Title:
A5.2-D3 3.3.1 Alignment Editor Specification
Editor(s)/Organisation(s):
Thorsten Reitz (Fraunhofer IGD)
Contributing Authors:
Thorsten Reitz (Fraunhofer IGD), Marian de Vries (TUD)
References:
A1.8-D4 User Involvement Document - Second Version (555/3) A5.2-D3 [3.1] Specification Introduction and Overview A5.2-D3 [3.3] Conceptual Schema Specification and Mapping

Status/Quality Assurance:	
☑ Document Outline	☐ Review WP Leader
☑ Content Draft	☐ Review dependent WP Leaders (WP08)
☑ Request for Comments	☐ Review ExB (_____)
☑ Final	☐ Review Associated Projects (_____)

Short Description:
This document provides an introduction to the HUMBOLDT Alignment Editor.
Keywords:
Framework specification, logical architecture, physical architecture, requirements, use cases.

Delivery Date:	**02.12.09**

History:			
Version	Author(s)	Status	Comment
001-381	Thorsten Reitz	FINAL	Finalised version extracted from Wiki.

Table of contents

Figures

1 Introduction

1.1. Purpose of this document

This document describes an application that is part of the HUMBOLDT Harmonization Toolkit, specifically the HUMBOLDT Alignment Editor [2.3.2] (→ *and as such it is part of Deliverable A5.2-D2*). Beside outlining the purpose of the application and it's collaboration with other components and applications, it contains information on the internal data models and message structures used by it. Data structures that it shares with other components are part of the HUMBOLDT Commons specification (→ *Document [2.1]*), which also represents Deliverable A5.3-D2.

The HUMBOLDT Alignment Editor, short HALE, is a rich graphical user interface for defining mappings between concepts in conceptual schemas (application schemas created with the HUMBOLDT Model Editor), as well as for defining transformations between attributes of these schemas. These mappings are expressed in a high-level language and can later be used by the Conceptual Schema Transformer processing component to generate an executable form of transformation, such as an XSLT for XML input/output.

To make this complex process more accessible to a domain expert and to increase the quality of transformations, HALE allows working with sample instances for visualization and validation. Furthermore, a sophisticated task-based system as it is often used in programming supports users in the creation of a mapping.

1.2. Abbreviations used in this document

This section summarizes the abbreviations used specifically for this service component. It does not repeat information found in the introduction and specification overview document [3.1].

Abbrev.	*Name*	*Definition*
CST	Conceptual Schema Transformer	The module that executes the mappings previously defined with the HUMBOLDT Alignment Editor.

1.3. Definitions valid in this document

For definitions of the terms used in this document, please refer to document [2.3], which provides the framework for conceptual schema translation components in HUMBOLDT. It should be noted though that the term FeatureType is used synonymously to Spatial Object Type (INSPIRE wording) and Concept (Ontology Engineering).

1.4. Standards used in this document

The Alignment Editor makes use of some of the core standards in geoinformation for the provision of maps and other products, as well as raw data. Most notably, these are:

1.4.1 OMWG Ontology Mapping Language (OML/D7.2)

The Ontology Mapping Language specified by the Ontology Mapping Working Group provides means to describe both assertions (such as subsumption and equality relationships) and transformations. It is being used to express conceptual-level transformations and is used for the exchange and storage of those.

1.4.2 OGC Web Feature Service (WFS)

The Web Feature Service is used for provisioning of instance data to HALE and for accessing schema information via `DescribeFeatureType` operations.

1.4.3 OGC Geographic Markup Language (GML)

GML 3.2.1 parts are used for the core data model of the Alignment Editor.

1.4.4 OGC Web Processing Service (WPS)

The WPS is used as a basic interface for the Transformer specification, which is used as the interface from the Alignment Editor to the Conceptual Schema Transformer.

1.4.5 ISO 19110:2005 Methodology for Feature Cataloguing

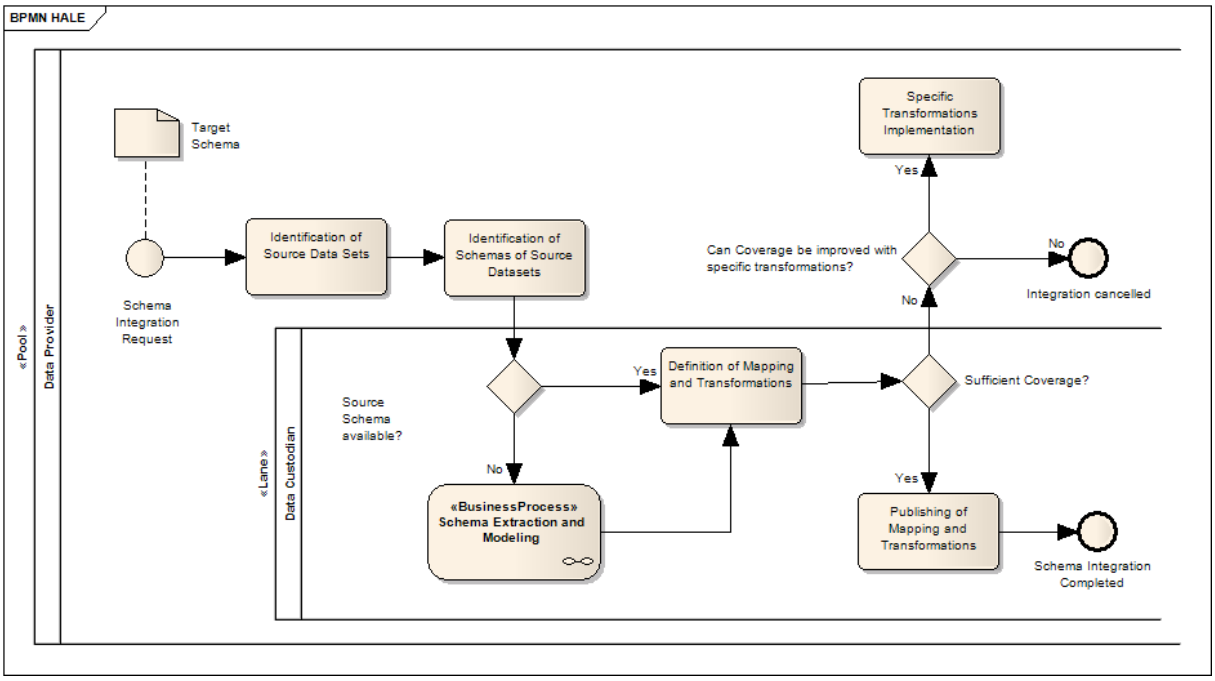
The structures presented in this standard are used to model conceptual schemas in the HUMBOLDT framework.

