Analysing Complexity of XML Schemas in Geospatial Web Services

Alain Tamayo
Institute of New Imaging Technologies
Universitat Jaume I, Spain
Ave Vicent Sos Baynat, SN,
12071, Castellón de la Plana
atamayo@uji.es

Carlos Granell
Institute of New Imaging Technologies
Universitat Jaume I, Spain
Ave Vicent Sos Baynat, SN,
12071, Castellón de la Plana
carlos.granell@uji.es

Joaquín Huerta
Institute of New Imaging Technologies
Universitat Jaume I, Spain
Ave Vicent Sos Baynat, SN,
12071, Castellón de la Plana
huerta@uji.es

ABSTRACT

XML Schema is the language used to define the structure of messages exchanged between OGC-based web service clients and providers. The size of these schemas has been growing with time, reaching a state that makes its understanding and effective application a hard task. A first step to cope with this situation is to provide different ways to measure the complexity of the schemas. In this regard, we present in this paper an analysis of the complexity of XML schemas in OGC web services. We use a group of metrics found in the literature and introduce new metrics to measure size and/or complexity of these schemas. The use of adequate metrics allows us to quantify the complexity, quality and other properties of the schemas, which can be very useful in different scenarios.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*complexity measures, process metrics, product metrics*

General Terms

Measurement, Standardization

Keywords

XML Schema, Web Services, Geospatial Information, Complexity Analysis, Software Metrics

1. INTRODUCTION

Service-Oriented Architecture (SOA) is widely used in the Geographic Information Systems (GIS) field to access geospatial data. This architecture presents an approach for building distributed systems that deliver application functionality as services to either end-user applications or other services. One of the main requirements when building systems based on SOA is *interoperability*, which ensures that information can be exchanged in a way that can be understood by service providers and consumers.

Interoperability in the GIS field is achieved by using standards [6]. Some widely known standards are: Web Map Service Implementation Specification (WMS) [16], Web Coverage Service Interface Implementation Specification (WCS) [24] and Web Feature Service Implementation Specification (WFS) [25], among others. These specifications are known as a whole as OGC[1] Web Services (OWS), and they allow GIS clients to access geospatial data without knowing details about how this data is gathered or stored. These specifications define the interface of the operations that must be supported by a web service provider and the structure of messages exchanged between providers and web service clients using XML Schema [31][32]. Nowadays, when data is increasingly available at growing rates, services play a critical role as entry points to access and manage this data.

The efficient processing of XML messages has a great influence in the overall performance of concrete implementations of these services. The implementation of XML processing for the OWS schemas is a complicated task because the size of the schemas has been growing with time, reaching a state that makes developers' work very difficult. Writing code to manipulate the resulting XML instances is complex whether we write this code manually or use a code generator. The first option is recognized to be difficult and error-prone producing code that is hard to modify and maintain [12]. The second option based on the use of code generators usually produces a large number of classes that either offers a poor performance or occupy a large disk space that limits its applicability in certain environments with constrained resources, such as mobile devices.

An example of the effect of complexity in a real implementation is the OX-Framework [5] , which defines an architecture to access data located in several kinds of OWS servers. The framework includes a client whose binary distribution occupies 58.8 MB, of which 46.4 MB is binary code for XML processing. This code, generated with XMLBeans[2], represents 79% of the size of the distribution and contains 33,437 classes. The framework libraries present a lot of redundancy on the generated XML processing code, as common depen-

[1]Open Geospatial Consortium
[2]http://xmlbeans.apache.org/

dencies of service specifications have not been properly factorised in the final code. In any case, even when eliminating this redundancy will largely decrease the overall size of the framework, the fraction of the code related to XML processing will still be the largest part.

In order to cope with these problems a first step is to measure the complexity of the schemas. For this reason, in this paper we present an analysis of the complexity of OWS schemas using a group of metrics found in the literature. We also introduce three new metrics to measure the complexity introduced by the use of the XML Schemas subtyping mechanisms. The use of adequate metrics allows us to quantify the complexity, quality and other properties of the schemas. The remainder of the paper is structured as follows. Next section presents a brief introduction to XML Schemas. Section 3 introduces OGC Web Services. After this, Section 4 presents the complexity analysis. In this section, we present metrics and their values for the considered specifications. Section 5 presents related work. Last, we present conclusions of our work.

2. XML SCHEMA

XML Schema files are used to assess the validity of well-formed element and attribute information items contained in XML instance files[31][32]. An XML Schema document mainly contains components in the form of complex and simple type definitions, element declarations, attribute declarations, group definitions, and attribute group definitions.

This language allows users to define their own types, in addition to a set of predefined types defined by the language, in the form of complex and simple types. Elements are used to define the types content and when global, to define which of them are valid as top-level element of a XML instance document. Figure 1 shows a fragment of an XML Schema file, which contains the declaration of three global complex types and a global element. Figure 1 also shows how recursive structures can be defined as for instance *ContainerType* that contains an element of the same type.

XML Schema provides a derivation mechanism to express subtyping relationships. This mechanism allows types to be defined as subtypes of existing types, either by extending the base types content model in the case of derivation by extension (*ChildType* in Figure 1); or by restricting it, in the case of derivation by restriction. Apart from type derivation, a second subtyping mechanism is provided through substitution groups. This feature allows global elements to be substituted by other elements in instance files. A global element E, referred to as *head element*, can be substituted by any other global element that is defined to belong to the E's substitution group.

