RSW GEOMATICS LTD

1Spatial

Rob Walker Consultancy

Prototype Report
for the
INSPIRE Schema Transformation Network Service

EC JRC Contract
Notice 2009/S 107-153973

Authors: Matt Beare, Simon Payne, Richard Sunderland

Date: 1 September 2010
Version: 3.0
Status: Final

Document Information

This is the Prototype Report for the INSPIRE Schema Transformation Network Service.

Purpose

This report provides a description and evaluation of the Prototype Demonstrator, as produced to verify the feasibility of the Technical Guidance for INSPIRE Transformation Network Services (TNS). A Conformity Report, which provides an assessment on whether or not the prototype has met key non-functional requirements, is included.

It forms part of the final deliverable within the scope of work for the EC JRC Contract Notice 2009/S 107-153973, as awarded to RSW Geomatics, 1Spatial and Rob Walker Consultancy.

Legal Notice

Executive Summary

The Technical Guidance (TG) for INSPIRE Schema Transformation Network Services (TNS) [1] explains how the functional requirements of the INSPIRE Regulation [3] (as amended by [4]) and Implementing Rules for Transformation Network Services can be realised. It aims to provide pragmatic and unambiguous advice to Schema Transformation Network Service implementers and service integrators working on behalf of INSPIRE Legally Mandated Organisations (LMOs).

To demonstrate the practical feasibility of the TG, a prototype demonstrator has been developed. The prototype is based exclusively on the TG, i.e. no assumptions or service parameters have been added. The TG is expected to provide the authoritative source for the operation signatures, parameter data types and modes of transport of parameter data, without requiring reference to any other document. The objective of the prototyping phase has been to prove the principles of the various aspects of the TG deliverable and refine the guidance if required.

This report is intended to document the approach taken to implement the prototype, the results attained and lessons learned.

The prototype has been developed to include pre-existing software tools from four separate organisations and open source communities, in order to deploy the components needed for the example Transformation Network Service. These include:

- GeoServer [7], an open source Web Feature Service (WFS), for publishing source data/schema;
- Humboldt Alignment Editor (HALE) [8], an open source tool for defining schema mappings;
- Radius Studio™[9], a commercial geospatial rules engine for performing the transformations;
- TatukGIS Viewer [10], a free to use application for visualising transformed data.

It is possible, by exercising the primary recommendations made by the Technical Guidance, to bring these independent and disparate tools together to provide a flexible and effective transformation service. This involves use of international standards for the exchange of data, logical schema descriptions and schema mappings. Of particular note are the recommendations to use:

- Geography Markup Language (GML) standard [5] , from Open Geospatial Consortium (and already endorsed by INSPIRE), for data and schema descriptions;
- Rules Interchange Format (RIF) [6], a recently adopted standard from World Wide Web Consortium, for schema mapping definitions.

To test and demonstrate the prototype service, six test datasets were kindly provided by six thematic data partners, presenting a broad coverage of transformation scenarios in respect of three INSPIRE data themes (Cadastral Parcels, Hydrography and Transport Networks)

This report concludes that the Technical Guidance shows that transformation is possible using many different tools if industry standards are followed throughout the process. Although data providers or publishers can carry out transformations themselves, they can also be offered as Network Services available to a broader community. The recommendations of the Technical Guidance are considered applicable to a variety of deployment scenarios, not just network service environments. Member States and their data providers will, of course, decide for themselves exactly where and how these transformations are delivered.

The structure of the report is as follows:

- Section 1 - Contains general document information.
- Section 2 - Provides an overview of the prototype, in non-technical terms, identifying the tools used and purpose they serve in the context of an end-to-end transformation process.

- Section 3 - Presents a more technical description of the prototype design and implementation, using formal methods to describe the key architectural aspects. It also provides a critique on the performance characteristics of the prototype.
- Section 4 - Reports on the conformity of the transformed data, having passed through the prototype INSPIRE Schema Transformation Service (TNS).
- Section 5 - Concludes the report, summarising key findings and lessons learned.

Table of Contents

Prototype Report

1. Introduction

1.1 Purpose

The Prototype Report presents the results of the prototype demonstrator, as developed to verify the feasibility of the Technical Guidance (TG) for INSPIRE Schema Transformation Network Services [1].

The TG is expected to provide the authoritative source for the operation signatures, parameter data types and modes of transport of parameter data, without requiring reference to any other document. The objective of the prototyping phase has been to prove the principles of the various aspects of the TG deliverable and refine the guidance if required.

The report includes an overview of the prototype, with analysis of performance, input/output data models and mappings, constraints and pre-requisites on the models and mappings and lessons learnt during implementation.

The report is supplemented with a Conformance Report (Section 4), which provides an assessment of whether or not the prototype has met key non-functional requirements; with advice on corrective actions as required.

1.2 Scope

This report forms part of the final deliverable within the scope of work for the EC JRC Contract Notice 2009/S 107-153973, as awarded to RSW Geomatics, 1Spatial and Rob Walker Consultancy.

The report makes specific reference to the Technical Guidance document, as developed for this contract.

Disclaimer: It should be noted that this Prototype Report does not seek to endorse any of the technologies mentioned. The final choice of which technologies to use in specific solutions will be for system developers to establish, based on the operational requirements within which a TNS is expected to be deployed.

1.3 Intended Audience

This document is intended for Schema Transformation Network Service implementers and service integrators working on behalf of INSPIRE LMOs.

Readers are assumed to have a general understanding of the INSPIRE Directive [2] and, for some sections, an understanding of web services and related technology.

Furthermore, readers are expected to have read the final version of the Technical Guidance [1].

1.4 Terms, Definitions and References

The definitions of any terms, abbreviations and acronyms used throughout this document can be found in Appendix A.

References are provided in Appendix B.

2. Prototype Overview

In order to verify the practicality of the TG, the authors have applied the directions provided in it to develop a prototype implementation of an INSPIRE TNS. A key characteristic of the TG is the advocacy of a standards-based approach to developing the TNS. This characteristic is described first in this overview section.

2.1 Standards-Based Approach

The TG advocates the use of standards to support the implementation of a TNS. In particular, it recommends the use of two standards, as illustrated in Figure 1:

- The Geography Markup Language (GML) [5] from the Open Geospatial Consortium (OGC) for describing encoding documents and logical schemas;
- Rule Interchange Format (RIF) [6] from the World Wide Web Consortium (W3C) for the definition of schema mappings.

Both these standards have been defined rigorously and are considered very capable of managing the problem domain of schema transformation.

The use of standards in this domain, where currently this is not the case, will help foster vendor neutrality, affording system developers greater flexibility to deploy solutions that meet with customer requirements, avoiding issues of non-interoperability and vendor lock-in.

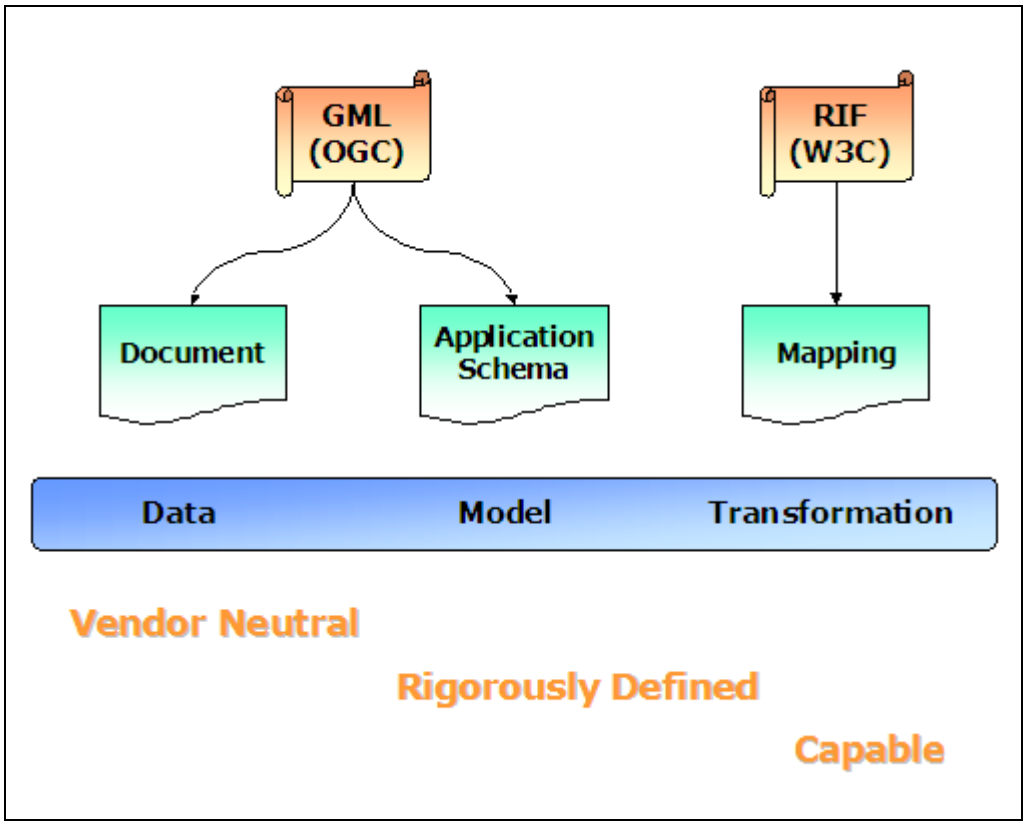

Figure 1 TG recommended standards

2.2 Prototype Components

To demonstrate the TNS, a number of other components are required to illustrate the overall process. These components are described as follows.

Based on the components suggested by the TG [1], it is possible to select alternative technologies to implement the required capabilities.

In the prototype demonstrator, the project team elected to use technologies from a cross-section of tool suppliers, ranging from proprietary to open source. This is intended to highlight the vendor neutrality aspect of the TG, with the recommended standards providing the interoperable "glue".

The suggested components and the selected technologies used in the prototype demonstrator are illustrated in Figure 2.

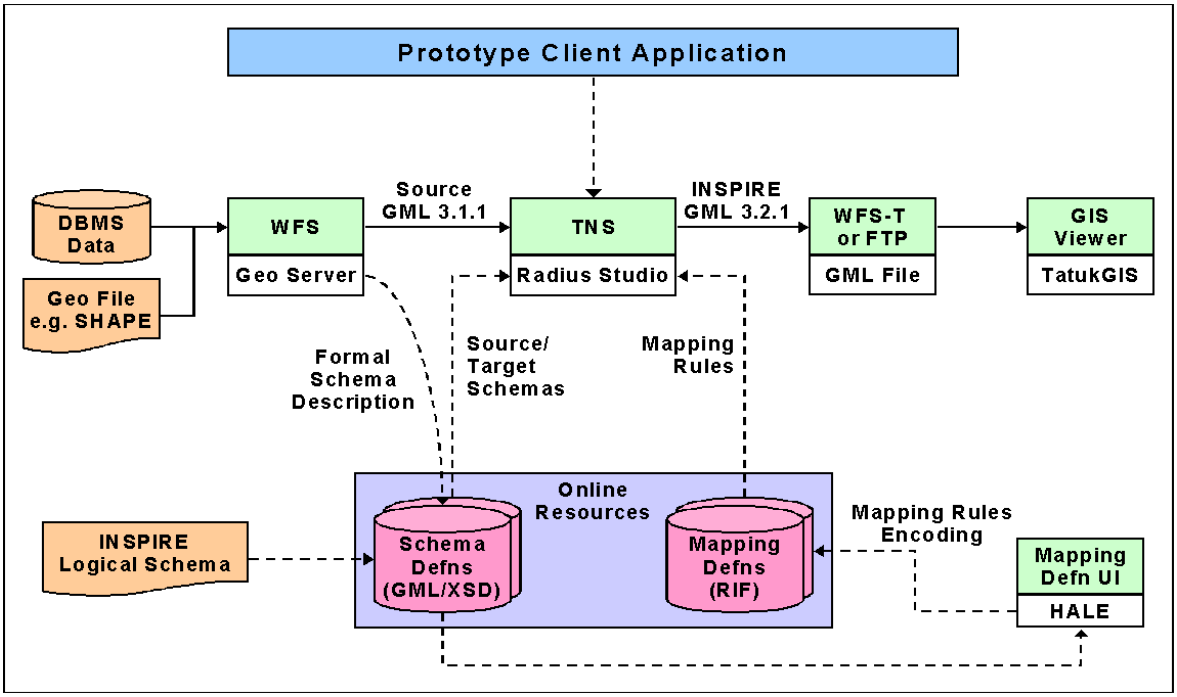

Figure 2 Prototype demonstrator components

The rest of this section introduces each of the prototype components (and the technology used to implement them), providing a brief description of the role they serve in the context of an end-to-end transformation process scenario. This is based on the suggested architecture of the TNS TG. It should be noted that the TNS TG does not recommend the need for all of these components, such as GIS Viewer, but they have been used as part of this prototype to help demonstrate the TNS TG in a practical scenario. Where the TNS is concerned, the TG has been used exclusively as the source of requirements on what constitutes a valid INSPIRE TNS. However, this does not mean that other sources of information have not been consulted (for example, the RIF specification [6] provides essential information to aid in developing a proper understanding of the RIF-PRD format).

2.2.1 Provide Online Resources

For the prototype, basic upload and download functionality was provided using the Apache web server and an FTP service. Apache served the INSPIRE XML Schema documents to the TNS, and the FTP service supplied the location for the RIF-PRD schema mapping definition documents and the target location for output of the transformed GML (see Figure 2). It had been the authors' intention to deploy the open source freebXML repository ([21]). However, this was not feasible within time constraints. See Section 5 Conclusions for a discussion of the issues involved here.

2.2.2 Prototype Client Application

The client application was a lightweight web service client written using Google Web Toolkit [22] and accessible from a web browser. It provided fields for the user to specify the URL of the Web Feature Service to be used to publish the source data, the locations of the INSPIRE logical schemas and schema mapping definition, the required source feature types to be transformed, and the target location to which to publish the transformed data. It also included a button to invoke the TNS web service with these parameters. The client reported on the initiation of the web service transformRequest() operation and its eventual completion or any errors that were thrown by the service.

2.2.3 Publish Source Data

To avoid additional work on the part of our data partners, they provided us with geospatial data sets in the encoding that they would typically use. The prototype demonstrator therefore needed to manage the encoding translation, to publish source data in the form required by INSPIRE, i.e. GML 3.1.1 or 3.2.1 (see TG [1] Section 4.5 Use Case Transform Data, precondition 1).