2 Enterprise viewpoint

The Enterprise viewpoint describes the functional purpose of the component and it's integration with business processes. This chapter makes use of BPMN and of UML 2.0 Use Cases.

2.1. Business process overview

The Alignment Editor supports the core harmonisation activity of defining schema translations. It is therefore used in harmonisation preparation as figure 1 shows.



The Alignment Editor's place is therefore rather early in the overall harmonisation business process. The work done with it has to be done prior to actually harmonising geodata, just as the definition of target schemas via the HUMBOLDT Model Editor or another tool had to be done prior.

The value and unique selling proposition that HALE provides to DATA_CUSTODIANs is manifold:

1. It makes use of both conceptual schemas and of geographic instances to ensure high-quality mappings;
2. It is based on a powerful conceptual-level mapping and transformation model that is based on developing standards;
3. It provides a rich, textual and graphical Domain-Specific Language specifically adopted for GI Experts;
4. It gives instant feedback about the progress of mapping data from one schema to another;

2.2. Application in the Protected Areas Scenario

This component plays a role for all scenarios, since it addresses a core requirement to capture domain expert knowledge required for geospatial data harmonisation. This harmonisation is a driver behind most Use Cases, but for clarity's sake, one specific use case from one scenario was selected to show the usage of HALE in application. This use case is the “Tourism Valorisation” case from the protected areas scenario:

Mario has created the complete application schema for his protected areas management applications. It is a cross-theme schema that includes quite different INSPIRE themes:

- *From Annex I: Administrative units (1:5.000), Transport networks (Railway Network and Roads, 1:10.000), Hydrography (1:10.000), Protected Sites (1:25.000)*
- *From Annex II: Land-use and land-cover (1:25.000), Elevation, Orthoimagery*
- *From Annex III: Bio-geographical regions, Habitats and biotopes, Species distribution*

Furthermore, there are some specific data sets that are not represented in INSPIRE themes:

- *Burnt Areas (1:10.000),*
- *Monument Trees,*
- *Stopping Places,*
- *Foot-paths and Hiking Trails,*
- *and a regional topographic map (1:25.000) that is used as a backdrop.*

Now he needs to transform the individual data sets he already has into the harmonized schema. To do so, he switches from the HUMBOLDT Model Editor to HALE and first of all loads his scenario application schema based on the INSPIRE models, such as the Hydrography model, as the target model. The target model is now being shown in a tree form in one part of the UI:

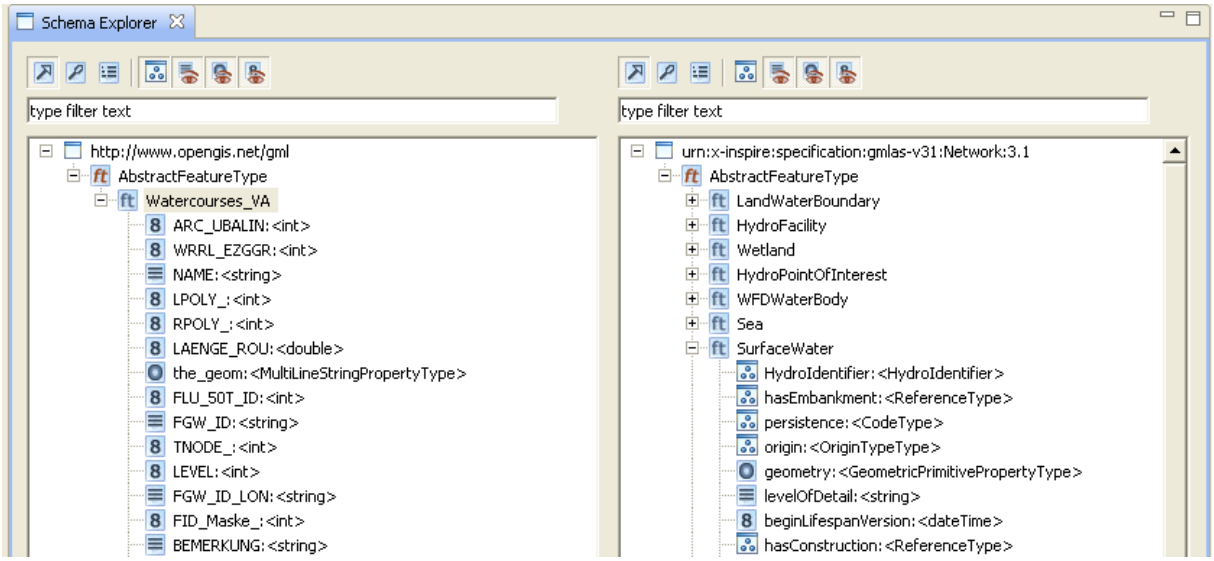


Figure 2: Exploring loaded schemas in the Alignment Editor's Schema Explorer View

