Open Geospatial Consortium

Publication Date: 2014-07-16

Approval Date: 2014-06-14

Posted Date: 2014-05-19

Reference number of this document: OGC 14-037

Reference URL for this document: http://www.opengeospatial.net/doc/PER/testbed10/fixm-gml-schema

Category: Public Engineering Report

Editors: Thomas Forbes, Ballal Joglekar

OGC® Testbed 10 Flight Information Exchange Model GML Schema

Document type:	OGC® Engineering Report
Document subtype:	NA
Document stage:	Approved for public release
Document language:	English

Preface

This report provides guidance for implementing the Flight Information Exchange Model (FIXM) using the same best practice as the Aeronautical Information Exchange Model (AIXM) and the Weather Information Exchange Model (WXXM) by adopting ISO and OGC standards.

The report is aimed at system and client developers that shall use the FIXM data encoding for the exchange of flight information.

This document is a deliverable for the OGC Testbed 10 (Testbed-10) testbed activity. OWS testbeds are part of OGC's Interoperability Program, a global, hands-on and collaborative prototyping program designed to rapidly develop, test and deliver proven candidate standards or revisions to existing standards into OGC's Standards Program, where they are formalized for public release. In OGC's Interoperability Initiatives, international teams of technology providers work together to solve specific geoprocessing interoperability problems posed by the Initiative's sponsoring organizations. OGC Interoperability Initiatives include testbeds, pilot projects, interoperability experiments and interoperability support services - all designed to encourage rapid development, testing, validation and adoption of OGC standards.

The Testbed-10 sponsors are organizations seeking open standard for their interoperability requirements. After analyzing their requirements, the OGC Interoperability Team recommends to the sponsors that the content of the Testbed-10 initiative be organized around the following threads:

- Cross-Community Interoperability (CCI)
- Open Mobility
- Aviation

More information about the Testbed-10 tested can be found at:

http://www.opengeospatial.org/standards/requests/103

Keywords

ogcdoc, ogc documents, fixm, gml, Testbed10, schemas

Contents

Page

Figures

Tables **Page**

OGC® Testbed 10 Flight Information Exchange Model GML Schema

1 Introduction

1.1 Background

Built on a foundation of OGC and ISO standards, AIXM and WXXM have demonstrated that significant barriers to adoption (complexity and cost of custom system builds) can be mitigated through engaging with the wider open standards community and reusing best practice patterns for application schema design and service delivery. At the data level AIXM and WXXM share common types and patterns from ISO 19136 Geography Markup Language (GML) making them interoperable between each other, as well as saving cost and time through the reuse of globally implemented types such as feature models, geometry, temporality, unit of measure and coordinate reference systems. Being built on such a globally adopted standard as GML, AIXM and WXXM are compatible with hundreds of existing tools as well as being understandable by communities across the globe. The result has been a switch from industry specific custom software builds to off-the-shelf technology choices and with that a dramatic reduction in implementation cost. This interoperability and openness is at the heart of the net-centric vision of System Wide Information Management (SWIM). This OGC document demonstrates that the benefits of interoperability and open standards architecture are agnostic of exchange model and that FIXM can be implemented following the same best practice as AIXM and WXXM and therefore realizing the same operational benefits.

1.2 Scope

The scope of this work is to:

1. Demonstrate ISO 19136 compliance within FIXM 2.0 and document the process required to make best use of OGC Aviation Architecture as supported by AIXM and WXXM;
2. Provide an update to the FIXM 2.0 application schemas adhering to ISO 19109 (Geographic Information - Rules for Application Schema) where appropriate;
3. Propose recommendations for future FIXM development.
4. Demonstrate the operational benefits of interoperability and open standards architecture.

1.3 Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization
Thomas Forbes	Snowflake Software Ltd.
Ballal Joglekar	Snowflake Software Ltd.

1.4 Revision history

Date	Release	Editor	Primary clauses modified	Description
2014-01-31	Version .1	T. Forbes & B. Joglekar	All	Initial draft.
2014-04-14	Version .2	T. Forbes	All	Final Draft ER for review.
2014-05-14	Version .3	T. Forbes	All	Final ER

1.5 Future work

Improvements in this document are desirable as the OGC community continues its efforts to advance interoperability using OGC and ISO standards within the aeronautical domain.

The following topics should be addressed in follow-up activities:

- XML attribute removal performance impact assessment.
- Address the issues raised in Section 9 and 10.

1.6 Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

2 References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

ISO 19103:2005 – *Geographic Information – Conceptual Schema Language*

ISO 19107:2003 – *Geographic Information – Spatial Schema*

ISO 19109:2005 – *Geographic Information – Rules for Application Schema*

ISO 19110:2005 – *Geographic Information – Methodology for Feature Cataloguing*

ISO 19136:2007 – *Geographic Information – Geography Markup Language (GML)*
OGC 06-121r3, *OGC® Web Services Common Standard*

NOTE - This OWS Common Standard contains a list of normative references that are also applicable to this Implementation Standard.

3 Conventions

3.1 Abbreviated terms

AIRM	ATM Information Reference Model
AIXM	Aeronautical Information Exchange Model
FIXM	Flight Information Exchange Model
GML	Geography Markup Language
ICAO	International Civil Aviation Organization
ISO	International Organization for Standardization
IWXXM	ICAO WXXM
OCL	Object Constraint Language
OGC	Open Geospatial Consortium
OMG	Object Management Group
OWS	OGC Web Services
SWIM	System Wide Information Management
UGAS	UML to GML Application Schema
UML	Unified Modeling Language
WXXM	Weather Information Exchange Model
XML	Extensible Markup Language

3.2 UML notation

Diagrams that appear in this document may be presented using the Unified Modeling Language (UML) static structure diagram, as described in ISO 19103 (Geographic Information – Conceptual Schema Language) and Subclause 5.2 of [OGC 06-121r3].

4 ISO 19136 Compliant FIXM 2.0 Schema Generation

4.1 Approach

A two stage approach was undertaken to generate ISO 19136 compliant FIXM 2.0 XSDs from the FIXM 2.0 UML[1]:

1. Modification of FIXM 2.0 UML to adhere to the ISO 19136 UML Profile for GML within Enterprise Architect.
2. Transform the modified UML model into a physical implementation model using the ShapeChange UML to GML Application Schema (UGAS) tool.

4.2 Transformation Tools

The implemented approach required two tools:

- Sparx Systems Enterprise Architect (v8.0)
- Interactive Instruments ShapeChange (v2.0)

4.2.1 Enterprise Architect

Sparx Systems Enterprise Architect is a visual modeling and design tool based on the OMG UML. The FIXM 2.0 schemas were developed using Enterprise Architect; it is the recommended tool for extending or modifying the existing FIXM UML model.

[1]FIXM 2.0 UML available for download at: http://www.fixm.aero/fixm_20

4.2.2 ShapeChange

Developed in the OWS-3 testbed by Interactive Instruments, ShapeChange[2] is a Java tool that takes application schemas constructed according to ISO 19109 from a UML model and derives implementation representations (Figure 1). The most commonly used target representation is XML Schema (GML, ISO/TS 19139, SWE Common); ShapeChange is also capable of generating:

- Code list dictionaries in GML and SKOS.
- RDF schemas.
- Feature catalogues (ISO 19110) in DOCX and HTML.
- Schematron documents from OCL constraints defined in the UML model.

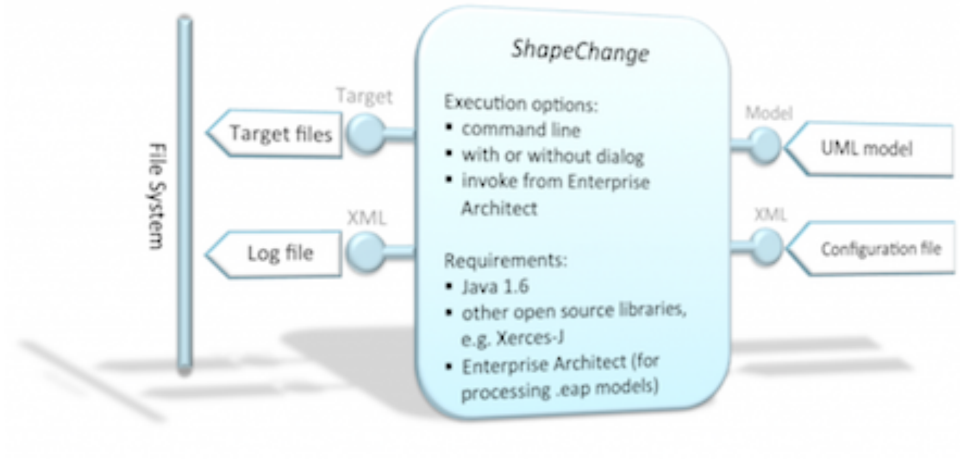


Figure 1 The ShapeChange implementation process overview

ShapeChange supports a range of standardized encoding rules for converting UML to GML application schemas[3]:

- ISO 19136 / GML 3.2 encoding rule for GML application schemas
- ISO/CD 19136-2 / GML 3.3 extensions
- ISO/TS 19139 encoding rule
- SWE Common 2.0 Data Model encoding rule
- INSPIRE encoding rule
- GSIP encoding rule
- ShapeChange extensions

[2] http://shapechange.net/

[3] http://shapechange.net/targets/xsd/

These encoding rules have been based upon the encoding rules specified in Annex E of the GML 3.2 standard, in the GML 3.3 standard and ISO/TS 19139 – as well as a series of extensions to the standardized rules.

5 Transformation Rules

To transform the FIXM 2.0 UML model to comply with the ISO 19136 profile, two sets of transformation rules are required:

1. MDA Transformation Rules: these define how the FIXM UML will be modeled in order to conform to the ISO 19136 UML Profile.