Schema components defined in a schema document can be reutilized in other documents through the use of *include* and *import* tags. Components defined in the same namespace can be accessed in a schema file by using the include tag, which specifies in the *schemaLocation* attribute where the external schema is located. Similarly, components defined in a different namespace may be accessed by *importing* the namespace and optionally specifying where the external schema is located.

```
<complexType name="BaseType">
  <sequence>
    <element name="baseElement"
      type="string"minOccurs="1"/>
  </sequence>
  <attribute type="string" use="required"
    name="id"/>
</complexType>

<complexType name="ChildType">
  <complexContent>
    <extension base="BaseType">
      <sequence>
        <element name="childElement"
          type="string"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<complexType name="ContainerType"
  <sequence>
    <element name="containerElement" type="BaseType"
      maxOcurrs="unbounded"/>
    <element name="recursiveElement"
      type="ContainerType" minOcurrs="0"/>
  </sequence>
</complexType>

<xs:element name="container" type="ContainerType" />
```

Figure 1: XML Schema file fragment.

3. OGC WEB SERVICES

As mentioned before, geospatial web services interfaces, defined by OGC, describe both data formats and structure of messages exchanged by web services clients and providers using XML Schema. Table 1 shows a short description of the web service interfaces used later on this paper.

For each service specification we have chosen the last approved version at the time of writing this paper. For some of them, such as those related with sensors, new versions will be available soon, but their schemas are not still available on the official schemas repository in the OGC website[3]. These specifications are included in the Sensor Web Enablement (SWE) initiative, a framework of open standards for exploiting Web-connected sensors and sensor systems of all types [23]. A comprehensive review of the new features included on SWE can be found in [4].

An important point about service specifications is that, as reusability is an important requirement when building software systems, they have been built on the foundation provided by other standard specifications and data models such as the Geography Markup Language (GML)[13] and OWS Common [18] specifications (Figure 2). The reutilization of existing components simplifies the specification design task, but it also brings the complexity of the reutilized specifications into the other specifications as well. The schemas in

[3]http://schemas.opengis.net/

Table 1: Geospatial web service interfaces

Name	Description
Web Map Service (WMS)	It produces maps of spatially referenced data dynamically from geographic information [16]
Web Feature Service (WFS)	It allows a client to retrieve and update geospatial data encoded in GML format [25]
Web Coverage Service (WCS)	It provides access to rich sets of spatial information, in forms useful for client-side rendering, multi-valued coverages, and input into scientific models [24]
Sensor Observation Service (SOS)	It provides an API to retrieve sensor and observation data [22]
Web Processing Service (WPS)	It defines a standardized interface to publish geospatial processes [21]
Sensor Planning Service (SPS)	It defines interfaces for queries that provide information about the capabilities of a sensor and how to task the sensor[20]

OWS services are organized in a folder structure containing at the same level a folder for every specification. Each of them contains a folder for every version of the specification. Last, this folder contains the schema files, which we call *main specification schemas*. We differentiate these schemas from *external schemas*, which are included or imported from the main specification schemas.

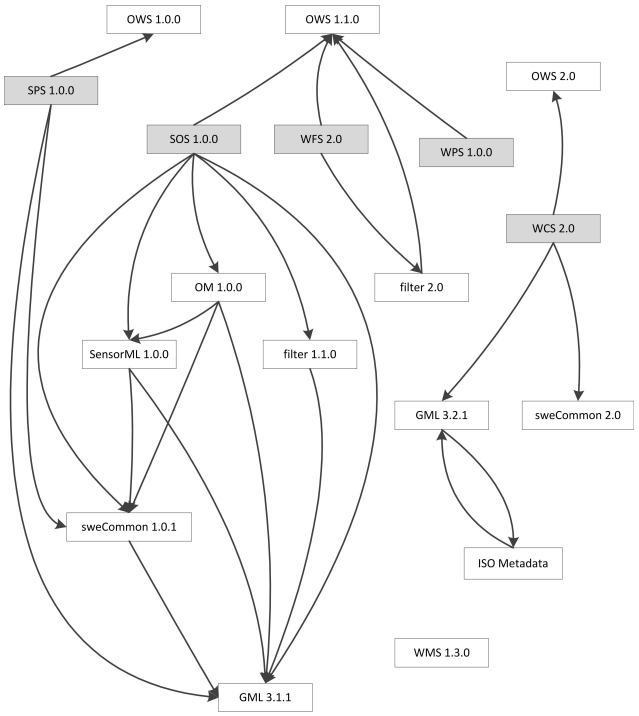

Figure 2: Dependencies between OGC specifications

4. COMPLEXITY ANALYSIS

In this section we present the complexity analysis for the schemas in the specifications listed in Table 1.

4.1 Metrics

For our study we have selected a set of metrics taken mainly from [9] and [11]. According to [9] metrics are categorized in *XML-agnostic*, *XSD-agnostic* and *XSD-aware*. *XML-agnostic metrics* do not consider any XML-related information. In this category we include for our analysis:

- *Lines of Code (LOC)*: Total number of lines of code on the specifications' schemas.
- *Number of files (#F)*: Total number of files related to the specification. Here, we consider recursively all of the files referenced through include and import XML schema statements.

