# A generic preprocessing service for more usable geographical data processing services

Bénédicte Bucher, Sandrine Balley
Institut Géographique National
Laboratoire COGIT
Saint Mandé, France
{benedicte.bucher, sandrine.balley}@ign.fr

## SUMMARY

This paper aims at enhancing the usability of web services that perform complex processes on geographical data, like data integration or generalisation. These services usually have several requirements about the structure of data they can process. Enhancing the usability of such services for external users who are not familiar with the underlying software and algorithms and who use the service on their own data is a challenging task. It implies giving users enough information for them to correctly build their query message, which often necessitates pre-processing their data. Not all structure requirements are expressed through XML type definition in the service interface. Formalisms exist to enrich this description but they are quite complex both for the service provider to create the service metadata and for the user to understand the metadata. This paper presents our proposal to assist users who want to invoke a data processing service on their own data. One component of our proposal is a 'requirements publishing application' which assists service providers in publishing a generic pre-processing activity that is dedicated to their service. This pre-processing activity will be interpreted afterwards by the other component of our proposal, the generic pre-processing service, to pre-process user data

## INTRODUCTION

Spatial infrastructures have been identified as a crucial basis for monitoring environmental policies and for communicating with citizens about these policies. The INSPIRE directive lay down guidelines to build national spatial infrastructures and a European spatial infrastructure above them. "*Infrastructure for spatial information* means metadata, spatial data sets and spatial data services; network services and technologies; agreements on sharing, access and use; And coordination and monitoring mechanisms, processes and procedures, established, operated or made available in accordance with this Directive" (European Union, 2007) article 3. Several types of services are needed to diffuse meaningful geographic information. "Member States shall establish and operate a network of the following services for the spatial data sets and services for which metadata have been created in accordance with this Directive: [..] (d) transformation services, enabling spatial data sets to be transformed with a view to achieving interoperability; (e) services allowing spatial data services to be invoked. Those services shall take into account relevant user requirements and shall be easy to use, available to the public [..]" (European Union, 2007) article 11. Most work related to the latter category (e) of services aim at designing services that will automatically chain data services, like a feature server service followed by a generalisation service followed by a feature portrayal service. In this paper, we present of work in progress to design a service belonging to the same category but that focuses on enhancing the usability of spatial data services.

More specifically, we address the issue of usability of services that perform complex processes on geographical data. These can have very simple requirements regarding user data like (Harrower, 2006) generalisation service. It is quite simple for anyone to use MapShaper service. But the implementation of complex processes may rely on more complex structures than points and lines, as explained in (Neun, 2006). Thus, services that perform complex process are likely to have strong requirements regarding the structure of input data, which decreases their usability; users must

understand the service input structure requirements and pre-process their data to build an appropriate request to the service.

Both activities (automatic chaining or assisting users in pre-processing) heavily rely on standards and metadata. Some standards are being provided by ISO/OGC. For instance, OGC is designing standard interface specification for services that process geographical data (Web Processing Services (OGC, 2005)). Other standards come from the Web like the Web Service Description Language (WSDL) and the Ontology Web Language for Services (OWL-S). The provider of a service should provide metadata about his service according to standard guidelines, e.g. by writing down a WSDL document or by implementing the OGC *GetCapabilities* operation. These metadata will be read either by another service or by a human user. Our approach consists in (1) providing mechanisms to assist service providers in publishing input structure requirements of their service into formal metadata compliant with these standards, (2) providing mechanisms to assist service users in interpreting these requirements and pre-processing their data.

## SERVICE INPUT STRUCTURE REQUIREMENTS

The following of the paper focuses on vector data and services that process vector data.

### Data structure

As explained in (OGC, 1999), representing the geographic space into digital data takes specific abstraction and encoding steps. Models designed or followed at each abstraction or encoding step by the creator of a geographical data set altogether make the structure of the representation. As illustrated on **Figure 1**, the structure can be seen as the set of answers to the questions: what do I call a road (for instance), which road do I represent in my data, how do I observe and model a road, how do I implement and diffuse road data?

