Open Geospatial Consortium

Approval Date: 2013-01-18

Pending Date: 2012-12-21

Publication Date: 2013-2-05

Reference number of this document: OGC 12-094

Reference URN for this document: http://www.opengis.net/def/doc-type/per/ows9-aviation-airm

Category: Engineering Report

Editors: Debbie Wilson, Clemens Portele

OGC® OWS-9 Aviation: AIRM Derivation

Warning

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements.

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

Abstract

This report describes the architecture, rules and tools developed within the OWS-9 Aviation Thread AIRM Derivation task. These rules and tools were demonstrated by transforming the AIRM Meteorology package into a Weather Exchange Model (WXXM) and GML/JSON implementation schema. (See also executive summary)

Keywords

ows9, ows-9, airm, aviation, wxxm, gml, json

What is OWS-9?

OWS-9 builds on the outcomes of prior OGC initiatives and is organized around the following threads:

- **Aviation**: Develop and demonstrate the use of the Aeronautical Information Exchange Model (AIXM) and the Weather Exchange Model (WXXM) in an OGC Web Services environment, focusing on support for several Single European Sky ATM Research (SESAR) project requirements as well as FAA (US Federal Aviation Administration) Aeronautical Information Management (AIM) and Aircraft Access to SWIM (System Wide Information Management) (AAtS) requirements.

- **Cross-Community Interoperability (CCI)**: Build on the CCI work accomplished in OWS–8 by increasing interoperability within communities sharing geospatial data, focusing on semantic mediation, query results delivery, data provenance and quality and Single Point of Entry Global Gazetteer.

- **Security and Services Interoperability (SSI)**: Investigate 5 main activities: Security Management, OGC Geography Markup Language (GML) Encoding Standard Application Schema UGAS (UML to GML Application Schema) Updates, Web Services Façade, Reference Architecture Profiling, and Bulk Data Transfer.

- **OWS Innovations**: Explore topics that represent either new areas of work for the Consortium (such as GPS and Mobile Applications), a desire for new approaches to existing technologies to solve new challenges (such as the OGC Web Coverage Service (WCS) work), or some combination of the two.

- **Compliance & Interoperability Testing & Evaluation (CITE)**: Develop a suite of compliance test scripts for testing and validation of products with interfaces implementing the following OGC standards: Web Map Service (WMS) 1.3 Interface Standard, Web Feature Service (WFS) 2.0 Interface Standard, Geography Markup Language (GML) 3.2.1 Encoding Standard, OWS Context 1.0 (candidate encoding standard), Sensor Web Enablement (SWE) standards, Web Coverage Service for Earth Observation (WCS-EO) 1.0 Interface Standard, and TEAM (Test, Evaluation, And

Measurement) Engine Capabilities.

The OWS-9 sponsors are: AGC (Army Geospatial Center, US Army Corps of Engineers), CREAF-GeoViQua-EC, EUROCONTROL, FAA (US Federal Aviation Administration), GeoConnections - Natural Resources Canada, Lockheed Martin Corporation, NASA (US National Aeronautics and Space Administration), NGA (US National Geospatial-Intelligence Agency), USGS (US Geological Survey), UK DSTL (UK MoD Defence Science and Technology Laboratory).

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable

Executive Summary

The ATM Information Reference Model (AIRM) is a consolidated logical UML model representing the civil, military and civil-military information constructs relevant to ATM. that is currently being developed within SESAR. While the AIRM defines the consolidated logical data model, physical implementation models are required for data exchange.

The Model Driven Architecture (MDA) approach advocates that physical implementation models can be automatically derived from the logical UML model. Therefore, the AIRM Derivation task has developed a two stage approach for generating a logical ATM Exchange Models (UML) from which the physical implementation schemas (GML/JSON) shall be automatically derived.

The objectives of the AIRM Derivation work were to:

1. *Develop rules and tools for transforming AIRM into physical implementation models: GML and JSON*
2. *Demonstrate the rules and tools by transforming the AIRM Meteorology package into a Weather Exchange Model (WXXM)*

The key outcomes were:

1. Definition of transformation rules for generating a logical ATM exchange model (GML application schema) from the AIRM
2. Developed MDA transformation templates for Enterprise Architect
3. Defined additional mapping rules to be executed by ShapeChange to map AIRM data types into types that can be instantiated in GML/JSON
4. Successfully demonstrated end-to-end transformation of the AIRM Meteorology Package into WXXM and its associated GML/JSON implementation schemas
5. Identified several improvements required to the AIRM Foundation Rules to ensure that the generated ATM exchange models comply to ISO 19109/ISO 19136 rules for application schema

The key accomplishments were:

1. Developed a simple, re-usable process for transforming AIRM packages into an ATM exchange model.
2. Leveraged existing, industry standard software for generating implementation schemas ensuring that the resultant implementation schemas adhere to the ISO 19109/ISO 19136 rules and increase consistency.
3. Demonstrated that the tools are highly configurable enabling the transformation and mapping rules to be extended to meet requirements for different ATM exchange models without needing software development.

Contents

Page

Figures

Page

Tables

Page

OGC® OWS-9 Aviation: AIRM Derivation

1 Introduction

1.1 Background

The ATM Information Reference Model (AIRM) is a consolidated logical UML model that shall be used as a common reference for the different domain applications within Air Traffic Management that is currently being developed within SESAR[1]. The AIRM shall represent civil, military and civil-military information constructs relevant to ATM.

While the AIRM defines the consolidated logical data model, physical implementation models are required for data exchange. The Model Driven Architecture (MDA) approach advocates that physical implementation models (e.g. XML, JSON) can be automatically derived from the logical UML model.

SESAR Project 08.01.06 is responsible for the synchronization between the evolutions of the AIRM managed by SESAR and the WXXM developments undertaken jointly by the US FAA and EUROCONTROL. SESAR 08.01.06 conducted a preliminary study which demonstrated the feasibility to derive a "lean and mean" WXXM from the AIRM; this study delivered in particular an initial set of "AIRM to WXXM" mapping rules and a proof-of-concept tool, developed in Java and based on the Sparx EA Java API, able to realize a partial conversion of the AIRM into a WXXM-like model.

A complete set of "AIRM to ISO 19109 Application Schema" mapping rules and according implementation is not yet available. Such functionality would not only enable future derivations of the WXXM from the AIRM in the context of SESAR, but will also enable the creation of "fit for purpose" ISO 19109 Application Schemas for other ATM Domains, in particular those domains that do not yet have an internationally agreed "-XM" format (like AIXM or WXXM).

1.2 Scope

The objectives of the AIRM Derivation work were to:

3. *Develop rules and tools for transforming AIRM into physical implementation models: GML and JSON*
4. *Demonstrate the rules and tools by transforming the AIRM Meteorology package into a Weather Exchange Model (WXXM)*

The outcome of this work shall:

[1] http://www.sesarju.eu/

- Describe the corresponding configuration of the software framework
- Describe of the requirements that shall be satisfied by the input UML model
- Where required, deliver an adaptation of WXXM 1.1.3 that includes the additional information necessary to perform the derivation
- Identify and document “WXXM 1.1.3 to GML2.1 / JSON based physical model” derivation rules
- Provide an implementation of these rules
- Create a GML / JSON based physical model, validate and document it

Note: validation of the physical models through actual exchange of sample datasets via the OWS-9 infrastructure did not need to be realized. The generation of instance document examples is sufficient.

Document potential enhancements and issues identified during the realization of this requirement in change requests against relevant OGC documents, especially Standard and Best Practice documents.

The creation of the "UML to JSON" encoding rules is a cross-thread activity with the SSI Thread - see OGC 12-093: OWS-9 SSI UGAS Engineering Report for further details.

1.3 Document contributor contact points

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

Name	Organization
Debbie Wilson (Editor)	Snowflake Software Ltd
Clemens Portele	Interactive Instruments

1.4 Revision history

Date	Release	Editor	Primary clauses modified	Description
2012/07/16	0.1	D. Wilson, C. Portele	All	Initial document template and contents
2012/11/16	0.2	D. Wilson, C. Portele	All	Final Draft version
2012/12/21	0.3	D. Wilson, C. Portele	All	Final version

1.5 Future work

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

- Address the issues identified in 9.1. and 9.2
- Encode data using the implementation schemas.
- Explore the use of JSON for aviation data.
- If needed, standardize JSON Schema implementations for ISO 19100 types.
- Evaluate the ability of the tools to generate other ATM exchange models (e.g. AIXM, FIXM)
- Implement the MDA transformation improvements

1.6 Forward

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.

Geography Markup Language, Version 2.1.2, Open Geospatial Consortium (OGC)

Geography Markup Language, Version 3.2, Open Geospatial Consortium (OGC)

Geography Markup Language, Version 3.3, Open Geospatial Consortium (OGC)

ISO/TS 19103:2005, Geographic Information – Conceptual Schema Language

ISO 19109:2004, Geographic Information – Rules for Application Schema

ISO 19115:2003, Geographic information – Metadata

ISO 19115:2003/Corr:2006, Geographic information – Metadata – Technical Corrigendum 1

ISO 19156:2011, Geographic information -- Observations and measurements

ISO/IEC 19757-3:2006 Information technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation — Schematron

JSON Schema, IETF Draft 3, http://tools.ietf.org/html/draft-zyp-json-schema-03

OMG Object Constraint Language, Version 2.2, OMG Document Number formal/2010-02-01

OGC 11-091: 2011 OWS-8 Review of the WXXS exchange schemas

OGC 12-093: OWS-9 SSI UGAS Engineering Report

SESAR 08.01.03 D05: AIRM Primer

SESAR 08.01.03 D05: AIRM Foundation Rulebook

W3C XML Schema Part 1: Structures Second Edition. W3C Recommendation (28 October 2004)

W3C XML Schema Part 2: Datatypes Second Edition. W3C Recommendation (28 October 2004)

In addition to this document, this report includes several XML Schema Document files as specified in Annex A.