XSD-aware metrics, consider metrics concerned with schema information. Here we consider a larger number of metrics:

- *Number of complex types (#CT)*: All complex types, including global and anonymous complex types.
- *Number of simple types (#ST)*: All simple types, including global and anonymous simple types
- *Number of global elements (#EL)*: All global element declarations.
- *Number of global model groups (#MG)*: All global model groups definitions
- *Number of global attributes (#AT)*: All global attribute declarations
- *Number of global attribute groups (#AG)*: All global attribute groups definitions
- *Number of global items (#ALL)*: Number of global schema components (types, elements, model groups, attributes and attribute groups).
- *Wildcards*: Number of times wildcards are used.
- *C(XSD)*: This metric calculates a complexity weight taking into account the internal schema components' structure [3].
- *Use of subtyping features of XML schema*: Here, we consider first basic counting metrics such as the number of times certain features such as substitution groups, specialization by restrictions and specialization by extension, are used. After this, we introduce three new metrics to measure the influence of subtyping on the complexity of the schemas: *Data Polymorphism Rate* (DPR), *Data Polymorphism Factor*(DPF) and *Schemas Reachability Rate* (SRR).

XSD-agnostic metrics do not consider any information related with XML schema but use XML-related information. Examples of these metrics are *Number of XML nodes* or *Number of XML nodes for annotations* [9]. In our analysis

we do not include metrics in this category because they are a measure of the size of the schemas, and this property is already measured by other considered metrics (e.g. LOC, #F).

To calculate the metric values for a given specification, all of the schema files imported directly or indirectly from the main specification schemas are included. For this reason, it must be noticed that an actual implementation may not provide support for all of the schema components included in all of the schema files, but only for a subset of it. The size of this subset will depend directly from the specific implementation requirements.

There are a lot of other metrics that could be included in this study, but we chose those we consider more relevant. In some cases we discard some metrics because the information they provide is similar to that provided by other metrics. Next, we present in detail the *C(XSD)* metric because it is more sophisticated than the other metrics considered in this study. The new metrics introduced in this paper will be explained in Section 5.4.

4.2 C(XSD) Metric Definition

In our study we include the metric presented in [3] that measures the schemas' complexity based in its internal structure, opposed to the metrics presented so far that limit themselves to just count schema components or features. It pays special attention to the use of recursive structures as a source of complexity to schema users. A complexity value, or *weight*, is calculated for each schema component as an aggregation of the weights of the components it contains. The overall value of the metric is calculated with the following formula:

$$C(XSD) = \sum_{i=1}^{N} C(E_{gi}) + \sum_{j=1}^{M} C(A_{gj}) + \sum_{t=1}^{K} C(EG_{gt}) + \sum_{r=1}^{P} C(AG_{gr}) + \sum_{s=1}^{R} C(CT_{gs}) + \sum_{q=1}^{Q} C(ST_{gq}) \quad (1)$$

where, the first two terms are the summation of weights of global element and attribute definitions respectively. The remaining terms are summation of weights of global unreferenced model groups, attribute groups, complex and simple types that are declared/defined in the main specification schemas. The values N, M, K, P, R and Q are the number of global elements, attributes, unreferenced element groups, unreferenced attributes groups, unreferenced complex types and unreferenced simple types respectively. In the second group of terms only unreferenced components are considered to avoid counting them several times as they are used in the declaration of global elements and attributes.

In [3] formulae are provided to calculate the weight of the different types of schema components. For example to calculate the weight of a complex type we use the following formula:

$$w_{type} = w_{baseType} \pm [\sum_{i=1}^{N} C(E_{gi}) + \sum_{j=1}^{M} C(A_{gj}) + \sum_{t=1}^{K} C(EG_{gt}) + \sum_{r=1}^{P} C(AG_{gr})] + NRC * R \quad (2)$$

where, $w_{baseType}$ [4] is the weight of the base type. If derivation is not explicit (*anyType* is the base type) the weight of the base type is 1. If we are in the case of derivation by extension, *N, M, K, P,* are the number of not inherited local or referenced elements, attributes, element and attribute groups referenced, that are not related to any element containing recursion. The sum of all these values is added to the weight of the complex type. If the complex type is derived by restriction, *N, M, K, P*, are the corresponding number of schema components not inherited from the base type, and its weight is subtracted from the weight of the base type. In both cases *NRC* is the number of child elements that contains recursion; and R is an integer value greater or equal than 1 that can be understood as the weight given to recursion in a schema set.

5. RESULTS

The results of applying the metrics mentioned before to the specification schemas listed in Table 1 are shown in the following subsections.

5.1 XML-Agnostic Metrics

We start with *XML-agnostic metrics*, which are those that do not consider XML-related information. The total amount of lines of code (LOC) and the number of files (#F) give us a raw idea of the size of a given schema set. Table 2 shows the value of the metrics for the considered OWS specifications. According to the categorization for LOC values presented in [9], a schema set with between 10,000 and 100,000 LOC is considered *large*. Values between 1,000 and 10,000 correspond to *medium*-sized schemas. And values between 1,000 and 100 correspond to *small* schemas. There are also categories for *mini* schemas, below 100 LOC, and *huge* schemas, above 100,000 LOC. Considering the overall values, 3 out of 6 of the specifications are considered large, two of them are considered medium-sized and the last one is considered small.