The next thing he does is to select the WFS providing the Protected Sites data for one of the regions. The system retrieves the schema for this input data set and displays it's FeatureTypes in a view similar to the one for the harmonized target model. If no schema can be identified for the data set, Mario will be asked by the system to provide one manually.

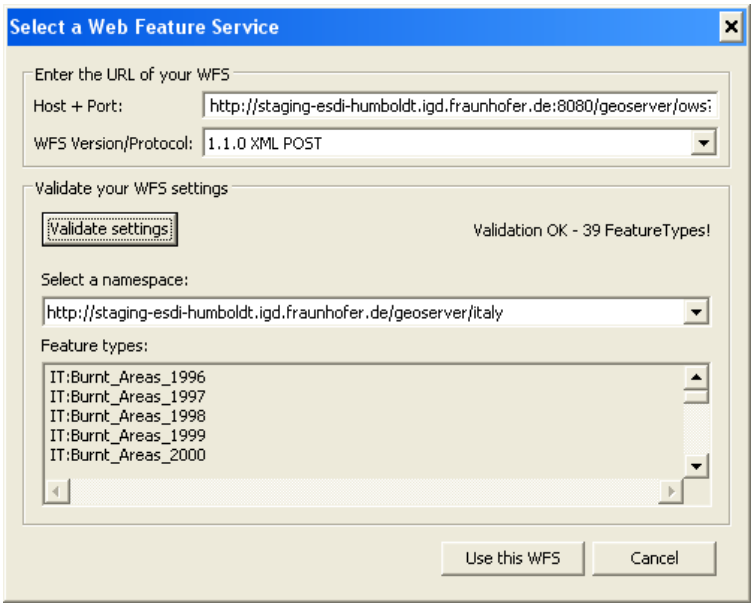


Figure 3: This screenshot shows the dialogue used to load a source schema from a Web Feature Service's DescribeFeatureType operations.

After this step is completed, Mario can instruct HALE to download a sample of the data provided by the WFS to analyze it's attribute values and to visualize it in it's map component.