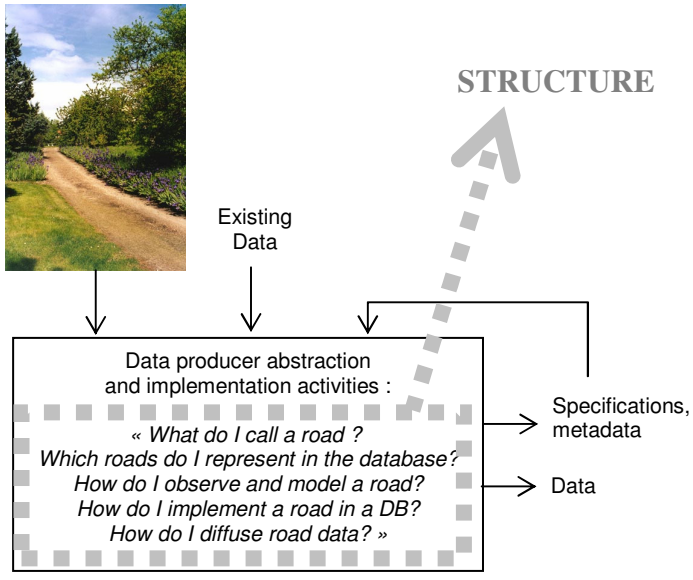


***Figure 1.*** The creation of geographical data implies defining a structure for the representation of the geographical space into data.

Ideally, the structure should be described in metadata accompanying the data for the user to correctly interpret the data. ISO has provided standards to describe all items making up a structure, except for the ontology of the real world. Many standards are needed to describe the structure of a geographical data set. ISO19131 describes product specifications. ISO 19110 describes the conceptual model of a representation: the Feature Catalogue. ISO19115 describes a data set, including its set of Features referring to a Feature Catalogue. ISO 19 109 describes how features relate to spatial objects. ISO19 107 describes how spatial objects are positioned in a reference system. A European profile for ISO 19 115 and ISO 19 131 is currently being defined by experts assigned by the Commission.

Eventually, there should soon be a standard way to build metadata about the structure of a data in the broad sense on the INSPIRE infrastructure. In our approach, we assume that such implemented metadata models are available (INSPIRE implementation rules), and possibly metadata structured after these models.

## Service input structure requirements: what are they?

We define 'service input structure requirements' as the set of requirements related to the structure of a data set that must be fulfilled for a service to actually perform its functionality on the data. Examples of requirements are: 'all inputs should be embedded in a SOAP document of the form SOAP literal', 'the input data should be in gml3', 'the input data should be a feature encoded thanks to the XML schema given in this description', or 'the input data should be a connected network with no cycle'. Two important aspects of a service input structure requirements are:

- requirements are not always explicit,
- not fulfilling the set of requirement of a service does not always lead to a visible error.

These two aspects show the importance of controlling the set of requirements: a user of a service may think the service has performed all right whereas it actually did not because the user data did not meet some hidden requirements.

We try to analyse where these requirements come from.

The processing software must somehow read serialized data before processing them. It is impossible to develop a generic mechanism that will read and interpret any data. Thus, at some point, the developer has to make assumptions about the structure of the representation of the geographic space his program will read. Ideally, these assumptions should be that the representation will come with standard documented metadata and that the software will parse these metadata to correctly read the data.

The processing software must also process the data. This is done by applying operations that manipulate specific types, like points, lines, rings, networks. If the implementation of these types is very specific, like a class *MyNetwork*, the developer may provide factory mechanisms (or import mechanisms) to build them from standard structures. Yet, such factories will still have requirements, like 'input data should be connected arcs with no cycle'.

