International Journal of Geographical Information Science
Vol. 00, No. 00, Month 200x, 1–28

RESEARCH ARTICLE

Measuring Complexity in OGC Web Services XML Schemas: Pragmatic Use and Solutions

(*Received 00 Month 200x; final version received 00 Month 200x*)

The use of standards in the geospatial domain, such as those defined by OGC, for exchanging data has brought a great deal of interoperability upon which systems can be built in a reliable way. Unfortunately, these standards are becoming increasingly complex making their implementation an arduous task. The use of appropriate software metrics can be very useful to quantify different properties of the standards that ultimately may suggest different solutions to deal with problems related to their complexity. In this regard, we present in this article an attempt to measure the complexity of the schemas associated to OGC implementation specifications. We use a comprehensive set of metrics to provide a multidimensional view of schemas complexity. **These metrics can be used to evaluate the impact of design decisions, study the evolution of schemas, etc.** We also present and evaluate different solutions that could be applied to overcome specifications complexity.

Keywords: XML Schema, OGC Web Services, Geospatial Information, Complexity Analysis, Software Metrics, Standards

For Peer Review Only

ISSN: 1365-8816 print/ISSN 1362-3087 online
© 200x Taylor & Francis
DOI: 10.1080/1365881YYxxxxxxxx
http://www.informaworld.com

1. Introduction

The standards defined by the Open Geospatial Consortium (OGC) are a vehicle for the interoperability between geographical information providers and consumers and allow the integration of data coming from a continuously growing number of sources. The number and size of the standards have been growing in the last few years as the number of supported use cases for geospatial data exchange or processing increases at rapid rates. The complexity of the standards is well-known, but rarely mentioned in research literature. Besides, little effort has been made to measure it properly using well-defined software metrics. Understanding how and why the specifications have grown and finding solutions to deal with this complexity is something that can be hardly accomplished without the use of appropriate metrics to control and assess the evolution of the specifications.

Complexity in geospatial web services comes mostly from the complexity of the problem domain. As stated in (Goodchild *et al.* 2007), *"geographic representation has become more complex through time as researchers have added new concepts, leading to apparently endless proliferation and creating a need for simplification"*. Although we cannot deny the inherent complexity of the problem domain, during the modelling process of geospatial standards, some extra complexity is introduced due to design decisions. Besides, as the standards are implementation specifications, the path is not complete until concrete implementations are built, hence, other implementation-related aspects, such as poor tool support, may increase the perceived complexity of the specifications. **In (Fu and Sun 2010) the need for easier standards is highlighted. According to the authors easier standards get wider adoption, such as the case of GeoRSS**[1] **and WMS (OGC 2006b) that have been applied to a great variety of real-world applications.**

Of special interest to our work are the schemas included in the OGC Web Services specifications (OWS) as they define the structure of exchanged messages and data. **The schemas, defined using the XML Schema language (W3C 2004a,b), are in most cases the only machine-processable information associated to the specifications. They can be used with different purposes such as evaluating if an existing implementation complies with the standard or even more, can be used to generate substantial portions of these implementations.** In our opinion, the complexity of the schemas reflects the complexity of the whole web service specifications. This is because, on the one hand, these web services interfaces do not contain a large number of operations requiring users to have a complex flow of interactions with the servers. On the other hand, the amount of increasingly structured data exchanged between clients and servers has reached levels that may represent a serious challenge when building a real system. **For example, (Tamayo *et al.* 2011c) present the obstacles to produce XML processing code for a mobile client for the Sensor Observation Service (SOS) specification (OGC 2007g) using existing XML data binding technologies. As a consequence of the large size of the SOS schemas, the code generated by these tools exceeds the hardware limitations of mobile devices. The alternative, writing the XML processing code manually, in the words of Tim Bray one of the co-creators of XML is *"...irritating, time-consuming, and error-prone"*(Bray 2003).**

In this context, we present in this article an attempt to measure the complexity of XML schemas included in the OWS specifications. The complexity

[1] http://www.georss.org

For Peer Review Only

is measured using a set of metrics extracted from the literature (Lämmel *et al.* 2005, Basci and Misra 2009, McDowell A. 2005), as well as a set of metrics defined by ourselves. The main goal of this study is to provide a multidimensional view of the schemas complexity, as the use of a single metric cannot comprise all of the aspects that influence their perceived complexity (Kaner *et al.* 2004, Mens and Demeyer 2001). Therefore, the metrics presented here should not be seen as competing metrics but as complementary ones. We also highlight the practical use that can be given to these metrics, for example, to compare competing specifications, to evaluate design decisions, to estimate implementation effort of adopting or migrating to a given specification, etc.

This work is an extended version of (Tamayo *et al.* 2011b), considering a wider set of OWS specifications and a discussion on pragmatic uses of these metrics. The remainder of this article is structured as follows. Section 2 introduces briefly the XML Schema language and the OGC specifications considered in this article. Next, Section 3 presents related work in the subject. After this, the metrics included in the complexity study and the results of calculating their values are exposed in sections 4 and 5. Sections 6 and 7 present pragmatic aspects related to the metrics such as practical use and solutions to deal with complexity. Lastly, conclusions and future work are presented.

2. Background

In this section we briefly introduce the XML Schema language, which is used to assess the validity of well-formed element and attribute information items contained in XML documents. We also introduce the OGC specifications that will be used in the complexity study.

2.1. *XML Schema*

An XML Schema document mainly contains components in the form of complex and simple type definitions, element declarations, attribute declarations, group definitions and attribute definitions. Figure 1 shows a fragment of an XML Schema file, which contains the declaration of three global complex types and a global element. The figure also shows how recursive structures can be defined. For instance, *ContainerType* contains an inner element of the same type.

Figure 1. XML Schema file fragment.

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 their content models in the case of *derivation by extension* (*ChildType* in Figure 1); or by restricting them, in the case of *derivation by restriction*. Apart from type derivation, the language provides a mechanism called *element 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. Because of these mechanisms the actual or *dynamic type* of an XML node may differ from the type declared in the schemas (*declared type*).

For Peer Review Only

Schema components defined in a schema document can be reutilised 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.

XML Schema is considered a very complex language. (Martens *et al.* 2006) stated that complexity of the XML Schema and the difficulty of understanding the effect of constraints on typing and validation of schemas might be the cause that in practice the extra expressiveness of these schemas over its predecessor, *Document Type Definition* (DTD), is only used to a very limited extent. (Mller and Schwartzbach 2006) present a list of limitations of XML Schema arguing that one important factor of its complexity is the type system, as information items cannot be matched directly to its associated constraints like it is done with DTD. These authors also consider the inclusion of both subtyping mechanisms introduced previously to the presence of conflicting design approaches in the W3C XML Schema Working Group, that have resulted in an unnecessarily complex specification. On the other hand, (Hosoya 2010) attributes the complexity of the schema language to the fact of attempting to mix two completely different notions: regular expressions and object orientation. Regular expressions was the most common approach to deal with XML, as they are very convenient to handle structured text. But, this approach based on automata theory cannot be seamlessly integrated with the hierarchical model proposed by object orientation.

2.2. *OGC Specifications*

OGC specifications include web services interfaces as well as data encodings to be used in geospatial applications. As reusability is an important requirement in the specifications' design, they are built reutilising components defined in other specifications. The reutilisation of existing components simplifies the specification design task, but it also brings the complexity of the reutilised specifications into the new specifications as well. The specifications used in our study are listed in Table 1. In this article, when referring to schemas in a specification, we differentiate between *main schemas*, those defined completely in the specification, and *external schemas*, those included or imported from the main specification schemas.

