OGC Schemas Browser: Visualizing OWS' XML Schemas

Alain Tamayo, Carlos Granell, Joaquín Huerta
Institute of New Imaging Technologies, Universitat Jaume I, Spain

INTRODUCTION

OGC Web Services (OWS) are a set of implementation specifications to exchange geospatial information using a Service Oriented Architecture (SOA). These specifications use XML Schema (W3C, 2004) as the language to express the structure of the exchanged messages. The size of the schemas associated to the specifications has been growing with time reaching a point that they are very hard to understand and use to build real systems. Source code to manipulate XML instances based on these schemas sometimes is generated using XML Data binding generators such as XMLBeans[1], JAXB-RI[2] or JiBX[3].

The generated code frequently presents a poor performance due mainly to the large number of classes in the final code. Understanding why the number of generated classes is so high or trying to find a way to optimize this code is a task that is very hard to accomplish using the features included in modern XML editors or other XML-processing related tools. Specifically, these tools do not provide features to analyze types independently of the rest of XML Schema main components (element, attributes, groups and attribute groups). We consider the concept of *type* as the main concept in the translation of schemas to source code in a target programming language.

In this paper we present a tool that provides users with a different way of analyzing the schemas of OWS specifications. The tool is focused on the visualization of dependency relationships at different levels. At the time of this writing, the visualization of specification dependencies, namespace dependencies, file dependencies, and type dependencies are supported. The tool also checks at loading time for errors that might go undetected when using XML editors that not perform validation in a thorough way.

The remainder of this paper is structured as follows. Next section presents some of the OWS implementations. After this, a brief introduction to the XML Schema recommendation and to widely know XML editors is presented. Next, we discuss the main features offered by OGC Schemas Browser and present examples of how to use them. Finally, we present conclusions and future work.

OGC WEB SERVICES

OGC Web Services provide standardized distributed access to geospatial information. Some of the main specifications are listed next:

- *Web Map Service (WMS)*: It produces maps of spatially referenced data dynamically from geographic information. (OGC, 2006).

[1] http://xmlbeans.apache.org/
[2] https://jaxb.dev.java.net/
[3] http://jibx.sourceforge.net/

- *Web Feature Service (WFS)*: It allows a client to retrieve and update geospatial data encoded in Geography Markup Language (GML) from multiple services (OGC, 2005).
- *Web Coverage Service (WCS)*: It provides access to potentially detailed and rich sets of geospatial information, in forms that are useful for client-side rendering, multi-valued coverages, and input into scientific models and other clients (OGC, 2008).
- *Sensor Observation Service (SOS)*: SOS provides an API for managing deployed sensors and retrieving sensor data and specifically "observation" data (OGC, 2007).
- *Web Processing Service (WPS):* It defines a standardized interface that facilitates the publishing of geospatial processes and the discovery of and binding to those processes by clients (OGC, 2007a).
- *Sensor Planning Service (SPS)*: It provides a standard interface to collection assets (i.e., sensors, and other information gathering assets) and to the support systems that surround them (OGC, 2007b).

The structure of the messages used for every specification is defined using XML Schema. The level of complexity of the specifications varies from relatively simple, self-contained specifications, such as WMS; to more complex ones such as SOS that presents several dependencies from other specifications. Some of these specifications will be used as examples in the following sections.

XML SCHEMA AND EDITORS

XML Schema is W3C recommendation to define the structure of XML instance documents. An XML Schema document mainly contains complex and simple type definitions, element declarations, attribute declarations, group definitions and attribute definitions. This language is recognized to be very complex (Martens, 2006)(Bex, 2009) and the use of modern editors can greatly simplify the work of schema designers and developers. Two widely known XML editors are *XMLSpy*[4] and *<oXygen/>*[5]. They provide advanced capabilities for editing and validation of XML Schema files and XML instance documents based on these schemas.

Although these XML editors offer a large set of functionality they do not include features that could help understanding the relationships between all the types included in a set of schema files. XML Schema provides a sophisticated type system allowing the definition of user-defined types, subtyping relationships, and features such as substitution groups that resemble the concept of *polymorphism* in object-oriented programming languages. Understanding the type structure of a schema files set is very important if we want to produce efficient source code to process XML instance files, whether we write the code manually or we use a code generator. The possibility of analyzing type relationships in a separate way from the rest of other XML Schema components is not included in any tool known by the authors.