Requirements may also be induced by the programming activity without being explicitly assumed by the developer, for instance if the developer inverts a variable without testing if it is non null. These constrains may not be detected during tests. The developer will not know that his program yields irrelevant results on data that have at least one null value for an altitude, because he tested them on data that don't have any null value for this attribute. 'Unit testing' is an important activity in software design to trace the compliance of the software interface with explicit specifications. The issue of finding such requirements is not addressed in this work. We assume the developer knows every requirement on the data structure induced by the processing software.

Requirements are also specified by the service deployer when he defines a XML encoding to interact with the program. Most of the time, this deployer is itself a program like the AXIS library.

Last, another type of requirements is usually referred to as the semantic of a process. For example, let us consider a route processing service which input data should be a collection of features of type BasicEdge with a 'weight' attribute of type Integer and that this 'weight' attribute is interpreted as a speed indice to calculate shortest paths. The service will produce wrong results if applied on 'road' features which 'weight' attribute represents a daily number of users.

## Service input structure requirements: where are they described?

Ideally, requirements listed in the previous section should be described in the 'component interface contract'. Today, there is no such document as an exhaustive description of a service input structure requirements. Instead, there are different ways a user may get (complementary) clues about these requirements.

One way is to read the WSDL document associated with the service. WSDL schema is the W3C recommendation to describe Web Services (W3C, 2001). This file describes the functionalities provided by a service as portTypes elements, e.g. <wdl:portType name='GeneraliseARoad'>. Each functionality (or portType) is associated with underlying abstract operations, e.g. <wsdl:operation name='GeneraliseARoadSegment'>. Each abstract operation is associated with a set of XML messages, e.g. <wsdl:message name='SendRoadSegment'>. Each XML message is described thoroughly, often based on type definition embedded within the WSDL <types> tags. To summarize, the WSDL file contain the XML structure to send data to a service. WSDL files are often automatically interpreted by programs to generate 'Stubs'. A stub is a piece of software that can be integrated into a client and that provides the same functionality as the service but through a local interface.

Any requirement can be expressed in the WSDL file, provided that it can be formalised as an XML type definition. The OWL schema can be used to express accurate structure definition like 'non null value for altitude attribute' or 'a network with no cycle'. Yet, there does not exist so far stub generators that can translate complex OWL constraints for example into a java type definition. So the user himself has to parse the document.

If the service happens to belong to an OGC category (WMS, WFS, WCS, etc.), some requirements are described in the corresponding OGC specifications. For instance the MapContext structure is described through an XMLSchema and a text document. In this case also, more specific information are provided by the service response to the getCapabilities request. For instance, the getCapabilities response of a WFS service will list the names of its FeatureTypes.

Another way to get clues about structure requirements of a service is to use it with test data and interpret the result. Since an important aspect of Web Service programming is to define meaningful fault messages, the developer is supposed to throw meaningful error message for each violated requirements he knows, like 'Error : there is a null altitude in your data'. This testing is also a way to find requirements the developer was not aware of. If the user observes inconsistent results he may assume that the test data did not fulfil all hidden structure requirements of the service.

Last, informal debriefing with the service provider can also be fruitful for instance to know on what data the service has been tested or to understand the semantics of some variables.

### Formalising service input structure requirements for service composition

Service input structure requirements are studied in the field of service composition. Service composition relies on two important activities. The first activity is identifying elementary functionality that can be chained to yield a global process the end user is looking for. The second activity is checking if the services providing each of these elementary functionalities can be actually chained. This means that, during the workflow realization, inputs for the next service can be generated from outputs from past services. This has led authors to propose formal languages to accurately describe structure constraint to avoid or mend heterogeneity issues between services for automatic composition. WSDL-S, the semantic extension of WSDL (Akkiraju, 2005), and the OWL-S *grounding* data type, provide a framework to semantically describe a variable and match it with a concrete input/output parameter. Requirements that cannot be expressed in parameters data types are often expressed as formal preconditions or as textual description. Accompanied by domain ontologies of geographic operations and data types, such models greatly promote service discovery and chaining (Lemmens, 2006a) (Lutz, 2005). These descriptions are quite complex and cannot be written by any service provider nor interpreted by any user. This complexity is necessary to automatically handle semantics. For example, in the ontology of geographic operation OPERA of (Lemmens, 2006b) the following description:

$$\begin{aligned} &\text{opera:LocSpat} \subseteq \\ &\text{opera:AcrossAttributeTypes} \cap \\ &(\exists\forall \text{opera:appliesToDataStrucType. (symbol:ObjectFeature} \cup \text{symbol:GridCell})) \cap \\ &(\exists\forall \text{opera:hasInputPar.}(\exists \text{opera:hasParType. Symbol:GF\_LocationAttributeType})) \cap \\ &(\exists\forall \text{opera:hasOutputPar.}(\exists \text{opera:hasParType. Symbol:GF\_SpatialAttributeType})) \cap \\ &(\geq \text{opera:isCoupledToDataset}) \end{aligned}$$

refers to an operation type that reads a location attribute type and produces a spatial attribute type (after (Lemmens, 2006b) p129).

## OUR PROPOSAL

Our objective is to improve the usability of services that process geographical data. Our approach, already expressed in the introduction, is to (1) provide mechanisms to assist service providers in publishing input structure requirements of their service, (2) provide mechanisms to assist service users in interpreting these requirements and pre-processing their data. We do not handle requirements that are related to the XML encoding. There are more and more agents that automatically make and publish these necessary constraints on the server side as well as client agents that interpret these descriptions and deal with them (the Stubs).

## Main lines

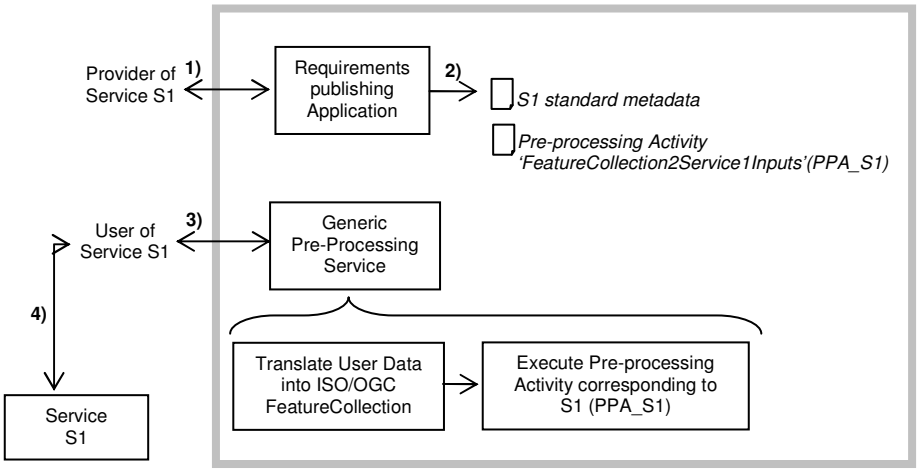


***Figure 2***. Overview of our pre-processing architecture. The grey line is the boundary of our system.

**Figure 2** gives an overview of our proposed architecture.

1) The provider of a Web Service, say S1, uses the requirements publishing application to express requirements regarding his service input structure.

2) This application records the requirements in two forms : standard metadata (wrt WSDL-S or OWL-S) and an Activity object that describes how to get document the service input variables from a generic structure, the ISO/OGC FeatureCollection. So far we have concentrated on generating this activity and not on generating standard metadata.

3) The user of Service S1 is somehow redirected to the generic pre-processing service. He loads his data on the server and he specifies which service he wants to pre-process his data for.

    a. A component translates these data into a pivot format. This pivot format is the implementation of the FeatureCollection (and related ISO/OGC concepts) in the GeOxygene platform[1]. Geoxygene is an OpenSource java platform developed and used within the COGIT laboratory to implement ISO/OGC concepts. It will implement the GeoAPI in future versions.

    b. Another component retrieves the Pre-processing activity corresponding to service S1 and executes it interactively with the user.

    c. The service final response to the user is the pre-processed data.

4) The user invokes the service S1 on his pre-processed data.