3. Related Work

Literature about measuring XML schemas complexity has increased in the last few years. They are based mainly on adapting metrics for assessing complexity in software systems (McCabe 1976, Chidamber and Kemerer 1994) or in XML documents (Barbosa *et al.* 2005, Qureshi and Samadzadeh 2005). To our best knowledge the most relevant attempt in this topic is presented in (Lämmel *et al.* 2005). The authors conducted a comprehensive study of a sample of XML schemas and proposed a categorisation of schemas according to its size. Most of the metrics considered in that study are limited to count distinct XML Schema features, although more complex metrics such as the Mcabbe's ciclomatic complexity (McCabe 1976) were adapted to measure schemas complexity. Another relevant study is (McDowell A. 2005) which defines fourteen metrics to measure the quality and complexity of XML Schemas. **The authors considered five of the metrics already**

For Peer Review Only

included in (Lämmel *et al.* 2005): number of *complex types*, *simple types*, *annotations*, *derived complex types* and *global type definitions*. In addition to these and other simple metrics, they also defined two composite indices to measure a *quality index* and a *complexity index*.

In (Basci and Misra 2009), the authors presented 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. A weight is assigned to every schema component based on the weight of its inner components. These metrics also recognise recursive structures as a feature that increases schemas complexity.

In a similar way, (Visser 2006) proposed more advanced schema metrics, arguing that previous works in the topic only measure size as an approximation for complexity. The author presented a set of metrics to measure other structural properties of the schemas but did not provide any evidence of why these metrics are a better approximation to complexity than others. **In that study, metrics *fan-in* and *fan-out*, to measure the number of incoming and outgoing edges from a graph representing schema components as nodes, are very similar to the *element fanning* metrics presented in (McDowell A. 2005).**

Last about schemas complexity, (Pichler *et al.* 2010) presented a set of schema metrics in the context of schema mapping. Based on a set of metrics that consider schemas size, use of different schema feature and naming strategies a combined metric was defined. The combined metric was evaluated in the context of business document standards.

Similar studies in the geospatial domain are scarce, though, an interesting discussion of complexity in the context of the Geographic Markup Language (GML) can be found in Ron Lake's blog[1]. This discussion tries to identify the origin of GML complexity and uses some of the metrics in (Lämmel *et al.* 2005) to categorise GML schemas. Our research attempts to extend this discussion to the OWS specifications, but focusing more on the complexity of the schemas themselves.

4. Metrics

In this section we present the metrics used in the complexity analysis of the specifications listed in Table 1. We consider here a wide set of metrics, ranging from simple metrics that simply count schema features to more complex metrics that attempt to give an overall value for complexity or measure specific aspects of the schemas. The main criteria to select the metrics is that they be *intuitively* understood, i.e., that we could easily understand what we are measuring, in contrast to the use of *esoteric* **or *artificial*** metrics, which end up having few interest for the industry (Fenton and Neil 1999).

The first group of metrics, named here as *XSD-aware metrics* after the terminology introduced in (Lämmel *et al.* 2005), is listed next:

- *Number of complex types (#CT)* : All complex types, including global ($\#CT_G$) and anonymous ($\#CT_A$) complex types **(Lämmel *et al.* 2005, McDowell A. 2005)**
- *Number of simple types (#ST)*: All simple types, including global ($\#ST_G$) and anonymous ($\#ST_A$) simple types **(Lämmel *et al.* 2005, McDowell A. 2005)**
- *Number of global elements (#EL)*: All global element declarations **(Lämmel *et al.* 2005)**

[1] http://www.galdosinc.com/archives/140, http://www.galdosinc.com/archives/186

For Peer Review Only

- *Number of global model groups (#MG)*: All global model groups definitions **(Lämmel *et al.* 2005)**
- *Number of global attributes (#AT)*: All global attribute declarations **(Lämmel *et al.* 2005)**
- *Number of global attribute groups (#AG)*: All global attribute groups definitions **(Lämmel *et al.* 2005)**
- *C(XSD)*: This metric calculates a complexity weight taking into account the internal schema components' structure. It calculates approximately the number of primitive (or atomic) information items that must be considered to fully understand a set of schemas, as well as the number of recursive branches contained on the schemas (Basci and Misra 2009).

The second group contains metrics that attempt to measure the influence in complexity of the subtyping mechanism of XML Schema. The term *Data Polymorphism (DP)* is used to refer to the fact that XML nodes may have a a *dynamic type* that differs from its *declared type*. This situation is similar to polymorphism in the context of Object-Oriented Programming (OOP) Languages. This group includes the following metrics:

- *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, specialisation by restrictions and specialisation by extension **(Lämmel *et al.* 2005, McDowell A. 2005)**
- *Data Polymorphism Rate* (DPR): This is a measure of how much polymorphism is contained in the schemas.
- *Data Polymorphism Factor*(DPF): This metric attempts to measure how much the polymorphic elements affects the overall complexity.
- *Schemas Reachability Rate* (SRR): This metric measures the fraction of the schemas that are *"hidden"* for schema users such as developers or schema designers. By hidden we mean that main schemas have dependencies on them, but these dependencies are not explicit.

DP metrics attempt to offer a view of complexity that is very relevant to OWS specifications, where the subtyping mechanisms are widely used without a measure of the real impact they have on schemas complexity. These metrics were first introduced in (Tamayo *et al.* 2011b), and are developed in greater depth in the following subsections.

4.1. *Data Polymorphism Rate*

The *Data Polymorphism Rate (DPR)* is a measure of how much polymorphism is contained in the schemas. It calculates the fraction of elements contained in complex type declarations that are *polymorphic*, i.e., its dynamic type may differ from its declared type. It is expressed by the formula:

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

In the formula, N is the total number of complex types, PE_{CTi} is the number of

For Peer Review Only

elements in the declaration of the complex type CT_i that are polymorphic. E_{CTj} is the number of elements in the type declaration of type CT_j. For every type, references to a global elements and 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 the 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 (Abreu and Melo 1996).

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:

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

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 number of 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. The denominator is the same as in the previous metric. Furthermore, in the formula $OE_{CTi} >= E_{CTi}$, where i is a natural number in the interval $[1, N]$, the values of DPF are always equal or greater than 1. This result represents the factor in which the number of elements to be considered may grow when polymorphic elements are taken into account.

Let us consider, for example, the schema fragment in Figure 1. In the example *ContainerType* has two element declarations, and one of them, *containerElement*, of declared type *BaseType*, may have a different dynamic type: *ChildType*. As there are no substitution groups in the schema fragment the value of $OE_{ContainerType} = 2 + 1 = 3$. The value of OE_{CTi} for the other two types is straightforward to calculate as none of them contains polymorphic elements. The metric value for the whole fragment is calculated as follows: $DPF = (1 + 1 + 3)/(1 + 1 + 2) = 1.25$

4.3. *Schemas Reachability Rate*

The last subtyping metric, *Schemas Reachability Rate (SRR)*, attempts to measure the fraction of *imported schema components* that are hidden (not referenced explicitly) by the subtyping mechanisms. 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. But, it can also be referenced *indirectly*, if it is in the substitution group of a referenced element, or it is derived from a type that is referenced directly. For example, **if GML 3.1.1 is imported in a set of schemas where an inner element of type *gml:AbstractFeatureType* is declared, it is not straightforward to realise that**

For Peer Review Only

this element may have 13 different dynamic types (considering only GML types) in XML documents based on such 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 reachable components through explicit and non-explicit dependencies. Using these vertex sets the SRR metric is calculated as follows:

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

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

Figure 2 shows the graph of component relations for the schema fragment in Figure 1. If we ignore the hidden dependency between *ContainerType* and *ChildType* we obtain G_S, otherwise 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:

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

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

For Peer Review Only

Figure 2. Graph of schema component relations for schema fragment in Figure 1

5. Results

The results of applying the previous set of metrics to the OWS specification schemas are shown in the following subsections.

5.1. *XSD-aware Metrics*

In this section we calculate the values of *XSD-aware metrics*, which are those concerned with schema information. The metrics are divided into those that simply count main schema features and C(XSD), which takes the internal structure of components into account to assign weight values to schema components.