3 Conventions

AIRM Aeronautical Information Reference Model

AIXM Aeronautical Information Exchange Model

FIXM Flight Information Exchange Model

GML Geography Markup Language

ISO International Organization for Standardization

JSON JavaScript Object Notation

OCL Object Constraint Language

OGC Open Geospatial Consortium

OWS OGC Web Services

UML Unified Modeling Language

WXXM Weather Information Exchange Model

XML eXtended Markup Language

XPath XML Path Language

3.1 UML notation

Diagrams that appear in this standard are presented using the Unified Modeling Language (UML) static structure diagram, as described in ISO/TS 19103.

4 AIRM Derivation

4.1 Architecture

Transforming the AIRM subject field packages and their related data types and code lists into a physical implementation model shall be a two stage approach (Figure 1):

Stage 1. Generate a logical UML model representing the ATM exchange model from the AIRM that conforms to the ISO 19136 UML Profile for GML using MDA Transformations within Enterprise Architect

Stage 2. Transform the UML model into a physical implementation model using a UML to GML Application Schema (UGAS) tool

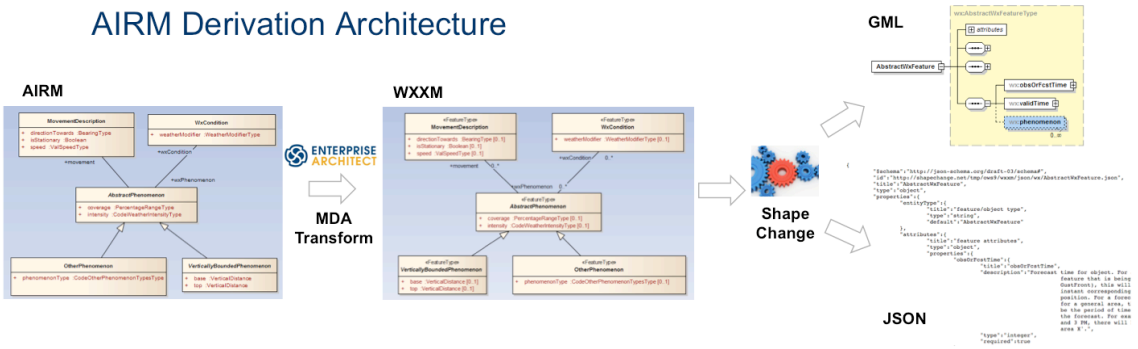


Figure 1. Architecture for transforming AIRM packages into implementation models (GML/JSON)

4.2 Transformation Tools

Two tools were selected for the AIRM Derivation:

- Enterprise Architect
- ShapeChange

4.2.1 Enterprise Architect

Enterprise Architect is a comprehensive UML analysis and design tool for modelling business, software and systems. Enterprise Architect was selected as the UML tool for developing the AIRM within SESAR. Enterprise Architect contains in built model

transformation capabilities that provide a configurable method for converting model elements and model fragments from one domain to another. This typically involves converting Platform-Independent Model (PIM) elements to Platform-Specific Model (PSM) elements (Figure 2).

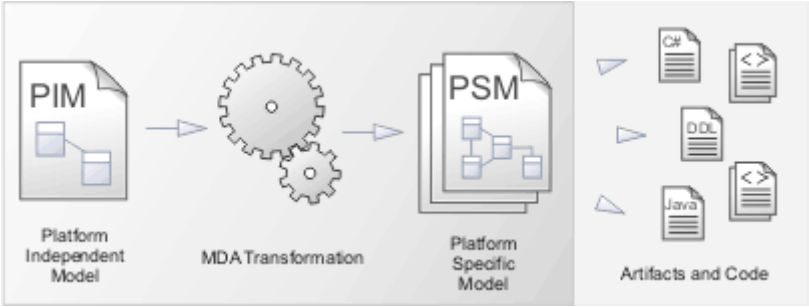


Figure 2. MDA Transformation within Enterprise Architect

This MDA Tranformation capability shall be used to generate a logical ATM exchange model based on the ISO 19136 UML Profile for GML which can then be transformed by the UGAS tool (ShapeChange) to generate the physical implementation models: GML and JSON.

4.2.2 ShapeChange

ShapeChange[2] is a Java tool that takes application schemas constructed according to ISO 19109 from a UML model and derives implementation representations. The most commonly used target representation is XML Schema (GML, ISO/TS 19139, SWE Common), however other implementation models are supported such as JSON Schema, RDF.

In addition to the generation of XML Schema documents, ShapeChange can automatically generate:

- **Schematron** documents that implement validation rules defined as OCL constraints in the UML model.
- **Feature catalogues** (ISO 19110) that describe the model in XML and HTML,
- **RDF** schemas,
- **Code list dictionaries** in GML and SKOS.

To transform the UML Model, ShapeChange can either directly access an Enterprise Architect model via the EA Java API or can read an export model (XMI 10).

[2] http://shapechange.net/. The source code of ShapeChange is available under the GNU General Public License.

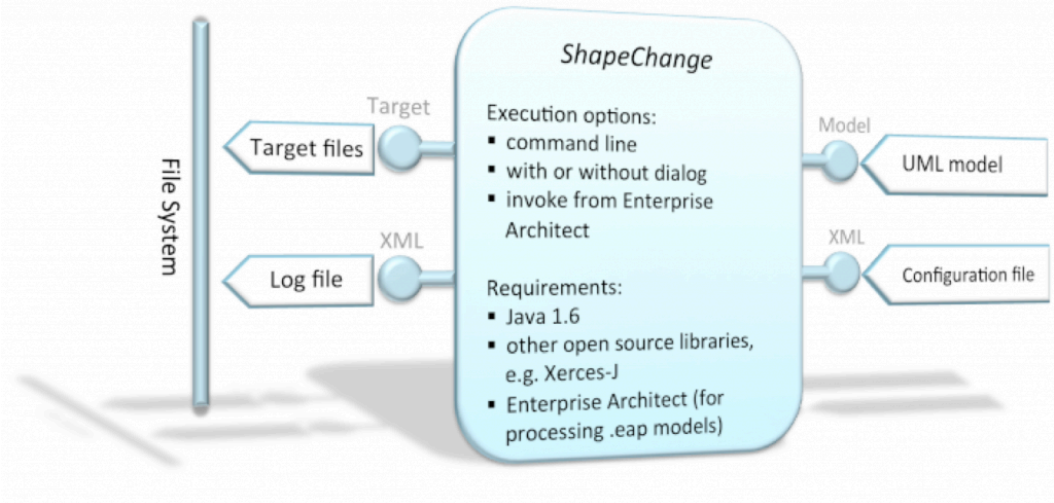


Figure 3. Overview of ShapeChange

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

- 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

These standardized encoding rules have been used for converting the transformed PSM exchange models into GML and JSON implementation models.

5 Transformation Rules

Transforming the AIRM is a two-stage process therefore two sets of transformation rules are required:

1. **MDA Transformation Rules:** these define how the AIRM UML model shall be converted into a logical model conforming to the ISO 19136 UML Profile
2. **Encoding Rules:** for transforming ISO TC 211 and AIRM types into GML and JSON types

5.1 MDA Transformation Rules

MDA transformation rules have been defined to convert the consolidated logical AIRM model into a logical model that implements the ISO 19136 UML Profile for GML application schemas. Therefore, the resulting UML model creates realizations of the AIRM types as the AIRM is at a different level of abstraction from a logical model developed using the ISO 19136 UML Profile.

To transform the AIRM into the ISO 19136 UML profile shall require two types of transformation:

- **Model constructs** (Packages, Classes, Attributes, Associations) to be assigned ISO 19136 UML Stereotypes and associated tagged values
- **Multiplicity transformation:** to apply the default AIRM Multiplicity Rules

The following sections shall describe the MDA Transformation Rules applied to transform the AIRM model constructs into ISO 19136 UML Profile model constructs.

NOTE: additional model transformations to transform AIRM types into types that are instantiated in GML or JSON were defined within ShapeChange (6.2 and 8.2).

5.1.1 Packages

Within the AIRM, the common and subject field packages differ in their complexity: some contain two or more sub packages, others no packages.

The ISO 19109 Rules for Application Schema require that a package can only contain one level of nested sub-packages. The parent package shall be assigned a stereotype <<Application Schema>> and the sub packages shall be assigned a stereotype of <<Leaf>>. NOTE: Leaf packages cannot contain sub packages.

The structure of the AIRM adds further complexity to the transformation as the exchange model may be comprised of model elements from the subject field packages and several data types packages. Therefore, the MDA Transformation must support the ability to include model elements from different packages in the AIRM.

Another issue encountered was understanding how to transform and re-use common data types, particularly those that have already been defined in an existing ATM exchange model (e.g. AIXM 5.1).

GOVERNANCE ISSUE: Common Types

There are types such as AIRM geometry types (Point, Curve, Surface) that shall be used in many ATM exchange models. There are two possible approaches for managing these common types within the ATM exchange models:

1. If an AIRM type has already been defined in an existing ATM Exchange Model (e.g. AIXM), develop encoding rules to re-use them.

2. If an AIRM type is used in multiple ATM Exchange Models, develop an AIRM exchange model that is imported by any domain specific ATM exchange model.

In the short to medium term option 1 is most appropriate, as this would require no major change to existing exchange models[3]. However, this approach has long-term governance constraints if external application domain requires the common type to be updated. This may not be possible if the maintenance of the two exchange models is not in sync.

Option 2 could be considered as a long-term approach for managing common types as these can have a more agile governance mechanism.

The following table summarises the MDA transformation rules that were used for transforming packages:

Rule	Description
Packages	
Rule 1	If the AIRM subject field is a simple package containing no sub packages then the transformed package shall be: • Contained within a generic UML package called GML Application Schema. • The transformed AIRM package shall be assigned the stereotype <<ApplicationSchema >> and the associated GML tagged values for that class stereotype.
Rule 2	If the AIRM subject field is a complex package containing sub packages: • The selected AIRM package shall form the parent package • This package shall be assigned the name GML Application Schema[4] and the stereotype: <<ApplicationSchema>> with the associated GML tagged values for that class stereotype • Each sub package shall retain its original name and be

[3] NOTE: Both options were tested in the development of the tools: Option 1 was adopted for geometry types already defined within AIXM, Option 2 was adopted for complex data types.

