Complexity Metric for XML Schema Documents

Dilek Basci and Sanjay Misra

Department of Computer Engineering, Atilim University, Ankara, Turkey

dilek_basci@hotmail.com, smisra@atilim.edu.tr

Abstract. Web Services, as a new type of distributed application, use XML documents for their data representations, so design of XML schemas play an important role in software development process and needs to be quantified for ease of maintainability. In this paper, we propose a new complexity metric for XML Schema documents (XSD). On the contrary of the available complexity metrics, the proposed metric is based on the internal architecture of the XSD components and hence considers the complexities of its building components. The proposed metric has been demonstrated with examples. Further a comparative study with other similar metrics proves its soundness and robustness.

Keywords: Complexity metric, W3C XML Schema, Web services, WSDL, SOAP

1. Introduction

Connectivity among users and different type of applications via World Wide Web continues to proliferate at an astounding rate since the invention of the Internet. With the emergence of web applications the idea of integrating them as a very loosely coupled software components leads to the development of Web Services as a new type of distributed applications that are based on web technologies. Because the design purpose of these components is to provide services available from anywhere on the Internet, the idea of dynamically integrating them at runtime have been gaining a great deal of acceptance by different types of parties that are connected to the internet for different purposes. For integration of these components the interoperability requirement has become a major concern of the service suppliers and gained much bigger priority across the industry. In order to satisfy this requirement the applications must have the capability of communication with the other applications via Internet protocols and sending and receiving data. In order to work in interoperable manner these distributed applications have the concise and clear agreement on the common specifications of protocols and data format. Additionally, such applications should be developed as fully autonomous components, that is, they

must have free of dependence to run on different types of platforms. Since underlying business and data models used by applications that are indented to be integrated may change over time, in order for accommodation of these changes building a flexible document structure that can be extended will pay off in the future.

The desired integration of such applications running on different software platforms is provided by Web Services[3], [8] that are based on an open standardized suite of technologies such as eXtensible Markup Language (XML) [14], Hyper Text transport Protocol (HTTP)[3], [8], [4], Simple Object Access Protocol[3], [8] (SOAP) and Web Service Description Language[3], [8] (WSDL). Further, this integration is achieved more rapidly, easily, and cheaply than ever before.

The usage of Web service described by the WSDL [3], [8] documents requires a service provider and consumer to exchange XML messages. The message format must be well defined in order for the message sender can easily construct and the message receiver can process. The WSDL document uses a schema to define the name and types of the elements and attributes conveyed by the message. Once service consumers have the WSDL file they can communicate with the Web services by using SOAP [3], [8]. If we think Web services as remote objects that can be exposed through WSDL, the SOAP provides a mechanism to remotely access to these objects across the Internet without having the problems in integration and interoperability issues between enterprises. In this mechanism XML documents are used for representing and transporting data to and from integrated applications' public interfaces.

Representing the application data with XML documents requires making strategic decisions that take into consideration some design issues which should be handled at design time, such as performance, security, extensibility, reusability, data access etc. In XML context, the data representations are made by designing schemata which can be written in different XML schema languages such as DTD [14], W3C XML Schema [15], RELAX [5], [20]. W3C XML Schema [15] and DTDs [14] are the most favored schema languages for generating XML documents.

Deploying XML documents is a challenge problem for an application without using supporting schema technology. In order for XML documents to provide a common understanding about data exchanged between applications these XML documents should be validated against the XML schema definition (XSD). For instance, the application that requests customer information and the application that provides information as a response to the requester application should agree on that the exchanged data is exactly about the customer information. In this point of view XML schemas play an extremely important role in software construction projects. From the Web service design perspective the decisions in XML schema definition (XSD) design can have significant impact on the Web service design. Neglecting schemas implies that the schema validators are not used to determine if a given XML document satisfies desired data transported among applications. In such a case the required check have to be performed by the application programs implying that the application developers have to write lengthy code. Using schemas not only provides common understanding about exchanged data but also the ability of easy access methods for XML documents to be validated.