5.1.1. Simple Metrics

The first metric analysed in this category is the number of complex types (#CT). 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 (Lämmel *et al.* 2005). The values of the metric for all of the 11 specifications belong to these three ranges, 7 of them are large, two of them is medium-sized and the other two are small schemas. **The specifications related with sensors (SOS, SPS, SensorML, O&M), as well as WCS, exhibit the higher values for the metric (Table 2). It is not a coincidence that they have higher number of dependencies from other specifications. On the other hand, WMS turns out to be the simplest of the specifications being, in terms of #CT, about 20 times smaller than SOS. It might be a little bit surprising that WFS presented such small figures when compared to other web service interface specifications. The reason for this is that WFS schemas do not referenced directly the schemas for the data exchanged between clients and servers. An actual implementation of WFS should include some version of GML; hence if we use for example GML 3.2.1, the combined #CT value would be similar to other large specifications. The same applies for WPS, which is designed to be complemented with application-specific schemas, so its final metric value will depend from the specific implementation.**

Beyond a simple categorisation of the specifications, this metric can be used with different practical purposes. (Lämmel *et al.* 2005) states that #CT is a measure of the number of structured concepts modelled by the schemas, as such, it can be used as a initial estimation of the *conceptual complexity* of a certain specification. This conceptual complexity can be used to compare competing specifications, such as for example GML and KML, which offer some overlapping features. According to this metric GML is far more complex than KML and this difference seems to be growing as GML evolves. If we need support in an application for some of the features included in both specifications, and we have to choose which of them to use, the lower numbers for KML can be considered as an argument in its favour[1].

[1]In a real scenario other aspects must be considered as well, such as available implementations, existing support in different mapping systems, etc.

For Peer Review Only

Moreover, we can use #CT to estimate certain facts about the implementation process or even the final software product. Types are a fundamental concept when schemas are used to write (or generate) XML processing code. Generators typically produces one or more classes for each complex type defined in the schemas. Although some generators produce code for other schema components, complex types have the central role during this mapping. As a consequence, we can use #CT to predict the number of classes that will be contained in the final generated code, which also will allow the estimation of the final size of the application, information that can be very useful in certain applications such as those targeted to resource-constrained environments. If XML processing code is produced manually, as #CT can be considered a size metric, it can be used as part of the resource estimation process (Brito e Abreu and Carapuça 1994).

If we compare the values of #CT with the results presented in (López-Pellicer F. 2010) that analyse the number and distribution of OGC Web services in Europe, the question about if there is a correlation between the complexity of the specifications and its adoption becomes very relevant. If we consider the numbers of available web services presented in (López-Pellicer F. 2010), along with those presented for SOS server instances in (Tamayo *et al.* 2011a), and analyse its correlation to #CT, the value for the *Pearson's correlation coefficient* is -0.57. This value suggests a moderate negative correlation, although a more detailed analysis should be done as we have not considered #CT values for different versions of the service specifications.

A categorisation 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 3 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, WCS, both GML versions, SensorML and O&M), a second group containing medium-sized specifications (WFS, KML), and a last group with 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.

Figure 3. Number of main XML Schema components

The metrics mentioned above have been used in (Tamayo *et al.* 2011a) in the context of an empirical study to measure the percentage of the schema components included in the SOS specification that were actually used by a set of 56 SOS servers available online. These metrics were also used in (Tamayo *et al.* 2011c) to measure the effectiveness of an algorithm to customise the SOS schemas to the needs of individual applications (see Section 7.1).

5.1.2. C(XSD)

C(XSD) is a metric introduced in (Basci and Misra 2009) specifically designed to correct an important flaw of the metrics presented in the previous section: they do not take the internal structure of schema components into account. C(XSD) calculates a weight for each schema component based on its internal structure. These values are then aggregated to calculate an overall complexity value

For Peer Review Only

for the schemas. This result is an approximation of the number of primitive (or atomic) information items that must be considered to fully understand a set of schemas. The metric also considers the influence in complexity of recursive branches, which are counted in terms of the metric value. The weight of each recursive branch is denoted by R, an integer value greater than 1.

The values of this metric for the specifications are much in the same course of the previous metrics (Table 4). A couple of interesting facts deserve more attention, though. For example, the structural complexity of elements in SensorML and O&M is higher than in WCS even when the latter contains more complex types in its definition. SensorML and O&M contain the most complex schema components if analysed individually. The schema component with the higher value for the metric is *Component* with 23016 + 219R. Coincidentally, this element contains the highest number of recursive branches in its definition. Similarly, the structural complexity of KML is similar to the one of GML 3.1.1 even when it contains less than half of its complex types.

About the practical usefulness of C(XSD), it can be used as a complement to other metrics to compare complexity of different specifications. As metrics such as #CT do not take into account the internal structure of these types, schemas with the same #CT value but with distinct internal complexity cannot be distinguished by using this metric in isolation. In this case C(XSD) can spot these differences (see Section 7.2) because it makes visible the internal complexity of individual schema components.

5.2. *Subtyping Mechanisms*

XML Schema subtyping mechanisms were introduced in Section 2.1. In this category we count first the number of abstract elements or types (#AET), the number of substitution groups (#SG) and the number of complex types derived by restriction or extension (#TE and #TR). The values of these metrics are shown in Table 5.

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. An important detail about type derivation by restriction is that it cannot be mapped *smoothly* to an object-oriented programming language (Obasanjo 1995, Dashofy 2001). Hence, specifications that make a large use of this feature may suffer from difficulties when schemas are mapped to these languages, presumably using a code generator. High values of #TR could also interpreted as a potentially flawed design, because if a large number of subtypes requires to be redefined, then their ancestors likely have not been selected/defined in a correct manner.

From the values of #AET in Table 5 the one corresponding to KML stands out from the rest. KML makes a wide use of abstract elements and types. On the other hand, it has a relative low use of substitution groups and it does not use derivation by restriction at all. These values contrast with those of GML, which has lower values for #AET but have higher values for the rest of the metrics in Table 5.

5.2.1. Data Polymorphism Metrics

The values for the DPR metric are presented in Table 6. From these results we can observe that simpler specifications contain zero or a low degree of polymorphism. The rest of the specifications have a similar degree ranging between 12 and 15%. Whether these values are too high or not is not a trivial to say, however in (Abreu and Melo 1996) who

For Peer Review Only

analysed polymorphism in the context of OOP, is stated that a values of POF above 10% are expected to reduce the benefits obtained with an appropriate use of polymorphism. This is because highly polymorphic hierarchies will be harder to understand, debug and maintain.

By themselves DPR values do not show the real effect of data polymorphism in complexity as depending on the number of valid *polymorphic* substitutions the complexity can be perceived as higher or lower. That is the reason why DPF has been defined to quantify all of the possible valid substitutions introduced by the subtyping mechanism. The values of DPF are also shown in Table 6. These results indicate that the effect of polymorphic elements on SOS and SPS is higher than in WFS and WCS. **The values of the metric suggests that the effort needed to consider all of the valid combinations derived from the effect of data polymorphism must be doubled.** As the simplest specifications barely contain polymorphic elements, the values of DPF for them are equal or close to the minimal value, 1. **DPF can be easily adapted to calculate a metric value for individual schema components. This way it can be used, similarly to the case of C(XSD), to detect potential design issues looking for components with disproportionated values for the metric.**

5.2.2. Schemas Reachability Rate

The different values used to calculate the SRR metric, as well as the actual value of this metric are shown in Table 7. 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).