[4] There is a limitation in Enterprise Architect when using the packageName macro which sets the package name to the root package name: AIRM rather than the name of the subject field package. NOTE: There are advantages to assigning the package name to the default "GML Application Schema" for a complex AIRM subject field. If the exchange model must include type or code list classes from the common packages these can be added automatically within leaf packages.

	assigned the stereotype <<Leaf>> and the associated GML tagged values for that class stereotype.
Rule 3	Selected type and code list classes that are specific to the resulting ATM exchange model shall be added within <<Leaf>> packages.

5.1.2 Classes

The AIRM Foundation Rules defines the following rules for classes:

1. Only classes representing an enumeration shall be assigned a stereotype
2. The class name shall be used to distinguish between different classes:
    a. ***Features***: The class name shall have no prefix or suffix
    b. ***Code lists***: The class name shall be prefixed by "Code" and suffixed with "Type" (e.g. CodePrecipitationType)
    c. ***Types***: The class name shall be suffixed with "Type" (e.g. BearingType).

These rules could be used to develop an MDA transformation to add the correct stereotype and tagged values to classes, however due to the complex structuring of the AIRM, a simpler transformation approach was undertaken based on which package the classes were contained within.

The AIRM has been structured so that:

1. ***Subject Field*** packages only contain classes representing features
2. ***Data Types*** packages contain classes representing either data types or code lists.

Based on these rules MDA transformations were developed to add the correct stereotype and tagged values to classes based on which package they were contained within. The only exception was with developing an MDA transformation to programmatically assign the correct stereotype and tagged values to Type classes.

Within the ISO 19136 UML Profile, *type* classes can be divided into two types:

- **Data Type**: A structured data type without identity [ISO/TS 19103] and with no operations whose primary purpose is to hold the information [ISO 19136], or;
- **Type**: Structured data types with identify and may have may have zero or more operations (these are not mapped to the GML application schema), attributes or associations.

Most AIRM type classes are Data Types however, there are instances where the type classes should be assigned the Type stereotype. Based on the AIRM Foundation Rules, they should all be suffixed with "Type", but some of the Type classes do not follow this convention. For example: ElevatedPoint, Curve, Surface (Figure 4).

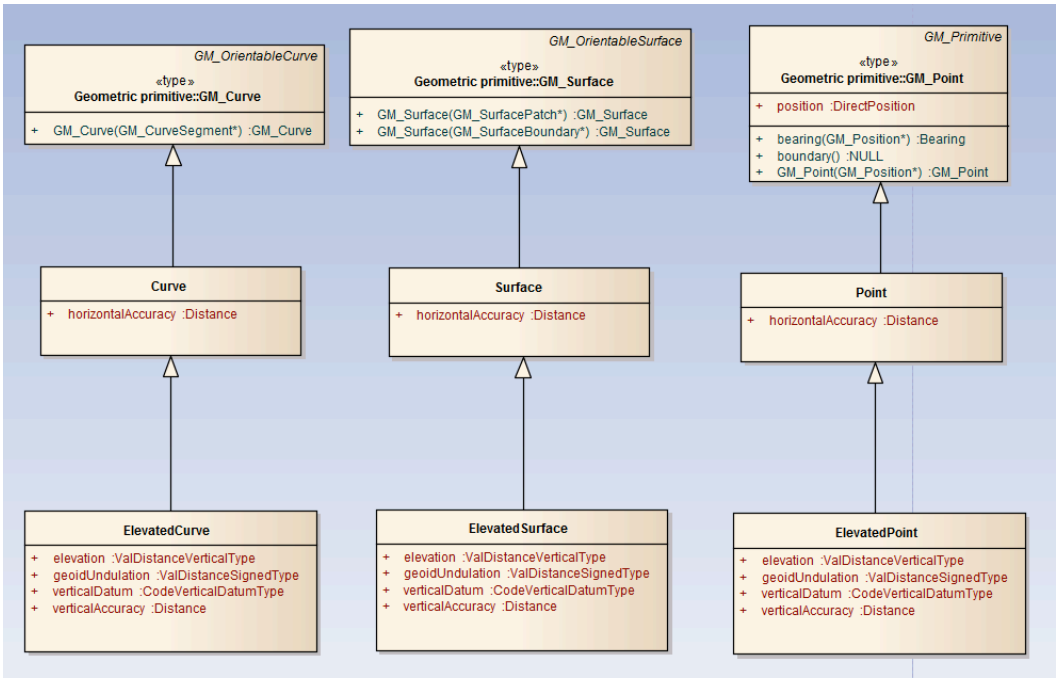


Figure 4. Example of structured data types that have identity

Therefore, the naming convention for structured data types is not reliable due to the AIRM still being in development. Due to the inability to programmatically distinguish between feature, type and data type classes in the AIRM, the MDA transformation tool has been designed to allow s the modeller to manually determine which classes should be transformed into which ISO 19136 class type.

Recommendation: Extend AIRM Foundation Rules to distinguish between structured data types with or without identity

The current version of the AIRM Foundation Rules does not sufficiently distinguish between the two types of structured data type. It is intended that in the future versions of AIRM model, classes that represent an item of interest shall be assigned the stereotype <<Entity>>.

If an Entity is intended to represent a domain object rather than an information object then this will make it easy to identify the classes that should be represented as a <<FeatureType>>.

Therefore it would be easy to extend the naming convention rules for data types to state that:

1. A structured data type without identity and no operations whose primary purpose is to hold the information shall be assigned no stereotype and the class name shall end with “Type”.

2. A structured data type that may have may have zero or more operations attributes or associations and identity shall be assigned no stereotype.

This would allow the definition of transformation rules to automatically transform the AIRM classes into the relevant ISO 19136 UML Profile classes.

Rule	Description
Classes	
Rule 1	An AIRM class assigned the stereotype <<enumeration>> shall retain its stereotype but be assigned the GML tagged value xsdEncoding= iso19136_2007.
Rule 2	If a class is contained within a Subject Field Package and does not carry the stereotype <<enumeration>> it shall be assigned the stereotype <<FeatureType>> and its associated tagged values.
Rule 3	If the class is contained within the Data Types/Codelists package it shall be assigned the stereotype <<Codelist>> and its associated tagged values
Rule 4	If a code list is only used in a single ATM exchange model it shall belong to that exchange model only.
Rule 5	If a code list is used in multiple ATM exchange model it shall either belong to the first exchange model in which it was encoded or belong to a generic AIRM application schema
Rule 6	If a class is contained within any of the other packages within the Data Types package it shall only be encoded if it represents a complex property (e.g. BearingType, PercentageRangeType, Curve). If the class represents a simple data type such as a measure or CharacterString for which there is a corresponding type in GML/JSON, then these shall not be transformed into the implementation model. Instead a UML to GML mapping rule shall be defined to map the AIRM simple type to the corresponding GML simple type.
Rule 7	A class shall be assigned the stereotype <<Type>> and its associated tagged values when the class: - Extends from an external class assigned the stereotype <<Type>> (e.g. Point, Curve and Surface extend ISO 19107 <<Type>> Classes). - Represents an information object that should carry identify. These shall be manually identified and transformed on a case-by-case basis.

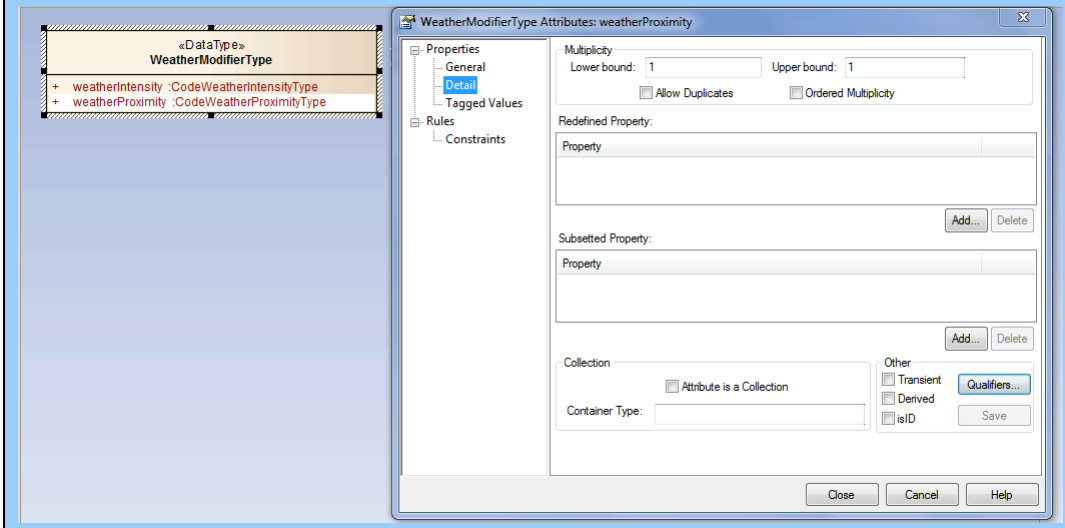
5.1.3 Attributes and Association Roles

The AIRM Foundation Rules state that all attributes and association roles are assigned a default multiplicity unless there is an operation constraint to define an alternate multiplicity:

- Attributes shall be represented with a multiplicity of [0..1] (zero to one), by default. If an operational constraint has been identified then multiplicities should be chosen to reflect such constraints (***AIRM Foundation Rule 22)***.
- Association roles shall be represented with a multiplicity of [0..*] (zero to many), by default. If an operational constraint has been identified then multiplicities should be chosen to reflect such constraints.