All above considerations related with XML documents imply that the schemas need to be properly designed, so that it can be easily maintained in order for XML

data to be effectively and properly used by distributed applications. Further, schema metrics must be developed to enable quantification of schema size, complexity, quality and the other properties; however, a few researches that deal with schema quality and complexity metric has been done. Klettke *et al.* [6] used some well known procedural metrics for evaluating complexity of DTD, such as LOC, McCabe, Fan-in and Fan-out, DIT. An extension of this paper is [7] that present eleven metrics for XSDs and two formulae that use the metrics to compute quality indices for XSDs and complexity indices for conforming XML documents. The metrics reported in that paper are mostly related with XSD components' counts such as number of elements, complex and simple types, annotations, type references, unbounded elements definitions/declarations. *Mustafa et al.* [10] demonstrated that the XML documents that are generated by the DTD with higher nesting levels have higher weights and more complicated compared to the documents with lower nesting levels. In this demonstration various techniques were used to represent XML documents as a regular expression and by determining complexity values of regular expression; a tree representation of XML documents and the implementation of Weight Allocation (WA) algorithm. A comprehensive analysis was made for XML Schema documents usage in [11] and, in addition to, the measures for XSD-agnostic schema size such as number of all XML nodes; XSD-aware counts such as number of all element and attribute declaration; all type, and model group definitions, the metrics LOC, McCabe were also revisited. In [13] to measure structural complexity of XSDs the metrics which are Tree Impurity, Efferent and Afferent Coupling, Instability, Cohesion, Normalized Count of Modules were evaluated by the adaptation of some well known existing metrics developed for other software artifacts. Besides these papers many online articles are also available on the web [16].

The common approach to measure the complexity of XML schema documents in [7], [11] is to count the number of schema components. However, the metrics that measure schema's complexity by counting the number of each component do not give sufficient information about complexity value of a given schema and the complexity of each independent component is also important, which were neglected in those papers. This is the main motivation for us to develop new metric for XSDs. Another motivation to focus on to develop the complexity metric for XSDs is that W3C XML Schema language [15] has the stronger capability than DTD to describe the vocabularies of XML documents and has general agreement of being the schema language of the future for XML. We suggest that the complexity of a given XML schema document written in W3C XML Schema language closely depends on complexities of internal complexities of its building components, that is, each component contributes their complexity values on the basis of their design architectures to the schema document's complexity. In this point of view, it will be meaningful to assign a weight value for each component that reflects complexity of each component called complexity degree. Further, for calculating the complexity of the schema document each of its component's weight values should be summed up in order to evaluate a single complexity value.

In section 2 we define our metric for XSD. The proposed metric is demonstrated by examples in section 3. A comparative study with other measures has been done in the same section. Lastly, section 4 provides concluding remarks and a reflection on future work.

2. Proposed Metric

Major building components of XML Schema are elements having simple or complex type as a type reference; attributes, simple and complex types, elements and attributes group definitions/ declarations [18]. The schema document may not necessarily validate any XML document and can be designed as a library document. Based on its design style [12] a given schema may have different number of components declared/defined locally or globally. For example, the number of complex or simple type definitions may be greater than element with or without attributes declaration or vice versa or the schema may use global elements and attributes group definitions or encode all groups inside complex type's content model definition instead. Based on it, we proposed that, the complexity of XSD depends upon the following factors;

a) The complexity due to elements and attributes definitions/declarations.
b) The complexity due to elements and attributes group definitions/ declarations.
c) The complexity due to all types including user defined and built-in simple type and complex type definitions.

