum
0..1
«D2Class»
GmlMultiPolygon
«D2Attribute»
+ gmlAreaName: MultilingualString [0..1]
0..1
NamedArea
«D2Class»
NamedArea
«D2Attribute»
+ areaName: MultilingualString
+ namedAreaType: NamedAreaTypeEnum [0..1]
+ country: CountryCode [0..1]

Fig. 40: DATEX II – area location

Note that in the GML version, the DATEX II class "GmlMultiPolygon" has been mapped into an attribute of the same name (in class loc-AreaLocation):

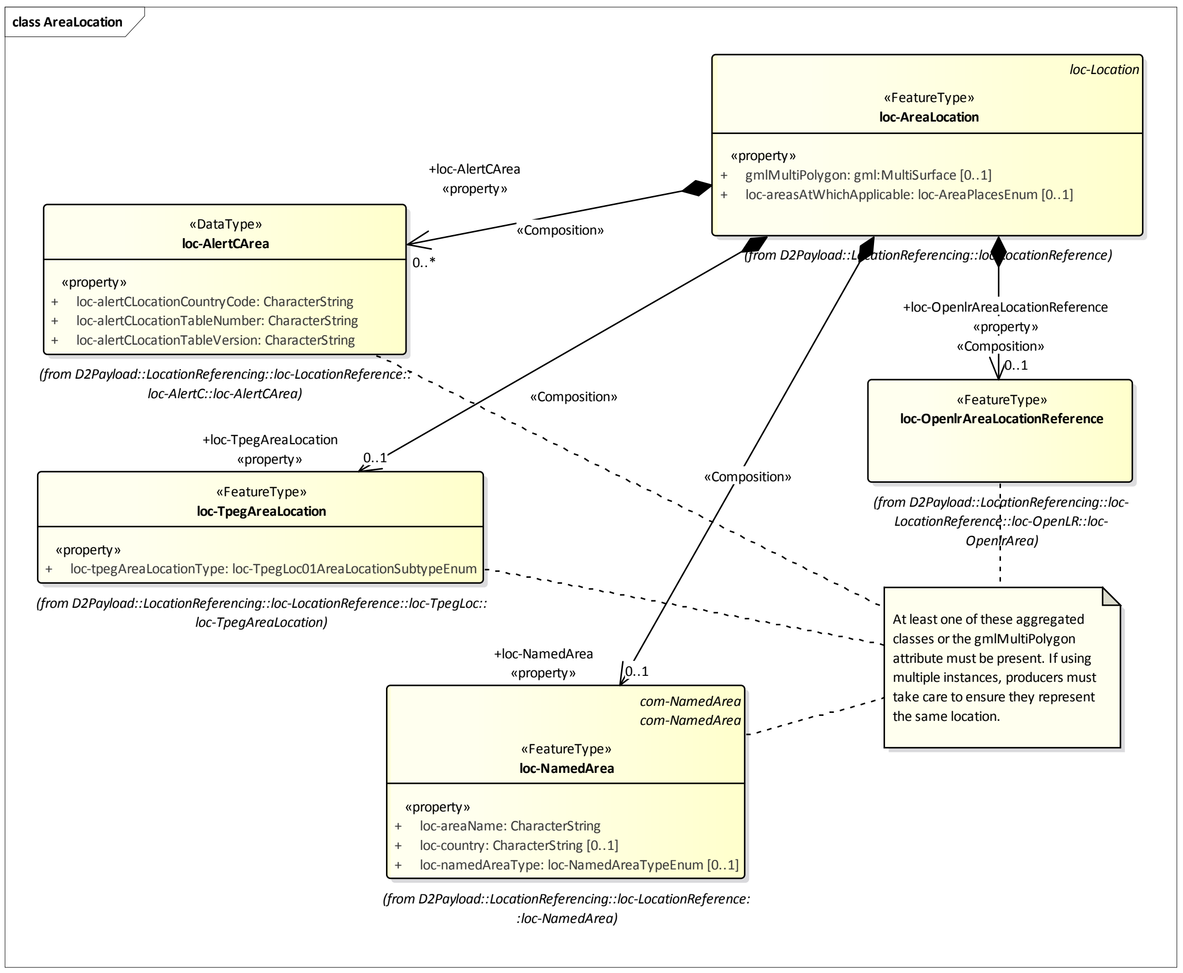

class AreaLocation
loc-Location
«FeatureType»
loc-AreaLocation
«property»
+ gmlMultiPolygon: gml:MultiSurface [0..1]
+ loc-areasAtWhichApplicable: loc-AreaPlacesEnum [0..1]
(from D2Payload::LocationReferencing::loc-LocationReference)
+loc-AlertCArea
«property»
«Composition»
0..*
«DataType»
loc-AlertCArea
«property»
+ loc-alertCLocationCountryCode: CharacterString
+ loc-alertCLocationTableNumber: CharacterString
+ loc-alertCLocationTableVersion: CharacterString
(from D2Payload::LocationReferencing::loc-LocationReference::loc-AlertC::loc-AlertCArea)
+loc-OpenlrAreaLocationReference
«property»
«Composition»
0..1
«FeatureType»
loc-OpenlrAreaLocationReference
(from D2Payload::LocationReferencing::loc-LocationReference::loc-OpenLR::loc-OpenlrArea)
«Composition»
+loc-TpegAreaLocation
«property»
0..1
«FeatureType»
loc-TpegAreaLocation
«property»
+ loc-tpegAreaLocationType: loc-TpegLoc01AreaLocationSubtypeEnum
(from D2Payload::LocationReferencing::loc-LocationReference::loc-TpegLoc::loc-TpegAreaLocation)
«Composition»
At least one of these aggregated classes or the gmlMultiPolygon attribute must be present. If using multiple instances, producers must take care to ensure they represent the same location.
+loc-NamedArea
«property»
0..1