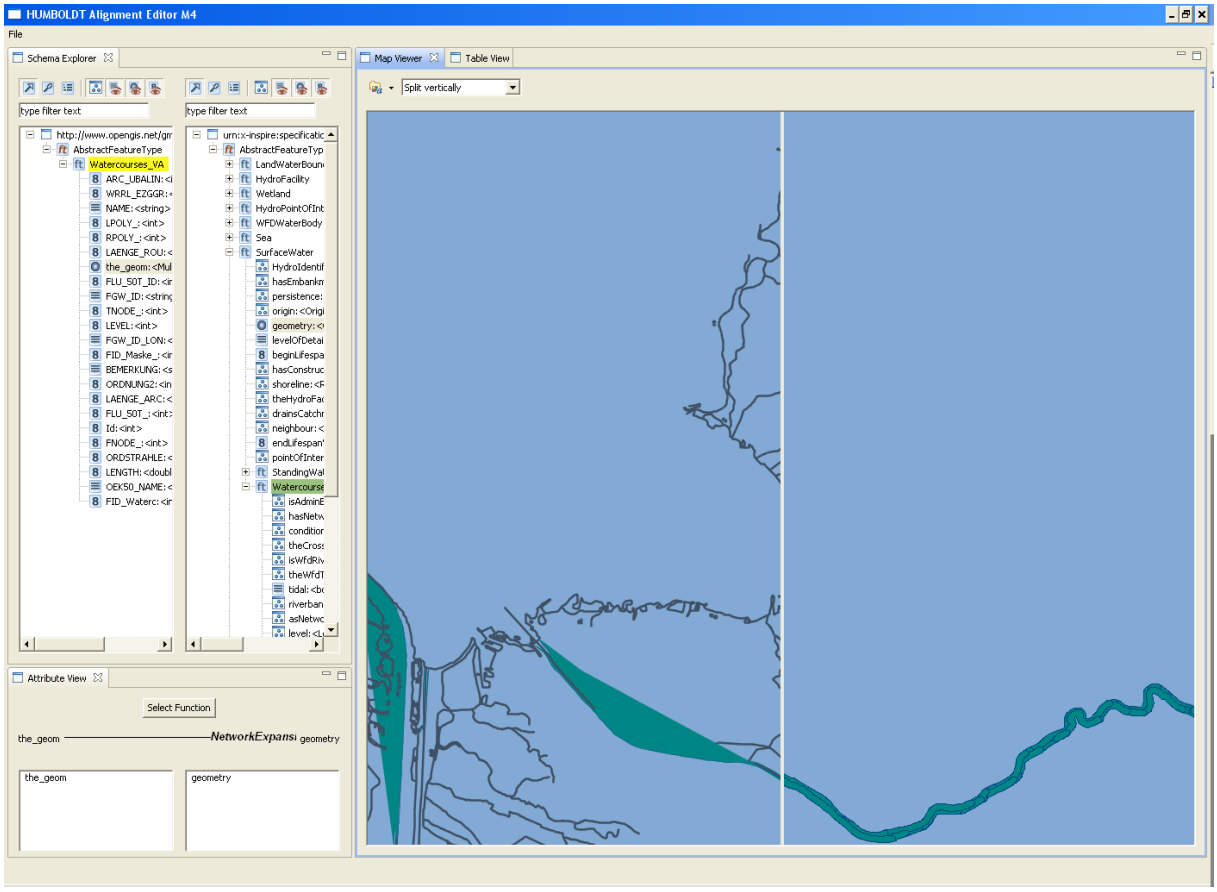


Figure 4: This figure shows the main view of HALE after the loading of target and source schemas and accompanying data sets.

In a next step, Mario can set some quality control parameters for his mapping projects, including completeness and consistency requirements. This is done in the Analysis Configuration dialogue.

After the analysis is completed, HALE presents him with a list of tasks: Each one represents the connection of one FeatureType (Spatial Object Type) of the source WFS to corresponding types in the harmonised application schema. The tasks are ordered according to their value, which is derived from the number of instances of a type and a few other parameters.

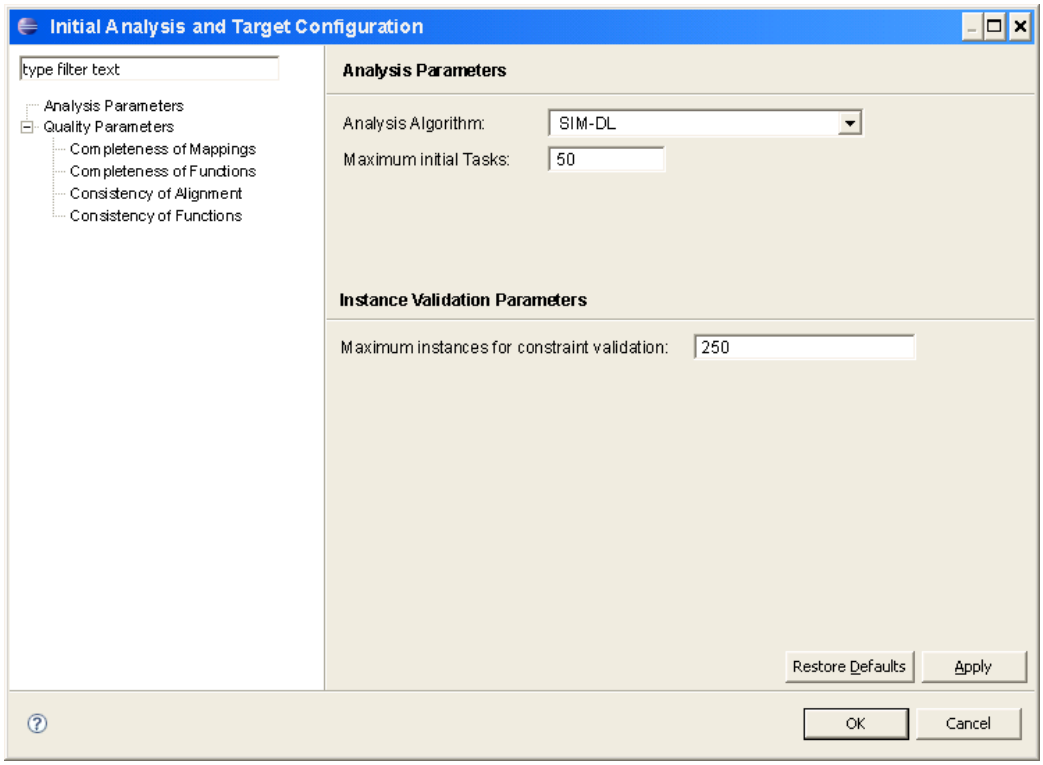


Figure 5: This figure shows the Analysis Setup Dialogue. It allows to specify quality constraints and to set up the internal validation processes used during mapping.

Mario therefore starts with the task with the highest value. He double-clicks it, and the corresponding source FeatureType is selected and shown in the lower left corner of the GUI, together with geometric and other attributes. Mario can now select one or multiple FeatureTypes of the target application schema. He can also add additional source FeatureTypes, if the mapping tasks represents a union of multiple FeatureTypes to one.

