



Mapping UML to GML Application Schemas Guidelines and Encoding Rules

Version: 1.0rc

Date: 2008-06-20

Status: Baseline

Filename: UGAS-Guidelines-and-Encoding-
Rules.doc

Path: N/a

Author: C. Portele

interactive instruments GmbH

Version	Date	Changes	Editor
0.1-a	11.01.03	First Draft	C. Portele
0.1-b	08.05.03	Document updated	C. Portele
0.3	24.10.05	Document updated	C. Portele
0.3.1	04.06.06	Document updated (GML 3.2.0 support)	C. Portele
0.4	04.08.07	Document updated	C. Portele
1.0	20.06.2008	Document updated	C. Portele

Table of Contents

1	Introduction	3
2	Guidelines for UML Application Schemas	4
2.1	Guidelines with respect to the use of UML	4
2.2	Guidelines with respect to XMI	4
3	Encoding Rules	6
3.1	General Remarks	6
3.2	Encoding Rules.....	6
3.2.1	General encoding requirements.....	6
3.2.2	Input data structure	6
3.2.3	Output data structure	6
3.2.4	Conversion rules.....	6
3.2.5	Extension: tagged values as appinfo elements.....	7
3.2.6	Extension: documentation.....	7
3.2.7	Extension: components with non-public visibility	7
3.2.8	Extension: enumeration as global or anonymous types	7
3.2.9	Extension: dictionary of definitions	7
3.2.10	Extension: simple types with restrictions	7
3.2.11	Extension: use of GML basic types.....	8
3.2.12	Extension: nillable, nilReasonAllowed and implementedByNilReason	8
3.2.13	Extension: asGroup	10
3.2.14	Extension: Mixin classes.....	11
3.2.15	Extension: OCL Constraints	12
3.2.16	Extension: WSDL definitions from <<Interface>> types.....	16



1 Introduction

This document describes a mapping from an UML Application Schema into a GML Application Schema via an XMI representation of the UML Application Schema. The UML Application Schema must conform to ISO 19109:2005 and follow some additional modeling guidelines described in this document.

The rules have been tested in several projects and are based on the definitions of GML 3.2.1 (ISO 19136:2007), ISO/TS 19139:2007, ISO 19118 rev 1, ISO/TS 19103:2005, and ISO 19109:2005. Knowledge of these documents is required to understand this document.



2 Guidelines for UML Application Schemas

2.1 Guidelines with respect to the use of UML

To be a valid input into the mapping the UML Application Schema should conform to GML 3.2.1 (ISO 19136) Annex E and/or ISO/TS 19139. In some cases, extensions are supported, too, but must be activated using non-standard switches.

2.2 Guidelines with respect to XMI

To be valid input into the mapping process, the XMI representation of the UML Application Schema must conform to the following:

- The UML model containing the application schema and all other required model elements shall be stored in a single XMI document.
- The XMI document shall be well-formed.
- The XMI document shall conform with XMI version 1.0.
- The XMI document shall be valid, i.e. contain a DOCTYPE declaration and the document must validate against this document type definition. This DTD must be the normative DTD that is part of UML 1.3.
- Only the contents of the <XMI.header> and the <XMI.content> elements are be used by the UGAS Tool. All other elements will be ignored.