2. Encoding Rules: used for transforming ISO TC 211 and FIXM types into GML types.

Coupled with the above transformation rules, minor FIXM 2.0 model structure modification was required.

5.1 MDA Transformation Rules

MDA transformation rules are required in order to specify how the FIXM 2.0 UML model will be modified to conform to the ISO 19136 UML Profile. In order to achieve compliance, ISO 19136 UML stereotypes and associated tagged values need to be applied to the model constructs (Packages, Classes, Attributes and Associations).

UML contains three extensibility mechanisms: stereotypes, tagged values and constraints. Model constructs (Packages, Classes, Attributes and Associations) were assigned ISO 19136 UML stereotypes and tagged values where appropriate.

5.1.1 Stereotypes

A UML stereotype is an extension mechanism for existing UML concepts. It is a model element that is used to classify (or mark) other UML elements so that they, in some respect, behave as if they were instances of new virtual or pseudo meta-model classes, whose form is based on existing base meta-model classes. Stereotypes augment the classification mechanisms on the basis of the built-in UML meta-model class hierarchy. Therefore, names of new stereotypes must not clash with predefined meta-model elements or other stereotypes.

Stereotypes can be applied to most of the model elements, including Packages, Classes, Attributes and Associations. They enable the use of platform or domain specific terminology within the model.

UML 2.0 defines a core set of Stereotypes. ISO 19103 and ISO 19109 contain UML Profiles which have extended this core set of Stereotypes to define additional stereotypes which provide a consistent set of meta-model elements for developing geospatial information models. The extended ISO 19136 UML profile used to underpin the FIXM 2.0 UML model defines various stereotypes, these are discussed below.

5.1.1.1 Package Stereotypes

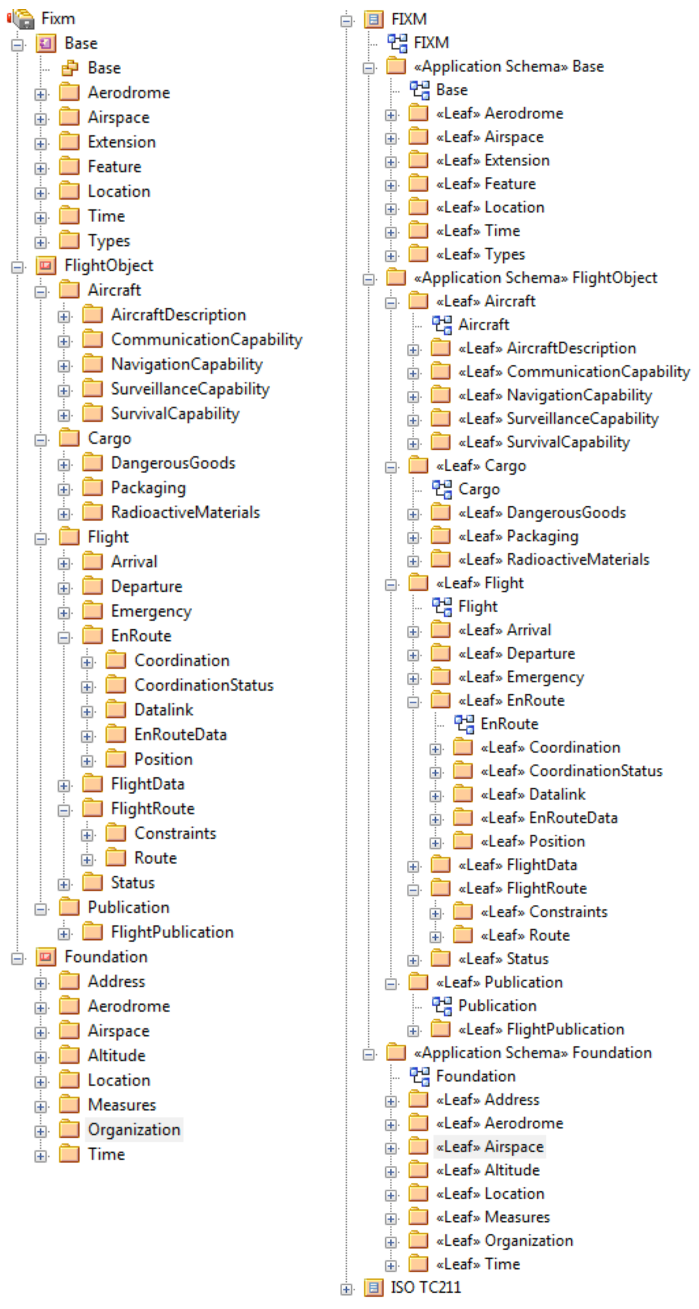
The UML Application Schema shall be represented by a package with the stereotype <<ApplicationSchema>>. The ISO 19109 Rules for Application Schema state this package shall contain all UML model elements to be mapped to object types in the GML application schema. The package may include other child packages without the stereotype <<ApplicationSchema>> to group the different UML model elements within the application schema. Typically these child packages are assigned a stereotype <<Leaf>>. Packages with the stereotype <<Leaf>> cannot contain child packages themselves.

Table 1 Package stereotype overview

Stereotype	Use	XML Implementation
<<ApplicationSchema>>	Complete application schema	An XML Schema in a single namespace.
<<Leaf>>	Convenient group of elements within an application schema	Single XML Schema document.

Package Stereotype Example

The parent UML packages Base, FlightObject and Foundation were assigned an <<ApplicationSchema>> stereotype, whilst all other child packages were assigned a <<Leaf>> stereotype (Figure 2).



```
Fixm
  Base
    Base
    Aerodrome
    Airspace
    Extension
    Feature
    Location
    Time
    Types
  FlightObject
    Aircraft
      AircraftDescription
      CommunicationCapability
      NavigationCapability
      SurveillanceCapability
      SurvivalCapability
    Cargo
      DangerousGoods
      Packaging
      RadioactiveMaterials
    Flight
      Arrival
      Departure
      Emergency
      EnRoute
        Coordination
        CoordinationStatus
        Datalink
        EnRouteData
        Position
      FlightData
      FlightRoute
        Constraints
        Route
      Status
    Publication
      FlightPublication
  Foundation
    Address
    Aerodrome
    Airspace
    Altitude
    Location
    Measures
    Organization
    Time
```

```
FIXM
  FIXM
  «Application Schema» Base
    Base
    «Leaf» Aerodrome
    «Leaf» Airspace
    «Leaf» Extension
    «Leaf» Feature
    «Leaf» Location
    «Leaf» Time
    «Leaf» Types
  «Application Schema» FlightObject
    «Leaf» Aircraft
      Aircraft
      «Leaf» AircraftDescription
      «Leaf» CommunicationCapability
      «Leaf» NavigationCapability
      «Leaf» SurveillanceCapability
      «Leaf» SurvivalCapability
    «Leaf» Cargo
      Cargo
      «Leaf» DangerousGoods
      «Leaf» Packaging
      «Leaf» RadioactiveMaterials
    «Leaf» Flight
      Flight
      «Leaf» Arrival
      «Leaf» Departure
      «Leaf» Emergency
      «Leaf» EnRoute
        EnRoute
        «Leaf» Coordination
        «Leaf» CoordinationStatus
        «Leaf» Datalink
        «Leaf» EnRouteData
        «Leaf» Position
      «Leaf» FlightData
      «Leaf» FlightRoute
        «Leaf» Constraints
        «Leaf» Route
      «Leaf» Status
    «Leaf» Publication
      Publication
      «Leaf» FlightPublication
  «Application Schema» Foundation
    Foundation
    «Leaf» Address
    «Leaf» Aerodrome
    «Leaf» Airspace
    «Leaf» Altitude
    «Leaf» Location
    «Leaf» Measures
    «Leaf» Organization
    «Leaf» Time
ISO TC211
```

Figure 2 FIXM UML Package stereotypes - before (left) and after (right) modification

5.1.1.2 Class Stereotypes

The following UML Class stereotypes were assigned:

- Features shall be modeled as UML classes with stereotype <<FeatureType>>.
- Enumerations shall be modeled as UML classes with stereotype <<Enumeration>>.
- Union types shall be modeled as UML classes with stereotype <<Union>> (as specified in ISO 19107).
- All other data types shall be modeled as UML classes with stereotype <<DataType>>.

Table 2 Class stereotypes overview

Stereotype	Use	XML Implementation
<<FeatureType>>	Feature type	XML element whose XML Schema type is derived from gml:AbstractFeatureType.
<<Enumeration>>	Fixed enumeration	Enumeration of string values.
<<Union>>	Arbitrary set of alternative classes	Choice group whose members are GML Objects or Features, or objects corresponding to DataTypes.
<<DataType>>	Structured data type	XML element with a complex content model; does not have identity and must appear inline.

Class Stereotype Example

For example, within the Foundation::Organization package, the class stereotypes <<Union>>, <<DataType>> and <<Enumeration>> are specified.

