International Journal of Geographical Information Science

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tgis20

Measuring complexity in OGC web services XML schemas: pragmatic use and solutions

Alain Tamayo a, Carlos Granell b & Joaquín Huerta a

a Institute of New Imaging Technologies, Universitat Jaume I, Castellón de la Plana, Spain
b Institute for Environment and Sustainability, European Commission Joint Research Centre, Ispra, Italy

Available online: 11 Jan 2012


To link to this article: http://dx.doi.org/10.1080/13658816.2011.626602

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.tandfonline.com/page/terms-and-conditions

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae, and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand, or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.
Measuring complexity in OGC web services XML schemas: pragmatic use and solutions

Alain Tamayo*, Carlos Granell and Joaquín Huerta

*Institute of New Imaging Technologies, Universitat Jaume I, Castellón de la Plana, Spain;
†Institute for Environment and Sustainability, European Commission Joint Research Centre, Ispra, Italy

(Received 6 April 2011; final version received 18 September 2011)

The use of standards in the geospatial domain, such as those defined by the Open Geospatial Consortium (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 with the OGC implementation specifications. We use a comprehensive set of metrics to provide a multidimensional view of this complexity. These metrics can be used to evaluate the impact of design decisions, study the evolution of schemas, and so on. We also present and evaluate different solutions that could be applied to overcome some of the problems associated with the complexity of the schemas.

Keywords: XML schema; OGC web services; complexity analysis; software metrics; standards

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.


*Corresponding author. Email: atamayo@uji.es

ISSN 1365-8816 print/ISSN 1362-3087 online
© 2012 Taylor & Francis
http://dx.doi.org/10.1080/13658816.2011.626602
http://www.tandfonline.com
Complexity in geospatial web services comes mostly from the complexity of the problem domain. As stated by 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. Fu and Sun (2010) highlighted the need for easier standards. According to these authors, easier standards get wider adoption such as the case of GeoRSS\textsuperscript{1} and web map service (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 (OWS) specifications as they define the structure of exchanged messages and data. The schemas, defined using the XML schema language (W3C 2004a, 2004b), are in most cases the only machine-processable information associated to the specifications. They can be used with different purposes, such as evaluating whether 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 service 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. (2011b) presented 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 is measured using a set of metrics extracted from the literature (Lämmel et al. 2005, McDowell et al. 2004, Basci and Misra 2009), as well as a set of metrics defined by ourselves. The main goal of this study is to provide a multidimensional view of the complexity of the schemas, as the use of a single metric cannot comprise all the aspects that influence their perceived complexity (Mens and Demeyer 2001, Kaner and Bond 2004). 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 and so on.

This article extends the work presented by Tamayo et al. (2011a), considering a wider set of OWS specifications and a discussion on the pragmatic uses of these metrics. The remainder of this article is structured as follows: Section 2 briefly introduces the XML schema language and the OGC specifications considered in this article. Section 3 presents related work in the subject. Sections 4 and 5 expose the metrics included in the complexity study and the results of calculating their values. Sections 6 and 7 present pragmatic aspects related to the metrics such as practical use and solutions to deal with complexity. Finally, Section 8 presents the conclusions and future work.
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.

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 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).

