mdWFS – Model-driven Schema Translation for Integrating Heterogeneous Geospatial Data

Tatjana Kutzner, Andreas Donaubauer

Chair of Geoinformatics
Technische Universität München
kutzner@tum.de

08.10.2013
INSPIRE KEN Schema Transformation Workshop, Marne la Vallée
Theoretical Foundations of the Schema Translation Approach
Geospatial Data Modelling – Modelling Levels

Modelling levels

M3
- Meta-metamodel
- MOF

M2
- Metamodel
- UML Metamodel/
UML Profile

M1
- Model
- INSPIRE
Data Specifications

M0
- Data about real
world objects
- GML file

Conceptual
Schema

Transfer format
Use of Conceptual Schemas in Geospatial Data Modelling

► Schemas for Communication Purposes
  ● Support communication between people from different backgrounds
  ● Create common understanding
  ● e.g. INSPIRE Data Specifications

→ Schemas have to be **machine-readable**

► Schemas for Controlling Run-time Systems
  ● Derive transfer format / database schema from conceptual schemas
  ● Transform between different schemas
  ● e.g. transformation from INSPIRE Data Specifications (PIM) to INSPIRE compliant GML (PM)

→ Schemas have to be **machine-interpretable**
Model-driven Architecture (MDA)

- Software development through definition of formal schemas and transformation between them
- Separation of platform-independent from platform-specific aspects

<table>
<thead>
<tr>
<th>Types of schema levels in MDA</th>
<th>Relation to Geospatial Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform Independent Model (PIM)</td>
<td>AFIS-ALKIS-ATKIS Reference Model (UML)</td>
</tr>
<tr>
<td>Platform Specific Model (PSM)</td>
<td>AFIS-ALKIS-ATKIS Implementation Schema (UML)</td>
</tr>
<tr>
<td>Platform Model (PM)</td>
<td>AFIS-ALKIS-ATKIS Transfer Format NAS (GML)</td>
</tr>
</tbody>
</table>

- Automatic execution of transformation between schemas (e.g. PIM → PSM, PIM → PIM) by transformation tools
- Relevant for web-based schema translation, e.g. INSPIRE Transformation Service

→ Prerequisite: Machine-interpretable schemas
Conceptual Schemas and Semantic Transformation

- Transformation on Conceptual Schema Level (PIM → PIM)
  - Definition of mapping rules between source and destination schema
  - Automatic execution of the defined rules
  - Rules are defined just once, afterwards transformation into various formats through encoding rules
  - Modelling expert doesn’t need to be programming expert
Model-driven Schema Translation in the Lake Constance Region
Projects Using this Approach

- Model-driven Approach for Accessing Distributed Spatial Data Using Web Services – Demonstrated for Cross-border GIS Applications (mdWFS), 2006-2011
  - Sponsors:
    - Federal Agency for Cartography and Geodesy (BKG), Germany and
    - Swiss Federal Office for Topography (swisstopo)

- Prototypical Transformation of Spatial Data to INSPIRE in the Cross-border Lake Constance Region (Geodatenpool Bodensee), 2010-2012
  - Sponsors:
    - State Agency for Spatial Information And Rural Development Baden-Wuerttemberg (LGL BW),
    - Bavarian Agency for Surveying and Geoinformation (LVG BY),
    - Austrian Federal Office of Meterology and Surveying (BEV) and
    - Swiss Federal Office for Topography (swisstopo).
## Schemas from the Lake Constance Region

### Germany: AFIS-ALKIS-ATKIS Reference Model
- Modelling language: UML 1.4.2 + UML Profile from ISO 19103
- Metamodel: General Feature Model ISO 19109
- Transfer format: NAS, ISO 19136 Annex E / ISO 19139 encoding

### Switzerland: TLM Model, MOpublic Model
- Modelling language: INTERLIS 2
- Metamodel: UML 1.4.2 Metamodel extension + INTERLIS Metamodel
- Transfer format: Interlis 1, Interlis 2

### Austria: -
- Transfer format: Shape, ASCII

### Europe: INSPIRE Data Specifications
- Modelling language: UML 2.1 + UML Profile from ISO 19103
- Metamodel: General Feature Model ISO 19109
- Transfer format: GML, ISO 19136 Annex E / ISO 19139 encoding
mdWFS – Model-driven Web Feature Service

Goal:
- Translation of spatial data through defining the schema translation on the level of the conceptual schemas
- Embedding the model-driven translation of spatial data in a web-based environment
- Demonstrating the feasibility of this approach

Realisation of this approach includes:
- Development of the transformation language UMLT for defining schema mappings
- Development of plugins for FME
  - XMI Reader: Reads UML schemas in XMI format
  - Transformer UMLTApplier: Parses UMLT mapping rules in XMI format and translates them into internal FME functions
- Definition and implementation of a web interface (transformation service) for executing schema translation
mdWFS – General Concept

1. Request information on source schema [XMI]

2. Define transformation rules [XMI]

3. DoTransform (Source schema [URL], target schema [XMI], transformation rules [XMI])

4. GetFeature (feature type)
   Spatial data conforming to target schema

mdWFS

Configure

Schema translation

New WFS instance

Configure

configure

execute

configure

Spatial data conforming to target schema

URL of the new WFS
UML Transformations (UMLT)