Accordingly, the total complexity of the XSD is given by the following formula

$$C(XSD) = C(V_g) + C(G_g) + C(T_g) \quad (1)$$

where C(XSD) is the complexity value of the schema document(XSD) written in XML Schema language; $C(V_g)$ is the total complexity values of all *unreferenced* global elements and attributes that is assigned by the weight values of reflected by their type complexity values; $C(G_g)$ is the total complexity values of *unreferenced* global elements and attributes group, and $C(T_g)$ is the total complexity values of *unreferenced* global complex and simple type definitions/declarations of XML Schema document. By the word "*unreferenced"* we mean components that have no reference made within any component definitions of the current schema. The reason for considering unreferenced components is that; since an element or attribute being declared locally or globally have type reference to any globally defined complex or simple type definitions, we are at risk adding two times both element's, attribute's complexity values and their respective type complexity values to C(XSD). Similarly, since global elements and attributes group can be referenced inside any complex type definitions or the other group definitions we again risk for adding two times complexities of both complex type's and group's definitions.
Definitions of each component of C(XSD) are given below:

$C(V_g)$ can be defined as:

$$C(V_g)=C(E_g)+C(A_g) \quad (2)$$

where $C(E_g)$, and $C(A_g)$ are complexities of *unreferenced* global elements and attributes definitions/declarations respectively and given by:

$$C(E_g)= \sum_{i=1}^{N} we_i E_{g\,i}\,; \quad (3)$$

$$C(A_g)= \sum_{j=1}^{M} wa_j A_{g_j} ; \quad (4)$$

where, ***N, and M*** are the total number of *unreferenced* global element, attribute declarations; we_i , and wa_j are corresponding weight values for the type definition of the *unreferenced* global element E_{g_i} and attribute A_{g_j} .

$C(G_g)$ can be defined as:

$$C(G_g) = C(EG_g) + C(AG_g) \quad (5)$$

where, $C(EG_g)$, and $C(AG_g)$ are the complexity of the *unreferenced* global elements and attributes group definition/declaration respectively and are defined as:

$$C(EG_g) = \sum_{t=1}^{K} weg_t EG_{g_t} \quad (6)$$

$$C(AG_g)= \sum_{s=1}^{P} wag_s AG_{g_s} \quad (7)$$

where, ***K, and P*** are the total number of *unreferenced* global elements and attributes group declarations/definitions; weg_t ,*and* wag_s are corresponding weight values of the elements group EG_{g_t} and attributes group AG_{g_s} respectively.

$C(T_g)$ is defined as:

$$C(T_g)=C(cT_g) + C(sT_g) \quad (8)$$

where, $C(cT_g)$, and $C(sT_g)$ are complexity of global complex and simple type definition respectively and defined as:

$$C(cT_g)= \sum_{r=1}^{R} wc_r cT_{g_r} \quad (9)$$

$$C(sT_g)= \sum_{q=1}^{Q} ws_q sT_{g_q} \quad (10)$$

where, *R, and Q* are the number of global *unreferenced* complex-type and simple-type definitions; wc_r,*and* ws_g are corresponding weight values of complex and simple type definitions cT_{g_r}, sT_{g_q} respectively. Thus, the equation for C(XSD) *(see the equation 1)* can be rewritten as:

$$C(XSD) = C(V_g) + C(G_g)+ C(T_g)$$

$$= [C(E_g) + C(A_g)] + [C(EG_g) + C(AG_g)] + [C(cT_g) + C(sT_g)]$$

$$=[\sum_{i=1}^{N} we_i E_{g_i} + \sum_{j=1}^{M} wa_j A_{g_j}]+[\sum_{t=1}^{K} weg_t EG_{g_t} + \sum_{s=1}^{P} wag_s AG_{g_s}]+$$

$$[\sum_{r=1}^{R} wc_r cT_{g_r} + \sum_{q=1}^{Q} ws_q sT_{g_q}] \quad (11)$$

As explained earlier, weight values for each schema component can reflect the complexity degree of corresponding component and are assigned on the basis of their design structures i.e. its internal architectures, since components of XSDs can be dependent on each other in the sense that the definition/declaration of any component

may use the other components [17]. As a result, while the weight value of element depends on its type's weight value, that type's weight value depends on its internal structure. In this point of view, due to the complex type definition can include nested compositors or particles with different number of occurrences [17], [18] based on its content model, the weight value for an element having simple type as a type reference differs from the element having complex type. Similarly, the weight value for a complex type with simple content model may differ from a complex type with complex content model. Hence, while assigning weight value to a complex type definition, weight values of each constituent member encoded in the content model of it should be considered. This is also valid to evaluate weight values for the element and attribute groups definitions since each member of any type of groups definitions may have different complexity weight values.