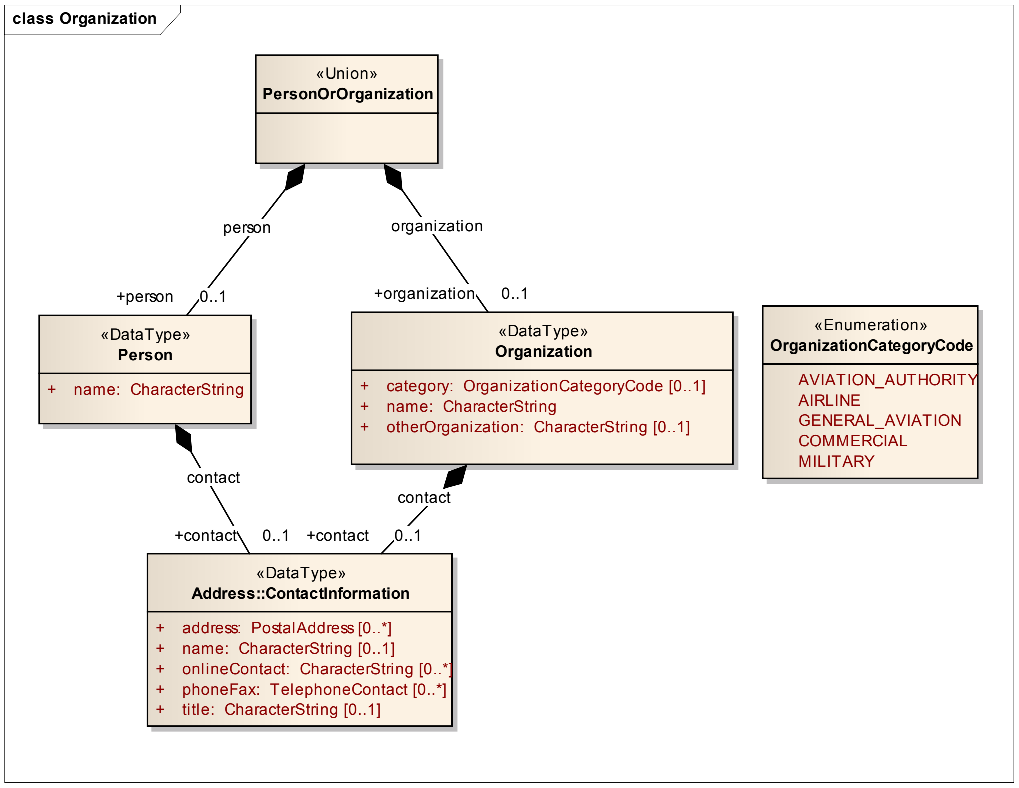


Figure 3 Class stereotype example within the Foundation::Organization package

5.1.2 Tagged Values

The ISO 19136 UML Profile requires specific Tagged Values to be applied to specific model constructs. These tagged values are required to successfully execute specific encoding rules when transforming the UML model into GML application schema.

5.1.2.1 Packages

A unique XML namespace shall be associated with the UML Application Schema. Tagged values targetNamespace (used for specifying the target namespace URI) and xmlns (used for specifying the XML prefix) shall be set if and only if the package represents a UML application schema.

Where a package shall be mapped to its own XML Schema document, a tagged value xsdDocument shall be set providing a valid relative file name of the schema document. The tagged value shall be set for every package representing the UML Application Schema. All tagged values xsdDocument in a UML model shall be unique.

One XML Schema document is generated per package with the tagged value xsdDocument with the file name specified by the tagged value. If the tagged value xsdDocument is set for a package, then the schema document contains all the XML Schema components resulting from the UML classes directly owned by the package. If the package is not a UML application schema, the schema document shall be included by the schema document that contains the schema components of the package that owns the package.

If the tagged value xsdDocument is not set for a package, all schema components are declared in the schema document that contains the schema components of the package that owns the package.

For every schema document, the targetNamespace and the version attributes of the root element shall be set in accordance with the tagged values of the same name in the package representing the UML Application Schema that owns the schema components within the schema document; if the version tagged value is not specified, the value "unknown" shall be used. In addition an xmlns attribute shall be specified for the target namespace with the value of the tagged value xmlns as the prefix.

Table 3 summarises the Stereotypes and Tagged Values that shall be assigned to Packages.

Table 3 Package Stereotypes and Tagged Values

Stereotype	Description	Tagged Value	Description
Application Schema	An application schema according to the ISO 19109 and the INSPIRE Generic Conceptual Model	targetNamespace	Target XML namespace of the application schema (ISO 19136).
		xmlns	Namespace prefix to be used as short form of the target namespace (ISO 19136).
		version	Current version of the application schema (ISO 19136).
		gmlProfileSchema	URL of the schema location of a GML profile (where applicable) (ISO 19136).
		xsdDocument	Name of an XML Schema document to create representing the content of this package (ISO 19136).
		xsdEncodingRule	XML Schema encoding rule to apply (example: iso19136_2007, gml33 or iso19139_2007 etc).
Leaf	A package that is not an application schema.	xsdDocument	Name on an XML Schema document to create representing the content of this package (ISO 19136).
		xsdEncodingRule	XML Schema encoding rule to apply (example: iso19136_2007, gml33 or iso19139_2007 etc).

5.1.2.2 Classes

Table 4 summarises the Class Stereotypes and respective Tagged Values.

Table 4 Class stereotypes and tagged values

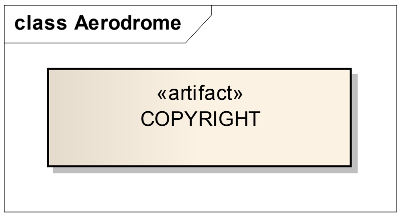
Stereotype	Description	Tagged Value	Description
FeatureType	A spatial object type (ISO 19136)	xsdEncodingRule	XML Schema encoding rule to apply (example: iso19136_2007, gml33 or iso19139_2007 etc)
		noPropertyType	Suppress creation of a standard property type that supports inline or by-reference encoding (applies to ISO 19136:2007 encoding rule). Always set to false (ISO 19136)
		byValuePropertyType	Create a property type that requires that the instance is encoded inline (applies to ISO 19136:2007 encoding rule). Always set to false (ISO 19136)
		isCollection	Identifies the feature type as a feature collection (ISO 19136). Default = false
Type	A structured data type with identity	xsdEncodingRule	XML Schema encoding rule to apply (example: iso19136_2007, gml33 or iso19139_2007 etc)
		noPropertyType	Suppress creation of a standard property type that supports inline or by-reference encoding (applies to ISO 19136:2007 encoding rule). Always set to false (ISO 19136)
		byValuePropertyType	Create a property type that requires that the instance is encoded inline (applies to ISO 19136:2007 encoding rule). Always set to false (ISO 19136)
		isCollection	Identifies the feature type as a feature collection (ISO 19136). Default = false
		gmlMixin	Identifies the feature type as a mixin type that will not be encoded as a separate element/type in the GML encoding
		xmlSchemaType	If the type has a canonical XML Schema encoding the XML Schema typename corresponding to the data type shall be given as the value (applies to ISO 19136:2007

Stereotype	Description	Tagged Value	Description
			encoding rule) (ISO 19136)
DataType	A structured data type without identity (ISO 19103)	xsdEncodingRule	XML Schema encoding rule to apply (example: iso19136_2007, gml33 or iso19139_2007 etc)
		noPropertyType	Suppress creation of a standard property type that supports inline or by-reference encoding (applies to ISO 19136:2007 encoding rule). Always set to false (ISO 19136)
		isCollection	Identifies the feature type as a feature collection (ISO 19136). Default = false
Union	A structured data type without identity where exactly one of the properties of the type is present in any instance. (ISO 19103)	xsdEncodingRule	XML Schema encoding rule to apply (example: iso19136_2007, gml33 or iso19139_2007 etc)
		noPropertyType	Suppress creation of a standard property type that supports inline or by-reference encoding (applies to ISO 19136:2007 encoding rule). Alwas set to false (ISO 19136)
Enumeration	An enumeration	xsdEncodingRule	XML Schema encoding rule to apply (example: iso19136_2007, gml33 or iso19139_2007 etc)
CodeList	A Code list	xsdEncodingRule	XML Schema encoding rule to apply (example: iso19136_2007, gml33 or iso19139_2007 etc)
		asDictionary	Encode code list as externally managed dictionary (applies to ISO 19136:2007 encoding rule). Always set to true (19136).
		extensibility	Used to define whether the code list is extensible by third parties.
		vocabulary	URI of the vocabulary/code list in the code list registry or in some external registry. The value has to be provided, if an online version of the vocabulary exists.
		codespace	The value should identify a dictionary, thesaurus or authority for the term, such as an organisation who assigned the value, or the dictionary from which it is taken. The value should be represented as a HTTP URI.

5.1.2.2.1 Examples of Class Tagged Values

Suppressing the COPYRIGHT classes

All classes (with the exception of the COPYRIGHT classes) did not have specified Tagged Values. The COPYRIGHT classes present within each package were suppressed using the Tagged Value xsdEncodingRule with the value notEncoded.



Tag	Value
xsdEncodingRule	notEncoded

Figure 4 Example COPYRIGHT class and associated tagged values

5.1.2.3 Attributes

Table 5 summarises the possible tagged values to assign to UML attributes.

Table 5 UML attribute stereotypes and tagged values

Tagged Value	Description
sequenceNumber	Unique integer value for properties of the type used to sort properties (applies to ISO 19136:2007 encoding rule) (ISO 19136)
inlineOrByReference	Controls whether property values may be encoded inline or by reference (applies to ISO 19136:2007 encoding rule) Valid values are: - inline - byReference and - inlineOrByReference Default is inlineOrByReference
isMetadata	Indicates whether the property is considered metadata about the instance and not information about the phenomenon in the real world (ISO 19136).
xsdEncodingRule	XML Schema encoding rule to apply (example: iso19136_2007, gml33 or iso19139_2007 etc).
xsdAsAttribute	Indicates whether the UML attribute should be encoded as a XML attribute rather than an XML element.

5.1.3 Relationships and Associations

Table 6 summarises the Stereotypes and Tagged Values that shall be assigned to Relationships and Associations.

Table 6 Relationships and Associations stereotypes and Tagged Values

Stereotype	Description	Tagged Value	Description
import	Dependency	n/a	The model element of the supplier package are imported.
n/a	Association Role	sequenceNumber	Unique integer value for properties of the type used to sort properties (applies to ISO 19136:2007 encoding rule) (ISO 19136)
		inlineOrByReference	Controls whether property values may be encoded inline or by reference (applies to ISO 19136:2007 encoding rule) Valid values are: - inline - byReference and - inlineOrByReference Default is inlineOrByReference
		isMetadata	Indicates whether the property is considered metadata about the instance and not information about the phenomenon in the real world (ISO 19136)
n/a	Generalisation	gmlMixin	Optional tagged value to be assigned to a Class with Multiple inheritance. This is only required if a GML application schema shall be generated from the model.