The data specifications provided by the data partners came in a range of forms and levels of completeness and usefulness (see Section 3.1 for more details). For many data providers wishing to implement INSPIRE transformation services, there may be an issue of how to document logical schema descriptions in the recommended standards-based way using GML XSD, without incurring excessive costs.

The technology selected to support the prototype demonstrator was GeoServer [7], which mitigates this potential cost.

GeoServer is an open source server-side application that allows users to share and edit geospatial data. Designed for interoperability, it publishes data from any major spatial data source using open standards. It is a good reference implementation of the OGC Web Feature Service (WFS).

GeoServer offered two advantages to the prototype:

1. It can publish data according to the GML standard. However, it is noted that a current limitation of this software (and many others available at this time) is that it does not support the latest GML version 3.2.1 standard, as required by INSPIRE. Despite this limitation, it was felt that GML version 3.1.1 support was sufficient to meet the needs of the prototype. This was a prototype-specific decision, and does not restrict the source data format that a TNS implementation can accept to GML version 3.1.1 alone.
2. It can auto-document the source logical schema, as derived from the source data file, and present this logical schema description in GML, complementing the TG recommendation, and avoiding excessive costs on the part of the data provider.

Generation of these two outcomes (source logical schema in GML and source data in GML) from the source input is illustrated in Figure 3. Note that, in Figure 3, the Source Logical Schema and Source Data Set are in GML 3.1.1 format.

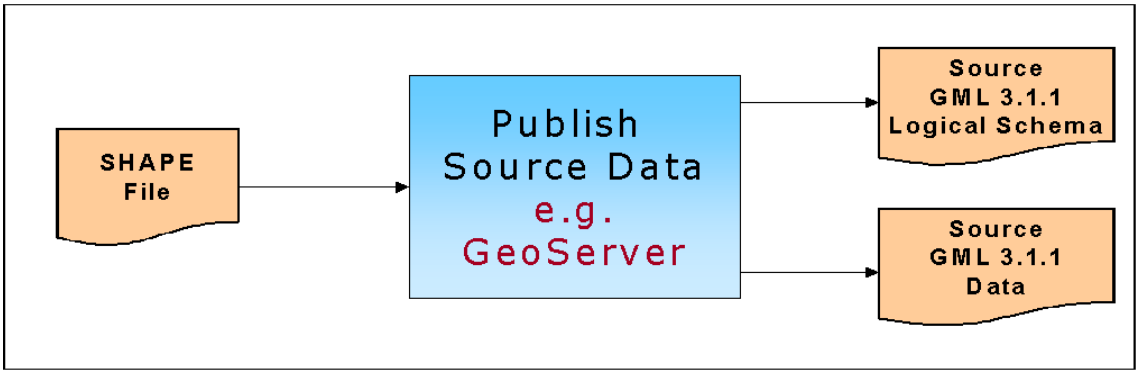

Figure 3 Prototype component for publishing source data

2.2.4 Define Mapping

The source logical schemas had been published as described above, and the target INSPIRE logical schemas had already been published using, for the purposes of the prototype, an Apache web server, with the INSPIRE XSDs made available as static content. The next activity was to use these to define the schema mappings to transform data from the source logical schema to the target logical schema.

For the prototype demonstrator the Humboldt Alignment Editor (HALE) was selected as an emerging tool available to the open source community [8]. This is an outcome of a separately funded EC project, and the TNS project team were keen to demonstrate the re-use opportunity of the outcomes of other such projects.

As an emerging tool, there are limitations in the richness and complexity of transformation mappings that can be defined in HALE, with respect to the INSPIRE logical schemas, but it was felt to be sufficiently capable to demonstrate the recommendations of the TG with respect to one of the data themes (cadastral parcels). The mappings for the two other themes (hydrography and transportation) were prepared manually using a text editor.

The inputs to the mapping definition process were the source and target logical schemas. The source logical schemas were obtained from the WFS using the DescribeFeatureType operation. The target logical schemas were served by an Apache web server version 2.2, hosted on another machine on the same network as the host on which the TNS was deployed. They were served as static content in the form of XSD files.

HALE offered two advantages to the prototype:

1. The mapping definition functionality is de-coupled from the Humboldt transformation engine, the Conceptual Schema Transformer (CST).
2. The tool was shown to be easily extensible, allowing a RIF exporter to be developed as a plug-in to HALE. This plug-in was developed by 1Spatial as an in-house exercise. We expect to make it available in future for download at an appropriate location on the internet.

Generation of a RIF mapping document, using the standards-based descriptions of source and target logical schemas, is illustrated in Figure 4. HALE stores the result as an XML document on the file system, from whence it is copied manually to an FTP location from which it is able to be consumed as an input to the TNS service. It is also possible to consider modification to HALE to permit the result to be persisted to an XML Repository although this was not prototyped. Note that, in Figure 4, the Source Logical Schema is in GML 3.1.1 format and the Target INSPIRE Logical Schema is in GML 3.2.1 format.

1Spatial possesses significant expertise on the input spatial datasets that were supplied by the data providers who contributed to this prototyping, and have gained familiarity with the data models over many years. Thus it was not necessary to seek assistance from data providers in the preparation of the mappings used to demonstrate the schema TNS. The issues discovered by the data experts during the mapping exercise have therefore been incorporated into this report (see in particular Section 4 Report of Output Data Conformance to Target Schema) rather than being treated separately.

The process used to define the mappings is described in detail in Section 3.1.3.

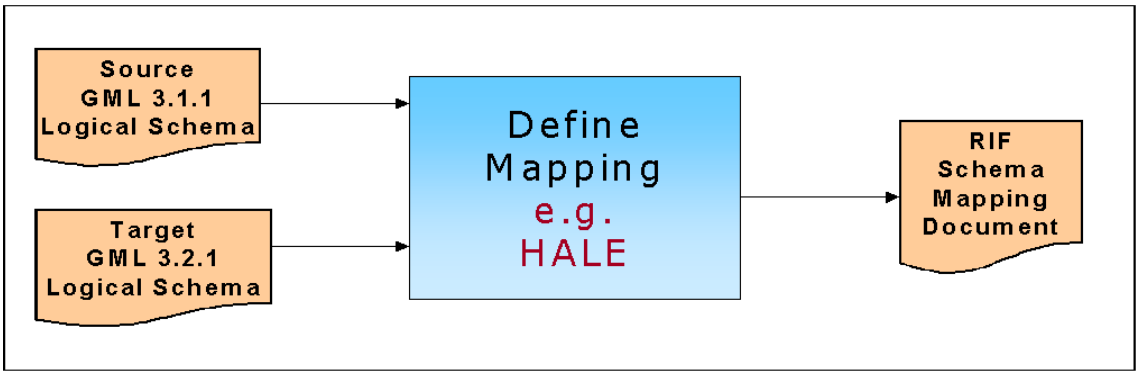

Figure 4 Prototype component for defining schema mappings

2.2.5 Perform Transformation

The TNS interface defines the format required for the messages that are sent as input to the respective operations, and for pre-configured errors that may be thrown by them on encountering exceptional conditions. All the inputs (source data, source and target logical schemas and schema mapping definition document) are required for compliance with the INSPIRE Directive [2].

At this stage, the process has available to it the source logical schema, target logical schema, schema mapping definitions and the source data, all in standards-based form. These are the inputs to the TNS, to perform the main transformation activity of data from the source logical schema to the target logical schema.

For the prototype demonstrator, we used the Radius Studio™ geospatial rules engine [9] to perform the transformations, sitting behind the generic INSPIRE TNS interface. Radius Studio is a proprietary product from 1Spatial, selected to emphasise the vendor neutrality aspect of the TG. It demonstrates that the introduction of standards can allow two pre-existing and previously un-related tools to be brought together to provide an overall operational capability.

Radius Studio offered four advantages to the prototype:

1. Capable of performing the transformation mappings appropriate to the problem domain.
2. The transformation engine is suitably de-coupled from the mapping definition functionality.
3. Proven to be scalable in large spatial data management operational scenarios.
4. Able to access and conflate multiple datasets/sources if required.

Furthermore, with access to the source code, the project team were able to develop an import plug-in, to re-cast the RIF mapping definitions into the Radius Studio rules language. It is anticipated that developers of other tools, open source and proprietary, will be able to create similar adapters for their respective transformation engines. However, the project team cannot estimate the relative ease of implementing such extensions.

As Figure 2 shows, the source data for the prototype is provided by a WFS and the source/target logical schemas and schema mapping definition document for the prototype are provided by an FTP service and web server, respectively (it is recommended in the TG [1] Section 5.5 that an XML Repository be used for this purpose).

The Transform Data function returns data that has been transformed into the target INSPIRE logical schema. This is illustrated in Figure 5. Note that, the Source Logical Schema and the Source Data Set are in GML 3.1.1 format, and the Target INSPIRE Logical Schema is in GML 3.2.1 format, which is also the format of the returned data.

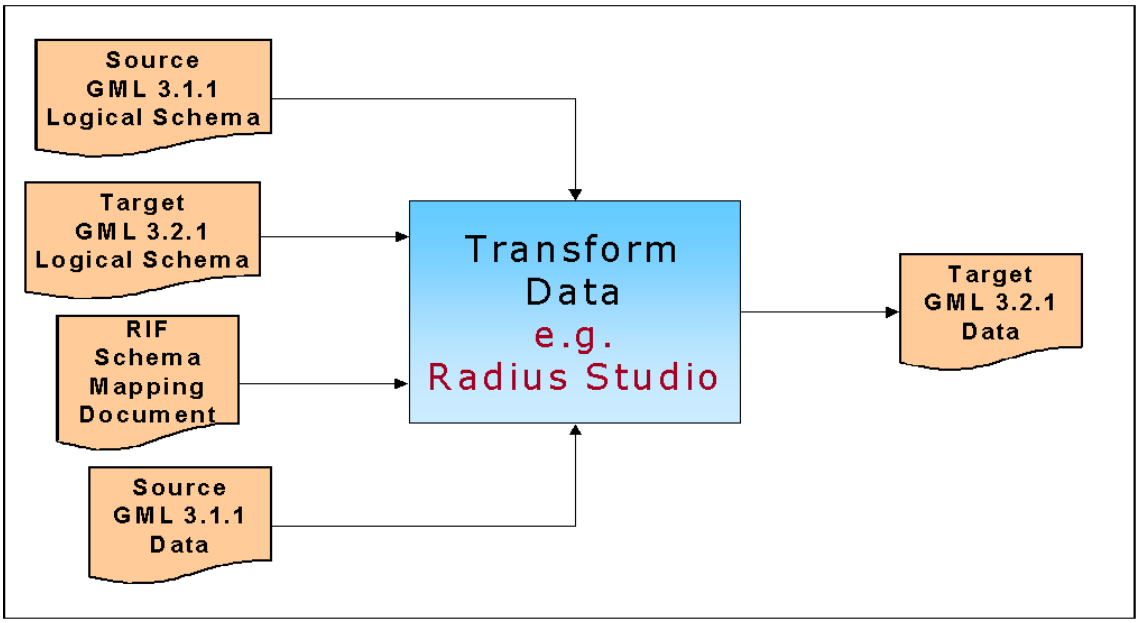

Figure 5 Prototype component for transforming data

In the prototype, the INSPIRE data is stored as GML files on an FTP site. It is recommended that an XML Repository be used for this purpose (see TG [1] Section 5.5).

2.2.6 View and Use Data

To complete the prototype demonstrator, a simple viewing capability is introduced to visualise the transformed and GML encoded data. This is illustrated in Figure 6.

For the prototype, the TatukGIS Viewer [10], a free-to-use application for viewing and querying GIS data, was selected. TatukGIS is capable of rendering most of the GML version 3.2.1 geometries and displays the generated INSPIRE attributes. (It is also of Polish origin, which was nice for demonstration at the INSPIRE Conference 2010 in Kraków.) The TG [1] Section 5.13.2 describes how the transformation service writes the output GML data to the Target Datastore. As Figure 2 shows, this can be a WFS-T or an FTP site. In the prototype, TatukGIS reads the INSPIRE data from a file system location, to which the transformation service has output the data. This is illustrated by Figure 6.

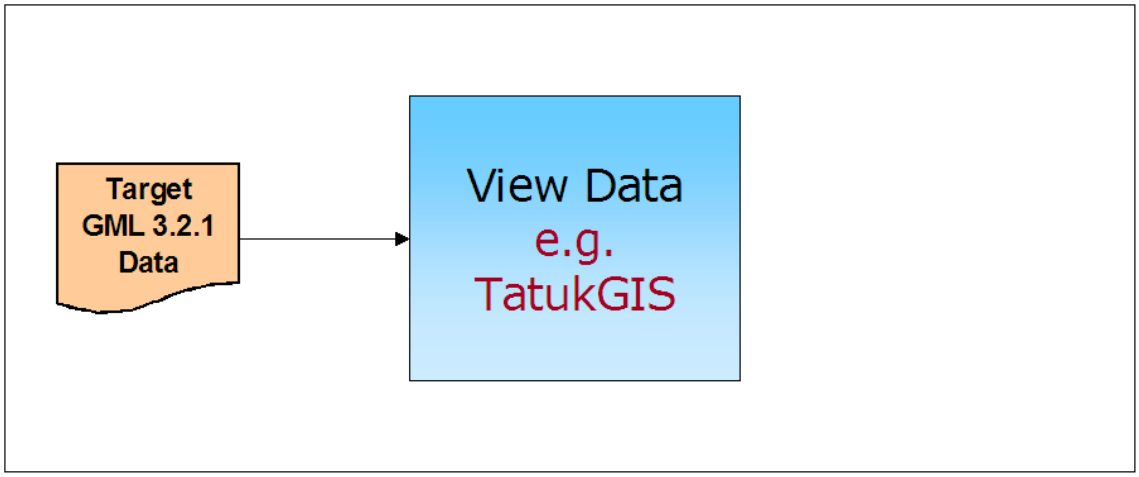

Figure 6 Prototype component for viewing transformed data

3. Prototype Evaluation

The TG is expected to provide the authoritative source for the operation signatures, parameter data types and modes of transport of parameter data, without requiring reference to any other document. The objective of the prototyping phase has been to prove the principles of the various aspects of the TG deliverable and refine the guidance if required.

The outcome of the prototyping exercise is that:

- Prototype development gives engineers the opportunity to overcome the learning curve in the more specialised aspects of the selected technology.
- Individual high-risk elements in the design, such as use of the newly-adopted W3C recommendation RIF, and the use of an XML Repository for storage of mapping definitions and application schemas, have been checked for viability and either confirmed or rejected from the implementation (RIF was confirmed as part of the prototype, whereas the XML Repository was not prototyped).
- Test data made available to the project team provided the means to conduct rough assessment of system performance characteristics.
- Decision-makers have available a visual tool to aid assessment of the potential usefulness of the final system.

