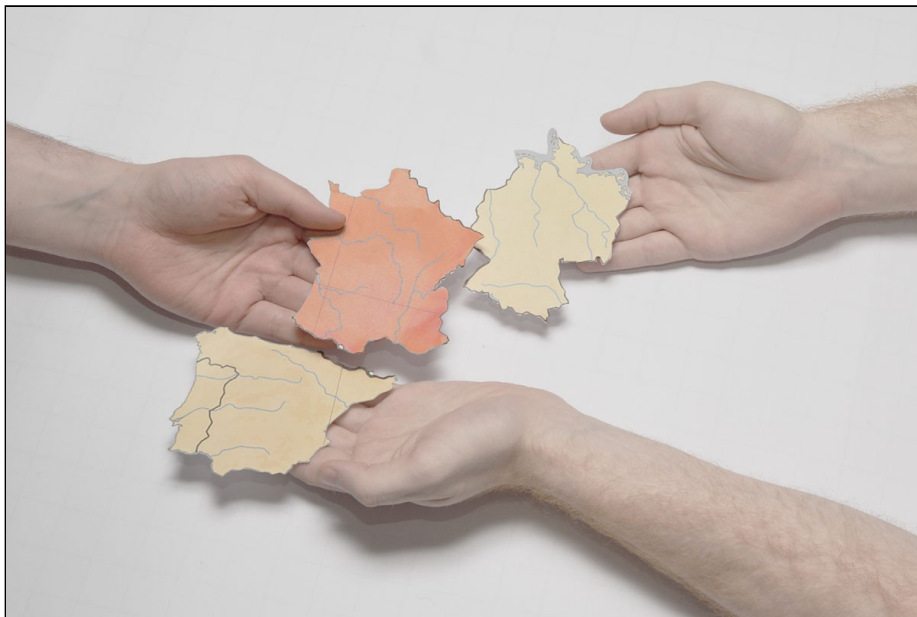

Efficient Conceptual Schema Translation for Geographic Vector Data Sets

Thorsten Reitz, Daniel Fitzner, **Dr. Eva Klien**, Ulrich Schäffler



13th AGILE
International Conference
on Geographic Information Science,
Guimarães, Portugal

13.05.2010

Outline

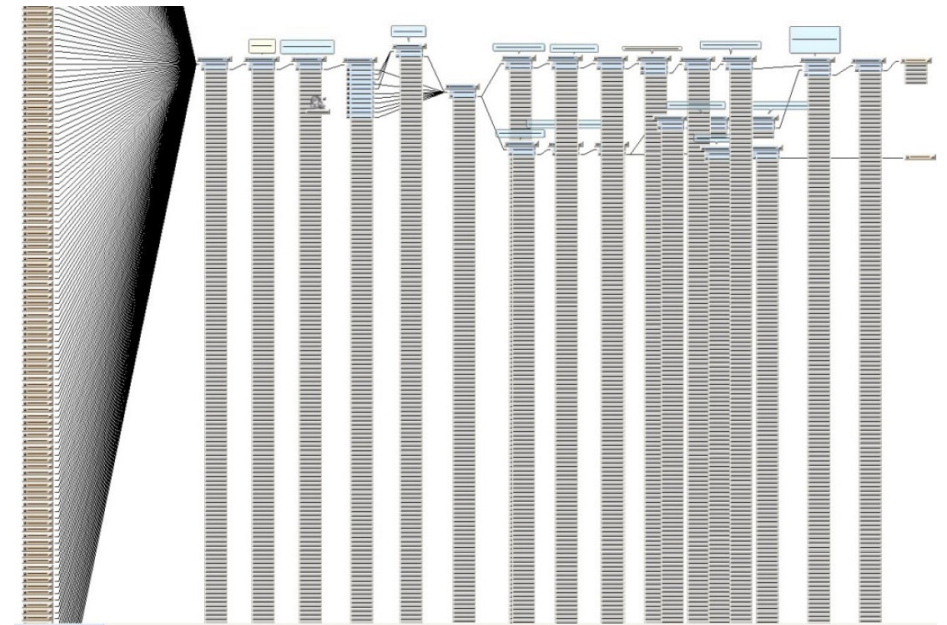
- Motivation & Current Issues in Schema Translation
- Approach
 - The Ontology Mapping Language
 - CST Algorithm
- Implementation
 - HUMBOLDT CST Service
- Evaluation and Performance Analysis
 - Scenario description
 - Performance
- Conclusion and Further Research