Modelling Recommendation 1: Modelling default attribute multiplicities

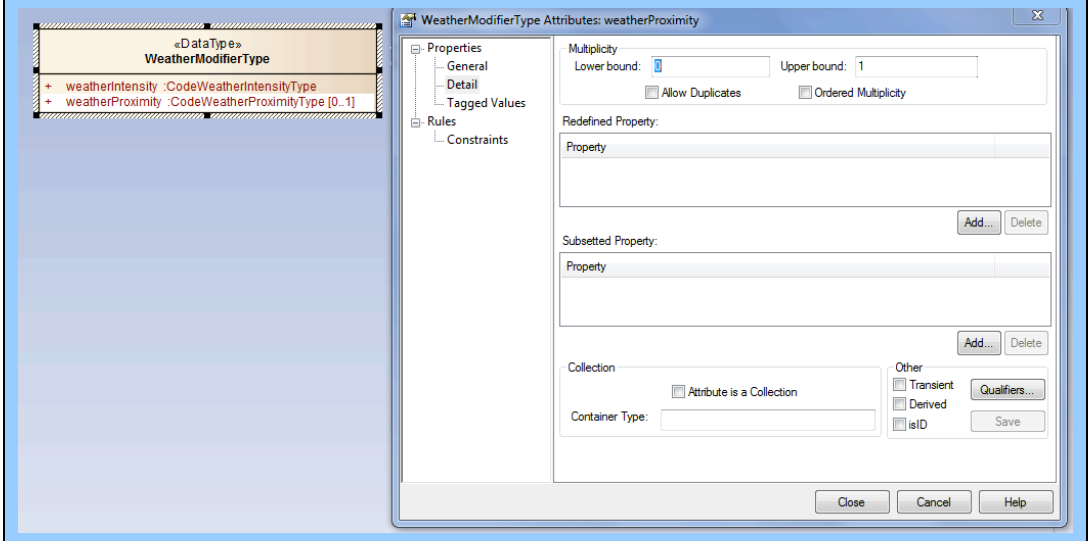
In the current version of the AIRM, it is intended that if a multiplicity is not explicitly shown on the diagram that the multiplicity is the default 0..1. However, in Enterprise Architect all attributes are automatically assigned a multiplicity of 1 when they are created but this is not visible on the diagram.



This poses a technical issue for generating the MDA Transformation. Although a generic rule can be defined to state when the multiplicity lowerBound =1 then transform it to 0 there may be instances when there is an operational constraint for the multiplicity to be 1. Therefore it will be transformed in error.

Enterprise Architect does not support the ability to set the multiplicity lowerBound and upperBound values to null. Therefore, it would be better if all attributes to be represented by the default multiplicity are assigned the correct

multiplicity of 0..1.



There is also a discrepancy between the AIRM Foundation Rules and the ISO 19109 Rules for Application Schema. In the AIRM, if an association is navigable it shall be assigned a multiplicity but it is not required to have a name. This rule is at odds with ISO 19109 that requires all navigable association roles to be assigned both a multiplicity and a name. If an association role is not assigned a name it will be ignored and shall not be encoded in the physical implementation model.

While a modeller can manually update the transformed ISO 19136 UML Profile model to assign role names to the model, this means that the model is different to the original AIRM model.

Recommendation: Constrain AIRM Foundation Rule for Association Role Names

Amend the AIRM Foundation Rules to require all navigable associations to be assigned a name.

The MDA transformation rules for attributes and association roles do not assign tagged values to attributes or association roles. The default encoding rules for attributes and association roles applied by ShapeChange.

There may be a future requirement for the MDA transformation to automatically assign tagged values to all association roles assigned a name and multiplicity. In AIXM, all association roles are encoded as byReference, only. This requirement is not currently defined in the AIRM Foundation Rules, therefore all associations shall be encoded as inlineOrByReference.

5.1.4 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 a physical implementation model such as GML.

The MDA Transformation shall automatically assign the required set of tagged values to model elements. Mandatory tagged values shall be assigned a default value, optional tagged vales will have no value. If the default value is set to FIXME these must be manually edited to assign them a relevant value. This must be undertaken prior to transforming the UML model into an implementation models using the UGAS tool otherwise, the transformation may fail. If a tagged value has been defined default value these can be manually edited to change the value to an allowable alternate value.

5.1.4.1 Tagged values assigned to <<ApplicationSchema>> packages

Tag	Tag Description	Default Value
gmlProfileSchema	URL of the schema location of a GML profile (optional)	
version	Current version of the application schema	FIXME
targetNamespace	Target XML namespace of the application schema. This should be defined as a HTTP URI. For example: http://www.wxxm.aero/schema/wx/1.1	FIXME
xmlns	Namespace prefix to be used as short form of the target namespace. For example: wx	FIXME
xsdDocument	Name of an XML Schema document to create representing the content of this package. For example: wx.xsd NOTE: the file extension .xsd must be included	FIXME
xsdEncoding	Encoding rule to be executed by the UGAS tool to generate the XML schema.	iso19136_2007

5.1.4.2 Tagged values assigned to <<Leaf>> packages

Tag	Tag Description	Default Value
xsdDocument	Name of an XML Schema document to create representing the content of this package. NOTE: the file extension .xsd must be	*PackageName*

	included (e.g. *packageName*.xsd). The package name shall be automatically generated but this needs to be manually updated to add the file extension ".xsd".	
xsdEncoding	Encoding rule to be executed by the UGAS tool to generate the XML schema.	iso19136_2007

5.1.4.3 Tagged values assigned to <<FeatureType>> classes

The following default tagged values are applied when the classes are transformed to <<FeatureType>> classes. These tagged values can be changed on a case-by-case basis:

Tag	Definition	Default Value
noPropertyType	Suppress creation of a standard property type that supports inline or by-reference encoding [ISO 19136]. If set to true then the property shall support inline encoding only. If set to false both inline or by-reference encoding is supported.	false
byValuePropertyType	Create a property type that requires that the instance is encoded inline [ISO 19136]. Should usually be set to false.	false
isCollection	Identifies the type as a collection [ISO 19136].	false
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 [ISO 19136].	
xsdEncoding	Encoding rule to be executed by the UGAS tool to generate the XML schema.	iso19136_2007

5.1.4.4 Tagged values assigned to <<Type>> classes

Tag	Definition	Default Value
noPropertyType	Suppress creation of a standard property type that supports inline or by-reference encoding [ISO 19136]. If set to true then the property shall	false

	support inline encoding only. If set to false both inline or by-reference encoding is supported.	
byValuePropertyType	Create a property type that requires that the instance is encoded inline [ISO 19136]. Should usually be set to false.	false
isCollection	Identifies the type as a collection [ISO 19136].	false
xsdEncoding	Encoding rule to be executed by the UGAS tool to generate the XML schema.	iso19136_2007

5.1.4.5 Tagged values assigned to <<DataType>> classes

The following default tagged values are applied when the classes are transformed to <<DataType>> classes. These tagged values do not need to be changed.

Tag	Definition	Default Value
noPropertyType	Suppress creation of a standard property type that supports inline or by-reference encoding [ISO 19136]. For data types only inline encoding is supported.	false
xsdEncoding	Encoding rule to be executed by the UGAS tool to generate the XML schema.	iso19136_2007

5.1.4.6 Tagged values assigned to <<CodeList>> classes

The following default tagged values are applied when the classes are transformed to <<CodeList>> classes.

Tag	Definition	Default Value
asDictionary	If the value is 'false' the code list will be encoded with the pre-defined enumerants as values in the schema, if 'true' all enumerants will only be maintained in external dictionaries. The default depends on the encoding rule used [ISO 19136, GML 3.3].	true
codeSpace	The URI of the default dictionary that contains code list [ISO 19136]. It is recommended that the URI is a	

Tag	Definition	Default Value
	HTTP URI (e.g. http://[*Domain*]/codelist//[*CodelistName*] Example: http://www.wxxm.aero/codelist/CodeWeatherIntensityType	
xsdEncoding	Encoding rule to be executed by the UGAS tool to generate the XML schema.	iso19136_2007

The tagged value “asDictionary” has been assigned the default value “true”. It is recommended that code lists are maintained externally to the implementation model within a code list register within the SESAR Registry. This will allow the code lists to be governed at a more agile maintenance frequency.

When ShapeChange processes the ISO 19136 UML Profile model it shall generate a GML dictionary for each code list that can be inserted into a code list registry.

If an alternate code list encoding is required then this could be developed in future versions of ShapeChange as the requirement arises.

5.2 UML to Implementation Model Encoding Rules

Once the AIRM model constructs have been converted into the ISO 19136 UML Profile (see the general encoding requirements stated in GML 3.2.1 / ISO 19136:2007, sub-clause E.2.1.1), the resulting Application Schema packages can be transformed by ShapeChange into the implementation schemas: GML application schemas (XML Schema) and JSON schemas (JSON Schema).

ShapeChange is a powerful tool for generating implementation schemas from the UML models as it is highly configurable allowing users to:

- Support alternate stereotypes to the ISO 19136 stereotypes by defining aliases
- Extend the standard ISO/TC 211 mapping rules to support external application schemas or encoding rules
- Develop bespoke encoding rules building upon the standardized ISO 19136 schema conversion rules.

ShapeChange implements the following mapping and schema conversion rules[5]:

[5] These mapping rules can be found: http://shapechange.net/resources/config/

Mapping Rules:

Data Specification	Description
ISO 19136	Mappings for structural data types defined in ISO 19103, ISO 19136 and ISO 19118 (e.g. strings, numerics, measures, date/times)
ISO 19107	Mappings for geometry types
ISO 19108	Mappings for temporal types
ISO 19111	Mappings for Spatial referencing by coordinates
ISO 19115	Mappings for metadata types
ISO 19123 and gmlcov	Mappings for coverage types
ISO 19156	Mappings for observation types
sweCommon	Mappings for types used sweCommon
GSIP	Mappings for GSIP types
JSON	Mappings between ISO 19103 and ISO 19107 types and JSON Schema

Encoding Rules (built-in):

Encoding Rule	Description
iso19136_2007	The GML 3.2 (ISO 19136:2007) Annex E encoding rule. If not encoding rule is specified, this is the default encoding rule.
gml33	The GML 3.2 encoding rule (iso19136_2007) plus extensions for association classes and code lists.
iso19139_2007	The ISO/TS 19139 encoding rule.
sweCommon	The SWE Common Data Model 2.0 encoding rule.

These standardized encoding rules are fixed as part of the Java code of ShapeChange. The standardized encoding rules build on a common set of schema conversion rules[6]. While the GML 3.2 and GML 3.3 encoding rules have been specified in the context of these specific versions of GML, the encoding rules can also be applied – with limitations – with a target version of GML 2.1 or GML 3.1.