Specifications with higher values of SRR are those having a larger number of dependencies and higher values of DPR and DPF. The consequence of having a large number of polymorphic elements, which in turn can be substituted by a large number of elements or types, is that many of the schema components dependencies are not explicit. This can be the cause of errors or unexpected situations to schema designers and users. Let us consider an example for GML 3.1.1: *gml:AbstractFeatureType* references the global element *gml:name*, which is the head element of a substitution group containing other global elements such as *gml:srsName*, *gml:csName*, *gml:ellipsoidName*, etc. This reference to *gml:name* is inherited by a large set of types which derives, directly or indirectly, from *gml:AbstractFeatureType* such as *gml:PolygonType*. At this point, a polygon instance whose *gml:name* element is substituted by an *gml:srsName* or *gml:ellipsoidName* element is valid against the schemas, although this substitution may not make sense at all.

An early version of this metric inspired the development of the algorithm presented in (Tamayo *et al.* 2011c) to adapt schemas to the specific needs of individual applications. In (Tamayo *et al.* 2011a) it is reported that only 29.5% of the components of the SOS schemas are used in a group of 53 server instances. The cause of this low usage may be that the schemas are perhaps more complex than needed, but may also be that server developers do not fully comprehend all of the relations between schemas components mostly because these relations are not explicit nor easy to discover.

For Peer Review Only

5.3. *Discussion*

The results of the analysis show that at least half of the presented specifications can be considered as large and/or complex according to all of the metrics that provide some sort of categorisation. Most of the metrics coincide in finding a clear diferentiation between a first group containing large specifications (SOS, SPS, WCS, SensorML, O&M, GML), a second group containing medium sized specifications (WPS, WFS and KML), and a third group of simple specifications (WMS). **At first glance, the relation between the complexity of the specifications and its adoption suggests some level of negative correlation. Nevertheless, this topic deserves a deeper analysis as different services versions have not been considered and more sources quantifying the numbers of available services must be also included in the analysis.**

A much more difficult question to answer would be if a cause-effect relationship exists between the complexity of the schemas and its adoption. In our opinion, other aspects such as how useful the specification might be for a community of users, or in the case of competing specifications, how much support for each specification is readily available will have a stronger influence in the adoption of a given specification. Being simpler is not a guarantee to have wider adoption; this is the case for example in the context of schemas languages of XML Schema v. Relax NG (OASIS 2004). The latter is considered simpler and more expressive but still XML Schema has had a larger adoption (Duckett *et al.* 2001, Martens *et al.* 2006, Hosoya 2010).

6. Use of Metrics

The low penetration of software metrics in the software industry has been frequently highlighted (Fenton and Neil 1999, Mens and Demeyer 2001, Kaner *et al.* 2004). This has been attributed to several factors such as much academic research is irrelevant to industrial needs, mainly because academic models often relies in parameters which cannot be measured precisely in practice or they mostly focus on detailed code metrics instead of relevant metrics for process improvement. Another factor is that sometimes it is hard to prove that a metric actually measures the attribute it claims to measure, specially if these attributes are qualitative and subjective in nature, as it is frequently the case for software attributes such as quality, maintainability or reliability. For all of these reasons, we consider pertinent to present more detailed examples of how the metrics presented in this article could be used in practice. Still, the use case scenarios are presented in a succinct way due to space limitations. Specifically we illustrate how metrics aid the evaluation of design decisions and how can be used to follow up the evolution of different versions of the schemas. These use case scenarios provide examples of the use of software metrics in the two different ways introduced in (Mens and Demeyer 2001): *predictive*, before evolution occurs; and *retrospective*, after evolution occurs.

6.1. *Use Case Scenario: Evaluating Design Decisions*

Using metrics in a predictive way can be very helpful to *decision making* during the specification design phase. For example, a typical decision that must be made is *redefine vs. reuse*, when we must choose between reusing components in existing specification

For Peer Review Only

schemas or redefining them. Using appropriate metrics we could have an idea of the effect of one decision or the other in the final size and complexity of the schemas.

For example lets us consider the hypothetical example of analysing how WMS 1.3.0 could be affected if we make it compatible with OWS Common Specification 1.1.0 (OGC 2007b) and GML 3.1.1[1]. With a few simple transformations we could change the *Capabilities* file of this specification to follow the structure defined in OWS common. From Figure 4 we can see that using OWS Common does not make WMS too much bigger, with the added value that support for OWS common can be reutilised from other specifications if it has been already implemented.

In a second step we could try to reuse components from GML 3.1.1 to define WMS layers. In its broadest sense, a *feature* is defined as an abstraction of a real world phenomenon, and a *geographic feature* is a feature associated to a location relative to the Earth. According to this definition we might consider a WMS layer as a geographic feature and as such, we could define it as a subtype of *gml:AbtractFeatureType*, defined on the GML schemas. After modifying the WMS schemas to point to *gml.xsd* and calculating the values of the metrics shown in Figure 4, the addition of GML is not very helpful, as complexity and size of the specification are dramatically raised.

Figure 4. XSD-Aware simple metrics values for WMS 1.3.0 and its merging with OWS 1.0.0 and GML 3.1.1

6.2. *Use Case Scenario: Studying Specifications Evolution*

A valid use of metrics in a retrospective way could be the analysis of schemas evolution for a given specification. Evolution of the schemas may be useful for understanding how and why schemas have grown (or shrunk), to help choosing which version is more appropriate for a given application or to estimate development effort when related web services specifications are implemented. They can also give us some hints about what to expect for the future for a certain specification.

For example let us consider the evolution of GML. Figure 5 shows the values of the metrics #CT, #EL, #ST, #AT, #AG, and #MG[2]. In the figure we can observe the growing trend followed by GML schemas size since its first version. Until the advent of the last version of GML the trend was that subversions corresponding to the same major revision (1.x, 2.x) were similar, but in version 3.x the number of complex types has grown more than 90% from version 3.0.0 to version 3.2.1. These values of the metrics can help to estimate the effort to upgrade from one version of GML to another. It can be very useful to figure out that upgrading a system from using GML 3.1.1 to version 3.2.1 could not be as straightforward as someone might expect based on the premise that the latter is supposed to be a *not-so-large* revision of the former.

Apart from the metrics introduced in previous sections we could use other metrics to explore and understand the changes over different GML versions such as the number of types kept or erased between versions. For example, considering that types are the same

[1]**The purpose of this example is to demonstrate how metrics can be used in similar scenarios, we are not suggesting here that WMS schemas have to be combined with OWS Common or GML schemas.**

[2]GML 1.0.0 is not defined using XML Schema. The metric values shown for that version of the specification are the result of converting from DTD to XML Schema using the XML editor <oXygen/>: http://www.oxygenxml.com

For Peer Review Only

if they have the same name, we can say that 305 types were kept, 153 were erased and 113 new types were introduced in the GML namespace in version 3.2.1 if compared with version 3.1.1. We can refine the metrics to consider the location of types or to use other equality criteria between types if we need more precision in the comparison.

Figure 5. Following GML evolution through metrics

7. Pragmatic Solutions to Complexity

In this section we expose several possible solutions to manage the complexity of the OWS specification schemas. Some of the metrics presented before are used to illustrate the effectiveness of each solution.

7.1. *Profiles*

The use of *profiles*, used here in its broadest sense as *subset of schemas*, is a well-known solution to the schemas complexity problem. Even, a subsetting tool is included with GML 3, to extract subsets of the GML schemas. In addition, a set of standard profiles for GML has been defined such as the Simple Feature Profile (SFP)(OGC 2006a), Common CRS (OGC 2005a) or CRS Support profile (OGC 2005b).

A similar solution to the use of profiles is what we call *'selective importing'*, where only the used schemas of a given specification are imported instead of the whole specification. For example, if we need support for GML features in our schemas, we could import *feature.xsd* directly instead of *gml.xsd*[1].

Following the steps of the GML subsetting tool, (Tamayo *et al.* 2011c) presented an algorithm to extract customised schemas depending on specific application needs. This algorithm uses a set of instance files that must be processed by an application to identify which parts of the schemas are really necessary. Although the example presented there is related to mobile applications based on the SOS specification the algorithm can be applied to other specification schemas as well.