Motivation & Current issues

- Conceptual Schema Mapping and the Translation based on the Mappings is an important means for achieving (geospatial) data interoperability;
- Schema Mappings are **hard to create and to maintain**, and it is difficult to document them;
 - Even relatively simple mappings become multi-thousand line XSLT scripts or very complex pipes-and-filters arrangements
- **Re-usability of mappings** and of data transformed on their base is limited;
- It is generally assumed that Schema Translation cannot be done in **real time/on demand**, while at the same time more and more live data needs to be integrated

Goals:

- Adoption of an executable, declarative and compact language for conceptual schema mapping to complex geospatial data integration/harmonisation scenarios
- Development and implementation of an algorithm for the efficient execution of such declarative mappings in the aforementioned geospatial data integration scenarios



Source: Markus Müller, AED SICAD

Approach I: Ontology Mapping Language

- *First goal: find a suitable language for expressing conceptual schema mappings*
- The Ontology Mapping Language was developed in the course of several Semantic Web projects, originally not with geospatial schema translation in its scope.
 - More Information:
<http://www.omwg.org/TR/d7/>
- It provides the following capabilities:
 - Expressiveness: Even complex conceptual mappings and attributive transformations can be expressed, some mismatches are thus avoided
 - Loose coupling: Mappings are not bound to a specific Schema Language (UML, OWL, GML Application Schemas...) and can thus be used with different schema formalisms
 - Usage in semantic web technologies possible to validate semantic correctness of created mapping (and thus of translated data sets)

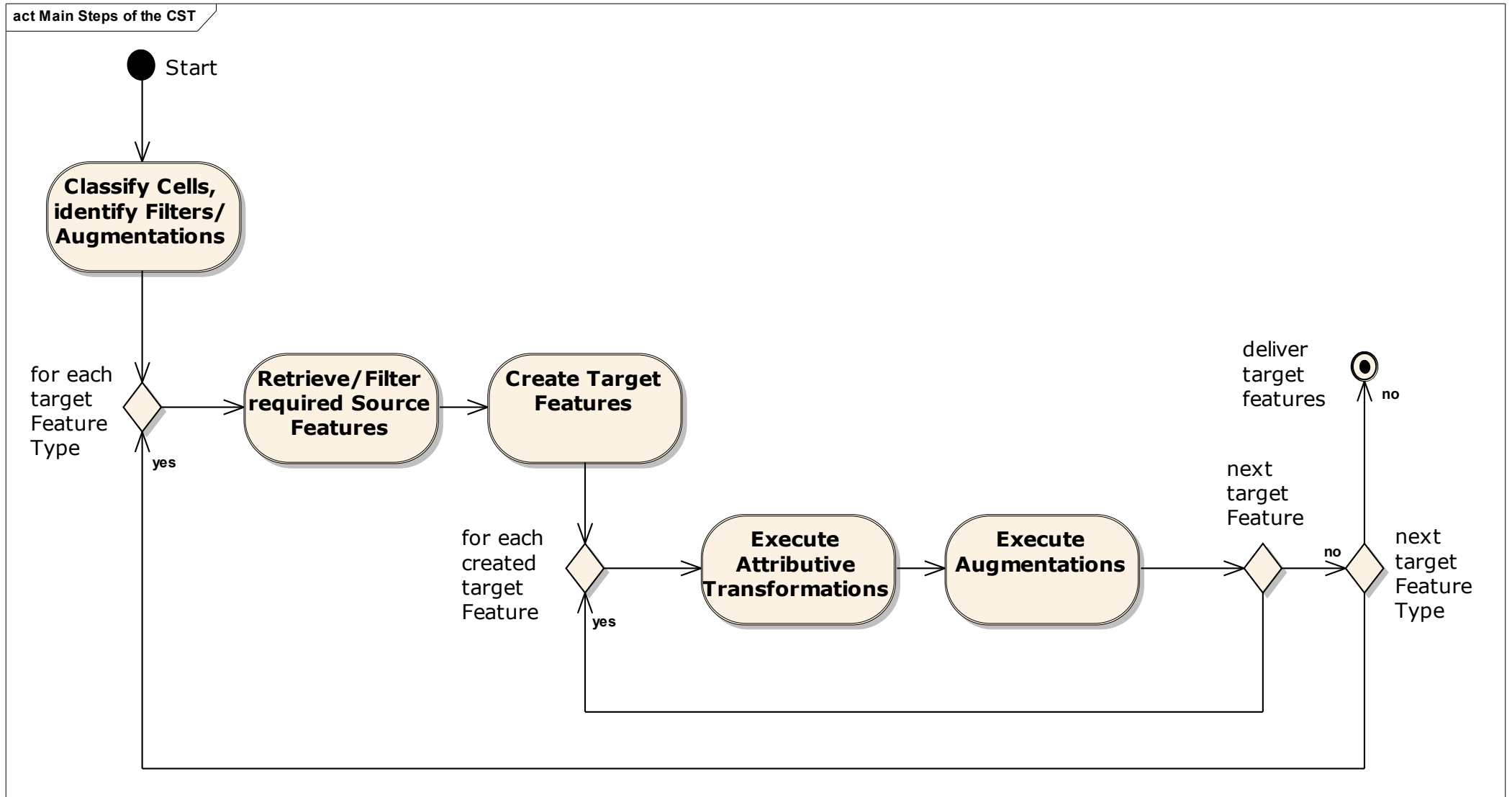
Ontology Mapping Language: Example

```
<align:Alignment ... >
  <align:onto1>
    <align:Ontology><align:location>http://www.esdi-humboldt.org/waterVA</align:location></align:Ontology>
  </align:onto1>
  <align:onto2>
    <align:Ontology>
      <align:location>urn:x-inspire:specification:gmlas:HydroPhysicalWaters:3.0</align:location>
    </align:Ontology>
  </align:onto2>
  <align:map>
    <align:Cell>
      <omwg:entity1>
        <omwg:Class rdf:about="http://www.esdi-humboldt.org/waterVA/Watercourses_VA">
          <omwg:transf
rdf:resource="eu.esdihumboldt.cst.transformer.service.rename.RetypeFeatureFunction"></omwg:transf>
          <omwg:attributeValueCondition>
            <omwg:Restriction>
              <goml:cqlStr>LEVEL == 1 OR LEVEL == 2 OR LEVEL == 3 OR LEVEL == 4 OR LEVEL == 5
</goml:cqlStr>
            </omwg:Restriction>
          </omwg:attributeValueCondition>
        </omwg:Class>
      </omwg:entity1>
      <omwg:entity2>
        <omwg:Class rdf:about="urn:x-inspire:specification:gmlas:HydroPhysicalWaters:3.0/Watercourse"/>
      </omwg:entity2>
      <align:relation>Equivalence</align:relation>
    </align:Cell>
  </align:map>
```