- Mapping language for describing mappings between UML schemas
- Based on UML2 Activity Diagrams, developed as independent extension of the UML2 meta model at ETH Zürich
- A UMLT editor was developed at TUM for visual definition of the mappings and export as XMI transfer format for further processing

Example: Transformation of DKM (Austrian cadastre) to INSPIRE Cadastral Parcels

```plaintext
cp_1.geometry := gst_4.ausdehnung
cp_1.inspireId.namespace := "AT.0002.01.03"
cp_1.label.elements := gst_4.gnr.elements
cp_1.areaValue.area.value := gstdb_3.Flaeche
cp_1.areaValue.area.uom.uomSymbol.elements := "m2"
cp_1.referencePoint.gm_point := gnr_1.position
cp_1.beginLifespanVersion.voidReason := "Unpopulated"
```
Implementation of the Schema Translation in FME

Workspace Resource
- The target schema which has been read using the XMI Reader

UMLTApplier
- Converts UMLT mapping rules [XMI] into internal FME functions

Generic Writer
- Uses the target schema provided as Workspace Resource
Implementation of the mdWFS Web Interface

- mdWFS operations are provided as WPS processes

Implementation

- mdWFS GetCapabilities and DescribeFeatureType: Java
- DoTransform: mdWFS.fmw, provided via FME Server
Findings from the prototypical implementation
Problems arising from the state-of-the-art use of conceptual schemas (I)

► Discrepancy in visual and machine-interpretable representation of UML due to

- identical data type names with different semantics (e.g. ISO 19103 CharacterString ≠ EA CharacterString)
- disregarding the UML specification (e.g. code list with generalization relationship)
- different UML versions (e.g. UML 2.1 Boolean ≠ ISO 19103 Boolean)

→ MDA cannot be applied consistently in modelling and in the automatic transformation process since UML models are not fully machine-interpretable

- More accuracy in defining schemas and UML profiles is required
Discrepancy UML visual/machine-interpretable

Semantics of Data types

Visual representation is identical
XMI representation is differing

→ Type: EAJava_CharacterString
→ Type: CharacterString as complex type with the attributes „size“, „characterSet“, „elements“, „maxLength“
Problems arising from the state-of-the-art use of conceptual schemas (II)

- Non-machine-interpretable schemas due to semantic modification of the UML specification (e.g. stereotypes «codelist», «union» and «voidable»)
  - Schemas are not compliant with the UML profile definition of the OMG

- Schemas are based on different UML profiles
  - D: ISO-19103-based UML profile, UML version 1.4.2
  - CH: INTERLIS 2 UML profile, not ISO-based, but ISO conform
  - INSPIRE: ISO-19103-based UML profile, UML version 2.1
  - Possible other profiles from administrations using geospatial base data

- In the mdWFS project transformation was only possible using a common profile, i.e. the UML schemas of AAA and INSPIRE had to be remodelled according to the INTERLIS profile
Semantic Modification of the UML Specification

► Stereotypes from ISO 19103

«codeList»: extensible value range

<table>
<thead>
<tr>
<th>AdministrativeHierarchyLevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ 1stOrder</td>
</tr>
<tr>
<td>+ 2ndOrder</td>
</tr>
<tr>
<td>+ 3rdOrder</td>
</tr>
<tr>
<td>+ 4thOrder</td>
</tr>
<tr>
<td>+ 5thOrder</td>
</tr>
<tr>
<td>+ 6thOrder</td>
</tr>
</tbody>
</table>

«union»: exactly one of several alternative values during run-time

<table>
<thead>
<tr>
<th>AA_UUID</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ UUID: CharacterString</td>
</tr>
<tr>
<td>+ UUIDAndDundZeit: CharacterString</td>
</tr>
</tbody>
</table>

→ These stereotypes represent a not allowed extension of the UML language

→ ISO 19103 is not a real UML profile in the sense of the UML profile definition of the OMG
Semantic Modification of the UML Specification

- **Stereotype «voidable» from INSPIRE**

  - **According to UML:** Exactly one value of type „DateTime“
  - **With «voidable»:** Also „unpopulated“ and „unknown“ as value

  ![UML diagram showing stereotype «voidable»](image)

  - Not conforming to the UML Specification
Enhanced prototypical implementation: Multi-level schema translation
Approach for solving the UML profile problem

► Use of a Core UML Profile

- Represents an intersection of the elements of the individual profiles; a comprehensive UML profile, which contains elements not common to each profile, could constrain extensive use of spatial data beyond the GI domain.
- The concepts of the core UML profile have to comply to the UML specification for profiles.