Caveats to the prototyping process are:

- A prototype is constructed rapidly, without the diligence applied to a full-scale product.
- Performance statistics generated by a prototype are unlikely to match up to a production system just by applying simple linear scaling.

3.1 Evaluation Test Data

Our thematic data provider partners, listed in Table 1, were selected on the basis that they are LMOs for the nominated INSPIRE data themes.

The test data was supplied from existing source datasets in ‘as-is’ format, with the view that these would be a typical representation of the data they will be required to publish to INSPIRE.

Acknowledgement: Due to project time constraints we have not been able to demonstrate all of the data made available to us, and we would like to apologise to partners where we have not been able to make use of their data. We thank all of our partners for the provision of data and their support of this project.

Table 1 - Thematic data partners.

INSPIRE Theme	Data Provider	Data Used In Prototype?
Cadastral Parcels	Belgian Cadastre	Yes
	Dutch Kadaster	Yes
	France Cadastre	No
	National Land Survey Sweden	No
	National Land Survey Finland	No
Hydrography	National Land Survey Sweden	Yes
	Statens Kartverk Norway	Yes
	NVE Norway	No
Transport Networks	OS Ireland	Yes
	LPS Northern Ireland	Yes

3.1.1 Test Data Characteristics

Table 2 provides a brief description of the characteristics of the test data used in the prototype demonstrator.

Table 2 - Test data characteristics.

Dataset	Provided Format	Pre-GeoServer Translation *	Number of Features used by Prototype **
Belgian Cadastre	SHAPE	None	136903
Dutch Kadaster	SHAPE, GML	None	34034
National Land Survey Sweden	SHAPE	None	4327
Statens Kartverk Norway	SOSI	SOSI files were transformed to SHAPE by 1Spatial Norway using a SOSI2SHAPE transformer.	3230
OS Ireland	AutoCAD DWG and NTF	NTF files were merged and transformed to SHAPE using FME	34236
LPS Northern Ireland	SHAPE	None	25923

For each of the test datasets, the source logical schema descriptions have been packaged and provided in addition to this report, as GML logical schema definition files.

* For the Norwegian hydrography dataset, it was necessary to transform the data from SOSI files into SHAPE files. This was done as a manual step by 1Spatial Norway, using a SOSI2SHAPE transformer.

** Where multiple SHAPE files were provided, only one was actually used by the prototype. In a production environment the whole dataset could have been accessed by either merging the SHAPE files, installing them into a temporary database, or by-passing the SHAPE files and providing access to the dataset in its native format.

3.1.1.1 Representative Character of the Test Datasets

The following table provides background information to the types of challenge encountered when transforming the source datasets into the INSPIRE logical schema. This gives an indication of their representativeness within the overall INSPIRE schema transformation context. These datasets appear to offer a good representativeness and to exercise the TNS interface in a suitably diverse range of ways. Table 3 gives an overview of the test datasets.

Table 3 - Representativeness of Test Datasets

Dataset	Comments
Belgian Cadastre	Data was supplied for the Essen, Kalmthout, Wuustwezel and Hoogstraten municipalities on the Belgian-Dutch border. Mapping involved straightforward class and attribute renaming plus a spatial join (a “contains” function) between source points and polygons, so that cadastral reference points could be assigned correctly to the target features.

Dutch Kadaster	Data was supplied for parts of Zuid-Limburg (South Limburg) bordering Belgium. Mapping involved straightforward renaming of classes and attributes. In addition, the target data was enriched during transformation by calculation of the area of the source polygons and the addition of this information to the target features.
National Land Survey Sweden	Data was supplied for Norrbotten County (one of 21 administrative subdivisions of Sweden, which borders Norway) in the standard Swedish 1:250,000 scale General Map series. From this, two hydrographic feature layers were selected for transformation testing. Mappings used mainly straightforward copying of classes and attributes, with some merging of source classes to produce output classes.
Statens Kartverk Norway	Data was supplied for Østfold and Akershus counties (two of Norway's 19 administrative regions, bordering Sweden) in the Norwegian 1:250,000 scale series. Mappings used mainly straightforward copying of classes and attributes, with some merging of source classes to produce output classes.
OS Ireland	Data was supplied for County Louth (one of 26 Southern Irish counties, which borders Northern Ireland). Mappings demonstrated the use of filtering of single source class features to produce target features in multiple classes (i.e. splitting out) and also instantiation of INSPIRE network properties and network references to associate them with the owning features.
LPS Northern Ireland	Data was supplied for the Newry district of Armagh, bordering the Irish Republic, as an extract from the 1:50,000 scale General Map series. Mappings demonstrated the use of filtering of single source class features to produce target features in multiple classes (i.e. splitting out) and also instantiation of INSPIRE network properties and network references to associate them with the owning features.

3.1.2 Adjacent Cross-Border Datasets

This section shows the two Transportation theme cross-border datasets from Northern Ireland and the Irish Republic laid adjacent to one another. This demonstrates the benefits of schema transformation. Please note that CRS transformation or generalisation is outside the scope of the schema TNS. However, the source dataset CRS's are similar and the data is to the same 1:50,000 scale. Hence, it was possible to prepare this illustration without any additional manual processing.

Figure 7 shows the Transportation theme datasets adjacent to one another. The LPS Northern Ireland dataset is in CRS EPSG:29902 and the OS Ireland dataset is in CRS EPSG:29900. Both source datasets were in scale 1:50,000. The LPS Northern Ireland features are shown in blue and the OS Ireland features are shown in brown.

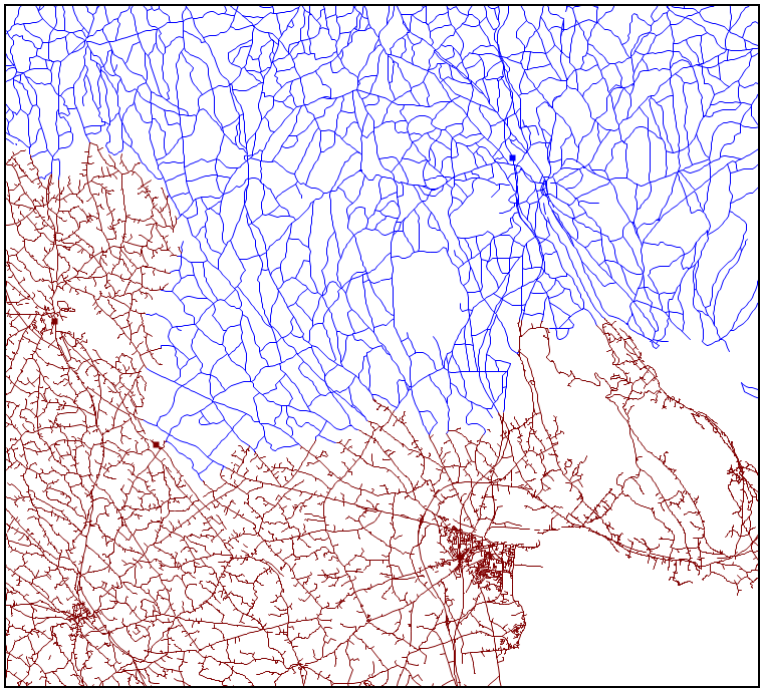

Figure 7 LPS Northern Ireland and OS Ireland transportation data

A close inspection of this data at scale 1:5,000 reveals discrepancies between the two datasets, such as in this example in Figure 8 of an undershoot between road links to the north and south of the border.

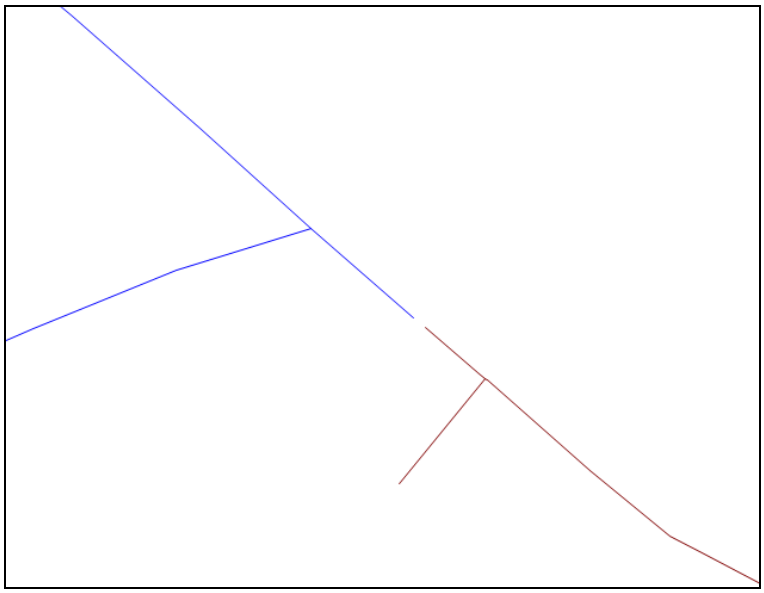

Figure 8 An undershoot between road lines on different sides of the border

3.1.3 Model Mapping Definitions

For each source test dataset, the schema mappings defined to transform the data to the target INSPIRE logical schemas, have been provided separately as a packaged set of RIF documents.

A three-stage process was used to define the schema mappings:

1. Template spreadsheets describing the INSPIRE classes and attributes were generated based on the INSPIRE data specifications. These spreadsheets were defined manually and contained an itemisation of INSPIRE feature classes and attributes taken from the INSPIRE Consolidated UML Model [12]. These included cardinality constraints and whether an attribute was voidable. The spreadsheets were completed by 1Spatial in-house data experts with detailed knowledge of the source datasets. A textual analysis of the source and target logical schemas identified which source attributes could be used to populate the INSPIRE attributes. Where an INSPIRE attribute could be derived from a source attribute indirectly (e.g., by concatenating existing variables or applying a function) this was described using natural language within the spreadsheet. For example, for the feature class CadastralParcel in the INSPIRE schema, the inspireId property can be derived from the concatenation of an attribute named PCVL_PRCL from the Dutch Kadaster dataset's Perceelvlak class with a constant defined for the data provider's selected namespace, NL.KAD.CP.
2. The spreadsheets were used to guide the authoring of formal mapping definitions. Since the prototype required the mapping definition to be expressed in the W3C standard RIF format, it was necessary to translate the HALE schema mapping (which was in HALE's OML format) into RIF. An alternative, which was used for the hydrography and transportation datasets, was to write the RIF schema mapping definition document by hand. This was permissible for the prototype because the interface, and not the schema mapping definition tool, was the focus of the testing. However, in production environments, the authoring of the mapping definitions would have to be performed using a model-mapping tool such as HALE because RIF presentation syntax has a verbose format and it is easy to make both syntactic and semantic errors when writing it by hand.
3. The resulting mapping definitions were used by the TNS to transform the source data. Once the INSPIRE data had been output by the TNS, it was possible to validate it (see Section 4 for the description, and the outcome, of the validation process).

At this proof-of-concept stage, many of the mappings are not fully valid according to the INSPIRE logical schema, as described in Sections 4.2 and 4.3.

The prototype has demonstrated that the interface is capable of expressing a wide range of mapping definitions. Some examples (not an exhaustive list) are: -

- Straightforward class and attribute renaming: e.g. Dutch Perceelvlak to INSPIRE CadastralParcel.
- Enrichment of target features with values derived by computation from source attributes using spatial functions: e.g. computation of the area of an INSPIRE Cadastral Parcel based on the geometry attribute of the Dutch Perceelvlak class.
- Enabling feature instances to be related in the INSPIRE dataset based on spatial characteristics: e.g. use of the spatial "contains" join to link cadastral reference points from the Belgian Cadastre dataset with the cadastral parcels within which they are located, which enables population of the INSPIRE CadastralParcels.referencePoint attribute.
- Merging of multiple source feature instances to form one target feature instance: e.g. in the Swedish hydrography dataset, taking instances of the hl (hydrography line) and ms (hydrography area) layers and merging them to form one INSPIRE Watercourse feature instance.

- Filtering source feature instances based on an attribute value to produce INSPIRE feature instances under certain conditions: e.g. in the OS Ireland transportation dataset, points are filtered based on the FEAT_CODE attribute: a value of 5826 or 5828 means that an INSPIRE RailwayTransportNetwork.RailwayStationNode instance needs to be created in the target dataset.

The authors are confident that a production version of the Schema Transformation Network Service can be produced that will populate all of the INSPIRE logical schemas (source data permitting). This will require further development to increase the completeness of the mapping definition documents, the functionality of the schema mapping design tool, and any new spatial functions that need to be supported in the chosen transformation engine.

In a production environment this three-step process would almost certainly be repeated several times, in an iterative process, with questions and answers following between the data experts and the schema mapping definition team.

3.1.4 Testing Policy

Testing of the TNS followed our standard software process.

- Individual classes were developed using a test-driven approach based on JUnit 4 [13]. This helped identify defects early (when it is cheapest to fix the defect).
- Static code analysis tools like PMD [14] and Checkstyle [15] were used to automatically detect standard programming errors.
- Small units of functionality (such as modifying the namespaces of attributes) were tested independently. To support these cases, very simple logical schemas and test datasets were manufactured. This helped detect regressions early.
- Larger modules (like the module that translated RIF into Radius Studio Actions) were tested in an end-to-end manner. As development continued, these end-to-end tests were extended until they covered the actual logical schemas and transformations that the prototype supported.
- The whole system was integration-tested in a deployed environment. This made sure that the modules worked together successfully.

All this testing was managed in a continuous integration environment using a platform called Hudson (see [17] for more information). This environment re-runs all the tests each time the code is changed. This helped make sure that defects were detected early and that it was possible to identify which change had introduced the defect.

3.2 Design and Implementation Overview

The main stages of the prototype exercise were as follows:

- Design
- Implementation and test
- Evaluation and reporting
- Technical Guidance updates.

The main input to the prototyping exercise was the JRC and community reviewed TG [1] (version 1.0 and version 2.0).

Lessons learnt during the exercise were continuously fed back into subsequent iterations of the TG (version 2.0 and version 3.0), resulting in a refined and strengthened TG, with basic flaws removed.