5.1.4 Constraints

A constraint is a UML element which represents some condition, restriction or assertion related to some element (that owns the constraint) or several elements. A constraint is usually specified by a Boolean expression which must evaluate to a true or false. Constraints must be satisfied (i.e. evaluated to true) by a correct design of the system. Constraints are commonly used for various elements on class diagrams. The UML specification does not restrict the language which could be used to express constraints however the Object Constraint Language (OCL) is predefined in UML and can be transformed into platform implementation constraints such as Schematron for validating XML document.

Constraint implementation was not executed in the UML to XSD conversion as inconsistencies exist between the data sources used in the testbed and the FIXM 2.0 constraints. Constraints could be later instated, both the CASE and UGAS tools offer functionality for implementing GML constraints.

5.2 Inclusion of ISO 19136 Geometry, Temporality & Basic Types

Where FIXM 2.0 uses XML types for geometry, temporality and basic types these will be substituted for GML types to adhere to ISO standards. Supported types and implementation examples are shown below.

5.2.1 Geometry

5.2.1.1 Overview of ISO 19107 Spatial Schema

ISO 19107 (Geographic Information – Spatial Schema) specifies conceptual schemas for describing the spatial characteristics of geographic features, and a set of spatial operations consistent with these schemas. It treats vector geometry and topology up to three dimensions. It defines standard spatial operations for use in access, query, management, processing, and data exchange of geographic information for spatial (geometric and topological) objects of up to three topological dimensions embedded in coordinate spaces of up to three axes.

5.2.1.2 Profile of ISO 19107 Spatial Schema

Geometry and topology types as specified in ISO 19107 may be used in the FIXM GML schema without restrictions (Table 7).

NOTE: This recommendation has two aspects: to use spatial geometries that are simple (e.g. use linear interpolation only) and to avoid spatial topology unless they are essential.

With respect to the use of topology the following should be considered: Topological constraints are important in the maintenance of many spatial data sets. However, as the data is shared between two systems, it is usually advisable to not use topological primitives for the following reasons:

- Many software packages are not able to handle topological primitives and usually just do not need them to draw a map or compute spatial relationships.
- The topological relations of two spatial objects based on their specific geometry and topology properties can in principle be investigated by invoking the operations of the types defined in ISO 19107 (or the methods specified in OGC 06-103r3). Indeed, this is also what GIS software products typically does today; they derive and maintain topological relationships from the spatial data as needed and based on topological rules in their model.

However, these are physical constraints; if the topological primitive is the best description of the conceptual logical relationship then it should be used and revised when physically implemented.

The following geometry classes can be used as the type for a geometry attribute (Figure 5 - Figure 6).

Table 7 ISO 19107 Geometry Classes

Name	Description
GM_Object	GM_Object is the parent class of the geometric object taxonomy and supports interfaces common to all geographically referenced geometric objects.
GM_Point	GM_Point is the basic data type for a geometric object consisting of one and only one point.
GM_Curve	GM_Curve is a descendant subtype of GM_Primitive through GM_OrientablePrimitive. It is the basis for 1-dimensional geometry.
GM_Surface	GM_Surface is a subclass of GM_Primitive and is the basis of 2-dimensional geometry.
GM_Solid	GM_Solid is a subclass of GM_Primitive, is the basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces.
GM_MultiPoint	GM_MultiPoint is an aggregate of disconnected point geometries that represent the extent of a feature.
GM_MultiCurve	GM_MultiCurve is an aggregate of disconnected linear geometries that represent the extent of a feature.
GM_MultiSurface	GM_MultiSurface is an aggregate of disconnected surface geometries that represent the extent of a feature.
GM_MultiSolid	GM_MultiSolid is an aggregate of disconnected point geometries that represent the extent of a feature.

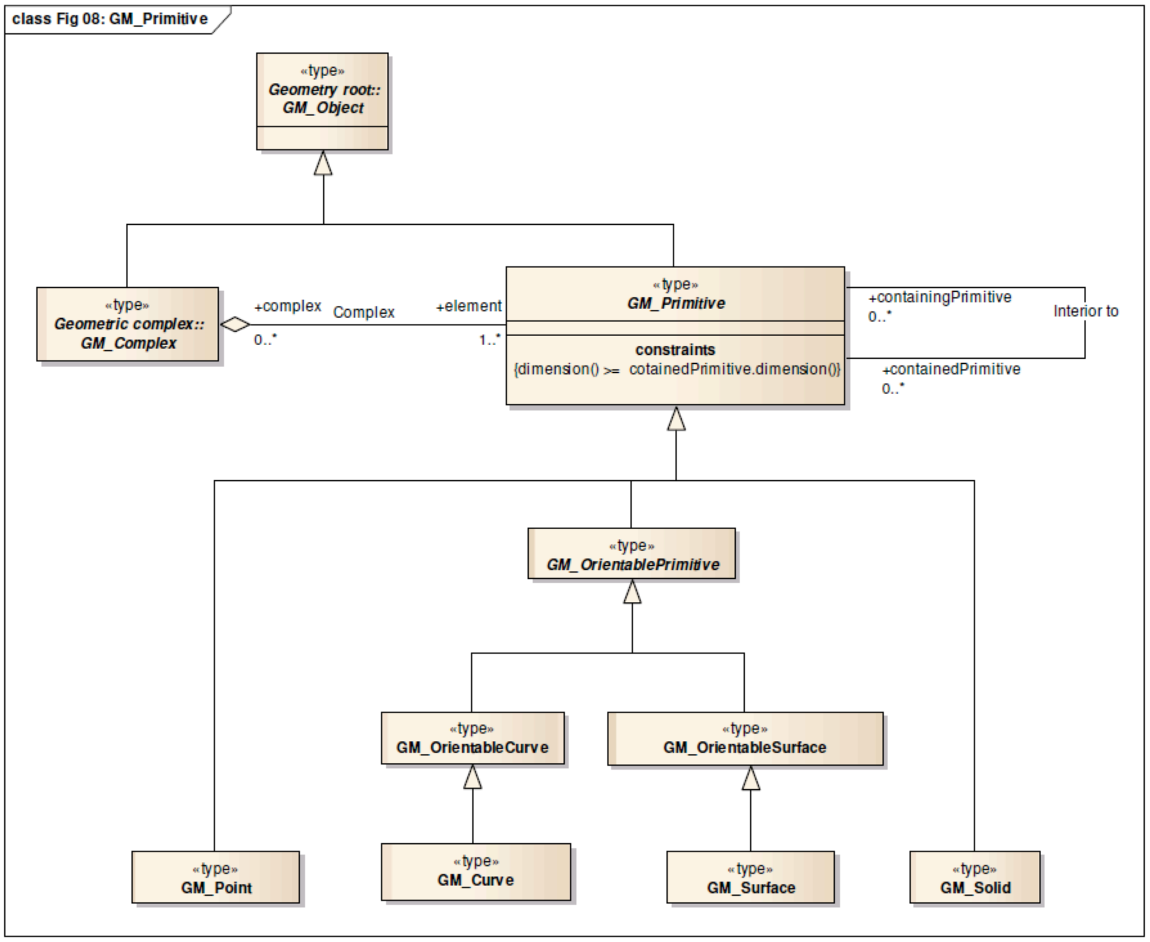


Figure 5 Primitive geometry types supported within the FIXM GML model (ISO 19107)