► Multi-level semantic translation

- The individual profiles are mapped to the common Core UML Profile as an intermediate step in transforming source into destination schemas.
- The Core UML Profile acts as pivot at the meta model level (star converter approach).
- Advantage: The mapping rules have to be defined only once for each UML profile. Each UML schema which is based on a certain UML profile can then be mapped to a schema based on the Core UML profile.
**Formal Definition of the Core UML Profile**

```
class CoreUMLProfile

  «metaclass» Package
    + URI : String

  «extends» Leaf
    constraints {Leaf}

  «extends» FeatureType

  «extends» Union
    constraints {Union}

context Leaf inv:
  base_Package.nestedPackage->size()=0

context Union inv:
  base_Class.ownedAttribute->select(p|p.association->notEmpty())->size()>=2
  and
  base_Class.ownedRule->notEmpty()
```
Mapping of «CodeList» to Core UML Profile (I)

► Use of «CodeList» according to ISO 19103 results in the following XMI:

```xml
<packagedElement xmi:type="uml:Class" xmi:id="12" name="AdministrativeHierarchyLevel" visibility="public">
  <ownedAttribute xmi:type="uml:Property" xmi:id="…" name="1stOrder" visibility="public">
    <lowerValue xmi:type="uml:LiteralInteger" xmi:id="…" value="1"/>
    <upperValue xmi:type="uml:LiteralInteger" xmi:id="…" value="1"/>
  </ownedAttribute>
  <ownedAttribute xmi:type="uml:Property" xmi:id="…" name="2ndOrder" visibility="public">
    <lowerValue xmi:type="uml:LiteralInteger" xmi:id="…" value="1"/>
    <upperValue xmi:type="uml:LiteralInteger" xmi:id="…" value="1"/>
  </ownedAttribute>
  <ownedAttribute xmi:type="uml:Property" xmi:id="…" name="3rdOrder" visibility="public">
    <lowerValue xmi:type="uml:LiteralInteger" xmi:id="…" value="1"/>
    <upperValue xmi:type="uml:LiteralInteger" xmi:id="…" value="1"/>
  </ownedAttribute>
...
</packagedElement>

<thecustomprofile:CodeList base_Class="12" />
```

<table>
<thead>
<tr>
<th>«CodeList» AdministrativeHierarchyLevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ 1stOrder</td>
</tr>
<tr>
<td>+ 2ndOrder</td>
</tr>
<tr>
<td>+ 3rdOrder</td>
</tr>
<tr>
<td>+ 4thOrder</td>
</tr>
<tr>
<td>+ 5thOrder</td>
</tr>
<tr>
<td>+ 6thOrder</td>
</tr>
</tbody>
</table>
Modelling of «CodeList» as extension of Enumeration results in the following XMI:

```xml
<packagedElement xmi:type="uml:Enumeration" xmi:id="12" name="AdministrativeHierarchyLevel" visibility="public">
  <ownedLiteral xmi:type="uml:EnumerationLiteral" xmi:id="…" name="1stOrder" visibility="public"/>
  <ownedLiteral xmi:type="uml:EnumerationLiteral" xmi:id="…" name="2ndOrder" visibility="public"/>
  <ownedLiteral xmi:type="uml:EnumerationLiteral" xmi:id="…" name="3rdOrder" visibility="public"/>
  ...
</packagedElement>
<theCustomProfile:CodeList base_Enumeration="12"/>
```

<table>
<thead>
<tr>
<th>AdministrativeHierarchyLevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>1stOrder</td>
</tr>
<tr>
<td>2ndOrder</td>
</tr>
<tr>
<td>3rdOrder</td>
</tr>
<tr>
<td>4thOrder</td>
</tr>
<tr>
<td>5thOrder</td>
</tr>
<tr>
<td>6thOrder</td>
</tr>
</tbody>
</table>
Multi-level semantic transformation

1. Metamodel Mapping Rules (UML Profile S) executes ATL
2. Metamodel Transformation Tool
   - Schema S' reads and displays
   - Mapping
   - Format S' takes into consideration
   - Encoding Rules S'
   - Encoding Rules S
   - Mapping
   - FME / XMI Reader
3. Editor
4. Schema Translation Tool
   - Schema Translation Rules TR'
   - Mapping
   - FME / UMLTApplier
5. Encoding Rules D'
6. Format D'
7. Format D
8. Metamodel Mapping Rules (UML Profile D)
9. Metamodel Transformation Tool
   - Schema D' reads and displays
   - Mapping
   - Encoding Rules D
   - FME / UMLTApplier
10. Editor
11. UMLT Editor
12. Core UML Profile
13. MetaModel Transformation Tool
14. Metamodel Mapping Rules
15. UML Profile S
16. UML Profile D

Compliant to:
- Schema S
- Format S
- Encoding Rules S
- Mapping
- Format S'
- Encoding Rules S'
- Mapping
- Format D'
- Encoding Rules D'
- Mapping
- Format D

Integration of Heterogeneous Geospatial Data
Conclusions

► The mdWFS approach can be applied successfully for
  ● defining the schema translation on the conceptual schema level
  ● deriving concrete transformations between different transfer formats

► Multi-level schema translation provides a solution to the problem of UML profiles not conforming to the UML specification

► Future work
  ● Developing a more automatic solution for the encoding of the data, which is currently done manually by FME workspaces due to missing machine-interpretable encoding rules
  ● Testing the approach with further UML schemas to analyse the mapping of UML schemas to the core UML profile for a wider range of UML constructs