com-NamedArea
com-NamedArea
«FeatureType»
loc-NamedArea
«property»
+ loc-areaName: CharacterString
+ loc-country: CharacterString [0..1]
+ loc-namedAreaType: loc-NamedAreaTypeEnum [0..1]
(from D2Payload::LocationReferencing::loc-LocationReference::loc-NamedArea)

Fig. 41: GML – area location

B.7 Situation record

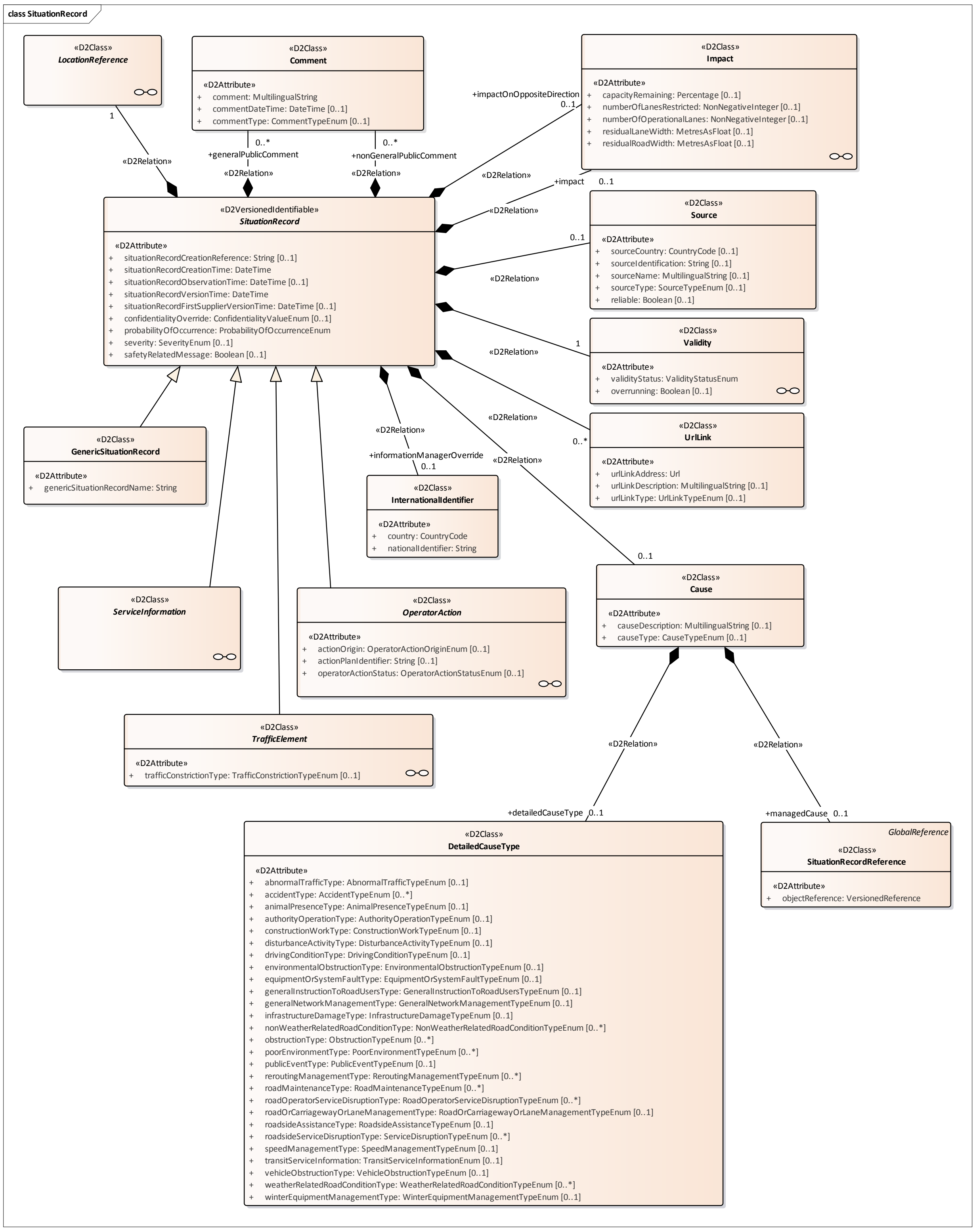

class SituationRecord
«D2Class»
LocationReference
1
«D2Relation»
«D2Class»
Comment
«D2Attribute»
+ comment: MultilingualString
+ commentDateTime: DateTime [0..1]
+ commentType: CommentTypeEnum [0..1]
0..*
+generalPublicComment
«D2Relation»
0..*
+nonGeneralPublicComment
«D2Relation»
«D2Class»
Impact
«D2Attribute»
+ capacityRemaining: Percentage [0..1]
+ numberOfLanesRestricted: NonNegativeInteger [0..1]
+ numberOfOperationalLanes: NonNegativeInteger [0..1]
+ residualLaneWidth: MetresAsFloat [0..1]
+ residualRoadWidth: MetresAsFloat [0..1]
+impactOnOppositeDirection
0..1
«D2Relation»
+impact
0..1
«D2Relation»
«D2VersionedIdentifiable»
SituationRecord
«D2Attribute»
+ situationRecordCreationReference: String [0..1]
+ situationRecordCreationTime: DateTime
+ situationRecordObservationTime: DateTime [0..1]
+ situationRecordVersionTime: DateTime
+ situationRecordFirstSupplierVersionTime: DateTime [0..1]
+ confidentialityOverride: ConfidentialityValueEnum [0..1]
+ probabilityOfOccurrence: ProbabilityOfOccurrenceEnum
+ severity: SeverityEnum [0..1]
+ safetyRelatedMessage: Boolean [0..1]
«D2Class»
Source
«D2Attribute»
+ sourceCountry: CountryCode [0..1]
+ sourceIdentification: String [0..1]
+ sourceName: MultilingualString [0..1]
+ sourceType: SourceTypeEnum [0..1]
+ reliable: Boolean [0..1]
0..1
«D2Relation»
«D2Class»
Validity
«D2Attribute»