OGC SCHEMAS BROWSER

OGC Schemas Browser (or just the *browser* from now on) is an open source tool that allows OWS specifications to be browsed in an intuitive way that helps understanding the relationships between files, specifications and types. Some of the features offered by this tool are visualizing specification and namespace dependencies, visualizing file dependencies, visualizing type dependencies, and detecting XML Schema design errors. The application has two main components: the *schema processor* and the *graph renderer*. The core is the *schema processor,* which is in charge of processing the schema files and all their dependencies, building an internal representation that will be used later to generate graphs of elements relationships. The *graph renderer* uses the component

[4] http://www.altova.com/xml-editor
[5] http://www.oxygenxml.com/

JGraph[6] to display graphs of entity relationships. The renderer contains a set of algorithms to generate the graph structure requested by the user that is visualized in JGraph.

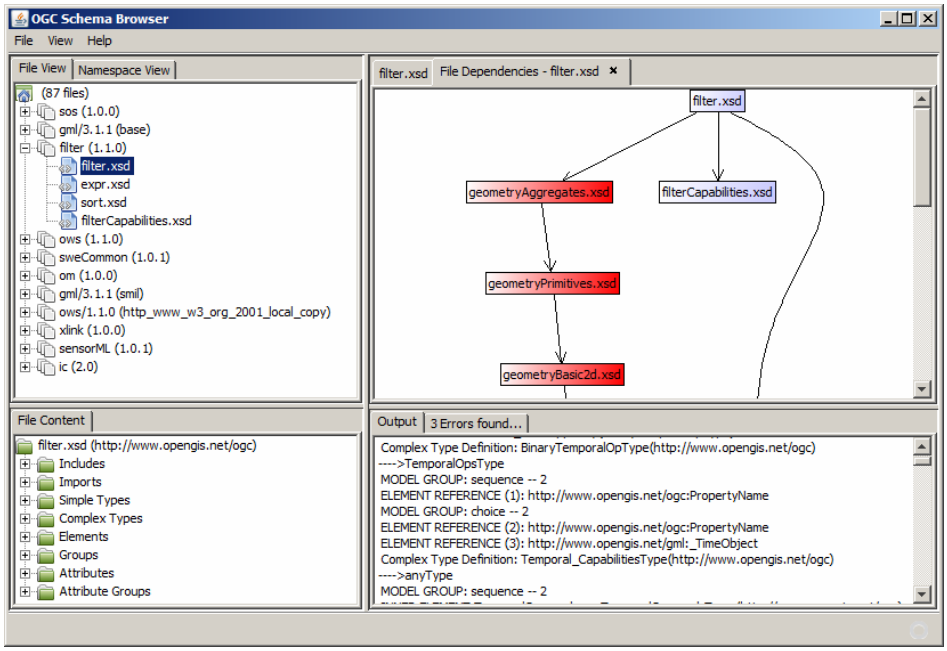

Figure 1: *OGC Schemas Browser user interface.*

Before analyzing the features provided by the browser, let us introduce the terms of *main specifications* and *external specifications*. The application is designed to analyze one OWS specification at a time, this specification is referred to as the *main specification*. OWS specifications are frequently built reusing schemas or concepts included in other specification, which are therefore considered as *external specifications*. In a similar way, the concepts of *main specification schemas* and *external schemas* are defined.

The user interface of the browser can be seen in Figure 1. It is composed of four main sections:

- *Navigation panels*: It includes the File and Namespace views, where files included in the main and external specifications are presented, grouped either by folder or by namespace.
- *File Content View*: This view shows the content of the file selected on the navigation panel. It lists the main XML schema components included in this file.
- *Text and Graphs View*: It allows displaying all the supported relationship graphs mentioned before, as well as the text of the file selected on the navigator panel.
- *Information Tabs*: These tabs show useful information to the user, such as the output generated by the schema processor and the errors detected while processing a given specification.

To process a specification, its containing folder must be opened from the File/Open Folder menu option. This folder is then processed by the schema processor, detecting automatically all the

[6] http://www.jgraph.com/

dependencies from external folders and files. Once this is done, errors detected while processing are displayed in the Errors tab and graphs showing different relationships are displayed on user request.