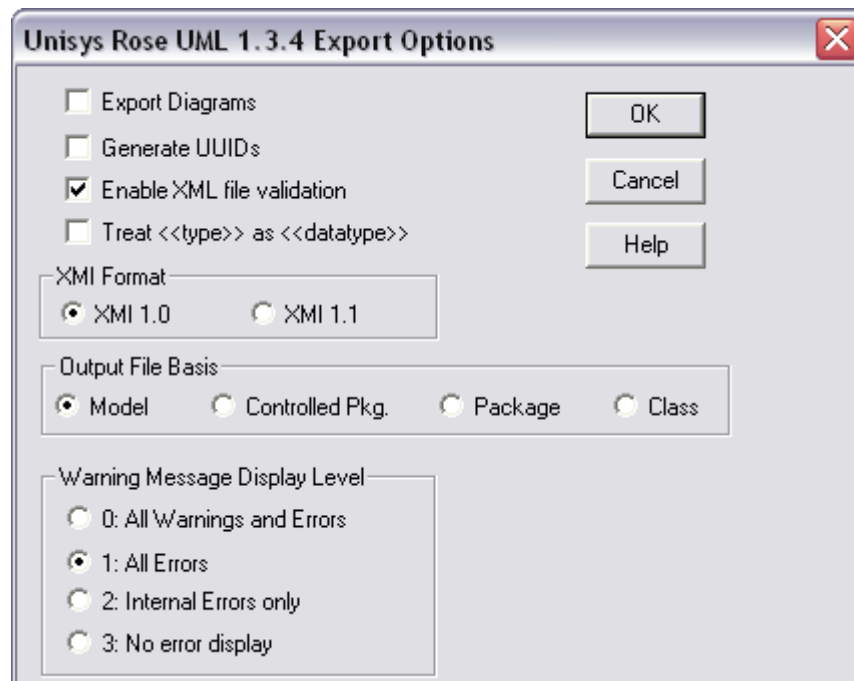


Figure 1 – Export options with the Unisys XMI export tool for Rational Rose

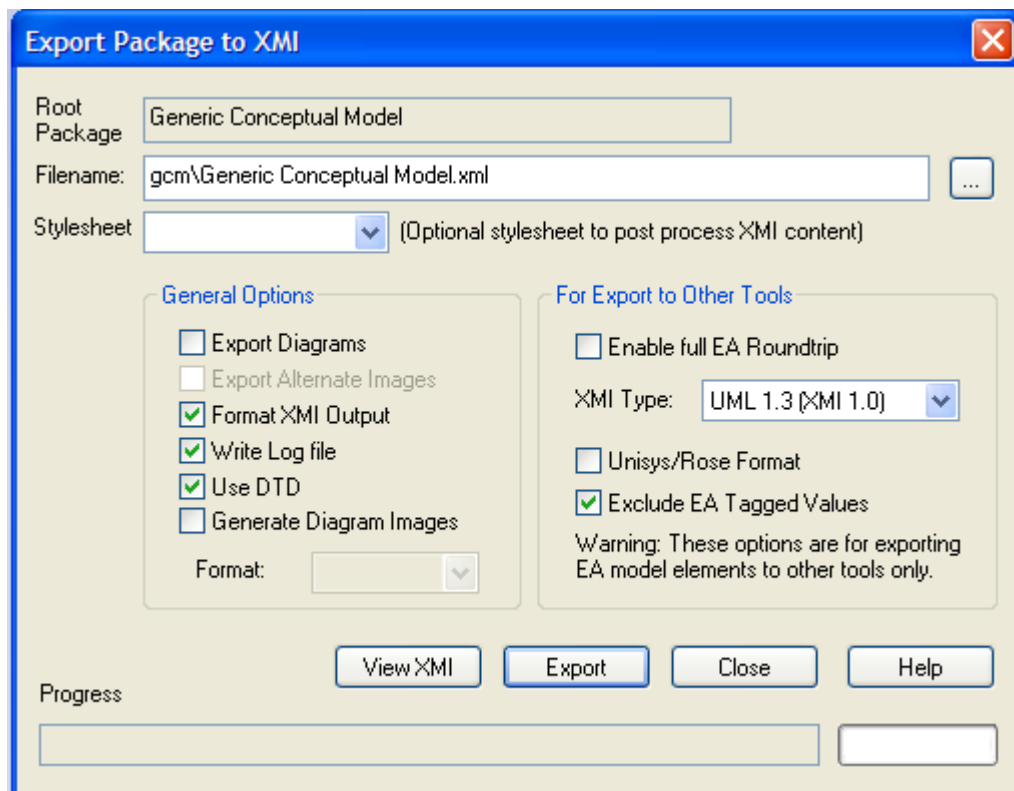


Figure 2 – Export options with Enterprise Architect



3 Encoding Rules

3.1 General Remarks

The mapping from an ISO 19109 conformant UML Application Schema to the corresponding GML Application Schema is based on a set of encoding rules. These encoding rules are identical with those specified in GML 3.2.1 (ISO 19136) Annex E, ISO/TS 19139 plus a set of extensions to the standardized rules. Only the extensions from the standardized encoding rules of GML 3.2.1 and ISO/TS 19139 are specified in this document.

The schema encoding rules are based on the general idea that the class definitions in the application schema are mapped to type and element declarations in XML Schema, so that the objects in the instance model can be mapped to corresponding element structures in the XML document.

The target version of GML is 3.2.1.

3.2 Encoding Rules

3.2.1 General encoding requirements

3.2.1.1 Application schema

The input Application Schema shall be defined in XMI 1.0 (UML 1.3 DTD) according to the guidelines specified in the previous chapter.

3.2.1.2 Character repertoire and languages

The character encoding of the XML Schema files (with the associated character repertoire) will be the same as the character encoding of the XMI file.

3.2.1.3 Exchange metadata

n/a

3.2.1.4 Dataset and object identification

n/a

3.2.1.5 Update mechanism

n/a

3.2.2 Input data structure

n/a

3.2.3 Output data structure

n/a

3.2.4 Conversion rules

The schema conversion rules define how to produce XML Schema documents (XSDs) according to an ISO 19109 application schema expressed in UML.

Different use cases result in different encoding rules. Currently, the geographic information standards specify two XML based encoding rules:

- GML 3.2.1 (ISO 19136) Annex E specifies a XML based encoding rule for ISO 19109 conformant application schemas that can be represented using a restricted profile of UML that allows for a conversion to XML Schema. The encoding rule has mainly been developed for the purpose of application schemas specifying feature types and their properties.



- ISO/TS 19139 specifies a XML based encoding rule for conceptual schemas specifying types that describe geographic resources, e.g. metadata according to ISO 19115 and feature catalogues according to ISO 19110. The encoding rule supports the UML profile as used in the UML models commonly used in the standards developed by ISO/TC 211.