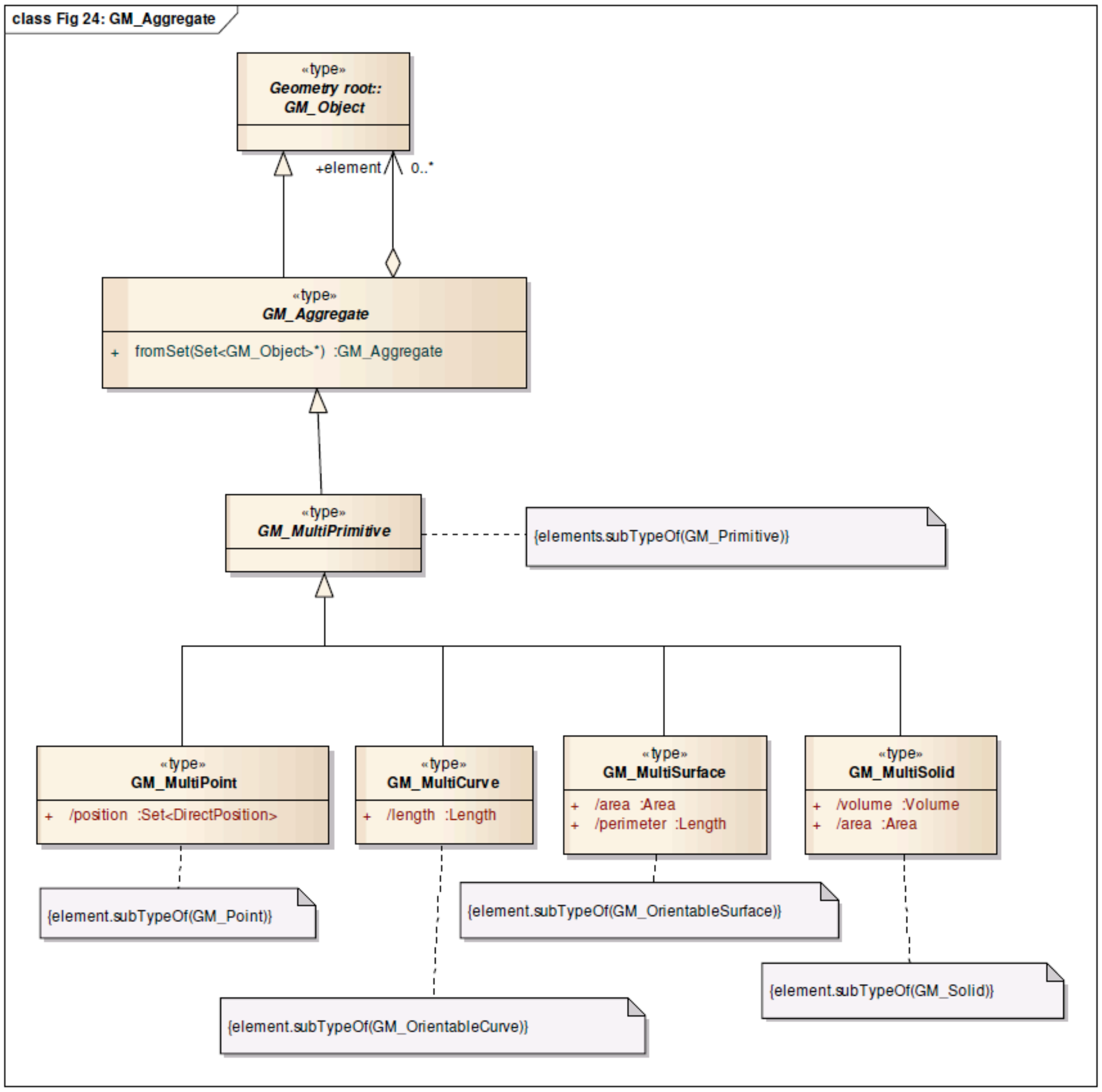

class Fig 24: GM_Aggregate
«type»
Geometry root::
GM_Object
+element
0..*
«type»
GM_Aggregate
+ fromSet(Set<GM_Object>*) :GM_Aggregate
«type»
GM_MultiPrimitive
{elements.subTypeOf(GM_Primitive)}
«type»
GM_MultiPoint
+ /position :Set<DirectPosition>
«type»
GM_MultiCurve
+ /length :Length
«type»
GM_MultiSurface
+ /area :Area
+ /perimeter :Length
«type»
GM_MultiSolid
+ /volume :Volume
+ /area :Area
{element.subTypeOf(GM_Point)}
{element.subTypeOf(GM_OrientableSurface)}
{element.subTypeOf(GM_Solid)}
{element.subTypeOf(GM_OrientableCurve)}

Figure 6 Aggregate geometry types supported within the FIXM GML model (ISO 19107)

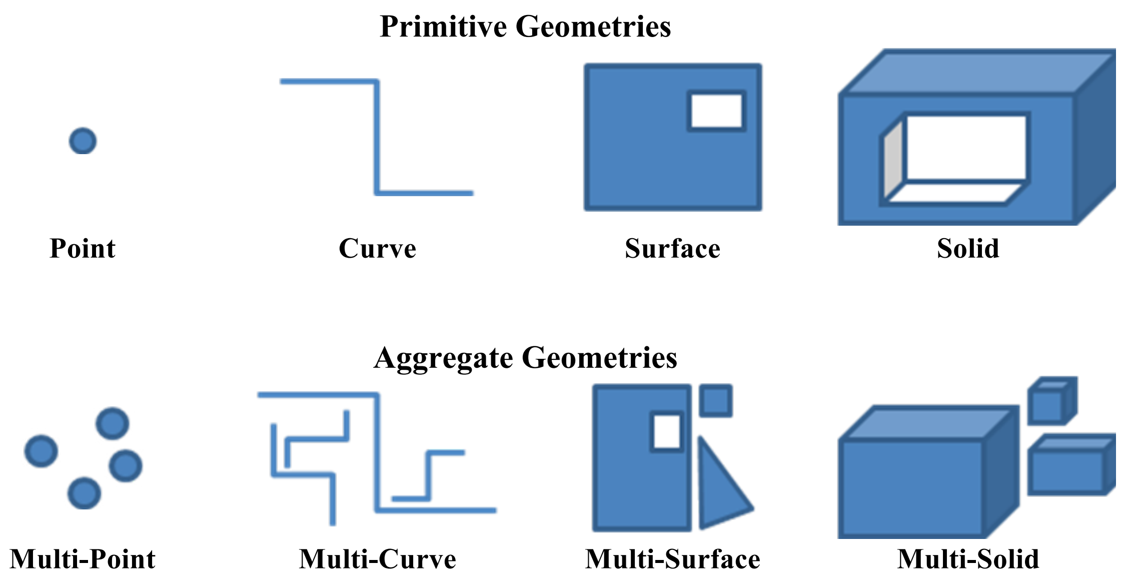


Figure 7 Illustrated examples of geometry types supported in the FIXM GML model (ISO 19107)

5.2.1.3 Coordinate Reference Systems Overview

ISO 19111 (Geographic Information – Spatial Referencing by Coordinates) provides a model to allow spatial and spatial temporal coordinate reference systems to be described. It describes the minimum data required to define one-, two- and three-dimensional spatial coordinate reference systems with an extension to merged spatial-temporal reference systems. It allows additional descriptive information to be provided. It also describes the information required to change coordinates from one coordinate reference system to another.

This model should be used as the reference model for describing local coordinate reference systems. The ISO 19111 model is used within many existing coordinate reference system registries such as the EPSG Registry3. For well-defined national or global coordinate reference systems, a geographic identifier (URN or HTTP URI) should be used from an existing registry. For example, the geographic identifier for WGS84 is either:

- **URN:** urn:ogc:def:crs:EPSG::4326
- **HTTP URI:** http://www.opengis.net/def/crs/EPSG/0/4326

5.2.2 Temporal Properties

There are two options for defining the type of a temporal property: (1) Basic Date/Time types defined within ISO 19103 or (2) Temporal geometry types defined in ISO 19108. Within ISO 19108 two core types of temporal Class are defined: Temporal Geometric Primitive and Temporal Topological Primitive (Figure 8).

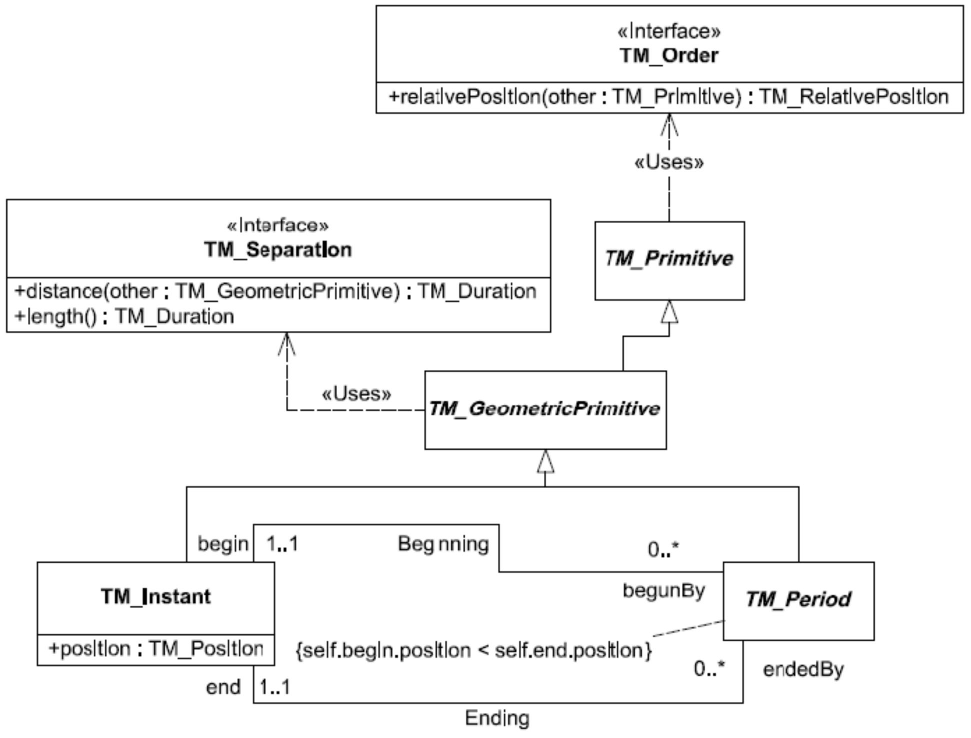


Figure 8 Temporal Geometry Primitive types (ISO 19108)

Table 8 Temporal Properties (ISO 19108)

Name	Description
TM_Period	TM _Period represents an extent in time defined using begin and end points (instants) and/or a duration.
TM_Instant	TM_Instant represents a single point in time.

The benefits of using TM_Period and TM_Instant rather than the Basic Temporal Types is that they allow the data provider greater flexibility in how the value is encoded using the TM_Position data type (Figure 9). The TM_Position data type is a <<Union>> so is a choice of either: Date, DateTime, Time or TM_TemporalPosition. Where the TM_TemporalPosition allows the definition time as a relative value:

- **unknown** - the date/time value is not currently known or computable. This is typically used to encode an end date for features that are still in use and there is no known end date.
- **now** - the specified value shall be replaced by the current date/time whenever the value is accessed.
- **before -** the actual date/time value is unknown, but it is known to be before the specified value.
- **after -** the actual date/time value is unknown, but it is known to be after the specified value.