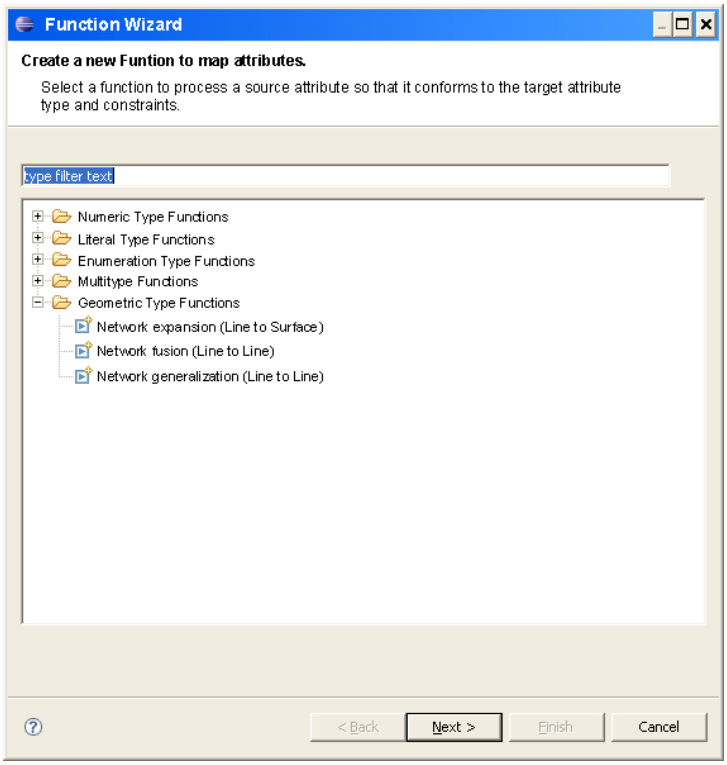


Figure 6: This figure shows the first dialogue of the Function Wizard, which enables users to select and configure an attribute transformation, such as a Geometric Type Transformation.

After doing so, he adds an attribute transformation: The geometry type of the selected FeatureType in the WFS is a MultiPolygon, but his harmonized application schema needs a simple Polygon per Feature. Also, he adds transformations and simply copying operations for numeric and textual informations.

After completing the mapping of all relevant attributes into the target model, he closes the task. Next, the system performs a few sanity checks and updates the task list if necessary. This can be the case if mappings have been established that contradict each other, or attribute transformations that would not work on the sample data, or transformations that would overwrite each other.

Mario can now select the next task and go through the process of defining relationships between FeatureTypes and mapping their attributes until he has mapped everything from the source WFS that he needs. He can now save the mapping.

Now, Mario has to repeat the exercise for the data provided by the next region, but this time, it's already easier. He can now use the data set he worked on before as a target reference data set. In this way, he gets a direct visual comparison of the non-harmonised data set and what it should look like in the end. Also, he can re-use parts of the mapping created beforehand, since parts of the source schema are also identical.

After a full day of work, he has created the mappings for the four regions involved and looks forward to seeing the transformed data work together….

The full process behind this user experience involves several additional steps, which are described in Chapter 3 of the Specification Introduction and Overview Document [3.1].

2.3. Actors in this component

This component is mostly important to DATA_CUSTODIANs and DATA_INTEGRATORs, with DATA_CUSTODIANs being the core target group and DATA_INTEGRATORs supporting them and their work with the alignment editor.

The actors for this component are consequently as follows:

Actor Name	***Actor Description***
SYSTEM_INTEGRATOR	This is the standard system integrator actor from the User Involvement document.
DATA_CUSTODIAN	This is the standard data custodian actor from the User Involvement document.

No additional component-specific actors are introduced.

2.4. Alignment Editor Use Cases

This section summarizes the Use Cases which this application has to fulfil. The Alignment Editor is a standalone application that can be used on its own, but it will benefit from being used together with a Model Repository service. Some of the Use Cases below assume the availability of such a Model Repository as a precondition. Furthermore, all Use cases have the precondition that source and target schema have been documented in some way, being explicit by providing a GML application schema or similar document or implicit by providing a logical schema description with the data source itself.

The main group of Use Cases relates to the Data Custodian actor, who needs to be supported in creating high-quality mappings that allow complex structural modifications, reclassifications and geometric transformations:

- *HALE01: Import source and target schemas:* This operation allows the user to select a specific source and target schema, and to import it into HALE. Possible sources include WFS, GML Application Schemas, `ecore` and others.
- *HALE02: Import reference data and source data:* This optional step allows the user to let the system perform an analysis of an actual data set represented by the source schema and to load it partially or fully. For comparison purpose, an additional reference data set can be loaded that has already been transformed to the target schema.
- *HALE03: Browse and filter schemas:* This use case describes ways that schemas can be viewed, sorted, organised and filtered. Example include hierarchical views (by inheritance or aggregation) and simple orderings, such as by name or by frequency of existing instances.
- *HALE04: Browse and filter source and reference data:* This use case describes means to interact with source and reference instance data, such as via a map or a table view. Specific to this use case is the possibility to attach different Styled Layer Descriptors (SLDs) to the source and target schema, which can be used to visualize harmonisation progress.
- *HALE05: Define a Class-Level Mapping:* This use case allows a user to make an assertion about types in the source and target schema, such as specifying that two types are conceptually the same, or that one type is a subtype of another.
- *HALE06: Define an Attribute-Level Transformation:* This operation allows a user to define a transformation between the attributes of one or more types of the source schema and the attributes of one or more types of the target schema.
- *HALE07: Configure Alignment Editor:* This use case allows the user to set up his specific quality models and some general settings, such as the size of the instance pool to use and the optional usage of the task-based view.

There are currently no in-detail use case descriptions provided. Instead, there is a detailed set of requirements as listed in the next section.

2.5. Requirements

1. *I/O Functional Requirements*
    1. The user must be able to import geodata sets.
        1. The user must be able to load a geodata set from a local GML 2.1.2, 3.1 or 3.2.1 File.
        2. The user must be able to load a geodata set from a remote Web Feature Service, V1.1.

3. The user interface for accessing a WFS should support entering OpenGIS Filter Encoding 1.1 constraints.
4. The user interface for accessing local GML files could support entering OpenGIS Filter Encoding 1.1 constraints.

