ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **14**, Number 1, 2011, 31–50

Document Type Definition (DTD) Metrics

Dilek BASCI[1], Sanjay MISRA[2]

[1]BILGI Geographic Information Conversion & Management System Co. Ltd., Ankara,Turkey

[2]Department of Computer Engineering, Atilim University, Ankara, Turkey

E-mail: `ferceran@gmail.com, smisra@atilim.edu.tr`

Abstract. In this paper, we present two complexity metrics for the assessment of schema quality written in Document Type Definition (DTD) language. Both "Entropy (E) metric: E(DTD)" and "Distinct Structured Element Repetition Scale (DSERS) metric: DSERS(DTD)" are intended to measure the structural complexity of schemas in DTD language. These metrics exploit a directed graph representation of schema document and consider the complexity of schema due to its similar structured elements and the occurrences of these elements. The empirical and theoretical validations of these metrics prove the robustness of the metrics.

Key-words: XML schema, Document Type Definition, software metrics.

1. Introduction

The eXtensible Markup Language (XML) technologies have been promising to provide a common standard mechanism for interoperable integration of diverse IT processes and have been gaining extraordinary acceptance from the basic to the most complicated business and scientific processes. Furthermore, many different domains, organizations and content providers have been publishing and exchanging information over the Internet by the usage of XML. In recent years, Web services [6, 12, 34 and 41] based on XML technologies have been emerging as the de-facto mechanism for exchanging structured information between organizations and applications. Due to its flexible nature and ease of implementation, XML serves very well as a ubiquitous, platform-independent data representation and transport format and hence, has been easily adopted in diverse fields.

Efficient implementation of XML in diverse domains requires well-designed XML schemas. In this point of view, design of XML schemas plays an extremely important role in software development process and needs to be quantified for the ease of maintainability. The maintainability is one of the important factors that affects the quality of the software project and effective management of any type of software projects requiring modeling, measurement, and quantification. Moreover, software metrics should be used in order to improve the productivity and quality of software. Since W3C XML Schema and DTDs are the more popular schema languages [28] for generating XML documents, we have tried to develop metrics [3–5] for these languages. We have presented [5] a complexity metric for the assessment of the quality of XML schema and the complexity value for a given XSD is evaluated by considering all of its components' complexities. This metric is developed for XML Schema Documents (XSD) written in W3C XML Schema language. In the present work, we develop two metrics for DTD language E(DTD) and DSERS(DTD), which can differentiate DTDs in terms of their physiological complexities due to the diversity and repetitions in their elements' structures. The E(DTD) metric is based on concept of entropy from information theory and follow the similar approach taken by Davis and LeBlanc[9]. The DSERS(DTD) metric is adopted from ARS metric proposed by Boxell [2] as an interface metric for reusability analysis of software components and is used to measure the consistency of the arguments that software interface uses. In the preliminary work of this study, we have introduced E(DTD) in [3] and DSERS(DTD) in a Workshop of SOA and Web-services [4]. In this paper, we extend our previous work, evaluate and validate our metrics through theoretical and empirical validation. Additionally, to prove the worth and importance of the metrics, we demonstrate a comparative study with similar measures.

The paper is organized in five sections. The relationship between entropy and software complexity is established in Section 2. The proposal of new metrics and its illustrations are given in Section 3. Theoretical and empirical validations are given in Section 4. Lastly, Sections 5 and 6 provide concluding remarks and a reflection on future work.

2. Entropy as a Complexity Measurement

The entropy adapted from communication information theory of Shannon [38] is defined as a measure of uncertainty or variety. According to Shannon's definition, if M is the message set (M_1, M_2, ...,M_n) from which the message consists of, the self-information of the ith message M_i with probability of occurrence $P(M_i)$ is:

$$I(M_i) = \log_2 P(M_i).$$

The probability P (M_i) is of ithmost frequently occurring message in a conveyed message set M consisting of N number of messages and can be defined as:

$$P(M_i) = f_i/N,$$

where f_i is the number of occurrences of ith most frequently occurring message.