+ validityStatus: ValidityStatusEnum
+ overrunning: Boolean [0..1]
1
«D2Relation»
«D2Class»
UrlLink
«D2Attribute»
+ urlLinkAddress: Url
+ urlLinkDescription: MultilingualString [0..1]
+ urlLinkType: UrlLinkTypeEnum [0..1]
0..*
«D2Relation»
«D2Class»
GenericSituationRecord
«D2Attribute»
+ genericSituationRecordName: String
«D2Relation»
+informationManagerOverride
0..1
«D2Class»
InternationalIdentifier
«D2Attribute»
+ country: CountryCode
+ nationalIdentifier: String
«D2Relation»
0..1
«D2Class»
Cause
«D2Attribute»
+ causeDescription: MultilingualString [0..1]
+ causeType: CauseTypeEnum [0..1]
«D2Class»
ServiceInformation
«D2Class»
OperatorAction
«D2Attribute»
+ actionOrigin: OperatorActionOriginEnum [0..1]
+ actionPlanIdentifier: String [0..1]
+ operatorActionStatus: OperatorActionStatusEnum [0..1]
«D2Class»
TrafficElement
«D2Attribute»
+ trafficConstrictionType: TrafficConstrictionTypeEnum [0..1]
«D2Relation»
«D2Relation»
+detailedCauseType 0..1
+managedCause 0..1
«D2Class»
DetailedCauseType
«D2Attribute»
+ abnormalTrafficType: AbnormalTrafficTypeEnum [0..1]
+ accidentType: AccidentTypeEnum [0..*]
+ animalPresenceType: AnimalPresenceTypeEnum [0..1]
+ authorityOperationType: AuthorityOperationTypeEnum [0..1]
+ constructionWorkType: ConstructionWorkTypeEnum [0..1]
+ disturbanceActivityType: DisturbanceActivityTypeEnum [0..1]
+ drivingConditionType: DrivingConditionTypeEnum [0..1]
+ environmentalObstructionType: EnvironmentalObstructionTypeEnum [0..1]
+ equipmentOrSystemFaultType: EquipmentOrSystemFaultTypeEnum [0..1]
+ generalInstructionToRoadUsersType: GeneralInstructionToRoadUsersTypeEnum [0..1]
+ generalNetworkManagementType: GeneralNetworkManagementTypeEnum [0..1]
+ infrastructureDamageType: InfrastructureDamageTypeEnum [0..1]