Table 2: Lines of code (LOC) and number of files (#F)

	SOS 1.0.0	WFS 2.0	WCS 2.0	SPS 1.0.0	WPS 1.0.0	WMS 1.3.0
LOC	17,581	3,631	15,416	14,361	3,326	761
#F	87	23	87	73	29	3

WCS and the specifications related with sensors, SOS and SPS, exhibit the higher values for the metrics. It is not a coincidence that they are the ones with higher number of dependencies from other specifications (Figure 2). On the other hand, WMS turns out to be the simplest (which is

[4]The notation $w_{typename}$ is equivalent to $C(CT_{typename})$

maybe why is the most widespread) of the specifications being, in terms of lines of code, about 20 times smaller than SOS. WMS does not depend on any major external schema for its definition. From the results might be a little surprising such low figures for WFS. This is because WFS schemas are not linked explicitly to GML schemas, in which case the values for LOC and #F would be very similar to those of WCS.

5.2 XSD-Aware Simple Metrics

In this section we present the results of applying the XSD-aware metrics that count the number of main schema components. We start with the number of complex types (#CT) which is considered paramount for measuring complexity because it measures the number of structured concepts modelled by the schemas [9]. Also, types are the fundamental concept when schemas are used to write (or generate) XML data binding code. The #CT metric includes global complex types, as well as anonymous complex types. The number of complex types by specification is shown in Table 3 and compared with other metrics in Figure 3.

Table 3: Number of Complex Types (#CT)

	SOS 1.0.0	WFS 2.0	WCS 2.0	SPS 1.0.0	WPS 1.0.0	WMS 1.3.0
#CT	740	163	797	587	99	38

Schemas with #CT in the range 256-1,000 are considered *large*, in the range 100-256 are considered *medium* and *small* with #CT between 32 and 100 [9]. The values of the overall metrics for all of the 6 specifications belong to these three ranges, 3 of them are large, one of them is medium-sized and the rest are small schemas. Again, WCS, SOS and SPS are among the most complex schemas and WMS is the simplest. As complex types model concepts, we can state that higher values of the metric imply higher conceptual complexity.

A categorization of the schemas based on the number of other schema components is not provided in the literature. Nevertheless, they can give us some idea of schemas size and how often these features are used in the specifications.

Table 4: Main XML features metrics (except #CT)

	SOS 1.0.0	WFS 2.0	WCS 2.0	SPS 1.0.0	WPS 1.0.0	WMS 1.3.0
#ST	118	46	74	105	18	5
#EL	727	156	754	593	64	60
#MG	28	3	14	19	7	2
#AT	23	15	20	5	7	0
#AG	40	12	17	44	10	7
#ALL	1498	350	1625	1266	179	80

Table 4 shows the overall values of the metrics for these schema components, which are also included in Figure 3. These values reinforce the idea of having a clear differentiation between a first group containing large specifications (SOS, SPS and WCS), a second group of medium-sized specifications (WFS), and a third group containing small specifications (WPS and WMS). In Figure 3 we can observe the correlation that exists between the values of the metrics. This observation suggests that the coding style used in the schemas is consistent through all of the specifications.

We would like to point out again that an actual implementation of WFS would require the use of other schemas, making them ascend to the category of large schemas.

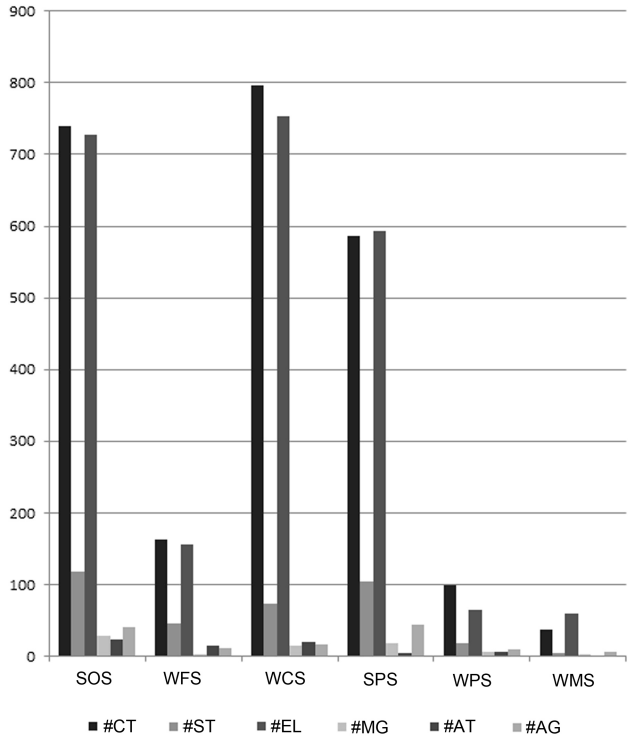

Figure 3: Number of main XML Schema components

Last, we count wildcards, which allow schema designers to specify extensibility points using <*any*> and <*anyAttribute*> tags. Using these tags in a complex type definition indicate that any global element or attribute can occupy that place in an instance document. The scope of the valid elements the wildcard is substituted for can be constrained by their namespace.