2. The user must be able to export mappings.
    1. The system must be capable of storing mappings in local files. For the encoding of these mappings, the RDF-XML syntax of the OMWG OML as described in [1] must be used.
    2. The system should be able to store mappings in the Model Repository component. As transfer format, the RDF-XML syntax of the OMWG OML as described in [2] must be used.
3. The user must be able to import target application schemas and attached reference data.
    1. The user must be able to load a target and a source application schema (“INSPIRE Model”) from a local XMI encoded file of the UML application schema used by the HUMBOLDT Model Editor.
    2. The user should be able to load a target application schema (“INSPIRE Model”) from the Model Repository.
4. The system should be able to load a reference data set as GML data from a local GML 2.1.2, 3.1 or 3.2.1 file or a remote WFS 1.1.
    1. The system could be able to load the fully portrayed reference data set via accessing a WMS 1.3, if the the data set is not accessible via WFS. In this case, the provision of a Target SLD is not necessary.
5. The user must be able to load a Styled Layer Descriptor (SLD 1.1) with Symbology Encoding (SE 1.1) for application on the dataset associated with both the target schema and the source schema.
    1. The user should be able to edit the loaded SLDs directly in their XML structure.
    2. The system could be able to load remote SLDs.

2. *User Interface Functional Requirements*
    1. Key Functionalities should be accessible via keyboard shortcuts and be arranged in a sensible tab order.
    2. The user interface must contain an interactive Map Component.
        1. The Map Component must support zooming and panning.
        2. The system must display the reference data set for the target schema (“INSPIRE Model”) in the Map Component, using the loaded target SLD.
        3. The system must display the source data set in the Map Component, using the loaded source SLD, if available. Otherwise, the target SLD should be used.
        4. The system must be able to synchronize interaction (panning, zooming) with the source and target reference data sets.
        5. The user must be able to select a splitting or overlay style for the two data sets displayed in the Map Component.
        6. The system should support horizontal, vertical and diagonal splitting of the Map component.

7. The system should support opaque and transparent overlay.
8. The system must update the Map Component so that the now-applicable target style is used on all Features of the mapped Feature Type of the source dataset after a new mapping has been submitted by the user.

3. The user interface should contain a Task List component.
    1. All Tasks should be of one of the three general task classes (Task, Warning, Error) and should be visually distinguished, such as by usage of an icon.
    2. The user should be able to filter the task list by category or by key word.
    3. The system should order the tasks in such a way that tasks which emerge from logical conflicts have to be resolved first. Such tasks should be highlighted as errors.
    4. All warnings and tasks emerging from the automatic schema mapping should be displayed in the Task List so that they are shortly summarized and understandable by the User.
    5. The Task List must be updated every time a user finishes a task by submitting his changes (i.e. a new or updated mapping).
4. The user interface must contain a Source Schema Explorer and a Target Schema Explorer component.
    1. The target schema should be automatically displayed in the Target Schema Explorer after loading and validation.
    2. The source schema should be automatically displayed in the Source Schema Explorer after loading and validation.
    3. The user should be able to filter elements in the Source Schema Explorer and the Target Schema Explorer by the currently selected task, by keyword (applicable on concept and attribute level), by hierarchy level or by mapping status (mapped/unmapped).
    4. The system should be able to display mappings between the schemas in the Source Schema Explorer and the Target Schema Explorer graphically.
    5. If the user selects a FeatureType from the Source Schema Explorer or the Target Schema Explorer, it must be displayed in the Mapping Source Concept component or the Mapping Target Concept component respectively.
5. The user interface must contain a Mapping Source Concept component and a Mapping Target Concept component, which are used to qualify the relationship between concepts and attributes.
    1. Establishing a mapping between concepts or attributes should be done done via a wizard.
    2. The system must be able to show all mappings established between the Mapping Source Concept and the Mapping Target Concept, and indicate their types by using different symbols.
6. The user interface must contain a menu containing two top-level entries: One for all the import and export operations and one for other operations.
7. Other operations that should be accessible via menu are editing a SLD, getting help and configuring the application.

3 Computational viewpoint

This chapter details the internal logical architecture of HALE, it's collaboration with other HUMBOLDT components and the APIs and other interfaces that it exposes.

3.1. Overview of the Alignment Editor and its relation to the CST

HALE is logically structured in two components – one is the Alignment Editor itself, the other one the bridge to the Conceptual Schema Transformer, which is needed to be able to perform on-the-fly analysis of the effect of user-defined transformations and mappings.

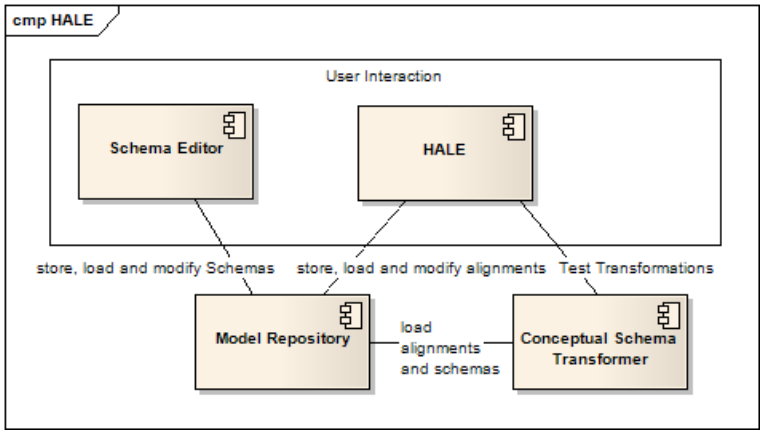


Figure 7: Interaction of HALE with other components of HUMBOLDT

Figure 7 shows the collaboration of HALE with other components of the HUMBOLDT Framework and Toolset:

1. This component is part of the Harmonisation Toolkit.
2. This component stores models and mappings using the Model Repository.
3. This component makes use of the harmonised application schemas created using the HUMBOLDT Model Editor.
4. This component embeds a version of the Conceptual Schema Transformer to assess effects of a user-defined transformation.