We assume that built-in simple types have the weight value of 1 since these types are simplest data type structure used in the schema document (XSD). In the schema document an element type that does not explicitly specify a structure type implicitly specifies *anyType* [2], [12], [19], as the structure type. The content of an element in an XML instance whose structure type is *anyType* is unconstrained. The simplest type structure for *anyType* can be a built-in simple type. For this reason we assumed that the weight value for any attributes or elements whose type definition is specified by *anyType* is 1. The <*any*> [12, 17, 18, 19] element provides a mechanism for specifying elements with what the XML Schema Recommendation [17] calls a wildcard. By the usage of the <*any*> element an XML validator validates elements in an XML instance document. The <*any*> element generally specifies a set of namespaces against which the XML validator may validate. The XML validator searches each namespace for global element types that might correspond to the elements referenced in the XML instance. Since, in the simplest case that global element types can be a simple type we made another assumption that the weight for an element declared by <*any*> element in the schema is 1. Similarly, we also assume that the weight value for an attribute declared by <*anyAttribute*> [12], [17], [18], [19] element in the schema is 1 since <*anyAttribute*> element is analogous to the <*any*> element of W3C XML Schema.

Another point that needs to pay attention is that we only take into considerations referenced components of the external schemas that are included to the current schema via *import, include* and *redefinition* mechanism [12], [17], [18], [19] while evaluating complexity value of the current schema document. Based on these assumptions, weight values for each XML schema component can be calculated as follows:

Element's weight value

we = ws, if elements have simple- type (12.1)

=wc, if element has complex-type (12.2)

=1, if element is declared by using **< any >** *element* (12.3)

=1, if element is declared by using ***anyType*** (12.4)

where *ws* is the weight value for simple type definition; *wc* is the weight value for the complex type definition.

Attribute's weight value:

wa=ws ,since attributes can only have simple-type (13.1)

=1, if attribute is declared by < ***anyAttribute*** *> elements* (13.2)

Weight values of elements group can be calculated by summing up all its elements' weight values, that is:

$$weg = weg_{baseGroup} \pm \sum_{i=1}^{N} we_i E_i \quad (14.1)$$

where, $weg_{baseGroup}$ is the weight value of base elements group of defined elements group if it is extended or restricted by redefinition mechanism of W3C XML Schema.; ***N*** is the number of newly declared elements inside group redefinition or the number of not inherited elements from base group to redefined group; $\boldsymbol{we_i}$ is the weight value of corresponding declared element $\boldsymbol{E_i}$.

Weight values of attributes group can be calculated by summing up all its attributes' weight values and define as:

$$wag = wag_{baseGroup} \pm \sum_{i=1}^{N} wa_i A_i \quad (15.1)$$

Here, *wag* definition is similar to *weg* definition, but, in this case we are mentioning about attributes. Note that the weight values of the attributes that are prohibited by the derived attribute group subtracted from the weight value of the base attribute group. The meaning of ± sign will be explained in the next paragraph.

The weight value of a complex-type can be calculated by all its constituent components (elements, attributes, and groups) and can be defined as:

$$wc = wc_{\boldsymbol{baseType}} \pm [\sum_{i=1}^{N} we_i E_i + \sum_{j=1}^{M} wa_j A_j + \sum_{t=1}^{K} weg_t EG_t + \sum_{s=1}^{P} wag_s AG_s] \quad (16)$$

where, $wc_{baseType}$ is the weight value of derived complex-type's parent; ***N, M, K, and P*** are the number of local or referenced elements, attributes, element groups and attribute groups definitions/declarations respectively. If a complex type is not derived complex-type these capitals represents the number of not inherited components from base type or the number of newly added components to derived complex-type definition; we_i, wa_j, weg_t *and* wag_s are corresponding weight values of element E_i, attribute A_j, element group EG_t and attribute group AG_s respectively. Note that the ± sign in equations (14.1), (15.1) and (16) indicates that if groups and complex types are derived by extension, then the weight values of all newly added components of the derived groups or complex types should be added to respective parent's weight value and if it is derived by restriction then weight values of all constituent members that are not inherited from parent should be subtracted from its parent's weight value. Note that element and attribute groups can be derived via redefinition mechanism of W3C XML Schema [12], [18].