The use of wildcards is widespread with the purpose of keeping schemas extensible, but they greatly complicate the processing of instance files. A discussion about why the use of wildcards should be avoided when designing web service interfaces can be found in [26]. We just want to highlight the fact that when parsing an XML instance we cannot be sure of what we will find in the place of the wildcards, so we must be ready to find almost anything. This obviously makes the source code for processing the instances files more complicated. If instead of writing the code manually we use a code generator, the presence of wildcards limits their possibilities of performing optimizations.

Table 5 shows how wildcards are used in the specifications. In this case, only the specifications already labelled as large make use of this feature.

Table 5: Use of wildcards

	SOS 1.0.0	WFS 2.0	WCS 2.0	SPS 1.0.0	WPS 1.0.0	WMS 1.3.0
Wildcards	13	13	5	9	0	0

5.3 C(XSD)

C(XSD) was introduced in Section 4.1. The metric calculates a weight for each schema component taking the internal structure of the component into account. Table 6 shows the value of the metric.

Table 6: C(XSD) values for OWS specifications

	SOS 1.0.0	WFS 2.0	WCS 2.0	SPS 1.0.0	WPS 1.0.0	WMS 1.3.0
C_{XSD}	261,238 + 2,381R	1,960 + 16R	209,997 + 1,171R	96,451 + 885R	1,578 + 2R	707 + 3R

Previous metrics have shown that SOS was among the most complex specifications. C(XSD) shows that considering the internal structure of the schema components SOS is significatively more complex than the next specification, WCS. This is motivated by the fact that SOS has the higher number of dependencies from other specifications (GML, SWE Common, OWS Common, O&M[17], SensorML[19], etc.), which are complex specifications as well. SensorML and O&M contain the most complex schema components if analysed individually. The schema component with the higher individual value for C(XSD) is *Component* in SensorML with 23,016 + 219R. Coincidentally, this element contains the highest number of recursive branches in its definition.

5.4 Subtyping Mechanisms

XML Schema subtyping mechanisms were introduced in Section 2. In this category we count first the number of abstract elements or types (#AET), number of substitution groups (#SG) and number of complex types derived by restriction or extension (#TD). The values of these metrics are shown in Table 7.

Table 7: Use of subtyping mechanisms

	SOS 1.0.0	WFS 2.0	WCS 2.0	SPS 1.0.0	WPS 1.0.0	WMS 1.3.0
#AET	61	15	74	52	2	2
#SG	69	11	123	61	2	0
#TD	349	56	371	297	31	4

The results show that subtyping mechanisms are widely used in the specifications leading to an elevated number of non-explicit dependencies between schema components. This may lead to inadvertently overlook important details when analysing dependencies.

In order to measure in greater detail the influence of the subtyping mechanisms on complexity we introduce three new metrics: *Data Polymorphism Rate(DPR)*, *Data Polymorphism Factor* (DPF) and *Schemas Reachability Rate* (SRR).

The term *Data Polymorphism (DP)* refers to the fact that nodes in XML instance files have a *declared type*, but also have a *dynamic type*. This is because global elements can be subtituted by any element in its substitution group. Similarly, an element in an instance file may be of any type derived from the declared type. This situation is similar to *polymorphism* in the Object-Oriented Programming (OOP) context.

5.4.1 Data Polymorphism Rate

The *Data Polymorphism Rate (DPR)* is a measure of how much polymorphism is contained in the schemas. It is expressed by the formula:

$$DPR = \frac{\sum_{i=1}^{N} PE_{CTi}}{\sum_{j=1}^{N} E_{CTj}} \quad (3)$$

In the formula, N is the total number of complex types, PE_{CTi} is the number of elements in the declaration of the complex type CT_i that are polymorphic, i.e., its dynamic type may differ from its declared type in instance files. E_{CTj} is the number of elements in the type declaration of type CT_j. For every type, a reference to a global element and an inner element declarations are considered as equals and count as 1. As a consequence, the numerator is the total number of polymorphic elements on the schemas. Similarly, the denominator is total number of elements contained in all complex types in the schemas.

The result value is in the interval [0, 1], indicating the fraction of the elements that are polymorphic. This metric is a variation of the Polymorphic Factor (POF) metric used in the OOP context [1].

Table 8: DPR values for the OWS specifications

	SOS 1.0.0	WFS 2.0	WCS 2.0	SPS 1.0.0	WPS 1.0.0	WMS 1.3.0
DPR	0.13	0.12	0.15	0.13	0.05	0

From the results shown in Table 8 we can observe that simpler specifications contains zero or a low degree of polymorphism. The rest of the specifications have a similar degree ranging between 12 and 15%. Saying if these values are too high or not is not a trivial task, however in [1], analysing polymorphism in the context of OOP is stated that a values of POF above 10% is expected to reduce the benefits obtained with an appropriate use of polymorphism. This is because highly polymorphical hierarchies will be harder to understand, debug and maintain.