Then, the value of entropy can be calculated by averaging the self-information over all messages in M message set. Thus, entropy is given by:

$$\mathrm{H} = -\sum_{i=1}^{n} \mathrm{P}(\mathrm{M}_i) \log_2 \mathrm{P}(\mathrm{M}_i).$$

The entropy concept has been applied by many researchers [7, 10, 11, 13, 14, 15, 27, 32, 36, and 43] for the assessment of the complexity of software products that are developed by using procedural or OO programming technologies. These measures are developed by adapting Shannon's entropy theory [38] as a measure of uncertainty or variety. Davis and LeBlanc [9] have used entropy that is adopted from Shannon's notion of entropy to assess syntactic complexity of FORTRAN and COBOL code. In their work, the software code is partitioned into "chunks". Chunks are defined as a group of related items, which can be formed into single mental concepts. The entropy is used to measure three aspects of chunks: the structure of chunk connections, chunk content and chunk size. In order to use entropy metric as a structure measurement LeBlanc et al. [9] have constructed a graph that exhibits the data dependency between these chunks. The chunks are distinguished by the number of their incoming and outgoing edges in the graph representation of a program. From this graph it is observed that a graph consisting of a number of similar substructures tends to have lower entropy than the other graph not having such regularity although these two graphs have the same number of nodes. The result drawn from this observation is that entropy is a potentially more useful measure than McCabe's V(G) [31] since entropy takes into account for greater variety in structure brought by nesting, but this variety is ignored by V(G).

Harrison [15] has used entropy as a measure of information content. Harrison's software complexity metric is based on empirical program entropy that allows the measurement of the information content of procedural programs. Torress and Samadzadeh [39] have shown that software reusability has an inverse relationship with entropy. In all of these works, the usage of entropy to evaluate code complexity is based on Zweben and Halstead's [44] work which shows that "operators" are distributed with a natural probability, hence, can be used to measure a product's information content. Bansiya et al. [1] have used "name strings" to measure Class Definition Entropy (CDE) assuming that all "name strings" represent approximately equal information. All these studies prove that entropy can be successfully applied to measurement of software/software products. On the other hand, the usage of entropy as a complexity metric has not yet been extended and validated for the assessment of XML schema documents written in DTD language.

For measuring complexity of DTDs, several researchers have proposed different criteria. Arnaud Sahuguet [37] have analyzed the typical characteristics of DTDs and have presented count based measures such as number of elements, attributes, ID and IDREF entities etc. Choi et al. [8, 29] have presented some criteria about how DTDs should be designed for ensuring the quality of XML projects and pointed out the challenges in XML design due to DTDs characteristics. Some well-known metrics like LOC, McCabe, Fan-in and Fan-out, Depth of Inheritance Tree (DIT), are also adapted for measuring the complexity of DTDs [26]. An improved version of this paper [26], which includes eleven metrics is produced by McDowell et al.[30]. Mustafa et al. [35] have demonstrated that the XML documents that are generated by the DTD with higher nesting levels have higher weights and are more complicated compared to the documents with lower nesting levels. In this demonstration, various techniques are used to represent XML documents as a regular expression and this is achieved by determining complexity values of regular expression; a tree representation of XML documents and the implementation of Weight Allocation (WA) algorithm.

None of the above-proposed metrics for schema documents written in DTD considers the structural similarities in the DTD components. In our opinion, similarity in the schema components' structure affects the effort required for comprehending the schema document. The more similar structured elements and attributes the schema document has, the easier it is to comprehend and remember the schema document. Our present work is based on this assumption. In the following paragraphs, we will introduce two metrics: "Entropy metric" and "Distinct Structured Element Repetition Scale metric" to measure the complexity of schema documents in Document Type Definition language.