Visualizing specifications and namespaces dependencies

Figure 2 shows the folder and namespace dependencies for the SOS specification. The folder dependencies graph is an approximation of a specification dependencies graph. Folders are a more low-level concept than specifications, but in the case of OWS specifications where schema files are grouped by folder, the resulting graphs are very similar. For example, in the graph to the left in Figure 2, we find two nodes corresponding to the GML specification; nevertheless the specification dependencies can be easily appreciated. The graph to the right shows the namespace dependencies which in this case looks very similar to the folder dependencies.

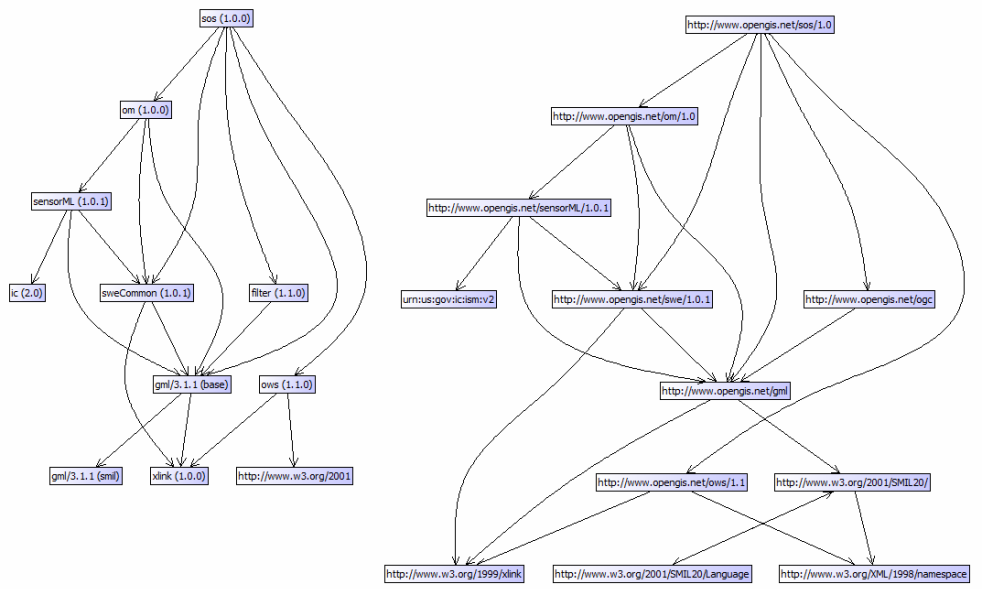

Figure 2: *Specification (left) and namespace dependencies (right) in SOS.*

Visualizing File Dependencies

Figure 3 shows file dependencies for the WFS specification. The graph shows the 46 schema files WFS depends on directly or indirectly. This graph can give us a first impression of how complex or large is a given specification. File dependencies graphs can be generated by the browser starting from any folder, namespaces or schema file. Although this is a useful feature to browse and understand the relationships between schema files, it is not usually offered by open source or commercial XML Editors. The exception to this is *SchemaAgent*[7], an application that can be used in conjunction with *XMLSpy* to manipulate group of schema files. Even if both products are developed by the same company they are sold separately.