5.4.2 Data Polymorphism Factor

The previous metric gives an idea of the number of polymorphic elements on the schemas, but does not measure their influence in the overall schemas complexity. For instance, a polymorphic element that can be substituted by

two other elements does not have the same effect in complexity as an element that can be substituted by twenty different elements. In this regard, we define the *Data Polymorphism Factor(DPF)*as follows:

$$DPF = \frac{\sum_{i=1}^{N} OE_{CTi}}{\sum_{j=1}^{N} E_{CTj}} \quad (4)$$

In this case, OE_{CTi} is the number of possible different elements that could be contained in a complex type. It is the summation of the number of elements declared in CT_i, the elements in the substitution groups of those elements, and the number of possible dynamic types that can have any element in CT_i different from its declared type. For example, $OE_{ContainerType} = 3$ in Figure 1, because it contains two element declarations and one of them, *containerElement*, may have a different dynamic type: *ChildType*. The denominator is the same as in the definition of the DPF metric. In the formula $OE_{CTi} >= E_{CTi}$ for all i, natural number in the interval $[1, N]$. As a consequence the values of DPF are always equal or greater than 1, representing the factor in which the number of elements to be considered might grow when polymorphic elements are taken into account.

Table 9: DPF values for the OWS specification schemas

	SOS 1.0.0	WFS 2.0	WCS 2.0	SPS 1.0.0	WPS 1.0.0	WMS 1.3.0
DPF	2.20	1.47	1.48	2.20	1.05	1

Table 9 shows the value of the metric for the schemas of the different specifications. These results show that the effect of polymorphic elements on SOS and SPS is higher that in WFS and WCS. Presumably this is caused by the larger number of dependencies of the sensor related specifications. SOS and SPS have a lot of common dependencies that is why DPF and DPFA values are basically the same for both specifications. As the simplest specifications barely contain polymorphical elements, the values of DPF for them are equal or close to the minimal value, 1.

5.4.3 Schemas Reachability Rate

The last metrics proposed in this paper, *Schemas Reachability Rate (SRR)*, attempts to measure the fraction of *imported schema components* that are hidden (not referenced explicitly) by the subtyping mechanisms. As mentioned in Section 3, OWS specification schemas reutilise other specification schemas by *importing* them in the main specification schemas. An imported component may be referenced directly if it is explicitly mentioned in the declaration of a schema component in the main schemas. But, it can also be referenced *indirectly*, if it is for example in the substitution group of a referenced element, or its derived from a type that is referenced directly. For example, it is not clear for everybody that if we are using GML 3.1.1 in our schemas and we define an inner element to be of type *gml:AbstractFeatureType*, this element may have 13 different dynamic types (considering only GML types) on instance files based on these schemas.

To calculate SRR, we define first G_S, G_{SH} and $V_{Rm}(G)$ as follows:

Definition 1: *We define G_S for the schemas in a specification S as the directed graph $G_S = (V_S, E_S)$, where vertices in V_S, are all of the global elements declared in all of the schemas related to S (main and external schemas). E_S are directed edges between these vertices. An edge from V_i to V_j exists if V_j is used somehow in the declaration of V_i.*

Definition 2: *We define G_{SH} for the schemas in a specification S as the directed graph $G_{SH} = (V_{SH}, E_{SH})$, where $V_{SH} = V_S$. E_{SH} extends E_S by including also non-explicit dependencies, i.e. an edge from V_i to V_j exists if V_j is used somehow in the declaration of V_i, or if V_j, is in the substitution group of an element referenced from V_i, or V_j is a type derived from a type used in the declaration of V_i.*

Definition 3: *We define, $V_{Rm}(G)$ for a directed graph $G = (V, E)$ and V_m, a subset of V, as the subset containing all of the vertices in V that are reachable from at least a vertex in V_m*

Based on these definitions, if we consider that $V_m(G)$ is the subset of $V(G)$ containing the schema components included in the main schemas, $V_{Rm}(G)$ would contain any schema component that is reachable from the main schemas. In the case of G_S, this will be components reachable through explicit dependencies, and in the case of G_{SH}, these are reacheable components through explicit and non-explicit dependencies. Using these vertex sets the SRR metric is calculated as follows:

$$SRR = \frac{|V_{Rm}(G_{SH})| - |V_{Rm}(G_S)|}{|V_S|} \quad (5)$$

The metric measures the fraction of schema components a specification depends from, but that are not explicitly referenced from any component in the main schemas, or any component reachable through explicit dependencies from the main schemas.