Internally, HALE consists of several different modules:

- A central controlling framework that synchronizes views and provides the elements for the user interface, based on the RCP framework being used;
- A Map Module that can be used to display geographic instances and the effects that a certain transformation has on them;
- A Table Module used to display tabular data including effects of transformation;
- A Schema Explorer Module that can be used to organize schemas in different ways. Ideally, this modules can be shared with the HUMBOLDT Model Editor;
- An Analysis Module that performs initial schema and instance analysis;
- A logical Reasoning Module that checks consistency of the mapping
- A internal Transformation Module that delegates to the Conceptual Schema Transformer component.

- A Task Manager Module.

All these components and their relationships are depicted in Figure 8.

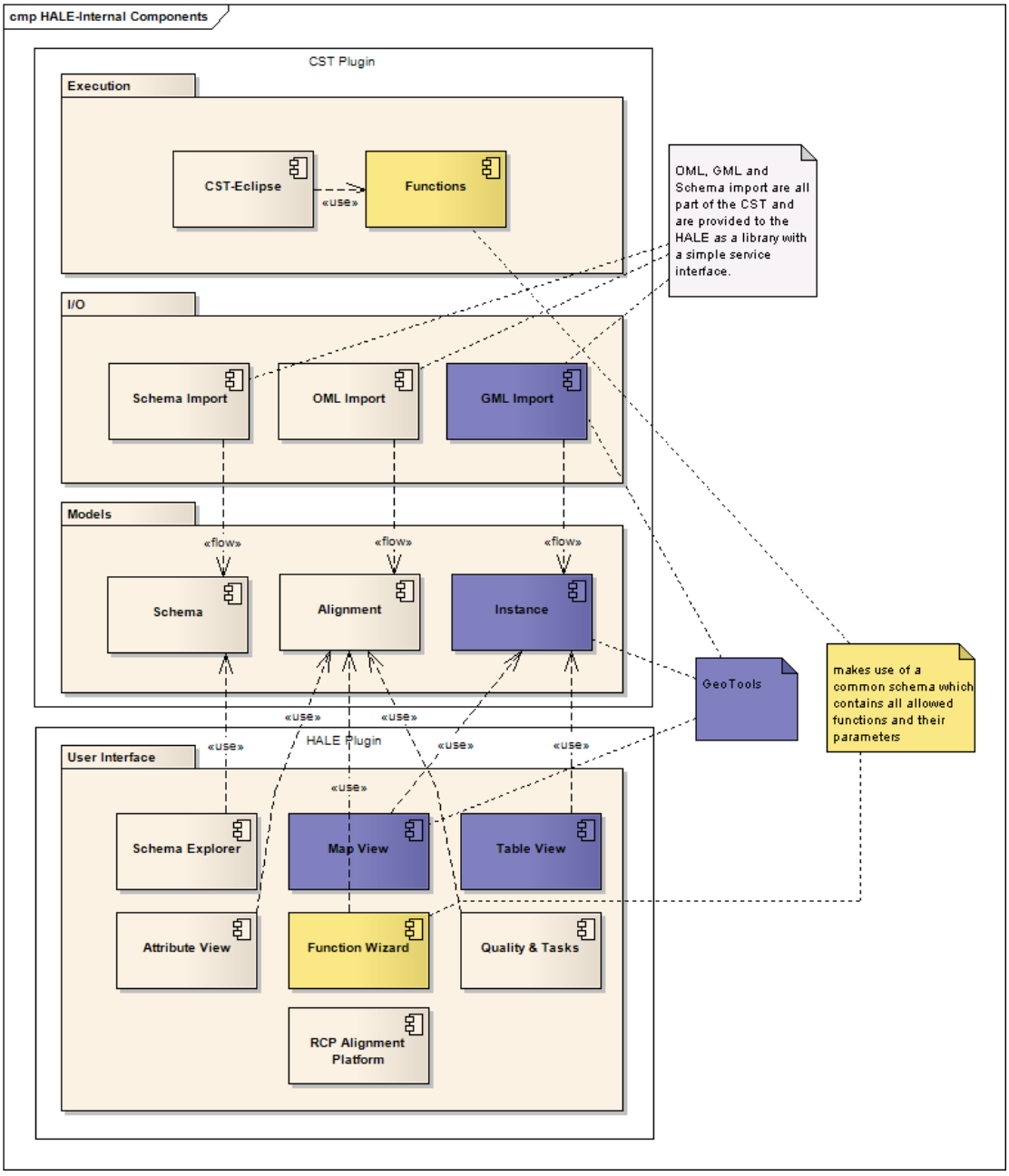


Figure 8: The internal composition of the Alignment Editor, showing the four major packages (CST Bridge, I/O, Model Services and UI/RCP Integration)