[7] http://link.altova.com/schemaagent.html

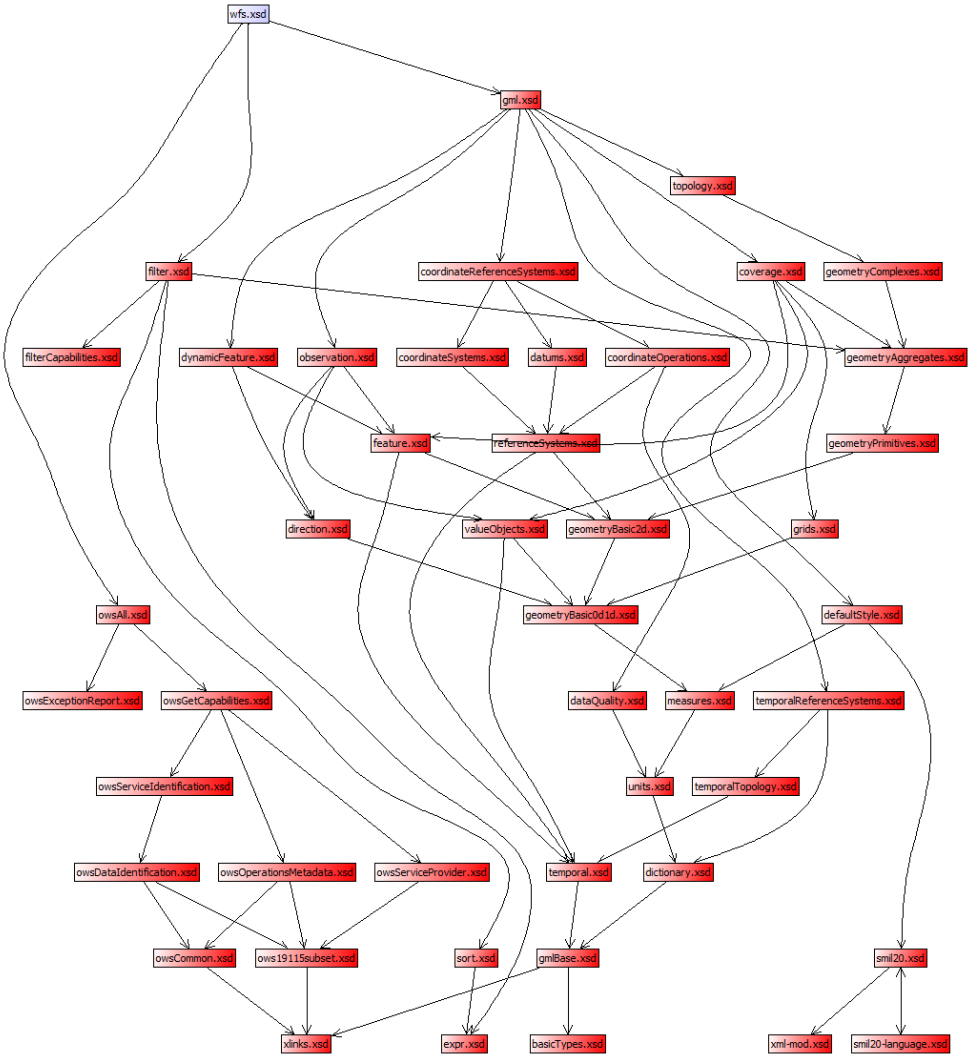

Figure 3: *File dependencies in WFS.*

Visualizing Type Dependencies

Understanding relationships between types modeled in XML schemas is very important if we are writing XML data binding code. The richness of the XML Schema type system requires features that allow the visualization of the relationships between all the types included in a set of related schema files. Support for these features is not included in any XML editor or XML-related application known by the authors. In OGC Schemas Browser graphs regarding type dependencies come in two flavors: Type definition hierarchies and types dependencies graphs.

A *type definition hierarchy* is defined in (W3C, 2004) as a tree including all of the derivation relationships between types. Type definition hierarchies can be constructed in the browser for all the types contained in a folder, namespace or single file. Figure 4 shows the type definition hierarchy for the types contained in the SOS specification. Blue nodes are types defined within SOS, red nodes are external types used as base for types inside SOS, and yellow nodes are XML Schema primitive types.