Weight value of a simple-type can be defined as:

ws =1, *if it is built in simple type* (17.1)

= **r**|r *is the number of restriction, if it is derived by restriction.* (17.2)

= **u** |u = $\sum_{i=1}^{P} w_i M_i$, *if it is derived by union.* (17.3)

= l | l *is the weight value of item type, if it is derived by list.* (17.4)

In (17.3*), P* is the number of members declared within union simple-type; w_i is the weight value of member M_i that can be built-in or derived simple type and equals to *ws* since the types of the members should only be simple type [18].

3. Illustration of the Proposed Metric

To illustrate the proposed metric we used one WSDL document example, *WS-BaseFaults.wsdl*, available online [22]. Since the WSDL document uses the <type> element as a container for data type definitions that can represented by using XML Schema. In figure 1.a the WSDL document example named as *WS-BaseFaults.wsdl* is shown and the schema document, *WS-BaseFaults.xsd* that is included inside its <*type*> element is given in figure 1.b. The *WS-BaseFaults.wsdl* document defines an XML Schema type for a base fault, along with rules for how this fault type is used by Web services. A designer of a Web services application often uses interfaces defined by others. Managing faults in such an application is more difficult when each interface uses a different convention for representing common information in fault messages. Support for problem determination and fault management can be enhanced by specifying Web services fault messages in a common way. When the information available in faults from various interfaces is consistent, it is easier for requestors to understand faults. It is also more likely that common tooling can be created to assist in the handling of faults [22].The calculation of the complexity value for the schema document *WS-BaseFaults.xsd* is explained in section 3.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="BaseFaults"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:wsbf="http://www.ibm.com/xmlns/stdwip/web-
services/WS-BaseFaults"targetNamespace="http://www.ibm.
com/xmlns/stdwip/web-services/WS-BaseFaults">
        <!-- ======== Types Definitions ======= -->
  <wsdl:types>
     <xsd:schema >
         <xsd:import
           namespace=
           "http://www.ibm.com/xmlns/stdwip/web-services/WS-
BaseFaults"
           schemaLocation=     "./WS-BaseFaults.xsd"/>
     </xsd:schema>
  </wsdl:types>
  <wsdl:message name="BaseFaultMessage" >
    <wsdl:part name="Fault" element="wsbf:BaseFault" />
  </wsdl:message>
</wsdl:definitions>
```

Figure1.a. The WSDL document *WS-BaseFaults.wsdl* uses the schema document *WS-BaseFaults.xsd* (see figure1.b) declared inside its <type> element.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
xmlns:wsbf="http://www.ibm.com/xmlns/stdwip/web-services/WS-
BaseFaults"
targetNamespace="http://www.ibm.com/xmlns/stdwip/web-
services/WS-BaseFaults">
<xsd:import
namespace="http://schemas.xmlsoap.org/ws/2003/03/addressing"
schemaLocation="http://schemas.xmlsoap.org/ws/2003/03/addressing
"/>
<xsd:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/xml.xsd"/>
    <!-- ----BaseFault Types------->
    <xsd:element name="BaseFault" type="wsbf:BaseFaultType"/>
    <xsd:complexType name="BaseFaultType">
        <xsd:sequence>
         <xsd:element name="Timestamp" type="xsd:dateTime"
minOccurs="1" maxOccurs="1"/>
            <xsd:element name="Originator"
type="wsa:EndpointReferenceType" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="ErrorCode" minOccurs="0"
maxOccurs="1">
                <xsd:complexType>
                    <xsd:complexContent mixed="true">
                        <xsd:extension base="xsd:anyType">
                            <xsd:attribute name="dialect"
type="xsd:anyURI" use="required"/>
                        </xsd:extension>
                    </xsd:complexContent>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="Description" minOccurs="0"