The following of the paper presents important components of this architecture that have already been implemented.

[1] http://oxygene-project.sourceforge.net/

## A model to describe activities

The first component is a model to represent pre-processing activities. There are several models to formalise processes in the literature, and within our laboratory, which meet different objectives. These objectives can be for example designing a system, implementing a simulation application, implementing a multi-agent system or implementing a task-planning application. In the context of this work, we want to plan a generic activity, store it, specify it and execute it. We use a model that has been designed in a previous work by the authors on a generic purpose: to build metadata about usage patterns of geographical data and software tools (Bucher et al. 05). This model is greatly inspired from UML2 activity diagram stereotypes but is simpler. For example, unlike UML2, Activity is used to denote even a simple action - that is an executable Activity - or a control –like a choice-. This model is connected to the metadata model proposed in (Abd-el-Kader and Bucher 06) to describe functions that are provided by implemented operations. Figure 3 summarizes the main elements of our model.

- The key element is the function, like 'Aggregate classes' or 'Delete attribute'. A function can be realised by an Activity or by an implemented operation. A Function has a textual description, variables, preconditions, post-conditions, effects. There are generalisation-specialisation relationships between Functions.
- An Activity is any manipulation, like 'designing a map', 'selecting a feature', 'invoking the method M', 'writing a code line s'. It can involve several performers, a human, a group of humans or a piece of software. There are generalisation-specialisation relationships between Activities. An Activity may have an explicit decomposition: a set of nodes (sub-activities) and a set of edges.
- An implemented operation is a piece of software that can be interpreted during a software process to yield a function. An implemented operation is not always exchangeable. It is often encapsulated in an exchangeable piece of software like a java library or a plugin.

There is a distinction between the function of an implemented operation, say the function of myMethod, and the function of the activity 'invocating myMethod'. The latter is more generic. It includes the choice of an interface to interact with the operation (in case the operation has several interfaces), it also includes the creation of necessary items for the operation to run. The function of the Activity 'Execute operation O' is thus more relevant to the user than the function of the operation O.

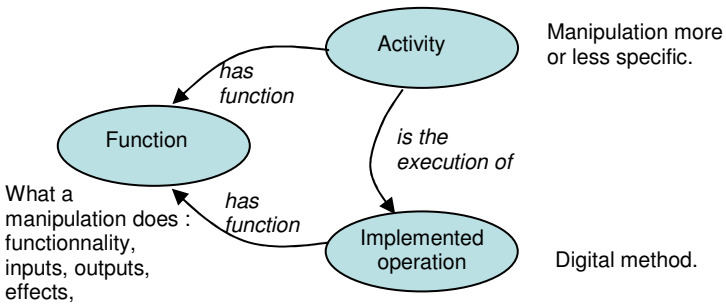


***Figure 3.*** Main elements of our formalism to represent manipulations.

## Restructuration operations

The second component is a set of implemented operations and of activities that will be building blocks for pre-processing activities. It has been developed in the context of a PhD work in the laboratory (Balley et al., 2006). The objective of this PhD work was to assist a user in adapting data diffused by a data provider to his platform and preferences. The application assists the user in performing two kinds of transformation operations: classical remodelling operations (rename a class, merge classes, aggregate features, etc) and derivations of implicit knowledge based on spatial analysis tools (derive road features from road segments, derive town features from buildings, etc.). Any such operation is described as a complex activity that has several sub activities:

- the transformation of the conceptual schema if any, like aggregating classes within the conceptual schema,
- the transformation of the logical schema if any, like creating a new java class (object logical schema) and a new table (relational logical schema),
- the transformation of the data.

This model ensures the consistent transformation of data and metadata. This is illustrated by a scenario on Figure 4. In this scenario, the user has chosen to transform his data based on the conceptual schema. He manipulates the conceptual schema and the system consequently transforms the data, the logical schema and the conceptual schema.