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 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
<?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 base="BaseType">
      <xs:extension>
        <xs:element name="childElement" type="xs:string" />
      </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:schema>
```

Figure 1. Fragment of an XML schema document.
XML schema is considered a very complex language. Martens et al. (2006) stated that the complexity of 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 this language over its predecessor, Document Type Definition (DTD), is only used to a very limited extent. Møller and Schwartzbach (2006) presented 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. Finally, Hosoya (2010) attributed the complexity of the schema language to the attempt of mixing two completely different notions: regular expressions and object orientation. Regular expressions were 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 design of specifications, they are built reutilising components defined by other specifications. 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 by the main specification schemas.

3. Related work

Literature about measuring the complexity of XML schemas 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 was presented by 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 features of XML schema, although more complex metrics such as the McCabe’s cyclomatic complexity (McCabe 1976) were adapted to measure complexity of the schemas. Another relevant study was presented by McDowell et al. (2004), defining 14 metrics to measure the quality and complexity of XML schemas. The authors included five of the metrics also considered by 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.

Basci and Misra (2009) presented a more sophisticated metric that takes into account not only the number of main schema components like the previous mentioned work, but also the internal structure of these components. A weight is assigned to every schema
Table 1. Geospatial web service interfaces.

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

Note: WMS, web map service; WFS, web feature service; WCS, web coverage service; SOS, sensor observation service; WPS, web processing service; SPS, sensor planning service; GML, geography markup language; SensorML, sensor markup language; O&M, observation and measurements; KML, keyhole markup language.

component based on the weight of its inner components. These metrics also recognise recursive structures as a feature that increases the complexity of schemas.

In a similar way, Visser (2006) proposed more advanced schema metrics, arguing that previous work in the topic only measures 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 \textit{fan-in} and \textit{fan-out}, to measure the number of incoming and outgoing edges from a graph representing schema components as nodes, are very similar to the \textit{element fanning} metrics presented by McDowell \textit{et al.} (2004).

Last about the complexity of schemas, Pichler \textit{et al.} (2010) presented a set of metrics in the context of schema mapping. The authors defined a combined metric based on a set of metrics considering the size of schemas, the use of different schema features and the naming strategies. The combined metric was evaluated in the context of business document standards.

Similar studies in the geospatial domain are scarce, although an interesting discussion of complexity in the context of the Geographic Markup Language (GML) can be found in
Ron Lake’s blog\textsuperscript{2}. This discussion tries to identify the origin of GML complexity and uses some of the metrics introduced by 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 criterion to select the metrics is that they be intuitively understood, that is, that we could easily understand what we are measuring, in contrast to the use of esoteric or artificial metrics, which end up being of little interest to the industry (Fenton and Neil 1999).

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

- **Number of complex types** ($\#CT$): All complex types, including global ($\#CT_G$) and anonymous ($\#CT_A$) complex types (Lämmel et al. 2005, McDowell et al. 2004).
- **Number of simple types** ($\#ST$): All simple types, including global ($\#ST_G$) and anonymous ($\#ST_A$) simple types (Lämmel et al. 2005, McDowell et al. 2004).
- **Number of global elements** ($\#EL$): All global element declarations (Lämmel et al. 2005).
- **Number of global model groups** ($\#MG$): All global model group 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 group definitions (Lämmel et al. 2005).
- **$C(XSD)$**: This metric calculates a complexity weight taking into account the internal structure of schema components. 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 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 the basic metrics that count the number of times certain features are used such as substitution groups, specialisation by restriction and specialisation by extension (Lämmel et al. 2005, McDowell et al. 2004).