                 maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:simpleContent>
                        <xsd:extension base="xsd:string">
                            <xsd:attribute ref="xml:lang"
                              use="optional"/>
                        </xsd:extension>
                    </xsd:simpleContent>
                </xsd:complexType>
            </xsd:element>
            <xsd:elementname="FaultCause"type="wsbf:BaseFaultTyp
e" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Figure1.b. The schema document *WS-BaseFaults.xsd*.

All the components definition/declarations with their weight values of the schema document *WS-BaseFaults.xsd is given in table1*. The schema document includes the other two schemas by import mechanism of W3C XML Schema [12], [17], [18], [19], thus, adds the complexities of those schemas referenced components to its complexity value. The complex type *EndpointReferenceType* is imported from the included schema *addressing.xsd* [23] as a type reference for the local element *Originator*. The attribute declaration of the element *Description* is imported by giving reference to the "*xml:lang"* attribute declaration of the other included schema document *xml.xsd*.

Table1. The components of the schema document *WS-BaseFault.xsd* (*see Figure1.b*).

QName	Notes	Weight		
		Symbol	Value	Equation No
BaseFault	*A complex-typed global element*	*we*	*12*	*12.2,16*
Description	*A global element having complex type derived by extension*	*we*	*2*	*12.2,16*
BaseFaultType	*A global complex type definition*	*wc*	*12*	*16*
Timestamp	*A simple-typed local element*	*we*	*1*	*12.1,17.1*
Originator	*A complex-typed local element*	*we*	*9*	*12.1,16*
ErrorCode	*A local element having complex type derived by extension*	*we*	*2*	*12.1,16*
dialect	*A built-in simple-typed local attribute*	*wa*	*1*	*13.1,17.1*
xml:lang	*An attribute imported from* xml.xsd *having user-defined simple type derived by union*	*wa*	*1*	*13.1,17.3*
wsa:EndpointReferenceType	*A complex type imported from* addressing.xsd	*wc*	*9*	*16*

From *Table1* it can be observed that elements may have different complexity degrees based on the complexity value of their type definitions. For example, the two global elements *"BaseFault"* and *"Description"* have different complexity degrees represented by their weight values even though both are complex-typed elements. The *BaseFault* element's weight value is assigned by calculating the weight value of its type definition which is *BaseFaultType.* For this assignment we use the equations (12.2) and (16). According to the equation (12.2) the element's weight value, *we*, is calculated by using the equation (16) since the element has complex type. In order to calculate the complex type weight value, *wc*, we should sum the weight values of all the components declared inside that complex type definition. *The BaseFaultType* complex type includes three elements declaration. Therefore we sum up the weight values of the elements *TimeStamp, Originator and ErrorCode. Thus, the weight value, wc, for BaseFaultType is:*

$$wc_{BaseFaultType} = 1 + 9 + 2$$
$$= 12$$

Since the complex type *BaseFaultType* is referenced by the element *BaseFault* as a its type structure definition the weight value for this element is assigned as 12. Note that in order to assign weight values for the imported components we should analyze the external schema documents included via import, include and redefine mechanism of W3C XML Schema. For example, the weight value for the local element *Originator* declared inside the complex type *BaseFaultType* definition is we should analyze the included external schema document *addressing.xsd* [23] since the imported complex type definition *EnpointReferenceType is* encoded inside this external schema. Similarly, the weight value for the attribute declaration of the element *Description* we should refer the other included external schema document *xml.xsd* [24].