Figure 4 shows the graph of component relations for the schema fragment in Figure 1. If we ignore the hidden dependency between *ContainerType* and *ChildType* we have G_S, otherwise we have G_{SH}. If we consider, for example, that the declaration of element *container* and type *ContainerType* are located in the main schemas, and *BaseType* and *ChildType* declarations are located in external schemas we could calculate the value of DPRF for the main schemas: $V_m = \{container, ContainerType\}$, $V_{Rm}(G_S) = \{container, ContainerType, BaseType\}$, $V_{Rm}(G_{SH}) = \{container, ContainerType, BaseType, ChildType\}$, so:

$$SRR = \frac{4-3}{4} = 0.25$$

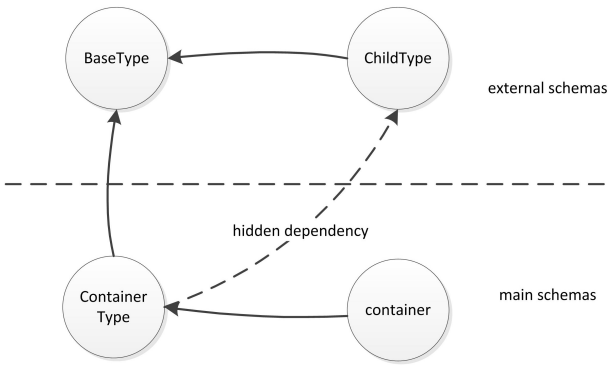

Figure 4: Graph of schema component relations for schema fragment in Figure 1

This value means that a quarter of the schema components are referenced through non-explicit dependencies.

Table 10 shows the value of the cardinalities of the involved vertex sets and the value of the SRR metric for the schemas of the different specifications. The results shows that for SOS, WCS, and SPS more than 60% of the schema components that could be used in instance files are not referenced explicitly from the schema component in the main schemas, or any component that is referenced from them. This high rate suggests that the effect of the subtyping mechanism in schemas complexity is enormous. For the rest of the specification the effect goes from moderate (WFS, WPS) to non-existent (WMS).

Table 10: SRR values for the OWS specifacion schemas

	SOS 1.0.0	WFS 2.0	WCS 2.0	SPS 1.0.0	WPS 1.0.0	WMS 1.3.0
$\|V_{Rm}(G_{SH})\|$	1277	321	1349	1058	146	71
$\|V_{Rm}(G_S)\|$	319	245	220	203	126	71
$\|V_S\|$	1498	353	1625	1266	179	80
SRR	0.64	0.22	0.69	0.68	0.11	0

6. RELATED WORK

Literature about measuring XML schemas complexity has increased in the last few years, based mainly on adapting metrics for assessing complexity on software systems or XML documents [2] [10] [28]. To our best knowledge the most relevant attempt in this topic is presented in [9]. Here, a comprehensive set of metrics is defined and applied to a large corpus of real-world XML schemas. Based on the resulting metrics, the authors define a categorization for a set of schema files according to its size. Another relevant study is [11] which defines eleven metrics to measure the quality and complexity of XML Schemas.

In [3], the authors present a more sophisticated metric that takes into account, not only the number of main schema components like the previous mentioned works, but also the internal structure of these components. In a similar way, [30] proposes more advanced schema metrics, arguing that previous work on the topic only measure size as an approximation for complexity. The authors present a set of metrics to measure other structural properties of the schemas.

Last about schemas complexity, in [27] the authors present a set of schema metrics in the context of schema mapping. A combined metric is defined based in simpler metrics considering schemas size, use of different schema features, and naming strategies. The combined metric is evaluated in the context of business document standards.

Similar studies in the geospatial domain are scarce, though, an interesting discussion of complexity can be found in [7] [8]. This discussion tries to identify the origin of GML complexity and use some of the metrics in [9] to categorize its schemas. Our research attempts to extend this discussion to the OWS specifications, but focusing more in the complexity of the schemas themselves.

It is also worth mentioning that the problem of schemas complexity has been usually dealt with by using XML data binding code generators [12] or by using *schema profiles* [29]. The first solution allows the automatic generation of XML processing code. Although these generators generally produce acceptable results, in presence of large schemas they may produce code that is excessively large or do not meet performance application requirements. The second solution is based on extracting subsets of the schemas that are relevant to an actual implementation or problem domain. By using only a portion of the schemas, the complexity of handling or understanding them is reduced in a large degree. Examples of profiles in the context of GML are the Simple Feature Profile[15] and the Common CRS Profile [14].

7. CONCLUSIONS

In this paper we have presented a quantitative way to analyse and measure the complexity of OWS' schemas. The results of the analysis have shown that at least half of the presented specifications can be considered as large and complex according to all of the metrics included in our study. Most of the metrics coincide in finding a clear differentiation between a first group containing large specifications (SOS, SPS and WCS), a second group containing medium sized specifications (WPS and WFS), and a third group of simple specifications (WMS). More complex specifications, as a general rule, are those that present a larger number of dependencies from other specifications.

The new metrics introduced here have shown from different views the effect of the use of subtyping mechanism on complexity. For example, DPR has shown the fraction of polymorphic elements, frequently high, included in the schemas. DPF has considered how the possible polymorphic situations for these elements increase the effort needed to fully understand the schema components definitions. Last, SRR has shown that more than 60% of the schema components included in large specifications are referenced in ways that cannot be seen explicitly, augmenting the risk of making mistakes while working with the schemas.

The metric set presented here should not be seen as a closed set, many other metrics can be useful in many different sce-

narios. The use of adequate metrics allows us to quantify the complexity, quality and other properties of the schemas. This can be very useful in different scenarios, such as, evaluating the impact of design decisions, assessing the effectiveness of different solutions to deal with schemas complexity (e.g. how the use of schema profiles simplifies the implementation and comprehension of a given problem). Metrics can also be very useful to detect potential design problems such as components with to many information items, or excessively deep subtyping hierarchies, just to mention some examples. Last, metrics can also suggests solutions about how to deal with the large size and complexity of geospatial schemas.