3.2.1 Prototype Design Goals

The objectives of this prototype are to:

1. Demonstrate that the interface described by the TNS TG [1] is:
   - Possible to implement using standard development tools
   - Capable of passing all the parameters required for successful transformation of source data into the INSPIRE logical schemas.
2. Demonstrate that the recommended RIF dialect (RIF-PRD) is:
   - Capable of describing the set of schema mapping definitions required to perform successful transformation from source logical schemas to the target INSPIRE logical schemas.
   - Capable of being translated into the configuration required by an existing spatial transformation engine.
3. Provide feedback that may help clarify and/or correct the current draft of the technical guidance.
4. Contribute towards videoed and live demonstrations that promote the understanding and acceptance of the TNS TG by both technical and non-technical audiences.

The prototype is constrained to work with currently available source data.

3.2.2 Main Prototype Design Packages

For an overview of the architecture as a whole, please refer to Section 6 of the Technical Guidance document [1].

The following details provide further information on the main components developed as part of the prototype. This information does not provide detail about the client used to trigger the service as part of the demo; this was a trivial web client application with a handful of text fields to enable configuration of the source dataset location, source and target logical schema locations and schema mapping definition filename, plus an ‘invoke’ button.

Note: The details in this section assume the reader to be an experienced software engineer, familiar with recognised software engineering and web service development terminology, as well as the Unified Modelling Language (UML).

3.2.2.1 Schema Transformation Network Service

Figure 9 shows the Schema Transformation Network Service sub-components.

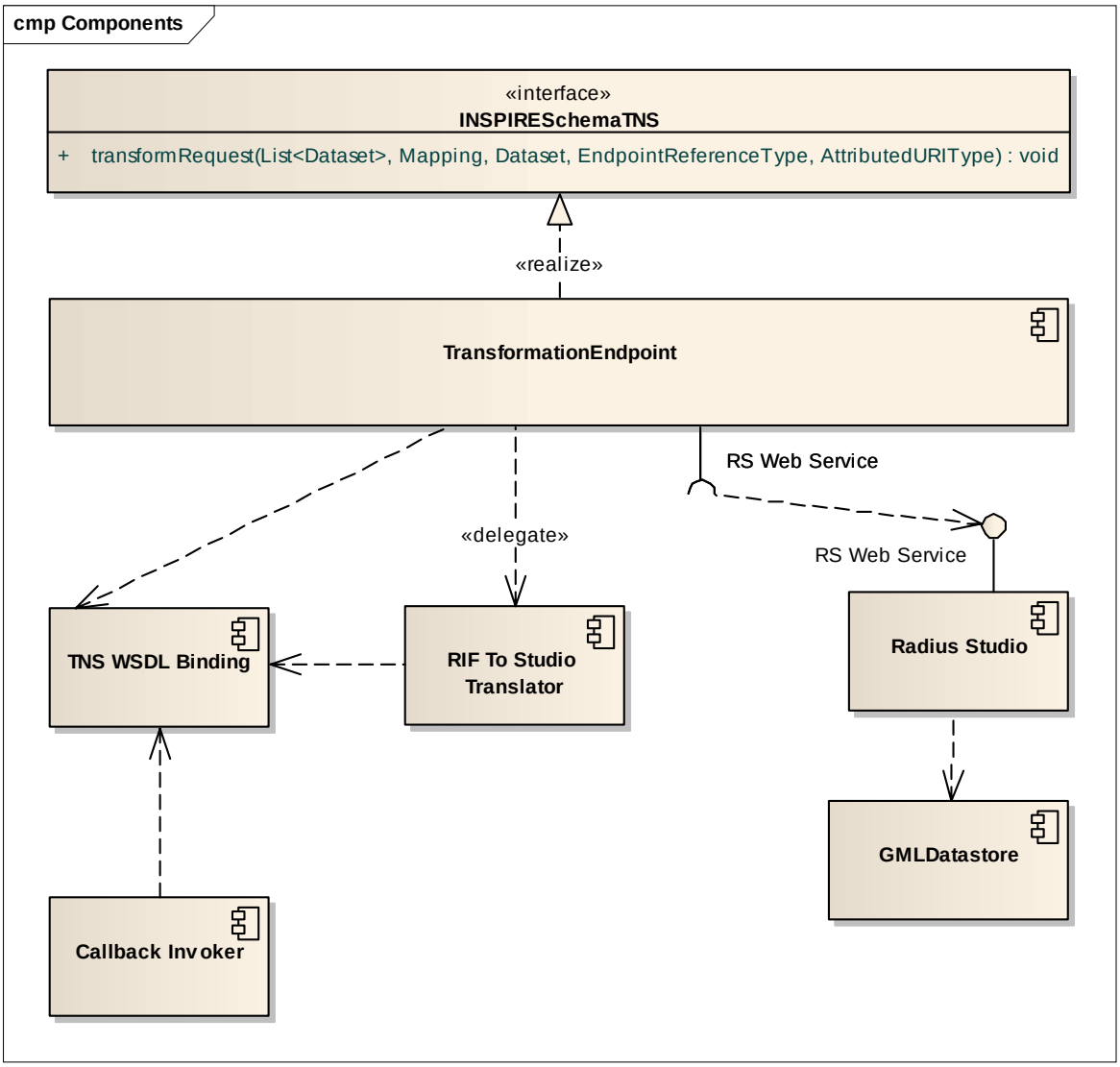

Figure 9 Prototype Schema Transformation Network Service sub components

TNS WSDL Binding

This component provides an auto-generated library of interfaces and classes that describe the SOAP interface of the Transformation Network Service. It also provides simple implementations that can invoke the service and any callback services. It is generated from the web service's WSDL and supporting XSD documents. It includes the RIF-PRD bindings, which is why the RIF to Studio Translator also requires it.

Transformation Endpoint

The Transformation Endpoint is responsible for receiving and responding to the SOAP synchronous requests. The interfaces and classes used to implement the service are drawn from the WSDL Binding component.

Once an incoming transformation request has been received, this component digests its arguments as required. The Transformation Endpoint component is responsible for managing the state of the transformation operation. Internally, it uses delegates to perform all interactions with external systems (relaying requests to and receiving responses from the Radius Studio Web Service) and to perform all complicated processing (RIF To Studio Translator).

It is responsible for managing any polling required by asynchronous operations. It is also responsible for collecting any exceptions thrown by its delegates and forwarding them, via the Callback Invoker, to the client.

Upon successful completion it is responsible for informing the client, again using the Callback Invoker.

Figure 10 shows the high-level class diagram for the Transformation Endpoint.

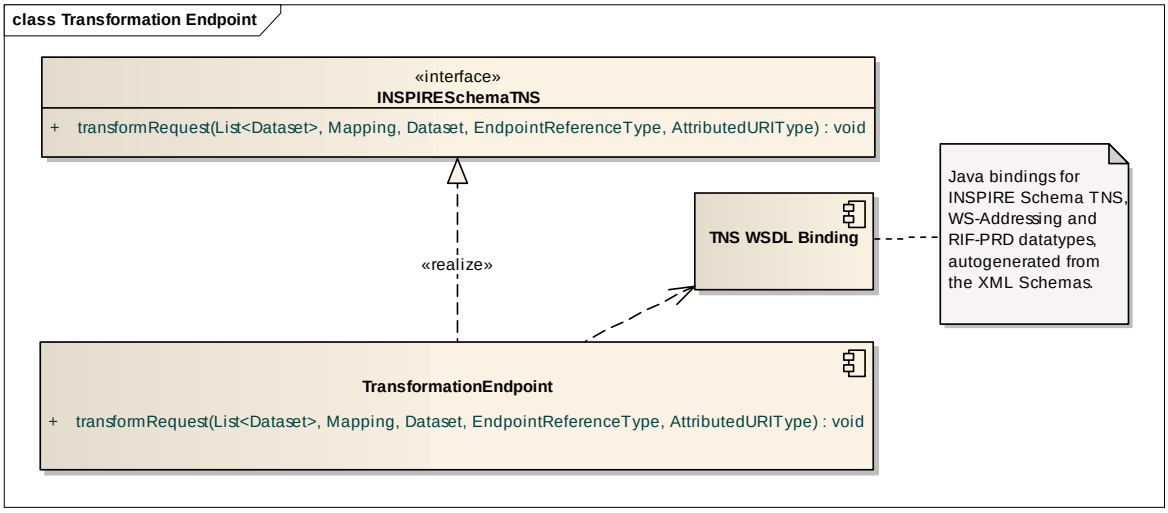

Figure 10 Transformation Endpoint class diagram

Table 4 describes the single method and its parameter datatypes for the Web Service Endpoint class. Note that the INSPIRESchemaTNS interface is implemented by this component and is not discussed separately.

Table 4 Method and Parameter Types for Web Service Endpoint

Name	Type	Description
transformRequest	Method	A request to the Schema TNS to transform one or more source datasets to the INSPIRE format. NB this is as specified by the INSPIRE regulation [4] (see Annexe III Section 1 of that document).
List<Dataset>	Parameter Datatype	The list of source datasets to be transformed (note, in the prototype only one dataset was able to be transformed in a request, however the interface permits multiple source datasets).
Mapping	Parameter Datatype	The RIF-PRD schema mapping definition file to supply the rules for the transformation.
EndpointReferenceType	Parameter Datatype	WS-Addressing datatype that provides the replyTo address of the client for callback purposes.
AttributeURIType	Parameter Datatype	WS-Addressing datatype that contains the request messageID to enable the callback to notify the client correctly.

Callback Invoker

The Callback Invoker component (shown in Figure 11) is used to send a callback to the clients where required, either in case of errors in processing of source data or to report the completion of the transformation process. As a stateless component, it is not responsible for tracking the state of operations, merely for providing the functionality to perform any required client callbacks.

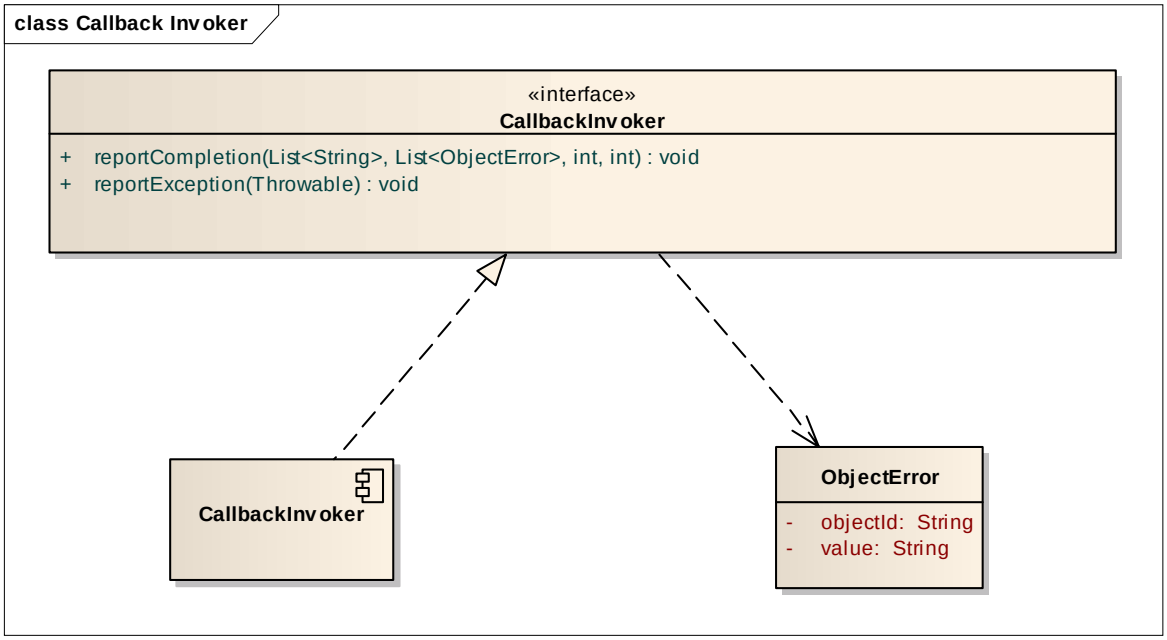

Figure 11 Callback Invoker class

This callback functionality is used both in the cases of successful and exceptional operations. The interfaces and classes used to perform the callback are drawn from the WSDL Binding component. The interface methods and parameter datatypes for the Callback Invoker are described in Table 5.

Table 5 Callback Invoker interface methods and parameters

Name	Type	Description
reportCompletion	Method	Notifies the client application that the transformation process has completed.
List<String>	Parameter Datatype	Contains information about the successful transformation (i.e. number of features processed, number of feature for which an error was raised, and any system exceptions that were reported during processing).
List<ObjectError>	Parameter Datatype	Provides a list containing the errors individual objects may have thrown within an overall successful transformation.
int	Parameter Datatype	Number of source objects transformed.
int	Parameter Datatype	Number of target objects created.
reportException	Method	Notifies the client application that an error has occurred in the processing of the transformation request.
Throwable	Parameter Datatype	The exception message.

RIF To Studio Translator

The RIF To Studio Translator (Figure 12) is responsible for converting a collection of RIF-PRD rules into a collection of Radius Studio Actions. In the prototype, this translator is a stateless component and performs no caching of its resources. If the translation process proves to be expensive, future production implementations may wish to revisit this design choice.