Let us suppose that the portion of SOS schemas in GML 3.1.1 needed in the application in (Tamayo *et al.* 2011c) are all included in SFP and the set of schemas that starts in *feature.xsd*. In this case, we could compare the following approaches according the some of the metrics presented before:

- Use the whole GML schemas (by importing gml.xsd)
- Use the Simple Feature Profile
- Use selective importing (by importing feature.xsd)
- Use the GML subset extracted using the algorithm in (Tamayo *et al.* 2011c)

The result of counting the main schema components in each case is presented in Figure 6. It is obvious from the figure that any of the approaches that try to avoid the use of the whole schemas could reduce considerably the overall schemas complexity. We could also conclude that the use of more specific solutions instead of generic ones could greatly simplify the implementation of real systems.

[1]This is not always possible such as in the case of GML 3.2.1. This version is designed in a way that schemas

For Peer Review Only

Figure 6. Comparing effectiveness of different approaches of using profiles

7.2. *Using the Linked Data Style*

A second pragmatic solution could be the use of the *'linked-data style'*. Linked data is a paradigm that advocates for the publication of data on the Web by making explicit the linkages among related datasets and documents (Berners-Lee 2006). This can be accomplished, among other basic principles, by means of the use of HTTP URIs to not only identify but also to access to data themselves. In this context, the idea behind the 'linked-data style' is based on the use of links instead of embedding directly the data definition. This is particularly possible, because the OGC Naming Authority just changed the resource identification schema to HTTP URIs (Cox 2010).

As pointed out in (Schade *et al.* 2010) since its inception GML has provided a mechanism to implement a linked-style data model. The GML specifications allow users to either use embedded objects or use references **(links)** to external objects. What would be the effect on schemas complexity if we force the use of data references instead of embedding data directly? Let us analyse this in the context of GML 3.1.1. We compare the value of the metric C(XSD) for the original schemas against a modified version of the schemas in which only the linked style is supported. It should be noticed that forbidding the embedded style does not change the number of global components in the specifications, for this reason we have not used any of the metrics based on counting these items.

Figure 7 shows the individuals values of C(XSD) for every component in GML 3.1.1 for both the original and the modified schemas. The x-axis represents the **entire set** of GML global schema components numbered starting from 1. The y-axis shows the value of C(XSD) for both the original schemas and the modified ones. **The overall value of C(XSD) for the full schemas is 74,609 + 611R. The overall value for the schemas following the reference data style *(GML Linked Profile)* has been reduced to 18,731 + 13R. If we give a weight of 2 (R=2) to recursive branches, this signifies a reduction of more than 75% of the complexity of the original schemas.**

The scenario presented here is an extreme one, as all the embedded data has been removed from the schemas. But, in a real implementation other aspects apart from schemas complexity must be considered. In the case above, removing the chance of embedding data will increase the number of messages exchanged on the network as each piece of information must be requested separately by following the links provided by other documents. Nevertheless, this does not necessarily imply that we will have a higher network traffic, as we could follow these links only if we needed, and we will not obliged to read all of the information as in the case it is embedded in other documents. In any case, in a real implementation different trade-offs could be made about what to link and what to embed in order to satisfy specific application requirements.

Figure 7. C(XSD) values for GML and the GML Linked Profile

using GML schemas can only be validated correctly if they import *gml.xsd*

8. Conclusions

In this paper we have presented a quantitative way to analyse and measure the complexity of OWS schemas. The use of adequate metrics allows us to quantify the complexity and other properties of the schemas. The results of the analysis show that at least half of the presented specifications can be considered as large and/or complex. **For all of the metrics we have tried to present simple examples of potential applications to practical situations problems, as well as references to specific scenarios where some of these metrics have been used with different purposes such as measuring the percentage of schema components used by actual implementation or measuring the effectiveness of an algorithm that reduces the schemas complexity.**

A set of new metrics have been introduced to show the effect of the use of the subtyping mechanisms in complexity. **This was motivated by the high use of these mechanisms in the OWS schemas. These new metrics try to complement the other metrics included in our study as a single metric cannot be used to measure all of the aspects that influence specifications complexity. The metrics show that for the most complex specifications, the effort needed to consider all of the valid combinations derived from the effect of data polymorphism must be doubled. They** have also shown that more than 60% of the schema components included in those specifications are referenced in ways that cannot be seen explicitly, augmenting the risk of making mistakes while working with the schemas.

We have also presented pragmatic solutions to complexity. In our opinion, the solution related to the use of the linked-data style suggests that a simple paradigm shift may reduce the complexity substantially, which indicates that much of this complexity is there because of design designs and not because of the complexity of the problem domain.

As future work we are working in analysing in more detail how the linked-data style can be combined with the Representational State Transfer (REST) architectural style (Fielding 2000) to reduce complexity of geospatial web service implementations. We are also considering the use of alternatives schema languages to define the structure of geospatial data, such as RelaxNG (OASIS 2004), and its impact in specifications complexity.

Acknowledgement(s)

This work has been partially supported by the "España Virtual" project (ref. CENIT 2008-1030) through the Instituto Geográfico Nacional (IGN). The authors thank Sven Schade for the lively comments.

References

Abreu, F.B. and Melo, W., 1996. Evaluating the Impact of Object-Oriented Design on Software Quality. *IEEE Int. Symposium on Software Metrics*, 0, 90.

Barbosa, D., Mignet, L., and Veltri, P., 2005. Studying the XML Web: Gathering Statistics from an XML Sample. *World Wide Web*, 8, 413–438.

Basci, D. and Misra, S., 2009. Measuring and evaluating a design complexity metric

For Peer Review Only

for XML schema documents. *Journal of Information Science and Engineering*, 25 (5), 1405–1425.

Berners-Lee, T., 2006. Linked Data. [online] Available from: http://www.w3.org/DesignIssues/LinkedData.html [Accessed 15 March 2011].

Bray, T., 2003. XML Is Too Hard For Programmers. [online] Available from: `http://www.tbray.org/ongoing/When/200x/2003/03/16/XML-Prog` [Accessed 15 March 2011].

Brito e Abreu, F. and Carapuça, R., 1994. Candidate metrics for object-oriented software within a taxonomy framework. *J. Syst. Softw.*, 26, 87–96.

Chidamber, S.R. and Kemerer, C.F., 1994. A Metrics Suite for Object Oriented Design. *IEEE Trans. Softw. Eng.*, 20, 476–493.

Cox, S., 2010. OGC Identifiers ? the case for http URIs. *OGC White Paper*.

Dashofy, E.M., 2001. Issues in Generating Data Bindings for an XML Schema-Based Language. *In*: *In Proceedings of the Workshop on XML Technologies and Software Engineering (XSE2001*.

Duckett, J., Cagle, K., and Williams, K., 2001. *Professional XML Schemas*. Birmingham, UK, UK: Wrox Press Ltd.

Fenton, N.E. and Neil, M., 1999. Software metrics: success, failures and new directions. *Journal of Systems and Software*, 47, 149–157.

Fielding, R.T., 2000. Architectural Styles and the Design of Network-based Software Architectures. Thesis (PhD). University of California, Irvine, Irvine, California.

Fu, P. and Sun, J., 2010. *Web GIS: Principles and Applications*. ESRI Press.

Goodchild, M.F., Yuan, M., and Cova, T.J., 2007. Towards a general theory of geographic representation in GIS. *Int. J. Geogr. Inf. Sci.*, 21, 239–260.

Hosoya, H., 2010. *Foundations of XML Processing: The Tree-Automata Approach*. The Edinburgh Building, Cambridge CB2 8RU, UK: Cambridge University Press.