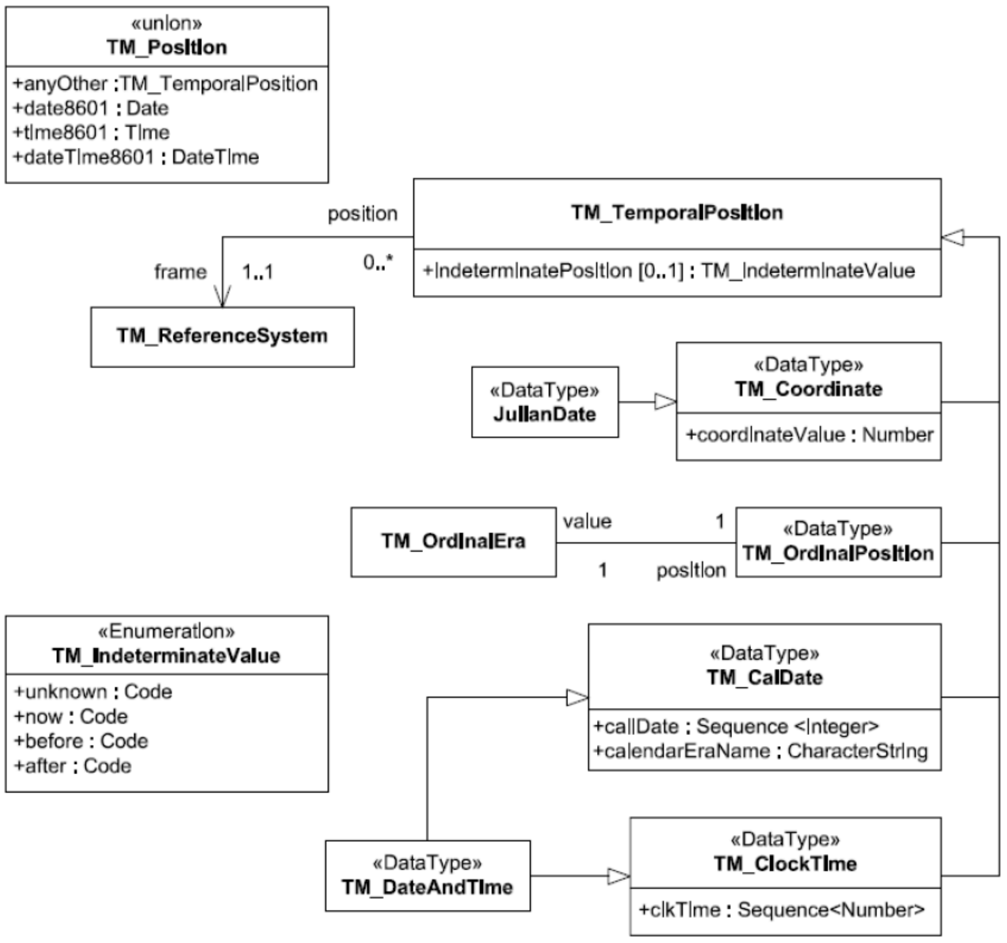


Figure 9 Data types for temporal position

5.2.3 Basic Types

5.2.3.1 Text

The following base text data types shall be supported (Table 9).

Table 9 Text data types (ISO 19103)

Name	Description
CharacterString	A CharacterString is an arbitrary-length sequence of characters including accents and special characters from repertoire of one of the adopted character sets.
Boolean	A value representing binary logic. The value can either be true or false.

5.2.3.1.1 Example

A Text Basic Type implementation example is shown in Figure 10.

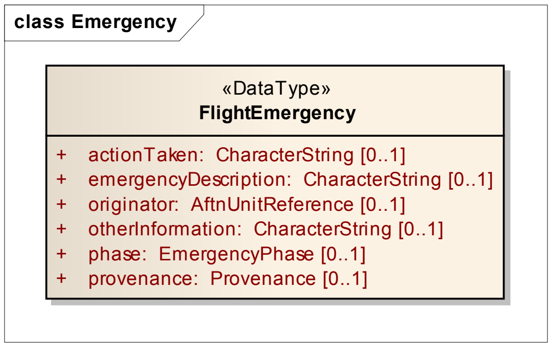

class Emergency
«DataType»
FlightEmergency
+ actionTaken: CharacterString [0..1]
+ emergencyDescription: CharacterString [0..1]
+ originator: AftnUnitReference [0..1]
+ otherInformation: CharacterString [0..1]
+ phase: EmergencyPhase [0..1]
+ provenance: Provenance [0..1]

Figure 10 ISO 19103 Test Basic Type implementation example

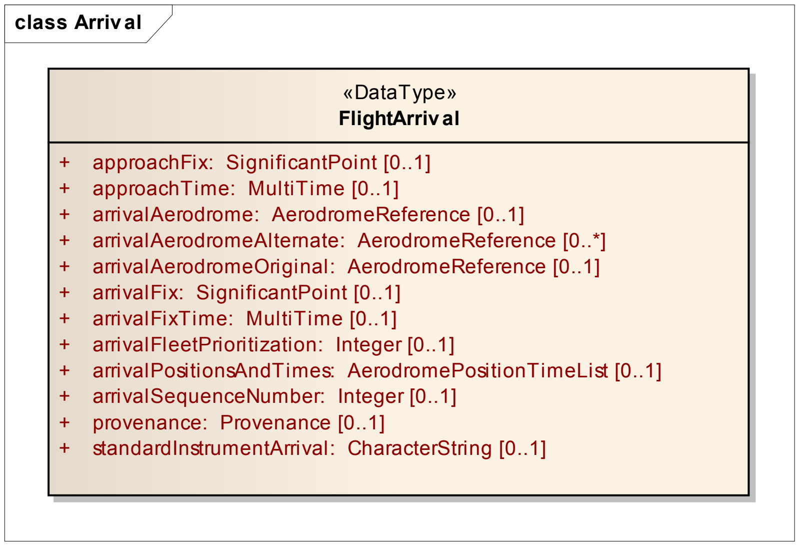
5.2.3.2 Numerics

The following base numeric data types shall be supported (Table 10).

Table 10 Numeric data types (ISO 19103)

Name	Description
Integer	A signed integer number, the representation of an integer is encapsulation and usage dependant. Example: 12, -6500
PositiveInteger	An unsigned integer number greater than 0.
NonNegativeInteger	An unsigned integer number greater than or equal to 0.
Real	A signed real (floating point) number consisting of a mantissa and an exponent, the representation of a real is encapsulation and usage dependent. Example: 23.56, -1.23

5.2.3.2.1.1 Example


class Arrival
«DataType»
FlightArrival
+ approachFix: SignificantPoint [0..1]
+ approachTime: MultiTime [0..1]
+ arrivalAerodrome: AerodromeReference [0..1]
+ arrivalAerodromeAlternate: AerodromeReference [0..*]
+ arrivalAerodromeOriginal: AerodromeReference [0..1]
+ arrivalFix: SignificantPoint [0..1]
+ arrivalFixTime: MultiTime [0..1]
+ arrivalFleetPrioritization: Integer [0..1]
+ arrivalPositionsAndTimes: AerodromePositionTimeList [0..1]
+ arrivalSequenceNumber: Integer [0..1]
+ provenance: Provenance [0..1]
+ standardInstrumentArrival: CharacterString [0..1]

5.2.4 Date/Time Properties

The following base temporal data types shall be supported. These are required as they are used within the temporal <<Union>> class TM_Position:

Table 11 Date/Time data types (ISO 19103)

Name	Description
Date	A date gives values for year, month and day according to the Gregorian Calendar. Character encoding of a date is a string which shall follow the calendar date format (complete representation, basic format) for date specified by ISO 8601. Example: 2013-09-19 (YYYY-MM-DD)
Time	A time is given by an hour, minute and second. Character encoding of a time is a string that follows the local time (complete representation, basic format) format defined by ISO 8601. Time zone according to UTC is optional. Example: 18:30:59 or 18:30:59+01:00 or 18:30:59Z
DateTime	A DateTime is a combination of a date and a time type. Character encoding of a DateTime shall follow ISO 8601. Example: 1998-04-12T10:11:40

6 FIXM 2.0 UML Model Structure Modification

FIXM 2.0 UML model structure modification was required in order for ISO19136 and ISO 19109 compliance. The requiredmodifications are documented below.

Note - modification of the UML model structure is also required for the addition of XML attributes in the outputted XSDs (see Annex A).

6.1 Creating UML Attributes

Where UML classes did not contain an attribute but were required to contain a value, UML attributes were created. For example, Figure 11 shows the Foundation::Time class which displays a generalization from the type xs:dateTime, however no attribute exists to store a value of xs:dateTime. In each instance where this occurred an attribute prefixed with 'value' was created. The previous link to xs:dateTime was removed. In this example the type was also modified to TM_Position to conform to ISO 19136 (Figure 12).

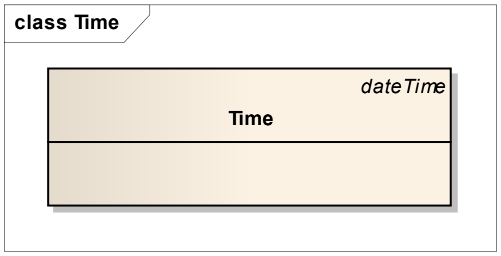


Figure 11 The Foundation::Time class from FIXM 2.0 UML model

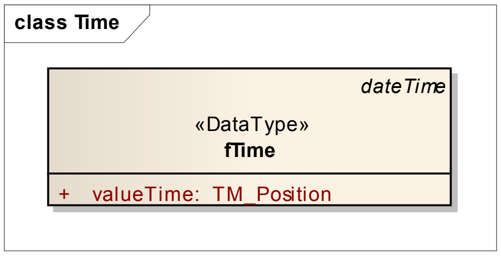


Figure 12 The Foundation::Time class from the modified FIXM 2.0 UML model

7 XSD Generation using ShapeChange

The ISO 19136 compliant FIXM 2.0 UML was exported as XMI 1.0 from Enterprise Architect. ShapeChange UGAS tool was used to transform the XMI 1.0 to XSDs using the following configuration.