+ nonWeatherRelatedRoadConditionType: NonWeatherRelatedRoadConditionTypeEnum [0..*]
+ obstructionType: ObstructionTypeEnum [0..*]
+ poorEnvironmentType: PoorEnvironmentTypeEnum [0..*]
+ publicEventType: PublicEventTypeEnum [0..1]
+ reroutingManagementType: ReroutingManagementTypeEnum [0..*]
+ roadMaintenanceType: RoadMaintenanceTypeEnum [0..*]
+ roadOperatorServiceDisruptionType: RoadOperatorServiceDisruptionTypeEnum [0..*]
+ roadOrCarriagewayOrLaneManagementType: RoadOrCarriagewayOrLaneManagementTypeEnum [0..1]
+ roadsideAssistanceType: RoadsideAssistanceTypeEnum [0..1]
+ roadsideServiceDisruptionType: ServiceDisruptionTypeEnum [0..*]
+ speedManagementType: SpeedManagementTypeEnum [0..1]
+ transitServiceInformation: TransitServiceInformationEnum [0..1]
+ vehicleObstructionType: VehicleObstructionTypeEnum [0..1]
+ weatherRelatedRoadConditionType: WeatherRelatedRoadConditionTypeEnum [0..*]
+ winterEquipmentManagementType: WinterEquipmentManagementTypeEnum [0..1]
GlobalReference
«D2Class»
SituationRecordReference
«D2Attribute»
+ objectReference: VersionedReference

Fig. 42: DATEX II – situation record

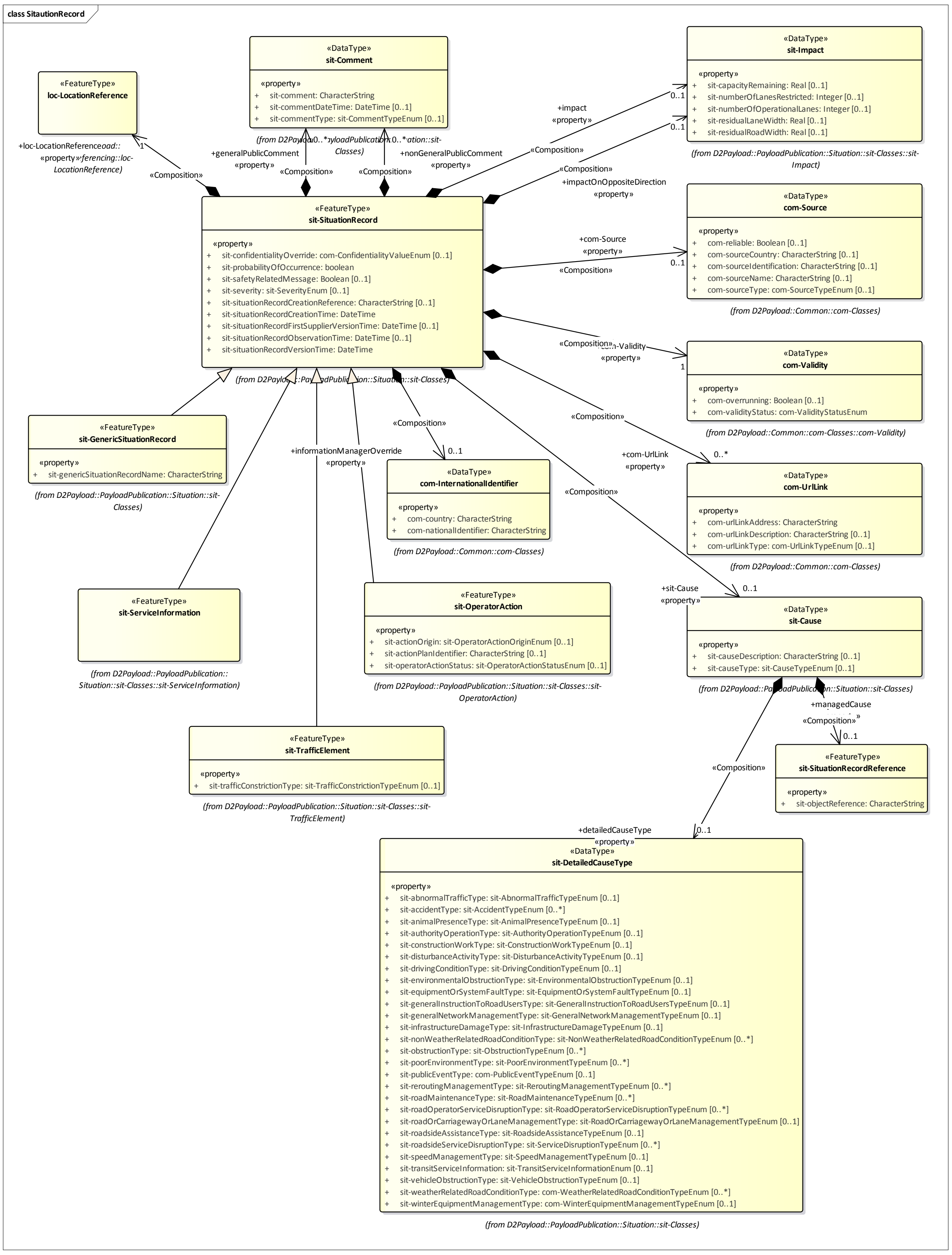

class SitautionRecord
«FeatureType»
loc-LocationReference
+loc-LocationReference
«property»
(from D2Payload::Referencing::loc-LocationReference)
1
«Composition»
«DataType»
sit-Comment
«property»
+ sit-comment: CharacterString
+ sit-commentDateTime: DateTime [0..1]
+ sit-commentType: sit-CommentTypeEnum [0..1]
(from D2Payload::PayloadPublication::Situation::sit-Classes)
0..*
0..*
+generalPublicComment
«property»
«Composition»
«Composition»
+nonGeneralPublicComment
«property»
«FeatureType»
sit-SituationRecord
«property»