Kaner, C., Member, S., and Bond, W.P., 2004. Software Engineering Metrics: What Do They Measure and How Do We Know?. *In*: *In METRICS 2004. IEEE CS* Press.

Lämmel, R., Kitsis, S., and Remy, D., 2005. Analysis of XML schema usage. *In*: *Proceedings of XML 2005*, 1–35.

López-Pellicer F., Béjar-Hernández R., F.A.M.M.P.Z.S., 2010. State of Play of OGC Web Services across the Web. *In*: *Proceedings of INSPIRE Conference 2010: INSPIRE as a framework for cooperation, Krakow, Poland*.

Martens, W., *et al.*, 2006. Expressiveness and complexity of XML Schema. *ACM Trans. Database Syst.*, 31, 770–813.

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

McDowell A., Schmidt C., Y.K., 2005. Analysis and metrics of XML schema. *In*: *Proceedings of Intl Conference on Software Engineering Research and Practice*, 538–544.

Mens, T. and Demeyer, S., 2001. Future trends in software evolution metrics. *In*: *Proceedings of the 4th International Workshop on Principles of Software Evolution*, IWPSE '01, Vienna, Austria New York, NY, USA: ACM, 83–86.

Mller, A. and Schwartzbach, M.I., 2006. *An Introduction to Xml And Web Technologies*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

OASIS, 2004. RELAX NG Specification. [online] Available from: http://relaxng.org/spec-20011203.html [Accessed 15 March 2011].

Obasanjo, D., 1995. Why You Should Very Carefully Use Restriction Of Complex Types. [online] Available from: http://www.xml.com/pub/a/2002/11/20/schemas.html?page=4#restriction [Accessed 15 March 2011].

For Peer Review Only

OGC, 2004. OpenGIS Geography Markup Language (GML) Encoding Specification 3.1.1. *OGC Document*, (03-105r1).

OGC, 2005a. GML 3.1.1 common CRSs profile. *OGC Document*, (05-095r1).

OGC, 2005b. GML 3.1.1 CRS support profile. *OGC Document*, (05-094r1).

OGC, 2006a. Geography Markup Language (GML) simple features profile. *OGC Document*, (06-049r1).

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

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

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

OGC, 2007c. OpenGIS Geography Markup Language (GML) Encoding Standard 3.2.1. *OGC Document*, (07-036).

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

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

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

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

OGC, 2008. OGC KML 2.2.0. *OGC Document*, (07-147r2).

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

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

Pichler, C., Strommer, M., and Huemer, C., 2010. Size Matters!? Measuring the Complexity of XML Schema Mapping Models. *Services, IEEE Congress on*, 0, 497–502.

Qureshi, M.H. and Samadzadeh, M.H., 2005. Determining the Complexity of XML Documents. *In*: *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II - Volume 02*, ITCC '05 Washington, DC, USA: IEEE Computer Society, 416–421.

Schade, S., Granell, C., and Díaz, L., 2010. Augmenting SDI with Linked Data. *In*: *Proceedings of the Workshop on Linked Spatiotemporal Data 2010 (LSTD 2010)*, Zurich, Switzerland.

Tamayo, A., Granell, C., and Huerta, J., 2011a. Empirical Study of SOS Server Instances. *In*: *Proceedings of 14th AGILE International Conference on Geographic Information Science*, Utrecht, The Netherlands Springer.

Tamayo, A., Granell, C., and Huerta, J., 2011b. Analysing Complexity of XML Schemas in Geospatial Web Services. *In*: *COM.Geo 2011, 2nd International Conference and Exhibition on Computing for Geospatial Research and Application*, Washinton DC, USA ACM.

Tamayo, A., Granell, C., and Huerta, J., 2011c. Dealing with Large Schema Sets in mobile SOS-Based Applications. *In*: *COM.Geo 2011, 2nd International Conference and Exhibition on Computing for Geospatial Research and Application*, Washinton DC, USA ACM.

Visser, J., 2006. Structure metrics for XML Schema. *In*: *Proceedings of XATA 2006*, 236–247.

W3C, 2004a. XML Schema Part 1: Structures Second Ed.. [online] Available from: http://www.w3.org/TR/xmlschema-1 [Accessed 15 March 2011a].

W3C, 2004b. XML Schema Part 2: Datatypes Second Ed.. [online] Available from:

For Peer Review Only

http://www.w3.org/TR/xmlschema-2 [Accessed 15 March 2011b].

For Peer Review Only

Tables

Table 1. Geospatial web service interfaces

Name	Description	Versions
Web Map Service (WMS)	WMS provides a simple HTTP interface for requesting geo-registered map images from one or more distributed geospatial databases (OGC 2006b)	1.3.0
Web Feature Service (WFS)	It allows a client to retrieve and update geospatial data encoded in GML format (OGC 2010b)	1.1.0 (2.0)
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 (OGC 2010a)	1.1.2 (2.0)
Sensor Observation Service (SOS)	It provides an API retrieving sensor and observation data (OGC 2007g)	1.0.0
Web Processing Service (WPS)	It defines a standardised interface to publish geospatial processes (OGC 2007f)	1.0.0
Sensor Planning Service (SPS)	It defines interfaces for queries that provide information about the capabilities of a sensor and how to task the sensor(OGC 2007e)	1.0.0
Geography Markup Language (GML)	GML is a grammar for expressing geographical features. It serves as a modelling language systems as well as an interchange format (OGC 2004, 2007c)	3.1.1 , 3.2.1
Sensor Model Language (SensorML)	SensorML is a language that specifies models and encodings that provide a framework within which characteristics of sensors and sensor systems can be defined (OGC 2007d)	1.0.1
Observation and Measurements (O&M)	O&M defines an abstract model and Sensor Model Language schema encoding for observations (OGC 2007a)	1.0.0
Keyhole Markup Language (KML)	KML is a language focused on geographic visualisation, including annotation of maps and images (OGC 2008)	2.2.0

For Peer Review Only

Table 2. Number of Complex Types (#CT)

	SOS 1.0	WFS 2.0	WCS 2.0	SPS 1.0	WPS 1.0	WMS 1.3.0	GML 3.2.1	GML 3.1.1	SensorML 1.0.1	O&M 1.0	KML 2.2.0
#CT	740	163	797	585	99	33	654	394	610	615	153

For Peer Review Only

Table 3. Main XML features metrics (except #CT)

	SOS 1.0	WFS 2.0	WCS 2.0	SPS 1.0	WPS 1.0	WMS 1.3.0	GML 3.2.1	GML 3.1.1	SensorML 1.0.1	O&M 1.0	KML 2.2.0
#ST	118	46	74	103	15	5	70	64	100	100	42
#EL	727	156	754	593	64	60	653	485	586	588	292
#MG	28	3	14	19	7	0	7	12	26	26	0
#AT	23	15	20	15	16	9	17	15	33	33	0
#AG	40	12	17	37	9	7	39	35	39	39	2
#ALL	1498	354	1,625	1,150	174	80	1,414	986	1,249	1,256	441

For Peer Review Only

Table 4. C(XSD) values for OWS specifications

	SOS 1.0	WFS 2.0	WCS 2.0	SPS 1.0	WPS 1.0	WMS 1.3.0	GML 3.2.1	GML 3.1.1	SensorML 1.0.1	O&M 1.0	KML 2.2.0
C_{XSD}	261,238 + 2,381R	1,960 + 16R	209,997 + 1,171R	96,451 + 885R	1,578 + 2R	707 + 3R	150,094 + 839R	74,609 + 611R	244,827 + 2,267R	233,194 + 2,137R	74,940 + 614R

For Peer Review Only

Table 5. Use of subtyping mechanisms

	SOS 1.0	WFS 2.0	WCS 2.0	SPS 1.0	WPS 1.0	WMS 1.3.0	GML 3.2.1	GML 3.1.1	SensorML 1.0.1	O&M 1.0	KML 2.2.0
#AET	61	15	74	52	2	2	63	47	53	53	124
#SG	83	11	123	72	4	0	112	60	72	72	10
#TE	291	55	356	243	30	4	300	182	241	243	55
#TR	59	1	15	54	1	0	13	53	57	57	0