- **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 affect the overall complexity.
• *Schema reachability rate* (SRR): This metric measures the fraction of the schemas that are *hidden* from schema users such as developers or schema designers. By hidden we mean that the 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 subtyping mechanisms are widely used without a measure of their real impact on the complexity of the schemas. These metrics were first introduced by Tamayo *et al.* (2011a) and are developed in greater depth in the following sections.

### 4.1. Data polymorphism rate

The *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*, that is, its dynamic type may differ from its declared type. It is expressed by the following formula:

\[
DPR = \frac{\sum_{i=1}^{N} PECT_i}{\sum_{j=1}^{N} ECT_j}
\]

In the formula, \(N\) is the total number of complex types and \(PECT_i\) is the number of elements in the declaration of the complex type \(CT_i\) that are polymorphic. \(ECT_j\) is the number of elements in the type declaration of type \(CT_j\). For every type, references to global-element and inner-element declarations are considered to be equal 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 on all the complex types in the schemas. The metric values are 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 in the schemas, but does not measure their influence in the overall complexity of the schemas. 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 20 different elements. In this regard, we define the *DPF* as follows:

\[
DPF = \frac{\sum_{i=1}^{N} OECT_i}{\sum_{j=1}^{N} ECT_j}
\]

In this case, \(OECT_i\) 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 \(CT_i\) that differs from its declared type. The
denominator is the same as in the previous metric. Furthermore, in the formula $OE_{CTi} \geq E_{CTi}$, where $i$ is a natural number in the interval $[1, N]$, as a consequence the values of DPF are always equal to 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. Schema reachability rate

The last subtyping metric, $SRR$, 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 another 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 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 4.1.** 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 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 4.2.** 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, that is an edge from $v_i$ to $v_j$ exists if $v_j$ is used somehow in the declaration of $v_i$, 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 4.3.** 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 $G_S$, these will be reachable components 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 with the following formula. Bars in the formula denote the cardinalities of the sets.

$$SRR = \frac{|V_{Rm}(G_{SH})| - |V_{Rm}(G_S)|}{|V_S|}$$
Figure 2. Graph of schema components for the schema fragment in Figure 1.

The metric measures the fraction of schema components that can be used in documents conforming to the main schemas but are not explicitly referenced from any component in the main schemas, or any component reachable through explicit dependencies.

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 that BaseType and ChildType declarations are located in external schemas, we could calculate the value of SRR for the main schemas: $V_m = \{\text{container}, \text{ContainerType}\}$, $V_{Rm} (G_S) = \{\text{container}, \text{ContainerType}, \text{BaseType}\}$, $V_{Rm} (G_{SH}) = \{\text{container}, \text{ContainerType}, \text{BaseType}, \text{ChildType}\}$ as follows:

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

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

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

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 weights to schema components.

5.1.1. Simple metrics
The first metric analysed in this category is #CT. Schemas with #CT in the range 256–1000 are considered large, in the range 100–256 are considered medium and small
Table 2. Number of complex types (#CT).

<table>
<thead>
<tr>
<th></th>
<th>SOS</th>
<th>WFS</th>
<th>WCS</th>
<th>SPS</th>
<th>WPS</th>
<th>WMS</th>
<th>GML</th>
<th>GML SensorML</th>
<th>O&amp;M</th>
<th>KML</th>
</tr>
</thead>
<tbody>
<tr>
<td>#CT</td>
<td>740</td>
<td>163</td>
<td>797</td>
<td>585</td>
<td>99</td>
<td>33</td>
<td>654</td>
<td>394</td>
<td>610</td>
<td>615</td>
</tr>
</tbody>
</table>

Note: WMS, web map service; WFS, web feature service; WCS, web coverage service; SOS, sensor observation service; WPS, web processing service; SPS, sensor planning service; GML, geography markup language; SensorML, sensor markup language; O&M, observation and measurements; KML, keyhole markup language.