In addition to these standardized encoding rules, ShapeChange also supports a range of extension rules that can be used to extend the basic iso19136_2007 and gml33 rules. To develop a custom set of encoding rules for the AIRM this can be configured using the ShapeChange mechanism for specifying custom encoding rules (see, e.g., the set of rules in use with some communities that use ShapeChange and which are provided with the ShapeChange distribution[7]).

The current version of the AIRM Foundation Rules does not state any requirements that would use the ShapeChange extended encoding rules, therefore the default iso19136_2007 encoding rules were used.

Encoding Rule Recommendation: Develop an AIRM extension encoding rule

While the current version of the tool met the requirements for developing the WXXM exchange model, it does not support all of the requirements for other ATM exchange models such as AIXM 5.x.

The transformation tools should be tested to develop other exchange models such as AIXM 5.x to identify additional encoding rules that should be supported.

For example, in AIXM 5.1 all attributes assigned the default multiplicity of 0..1 should also be nillable, in conformance to the AIXM Temporality Model. ShapeChange supports various tagged values that can be assigned to attributes to indicate they are nillable (e.g. <<voidable>>, <<nillable>>).

For a full list of supported extension rules and tagged values see:

- http://shapechange.net/app-schemas/uml-profile/
- http://shapechange.net/targets/xsd/basics/

[6] http://shapechange.net/targets/xsd/basics/

[7] http://shapechange.net/resources/config/StandardRules.xml

5.2.1 AIRM Mapping Rules

To simplify the generation of the AIRM exchange models the following mapping rules were developed (See 7.2 and Annex A for details on the ShapeChange configuration file that applies these rules):

Rule	Description
Rule 1	If an AIRM type class was already implemented in an existing exchange model (e.g. AIXM 5.1) then a mapping rule would be developed to re-use the existing encoding.
Rule 2	If an AIRM type class represented a simple measure type, then it shall not undergo MDA transformation into the ISO 19136 UML profile but an AIRM mapping rule shall be developed to map it to the relevant GML type (e.g. gml:MeasureType).
Rule 3	If an AIRM type class represents a complex measure type (e.g. PercentageRangeType) then this shall undergo MDA transformation and an AIRM data type shall be developed.
Rule 4	All code list and type classes shall be encoded in an AIRM Application Schema unless already implemented in an existing exchange model.

In addition to these generic rules, a set of mapping rules were developed to correct erroneous types that have been used on the AIRM models for which there is no corresponding mapping to GML and JSON elements.

Rule	Description
Rule 1	Use of geometry types which are not directly instantiated in GML or JSON: - DirectPosition shall be mapped to a Point - GM_PointArray shall be mapped to a MultiPoint NOTE: DirectPosition and GM_PointArray are used as the type of a property with the intention to provide location context. However, these types are not geometries and not one of the types identified in ISO 19109 for spatial attributes in application schemas. See ISO 19109, sub-clause 8.7, spatial rules.
Rule 2	The geometry class ExtentOf was originally defined as a <<Union>> of several geometry types to encode the approximate bounding extent

	of a feature. The use of <<Union>> classes is not allowed in the AIRM Foundation Rules. A simple implementation rule was defined to map the ExtentOf type to a Surface, only. In the resulting GML, the gml:boundedBy property can be used to optionally encode the extent as an Envelope or the airm:extentOf property can be used to encode the extent as a Surface. NOTE: one of the geometry types defined in the class is not an geometry type: CircleByCentrePoint.
Rule 3	Use of temporal types which are not directly instantiated in GML or JSON: - TM_PeriodDuration shall be mapped to TimePeriod

5.2.2 Encoding WXXM as GML 2.1

One of the requirements was to demonstrate the ability to encode WXXM as GML 2.1 rather than GML 3.2.1. While ShapeChange can generate WXXM as GML 2.1 this results in an incomplete schema. Many of the types used in the WXXM classes use ISOTC211 types that were not implemented in GML 2.1:

- ISO 19103 Measure types
- ISO 19108 Temporal types
- ISO 19115 Metadata types
- ISO 19123 Coverage types
- ISO 19156 Observations types

Also if the intended GML encoding version is GML 2.1 it is not possible to re-use existing AIRM types such as ElevatedSurface that have already been encoded within an ATM Exchange model such as AIXM 5.1. While it would be possible to mix and match GML 2.1 and GML 3.2 schema components within a GML application schema and instance documents, this is uncommon and may not be supported by software.

6 Transforming AIRM into ISO 19136 UML Profile for GML an Enterprise Architect MDA Transformation

6.1 Overview

The AIRM is intended to be a consolidated logical model that represents all of the ATM application areas. Various ATM exchange models shall be derived from this consolidated logical model. These ATM exchange models represent a subset of the AIRM logical model for a specific ATM application area:

- **AIXM**: management and distribution of Aeronautical Information Services (AIS) data
- **WXXM**: Aeronautical Meteorology
- **FIXM**: Flight Information

Before the required AIRM model constructs can be converted into an implementation schema, the corresponding ATM exchange model application schema must be generated.

This transformation step involves two processes:

1. Merging AIRM model constructs defined within the Abstract, Data Type and Subject Field packages into one or more application schema that shall represent the ATM exchange model (e.g. WXXM)
2. Adding the relevant stereotypes, tagged values and multiplicities to the model constructs

6.2 Creating the application schemas that represent an ATM exchange model

To create an ATM exchange model application schema, the Enterprise Architect MDA Transformation tool shall be used to copy the relevant model constructs into the relevant --XM application schema package and add the relevant stereotypes, tagged values and multiplicities.

Five MDA Transformation templates were developed to transform AIRM model constructs into an ATM exchange model:

MDA Transformation Name	Description
AIRM2GML(Simple)	Transforms the contents of an AIRM package that no sub packages into an application schema.
AIRM2GML(Complex)	Transforms the contents of an AIRM package into an application schema containing sub packages
AIRM2GML(CodeLists)	Transforms selected code list classes into a <<Leaf>> sub

	package within an application schema
AIRM2GML(DataTypes)	Transforms selected structural data type classes into a <<Leaf>> sub package within an application schema
AIRM2GML(Types)	Transforms selected identifiable structural data type classes into a <<Leaf>> sub package within an application schema

6.3 Importing the MDA Transformation Templates

The Enterprise Architect MDA Transformation Templates are available as an .xmi file[8]. These must be imported into the AIRM Enterprise Architect project.

To import the AIRM MDA Transformation Templates into the AIRM model Select Project --> Model Import/Export --> Import Reference Data

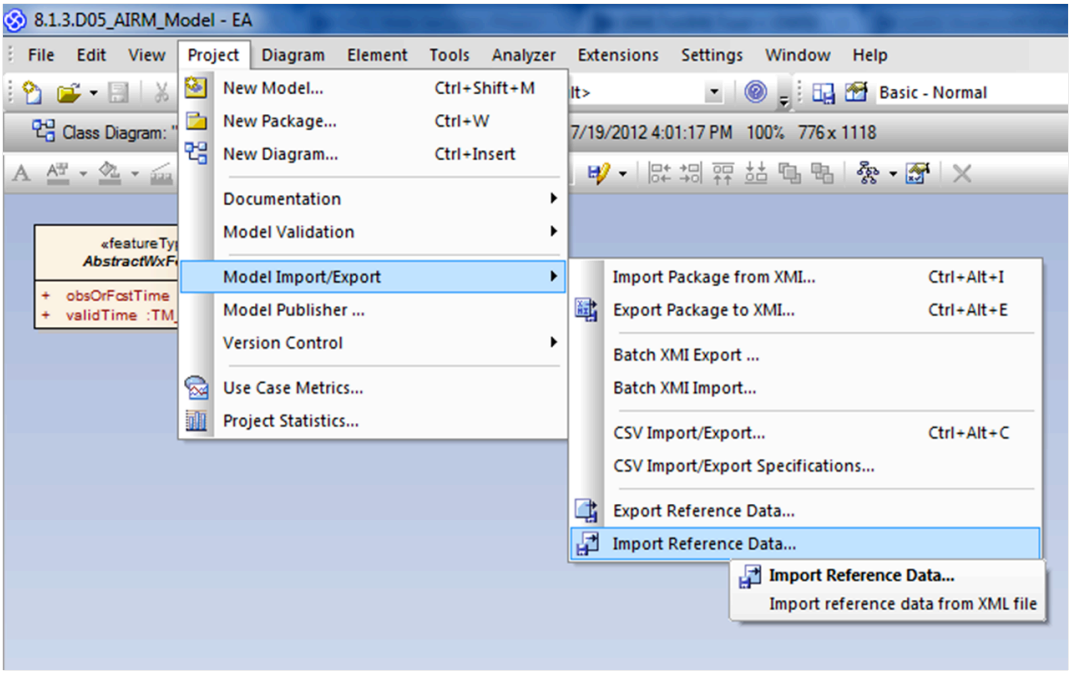


Figure 5. Select MDA Transformation Templates to import

In the Import Reference Data dialog, open Select File and navigate to the directory where the tool is located. Select all five datasets and Import.