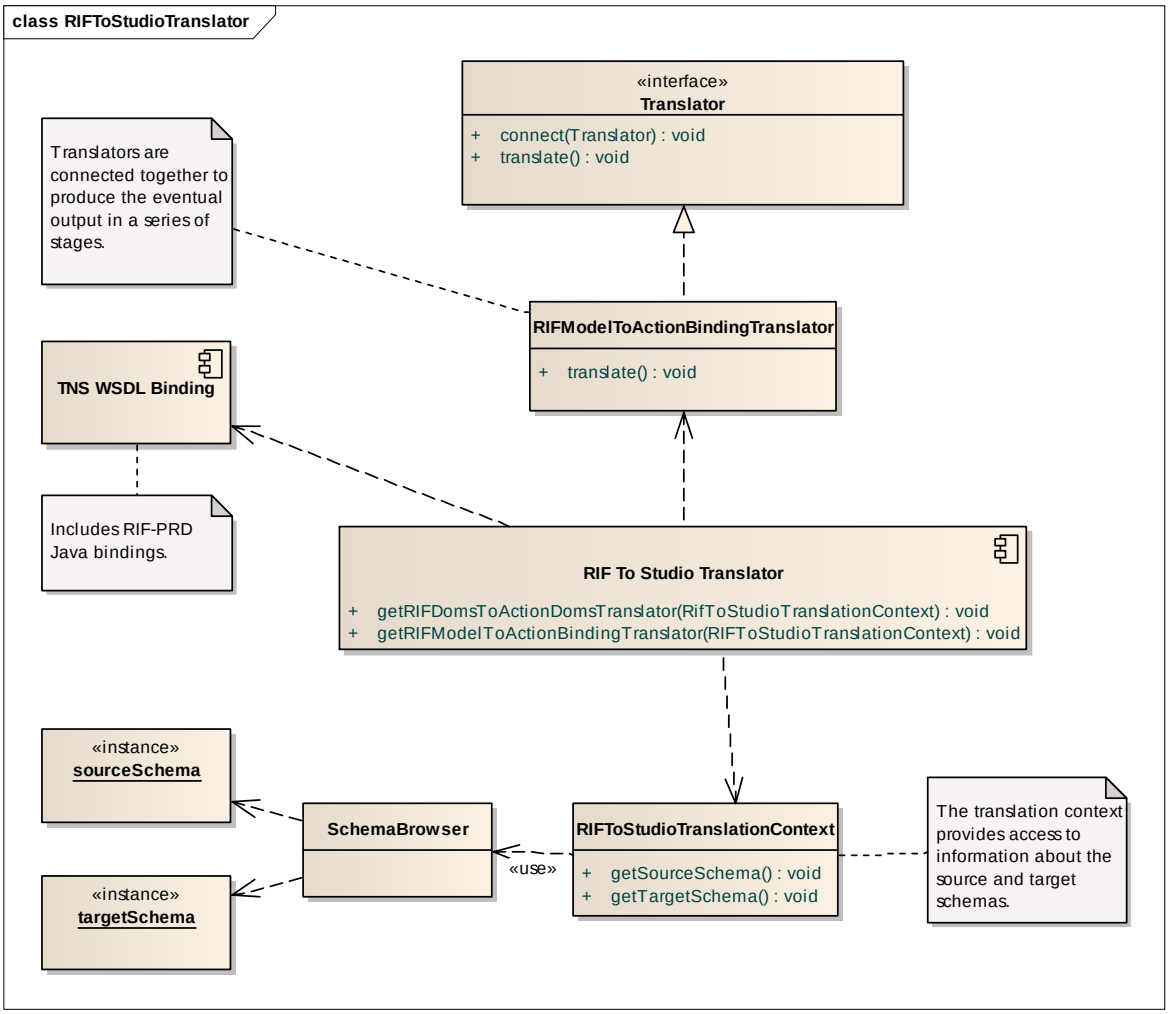

Figure 12 RIF To Studio Translator class diagram

The design of the RIF To Studio Translator involved some complexity as it required to convert transformation rules from their RIF-PRD format to 1Spatial's own internal rules language. We applied the Translator Pattern [19] to this problem.

In the prototype version, a series of translator objects is initialised, to process the translation task as a pipeline of discrete steps. Using the connection of one translator to another, it was possible for the RIF To Studio Translator to be composed of multiple translators, with this structure being invisible to the calling class which only had to call the translate() method on one translator. Each translator would automatically perform its own translation then call the next in the chain of connected translators until the whole chain was completed and the final result could be returned.

The full detail is not shown here, because in actual fact eight translator implementation classes were connected together to process activities such as re-writing of namespace prefixes, removal of redundancy from the rule structure and mapping of attributes.

In addition, to inform the translation process, it is necessary for the translators to have access to the RIF To Studio Translation Context. This context provides information about the source and target logical schemas, which is vital because different rules formats often put emphasis on different aspects of a schema and it is necessary to understand the relationship between feature classes and attributes so as to transcend the special assumptions made about a schema mapping by a specific rules format. A SchemaBrowser provides the interface to the sourceSchema and targetSchema instances.

The internal control flow of the RIF To Studio Translator is shown in Figure 13.

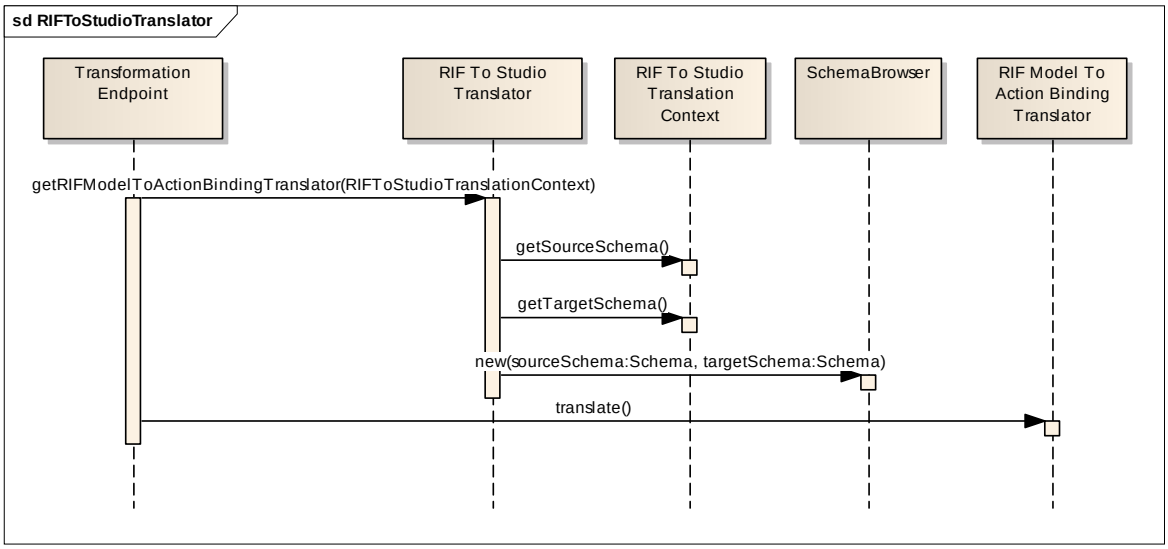

Figure 13 Process flow for the RIF To Studio Translator

The messages involved in this interaction are described in Table 6.

Table 6 RIF To Studio Translation process messages

Object	Method	Description
RIF To Studio Translator	GetRIFModelToActionBindingTranslator	Takes a translation context and returns the translator required to effect the translation from RIF-PRD XML Document to a set of Radius Studio Action Java bindings held in heap memory.
RIF To Studio Translation Context	getSource	Returns the structure of the source logical schema in terms of its classes and attributes.
RIF To Studio Translation Context	getTarget	Returns the structure of the target logical schema in terms of its classes and attributes.
Schema Browser	new	Instantiates a Schema Browser to enable browsing of the contents of both source and target schemas.
RIF Model To Action Binding Translator	translate	Translates the input RIF mapping into 1Spatial's own rules language so that Radius Studio can apply the transformations that are required.

Radius Studio

Radius Studio (version 2.1) is responsible for running the actual transformation process that executes a series of tasks to load the data, applies a series of actions that perform the schema translation, and generates the data to the required target destination.

GML Datastore

This component is responsible for converting the contents of the Radius Studio cache into a GML version 3.2.1 that is compliant with the target INSPIRE schema, in order for it to process and output the transformed data.

For this prototype, the logical schema will be exported to an external FTP site. Future production implementations should support the exporting of this data directly to a WFS-T (as per TNS TG [1]).

3.2.2.2 Model Mapping Designer

Figure 14 shows the prototype third-party schema mapping definition client's sub-components.

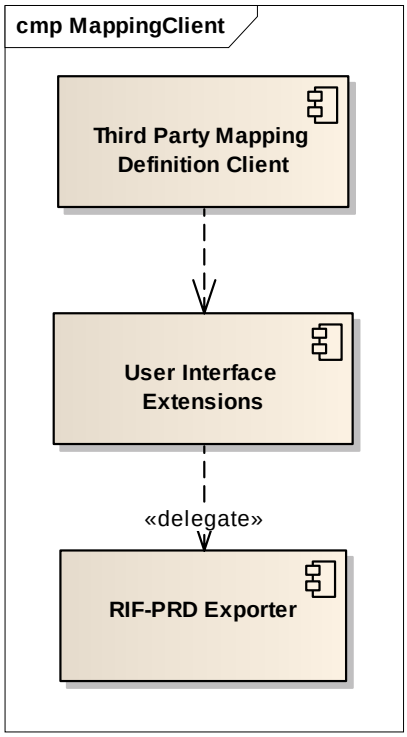

Figure 14 Prototype mapping client sub components

Third Party Mapping Definition Client

This is a third party component that allows the user to define, store, and load schema mapping definitions between logical schemas. In the prototype, the Humboldt Alignment Editor (HALE) was used to perform this role.

There is no requirement for this client to store its configurations using RIF-PRD.

User Interface Extensions

This component is responsible for providing the user interface extensions required to trigger the rule export process. This should ideally allow the user to select which mappings to export, and configure the connection details of the XML repository.

RIF-PRD Exporter

The RIF-PRD exporter is responsible for exporting a selection of schema mapping definitions from the mapping tool using the normative RIF-PRD XML encoding. These should be returned as a collection of W3C DOM Documents, so that they may be either serialised to disk, or passed directly to an XML Repository Proxy or other persistent format for storage. Note that the schema mapping definitions do not contain their own inline metadata, because it is anticipated that an XML Repository would handle the storage and update of metadata for the objects stored in it (see Section 5 Conclusions for details of the role the XML Repository – which is described in the TG [1] Section 5.5 - played during the prototype).

Figure 15 shows the class diagram for the RIF-PRD Exporter.

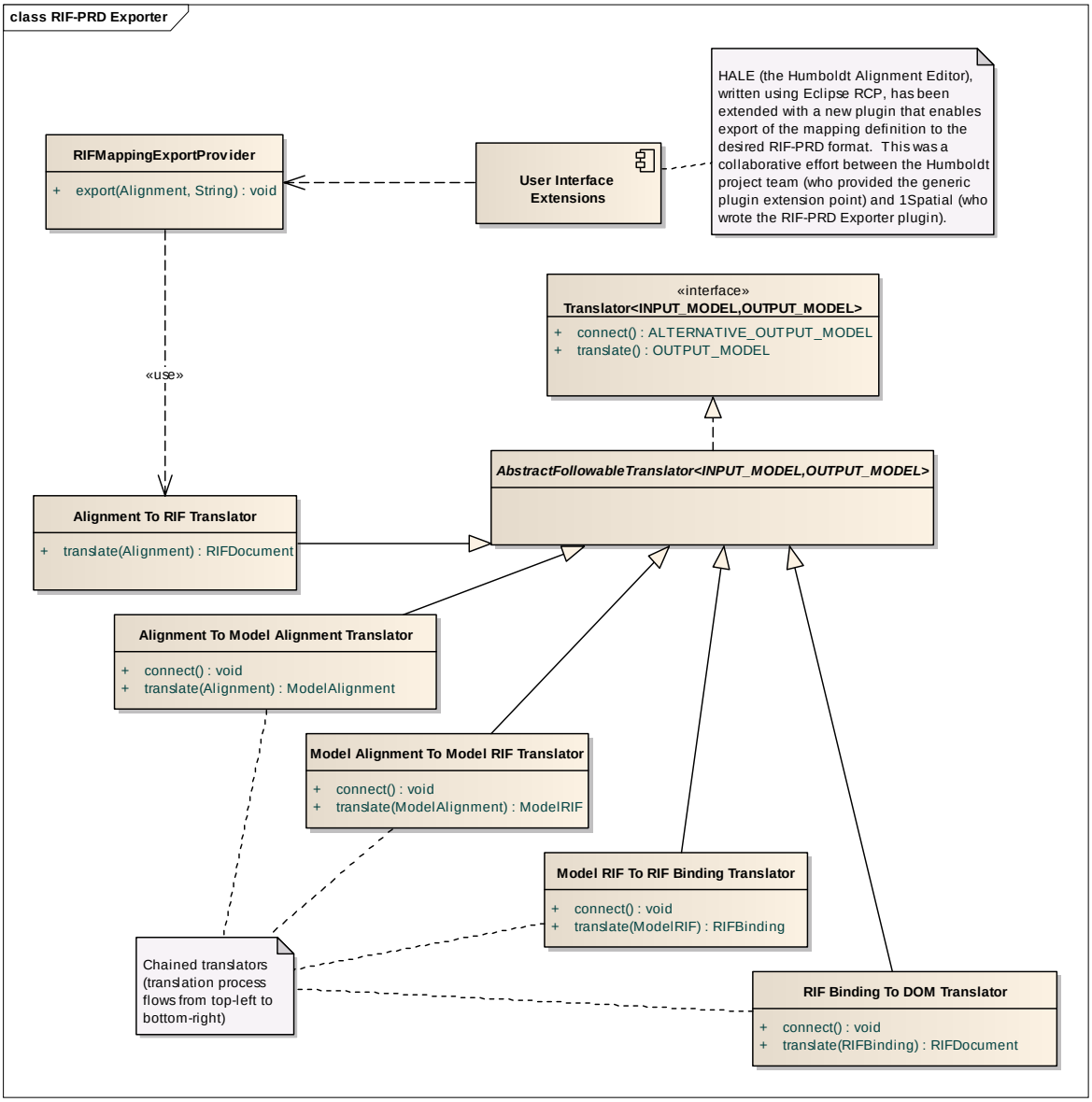

Figure 15 RIF-PRD Exporter class diagram

The RIF-PRD Exporter was a collaborative effort between the EC-funded Humboldt project and 1Spatial. The Humboldt team provided a generic Eclipse RCP mapping export plugin extension point to the HALE editor, for which 1Spatial then wrote a plugin to handle the export.

The design pattern used to code the AlignmentToRIFTranslator is described in Section 3.2.2.1 under the heading *RIF To Studio Translator*. The process of translation from a HALE alignment (written in Ontology Mapping Language) to RIF-PRD is essentially the same as that from RIF-PRD to 1Spatial's Radius Studio Rules Language. Some aspects of the translation pipeline were far from trivial to resolve. For example, it was found that OML defines mappings such that classes and properties are placed on the same level as one another, whereas RIF-PRD places all attributes in the context of the containing class. While processing a mapping, when the translator parses an attribute, it is necessary to discover which class it belongs to, in order to construct a mapping between classes in the output format, that includes that attribute. This required reading up the source logical schema's inheritance tree of classes to find the class in which a given attribute was defined. A corresponding activity was required in order to trace the class to which target attributes belonged. It was therefore necessary to develop a means of supplying a translation context containing information about all classes and attributes in both source and target schemas, to inform the translation process. This was done using an XSOMParser (for XSOM, see Appendix A and [20]). This development was as useful for the OML to RIF-PRD translation as from RIF-PRD to Radius Studio Actions and the relevant code was shared between the two parts of the prototype.

Table 7 describes the classes and methods shown in the above diagram.