with #CT between 32 and 100 (Lämmel et al. 2005). The values of the metric for all the 11 specifications belong to these three ranges, 7 of them are large, 2 of them are medium-sized and the other 2 are small schemas. The specifications related to sensors (SOS, sensor planning service (SPS), sensor markup language (SensorML), observation and measurements (O&M)), as well as web coverage service (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 web feature service (WFS) presented such small figures when compared to other web service specifications. The reason for this is that WFS schemas do not refer 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 value of #CT would be similar to other large specifications. The same applies for web processing service (WPS), which is designed to be complemented by application-specific schemas, so its final metric value will depend on the specific implementation.

Beyond a simple categorisation of the specifications, this metric can be used with different practical purposes. Lämmel et al. (2005) stated that #CT is a measure of the number of structured concepts modelled by the schemas. Therefore, we can use it as an initial estimation of the conceptual complexity of a specification. This conceptual complexity can be used to compare competing specifications, such as, for example, GML and keyhole markup language (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 shown by KML can be considered as an argument in its favour.

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 produce 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 is possible to use it as part of the resource estimation process (Abreu and Carapuça 1994).

If we compare the values of #CT with the results presented by López-Pellicer et al. (2010) that analyse the number and distribution of OWS in Europe, the question whether
Table 3. Main XML features metrics (except #CT).

<table>
<thead>
<tr>
<th></th>
<th>SOS 1.0</th>
<th>WFS 2.0</th>
<th>WCS 2.0</th>
<th>SPS 1.0</th>
<th>WPS 1.0</th>
<th>WMS 1.3.0</th>
<th>GML 3.2.1</th>
<th>SensorML 3.1.1</th>
<th>O&amp;M 1.0</th>
<th>KML 2.2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>#ST</td>
<td>118</td>
<td>46</td>
<td>74</td>
<td>103</td>
<td>15</td>
<td>5</td>
<td>70</td>
<td>64</td>
<td>100</td>
<td>42</td>
</tr>
<tr>
<td>#EL</td>
<td>727</td>
<td>156</td>
<td>754</td>
<td>593</td>
<td>64</td>
<td>60</td>
<td>653</td>
<td>485</td>
<td>586</td>
<td>588</td>
</tr>
<tr>
<td>#MG</td>
<td>28</td>
<td>3</td>
<td>14</td>
<td>19</td>
<td>7</td>
<td>0</td>
<td>7</td>
<td>12</td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>#AT</td>
<td>23</td>
<td>15</td>
<td>20</td>
<td>15</td>
<td>9</td>
<td>17</td>
<td>15</td>
<td>33</td>
<td>33</td>
<td>0</td>
</tr>
<tr>
<td>#AG</td>
<td>40</td>
<td>12</td>
<td>17</td>
<td>37</td>
<td>9</td>
<td>7</td>
<td>39</td>
<td>35</td>
<td>39</td>
<td>2</td>
</tr>
<tr>
<td>#ALL</td>
<td>1498</td>
<td>354</td>
<td>1625</td>
<td>1150</td>
<td>174</td>
<td>80</td>
<td>1414</td>
<td>986</td>
<td>1249</td>
<td>1256</td>
</tr>
</tbody>
</table>

Note: WMS, web map service; WFS, web feature service; WCS, web coverage service; SOS, sensor observation service; WPS, web processing service; SPS, sensor planning service; GML, geography markup language; SensorML, sensor markup language; O&M, observation and Measurements; KML, keyhole markup language.

there is a correlation between the complexity of the specifications and its adoption becomes very relevant. Considering the numbers of OWS in Europe, along with those presented for SOS server instances by Tamayo et al. (2011c), and analysing 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 the size of the schemas 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 the specifications.

The metrics mentioned above have been used by Tamayo et al. (2011c) in the context of an empirical study to measure the percentage of schema components included in the SOS specification that were actually used by 56 servers available online. These metrics were also used by Tamayo et al. (2011b) 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) was designed to correct an important flaw of the metrics presented in Section 5.1.1: 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 the schemas. The 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. 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). However, a couple of interesting facts deserve more attention.
Table 4. C(XSD) values for OWS specifications.

<table>
<thead>
<tr>
<th>OWS Specification</th>
<th>SOS 1.0</th>
<th>WFS 2.0</th>
<th>WCS 2.0</th>
<th>SPS 1.0</th>
<th>WPS 1.0</th>
<th>WMS 1.3.0</th>
<th>GML 3.2.1</th>
<th>GML 3.1.1</th>
<th>SensorML 1.0</th>
<th>O&amp;M 1.0</th>
<th>KML 2.2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>C(XSD)</td>
<td>261,238</td>
<td>1,960</td>
<td>209,997</td>
<td>96,451</td>
<td>1,578</td>
<td>707</td>
<td>150,094</td>