For Peer Review Only

Table 6. **DPR and DPF values for the OWS specifications**

	SOS 1.0	WFS 2.0	WCS 2.0	SPS 1.0	WPS 1.0	WMS 1.3.0	GML 3.2.1	GML 3.1.1	SensorML 1.0.1	O&M 1.0	KML 2.2.0
DPR	0.13	0.12	0.15	0.13	0.05	0	0.15	0.14	0.13	0.13	0.05
DPF	2.20	1.47	1.48	2.20	1.05	1	1.40	2.17	2.15	2.16	1.13

For Peer Review Only

Table 7. SRR values for the OWS specifications

	SOS 1.0	WFS 2.0	WCS 2.0	SPS 1.0	WPS 1.0	WMS 1.3.0	GML 3.2.1	GML 3.1.1	SensorML 1.0.1	O&M 1.0	KML 2.2.0
$\|V_{Rm}(G_{SH})\|$	1277	321	1349	1058	146	71	1334	975	1070	1076	398
$\|V_{Rm}(G_S)\|$	319	245	220	203	126	71	1033	975	325	180	398
$\|V_S\|$	1498	353	1625	1266	179	80	1414	986	1249	1256	399
SRR	0.64	0.22	0.69	0.68	0.11	0	0.21	0	0.59	0.71	0

For Peer Review Only

List of Figures

For Peer Review Only

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:complexType name="BaseType">
        <xs:sequence>
            <xs:element name="baseElement" type="xs:string" minOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:string" use="required"/>
    </xs:complexType>

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

    <xs:complexType name="ContainerType">
        <xs:sequence>
            <xs:element name="containerElement" type="BaseType" maxOccurs="unbounded"/>
            <xs:element name="recursiveElement" type="ContainerType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

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

Figure 1 : XML Schema file fragment
59x32mm (600 x 600 DPI)

Review Only

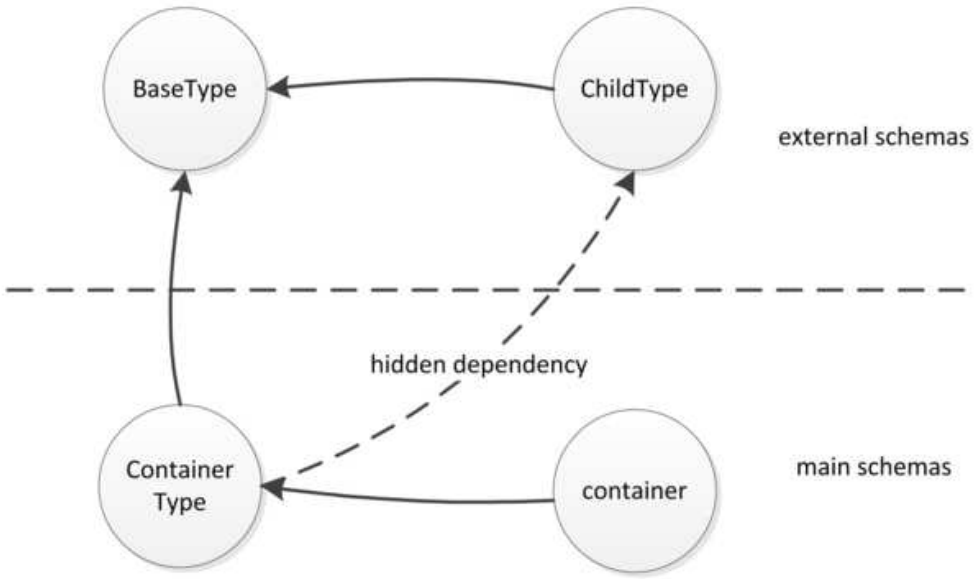

Figure 2 : Graph of schema component relations for schema fragment in Figure 1
51x33mm (300 x 300 DPI)

For Review Only

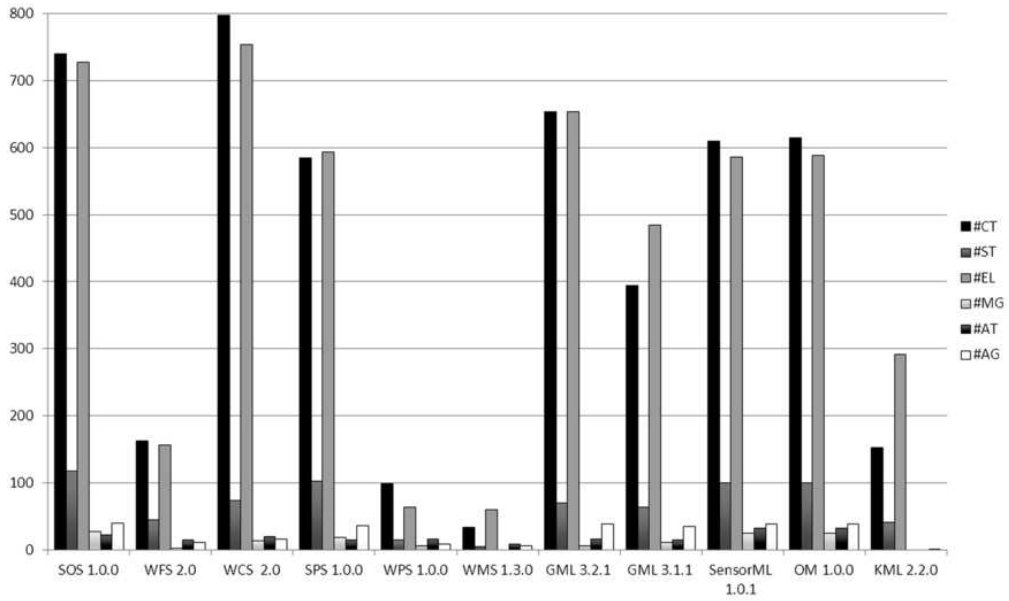

Figure 3: Number of main XML Schema components
73x44mm (300 x 300 DPI)

For Review Only

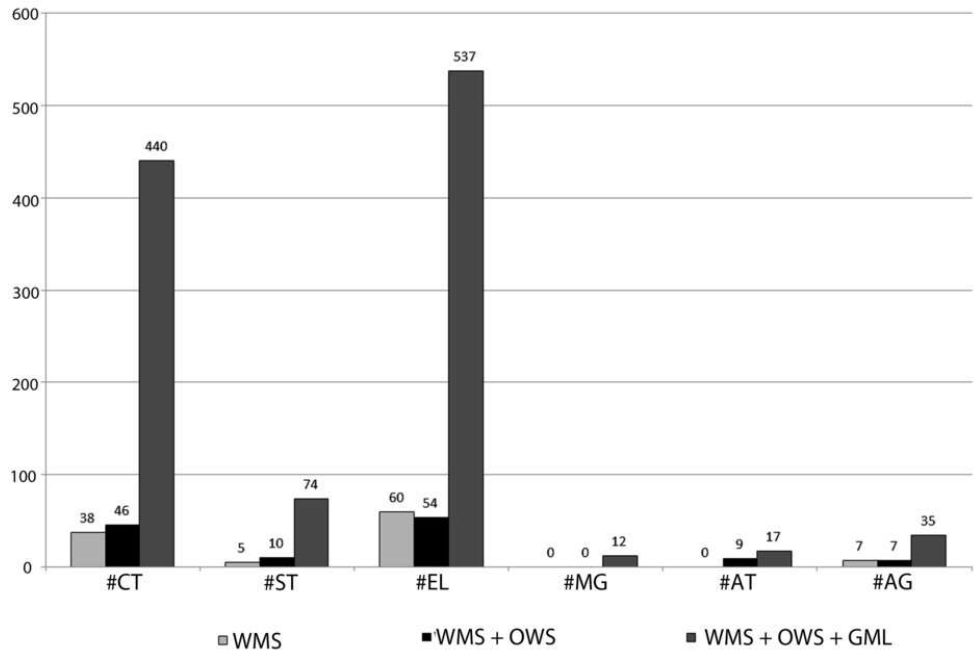

Figure 4: XSD-Aware simple metrics values for WMS 1.3.0 and its merging with OWS Common 1.1.0 and GML 3.1.1
81x54mm (300 x 300 DPI)