8. ACKNOWLEDGEMENTS

This work has been partially supported by the "España Virtual" project (ref. CENIT 2008-1030) through the Instituto Geográfico Nacional (IGN).

9. REFERENCES

[1] F. B. Abreu and W. Melo. Evaluating the Impact of Object-Oriented Design on Software Quality. In *Proceedings of the 3rd International Symposium on Software Metrics: From Measurement to Empirical Results, METRICS '96*, volume 0, page 90, Los Alamitos, CA, USA, 1996. IEEE Computer Society.

[2] D. Barbosa, L. Mignet, and P. Veltri. Studying the xml web: Gathering statistics from an xml sample. *World Wide Web*, 8:413–438, 2005.

[3] D. Basci and S. Misra. Measuring and evaluating a design complexity metric for XML schema documents. *Journal of Information Science and Engineering*, 25(5):1405–1425, September 2009.

[4] A. Bröring, J. Echterhoff, S. Jirka, I. Simonis, T. Everding, C. Stasch, S. Liang, and R. Lemmens. New Generation Sensor Web Enablement. *Sensors*, 11(3):2652–2699, 2011.

[5] A. Bröring, E. H. Jürrens, S. Jirka, and C. Stasch. Development of Sensor Web Applications with Open Source Software. In *Open Source GIS 2009*. University of Nottingham, Suchith Anand, June 2009.

[6] M. F. Goodchild, C. Kottman, and M. J. Egenhofer, editors. *Interoperating Geographic Information Systems*. Kluwer Academic Publishers, Norwell, MA, USA, 1st edition, 1999.

[7] R. Lake. GML Complexity. *http://www.galdosinc.com/archives/186*, (Last Accesed 2010-12-09).

[8] R. Lake. GML Complexity Re-visited. *http://www.galdosinc.com/archives/140*, (Last Accesed 2010-12-09).

[9] R. Lämmel, S. Kitsis, and D. Remy. Analysis of XML schema usage. In *Proceedings of XML Conference 2005*, pages 1–35, 2005.

[10] T. McCabe. A Complexity Measure. *IEEE Transactions on Software Engineering*, 2:308–320, 1976.

[11] Y. K. McDowell A., Schmidt C. Analysis and Metrics of XML Schema. In *International Conference on Software Engineering Research and Practice*, pages 538–544, 2005.

[12] B. McLaughlin. *Java and XML Data Binding*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2002.

[13] OGC. OpenGIS Geography Markup Language (GML) Implementation Specification 3.1.1. *OGC Document*, (03-105r1), 2004.

[14] OGC. GML 3.1.1 Common CRSs Profile. *OGC Document*, (05-095r1), 2005.

[15] OGC. Geography Markup Language (GML) Simple Features Profile. *OGC Document*, (06-049r1), 2006.

[16] OGC. OpenGIS Web Mapping Server Implementation Specification 1.3.0. *OGC Document*, (06-042), 2006.

[17] OGC. Observations and Measurements - Part 1 - Observation schema. *OGC Document*, (07-022r1), 2007.

[18] OGC. OGC Web Services Common Specification 1.1.0. *OGC Document*, (06-121r3), 2007.

[19] OGC. OpenGIS Sensor Model Language (SensorML) Implementation Specification 1.0.0. *OGC Document*, (07-000), 2007.

[20] OGC. OpenGIS Sensor Planning Service Implementation Specification 1.0.0. *OGC Document*, (07-014r3), 2007.

[21] OGC. OpenGIS Web Processing Service 1.0.0. *OGC Document*, (05-007r7), 2007.

[22] OGC. Sensor Observation Service 1.0.0. *OGC Document*, (06-009r6), 2007.

[23] OGC. OGC Sensor Web Enablement: Overview And High Level Architecture. *OGC Whitepaper,*, 2008.

[24] OGC. OGC WCS 2.0 Interface Standard - Core. *OGC Document*, (09-110r3), 2010.

[25] OGC. OpenGIS Web Feature Service 2.0 Interface Standard. *OGC Document*, (09-025r1), 2010.

[26] J. Pasley. Avoid XML Schema Wildcards For Web Service Interfaces. *IEEE Internet Computing*, 10:72–79, May 2006.

[27] C. Pichler, M. Strommer, and C. Huemer. Size matters!? measuring the complexity of xml schema mapping models. *IEEE Congress on Services*, 0:497–502, 2010.

[28] M. H. Qureshi and M. H. Samadzadeh. Determining the complexity of xml documents. *ITCC'05*, 2:416–421, 2005.

[29] R. Singh. Use Profiles to Overcome GML's Complexity. *GEO World*, 20:21–21, 2007.

[30] J. Visser. Structure metrics for XML Schema. In *Proceedings of XATA 2006*, pages 236–247, 2006.

[31] W3C. XML Schema Part 1: Structures Second Ed. *http://www.w3.org/TR/xmlschema-1*, (Last Accesed 2010-12-09).

[32] W3C. XML Schema Part 2: Datatypes Second Ed. *http://www.w3.org/TR/xmlschema-2*, (Last Accesed 2010-12-09).