To identify the applicable encoding rule, a tagged value "xsdEncodingRule" may be provided for packages and classifiers. The pre-defined values are:

- "iso19136_2007" (the default, if no value is provided): GML 3.2.1 Annex E encoding rule
- "iso19139_2007": ISO/TS 19139 encoding rule
- "iso19136_2007_ShapeChange_1.0_Extensions": GML 3.2.1 Annex E encoding rule with extensions specified in the remainder of this clause; note that these extensions are not standardised and may be experimental; i.e. they should be used with caution
- "iso19136_2007_NoGmlBaseTypes": GML 3.2.1 Annex E encoding rule, but without using the GML base types
- "notEncoded": The model element is ignored in the schema conversion

3.2.5 Extension: tagged values as appinfo elements

All tagged values that are associated with ShapeChange via the -V option are mapped to appinfo elements of the corresponding model element.

Example:

```
<complexType name="PAA010Type">
  <annotation>
    <documentation>Mine: An excavation made in the earth for the purpose
of extracting natural deposits.</documentation>
    <appinfo>
      <sc:taggedValue tag="primaryCode">Mine</sc:taggedValue>
      <sc:taggedValue tag="secondaryCode">PAA010</sc:taggedValue>
    </appinfo>
  </annotation>
  ...
</complexType>
```

3.2.6 Extension: documentation

It can be selected whether documentation elements are included in annotation elements in the XML Schema files (command line option -d/-D).

3.2.7 Extension: components with non-public visibility

By default only model elements with public visibility are encoded. However, this can be relaxed to include all model elements regardless of their visibility (command line option -p/-P).

3.2.8 Extension: enumeration as global or anonymous types

<<Enumeration>> classes may be mapped to a named simpleType or to an anonymous simpleType (command line option -e/-E).

3.2.9 Extension: dictionary of definitions

A GML dictionary with the types/properties and their definitions as well as key characteristics can be exported. XSLT stylesheets are available to style the dictionaries to HTML for direct viewing in a web browser.

3.2.10 Extension: simple types with restrictions

Basic types may be restricted with facets. The length of a subtype of CharacterString may be restricted (tagged value "length"), the allowed range of numeric values may be limited (tagged values "rangeMinimum" and "rangeMaximum").



Note that this extensions is deprecated and is expected to be replaced by the general support for OCL constraints (see below).

3.2.11 Extension: use of GML basic types

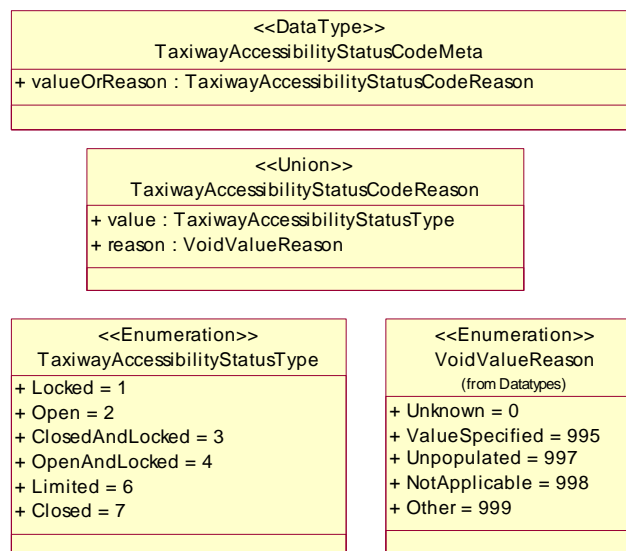
The use of the base types of GML (gml:AbstractObject, gml:AbstractGMLObject, gml:AbstractFeature) may be suppressed (command line option -G or individually by a tagged value "noGMLType").

3.2.12 Extension: nillable, nilReasonAllowed and implementedByNilReason

If an attribute has a tagged value "nillable" with value "true", the property element would be defined with "nillable" set to "true".

If a type has a tagged value "nilReasonAllowed" with value "true", all property types for this property would be defined with an optional nilReason attribute.

If a property of the conceptual model is implemented by the nilReason concept of GML, the tagged value "implementedByNilReason" is set. As a result, the following classes