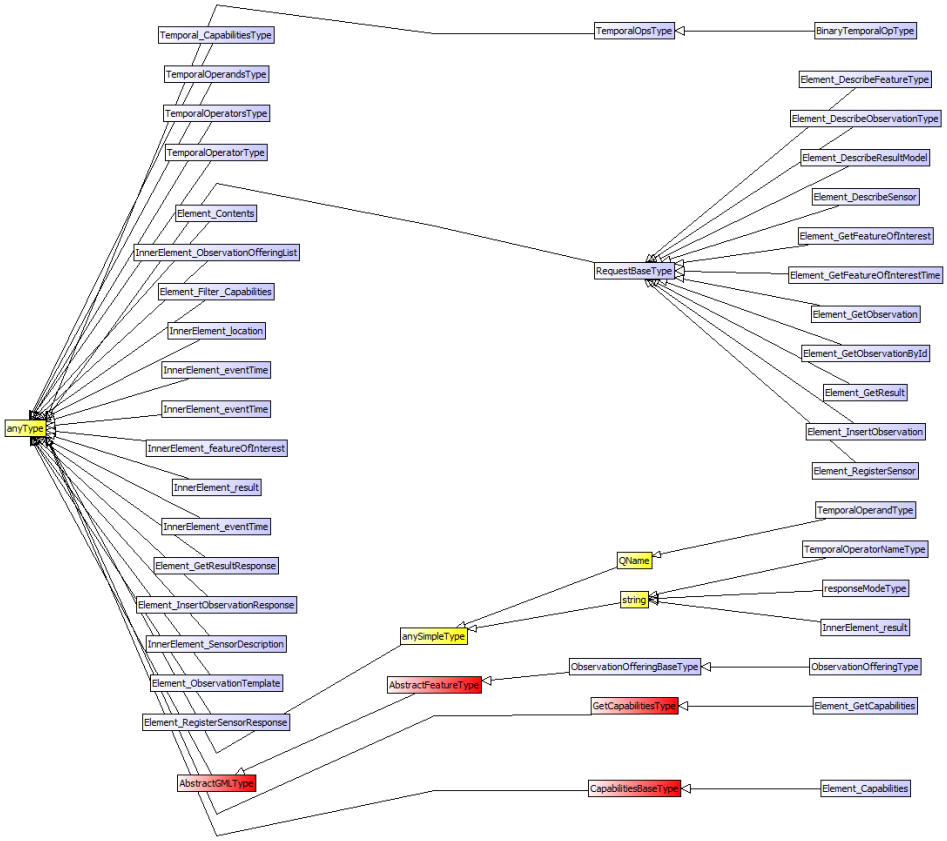

Figure 4: *SOS Type Definition Hierarchy.*

Type dependencies graphs show any possible dependency relationship between types in a given specification (Figure 5a). We say that type A depends on type B if:

- A is derived from B, (by extension or restriction)
- A is a simple type and:
  - It defines an union type which includes B as one of its members
  - It defines a list type with B as itemType.
- A is a complex type and:
  - Contains an attribute declaration of type B
  - Contains a reference to a global attribute of type B
  - Contains an element declaration of type B
  - Contains a reference to a global element of type B

- Contains a reference to a attribute group which contains an attribute declaration of type B, or contains a reference to a global attribute of type B.
- Contains a reference to a group which contains an element declaration of type B, or contains a reference to a global element of type B.

Due to the large number of types contained in some specifications, sometimes type dependencies graphs become too big to be useful. To overcome this limitation, our application allows the generation of these graphs at different levels. The first level is the folder or namespace level, where the relationships between all of the types contained inside o folder or namespace is generated. The second level is the file level where the types included in a file are used as starting point to build the dependencies graph. Finally, at the type level, the graph can be generated starting from a single type, showing all of the others types it depends on. Just in case the generated graphs are still too complex and extra feature to display a graph with only the immediate dependencies of a single type is provided (Figure 5b).

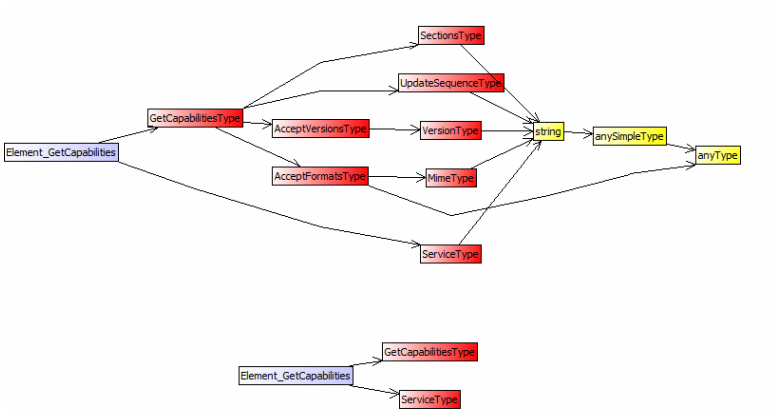

Figure 5:*Type dependencies for a single type: a) all dependencies(top) and b) only immediate dependencies(bottom)*

Detecting Errors

The last important feature included in OGC Schemas Browser is *error detection*. This tool does not perform full validation of the schemas, but it can detect some kind of errors such as duplicated component names or reference to elements that are not resolved correctly. This feature allows user to detect bugs and design error that have crawled in the schemas. We illustrate next with some examples how these errors are detected.