7.1 ShapeChange Configuration

ShapeChange is a powerful tool for generating GML application schemas (XSDs) from UML models as it is highly configurable.

7.1.1 Overview

The ShapeChange version (2.0) used to transform the modified FIXM 2.0 UML to implementation schema is available on http://shapechange.net/.

The ShapeChange configuration used in the conversion has the parameters listed in this sub-clause in addition to the standard aliases, map entries, XML namespaces and encoding rules.

7.1.2 General Parameters

7.1.2.1 <input> element

The <input> element defines the source of the model and some parameters controlling its interpretation[4] (Table 12).

Table 12 ShapeChange <input> element parameters

ShapeChange Parameter	Value	Description
inputModelType	XMI10	XMI 1.0 as exported from Enterprise Architect.
inputFile	./FIXM/FIXM_GML.xmi	Relative file pathway to XMI.
publicOnly	true	Only converts public elements (standard behavior).
checkingConstraints	disabled	OCL constraints in the model will not be validated.
sortedSchemaOutput	true	The classes in each application schema will be processed in ascending order of the class names.

[4] http://shapechange.net/get-started/config/input/

7.1.2.2 <targets> element

The <targets> element contains the configurations of each target format, such as XML schema. All target definitions are nested under the <targets> element. This element does not contain <parameters>, instead it acts as a top-level element under which <TargetXmlSchema> and <Target> definitions are nested.

7.1.2.2.1 <TargetXmlSchema>

The XML Schema target in ShapeChange, as the main and most configurable target, is the only to be represented by a unique target element <TargetXmlSchema>. This is a variant of a standard <Target> element except that the class attribute is fixed to *de.interactive_instruments.ShapeChange.Target.XmlSchema.XmlSchema*.

The <TargetXmlSchema> element contains: (1) <targetParameter> definitions, (2) several <rules> elements containing <EncodingRule> definitions, (3) <xsdMapEntries> element containing <XsdMapEntry> definitions, and (4) XInclude directives.

7.1.2.2.1.1 <targetParameters>

The general <targetParameter> definitions for <TargetXmlSchema> are as follows. Additional parameters are specified by some conversion rules and are documented in the description of these extensions.

Table 13 ShapeChange <targetParameter> configuration overview

<targetParameter> Attribute	Value	Description
outputDirectory	FIXM/Modified	Path to which the XSDs will be written.
sortedOutput	true	A Boolean, if “true” classes within the package will be sorted alphabetically prior to output to the XSD.
defaultEncodingRule	iso19136_2007	Specifies the encoding rule responsible for governing the conversion from UML to XSD.

7.1.2.2.1.2 <xsdMapEntries> and <XsdMapEntry> element(s)

The <xsdMapEntries> element may contain individual <XsdMapEntry> elements, which may represent additional mappings from UML (classes) to GML elements, types and attributes. The definition of an <XsdMapEntry> is a narrowing of the <MapEntry> concept.

Table 14 Shapechange <XsdMapEntry> configuration overview

<XsdMapEntry> Attributes	Value	Description
type	URI	The UML class name of the type.
xsdEncodingRules	iso19136_2007	The XSD encoding rules to which this mapping applies.
xmlPropetryType	anyURI	The type names of the XSD type to be used in a property element if the value of the property is in the UML class.
xmlType	anyURI	The global XML type that represents the XML content model of the UML class.
xmlTypeType	simple	Identifies if the xmlType is 'simple' or 'complex'.
xmlContentType	simple	Identifies if the content of the xmlType is 'simple' or 'complex'.

See Annex B for the full ShapeChange configuration.

8 Overview of XSDs Generated

The FIXM 2.0 folder structure was maintained within the FIXM 2.0 GML schema. All XSDs were regenerated as ISO 19136 compliant FIXM schema.

During FIXM 2.0 modification, element nesting reductions lead to removal of elements particularly from the Foundation package. For example, Foundation::Aerodrome.xsd and Foundation::Airspace.xsd can be removed without subsequent impact on schema validity.

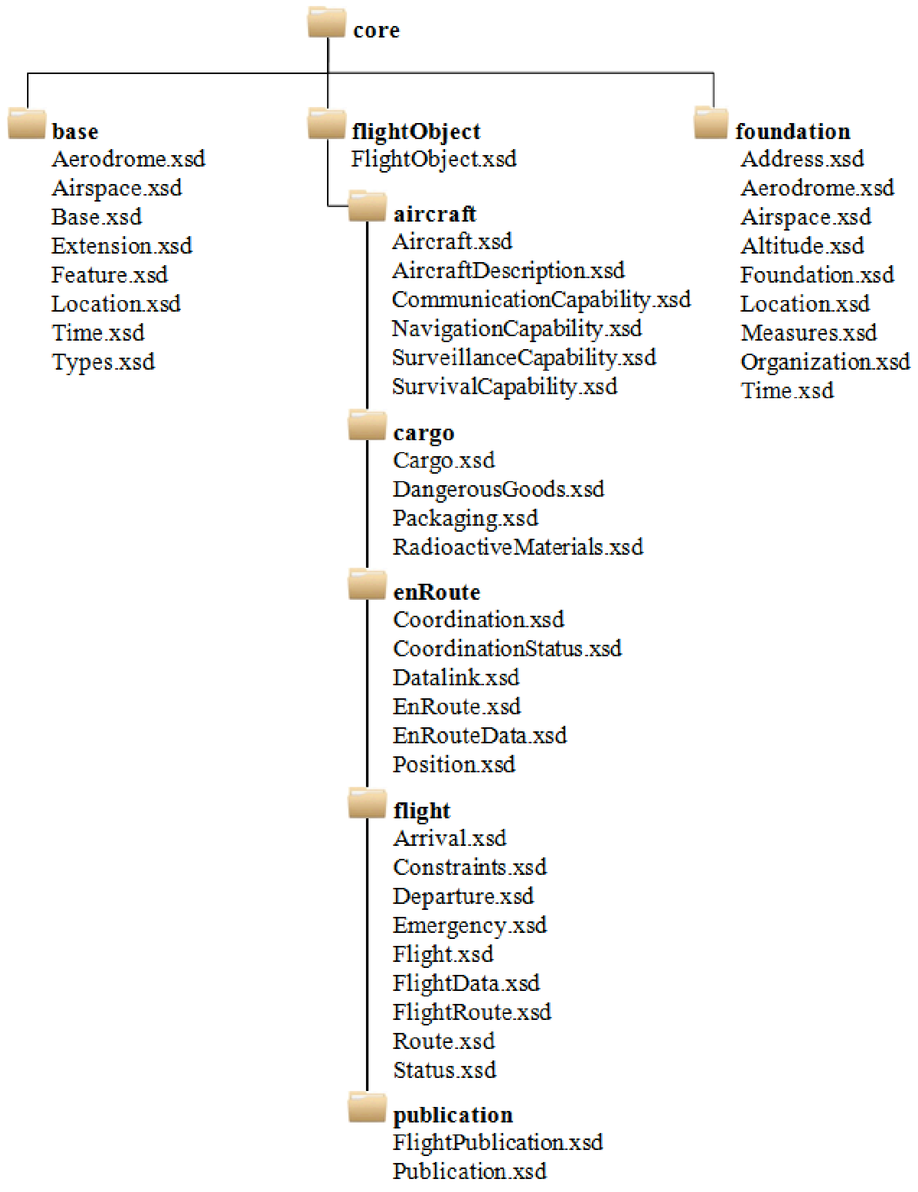


Figure 13 XSD Folder structure maintained in FIXM 2.0 GML

9 Notable Issues

Issues encountered throughout the FIXM 2.0 GML development and implementation process are described below.

9.1 XML Attribute Removal

During implementation services with varying capabilities were required to support the FIXM 2.0 GML encoding. As a result, vendor specific constraints disabled certain aspects of the FIXM 2.0 GML prototyping effort.

Notably, the decision to exclude XML attributes from FIXM 2.0 GML was undertaken for several reasons:

(1) Vendor specific constraints, in particular failure to support XML attributes via a WFS.

(2) Strictly adhering to UML to XSD encoding rules means that XML attributes should not be included in the FIXM GML schema.

Initial drafts of the FIXM GML schema included XML attributes. These were later removed following realization of vendor specific constraints. A description of how these XML attribute were implemented can be found in Annex A.

Of the two XML attribute removal driving factors above, the application of non-standard UML to XSD encoding rules was a less important instigator of XML attribute removal. The experimental nature of a testbed does not dictate such strict adherence to encoding rules; instead a testbed is meant to challenge existing and novel standards and technologies. The use of XML attributes, although non-standard practice, has also been implemented in other ATM exchange models, including AIXM and IWXXM.

9.2 Other Notable Issues

Table 15 contains a summary of some of the issues identified when modifying FIXM 2.0 to attain ISO 19136 compliance.

Table 15 Issues encountered during FIXM 2.0 UML modification and GML compliant XSD generation

Issue Number	Issue Encountered	Description
1	Conflicting Attribute and Class Names	Numerous class names used in the FIXM 2.0 UML model conflict with GML types. Despite different namespaces being present, ShapeChange recognizes the FIXM classes that have names the same as GML types, as GML. For example, the FIXM class names distance, volume and length conflict with GML types. To resolve this issue the FIXM class was prefixed with 'FIXM'.
2	Encoding Efficiency	FIXM 2.0 was developed without GML in mind. The inherent structure of FIXM 2.0 UML resulted in bloating caused by deep element nesting when converted to adhere to ISO 19136. Most of the newly introduced object levels are part of the larger GML object-property model and will not be encoded on their own or referenced by xlinks. To reduce this nesting, implementation of the GML object-property model only where necessary, for example where geometries are present, may be beneficial. Compression technologies may also be used to compensate for data bloating issues.
3	Non-Standard Encoding of XML Attributes	The CASE and UGAS tools used possess capabilities to implement XML attributes and this was done so for the initial drafts of the FIXM 2.0 GML schemas. However, vendor specific constraints coupled with strict adherence of the UML to XSD encoding rules meant XML attributes were removed from the final FIXM 2.0 GML encoding.
5	XSD Redundancy	In the original FIXM 2.0 schema, where classes extended basic types (such as 'string', 'int', 'decimal') and were referenced by another class. This middle basic type reference classes were removed. Having removed multiple classes from packages, several packages have becomes redundant, for example Foundation::Airspace and Foundation::Aerodrome. These packages are now empty and serve no purpose in the FIXM 2.0 GML XSDs.