<td>74,609</td>
<td>244,827</td>
<td>233,194</td>
<td>74,940</td>
</tr>
<tr>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>+</td>
<td>2,381R</td>
<td>16R</td>
<td>1,171R</td>
<td>885R</td>
<td>2R</td>
<td>839R</td>
<td>611R</td>
<td>2,267R</td>
<td>2,137R</td>
<td>614R</td>
<td></td>
</tr>
</tbody>
</table>

Note: OWS, OGC web services; WMS, web map service; WFS, web feature service; WCS, web coverage service; SOS, sensor observation service; WPS, web processing service; SPS, sensor planning service; GML, geography markup language; SensorML, sensor markup language; O&M, observation and Measurements; KML, keyhole markup language.

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 sml:Component\(^5\) 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.

From a practical perspective, C(XSD) can be used as a complement to other metrics to compare the 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 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 first we count the number of abstract elements or types (#AET), the number of substitution
Table 5. Use of subtyping mechanisms.

<table>
<thead>
<tr>
<th></th>
<th>SOS</th>
<th>WFS</th>
<th>WCS</th>
<th>SPS</th>
<th>WPS</th>
<th>WMS</th>
<th>GML</th>
<th>GML SensorML</th>
<th>O&amp;M</th>
<th>KML</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.0</td>
<td>2.0</td>
<td>2.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.3.0</td>
<td>3.2.1</td>
<td>3.1.1</td>
<td>1.0</td>
<td>2.2.0</td>
</tr>
<tr>
<td>#AET</td>
<td>61</td>
<td>15</td>
<td>74</td>
<td>52</td>
<td>2</td>
<td>2</td>
<td>63</td>
<td>47</td>
<td>53</td>
<td>124</td>
</tr>
<tr>
<td>#SG</td>
<td>83</td>
<td>11</td>
<td>123</td>
<td>72</td>
<td>4</td>
<td>0</td>
<td>112</td>
<td>60</td>
<td>72</td>
<td>5</td>
</tr>
<tr>
<td>#TE</td>
<td>291</td>
<td>55</td>
<td>356</td>
<td>243</td>
<td>30</td>
<td>4</td>
<td>300</td>
<td>182</td>
<td>241</td>
<td>47</td>
</tr>
<tr>
<td>#TR</td>
<td>59</td>
<td>1</td>
<td>15</td>
<td>54</td>
<td>1</td>
<td>0</td>
<td>13</td>
<td>53</td>
<td>57</td>
<td>0</td>
</tr>
</tbody>
</table>

Note: WMS, web map service; WFS, web feature service; WCS, web coverage service; SOS, sensor observation service; WPS, web processing service; SPS, sensor planning service; GML, geography markup language; SensorML, sensor markup language; O&M, observation and Measurements; KML, keyhole markup language.

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

<table>
<thead>
<tr>
<th></th>
<th>SOS</th>
<th>WFS</th>
<th>WCS</th>
<th>SPS</th>
<th>WPS</th>
<th>WMS</th>
<th>GML</th>
<th>GML SensorML</th>
<th>O&amp;M</th>
<th>KML</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.0</td>
<td>2.0</td>
<td>2.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.3.0</td>
<td>3.2.1</td>
<td>3.1.1</td>
<td>1.0</td>
<td>2.2.0</td>
</tr>
<tr>
<td>DPR</td>
<td>0.13</td>
<td>0.12</td>
<td>0.15</td>
<td>0.13</td>
<td>0.05</td>
<td>0</td>
<td>0.15</td>
<td>0.14</td>
<td>0.13</td>
<td>0.05</td>
</tr>
<tr>
<td>DPF</td>
<td>2.20</td>
<td>1.47</td>
<td>1.48</td>
<td>2.20</td>
<td>1.05</td>
<td>1</td>
<td>1.40</td>
<td>2.17</td>
<td>2.15</td>
<td>1.13</td>
</tr>
</tbody>
</table>

Note: DPR, data polymorphism rate; DPF, data polymorphism factor; OWS, OGC web services; WMS, web map service; WFS, web feature service; WCS, web coverage service; SOS, sensor observation service; WPS, web processing service; SPS, sensor planning service; GML, geography markup language; SensorML, sensor markup language; O&M, observation and Measurements; KML, keyhole markup language.

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 overlooked important details when analysing dependencies. An important detail about type derivation by restriction is that it cannot be mapped smoothly to an OOP 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 be 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 thing to say; however, Abreu and Melo (1996) analysed polymorphism in the context of OOP and stated that 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 the possible valid substitutions introduced by the subtyping mechanism. The values of DPF, also shown in Table 6, indicate that the effect of polymorphic elements on SOS and SPS is higher than on WFS and WCS. The values of the metric suggest that the effort needed to consider all 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. Therefore, we can use it to detect potential design issues looking for components with disproportionated values for the metric.

5.2.2. Schema reachability rate