[8] http://wiki.snowflakesoftware.com/display/LAB/OWS-9+Aviation%3A+AIRM+Derivation

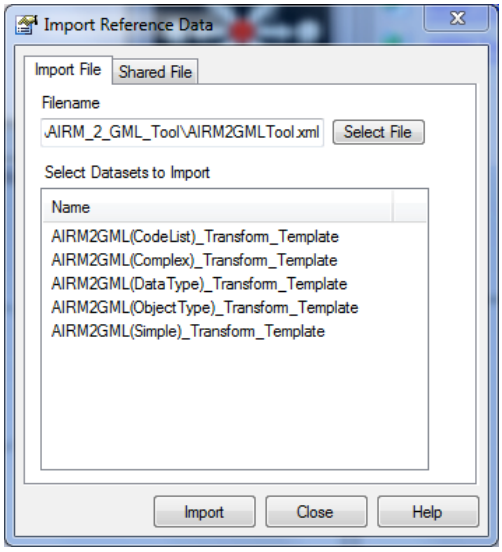


Figure 6. Import MDA Transformation Templates

Finally, hit OK in the Import complete dialog. To check to ensure that all of the MDA Transformation Tools were imported successfully Select Tools --> Model Transformation (MDA) --> Transform Current Package

The list of Transformations should now include:

- AIRM2GML (Simple)
- AIRM2GML (Complex)
- AIRM2GML (CodeLists)
- AIRM2GML (DataType)
- AIRM2GML (ObjectType)

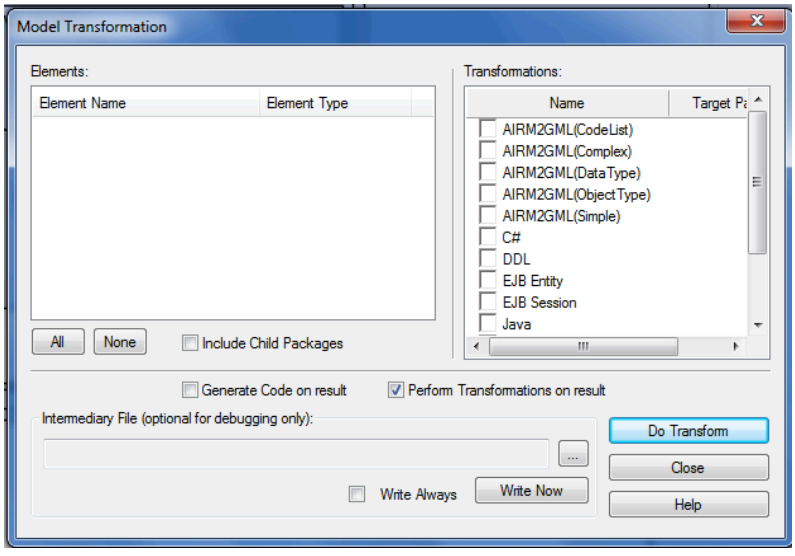


Figure 7. Model Transformation dialog

6.4 Generating an ATM exchange model using the MDA Transformation Templates

The first step in generating the logical model for an ATM exchange model is to create a new class view, which shall be the destination for the transformed exchange models. In the project browser, right-click the model root and select Add → Add View and assign the view a name (e.g. ATM Exchange Models).

The next step is to create a new package that shall represent the exchange model within the view.

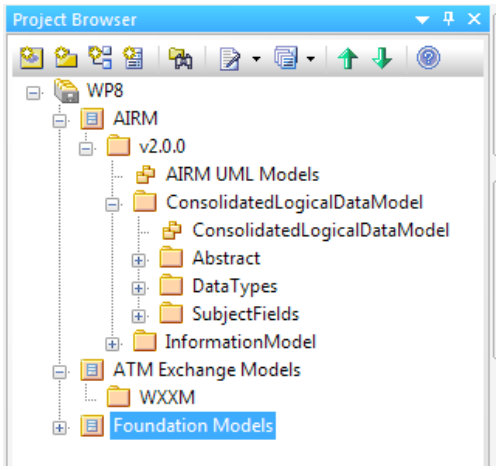


Figure 8. Create new model view and package for ATM Exchange Models

Before transforming any AIRM model constructs or elements, the structure of the application schema should be defined:

- Will the exchange model be comprised of multiple application schemas? If so how will this be structured?

For example, WXXM 1.1.3 is comprised of two application schemas: WX and AVWX which represent specific domain sub-divisions of the model. Whereas, AIXM 5.1 is comprised of 2 application schema: AIXM and Message

Depending on how the exchange model is to be structured will determine the process steps involved in generating the application schema. The first step shall be to generate the application schema package using either the AIRM2GML(Simple) or AIRM2GML(Complex) MDA transformation.

Most application schema will be comprised of modular sub-packages to ensure that implementation models are manageable, therefore use the AIRM2GML(Complex) MDA transformation.

6.4.1 Creating an application schema

To generate an application schema, select a relevant package from the subject field, right-click and select Transform Current Package (Ctrl + Shift + H). This will open the Model Transformation dialog.

Select the AIRM2GML(Complex) transformation and it will require the destination package to be set. So navigate to the WXXM package created in the ATM Exchange Model class view. Finally, select the Include Child Packages and this will list all of the classes contained within the selected package.

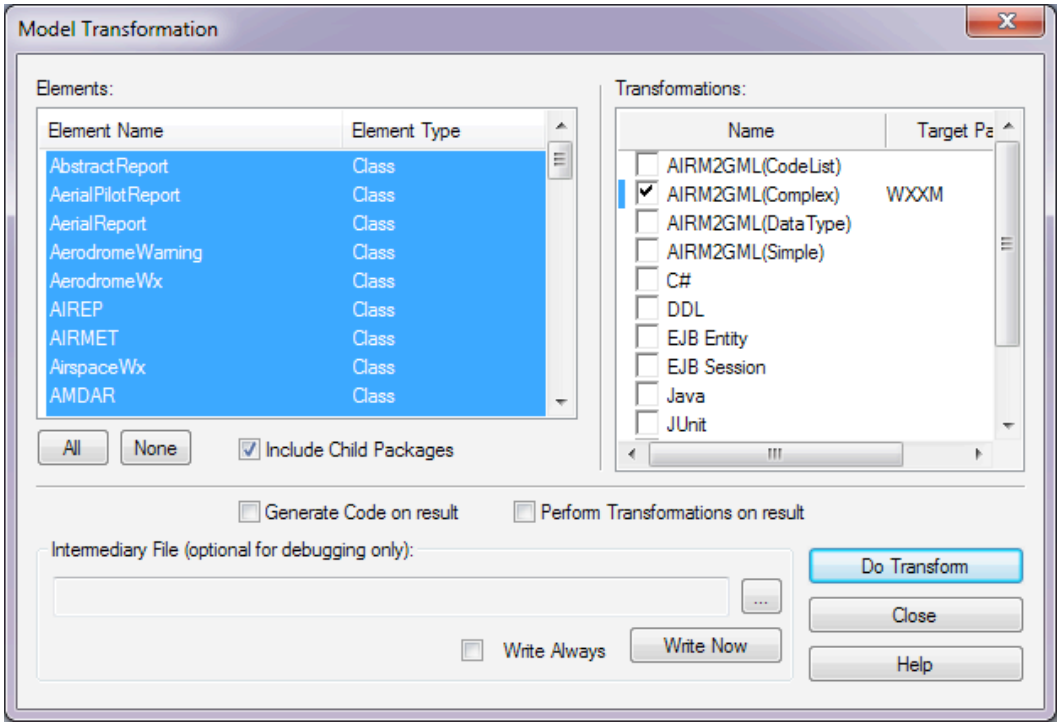


Figure 9. Transforming an AIRM package into a GML Application Schema

NOTE: If specific classes should not be included in the exchange model application schema then these can be de-selected from the list of classes to be transformed.

Finally hit Do Transform. This shall create a new package named “GML Application Schema” in the WXXM package.

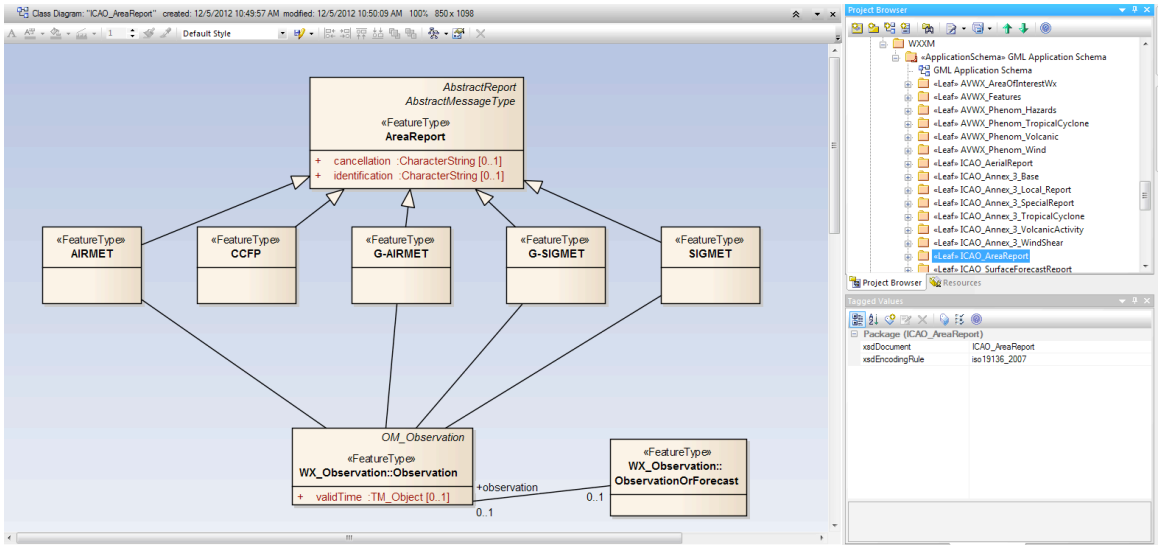


Figure 10. Example of a transformed <<Leaf>> package

NOTE: If the selected package contained sub-packages containing packages, the package shall be moved up a level and shall become a <<Leaf>> package in the application schema.

6.4.2 Adding abstract, data types and code lists

If the application schema should include domain specific data types from the Abstract and Data Types packages, then these can be transformed in two ways:

1. Transform the package and select only the relevant classes to be included
2. If the classes are included in a diagram in the subject field package, then select the classes on the diagram and right-click and select Transform.

These classes shall be added to a <<Leaf>> package within the application schema.

6.4.3 Re-name application schema and define Tagged Values

Once the relevant model constructs and elements have been added to the exchange model application schema, the final task is to rename the application schema and define the tagged values. The only tagged values that need to be edited are:

1. Update <<ApplicationSchema>> package tagged values set to FIXME
2. Update <<Leaf>> package tagged value xsdEncoding to add ".xsd" otherwise the XML schema shall not be assigned a file extension.