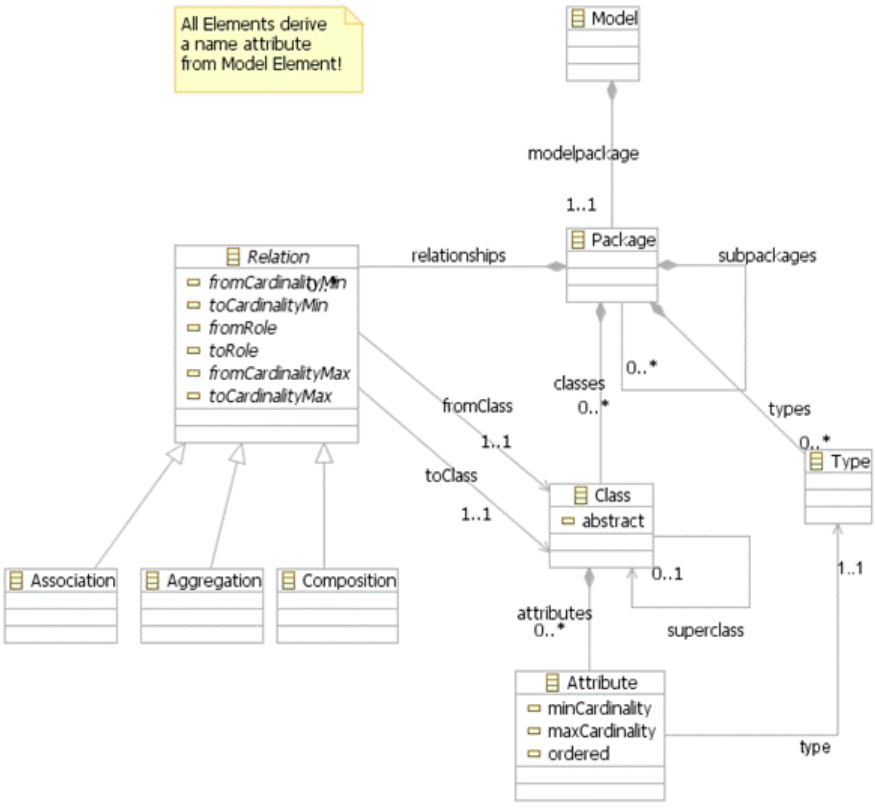
4 Information Viewpoint

This chapter describes the information viewpoint of the HUMBOLDT Alignment Editor as defined by the RM-ODP. This covers the various data structures that are being used for storage and exchange between the service components defined in the computational viewpoint.

For HALE, the main data structures are those used to express a conceptual schema and the one used to express conceptual schema mappings and transformations.

4.1. Conceptual Schema

For the expression of conceptual schemas, UML and specifically GML Application Schemas built on UML, as described in ISO 19101 and others are directly used. For detailed information, please refer to the specification of the HUMBOLDT Model Editor (as specified in [2.3.1]). The figure below shows the metamodel for this conceptual Schema Language. In addition to this metamodel, the CSL contains a typing system for the precise definition of attributes and especially of geometric attributes.



It is being evaluated whether specific Ontology elements should be added to this model.

4.2. Mappings and Transformations Model

For the Mapping of Conceptual Schemas, the OMWG's Ontology Mapping Language (D7.2) is used. Please refer to the [3.3] specification for details.

4.3. Feature Model

To describe Features and Feature Collections, both of vector geometry and of coverages, GML's (ISO 19136) Feature model is used.

These are the Conformance classes of the Feature Model in the MCM, as described in the GML 3.2.1 specification:

GML application schemas defining features and feature collections *A.1.4*

It should be noted that all `FeatureTypes` used are `FC_FeatureTypes` in the sense of ISO 19110 and as described in section FIXME of the Model Repository Service Component Specification..

4.4. Geometry Model

As the main model for 0D to 3D vector geometry, GML's (ISO 19136) Geometry model is used, which in turn is based on ISO 19107.

These are the Conformance classes of the Geometry Model in the MCM, as described in the GML 3.2.1 specification:

GML application schemas defining spatial geometries *A.1.5*

It should be noted that for V3.0, only those sub-clauses of the abstract test suite dealing with 0D/1D/2D data are to be fulfilled.

Consequently, an open point for later versions is the inclusion of specific 2D/3D Geometry types that are used in the CAD/DCC areas, such as NURBS. A candidate standard for alignment in this area is COLLADA.

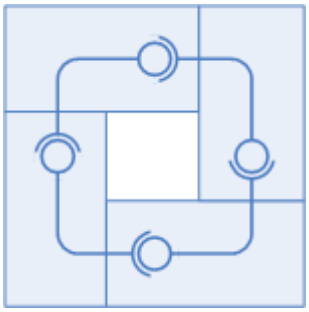
4.5. Coverage Model

These are the Conformance classes of the Coverage Model in the MCM, as described in the GML 3.2.1 specification:

GML application schemas defining coverages *A.1.9*

It should be noted that for V2.0, only Grid coverages (sub-clause of the abstract test suite A.2.15.6) are to be implemented. This is currently sufficient to meet the requirements of the scenarios, specifically the Ocean scenario.

5 Summary & Outlook



The HUMBOLDT Alignment Editor is currently the component receiving the biggest attention and the largest number of change requests both from within the project, but also from a wider community. These change requests (about 40 so far, out of which about 30 have been implemented already) are not part of this specification, but can be seen on the HUMBOLDT community website.

For a next specification to be developed after the HUMBOLDT project, several aspects on the task-based approach, the mapping language used (specifically the introduction of a mismatch description language) and the continuous quality assurance will have to be refined.

Annex A: Schemas

geoOntology Mapping Language Profile

The OML Schema is part of the [3.3] documentation.