The different values used to calculate the SRR metric as well as the final values of the metric are shown in Table 7. The results show 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 schema components in the main schemas, or any component that is referenced from them. This high rate suggests that the effect of subtyping mechanism on schema complexity is enormous. For the rest of the specifications the effect ranges 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 dependencies of the schema components are not explicit. This situation is a potential source of errors or unexpected situations for 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` and so on. This reference to `gml:name` is inherited by a large set of types that derives, directly or indirectly, from `gml:AbstractFeatureType` such as `gml:PolygonType`. At this point, a polygon instance whose `gml:name` element is substituted by a `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 by Tamayo et al. (2011b) to adapt schemas to the specific needs of individual applications.

Table 7. SRR values for the OWS specifications.

<table>
<thead>
<tr>
<th></th>
<th>SOS</th>
<th>WFS</th>
<th>WCS</th>
<th>SPS</th>
<th>WFS</th>
<th>WPS</th>
<th>WMS</th>
<th>GML 1.3.0</th>
<th>GML 3.2.1</th>
<th>GML 3.1.1</th>
<th>SensorML</th>
<th>O&amp;M</th>
<th>KML 2.2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>V_{gos}(G_{Sf})</td>
<td>)</td>
<td>1277</td>
<td>321</td>
<td>1349</td>
<td>1058</td>
<td>146</td>
<td>71</td>
<td>1334</td>
<td>975</td>
<td>1070</td>
<td>1076</td>
<td>398</td>
</tr>
<tr>
<td>(</td>
<td>V_{gos}(G_{S})</td>
<td>)</td>
<td>319</td>
<td>245</td>
<td>220</td>
<td>203</td>
<td>126</td>
<td>71</td>
<td>1033</td>
<td>975</td>
<td>325</td>
<td>180</td>
<td>398</td>
</tr>
<tr>
<td>(V_{S})</td>
<td>1498</td>
<td>353</td>
<td>1625</td>
<td>1266</td>
<td>179</td>
<td>80</td>
<td>1414</td>
<td>986</td>
<td>1249</td>
<td>1256</td>
<td>399</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRR</td>
<td>0.64</td>
<td>0.22</td>
<td>0.69</td>
<td>0.68</td>
<td>0.11</td>
<td>0</td>
<td>0.21</td>
<td>0.59</td>
<td>0.71</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: SRR, schema rechability rate; OWS, OGC web services; WMS, web map service; WFS, web feature service; WCS, web coverage service; SOS, sensor observation service; WPS, web processing service; SPS, sensor planning service; GML, geography markup language; SensorML, sensor markup language; O&M, observation and Measurements; KML, keyhole markup language.
Tamayo et al. (2011c) reported that only 29.5% of the components of the SOS schemas are used in a group of 56 server instances. The cause of this low usage may be that the schemas are more complex than needed, but may also be that server developers do not fully comprehend all the relations between schema components mostly because these relations are not explicit or easy to discover.

5.3. Discussion

The results of the analysis show that at least half of the presented specifications can be considered as large or complex according to all the metrics that provide some sort of categorisation. Most of the metrics coincide in finding a clear differentiation 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 service versions have not been considered and more sources quantifying the numbers of available services must also be 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 than complexity in the adoption of a given specification. Being simpler is not a guarantee to have wider adoption; this is the case in the context of schema languages of XML schema versus 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 and Bond 2004). This has been attributed to several factors such as that much academic research is irrelevant to industrial needs as academic models often relies in parameters that cannot be measured precisely in practice, or they 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 is frequently the case for software attributes such as quality, maintainability or reliability. For all these reasons, we consider pertinent to present more detailed examples of how the metrics presented in this article can be used in practice. Specifically we illustrate how metrics aid the evaluation of design decisions and how they 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. Due to space limitations use case scenarios are presented in a succinct way.

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
versus reuse, when we must choose between reusing components in existing specification schemas or redefining them. Using appropriate metrics we can have an idea of the effect of one decision or the other on the final size and complexity of the schemas.

For example, consider the hypothetical example of analysing how WMS 1.3.0 could be affected if one makes it compatible with OWS common specification 1.1.0 (OGC 2007b) and GML 3.1.1.\(^6\) With a few simple transformations we can change the Capabilities file of this specification to follow the structure defined by 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 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, consider a WMS layer as a geographic feature and, therefore, define it as a subtype of gml:AbstractFeatureType, defined in GML schemas. After modifying the WMS schemas to point to gml.xsd and calculating the values of the metrics shown in Figure 4, we can observe that the addition of GML is not very helpful, as the complexity and size of the schemas are dramatically raised.