are encoded in the GML application schema as

```
<element name="TaxiwayAccessibilityStatusCodeMeta"
  type="gsp:TaxiwayAccessibilityStatusCodeMetaType" substitutionGroup="gsp:DatatypeMeta"/>

<complexType name="TaxiwayAccessibilityStatusCodeMetaType">
  <annotation>
    <documentation>Taxiway Accessibility Status Code or Reason; with Metadata: A coded domain value
denoting the accessibility status type of a taxiway, accompanied by the reason that the value may be absent and
associated metadata.</documentation>
    <appinfo>
      <sc:taggedValue tag="primaryCode">TaxiwayAccessibilityStatusCodeMeta</sc:taggedValue>
    </appinfo>
  </annotation>
  <complexContent>
    <extension base="gsp:DatatypeMetaType">
      <sequence>
        <element name="valueOrReason" nillable="true">
```



```
<annotation>
  <documentation>Taxiway Accessibility Status Code Value: A taxiway accessibility status
code value.</documentation>
  <appinfo>
    <sc:taggedValue tag="primaryCode">value</sc:taggedValue>
  </appinfo>
</annotation>
<complexType>
  <simpleContent>
    <extension base="gsp:TaxiwayAccessibilityStatusTypeType">
      <attribute name="nilReason" type="gml:NilReasonType"/>
    </extension>
  </simpleContent>
</complexType>
</element>
</sequence>
</extension>
</complexContent>
</complexType>

<complexType name="TaxiwayAccessibilityStatusCodeMetaPropertyType">
  <sequence>
    <element ref="gsp:TaxiwayAccessibilityStatusCodeMeta"/>
  </sequence>
</complexType>

<simpleType name="TaxiwayAccessibilityStatusTypeType">
  <annotation>
    <documentation>Taxiway Accessibility Status Type: A coded domain value denoting the accessibility
status type of a taxiway.</documentation>
    <appinfo>
      <sc:taggedValue tag="primaryCode">TaxiwayAccessibilityStatusType</sc:taggedValue>
    </appinfo>
  </annotation>
  <restriction base="string">
    <enumeration value="1">
      <annotation>
        <documentation>Locked: Access is prevented by a physical barrier, requiring special means to
pass (for example: a key).</documentation>
        <appinfo>
          <sc:taggedValue tag="primaryCode">Locked</sc:taggedValue>
          <sc:taggedValue tag="secondaryCode">1</sc:taggedValue>
        </appinfo>
      </annotation>
    </enumeration>
    <enumeration value="2">
      <annotation>
        <documentation>Open: Access is officially allowed. [desc] May be covered and/or blocked by a
physical barrier that is temporarily passable.</documentation>
        <appinfo>
          <sc:taggedValue tag="primaryCode">Open</sc:taggedValue>
          <sc:taggedValue tag="secondaryCode">2</sc:taggedValue>
        </appinfo>
      </annotation>
    </enumeration>
  </restriction>
</simpleType>
```



```

    </appinfo>
  </annotation>
</enumeration>
<enumeration value="3">
  <annotation>
    <documentation>Locked Closed: Access is officially prohibited and is restricted by a physical
barrier, requiring special means to pass (for example: a key).</documentation>
    <appinfo>
      <sc:taggedValue tag="primaryCode">ClosedAndLocked</sc:taggedValue>
      <sc:taggedValue tag="secondaryCode">3</sc:taggedValue>
    </appinfo>
  </annotation>
</enumeration>
<enumeration value="4">
  <annotation>
    <documentation>Locked Open: Access is officially allowed although restricted by a physical
barrier that is currently open, requiring special means to close and prevent future passage (for example: a
key).</documentation>
    <appinfo>
      <sc:taggedValue tag="primaryCode">OpenAndLocked</sc:taggedValue>
      <sc:taggedValue tag="secondaryCode">4</sc:taggedValue>
    </appinfo>
  </annotation>
</enumeration>
<enumeration value="6">
  <annotation>
    <documentation>Limited: A limitation on access, but not function, has been imposed. [desc] Not
necessarily enforced by a physical barrier.</documentation>
    <appinfo>
      <sc:taggedValue tag="primaryCode">Limited</sc:taggedValue>
      <sc:taggedValue tag="secondaryCode">6</sc:taggedValue>
    </appinfo>
  </annotation>
</enumeration>
<enumeration value="7">
  <annotation>
    <documentation>Closed: Access is officially prohibited. [desc] May be covered and/or blocked
by a physical barrier.</documentation>
    <appinfo>
      <sc:taggedValue tag="primaryCode">Closed</sc:taggedValue>
      <sc:taggedValue tag="secondaryCode">7</sc:taggedValue>
    </appinfo>
  </annotation>
</enumeration>
</restriction>
</simpleType>

```

3.2.13 Extension: asGroup

If a <<Union>> class has a tagged value "asGroup" with a value "true" then it is encoded as a global group which is referenced wherever a property is defined that has the union class as its value. Note that this is only valid, if it is clear from the context how to map the individual values to the conceptual model.



3.2.14 Extension: Mixin classes

Due to the fact that several implementation platforms including XML Schema supports only type derivation from a single base type (element substitutability in XML Schema is restricted to a single element, too), the use of multiple inheritance is currently not supported by GML 3.2.1 Annex E.

However, for conceptual modelling, the ability to define abstract types which capture a set of properties that are associated with a concept is sometimes very convenient.

The following additional rules for such abstract types are therefore supported by ShapeChange:

If a class is a specialization of another class, then this class shall have one of the stereotypes <<FeatureType>>, <<DataType>>, no stereotype or <<Type>>.

The class shall have zero or one supertype with the same stereotype and zero or more abstract supertypes of the stereotype <<Type>>.

I.e., disregarding classes with stereotype <<Type>>, a generalization relationship shall be specified only between two classes that are either:

- both feature types (stereotype <<FeatureType>>),
- both object types (no stereotype), or
- both data types (stereotype <<DataType>>).

For every class <<Type>> all direct or indirect subtypes shall be either

- all feature or object types (stereotypes <<FeatureType>>, no stereotype or <<Type>>),
- all data types (stereotypes <<DataType>> or <<Type>>).

All generalization relationships between classes shall have no stereotype. The discriminator property of the UML generalization shall be blank.

The abstract mixin class (example in the GSIP: GeometryInfo) is encoded as a group with all properties (attributes and navigable association ends) encoded as usual. This group will be referenced from the subtype.