+ sit-confidentialityOverride: com-ConfidentialityValueEnum [0..1]
+ sit-probabilityOfOccurrence: boolean
+ sit-safetyRelatedMessage: Boolean [0..1]
+ sit-severity: sit-SeverityEnum [0..1]
+ sit-situationRecordCreationReference: CharacterString [0..1]
+ sit-situationRecordCreationTime: DateTime
+ sit-situationRecordFirstSupplierVersionTime: DateTime [0..1]
+ sit-situationRecordObservationTime: DateTime [0..1]
+ sit-situationRecordVersionTime: DateTime
(from D2Payload::PayloadPublication::Situation::sit-Classes)
«DataType»
sit-Impact
«property»
+ sit-capacityRemaining: Real [0..1]
+ sit-numberOfLanesRestricted: Integer [0..1]
+ sit-numberOfOperationalLanes: Integer [0..1]
+ sit-residualLaneWidth: Real [0..1]
+ sit-residualRoadWidth: Real [0..1]
(from D2Payload::PayloadPublication::Situation::sit-Classes::sit-Impact)
+impact
«property»
«Composition»
0..1
«Composition»
+impactOnOppositeDirection
«property»
0..1
«DataType»
com-Source
«property»
+ com-reliable: Boolean [0..1]
+ com-sourceCountry: CharacterString [0..1]
+ com-sourceIdentification: CharacterString [0..1]
+ com-sourceName: CharacterString [0..1]
+ com-sourceType: com-SourceTypeEnum [0..1]
(from D2Payload::Common::com-Classes)
+com-Source
«property»
«Composition»
0..1
«Composition»
+com-Validity
«property»
1
«DataType»
com-Validity
«property»
+ com-overrunning: Boolean [0..1]
+ com-validityStatus: com-ValidityStatusEnum
(from D2Payload::Common::com-Classes::com-Validity)
«Composition»
+com-UrlLink
«property»
0..*
«DataType»
com-UrlLink
«property»
+ com-urlLinkAddress: CharacterString
+ com-urlLinkDescription: CharacterString [0..1]
+ com-urlLinkType: com-UrlLinkTypeEnum [0..1]
(from D2Payload::Common::com-Classes)
«FeatureType»