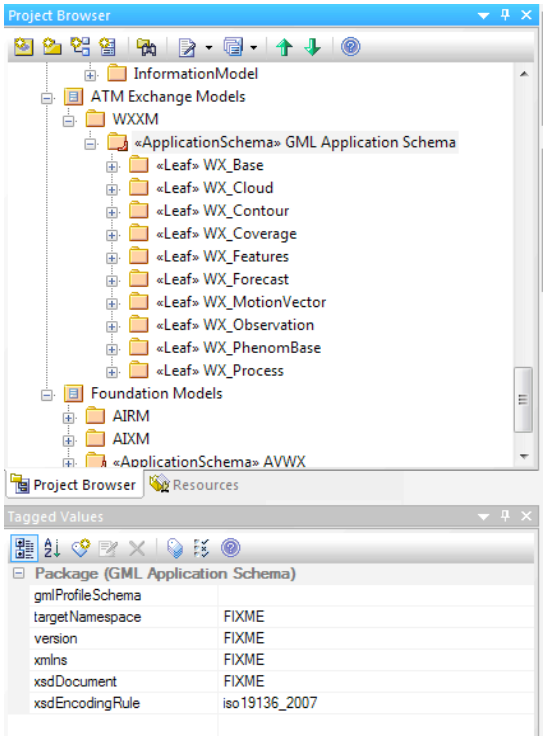


Figure 11. Populating the tagged values for the GML application schema

Once this is complete, repeat the process for each application schema that shall be included in the exchange model.

7 Generating WXXM implementation models using ShapeChange

7.1 Overview

The objective of the work described in this clause is to investigate the programmatic derivation of WXXM into two implementation schemas for Aviation Meteorology:

- XML-based implementation schema based on GML 3.2,
- JSON-based implementation schema.

The original plan was to use GML 2.1 as the basis for the XML Schema implementation. For the reasons stated in 5.2.2 GML 3.2 was used instead.

Initially the current version of the WXXM model (version 1.1.3) was used to explore potential issues of the automatic conversion of WXXM to implementation schemas. The issues detected in this process have been used to improve the transformation from the AIRM to the application schemas as well as the mapping to existing XML Schema and JSON Schema components.

The JSON encoding rule specified in OWS-9 and documented in the OWS-9 SSI UGAS Engineering Report.

Sub-clause 7.2 contains a description of the ShapeChange configuration for the derivation of the XML Schema and JSON Schema implementations from the WXXM model.

Sub-clause 7.3 provides links to the generated schemas and other documents.

7.2 ShapeChange configuration

7.2.1 Overview

The ShapeChange version used to derive the WXXM implementation schemas 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.

7.2.2 General parameters

Table 1 – Parameters for processing the model

Parameter	Value	Comment
inputModelType	EA7	Enterprise Architect
inputFile	AIRM_DEMO.eap	Link to WXXM model
appSchemaNamespaceRegex	^http://www\.opengeospatial\.org/ows9/.*	Convert all application schemas with a target namespace that matches this pattern, in this case the AIRM data types, WX and AVWX schemas
publicOnly	true	Only convert public elements (standard behaviour)
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
addTaggedValues	jsonBaseURI, jsonDirectory, jsonEncodingRule,	Tagged values used by the JSON encoding rule, but which have not yet included in the list of standard tagged

	jsonLayerTableURI	values

7.2.3 XML Schema target

Table 2 – Parameters of the XML Schema target

Parameter	Value	Comment
outputDirectory	wxxm/xsd	Directory in which the XML Schema documents are written
defaultEncodingRule	iso19136_2007	A WXXM-specific encoding rule extending the standard GML encoding rule is applied
xsdMapEntries/ XsdMapEntry	see Table 3	see 5.2.1

Table 3 – Pre-defined mapping of types in the application schemas to XML Schema components

Application schema type	XML type	XML element	Property type
AbstractFeature	gml:AbstractFeature Type	gml:Abstract Feature	gml:FeatureProperty Type
DirectPosition	gml:PointType	gml:Point	gml:PointProperty Type
ExtentOf	gml:Abstract GeometryType	gml:Abstract Geometry	gml:Geometry PropertyType
TM_PeriodDuration	gml:TimePeriod Type	gml:TimePeriod	gml:TimePeriod PropertyType
GM_PointArray	gml:MultiPointType	gml:MultiPoint	gml:MultiPoint PropertyType

AshConcentration	gml:MeasureType	n/a	gml:MeasureType
HorizontalDistance	gml:LengthType	n/a	gml:LengthType
HorizontalVisibilityDistance	gml:LengthType	n/a	gml:LengthType
VerticalVisibilityDistance	gml:LengthType	n/a	gml:LengthType
VerticalDistance	gml:LengthType	n/a	gml:LengthType
DistanceType	gml:LengthType	n/a	gml:LengthType
Depth	gml:LengthType	n/a	gml:LengthType
PercentageType	gml:MeasureType	n/a	gml:MeasureType
QPressure	gml:MeasureType	n/a	gml:MeasureType
RVRDistance	gml:LengthType	n/a	gml:LengthType
ValPercentType	gml:MeasureType	n/a	gml:MeasureType
ValPressureType	gml:MeasureType	n/a	gml:MeasureType
PressureType	gml:MeasureType	n/a	gml:MeasureType
ValSpeedType	gml:SpeedType	n/a	gml:SpeedType
WindGust	gml:MeasureType	n/a	gml:MeasureType
WindSpeed	gml:MeasureType	n/a	gml:MeasureType
TemperatureType	gml:MeasureType	n/a	gml:MeasureType
WindDirection	gml:MeasureType	n/a	gml:MeasureType
Point	aixm:PointType	aixm:Point	n/a
ElevatedPoint	aixm:ElevatedPoint Type	aixm:Elevated Point	n/a
Curve	aixm:CurveType	aixm:Curve	n/a
ElevatedCurve	aixm:ElevatedCurve Type	aixm:Elevated Curve	n/a
Surface	aixm:SurfaceType	aixm:Surface	n/a
ElevatedSurface	aixm:Elevated SurfaceType	aixm:Elevated Surface	n/a

7.2.4 JSON Schema target

Table 4 – Parameters of the JSON Schema target

Parameter	Value	Comment
outputDirectory	wxxm/json *or* wxxm/jsonext	Directory in which the JSON Schema documents are written
defaultEncodingRule	geoservices *or* geoservices_extended	Both encoding rules have been used
jsonBaseURI	http://shapechange.net/tmp/ows9/wxxm/json *or* http://shapechange.net/tmp/ows9/wxxm/jsonext	Base URI of all JSON schema documents
mapEntries/mapEntry	see Table 5	see 5.2.1

Table 5 – Pre-defined mapping of types in the application schemas to JSON Schema

Application schema type	JSON schema (geoservices)	JSON schema (geoservices_extended)
AbstractFeature	http://schemas.opengis.net/gsr/1.0/feature.json	
ExtentOf	http://schemas.opengis.net/gsr/1.0/geometry.json	
TM_PeriodDuration	integer	
AshConcentration	number	http://shapechange.net/tmp/ows9/json/measure.json
HorizontalDistance	number	http://shapechange.net/tmp/ows9/json/measure.json
HorizontalVisibilityDistance	number	http://shapechange.net/tmp/ows9/json/measure.json
VerticalVisibilityDistance	number	http://shapechange.net/tmp/ows9/json/measure.json
VerticalDistance	number	http://shapechange.net/tmp/ows9/json/measure.json
DistanceType	number	http://shapechange.net/tmp/ows9/json/measure.json
Depth	number	http://shapechange.net/tmp/

		ows9/json/measure.json
PercentageType	number	http://shapechange.net/tmp/ows9/json/measure.json
QPressure	number	http://shapechange.net/tmp/ows9/json/measure.json
RVRDistance	number	http://shapechange.net/tmp/ows9/json/measure.json
ValPercentType	number	http://shapechange.net/tmp/ows9/json/measure.json
ValPressureType	number	http://shapechange.net/tmp/ows9/json/measure.json
PressureType	number	http://shapechange.net/tmp/ows9/json/measure.json
ValSpeedType	number	http://shapechange.net/tmp/ows9/json/measure.json
WindGust	number	http://shapechange.net/tmp/ows9/json/measure.json
WindSpeed	number	http://shapechange.net/tmp/ows9/json/measure.json
TemperatureType	number	http://shapechange.net/tmp/ows9/json/measure.json
WindDirection	number	http://shapechange.net/tmp/ows9/json/measure.json
Point	ref:http://schemas.opengis.net/gsr/1.0/point.json	
ElevatedPoint	ref:http://schemas.opengis.net/gsr/1.0/point.json	
Curve	http://schemas.opengis.net/gsr/1.0/polyline.json	
ElevatedCurve	http://schemas.opengis.net/gsr/1.0/polyline.json	
Surface	http://schemas.opengis.net/gsr/1.0/polygon.json	
ElevatedSurface	http://schemas.opengis.net/gsr/1.0/polygon.json	

Note: that some of the mapped JSON representations are incomplete as no suitable JSON schema exists that could be referenced. For example, the geometries or the time period information does not represent all the information in the conceptual schema.

In addition, there are types used in the application schemas without an implementation in JSON. This is the same issue as the one discussed with respect to GML 2.1 (see 6.2.2). However, unlike the GML case, no alternate encoding of these types in JSON Schema exists. In the schema conversion, a warning is issued. The type in JSON Schema used in these cases is "string" in the geoservices" and "object" in the "geoservices_extended" encoding rule. While this represents such properties in the implementation schema, these parts of the schema may be of limited use.

7.3 Implementation schemas

The GML 3.2 application schemas, the JSON schemas for both the extended and simple GeoServices JSON encoding, the code list dictionaries, and a feature catalogue derived from the application schemas, as well as the associated ShapeChange configuration file are available at http://shapechange.net/tmp/ows9/wxxm.

8 Conclusions

8.1 Issues identified when transforming the AIRM

The following table contains a summary of some of the issues identified when transforming the AIRM into an ISO 19136 UML Profile

No	Issue	Fix
1	Some WXXM attributes still contain the stereotype <<property>>	Continue to clean up the AIRM to remove ISO 19136 stereotypes
2	Use of types that are not instantiated in GML or JSON	Change the types to use types that can be directly instantiated in the physical implementation models. See 6.2.1.
3	One directional association roles defined in the reverse direction are not encoded correctly in the resulting GML schema	Update corresponding association roles to ensure that they are defined in the direction of the association so that the association role values are encoded on the target role end.
4	Un-named directional association roles are ignored by ShapeChange	Update the AIRM Foundation Rules to require all directional association roles to be assigned a role name.
5	The model uses an old version of the ISOTC211 foundation schema and has deleted several key packages (e.g. ISO 19103 and ISO 19156 (draft).	Connect to one of the official ISOTC211 model repositories to ensure that the AIRM uses the latest versions of the foundation models and can remain in-sync.