EXAMPLE GeometryInfo and PointGeometryInfo:

```
<group name="GeometryInfoGroup">
  <annotation>
    <documentation>Geometry Information: An abstract modeling entity serving as a
superclass that collects shared properties (attributes and associations) of modeling
entities that specify geometric representation information about a feature. [desc] For
example, the horizontal and/or vertical metadata, notes, and/or restriction(s) and/or
security control(s) applicable to dissemination of data regarding the geometric
representation of the feature. [constraint] There exists an associated: Event Entity or
Feature Entity</documentation>
    <appinfo>
      <sc:taggedValue tag="primaryCode">GeometryInfo</sc:taggedValue>
      <sc:taggedValue tag="secondaryCode">ZI029</sc:taggedValue>
      <sc:taggedValue tag="oclExpressions">inv: self.eventEntity-&gt;notEmpty() or
self.featureEntity-&gt;notEmpty()</sc:taggedValue>
    </appinfo>
  </annotation>
  <sequence>
    <element maxOccurs="unbounded" minOccurs="0" name="eventEntity"
type="gml:ReferenceType">
      <annotation>
        <documentation>Geometry of Event Entity: An event for which this geometry
representation applies.</documentation>
        <appinfo>
          <targetElement xmlns="http://www.opengis.net/gml/3.2">
            gsip:EventEntity
          </targetElement>
          <reversePropertyName xmlns="http://www.opengis.net/gml/3.2">
            gsip:geometry
          </reversePropertyName>
          <sc:taggedValue tag="primaryCode">eventEntity</sc:taggedValue>
        </appinfo>
      </annotation>
    </element>
    <element maxOccurs="unbounded" minOccurs="0" name="featureEntity">
```



```
type="gml:ReferenceType">
  <annotation>
    <documentation>Geometry of Feature Entity: A feature entity for which this
geometry representation applies.</documentation>
    <appinfo>
      <targetElement xmlns="http://www.opengis.net/gml/3.2">
        gsip:FeatureEntity
      </targetElement>
      <reversePropertyName xmlns="http://www.opengis.net/gml/3.2">
        gsip:geometry
      </reversePropertyName>
      <sc:taggedValue tag="primaryCode">featureEntity</sc:taggedValue>
    </appinfo>
  </annotation>
</element>
<element name="horizontalCoordMetadata" type="gml:ReferenceType">
  <annotation>
    <documentation>Horizontal Coordinate Metadata: The horizontal coordinate metadata
of this geometry.</documentation>
    <appinfo>
      <targetElement xmlns="http://www.opengis.net/gml/3.2">
        gsip:HorizCoordMetadata
      </targetElement>
      <reversePropertyName xmlns="http://www.opengis.net/gml/3.2">
        gsip:geometryInfo
      </reversePropertyName>
      <sc:taggedValue tag="primaryCode">horizontalCoordMetadata</sc:taggedValue>
    </appinfo>
  </annotation>
</element>
<!-- ... -->
</sequence>
</group>

<element name="PointGeometryInfo" substitutionGroup="gml:Point"
type="gsip:PointGeometryInfoType"/>
<complexType name="PointGeometryInfoType">
  <annotation>
    <documentation>Point Geometry Information: A modeling entity collecting geometric
representation information about a feature that is modeled as a spatial point. [desc] A
spatial point is a 0-dimensional geometric primitive, representing a
position.</documentation>
    <appinfo>
      <sc:taggedValue tag="primaryCode">PointGeometryInfo</sc:taggedValue>
      <sc:taggedValue tag="secondaryCode">ZI007</sc:taggedValue>
      <sc:taggedValue tag="oclExpressions">
        </sc:taggedValue>
    </appinfo>
  </annotation>
  <complexContent>
    <extension base="gml:PointType">
      <sequence>
        <group ref="gsip:GeometryInfoGroup"/>
        <!-- ... -->
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

3.2.15 Extension: OCL Constraints

3.2.15.1 Overview

The tagged value "oclExpressions" on types allows capture of additional constraints related to the feature type using OCL. These constraints are typically also shown in the class diagram showing the feature type.

The tagged value "schPatterns" on types is a placeholder for a translation of the "oclExpressions" to Schematron in case the oclExpression cannot be converted automatically to Schematron.



On the GML application schema level Schematron (ISO/IEC 19757-3:2006¹) is used in most cases as the target language for constraints. Schematron is already used by GML to express constraints that cannot be represented in XML Schema. It is currently considered the most appropriate language to express constraints on the XML level. Tools exist to process Schematron constraints and assert the compliance of an instance document with the specified constraints².

It is not feasible to provide a general, full mapping between OCL and Schematron. Thus, this extension focusses on typical constraint patterns found in selected application schemas.

For restricting the values of types with simple content, xsd:restriction and the appropriate facets are used where possible.

It must be noted that Schematron is based on XPath expressions. This has the effect that Schematron constraints are in practice limited to a single document; in addition, Schematron is not Xlink-aware (without support by additional Xpath functions).

One Schematron file per GML application schema is created to capture the assertions related to this application schema.