- The application proposes a set of operations that can be applied to the current data set conceptual schema.
- The user selects an operation and parameterises it. For instance he deletes an attribute on the conceptual schema.
- The application reacts by creating a corresponding activity that describes the deletion of the attribute on the conceptual schema and on the logical schema and on the data themselves. This activity can be executed, i.e. its three sub-activities can be executed. The application launches the execution of the sub-activity that modifies the conceptual schema. The application adds the complex activity just created to a global activity that will describe the whole transformation of the data provider data set.
- The application proposes a set of operations that can be applied to the current refreshed data set conceptual schema, and so on.
- This sequence of activities can be automatically executed when the user presses a ‘commit’ button.

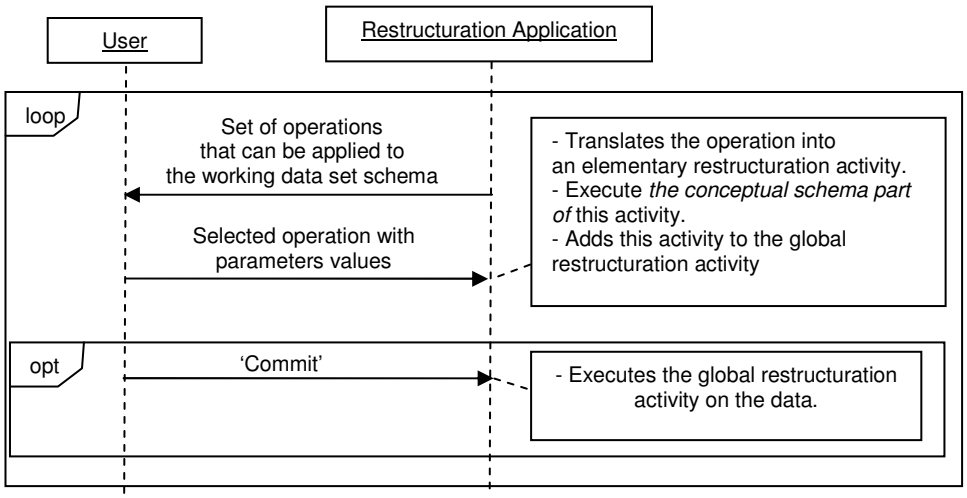


***Figure 4.*** Sequences of (Balley et al 06) restructuration application running.

## An activity browser

The last component that has been implemented so far is a library of graphical widgets to browse, specify and execute activities through a graphical interface. A component displays the list of available functions associated to activities. For example it displays the label 'Rename an attribute' that corresponds to the function called RenameAnAttribute. This function is associated to the Activity 'invokeImplementedMethodxxx'. When the user selects this label, the application builds a panel that contains the items to be specified, i.e. the variables. To specify the value of a variable, the user can create an object, or make a reference to the value of another variable, or make a reference to an object displayed on the interface. The library uses the java reflect mechanisms to translate user actions on a graphical interface into method calls and objects instantiations. Besides, if the domain of value of the variable is an enumerated set, a combo box displays the set of values.

## The pre-processing service concept

To illustrate how these components will be put together in our current work, let us go back to the example of the route processing service mentioned in the first part of this paper. The provider of this route processing service wishes to publish the requirements of his service. We detail just a few steps of how he does.

1. The service provider logs on the 'Requirements publishing application' and specifies his service URI.

2. The application displays a generic FeatureCollection schema with a set of restructuration operations that can be applied to it.

3. The service provider selects the operation 'Select an attribute'. The application will ask him to possibly specify selection filters. A filter may be a free text or a formal constraint. The service provider creates a first criterion that is a formal constraint: 'domainOfValue, equals,

integerType'. The service provider creates a second criterion that is free text: 'this attribute is minimised during the shortest path algorithm'.

4. The application displays again a set of restructuration operations. The service provider selects the operation 'Rename attribute'. The application will ask him to specify two items. First item is the attribute, the service provider creates a reference to the attribute selected during the preceding step. Second item is the name. The service provider will write down 'weight'.

5. and so on until the service provider has fully specified the structure of data his service can process.