8.2 MDA Transformation template improvements

The following improvements have been identified for the MDA Transformation templates:

1. Addition of tagged values to association roles to constrain the encoding rules to byReference only, if required

2. Define transformation rules to assign class stereotypes based on the Class Naming convention
3. Include package dependencies associations in the transformation
4. Improve ability to transform multiplicities of attributes and associations if AIRM continues to ignore Enterprise Architect multiplicities.

8.3 Encoding Rule improvements

No need for encoding rule improvements have been identified as part of the work. The GML and JSON encoding rules specified in GML 3.2 and OWS-9 SSI UGAS Engineering Report respectively have been sufficient for converting the application schemas derived from AIRM.

8.4 Key Accomplishments

The key accomplishments achieved within the AIRM Derivation work were:

4. Developed a simple, re-usable process for transforming AIRM packages into an ATM exchange model GML application schema following ISO 19109 and ISO 19136 rules.

5. This process was designed to be transferable to any ATM exchange model and is extensible to support more complex requirements for specific ATM exchange models.

6. Leveraged existing, industry standard software for generating implementation schemas from the UML model rather than bespoke scripts or tools. This ensures that the resultant implementation schemas adhere to the ISO 19136 rules for GML application schema encoding and rules for JSON, ensuring consistency.

7. Demonstrated that the tools are highly configurable enabling the transformation and mapping rules to be extended to meet requirements for different ATM exchange models without needing software development.

8.5 Future Work

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

- Address the issues identified in 9.1. and 9.2 to update and clean the AIRM UML model
- Encode data using the implementation schemas.
- Explore the use of JSON for aviation data.
- If needed, standardize JSON Schema implementations for ISO 19100 types.
- Evaluate the ability of the tools to generate other ATM exchange models (e.g. AIXM, FIXM)
- Implement the MDA transformation improvements

Annex A

Mapping Rules

A.1 AIRM Mapping Rules

To execute the additional mapping rules defined within section 6.2.1 and 8.2 map entries have been specified in the ShapeChange configuration (http://shapechange.net/tmp/ows9/wxxm/config_wxxm.xml).

XML Schema target

```
<xsdMapEntries>
	<XsdMapEntry type="AbstractFeature" xsdEncodingRules="iso19136_2007 gml33"
xmlType="gml:AbstractFeatureType" xmlElement="gml:AbstractFeature"
xmlPropertyType="gml:FeaturePropertyType"/>
	<XsdMapEntry type="DirectPosition" xsdEncodingRules="iso19136_2007 gml33" xmlType="gml:PointType"
xmlElement="gml:Point" xmlPropertyType="gml:PointPropertyType"/>
	<XsdMapEntry type="ExtentOf" xsdEncodingRules="iso19136_2007 gml33"
xmlType="gml:AbstractGeometryType" xmlElement="gml:AbstractGeometry"
xmlPropertyType="gml:GeometryPropertyType"/>
	<XsdMapEntry type="TM_PeriodDuration" xsdEncodingRules="iso19136_2007 gml33"
xmlType="gml:TimePeriodType
" xmlElement="gml:TimePeriod" xmlPropertyType="gml:TimePeriodPropertyType"/>
	<XsdMapEntry type="GM_PointArray" xsdEncodingRules="iso19136_2007 gml33"
xmlType="gml:MultiPointType" xmlElement="gml:MultiPoint" xmlPropertyType="gml:MultiPointPropertyType"/>
	<!-- NOTE: where the types have been encoded in an existing exchange model such as AIXM 5.1 these shall
be used -->
	<!-- AIRM Value Types -->
	<XsdMapEntry type="AshConcentration" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:MeasureType" xmlType="gml:MeasureType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
	<XsdMapEntry type="HorizontalDistance" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:LengthType" xmlType="gml:LengthType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
	<XsdMapEntry type="HorizontalVisibilityDistance" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:LengthType" xmlType="gml:LengthType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
	<XsdMapEntry type="VerticalVisibilityDistance" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:LengthType" xmlType="gml:LengthType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
	<XsdMapEntry type="VerticalDistance" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:LengthType" xmlType="gml:LengthType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
	<XsdMapEntry type="DistanceType" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:LengthType" xmlType="gml:LengthType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
	<XsdMapEntry type="Depth" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:LengthType" xmlType="gml:LengthType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
	<XsdMapEntry type="PercentageType" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:MeasureType" xmlType="gml:MeasureType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
	<XsdMapEntry type="QPressure" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:MeasureType" xmlType="gml:MeasureType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
	<XsdMapEntry type="RVRDistance" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:LengthType" xmlType="gml:LengthType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
```

```
        <XsdMapEntry type="ValPercentType" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:MeasureType" xmlType="gml:MeasureType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
        <XsdMapEntry type="ValPressureType" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:MeasureType" xmlType="gml:MeasureType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
        <XsdMapEntry type="PressureType" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:MeasureType" xmlType="gml:MeasureType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
        <XsdMapEntry type="ValSpeedType" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:SpeedType" xmlType="gml:SpeedType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
        <XsdMapEntry type="WindGust" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:MeasureType" xmlType="gml:MeasureType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
        <XsdMapEntry type="WindSpeed" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:MeasureType" xmlType="gml:MeasureType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
        <XsdMapEntry type="TemperatureType" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:MeasureType" xmlType="gml:MeasureType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
        <XsdMapEntry type="WindDirection" xsdEncodingRules="iso19136_2007 gml33"
xmlPropertyType="gml:MeasureType" xmlType="gml:MeasureType" xmlTypeContent="simple"
xmlTypeNilReason="false"/>
        <!-- AIRM Geometry Types -->
        <XsdMapEntry type="Point" xsdEncodingRules="iso19136_2007 gml33" xmlType="aixm:PointType"
xmlElement="aixm:Point" xmlPropertyType="_P_" />
        <XsdMapEntry type="ElevatedPoint" xsdEncodingRules="iso19136_2007 gml33"
xmlType="aixm:ElevatedPointType" xmlElement="aixm:ElevatedPoint" xmlPropertyType="_P_" />
        <XsdMapEntry type="Curve" xsdEncodingRules="iso19136_2007 gml33" xmlType="aixm:CurveType"
xmlElement="aixm:Curve" xmlPropertyType="_P_" />
        <XsdMapEntry type="ElevatedCurve" xsdEncodingRules="iso19136_2007 gml33"
xmlType="aixm:ElevatedCurveType" xmlElement="aixm:ElevatedCurve" xmlPropertyType="_P_" />
        <XsdMapEntry type="Surface" xsdEncodingRules="iso19136_2007 gml33" xmlType="aixm:SurfaceType"
xmlElement="aixm:Surface" xmlPropertyType="_P_" />
        <XsdMapEntry type="ElevatedSurface" xsdEncodingRules="iso19136_2007 gml33"
xmlType="aixm:ElevatedSurfaceType" xmlElement="aixm:ElevatedSurface" xmlPropertyType="_P_" />
</xsdMapEntries>
```

JSON Schema target

```
<mapEntries>
  <MapEntry type="AbstractFeature" rule="*" targetType="ref:http://schemas.opengis.net/gsr/1.0/feature.json" param=""/>
  <MapEntry type="TM_PeriodDuration" rule="*" targetType="integer" param=""/>
  <MapEntry type="ExtentOf" rule="*" targetType="ref:http://schemas.opengis.net/gsr/1.0/geometry.json"
param="geometry"/>
  <!-- AIRM Value Types -->
  <MapEntry type="AshConcentration" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="AshConcentration" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <MapEntry type="HorizontalDistance" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="HorizontalDistance" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <MapEntry type="HorizontalVisibilityDistance" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="HorizontalVisibilityDistance" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <MapEntry type="VerticalVisibilityDistance" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="VerticalVisibilityDistance" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <MapEntry type="VerticalDistance" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="VerticalDistance" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <MapEntry type="DistanceType" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="DistanceType" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
```

```
  <MapEntry type="Depth" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="Depth" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <MapEntry type="PercentageType" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="PercentageType" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <MapEntry type="QPressure" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="QPressure" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <MapEntry type="RVRDistance" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="RVRDistance" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <MapEntry type="ValPercentType" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="ValPercentType" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <MapEntry type="ValPressureType" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="ValPressureType" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <MapEntry type="PressureType" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="PressureType" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <MapEntry type="ValSpeedType" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="ValSpeedType" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <MapEntry type="WindGust" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="WindGust" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <MapEntry type="WindSpeed" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="WindSpeed" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <MapEntry type="TemperatureType" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="TemperatureType" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <MapEntry type="WindDirection" rule="geoservices" targetType="number" param=""/>
  <MapEntry type="WindDirection" rule="geoservices_extended"
targetType="ref:http://shapechange.net/tmp/ows9/json/measure.json" param=""/>
  <!-- AIRM Geometry Types, should be to AIXM types, but we do not have them, so this is the best we can do -->
  <MapEntry type="Point" rule="*" targetType="ref:http://schemas.opengis.net/gsr/1.0/point.json" param="geometry"/>
  <MapEntry type="ElevatedPoint" rule="*" targetType="ref:http://schemas.opengis.net/gsr/1.0/point.json"
param="geometry"/>
  <MapEntry type="Curve" rule="*" targetType="ref:http://schemas.opengis.net/gsr/1.0/polyline.json" param="geometry"/>
  <MapEntry type="ElevatedCurve" rule="*" targetType="ref:http://schemas.opengis.net/gsr/1.0/polyline.json"
param="geometry"/>
  <MapEntry type="Surface" rule="*" targetType="ref:http://schemas.opengis.net/gsr/1.0/polygon.json"
param="geometry"/>
  <MapEntry type="ElevatedSurface" rule="*" targetType="ref:http://schemas.opengis.net/gsr/1.0/polygon.json"
param="geometry"/>
</mapEntries>
```