Let us consider the first kind of errors: *duplicated names*. XML Schema defines the concept of *namespaces*, which allows the global space of names used into the schemas to be partitioned in named, separated spaces. This way we can use the same name in different namespaces without creating a conflict. In addition, XML Schema considers that names of different types of components belong to different spaces. This means that in a given namespace we can have, for example, a type and an element with exactly the same type. Avoiding duplicates names is not always an easy thing to do, especially if we are working with a large number of schema files. This is exactly the case of some OWS specifications such as SOS 1.0.0 and WCS 1.1.2[8].Loading SOS 1.0.0 in the browser produces

[8] The set of schemas used here are published by OGC at the following address: http://schemas.opengis.net/SCHEMAS_OPENGIS_NET.zip

output shown in Listing 1. As the output suggest the specification contains 3 duplicated names. The source of the error can be easily traced to the files *xml-mod.xsd* included in GML 3.1.1 and *xml.xsd* imported by OWS Common 1.1.0. When used separately there is no conflict between these specifications but when they are combined, which is the case for SOS 1.0.0, the problem arise when the duplicate declaration of attributes *http://www.w3.org/-XML/1998/namespace:lang* and *http://www.w3.org/XML/1998/namespace:space*, and the attribute group *http://www.w3.org/XML/-1998/namespace:specialAttrs* are detected.

```
ERRORS: 3
1 - Error in (http://www.w3.org/2001/xml.xsd): Element
     http://www.w3.org/XML/1998/namespace:lang already defined in
     SCHEMAS_OPENGIS_NET/gml/3.1.1/smil/xml-mod.xsd
2 - Error in (http://www.w3.org/2001/xml.xsd): Element
     http://www.w3.org/XML/1998/namespace:space already defined in
     SCHEMAS_OPENGIS_NET/gml/3.1.1/smil/xml-mod.xsd
3 - Error in (http://www.w3.org/2001/xml.xsd): Element
     http://www.w3.org/XML/1998/namespace:specialAttrs already defined in
     SCHEMAS_OPENGIS_NET/gml/3.1.1/smil/xml-mod.xsd
```

Listing 1:*Output errors after loading SOS 1.0.0*

Similar errors arise when WCS 1.1.2 is loaded but in this case in a much larger number. A short fragment of the errors is shown in Listing 2:

```
ERRORS: 187
1-  Error in (../gml/3.1.1/base/ coordinateReferenceSystems.xsd): Element
    http://www.opengis.net/gml:_CoordinateReferenceSystem already defined
    in ../wcs/1.1.2/GMLprofileForWCS/coordinateReferenceSystems.xsd
2 - Error in (../gml/3.1.1/base/ coordinateReferenceSystems.xsd): Element
     http://www.opengis.net/ gml:usesCartesianCS already defined in
     ../wcs/1.1.2/ GMLprofileForWCS/coordinateReferenceSystems.xsd
3 - Error in (../gml/3.1.1/base/coordinateReferenceSystems.xsd): Element
     http://www.opengis.net/gml:ImageCRS already defined in
     ../wcs/1.1.2/GMLprofileForWCS/coordinateReferenceSystems.xsd
      ...
```

Listing 2:*Output errors after loading WCS 1.1.2*

In a similar way we can trace the errors in Listing 2 by looking to the files or folder dependencies. Figure 6 shows a fragment of the file dependencies graph for WCS 1.1.2. In this figure we can appreciate why we found so many duplicate names: a set of schema files is included twice in the specification. This happens because the schema designers created a GML 3.1.1 profile for WCS. WCS schemas were supposed to reference this profile instead of the full GML specification. Nevertheless, some references to the full specification were kept, apparently by accident. As a result 14 duplicated files are included in the final schema set.

The second kind of errors, unresolved references to elements, was detected while loading GML 3.2.1 (Listing 3). In this case 41 element references could not be resolved to its declaration. To find where the missing declarations are located, in case they exist, we can use the File/Find menu item, which allows users to search for any XML schema component declaration. Most of the not properly referenced element declarations for some GML schema files were all located in *deprecatedTypes.xsd*.