sit-GenericSituationRecord
«property»
+ sit-genericSituationRecordName: CharacterString
(from D2Payload::PayloadPublication::Situation::sit-Classes)
+informationManagerOverride
«property»
«Composition»
0..1
«DataType»
com-InternationalIdentifier
«property»
+ com-country: CharacterString
+ com-nationalIdentifier: CharacterString
(from D2Payload::Common::com-Classes)
«Composition»
«FeatureType»
sit-ServiceInformation
(from D2Payload::PayloadPublication::Situation::sit-Classes::sit-ServiceInformation)
«FeatureType»
sit-OperatorAction
«property»
+ sit-actionOrigin: sit-OperatorActionOriginEnum [0..1]
+ sit-actionPlanIdentifier: CharacterString [0..1]
+ sit-operatorActionStatus: sit-OperatorActionStatusEnum [0..1]
(from D2Payload::PayloadPublication::Situation::sit-Classes::sit-OperatorAction)
+sit-Cause
«property»
0..1
«DataType»
sit-Cause
«property»
+ sit-causeDescription: CharacterString [0..1]
+ sit-causeType: sit-CauseTypeEnum [0..1]
(from D2Payload::PayloadPublication::Situation::sit-Classes)
+managedCause
«Composition»
0..1
«FeatureType»
sit-SituationRecordReference
«property»
+ sit-objectReference: CharacterString
«Composition»
«FeatureType»
sit-TrafficElement
«property»
+ sit-trafficConstrictionType: sit-TrafficConstrictionTypeEnum [0..1]
(from D2Payload::PayloadPublication::Situation::sit-Classes::sit-TrafficElement)
+detailedCauseType
«property»
0..1
«DataType»
sit-DetailedCauseType
«property»
+ sit-abnormalTrafficType: sit-AbnormalTrafficTypeEnum [0..1]
+ sit-accidentType: sit-AccidentTypeEnum [0..*]
+ sit-animalPresenceType: sit-AnimalPresenceTypeEnum [0..1]
+ sit-authorityOperationType: sit-AuthorityOperationTypeEnum [0..1]
+ sit-constructionWorkType: sit-ConstructionWorkTypeEnum [0..1]