For Review Only

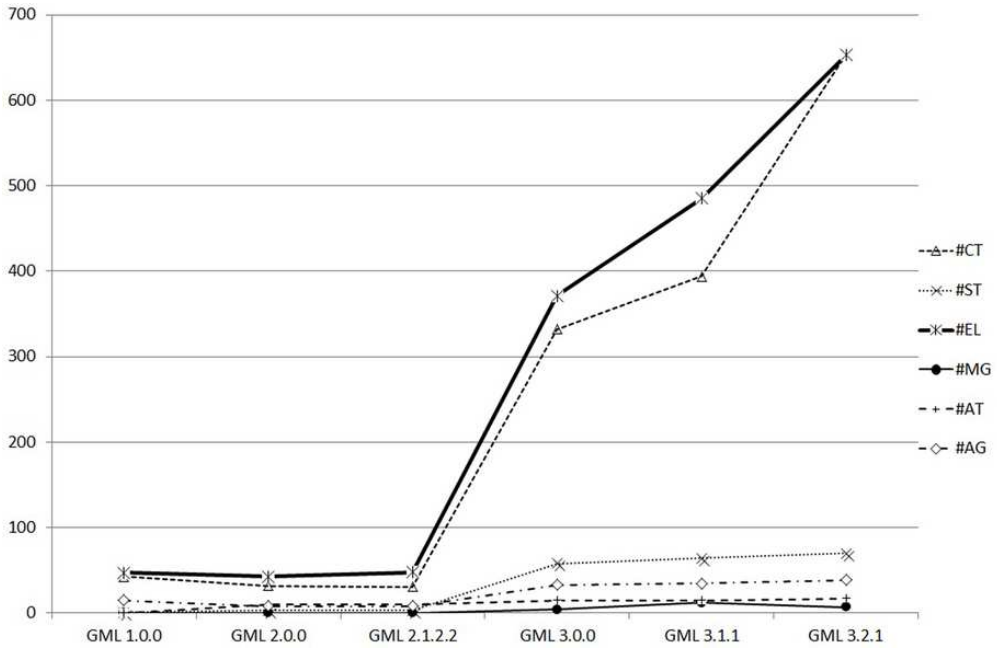

Figure 5 : Following GML evolution through metrics
77x49mm (300 x 300 DPI)

For Review Only

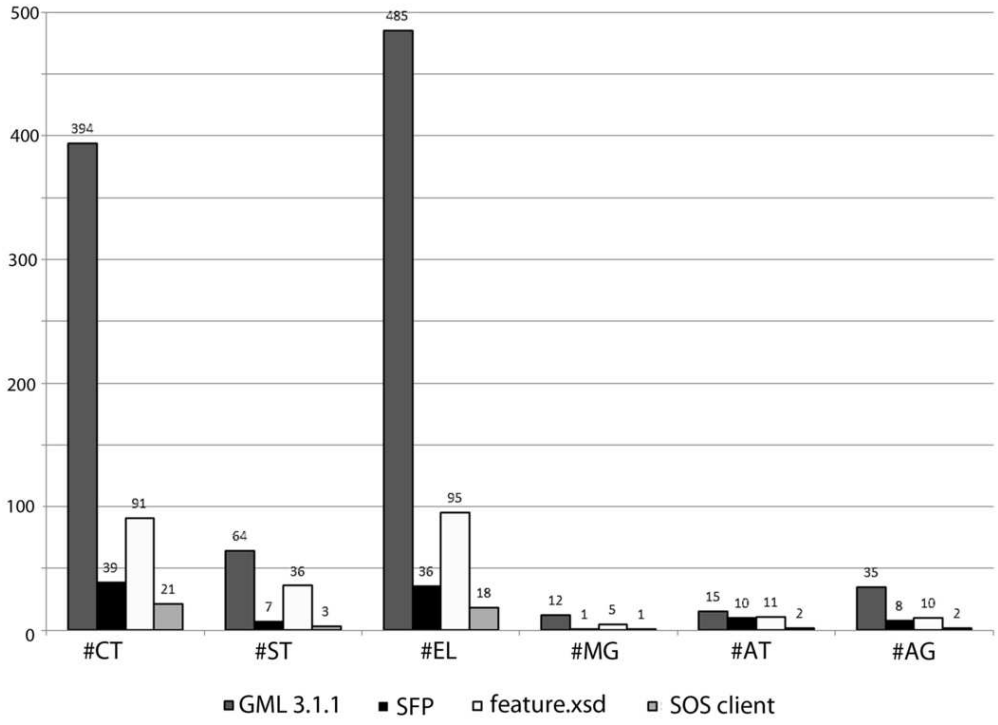

Figure 6 : Comparing effectiveness of different approaches of using profiles
89x65mm (300 x 300 DPI)

For Review Only

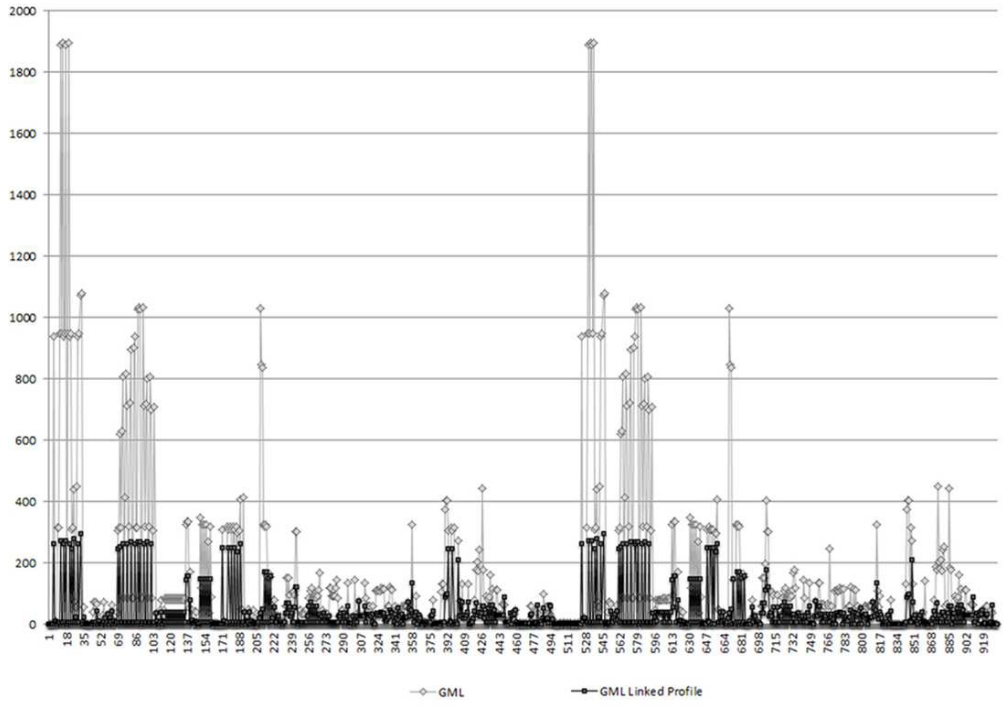

Figure 7 : C(XSD) values for GML and the GML Linked Profile
92x65mm (300 x 300 DPI)

view Only