Approach II: CST Algorithm

- *Second Goal: Develop an algorithm for the efficient execution of declarative mappings*
- Use information in the OML mapping to create an optimized “translation plan”
 - OML mapping is „unsorted“; it doesn’t contain any information on the order in which the transformation steps have to be executed.
 - Determine „Cell Cardinality“ with respect to affected Features and Feature Types:
 - Example 1: A Cell that will result in a single source Feature being split into multiple target Features (e.g. going from a MultiPoint to a Point-type object) gets a one - to - many *Instance Cardinality*.
 - Example 2: A Cell that contains declares a join between features of different Feature Types (e.g. taking the geometry from a different Feature Type than the other attributive values) is classified as a many - to - one *FeatureType Cardinality*
 - Determine which Features of which source FeatureType will later be required for which transformation step;
 - Based on the identified cardinalities, ensure that relevant transformations are bundled, so that source features only have to be retrieved when really required

Execution Algorithm Overview



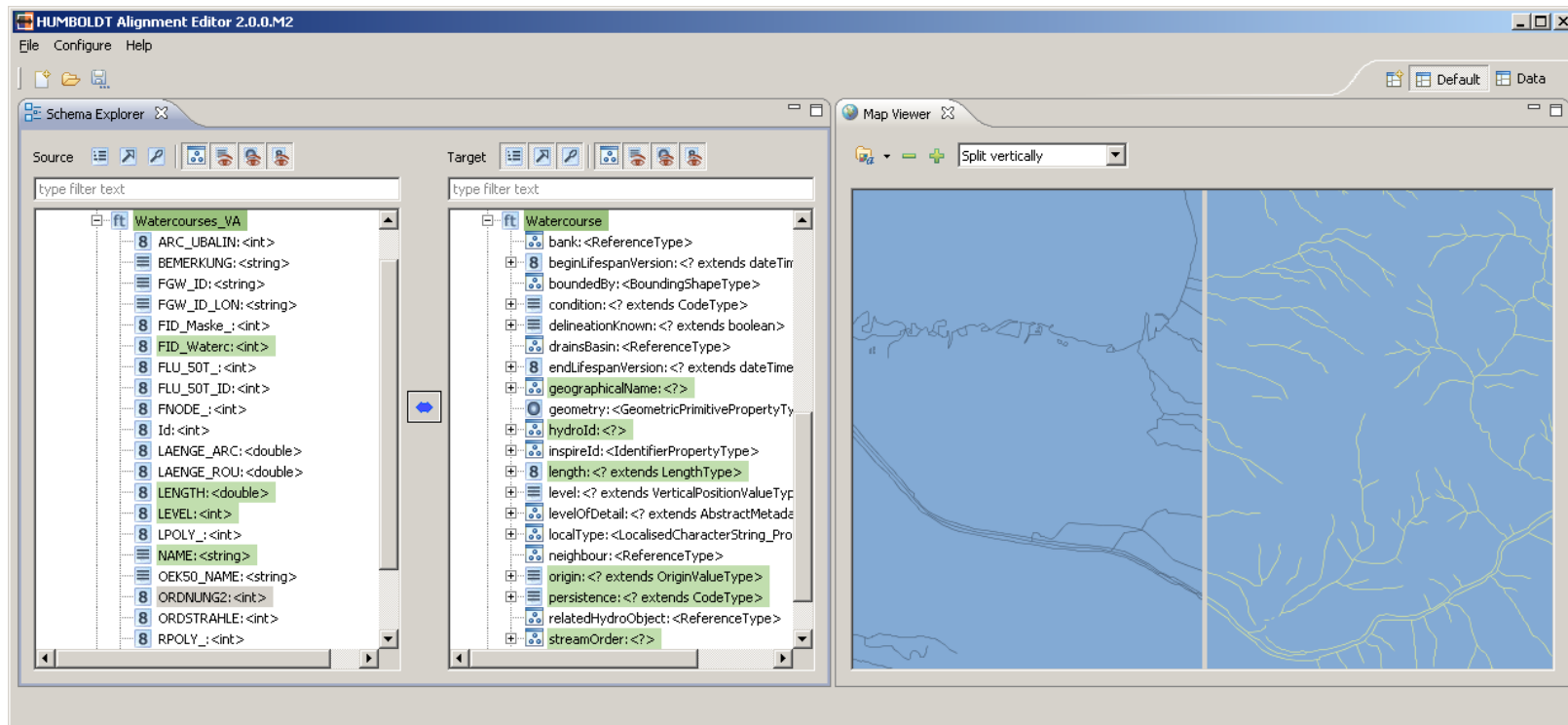
Implementation

- Implementation within the scope of the HUMBOLDT project
- Java 1.6, OSGI, Geotools as basis
- Also available as a Web Processing Service (WPS) 1.0.0 with a very simple web client
- Currently supports about 15 different transformation functions (such as *Attribute Renaming*, *Constant Value Assignment*, *Splitting/Aggregation/Concatenation*, *Mathematical Expressions*, *Buffering*, *Centroid Calculation*, *Classification Mapping*, *Date Extraction*, *Ordinates to Point Geometry...*)
- Is easy to extend with additional functions, especially in an OSGI context
- Expects three inputs:
 - Source data (GML 2.1 or 3.1)
 - Target Schema (GML Application Schema 2.1, 3.1 or 3.2)
 - OML Mapping file

Evaluation I

■ Evaluation Scenario:

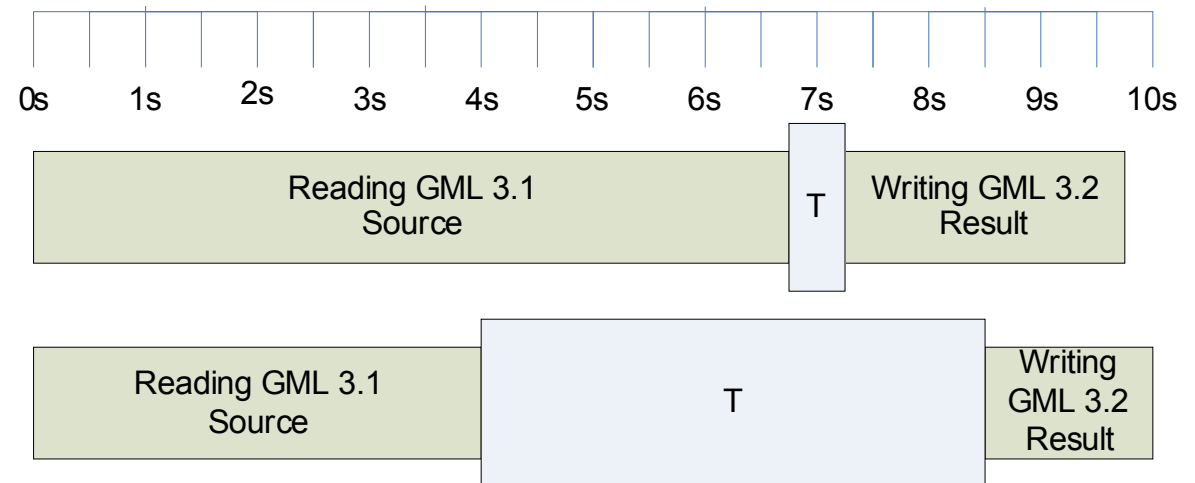
- Encoding of matching tables into OML documents with 9 to 31 Cells
 - Inclusion of different mapping scenarios, such as feature retyping, attribute renaming, reclassification of attribute values, unit and type conversions, geometric operations, feature splits, ...
- Usage of 2MB to 921MB GML files of hydrography data (line and polygon features)



Evaluation II

■ Performance:

- Reading/writing GML takes a huge portion of the overall time (around 95% in case of a simple retyping/renaming/reclassification mapping)
- Depends highly on the complexity of the individual attributive transformations
 - When only retyping, renaming and reclassification functions are used, the transformation service itself can achieve throughputs of more than 10.000 Features/s.
 - Using geometric operations such as buffer or topology reconstruction will make throughput go very low (50 to 500 Feature/s).
- Mapping can be analysed based on these experience to decide about on-the-fly transformation.