Table 7 RIF-PRD Exporter components and classes

Name	Type	Description
User Interface Extensions	Component	The extension point provided by the Humboldt team to enable extension of HALE to incorporate a RIF-PRD export functionality.
RIFMappingExportProvider	Class	The Eclipse RCP plugin that fits into the extension point and performs the translation and output of the RIF-PRD.
Translator	Interface	Defines the basic operations of the translation process flow.
AbstractFollowableTranslator	Class	Implements a generic follow-on method that enables configuration of translator chains.
AlignmentToRIFTranslator	Class	Builds the translator chain and provides the external point of contact for the Eclipse RCP plugin.
AlignmentToModelAlignmentTranslator	Class	Translates a HALE Alignment in OML format into an internal meta-format that is logically close to OML.
ModelAlignmentToModelRIFTranslator	Class	Translates the first internal meta-format into a second meta-format that is logically close to RIF-PRD.
ModelRIFToRIFTranslator	Class	Translates the second meta-format into a set of RIF-PRD Java binding objects.
RIFBindingToDOMTranslator	Class	Translates the RIF Java binding objects into a DOM Document ready for output as XML.

The process flow during the Alignment to RIF-PRD translation is shown in the sequence diagram in Figure 16.

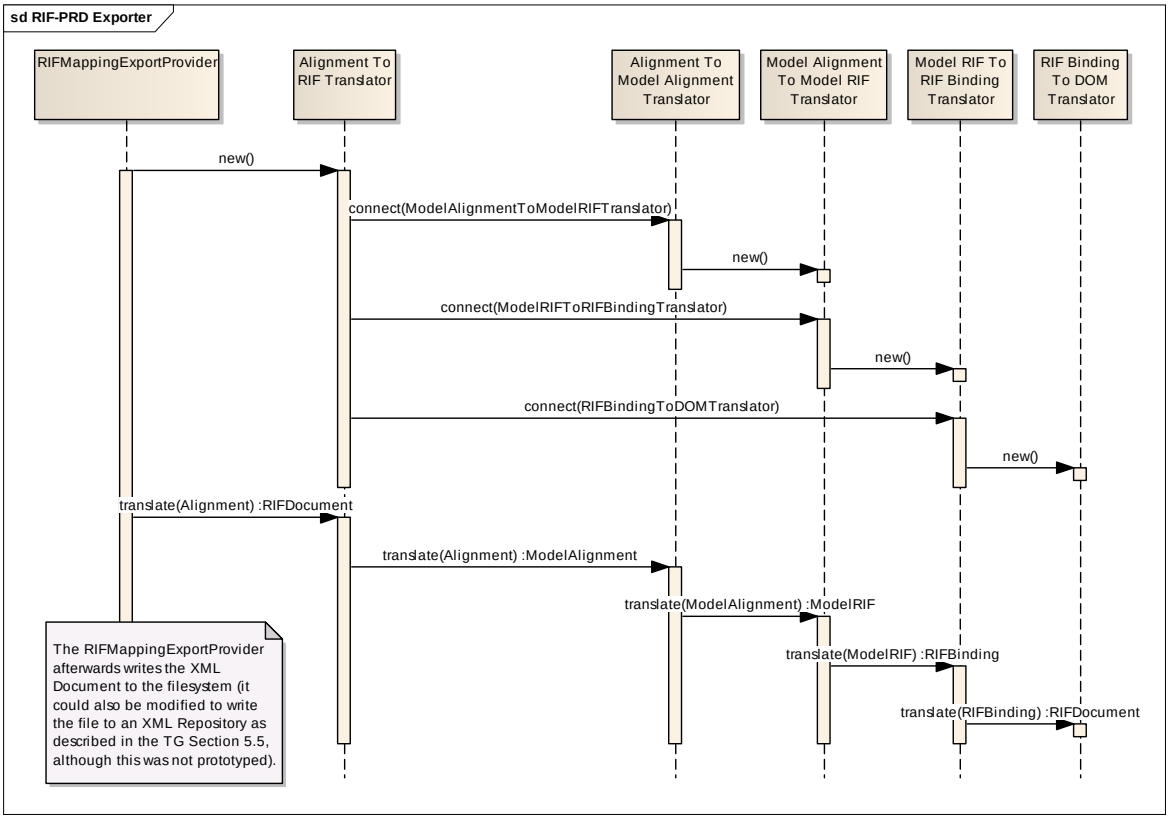

Figure 16 Sequence diagram for RIF-PRD Exporter

The messages for each object in the above sequence diagram can be described in free text without the need for a table. The role of the RIFMappingExportProvider is simply to create the first translator, call `translate()` on it, and then persist the resultant XML Document. In each case of a translator, the `new()` method constructs an instance of the translator; the `connect()` method instructs the translator to chain to a follow-on translator which is specified by the parameter to the `connect()` method; and the `translate()` method called on the first translator initiates the chain of translation which results in the final, fully-translated RIFDocument being produced.

3.2.3 Communication between Prototype Components

Communication between components within the prototype uses the protocols detailed in Table 8.

Table 8- Communication protocols.

Protocol	Version	Where Used
WFS	Download Service specification ISO19142 (which is based on 1.1.0). Most tools support 1.0.0. TNS should support both because WFS 1.1.0 is an increasingly widely supported standard.	Web Feature Service (WFS) protocol is used by the TNS to access the source data set(s) (i.e. feature data) to be transformed.
FTP	Anything supported by Java's URL class: for example, this could be	File Transfer Protocol (FTP) is used by the TNS to return the generated INSPIRE GML (version 3.2.1) document to the client.

	a local file system URL in the form `file://host/path` or a remote network URL in the form `http://host/context path`.	
JDBC	4.0	Radius Studio uses JDBC to connect to its configuration repository. However, JDBC is never used in the prototype to transfer feature data, it is only used for persisting the internal configuration of Radius Studio.
SOAP	1.1	SOAP web services are used to mediate interactions between: • The client and the TNS; • The Transformation Endpoint and the Radius Studio Web Service.
WS-Addressing	1.0	To provide asynchronous callback to the client application when the Transformation Endpoint has completed the Transform Data operation .

3.2.4 Two-Phased Transform Operation

The request to perform transformation is asynchronous. The following text and sequence diagrams describe the two phases of the operation. Please note, for reasons of brevity, we only display method parameters that are of central relevance for understanding the core workflow.

The creation of the thread to support the second phase could be implemented with a Message Driven Bean, a Timer or any other suitable mechanism. For the prototype it was felt acceptable to use Java SE style threading (Runnable or any of the java.util.concurrent Executor tools), however this would not be acceptable in a production environment as these approaches do not allow a container to manage concurrency in a scalable manner.

3.2.4.1 Phase 1: Client Sends Request to TNS

As shown in Figure 17, a client sends a transformation request to the Transformation Network Service endpoint. The endpoint digests the arguments and starts an internal thread to handle the process and then returns to the client.

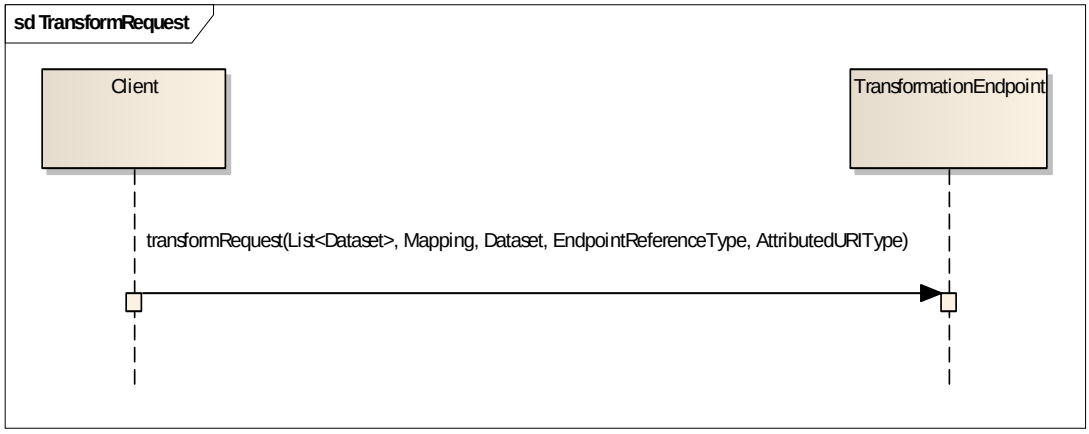

Figure 17 Transform request sequence diagram

Table 9 describes the objects involved in the Transform request process flow, and their messages.

Table 9 Transform request objects and messages

Name	Type	Description
Client	Object	The client application.
Transformation Endpoint	Object	Transformation Network Service endpoint.
transformRequest	Message	Requests the start of a transformation.

3.2.4.2 Phase 2: TNS Processes Request

As shown in Figure 18, the internal thread that handles the transformation process performs the following interactions. If any interaction throws an exception, it is caught and the process flow jumps to the last interaction (performCallback), where the exception details are passed to the TNS Client.

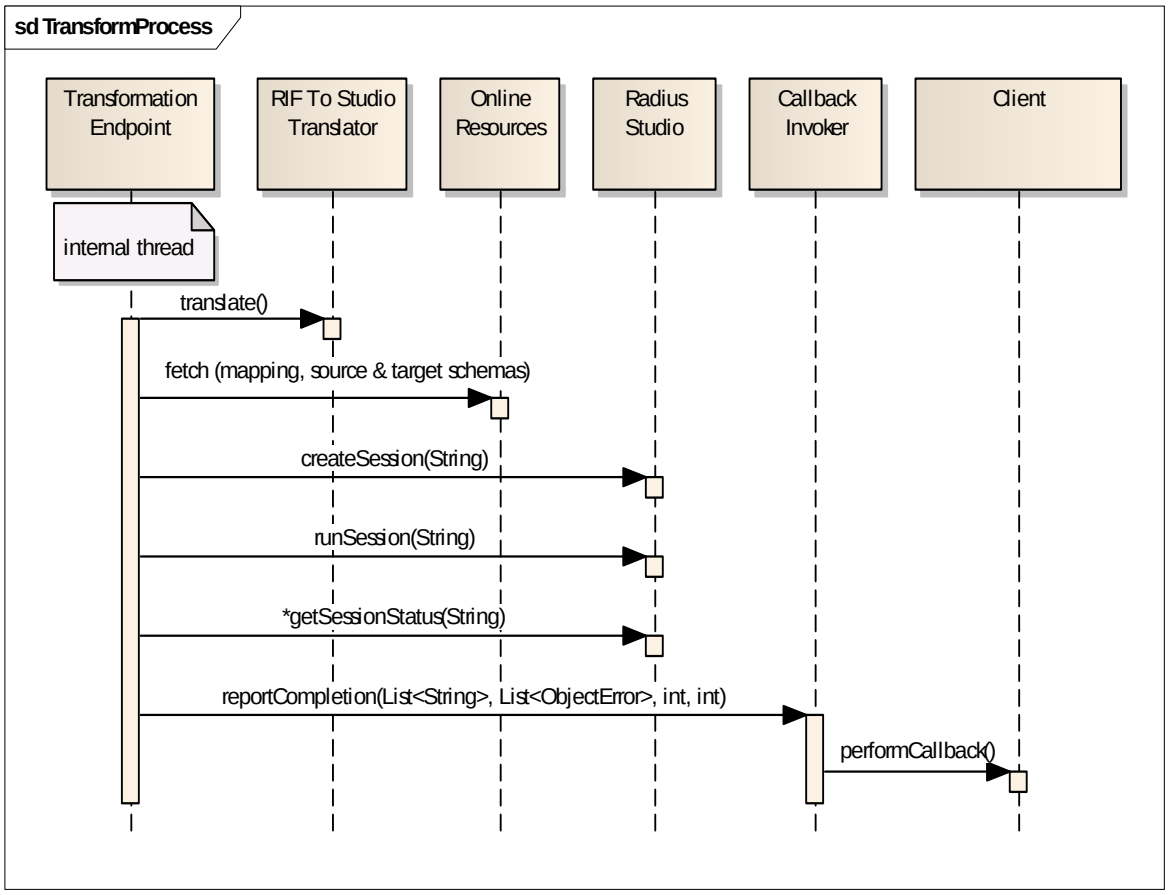

Figure 18 Transform process sequence diagram

Table 10 describes the objects and messages involved in the processing of the transformation request**.**

Table 10 Transform process objects and messages

Name	Type	Description

Transformation Endpoint	Object	Responsible for receiving and responding to the SOAP synchronous requests, as well as managing the internal state of the transformation operation.
RIFToStudioTranslator	Object	Responsible for converting a collection of RIF-PRD rules into a collection of Radius Studio Actions. NB this component gets the URL information for the online resources from parameters passed in from the Transformation Endpoint, which receives them as part of the parameters to the TNS transformRequest() operation.
translate	Message	Sends a request to translate the RIF-PRD mapping definition document into a set of Radius Studio actions, encoded using 1Spatial's own rules language.
Online Resources	Object	This is a logical component which describes the Apache web server, FTP service and WFS, which provide the TNS access, respectively, to the INSPIRE XSDs, RIF-PRD schema mapping definition documents and source logical schema XSDs.
fetch	Message	This is a logical operation which describes the HTTP get() operation that was used to retrieve the INSPIRE XSDs from the Apache web server, the FTP get() operation which retrieved the RIF-PRD mapping definition documents, and the WFS DescribeFeatureType operation which enabled the source XSDs to be made available to the transformation.
Radius Studio	Object	Radius Studio spatial data validation platform, accessed via its web service interface.
createSession	Message	Creates a Radius Studio session. The parameter is the id of the session to create.
runSession	Message	Starts a Radius Studio session. The parameter is the id of the session to start.
getSessionStatus	Message	Polls every 200 milliseconds to check if session has completed. The parameter is the id of the session of which to check the status.
Callback Invoker	Object	Responsible for notifying the client that the session has completed, providing the numbers of features input and output, and any errors raised by

		individual features or by the system.
reportCompletion	Message	Notifies the client that the session has completed and provide the location at which the results can be downloaded.
Client	Object	In the prototype, this was a simple web application client which enabled the user to specify the parameters to the transformation operation and initiate it, and notified the user of the progress and eventual completion of the transformation process or any service exceptions thrown.
performCallback	Message	Enables the client to be notified that the session has completed and that it is possible to download the transformed data (or that a service exception has occurred, and the details of it).

3.2.5 Web Mapping Client Sequence Diagram

Figure 19 is a sequence diagram showing the export of RIF from third-party mapping definition tool. In the prototype, there was a manual step of transferring the saved export file to the FTP server, although it could have been integrated into the mapping definition tool as this diagram suggests.