3.1 Calculation of C (XSD)

As mentioned in section 2 while evaluating complexity value of a given XML Schema document, C(XSD), we are adding complexity values of all *unreferenced* globally defined/declared elements, attributes plus unreferenced global element, attribute groups plus unreferenced global complex and simple type definitions/declarations. The C(XSD) value for the schema document *WS-BaseFaults.xsd* is calculated by only summing the weight values i.e. complexity values of two unreferenced global elements, namely *BaseFault* and *Description*, since this schema has neither unreferenced groups nor type definitions/declarations. The values of C (XSD) and its components are shown in table2.

Table2. The overall complexity value of the schema document *WS-BaseFault.xsd* & C(XSD).

Components	**Value**	**Equation no**
$C(E_g)$	14	3
$C(A_g)$	0	4
$C(EG_g)$	0	6
$C(AG_g)$	0	7
$C(cT_g)$	0	9
$C(sT_g)$	0	19
C(XSD)	14	11

The overall complexity value of the schema document *WS-BaseFaults.xsd* is evaluated by using equations (1) and (11):

$$C(XSD) = C(V_g) + C(G_g) + C(T_g)$$

$$= [\sum_{i=1}^{N} we_i E_{g_i} + \sum_{j=1}^{M} wa_j A_{g_j}] + [\sum_{t=1}^{K} weg_t EG_{g_t} + \sum_{s=1}^{P} wag_s AG_{g_s}]$$

$$+[\sum_{r=1}^{R} wc_r cT_{g_r} + \sum_{q=1}^{Q} ws_q sT_{g_q}]$$

$$= [\sum_{i=1}^{N} we_i E_{g_i} + \sum_{j=1}^{M} wa_j A_{g_j}] + 0 + 0$$

$$= \sum_{i=1}^{N} we_i E_{g_i} + 0$$

$$= we_{BaseFault} + we_{Description} \quad (18)$$

The global element *Description* has a derived anonymous complex type with simple content and its weight value is assigned based on its type definition's weight value. The anonymous complex type is derived from the built-in simple type *string* and hence, its base type weight value is 1. Further, this anonymous complex type has only one attribute declaration having user defined simple type derived by union from the built in simple type *language.* Note that to assign the weight value to this attribute we should refer the external schema document *xml.xsd* [24]. Hence the weight value for the element *Description* is evaluated by using the equations 12.2 and 16:

$$we_{Description} = wc$$

$$= wc_{baseType} \pm [\sum_{i=1}^{N} we_i E_i + \sum_{j=1}^{M} wa_j A_j + \sum_{t=1}^{K} weg_t EG_t + \sum_{s=1}^{P} wag_s AG_s]$$

$$= 1 + [0 + \sum_{j=1}^{1} wa_j A_j + 0 + 0]$$

$$= 1 + \sum_{j=1}^{1} wa_j A_j$$

$$= 1 + wa_{xml:lang}$$

The weight value, *wa*, for the attribute "*xml:lang"* is evaluated by the equations (13.1) and (17.3) since the attribute has user-defined simple type derived by union. When we look to the declaration of the attribute "*xml:lang*"encoded inside *xml.xsd*[24] ,we see that it has union simple type with only one member having the built-in simple type *language*. According to (13.1) and (17) the weight value for this attribute is:

$$wa_{xml:lang} = ws$$

$$= u$$

$$= \sum_{i=1}^{P} w_i M_i$$

$$= 1$$

where *wi* is the weight value for the type of union member having the built-in simple type *language* and its value is 1.

As a result, the weight value for the element *Description* is recalculated by (12.2) and (16):

$$we_{Description} = wc$$
$$= wc_{baseType} \pm [\sum_{i=1}^{N} we_i E_i + \sum_{j=1}^{M} wa_j A_j + \sum_{t=1}^{K} weg_t EG_t + \sum_{s=1}^{P} wag_s AG_s]$$
$$= 1 + wa_{xml:lang}$$
$$= 1 + 1$$
$$= 2$$

The weight value, $we_{BaseFault}$, for the element *BaseFault* is evaluated in a similar way and the weight value for it is 12. Hence, by turning back to the equation (18) and putting the weight values of the elements *BaseFault* and *Description* we evaluate the overall complexity value for the schema document *WS-BaseFault.xsd as:*

$$C(XSD) = we_{BaseFault} + we_{Description}$$
$$= 12 + 2$$
$$= 14$$

4. Concluding Remark and Future Work