3. Proposed Metrics

3.1. Definitions of E(DTD) and DSERS(DTD) Metric

Davis and LeBlanc [9] have introduced entropy metric for measuring the complexities of programs written in FORTRAN. The program code is partitioned into "chunks". The "chunks" are defined as a group of related items, which can be formed into single mental concepts. In our case, the chunks represent the declarations of elements in a schema document. These components (elements) can be related with the other elements in the form of parent-child relationship. An attribute declaration can also be referred to as a chunk since that attribute declaration may be required as a part of any element declaration in a schema document. A given schema document can be represented by a directed graph where the nodes represent elements and attributes, and the edges represent parent-child relations. The node that has no child nodes is called a leaf node. Hence, our chunks can be represented by a directed graph. We demonstrate the graph representation in the next section.

For the assessment of the structural complexity of a given DTD, the elements appearing in the corresponding directed graph are grouped into the equivalence classes based on their *fan-in, fan-out* and number of attributes. That is, elements that have the same value of *fan-in* and *fan-out* and number of attributes are interpreted as elements having the same structural complexity.

Accordingly, we propose Entropy metric for Document Type Definition language:

$$\mathrm{E(DTD)} = -\sum_{i=1}^{n} \mathrm{P}(\mathrm{C}_i)\log_2 \mathrm{P}(\mathrm{C}_i),$$

where n is the number of equivalence classes.

Entropy of a given schema document having n distinct classes of elements can be calculated using relative frequencies as unbiased estimates of their probabilities

P(C_i), $i = 1, 2, \ldots, n$. The distinct class of elements means that elements having the same structural complexities are grouped in the same class called equivalence class.

The entropy value of the schema document is calculated by considering its equivalence classes. The relative frequency of occurrence of the equivalence classes of the schema document is the number of elements and attributes inside the equivalence class divided by the total number of element and attributes of the schema document. When all elements are placed in the same equivalence class, the minimum entropy value is evaluated. Since there is only one class, *i.e.* $n = 1$, and all elements are grouped in that class, the relative frequency of occurrence of that class, $P(C_1) = 1$ and the entropy value is:

$$\mathrm{E(DTD)} = -\sum_{i=1}^{n} P(C_i) \log_2 P(C_i) = P(C_1)\log_2 P(C_1) = 0.$$

On the other hand, the possible maximum entropy occurs when each element/ attribute in the schema document has different fan-in and fan-out values i.e. each has a different structural complexity value. In such a case, the number of equivalence classes is equal to the number-of nodes in its directed graph representation and P (C_i) is the probability of i^{th} class (*i.e.* the number of elements inside the equivalence class divided by the total number of elements in the Schema document), and equal to $1/n$, where $i = 1, 2, \ldots, n$, and n is the number of nodes in the directed graph or the number of equivalence classes of the schema. The entropy value of the schema, in this case is:

$$\mathrm{E(DTD)} = -\sum_{i=1}^{n} P(C_i) \log_2 P(C_i) = -\sum_{i=1}^{n} (1/n) \log_2 (1/n) = \log_2 n.$$

Our second metric is Distinct Structured Repetition Scale (DSERS), which can be used for measuring the interface complexity of schema documents and can be defined as:

$$\mathrm{DSERS(DTD)} = \sum_{i=1}^{p} \mathrm{de}_i^2 \;/\#\mathrm{e},$$

where p is the number of equivalence classes, de_i is the number of members inside the i^{th} class and #e is the total number of element nodes in the graph representation of DTD.

Possible values for DSERS will be in the range $1 \leq$ DSERS $\leq$ #e. The relation between DSERS and E metric is that while higher DSERS value indicates lower data complexity of schema, the higher E metric value indicates the opposite. In other words, higher DSERS and lower E show that the developer makes less effort to understand the structures of elements due to repetition of similar structured messages.