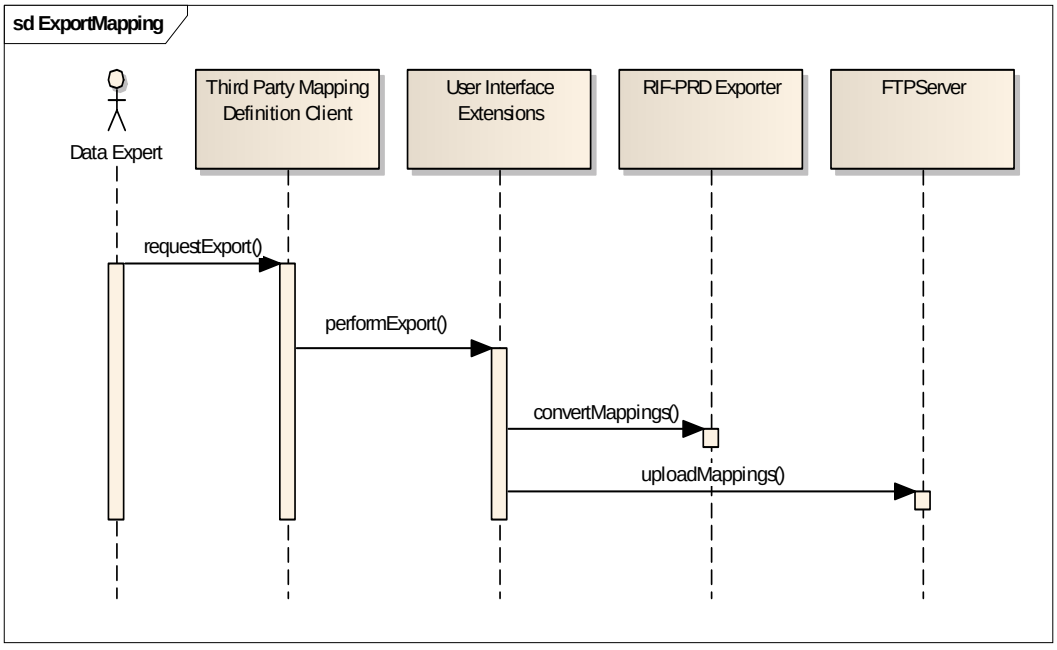

Figure 19 Export Mapping sequence diagram.

Table 11 describes the components that interact to perform the Export Mapping workflow, and their messages.

Table 11 Export Mapping components and messages

Name	Type	Description
Third-Party Mapping Definition Client	Component	A third party component that allows the user to define, store, and load schema mapping definitions between logical schemas.
requestExport	Message	Requests export of the schema mapping definition into an XML document in RIF-PRD format.
User Interface Extensions	Component	The extension point at which the RIF-PRD Exporter could be plugged into the schema mapping definition client.
performExport	Message	Exports the schema mapping definition in the selected format (the user interface provided several formats that could be selected; the prototype version was extended to include the option of RIF-PRD).
RIF-PRD Exporter	Component	This component handles the translation of the schema mapping definition from the internal format of the mapping definition client (in the case of the prototype this was Ontology Mapping Language [23]), into the W3C standard RIF-PRD format.
convertMappings	Message	Translates the schema mapping definition from its internal format to the RIF-PRD format.
FTP Server	Component	Provides the network location to which to persist the transformed data.
uploadMappings	Message	This was actually done as a manual file transfer process although it could have been integrated into the tool.

3.3 Prototype Deployment

The prototype demonstration environment is illustrated in Figure 20.

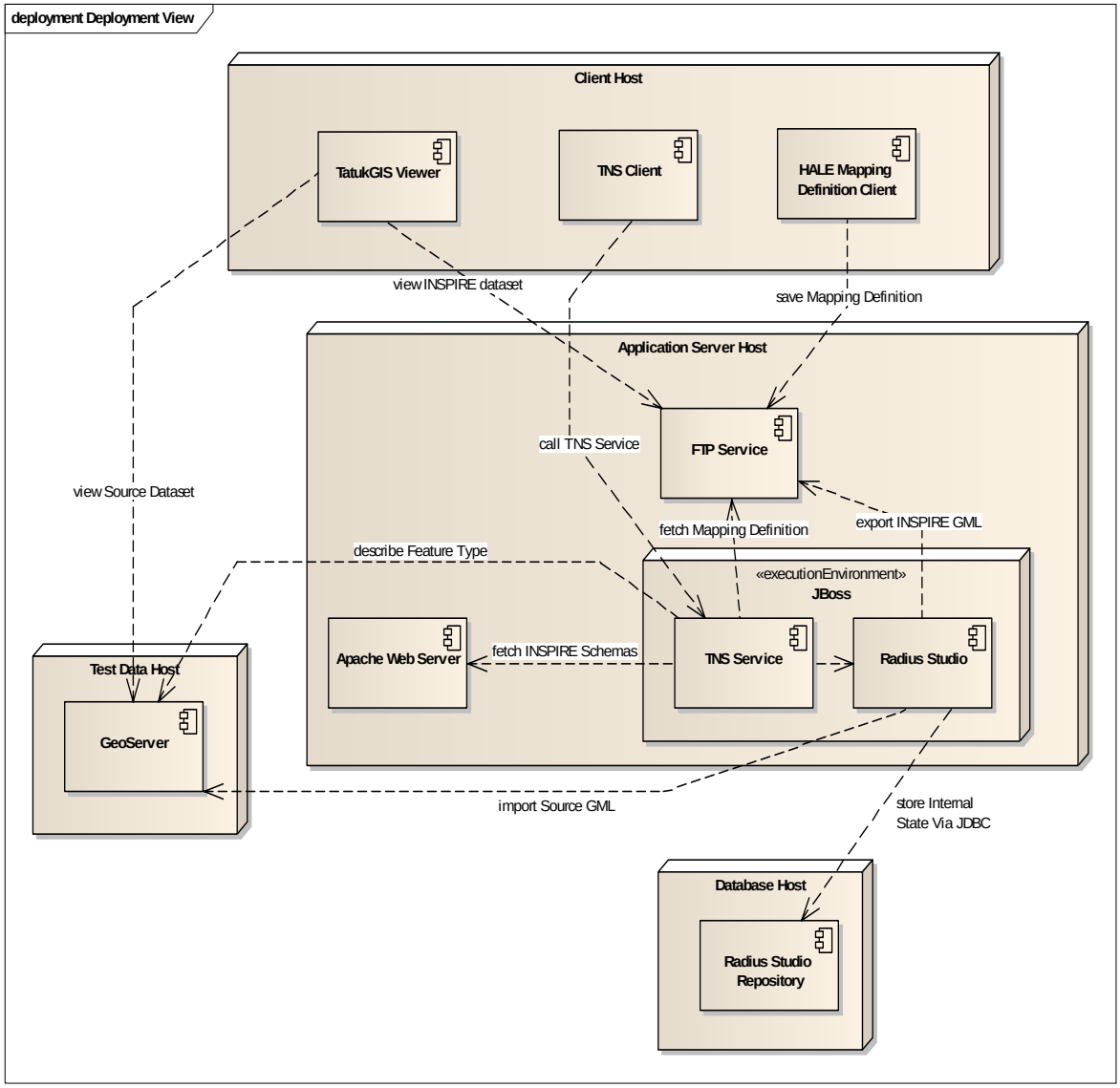

Figure 20 Prototype deployment view

In the above Figure 20, the TNS Client component is called by a web browser.

In the prototype, the Map Viewer component's contract was fulfilled by the TatukGIS GML Viewer [10]. This enabled the INSPIRE data, having been output by the transformation, to be visualised in a user interface.

3.4 Performance Analysis

Whilst performance statistics generated from prototypes are not usually precise indications of the production behaviour of a system, they do give broad information as to the feasibility of use (and implications for collaborating services) of selected technologies, architectural models and resource allocations.

Performance testing provides data enabling identification of the relationship (straight-line or more complex) between the hardware and software resources assigned to the transformation services, and the throughput of data.

3.4.1 Test Environment

Two machines were used in the following experiment. Both machines were virtual and hosted on VMWare ESX server version 3.5 running on HP ProLiant BL460c G1 blades. The specification of the virtual machines was as follows:

GeoServer Virtual Machine

Operating System: Redhat Enterprise Linux 5 (64bit)

CPU: Intel Xeon X5355 (1 core)

RAM: 4GB

Disk: 20GB on 1GB/s iscsi san using 7200RPM disks

Transformation Network Service Prototype

Operating System: Redhat Enterprise Linux 5 (64bit)

CPU: Intel Xeon X5355 (1 core)

RAM: 2GB

Disk 22GB on 1GB/s iscso san using 7200RPM disks

This represents a very modest test environment. For comparison, an enterprise-grade feature-processing engine might look more like 72GB of RAM, spread over 6 server nodes (with 8 cores per node).

3.4.2 Performance Results

The following results (Table 12) were taken from a single transformation call for each of the test datasets.

Table 12 - Prototype TNS performance results.

	Number (features)			Time taken (seconds)				Rate (features/second)			
Provider/Theme	Read (Source) A	Write (INSPIRE) B	Total (A+B) C	Read D	Transform E	Write F	Total (D+E+F) G	Load (A / D) H	Transform (C / E) I	Write (B / F) J	Overall (C / G) K
Dutch Kadaster	**136903**	**136903**	273806	**205.08**	**88.85**	**128.50**	422.42	667.58	3082	1065	648
Belgian Cadastre	**34034**	**17016**	51050	**68.77**	**20.32**	**27.50**	116.59	494.93	2512	619	438
National Land Survey Sweden	**4327**	**4325**	8652	**4.47**	**38.66**	**14.86**	58.00	967.36	224	291	149
Statens Kartverk Norway	**3230**	**3230**	6460	**42.95**	**4.84**	**10.09**	57.88	75.20	1334	320	112
OS Ireland	**34236**	**18053**	52289	**62.89**	**12.18**	**20.19**	95.25	544.42	4293	894	549
LPS Northern Ireland	**25923**	**77077**	103000	**62.84**	**33.08**	**58.69**	154.61	412.55	3113	1313	666

Note: Measured data is shown is **bold**, derived data is shown in normal weight font. Derivation formula is written under column names.

In general, we can see that the overall rate (K) is correlated to the theme, with the hydrographic theme being the slowest. This is not surprising, as hydrographic features typically have more complicated geometries (and therefore data volume) than either cadastral or transportation features. We can also see that the rate of reading/writing correlates to the size of the dataset. This is because both the read and write times comprise two components: a fixed set-up cost, and a per-feature transfer cost. For the smaller datasets, the transfer time is dominated by the fixed set-up cost. In a production system, judicious use of configuration caching could significantly reduce this setup cost.

The read cost for obtaining the source data was generally greater than the write cost for outputting the INSPIRE data. This seems to be contrary to general computing experience in that writing usually costs more since it involves obtaining exclusive locks and allocating memory resources, among other factors. However, it must be recalled that the read operation is being applied to GeoServer, which is deriving the source data from Shape files on-the-fly, whereas the write operation is being done to an FTP service, which is very lightweight and fast and does no format or encoding transformation. The only exception to the rule here is the Swedish hydrography data; this difference may be explicable on the basis of the complexity of the hydrography logical schema and/or the presence in the Swedish data of many large polygons with multiple vertices. However, it is equally possible, given the modest size of the samples, that environmental factors such as availability of memory and CPU time affected the readings.

Even with the modest infrastructure used by this test, reasonable transfer rates were achieved. The stateless design of the Schema Transformation Network Service means that adding additional processing resources can smoothly scale this throughput. Radius Studio, for example, is backed onto a scalable grid-based technology, which allows extra resources to be added to the engine to increase performance. This grid-based technology also delivers fail-over support, which in turn is essential in providing highly available services. Other vendors may use grid-based or other mechanisms to deliver similar non-functional characteristics; the prototype has proved that this kind of enterprise technology is compatible with the interface defined by the Technical Guidance [1].

3.5 Technical Guidance Update Traceability

Please note that the prototyping exercise did not trigger any updates to the Technical Guidance [1], as the interface definition and system components were found to handle the transformations that were required.

4. Report of Output Data Conformance to Target Schema

This section provides a description of the validation method and tool used to assess the conformance of the output data with the INSPIRE logical schema.

4.1 Validation Method and Tool

At this stage, basic schema conformance was tested using a well-respected third-party tool called XMLSpy® [16]. Version 4.3 of XMLSpy® was not capable of validating the larger datasets (the Dutch and Belgian cadastral parcel datasets as transformed into the INSPIRE format). When this happened, individual features (and the containing feature collection) were extracted from the generated dataset and validated independently, however the complete dataset was not validated.

For simplicity of testing, the generated documents referenced a logical schema location inside the 1Spatial firewall (for both the INSPIRE logical schema and the OGC logical schemas). To validate these documents in another environment, the schemaLocation hint would have to be updated accordingly.

To validate the documents using XMLSpy® two further modifications are required:

1) Removal of the schema location that starts with 'vfszip' (this referred to a jar file resource XSD file used by XSOM which the automated transformation process included in the generated INSPIRE GML data; this would be removed in a solution prepared for a production environment).

2) Removal of the duplicated XML Schema namespace declaration which was generated by the transformation, owing to a trivial defect in the GML Datastore (this could have been resolved by further investigation).

XMLSpy® was thus able to validate the transformed datasets, including the cardinality, controlled lists, domains and co-occurrence constraints.

4.2 Test Data Transformation Conformance

The non-conformances noted in Table 13 have been identified with the current transformation output. In a production system, each one of these non-conformances would be tracked as defects and resolved.

Some of them (like the date/time formatting issues) can be addressed in the mappings through the appropriate use of RIF built-ins. Others, like the gml:posList non-conformance in the transformation schemas, may need changes to the GML export code. These are required due to current limitations in the prototype GML datastore. The changes relate to the output of the XML rather than any aspects of the data or transformation process. They would not be needed in production systems. Table 13 shows test data non-conformances.

Table 13 - Test data non-conformances.