Flexible nature and ease of implementation of XML allows developer to create their own mark-ups to describe data, to define document types, to store, share information and to transmit documents across web, thus, XML has been gaining a general acceptance as a standard for data representation and exchange information since its development. As a new type of distributed application based on XML technologies, Web Services [3] use XML documents for their data representations. In this aspect designing XML schemas play an important role in software development process and needs to be quantified for ease of maintainability. For this purpose we have presented the complexity metric for the schema documents written in W3C XML Schema language since it has the stronger capability than DTD to describe the vocabularies of XML documents and has general acceptance of being the schema language of the future for XML. The proposed metric value was evaluated on the basis of the internal complexities of major building components of XSDs and computed by using the provided formulas. Further, we demonstrated that the internal architecture of XSDs' building components affect the overall complexity of XSD. From this demonstration we can insist on that our complexity metric gives better indication than the metrics which measures the complexity of a given schema based on the counts of schema's each components. In order for the proposed metric to be reliably applied for schema quantification it should be validated. The empirical validation of the proposed metric in this paper is one of our future works. As another future work we aimed to adopt existing grammar metrics [1], [9] to the schema documents written in DTD, W3C XML Schema, RELAX and to develop new ones since these languages can also be represented by tree grammars [21].

References:

1. Alve,T., Visser,J.: Metrication of SDF Grammars, Technical Report. Departamento de Informática, Universidade do Minho, DI-PURe-05.05.01, May 2005.
2. Binstock,C., Peterson, D., Smith, M., Wooding,M., Dix, C., Galtenberg,C.:The XML Schema Complete Reference. Addison Wesley Professional Publischers.(2002)
3. Erl,Thomas:Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services, Prentice HallPublischers.(2004)
4. Gourley, D., Totty, B.: HTTP: The Definitive Guide. O'Reilly Publischers(2002)
5. ISO/IEC:Information Technology – Text and Office Systems – Regular Language
6. Description for XML (RELAX) – Part 1: RELAX Core, 2000.D TR 22250-1.
7. Klettke, M. , Scneider,L. ,Heuer, A.,"Metrcis for XML document collections", XMLDM Workshop,Czech Republic,2002,pp.162-176
8. McDowell,A., Schmidt,C., Yue,K.:Analysis and Metrics of XML Schema. In SERP '04, Proceedings of the International Conference on Software Engineering Research and Practice, 538-544. CSREA Press(2004)
9. Newcomer,E.,Greg Lomow,G.: Understanding SOA with Web Services. Addison Wesley Professional.(2004)
10. Power,J.F., Malloy,B.A.: A metrics suite for grammar-based software. Journal of Software Maintenance and Evolution. John Wiley & Sons, Inc., 16(6), 2004, pp. 405-426.
11. Qureshi,Mustafa H., Smadzadeh,M.H.,:Determining the Complexity of XML Documents, Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II - Volume 02,pp. 416 – 421,April 2005.
12. R. L¨ammel,R., Kitsis, S., Remy,D.: Analysis of XML schema usage. In Conference
13. Proceedings XML 2005(2005)
14. Van der Vlist , Eric:XML Schema. O'Reilly Publication(2002)
15. Visser,J.:Structure Metrics for XML Schema, Proceedings of XATA.(2006)
16. http://www.w3.org/TR/1998/REC-xml-19980210
17. http://www.w3.org/TR/xmlschema-1/
18. http://www.xfront.com/GlobalVersusLocal.html; http://www.oreillynet.com/xml/blog/2006/05/metrics_for_xml_projects_1_ele.html
19. http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/
20. http://www.w3.org/TR/2001/PR-xmlschema-0-20010330/
21. http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/
22. http://www.xml.gr.jp/relax.
23. http://www.cs.ucla.edu/~ dongwon/paper/.
24. http://www-128.ibm.com/developerworks/library/specification/ws-resource/
25. http://schemas.xmlsoap.org/ws/2003/03/addressing.xsd
26. http://www.w3.org/2001/xml.xsd