Following are a sample of the OCL constraints currently supported by this extension, preceded by an "English equivalent" and followed by a translation into Schematron or XML Schema. Some constraints have some additional discussion about other representations, in particular directly in XML Schema.

3.2.15.2 Constraints on property values

These constraints could in principle also be represented directly in XML Schema by using an anonymous property type for the property element. However, this does not work, if the constraint is specified on a type but the original property is specified on a supertype. To use a common, general approach, OCL and Schematron will be used for these cases.

3.2.15.2.1 Constraints on enumeration values

For single-valued properties:

OCL	Schematron
inv: self.<ATT>.value = #<VAL> {or self.<ATT>.value = #<VAL>}*	<pre> <sch:rule context="<QUALIFIED_FEATURETYPE<NAME>"> <sch:assert test="<QUALIFIED_ATT>='<VAL>' {or <QUALIFIED_ATT>='<VAL>'}*"> ...text from constraint column goes here... </sch:assert> </sch:rule> </pre>

For multi-valued properties:

OCL	Schematron
inv: self.<ATT>.values->forAll(elem = #<VAL1> {or elem = #<VAL>}*) and self.<ATT>->isUnique (values.elements)	<pre> <sch:rule context="<QUALIFIED_FEATURETYPE<NAME>"> <sch:assert test="(<QUALIFIED_ATT>='<VAL>' {or <QUALIFIED_ATT>='<VAL>'}*) and count(<QUALIFIED_ATT>[.=preceding- sibling::<QUALIFIED_ATT>])=0"> ...text from constraint column goes here... </sch:assert> </sch:rule> </pre>

¹ [http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip)

² for example: <http://www.schematron.com/implementation.html>



3.2.15.2.2 Constraints on the value or void reason of a property

OCL	Schematron
inv: self.<ATT>.value = <VAL>	<pre><sch:rule context="<QUALIFIED_FEATURETYPENAME>"> <sch:assert test="<QUALIFIED_ATT>=<MAPPEDVAL>"> ...text from constraint column goes here... </sch:assert> </sch:rule></pre> <p>where <MAPPED_VAL> = 'false()' / 'true()' for <VAL> = 'false' / 'true' (other predefined value mappings may be configured).</p>

OCL	Schematron
inv: self.<ATT>.reason = <VAL>	<pre><sch:rule context="<QUALIFIED_FEATURETYPENAME>"> <sch:assert test="<QUALIFIED_ATT>/@nilReason=<MAPPED_VAL> and <QUALIFIED_ATT>/@xsi:nil='true'"> ...text from constraint column goes here... </sch:assert> </sch:rule></pre> <p>where <MAPPED_VAL> = 'notApplicable' etc. for <VAL> = '#NotApplicable', etc. (other predefined value mappings may be configured).</p>

3.2.15.2.3 Constraints on the value of (complex) data types

An example is a constraint that states that in a collection of obstruction heights no two members shall have the same value of their datum. Currently no such constraints are supported, but are likely to be added in the future. In OCL this constraint could be stated as, for example:

inv: self.obstructionHeight->isUnique(datum.name)

The value of obstructionHeight is a collection of height-with-accuracy-and-datum values and the datums are identified by a name property.

3.2.15.3 Constraints on the type of a property

OCL	Schematron
inv: self.<ATT>->forAll(g g.oclIsKindOf(<TYPE>) {or g.oclIsKindOf(<TYPE>)}*)	<pre><sch:rule context="<QUALIFIED_FEATURETYPENAME>"> <sch:assert test="count(<QUALIFIED_ATT>/*)= count(<QUALIFIED_ATT>/<QUALIFIED_TYPE>) {+ count(<QUALIFIED_ATT>/<QUALIFIED_TYPE>)}*"> ...text from constraint column goes here... </sch:assert> </sch:rule></pre> <p>This has the potential issue that no subtypes may be used, only the explicitly listed type. Support for subtypes would require Xpath2, but Schematron is based on Xpath.</p>

3.2.15.4 Constraints on a property value of an associated object

OCL	Schematron
inv: self.<ATT>->forAll(w w.<ATT2>.value = <VAL> {or w.<ATT2>.value = <VAL>}*)	<pre><sch:rule context="<QUALIFIED_FEATURETYPENAME>"> <sch:assert test="valueOf(<QUALIFIED_ATT>)/<QUALIFIED_ATT2>=<VAL> {or valueOf(<QUALIFIED_ATT>)/<QUALIFIED_ATT2>=<VAL>}*"></pre>



	<p>...text from constraint column goes here...</p> <pre></sch:assert> </sch:rule></pre> <p>which uses the valueOf() Xpath function from the current WFS 2.0 drafts; valueOf() supports resolution of references.</p>
--	--

3.2.15.5 Constraints on the existence of a property value

OCL	Schematron
<pre>inv: self.<ATT>->notEmpty() {or self.building->notEmpty()}</pre>	<pre><sch:rule context="<QUALIFIED_FEATURETYPENAME>"> <sch:assert test="count(<QUALIFIED_ATT> {+count(<QUALIFIED_ATT>)}* >0"> ...text from constraint column goes here... </sch:assert> </sch:rule></pre> <p>Similarly "empty()" would be mapped to "count(...)=0".</p>
<pre>inv: count(self.<ATT>) <op> <N></pre>	<pre><sch:rule context="<QUALIFIED_FEATURETYPENAME>"> <sch:assert test="count(<QUALIFIED_ATT>) <op> <N>"> ...text from constraint column goes here... </sch:assert> </sch:rule></pre> <p>where <op> may be, for example, '='.</p>

3.2.15.6 Constraints on the value domain of a property

Note that the following constraints are not represented as such in the NAS, but commonly occur. In the NAS some of them are currently represented in specific tagged values which are mapped to types with xsd:restrictions and restricting facets. The table below shows the OCL representations and the restricting facets.

OCL	XML Schema - restricting facets