Dataset	Non-Conformance Error Message	Explanation
Dutch Kadaster	gml_id on feature collection "1" is not accepted	This is a trivial defect in the generation of internal identifiers by the prototype GML datastore.
	areaValue needs uom attribute	This is a defect in the handling of units of measure within the prototype GML datastore.
Belgian Cadastre	gml_id on feature collection "1" is not accepted	This is a trivial defect in the generation of internal identifiers by the prototype GML datastore.

	areaValue needs uom attribute	This is a defect in the handling of units of measure within the prototype GML datastore.
National Land Survey Sweden	gml_id on feature collection "1" is not accepted	This is a trivial defect in the generation of internal identifiers by the prototype GML datastore.
	beginLifespanVersion text incorrectly formatted as e.g. "20000101" whereas it should be e.g. "2000-01-01T00:00:01"	Additional transformation functionality will need to be defined in the RIF-PRD mapping to apply the correct INSPIRE format to date/time values.
	HydroPhysicalWaters:localType element lacks some contents (origin, persistence, tidal, drainsBasin)	On a visual inspection of the INSPIRE hydrography logical schema, it became apparent that the GML is not defective. It is not known why this caused validation errors, and it would require further investigation.
Statens Kartverk Norway	gml_id on feature collection "1" is not accepted	This is a trivial defect in the generation of internal identifiers by the prototype GML datastore.
	beginLifespanVersion text incorrectly formatted as e.g. "20000101" whereas it should be e.g. "2000-01-01T00:00:01"	Additional transformation functionality will need to be defined in the RIF-PRD mapping to apply the correct INSPIRE format to date/time values.
	HydroPhysicalWaters:condition element missing	Information was not available in the source dataset (this is also a voidable attribute).
OS Ireland	gml_id on feature collection "1" is not accepted	This is a trivial defect in the generation of internal identifiers by the prototype GML datastore.
	gml:LineStringSegment/gml:posList does not validate	It is not known why this error occurred and would require more investigation.
LPS Northern Ireland	gml_id on feature collection "1" is not accepted	This is a trivial defect in the generation of internal identifiers by the prototype GML datastore.
	beginLifespanVersion text incorrectly formatted as e.g. "20000101" whereas it should be e.g. "2000-01-01T00:00:01"	Additional transformation functionality will need to be defined in the RIF-PRD mapping to apply the correct INSPIRE format to date/time values.
	inNetwork element is missing	Information was not available in the source dataset (this is also a voidable attribute).
	gml:LineStringSegment/gml:posList does not validate	It is not known why this error occurred and would require more investigation.

With regard to the feature classes that were provided and transformed, the majority of the INSPIRE mandatory (non voidable) attributes for those classes were found to be available for mapping in the original file. The exceptions to this are noted in Table 14.

To resolve these non-compliances, it is anticipated that data providers will be able to define default values for these attributes or identify other source attributes and/or business logic that can be used to attain this information during transformation.

Table 14 - Missing Mandatory Attributes

Dataset	Missing Non-Voidable Attributes
Dutch Kadaster	*None*
Belgian Cadastre	*None*
National Land Survey Sweden	Watercourse.levelOfDetail
Statens Kartverk Norway	StandingWater.levelOfDetail Watercourse.levelOfDetail Lock.levelOfDetail DamOrWeir.levelOfDetail
OS Ireland	RoadLink.endNode RoadLink.startNode RailwayLink.fictitious RailwayLink.endNode RailwayLink.startNode
LPS Northern Ireland	RoadLink.fictitious RoadLink.endNode RoadLink.startNode RailwayLink.fictitious RailwayLink.endNode RailwayLink.startNode

Of the above mandatory attributes, it is possible to derive the transportation schema `startNode` and `endNode` attributes by conducting a prior enrichment process on the source data. This process could build up the required nodes before inputting the data to the TNS. However, this was not included as part of the prototype.

4.3 Types of Transformation Error

Three types of transformation error (described in Table 15) were relatively common, even in the final version of the prototype, and are discussed below. Note that the design of the TNS allows for individual features to raise object errors without the transformation as a whole being stopped (see Section 3.2.2.1 Table 5 above for a description of this).

Table 15 Types of transformation error

Exception	Reason
Resource Not Found	This was typically caused by the wrong URL being passed to the service (either for the target logical schema or the schema mapping file), which could be resolved by checking the parameters used. Alternatively this may have been caused by the URL being correct, but the hosting location being unavailable (because the required Web or FTP service was not started).
WFS Exception	This was typically caused by invalid feature class names being provided.
Translation Exception	This was normally caused either by a defect in the RIF to Radius Studio translator which could be resolved by further investigation and development of the code that translates from RIF to Radius Studio's own internal rules language, or by an inconsistency in the supplied arguments. The latter could, in turn, be caused by: 1) A RIF sentence using a variable without providing enough information to 'bind' the variable to a specific part of the source or target logical schemas. 2) A RIF variable being bound to a non-existent part of the source or target logical schemas (e.g., being misspelled). In these cases, the exception included the name of the invalid variable or schema component to support diagnosis of the miss-configuration.

5. Conclusions

As part of the State of the Art Analysis conducted earlier in the project [18], a variety of existing tools were assessed with regard to their potential to offer transformation capabilities. A common drawback is that they are not based on standards and are therefore not interoperable, encouraging vendor lock-in.

The TG [1] has been proven to address this problem successfully, through the adoption of standards to manage the interoperable interchange of schema descriptions and model mapping definitions.

The prototype has demonstrated that with vendor and open source community support, as demonstrated by 1Spatial and the HUMBOLDT project, pre-existing tools can be adapted to use standards in order to work together.

We anticipate that other technology providers/developers (such as Safe Software, Snowflake, Interactive Instruments) could similarly adapt their tools to support the standards. Therefore it would be possible to use other technologies, as required, to fulfil the roles of the components identified in this work.

This would create flexibility for service developers and system integrators to use appropriate technology without requiring re-investment in describing schemas and/or model mappings.

5.1 Lessons Learned

During the implementation of the TG [1], the development team encountered some issues that required special effort or changes in the way the prototype was constructed. These issues are mentioned in this section. None of them were felt to invalidate the TG or require its modification (see our assertion in Section 3.5). Nevertheless, the authors feel they will be useful to others seeking to implement a TNS.

5.1.1 XML Repository

To perform a transformation, the TNS needs access to the source and target schemas and model mapping document. These need to be accessible via URLs that will, in turn, resolve to the respective documents. The TG (Section 5.5) recommends use of an XML Repository to supply this need.

However, during prototyping, installation and deployment of the TNS server involved building the service from source and was found to be relatively error prone. Furthermore, the registry's strict security model (based on public key/private key certificates) presented moderate programming challenges. Programmatic management of the repository, although very possible, required access to libraries that were generated during the deployment of the server.

Although a formal XML repository can meet this need, it can also be met by a range of other server technologies (e.g., standard web-servers or File Transfer Protocol (FTP) servers). It is recommend that a production environment should use a formal XML repository (like freebXML) (the rationale for this recommendation is given in the TG [1] Section 5.5). However, during development stages it may be preferable to use a simpler alternative, and this was in fact the route chosen by the prototyping team (see Figure 2 in Section 2.2).

Whilst the TG recommends that the XML Repository is used to track versioning formation, our experience with the prototype suggests this information could also be embedded in the RIF document itself. This means that versioning information would be available to the TNS irrespective of whether the document had been taken from a formal XML repository or a less structured network location.

5.1.2 Translation Between Different Schema Mapping Definition Formats

In Section 3.2.2.2, Figure 15 and Figure 16 show, respectively, the class and sequence diagram for the RIF-PRD Exporter. In addition to the above information, it is important to note that it was found necessary during the prototyping to progress through more stages of translation than initially expected. The two internal meta-formats produced by the AlignmentToModelAlignmentTranslator and ModelAlignmentToModelRIFTranslator classes are respectively close to each of their external formats, but neither is useful outside the translation pipeline. However, these meta-formats enabled the export plugin to span the differences between the two mapping definition formats (OML and RIF-PRD) in a way that was comprehensible for the developers involved. The concepts involved in translation of schema mapping definitions are highly abstract, and the developers found it a challenge to keep a clear focus on the objectives of each piece of code in turn.

5.1.3 Early Adoption of RIF

The prototype developers found it a challenge to build up a good understanding of RIF based on the published information, which comprises the RIF specifications (see [6] for an introduction to them). The specifications are expressed in a logical and mathematical, rather than programming terminology. For those with a general-purpose programming background and only modest pure mathematics experience, it is a cultural leap that needs to be traversed. However, owing to the internal coherence and rigour of the RIF specifications, this was an achievable task for the prototype developers. As RIF becomes a more widely adopted format and “real world” implementations increase, it is to be expected that the format will become more accessible to implementors.

Appendix A Definitions, Acronyms and Abbreviations

Definitions, acronyms, and abbreviations are available from the INSPIRE Glossary [11]. For convenience, we list those mentioned in this document below.

Acronyms, Abbreviations	Definitions
CAD	Computer Aided Design, the use of computer technology for the process of design and documentation of designs.
CRS	Coordinate Reference System, a coordinate-based local, regional or global system used to locate geographical entities.
DWG	AutoCAD's native file format DWG (an abbreviation of "drawing") is a well-known file format for CAD data interoperability, allowing exchange of 2-dimensional and 3-dimensional point, line and polygon geometries.
FTP	File Transfer Protocol, a standard network protocol used to copy a file from one host to another over a TCP/IP based network.
GML	Geography Markup Language, the XML grammar defined by the OGC to express geographical features
HTTP	Hypertext Transfer Protocol, a protocol for distributed, collaborative, hypermedia information systems
INSPIRE	Infrastructure for Spatial Information in the European Community, a European Union wide initiative to harmonise spatial data infrastructures in order to support policy and decision making in Europe.
JRC	Joint Research Centre, the European Union's scientific and technical research laboratory and an integral part of the European Commission.
LMO	Legally Mandated Organisation, an organisation that has or will have a legal mandate for one or more aspects of INSPIRE.
NTF	National Transfer Format was developed by the British Standards Institute for the transfer of spatial information and is the standard transfer format for Ordnance Survey digital data.
OGC	Open Geospatial Consortium, an international, non-profit organization engaged in development of standards for geospatial and location based services.
RIF	Rule Interchange Format, a W3C standard for exchanging rules among rule systems, in particular among Web rule engines.
RIF-PRD	RIF Production Rules Dialect, a standard XML serialization format for production rule languages.
SHAPE	The ESRI Shapefile is a well-known geospatial vector data format for sharing spatial data between geographic information systems.
SOAP	A protocol specification for exchanging structured information in the implementation of Web Services in computer networks (original meaning was Simple Object Access Protocol)
SOSI	Samordnet Opplegg for Stedfestet Informasjon ("Coordinated Approach for Spatial Information"), a popular Norwegian spatial data format.
TCP/IP	Transmission Control Protocol/Internet Protocol, a suite of protocols which are essential to the functioning of the Web; IP handles addressing and routing of messages across networks, and TCP manages the process of reliable exchange of

	data between network hosts.
TNS	Transformation Network Service, a Web service providing query or data instance transformation services.
UML	Unified Modeling Language, a standardised general-purpose modelling language in the field of software engineering.
URL	Uniform Resource Locator, an identifier that specifies where a network resource is available and the mechanism for retrieving it.
UUID	Universally Unique Identifier, an identifier standard used in software construction, that enables distributed systems to uniquely identify information without significant central coordination.
W3C	The World-Wide Web Consortium, an international community of member organisations working to develop Web standards.
WFS	Web Feature Service, a specification providing an interface enabling requests for geographical features across the Web using platform-independent calls.
WFS-T	Web Feature Service (Transactional), a WFS with additional support for transactions, which are atomic, consistent, isolated and durable pieces of information processing activity.
WS-Addressing	Web Services Addressing, a specification providing transport-neutral mechanisms to address Web services and messages.
WSDL	Web Services Description Language, an XML format for describing network services.
XML	eXtensible Markup Language, a set of rules for encoding documents electronically.
XSOM	XML Schema Object Model, a Java library that allows applications to easily parse XML Schema documents and inspect information in them.

Appendix B References

[1] Draft Technical Guidance for INSPIRE Schema Transformation Service, version 2.0, June 2010, http://inspire.jrc.ec.europa.eu/documents/Network_Services/JRC_INSPIRE-TransformService_TG_v2-0.pdf

[2] DIRECTIVE 2007/2/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community (INSPIRE), http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2007:108:0001:0014:EN:PDF.

[3] Regulation on INSPIRE Network Services (Commission Regulation (EC) No 976/2009 of 19 October 2009), http://ec.europa.eu/transparency/regcomitology/index.cfm?do=Search.getPDF&cI7TwVsORn+kLl9oziBPzRrPh2gD8ZmE8tZUqV9OrP7B7EJR+poTzWZ/2wT/z/JFTr7x0HnynbCJdi/BzR4ZvdPpAur0FOHhej8jYcN49FA=.

[4] Draft amendments to Regulation (EC) No 976/2009, http://ec.europa.eu/transparency/regcomitology/index.cfm?do=Search.getPDF&cI7TwVsORn+kLl9oziBPzRrPh2gD8ZmE8tZUqV9OrP7B7EJR+poTzWZ/2wT/z/JFTr7x0HnynbCJdi/BzR4ZvdPpAur0FOHhej8jYcN49FA=.

[5] OpenGIS Geography Markup Language, http://www.opengeospatial.org/standards/gml.

[6] RIF Overview, http://www.w3.org/TR/rif-overview/.

[7] GeoServer, http://geoserver.org/display/GEOS/Welcome

[8] Humboldt Alignment Editor (HALE), http://www.esdi-humboldt.eu/home/open-source.html

[9] Radius Studio, www.1spatial.com/products/radius_studio/index.php

[10] TatukGIS, http://www.tatukgis.com/Editor.aspx

[11] INSPIRE Glossary, http://inspire-registry.jrc.ec.europa.eu/registers/GLOSSARY

[12] INSPIRE Consolidated UML Model (April 2010), available from http://inspire.jrc.ec.europa.eu/index.cfm/pageid/541/downloadid/1707.

[13] JUnit 4 unit testing framework, see http://junit.sourceforge.net/.

[14] PMD Java source code quality improvement tool, http://pmd.sourceforge.net/.

[15] Checkstyle development coding standards tool, see http://checkstyle.sourceforge.net/.

[16] Altova XMLSpy®, see http://www.altova.com/xml-editor/.

[17] Hudson continuous integration platform, see http://hudson-ci.org.

[18] INSPIRE Schema Transformation Network Service “State Of The Art Analysis”.

[19] Thomas Kuehne’s Translator Pattern, see http://c2.com/cgi/wiki?TranslatorPattern.

[20] XML Schema Object Model (XSOM), https://xsom.dev.java.net/userguide.html.

[21] FreebXML, see http://www.freebxml.org/.

[22] Google Web Toolkit, see http://code.google.com/webtoolkit/.

[23] Ontology Mapping Language, see http://www.omwg.org/TR/d7/rdf-xml-syntax/.