\[6.2. \text{Use case scenario: studying evolution of specifications}\]

A valid use of metrics in a retrospective way is the analysis of the evolution of the schemas for a given specification. Evolution of schemas may be useful to understand 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 service specifications are implemented. They can also give us some hints about what to expect for the future for a certain specification.
For example, consider the evolution of GML. Figure 5 shows the values of the metrics \#CT, \#EL, \#ST, \#AT, \#AG and \#MG. In the figure, we can observe the growing trend followed by the size of GML schemas 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 \#CT 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 Section 4, 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 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 these metrics to consider the location of types or to use other equality criterion between types if we need more precision in the comparison.

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 problem of complexity of schemas. 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 have 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.

Following the steps of the GML subsetting tool, Tamayo et al. (2011b) presented an algorithm to extract customised schemas depending on specific application needs. This algorithm uses a set of XML 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 specifications as well.

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

- Use the whole GML schemas (by importing gml.xsd)
- Use the SFP
- Use selective importing (by importing feature.xsd)
- Use the GML subset extracted using the algorithm presented by Tamayo et al. (2011b)

Figure 6 shows the number of the main schema components in each of the cases listed before. It is obvious from the figure that all the approaches that try to avoid the use of the whole schemas reduce considerably the overall complexity of the schemas. We can also conclude that the use of more specific solutions instead of generic ones could greatly simplify the implementation of real systems.

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 data sets and documents (Berners-Lee 2006). This can be accomplished, among other basic principles, by means of the use of HTTP URIs not only to 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 by Schade et al. (2010), since its inception GML has provided a mechanism to implement a linked-data style model. GML allows 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? Analyse this question in the context of GML 3.1.1. We compare
Figure 6. Comparison of profile-based approaches to deal with complexity of the schemas.

8. Conclusions

In this article, 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 or complex. For all the metrics we have tried to present simple examples of potential applications to practical problems, as well as
A set of new metrics have been introduced to show the effect of the use of subtyping mechanisms in complexity. This was motivated by the high use of these mechanisms in the OWS schemas. These new metrics try to complement others included in our study as a single metric cannot be used to measure all the aspects that influence the complexity of the specifications. The metrics show that for the most complex specifications, the effort needed to consider all 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 decisions 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 the complexity of geospatial web service implementations. We are also considering the use of alternative schema languages to define the structure of geospatial data, such as RelaxNG (OASIS 2004), and its impact in specifications complexity.

Acknowledgements
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.

Notes
3. Prefix gml refers to the namespace available at http://www.opengis.net/gml
4. In a real scenario other aspects must be considered as well, such as available implementations, existing support in different mapping systems and so on.
5. Prefix sml refers to namespace available at http://www.opengis.net/sensorML/1.0
6. The purpose of this example is to demonstrate how metrics can be used in similar scenarios, and we are not suggesting here that WMS schemas have to be combined with OWS Common or GML schemas.
7. GML 1.0.0 is not defined using XML schema. The metric values shown for that version of the specification are the result of conversion from DTD to XML schema using the XML editor <oXygen/>: http://www.oxygenxml.com
8. This is not always possible such as in the case of GML 3.2.1. This version is designed in a way that schemas can only be validated correctly if they import gml.xsd.

References


OGC, 2008. OGC KML 2.2.0. OGC Document No. 07-147r2.