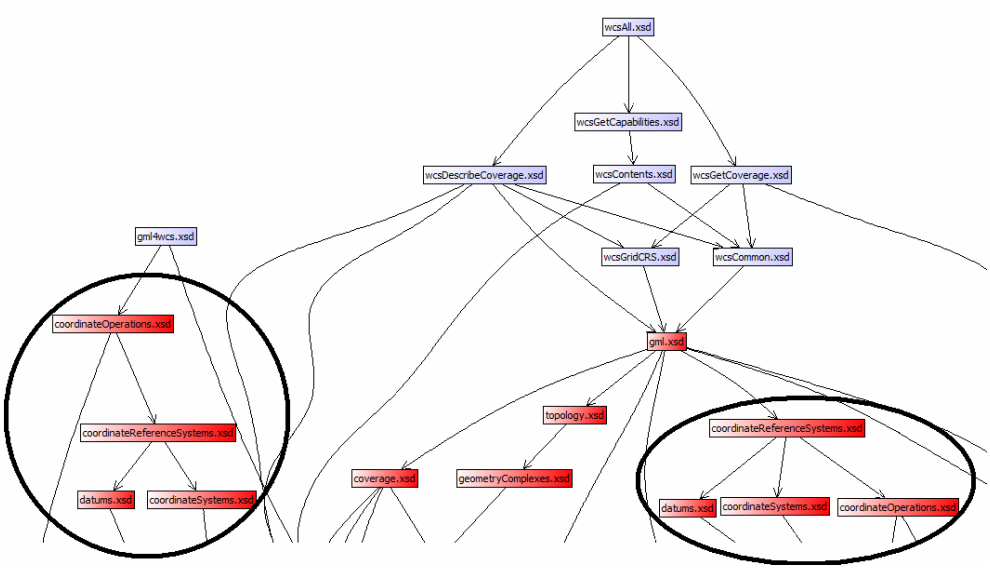

Figure 6: *Fragment of the file dependencies graph for WCS 1.1.2*

```
ERRORS: 41
Error in SCHEMAS_OPENGIS_NET/gml/3.2.1/coverage.xsd: Definition for
    http://www.opengis.net/gml/3.2:MappingRule not found
Error in SCHEMAS_OPENGIS_NET/gml/3.2.1/coverage.xsd: Definition for
    http://www.opengis.net/gml/3.2:IncrementOrder not found
Error in SCHEMAS_OPENGIS_NET/gml/3.2.1/ dictionary.xsd: Definition for
    http://www.opengis.net/gml/3.2:metaDataProperty not found
...
```

Listing 3: *Output errors after loading GML3.2.1*

CONCLUSION

Modern XML editors or other XML-processing related tools do not provide features to analyze types independently of the rest of XML Schema main components. This is an important feature when we try to understand and optimize XML data binding code. In this paper we presented an open source tool that allows users to visualize the relationships between types defined in the schema files associated to the OWS specifications. The relationships are visualized through type definition hierarchies and type dependencies graphs. The first ones are trees including all of the derivation relationships between types, and the second ones, are graphs showing any possible dependency relationship between types in a given specification. In addition, other information can be visualized such as specification dependencies, namespace dependencies and file dependencies. The tool also detects some errors that are found in some of the specifications such as duplicated files and unresolved references.

This tool is not meant to replace existing XML editors but to complement them to help understanding complex schemas such as those included with the OWS specifications. We also want to highlight that we do not claim support for the whole XML Schema recommendation which is large

and complex. We have been adding support for the schema components that are presented in the specifications used to test the product.

BIBLIOGRAPHY

Bex, J. B.,Gelade W., Martens W., Neven F., Simplifying XML Schema: Effortless Handling of Nondeterministic Regular Expressions. In SIGMOD'09, Providence, Rhode Island, USA. June 29–July 2, 2009.

Martens W., Neven F., Schwentick, Bex, J., Expressiveness and complexity of XML Schema. ACM Transactions on Database Systems (TODS) Volume 31 , Issue 3 : 770 - 813 , 2006

OGC, 2005 OpenGIS® Web Feature Service Implementation Specification. Version. 1.1.0. OGC Document Number 04-094

OGC, 2006 OpenGIS® Web Map Server Implementation Specification. Version. 1.3.0. OGC Document Number 06-042

OGC, 2007 Sensor Observation Service. 1.0.0. OGC Document Number 06-009r6

OGC, 2007a OpenGIS® Web Processing Service. OGC Document Number 05-007r7

OGC, 2007b OpenGIS® Sensor Planning Service Implementation Specification. OGC Document Number 07-014r3

OGC, 2008 OpenGIS® Web Coverage Service Implementation Specification. Version 1.1.2. OGC Document Number 07-067r5

W3C, 2004 XML Schema Part 1: Structures Second Edition, http://www.w3.org/TR/xmlschema-1. Last accessed 21-01-2010