+ sit-disturbanceActivityType: sit-DisturbanceActivityTypeEnum [0..1]
+ sit-drivingConditionType: sit-DrivingConditionTypeEnum [0..1]
+ sit-environmentalObstructionType: sit-EnvironmentalObstructionTypeEnum [0..1]
+ sit-equipmentOrSystemFaultType: sit-EquipmentOrSystemFaultTypeEnum [0..1]
+ sit-generalInstructionToRoadUsersType: sit-GeneralInstructionToRoadUsersTypeEnum [0..1]
+ sit-generalNetworkManagementType: sit-GeneralNetworkManagementTypeEnum [0..1]
+ sit-infrastructureDamageType: sit-InfrastructureDamageTypeEnum [0..1]
+ sit-nonWeatherRelatedRoadConditionType: sit-NonWeatherRelatedRoadConditionTypeEnum [0..*]
+ sit-obstructionType: sit-ObstructionTypeEnum [0..*]
+ sit-poorEnvironmentType: sit-PoorEnvironmentTypeEnum [0..*]
+ sit-publicEventType: com-PublicEventTypeEnum [0..1]
+ sit-reroutingManagementType: sit-ReroutingManagementTypeEnum [0..*]
+ sit-roadMaintenanceType: sit-RoadMaintenanceTypeEnum [0..*]
+ sit-roadOperatorServiceDisruptionType: sit-RoadOperatorServiceDisruptionTypeEnum [0..*]
+ sit-roadOrCarriagewayOrLaneManagementType: sit-RoadOrCarriagewayOrLaneManagementTypeEnum [0..1]
+ sit-roadsideAssistanceType: sit-RoadsideAssistanceTypeEnum [0..1]
+ sit-roadsideServiceDisruptionType: sit-ServiceDisruptionTypeEnum [0..*]
+ sit-speedManagementType: sit-SpeedManagementTypeEnum [0..1]
+ sit-transitServiceInformation: sit-TransitServiceInformationEnum [0..1]
+ sit-vehicleObstructionType: sit-VehicleObstructionTypeEnum [0..1]
+ sit-weatherRelatedRoadConditionType: com-WeatherRelatedRoadConditionTypeEnum [0..*]
+ sit-winterEquipmentManagementType: com-WinterEquipmentManagementTypeEnum [0..1]
(from D2Payload::PayloadPublication::Situation::sit-Classes)

Fig. 43: GML – situation record