3.2. Illustration of E(DTD) and DSERS(DTD) Metrics

We demonstrate our E(DTD) and DSERS(DTD) metrics by three example schema documents: `publications.dtd`, `order.dtd` and `tvschedule.dtd` shown in Listing 1, Listing 2 and Listing 3 respectively. The directed graph representations of the three

DTDs are given in Fig. 1, Fig. 2 and Fig. 3. From the directed graph representations of these three DTDs we evaluate the equivalence classes that are given in Listing 4, Listing 5 and Listing 6 respectively.

```
    <!ELEMENT publications
(book|article|conference)*>
<!ELEMENT book
(title, author+, edition, publisher)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (Firstname, Lastname)>
<!ELEMENT Firstname (#PCDATA)>
    <!ELEMENT Lastname (#PCDATA)>
<!ELEMENT edition (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT article ((#PCDATA)>
<!ELEMENT conference
(conftitle, city, year)>
    <!ELEMENT conftitle (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT year (#PCDATA)>
```

Listing 1. The schema document `publications.dtd`.

```
     <!ELEMENT Order (Customer, Part+)>
<!ATTLIST id CDATA #REQUIRED>
<!ELEMENT Customer (Firstname, Lastname, Email)>
<!ELEMENT Firstname (#PCDATA)>
     <!ELEMENT Lastname (#PCDATA)>
<!ELEMENT Email (#PCDATA)> \newline
<!ELEMENT Part (key, ExtendedPrice, Tax, Shipment+)>
<!ELEMENT key (#PCDATA)>
<!ELEMENT ExtendedPrice (#PCDATA)>
<!ELEMENT Tax (#PCDATA)>
<!ATTLIST Part color CDATA #REQUIRED>
<!ELEMENT Shipment (ShipDate, ShipMode, Address)>
<!ELEMENT ShipDate (#PCDATA)>
<!ELEMENT ShipMode (#PCDATA)>
     <!ELEMENT Address(#PCDATA)>
```

Listing 2. The schema document `order.dtd`.

```
     <!ELEMENT TvSchedule (Name,Channel+)>
<!ELEMENT Channel (Banner,Day+)>
     <!ELEMENT Name (#PCDATA)>
<!ELEMENT Banner (#PCDATA)>
<!ELEMENT Day(Date,(Holiday|ProgramslotT+)+)>
<!ELEMENT Holiday (#PCDATA)>
<!ELEMENT Date(#PCDATA)>
<!ELEMENT Programslot (Time, Title, Description?)>
```

```
<!ELEMENT Time (#PCDATA)>
<!ELEMENT Title (Rating, Language)>
     <!ELEMENT Rating (#PCDATA)>
<!ELEMENT Language (#PCDATA)>
     <!ELEMENT Description (#PCDATA)>
<!ATTLIST Channel Chan CDATA REQUIRED>
<!ATTLIST Programslot Vtr CDATA #IMPLIED>
```

Listing 3. The schema document `tvschedule.dtd`.

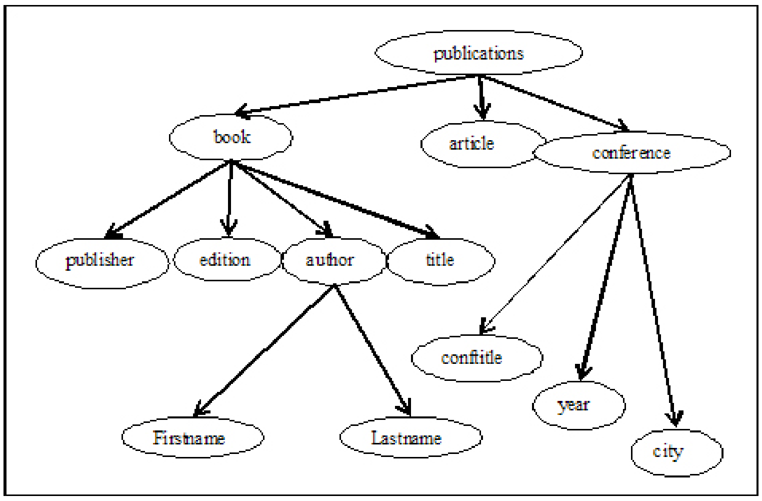

Fig. 1. The directed graph representation of schema document `publication.dtd`.