Next time a user wants to use the route processing service, the service redirects him to the generic pre-processing service.

6. The user sends his data to the pre-processing service as well as the URI of the service he is interested in, i.e. the route processing service.

7. The Pre-processing service retrieves, specifies and executes the corresponding pre-processing activity step by step on the user data.

   a. It executes the attribute selection: it parses the criteria of the 'choice' activity. It automatically interprets formal constraints like 'domaineOfValue, equals, integerType' and restricts the candidate attributes. Since there is one criterion left, it opens a dialog that displays the free text criterion 'this attribute is minimised during the shortest path algorithm' as well as graphical items to select the attribute among the candidates highlighted.

   b. It executes the renaming operation on this attribute.

## CONCLUSION

The problem tackled here is the necessity for a user to pre-process his data before using a complex processing service. Our proposal consists in representing the pre-processing activity to go from a pivot standard structure to the service required structure and in executing this activity on user data. Firstly we assist the service provider in describing a pre-processing activity going from a FeatureCollection to the required structure for his service. Secondly we assist the service user in executing this pre-processing activity on his data.

The originality of this approach lies in several aspects. One is the integrated representation of the notion of data structure that comprises and relates data, physical schemas, logical schemas and conceptual schemas. Another aspect is the formalization of the pre-processing activity. This model distinguishes between pre-processing knowledge that must be interpreted by the user (like the free text criteria in our example), and pre-processing knowledge that must be interpreted by the application.

On-going work is the implementation of a final prototype bringing together existing prototypes.

Remaining issues are numerous. First one is the writing of OWL-S metadata to translate the structure requirements as illustrated on **Figure 2**. Another one is the automatic or semi-

automatic generation of metadata elements that will possibly be needed by restructuration operations.

**BIBLIOGRAPHY**

Abd-el-Kader, Y., Bucher, B., 2006 Cataloguing GI Functions provided by Non Web Services Software Resources Within IGN, in proceedings of the 9th AGILE conference, Visegrad, Hungary

Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M-Th., Sheth, A., Verma, K., 2005 Web Service Semantics - WSDL-S, W3C Member Submission

Balley, S., Bucher, B., Libourel, S, 2006 A service to customize the structure of a geographic dataset, in proceedings of the Second International Workshop on Semantic-based Geographical Information Systems (SeBGIS'06), Montpellier, pp1703-1711

Bucher, B., Balley, S., Richard, D., Cébelieu, G., Hangouët, J.F., 2005 Shareable descriptions of data production processes, in proceedings of the 8th AGILE conference, Estoril, pp.13-22

European Union, 2007 Directive of the European Parliament and of the Council establishing an Infrastructure for Spatial Information in the European Community (INSPIRE), Brussels

Harrower, M., Bloch, M., 2006 MapShaper.org: A Map Generalization Web Service, in IEEE Computer Graphics and Applications, vol 26(4), pp.22-27

Lemmens, R., 2006 a, Semantic and syntactic service descriptions at work in geo-service chaining. in proceedings of the 9th AGILE conference, Visegrad, Hungary, pp.51-61.

Lemmens, R., 2006 b, Semantic interoperability in distributed geo-service, PhD thesis, ITC, Enschede

Lutz, M., 2005 Ontology-Based Service Discovery in Spatial Data Infrastructures, Workshop on Geographic Information Retrieval (GIR 2005), Bremen, Germany.

Neun, M., Burghardt, D., Weibel, R., 2006 Spatial structures as generalisation support services, in proceedings of the Joint ISPRS Workshop on Multiple Representation and Interoperability of Spatial Data, Hannover

OpenGIS Consortium, 1999 The OpenGIS™ Abstract Specification, Topic 5 : Features, version 4, Kottman (Editor), OGC 99-105r2

Open Geospatial Consortium, 2005 OpenGIS® Web Processing Service, Discussion Paper, OGC 056007r4, 2005

W3C, 2001 Web Services Description Language (WSDL), W3C Note