Issue Number	Issue Encountered	Description
6	Empty / Binary Enumerations	Multiple FIXM enumeration classes contain less than 2 attributes. Where an enumeration class contains zero attributes, values should be assigned accordingly. Where one or two enumeration attributes are present the class stereotype may be altered from an Enumeration to a Boolean.
7	Enumerations vs. CodeLists	Clarification is required to whether the classes specified as enumerations in FIXM 2.0 are complete, or whether the class stereotype CodeList should be used instead.
8	FIXM 2.0 GML Leniency	The FIXM 2.0 GML application schemas are more lenient than the FIXM 2.0 application schema. Where FIXM data do not conform to the FIXM 2.0 application schema constraints (for example different versions of GUFI are used between data providers) or where the constraints within FIXM 2.0 are incorrect, this leniency is beneficial. However, this leniency also permits the validation of potentially invalid data.

10 Conclusions

- FIXM 2.0 can be migrated to ISO and OGC standard whilst maintaining a similar structure.

- Vendor specific restrictions did not permit the use of XML attributes in this testbed – however these are possible using non-standard encoding rules.

- Further optimizations are possible to streamline FIXM 2.0 GML encoding.

- Demonstrating FIXM implementation can follow the same best practice as AIXM and WXXM by adopting ISO and OGC standards produces the following benefits:

    - Dramatic reduction in both cost and time for implementing FIXM.

    - Better aligns FIXM to AIXM and WXXM allowing them to interoperate and reference one another.

    - This interoperability and openness enables System Wide Information Management (SWIM).

10.1 Key Accomplishments

- Benefits of GML-based FIXM.
    - Snowflake has demonstrated that by applying ISO 19109 principles to FIXM, OGC service standards and configurable off the shelf software (from multiple vendors) is able to support FIXM out of the box. The key benefit being a dramatic reduction in both cost and time for implementing FIXM, furthermore, it better aligns FIXM to AIXM and WXXM allowing them to interoperate and reference one another.
- Demonstrated the implementation of an ISO 19136 compliant FIXM 2.0.
- Recognition and suggestions of areas of improvement within the FIXM 2.0 model.

Future Work

- Further rework to simplify FIXM 2.0 UML to increase encoding efficiency. This is required as FIXM 2.0 was designed without consideration for GML inclusion; therefore data structures are not optimized.
- Performance testing to test performance changes resulting from non-standard encoding removal (XML attributes).
- Address issues raised in Section 9 and amend the FIXM 2.0 model accordingly.

10.1.1 Future FIXM Improvements

Suggested FIXM improvements resulting from this work are presented below:

- Implement a means by which the schemas may include custom extensions
- Version control and change only update mechanism
- Rationalise and define interoperability rules for the use of temporality with AIXM and IWXXM.

Annex A

Inserting XML Attributes

The term ‘attribute’ in UML is not interchangeable with the same term in GML. For a UML attribute to appear as a XML Attribute in the XML Schema, the UML attribute must be assigned tagged values. It is non-standard to include XML attributes whilst converting from UML to XSD. The use of XML attributes is possible by assigning attribute tagged values and using a non-standard encoding rule in ShapeChange.

Tagged values on UML attributes were only specified where it was necessary for the ShapeChange UGAS tool to identify UML attributes to be converted to XML attributes. A non-standard encoding rule is used to generate XML attributes (Table 16). UML to XML encoding rules stipulate encoding of XML attributes is prohibited.

Table 16 UML attribute tagged values for output to XML attributes

Tag	Value
xsdEncodingRule	iso_19136_2007_ShapeChange_1.0_Extensions
xsdAsAttribute	true

For XML attribute creation, it was also necessary to modify the UML structure. The process of adding and modifying existing UML attributes in order to create XML attributes is documented below.

Each UML attribute to be output as an XML attribute must of a simple type. In order to satisfy this requirement each attribute was made a simple type where applicable through modification of inheritance and type. For example, where classes exhibited generalizations to FreeText, a UML attribute was created on the class of type CharacterString (extending ISO 19103); see Figure 14.

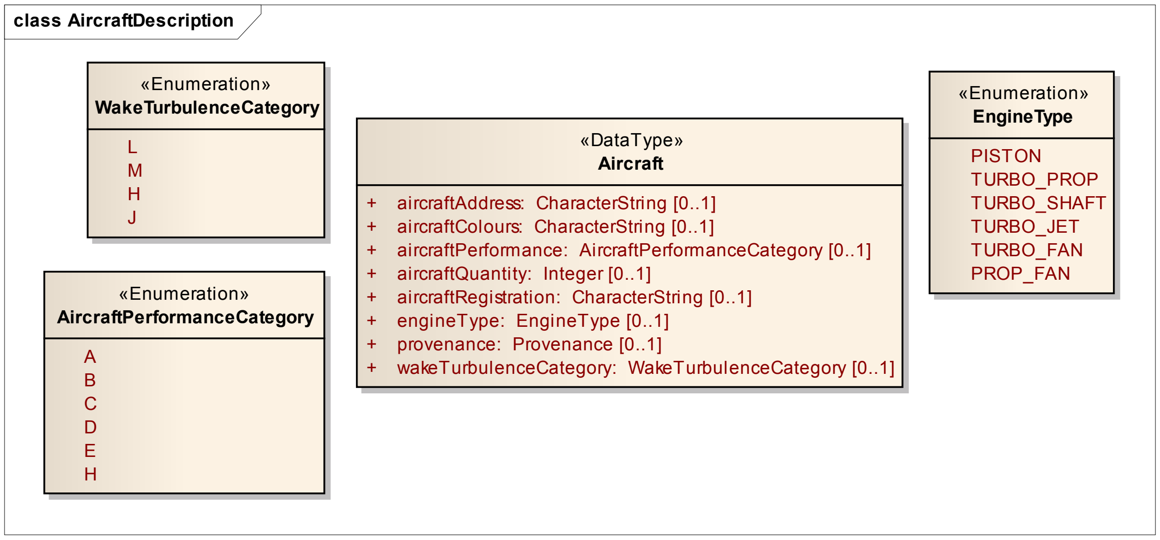

class AircraftDescription
«Enumeration»
WakeTurbulenceCategory
L
M
H
J
«Enumeration»
AircraftPerformanceCategory
A
B
C
D
E
H
«DataType»
Aircraft
+ aircraftAddress: CharacterString [0..1]
+ aircraftColours: CharacterString [0..1]
+ aircraftPerformance: AircraftPerformanceCategory [0..1]
+ aircraftQuantity: Integer [0..1]
+ aircraftRegistration: CharacterString [0..1]
+ engineType: EngineType [0..1]
+ provenance: Provenance [0..1]
+ wakeTurbulenceCategory: WakeTurbulenceCategory [0..1]
«Enumeration»
EngineType
PISTON
TURBO_PROP
TURBO_SHAFT
TURBO_JET
TURBO_FAN
PROP_FAN

Figure 14 Modified UML representation of the Aircraft element illustrating the process of XML Attribute specification

Having modified the UML model so that all UML attributes that are to be translated to XML Attributes are of simple type, the attributes must also be assigned Tagged Values as above (Table 16). When run, ShapeChange uses this configuration to output each UML attribute that satisfies the above requirements as an XML attribute.

Annex B

ShapeChange Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<ShapeChangeConfiguration xmlns:xi="http://www.w3.org/2001/XInclude"
xmlns="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1"
xmlns:sc="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.interactive-instruments.de/ShapeChange/Configuration/1.1
http://shapechange.net/resources/schema/ShapeChangeConfiguration.xsd">
    <input>
        <parameter name="inputModelType" value="XMI10"/>
        <parameter name="inputFile" value="./FIXM/FIXM_remodelled.xmi"/>
        <parameter name="publicOnly" value="true"/>
        <parameter name="checkingConstraints" value="disabled"/>
        <parameter name="sortedSchemaOutput" value="true"/>
        <xi:include href="./config/StandardAliases.xml"/>
    </input>
    <log>
        <parameter name="reportLevel" value="ERROR"/>
        <parameter name="logFile" value="FIXM/xmi/log.xml"/>
    </log>
    <targets>
        <TargetXmlSchema
class="de.interactive_instruments.ShapeChange.Target.XmlSchema.XmlSchema"
mode="enabled">
            <targetParameter name="outputDirectory" value="FIXM/Mod_Test"/>
            <targetParameter name="sortedOutput" value="true"/>
            <targetParameter name="defaultEncodingRule" value="iso19136_2007"/>
            <xi:include href="http://shapechange.net/resources/config/StandardRules.xml"/>
            <xi:include
href="http://shapechange.net/resources/config/StandardNamespaces.xml"/>
            <xi:include
href="http://shapechange.net/resources/config/StandardMapEntries.xml"/>
            <xsdMapEntries>
                <XsdMapEntry type="URI" xsdEncodingRules="iso19136_2007"
xmlPropertyType="anyURI" xmlType="anyURI" xmlTypeType="simple"
xmlTypeContent="simple"/>
            </xsdMapEntries>
        </TargetXmlSchema>
    </targets>
</ShapeChangeConfiguration>
```