<pre>inv: self <op> <N> (on a type inheriting from Number)</pre>	<p>restriction of the appropriate base type with a facet "minInclusive", "minExclusive", "maxInclusive", "maxExclusive" depending on the operator</p>
<pre>inv: self.uom.uomName=<UOM> (on a type inheriting from Measure)</pre>	<p>restriction of the base type with a restriction of the uom attribute to the fixed value of <UOM></p>
<pre>inv: self.size <op> <N> (on a type inheriting from CharacterString)</pre>	<p>restriction of the base type with a facet "minLength", "maxLength" or "length"</p>
<pre>inv: self->xsdPatternMatch(<PATTERN>) (on a type inheriting from CharacterString; note that since CharacterString does not provide an operation we define a non-standard additional operation "xsdPatternMatch(pattern : CharacterString) : Boolean" which return true if the value of self satisfies the XML Schema pattern. This is formally not correct, but all other alternatives discussed needed non-standard extensions, too.</pre>	<p>restriction of the base type with a facet "pattern"</p>

Currently, no support for the facets "totalDigits", "fractionDigits" and "whitespace" has been specified. If required, a similar approach like for "pattern" might be considered.



The "enumeration" facet and related constraints are supported in UML already by types with the stereotype <<enumeration>>.

3.2.16 Extension: Use of UML 2 notation

UML notation in accordance with UML 2 is supported including the placement of cardinality information for attributes (after the type name instead of after the attribute name) and the naming of all stereotypes uses lowerCamelCase (start with lower case, no blanks).

Also, labels in curly brackets associated with an attribute or association end (example: "names : CharacterString [0..*] {unique, ordered}") are parsed and represented in appinfo elements in the XML Schema. Note that this replaces the support for the collection types (Set<T> etc) defined in the current version of ISO/TS 19103, but which are expected to be removed in the revision of this standard.

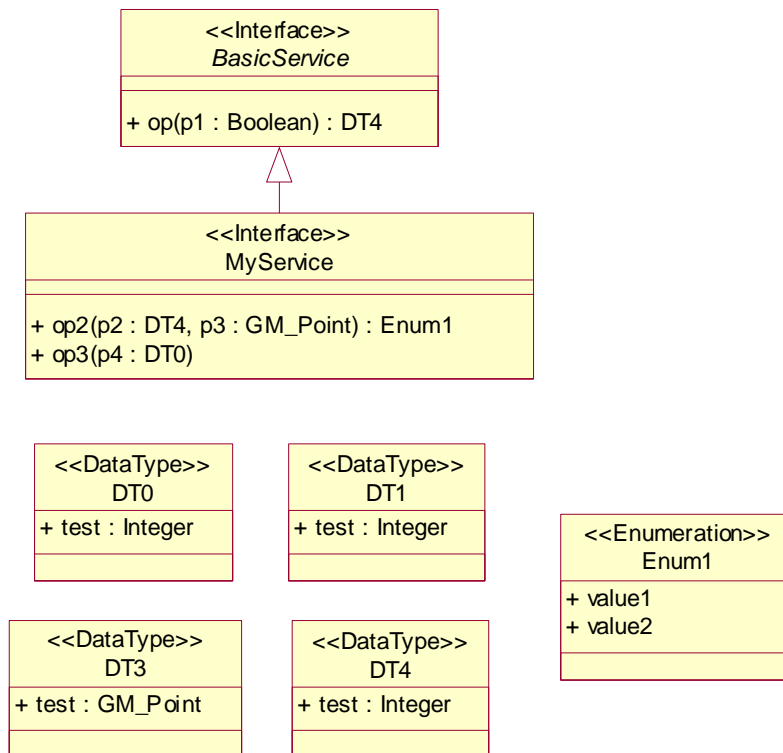
3.2.17 Extension: WSDL definitions from <<Interface>> types

This version supports an experimental extension to create WSDL definitions for types with the stereotype <<Interface>> and which represent a service interface.

The following encoding rules apply:

- For every operation of an <<interface>> type, a wsdl:message with the name of the operation plus "Request" is created. Every parameter is a wsdl:part with name and type.
- For every operation of an <<interface>> type with a return value, a wsdl:message with the name of the operation plus "Response" is created. The return value is a wsdl:part with name "return" and type.
- For every <<interface>> type, a wsdl:portType is created. One wsdl:operation with the name of the operation and input (plus output if not "void") is created based on the messages created above.
In this process, also wsdl:fault elements are created and associated with the port. The names of the types representing the fault are attached to the operation as a tagged value "faultTypeName" as a space-separated list, e.g. "OA_IllegalQuery OA_InternalError".
- The wsdl:binding is in this version always HTTP SOAP and "document"/"literal" for all operations.
- For every <<interface>> type, a wsdl:service and one wsdl:port based on the wsdl:portType created above is created.
- The XML Schema definition of the application schema is imported (note: a known open issue that all namespaces of types used in the WSDL definition still have to be added as xmlns attributes)

As an example, the types in the class diagram below (from application schema "WSDL Test") are mapped to the WSDL definitions shown after the figure.



BasicService.wsdl:

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:t2="urn:x-test:t2" xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/" name="BasicService"
targetNamespace="urn:x-test:t2">
  <types>
    <schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:x-test:t2" xmlns:t2="urn:x-
test:t2">
      <include schemaLocation="wsdltest.xsd"/>
    </schema>
  </types>
  <message name="opRequest">
    <part name="p1" type="boolean"/>
  </message>
  <message name="opResponse">
    <part name="return" type="t2:DT4PropertyType"/>
  </message>
  <portType name="BasicService">
    <operation name="op">
      <input message="t2:opRequest" name="opRequestRequest"/>
      <output message="t2:opResponse" name="opRequestResponse"/>
    </operation>
  </portType>
  <binding name="BasicService" type="t2:BasicService">
    <binding xmlns="http://schemas.xmlsoap.org/wsdl/soap/" style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="op">
      <operation xmlns="http://schemas.xmlsoap.org/wsdl/soap/" soapAction="op" style="document"/>
      <input>
        <body xmlns="http://schemas.xmlsoap.org/wsdl/soap/" use="literal"/>
      </input>
      <output>
        <body xmlns="http://schemas.xmlsoap.org/wsdl/soap/" use="literal"/>
      </output>
    </operation>
  </binding>
</definitions>
```



```
</operation>
</binding>
<service name="BasicService">
  <port binding="t2:BasicService" name="BasicService">
    <address location="http://add.service.url.here/port"/>
  </port>
</service>
</definitions>
```