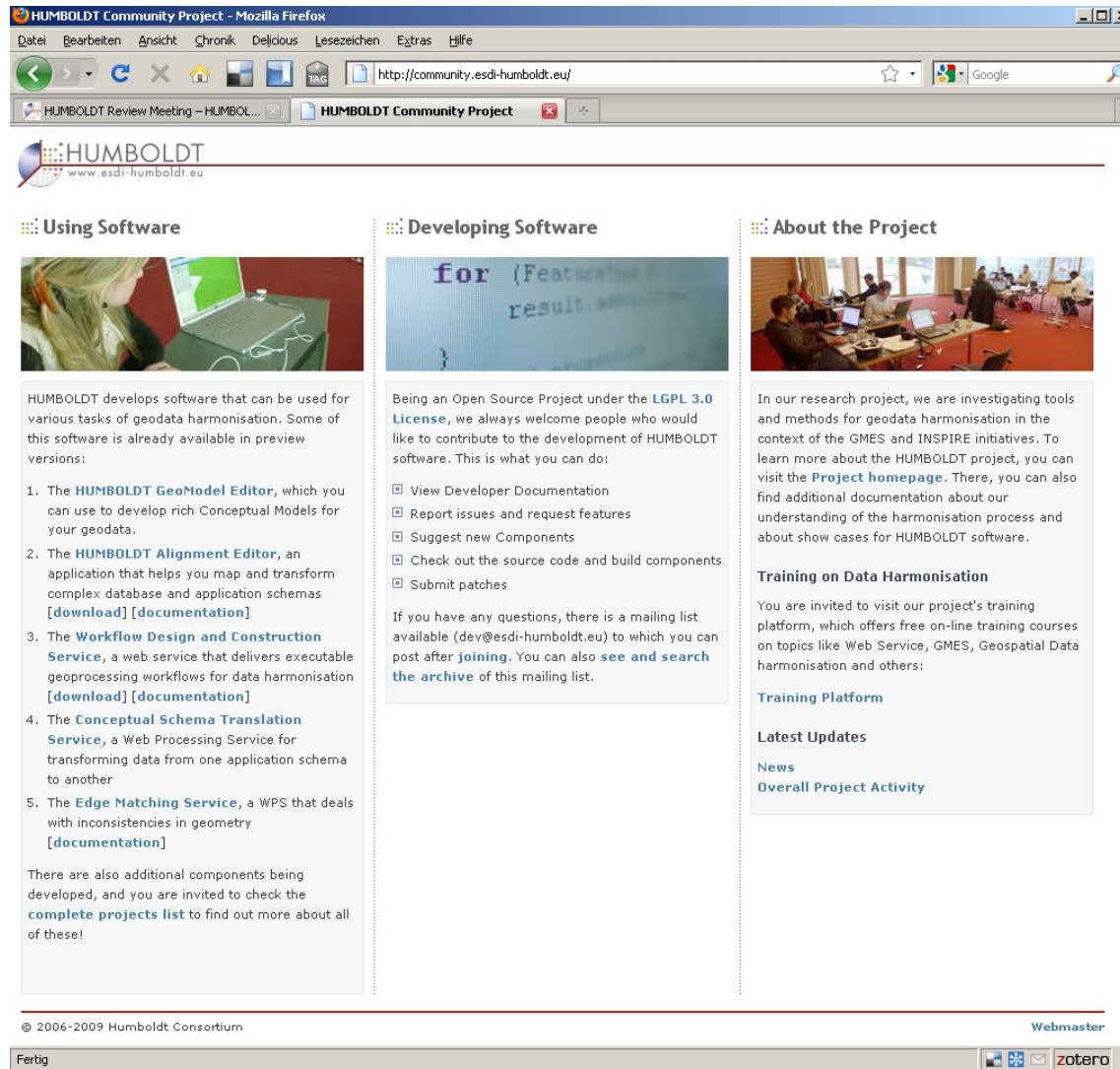


Conclusion and further research

- OML has been evaluated to be of sufficient expressivity and conciseness to be a useful conceptual schema mapping language that can be executed efficiently
- An algorithm was defined and implemented to efficiently perform complex schema translations based on OML documents
- For simple schema translation operations, performance of approaches that directly work on XML inputs is better, since GML I/O is currently taking a lot of time
- Open points:
 - Improve I/O of GML or use other encodings
 - Improve handling of large data sets
 - Improve handling of *merges based on spatial properties* and of *joins between Features of different Feature Types* cases

The HUMBOLDT CST implementation is Open Source

- To get more information on the CST Service implementation, please visit <http://community.esdi-humboldt.eu>
- Your questions, ideas and requirements are welcome!



The screenshot shows the Mozilla Firefox browser displaying the HUMBOLDT Community Project website. The browser's address bar shows the URL <http://community.esdi-humboldt.eu>. The website has a header with the HUMBOLDT logo and the URL www.esdi-humboldt.eu. The main content is organized into three columns:

- Using Software:** This section features a photograph of a woman working on a laptop. Below the image, it states: "HUMBOLDT develops software that can be used for various tasks of geodata harmonisation. Some of this software is already available in preview versions:" followed by a numbered list of five services: 1. **HUMBOLDT GeoModel Editor**, 2. **HUMBOLDT Alignment Editor**, 3. **Workflow Design and Construction Service**, 4. **Conceptual Schema Translation Service**, and 5. **Edge Matching Service**. Each item includes links for [download] and [documentation]. A concluding paragraph mentions additional components and a [complete projects list](#).
- Developing Software:** This section includes a photograph of a computer screen displaying code. The text reads: "Being an Open Source Project under the **LGPL 3.0 License**, we always welcome people who would like to contribute to the development of HUMBOLDT software. This is what you can do:" followed by a list of actions: View Developer Documentation, Report issues and request features, Suggest new Components, Check out the source code and build components, and Submit patches. It also provides a mailing list link (dev@esdi-humboldt.eu) and a link to [the archive](#).
- About the Project:** This section features a photograph of a meeting. The text describes the project's focus on geodata harmonisation in the context of GMES and INSPIRE, and provides a link to the [Project homepage](#). It also mentions training on data harmonisation and a [Training Platform](#). A "Latest Updates" section includes links for [News](#) and [Overall Project Activity](#).

At the bottom of the page, there is a copyright notice: "© 2006-2009 Humboldt Consortium" and a "Webmaster" link. The browser's status bar at the very bottom shows "Fertig" and the Zotero logo.

Thanks a lot for your attention!



Contact

Thorsten Reitz

Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt

Email thorsten.reitz@igd.fraunhofer.de

Telefon + 49 (0)6151 155 416

www.igd.fhg.de/igd-a5