MyService.wsdl:

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:t2="urn:x-test:t2" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" name="MyService"
targetNamespace="urn:x-test:t2">
  <types>
    <schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:x-test:t2" xmlns:t2="urn:x-
test:t2">
      <include schemaLocation="wsdltest.xsd"/>
    </schema>
  </types>
  <message name="opRequest">
    <part name="p1" type="boolean"/>
  </message>
  <message name="opResponse">
    <part name="return" type="t2:DT4PropertyType"/>
  </message>
  <message name="op2Request">
    <part name="p2" type="t2:DT4PropertyType"/>
    <part name="p3" type="gml:PointPropertyType"/>
  </message>
  <message name="op2Response">
    <part name="return" type="t2:Enum1Type"/>
  </message>
  <message name="op3Request">
    <part name="p4" type="t2:DT0PropertyType"/>
  </message>
  <message name="op3Response"/>
  <portType name="MyService">
    <operation name="op">
      <input message="t2:opRequest" name="opRequestRequest"/>
      <output message="t2:opResponse" name="opRequestResponse"/>
    </operation>
    <operation name="op2">
      <input message="t2:op2Request" name="op2RequestRequest"/>
      <output message="t2:op2Response" name="op2RequestResponse"/>
    </operation>
    <operation name="op3">
      <input message="t2:op3Request" name="op3RequestRequest"/>
    </operation>
  </portType>
  <binding name="MyService" type="t2:MyService">
    <binding xmlns="http://schemas.xmlsoap.org/wsdl/soap/" style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="op">
      <operation xmlns="http://schemas.xmlsoap.org/wsdl/soap/" soapAction="op" style="document"/>
      <input>
        <body xmlns="http://schemas.xmlsoap.org/wsdl/soap/" use="literal"/>
      </input>
      <output>
        <body xmlns="http://schemas.xmlsoap.org/wsdl/soap/" use="literal"/>
      </output>
    </operation>
    <operation name="op2">
      <operation xmlns="http://schemas.xmlsoap.org/wsdl/soap/" soapAction="op2" style="document"/>
      <input>
        <body xmlns="http://schemas.xmlsoap.org/wsdl/soap/" use="literal"/>
      </input>
```



```
</input>
<output>
  <body xmlns="http://schemas.xmlsoap.org/wsdl/soap/" use="literal"/>
</output>
<fault name="OA_TestFault">
  <fault xmlns="http://schemas.xmlsoap.org/wsdl/soap/" name="OA_TestFault" use="literal"/>
</fault>
<fault name="OA_TestFault2">
  <fault xmlns="http://schemas.xmlsoap.org/wsdl/soap/" name="OA_TestFault2" use="literal"/>
</fault>
</operation>
<operation name="op3">
  <operation xmlns="http://schemas.xmlsoap.org/wsdl/soap/" soapAction="op3" style="document"/>
  <input>
    <body xmlns="http://schemas.xmlsoap.org/wsdl/soap/" use="literal"/>
  </input>
</operation>
</binding>
<service name="MyService">
  <port binding="t2:MyService" name="MyService">
    <address location="http://add.service.url.here/port"/>
  </port>
</service>
</definitions>
```