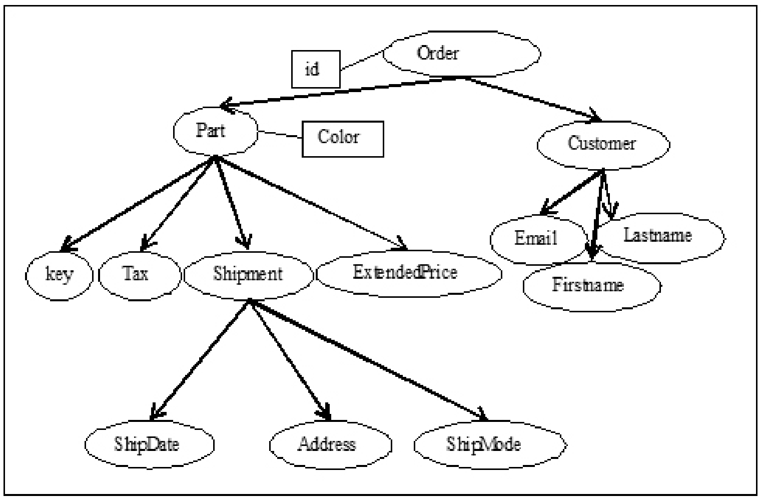

Fig. 2. The directed graph representation of schema document `order.dtd`.

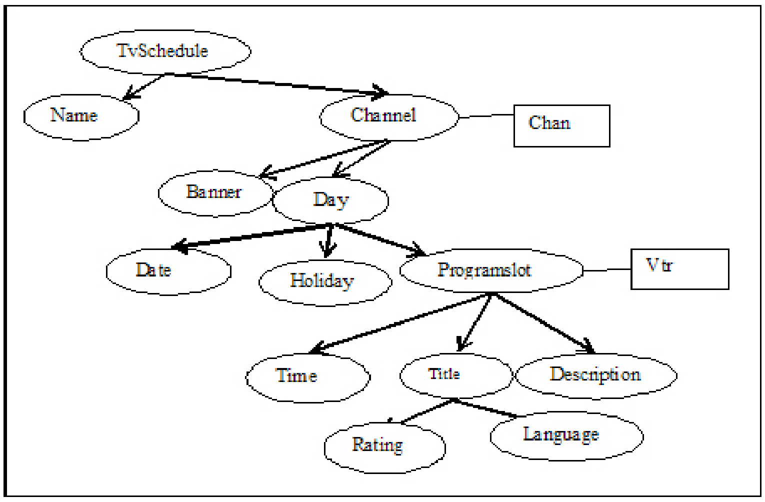

Fig. 3. The directed graph representation of schema document `tvschedule.dtd`.

In order to obtain the equivalence classes of these three example DTDs we group their elements according to the elements' fan-in, fan-out and number of attributes.

C_1 = {publications},
C_2 = {book},
C_3 = {author},
C_4 = {conference},
C_5 = {article, publisher, edition, title, conftitle, year, city, Firstname, Lastname}.

Listing 4. Equivalence classes of the DTD `publications.dtd`.

C_1= {Order},
C_2= {Part},
C_3= {Customer, Shipment},
C_4= {Key, Tax, ExtendedPrice, Email, Firstname, Lastname, Shipdate, Address, ShipMode}.

Listing 5. Equivalence classes of the DTD `order.dtd`.

C_1= {TvSchedule},
C_2= {Channel},
C_3= {Day},
C_4= {Programslot},
C_5= {Title},
C_6={Name, Banner, Language Date, Holiday, Time, Description, Rating }.

Listing 6. Equivalence classes of the DTD `tvschedule.dtd`.

Having grouped the elements of each schema document into equivalence classes now we can calculate E(DTD) and DSERS(DTD) metrics values for them.

The E(DTD) and DSERS(DTD) values for the schema document given in Listing 1 is:

$$\mathrm{E(DTD)}_{\mathrm{publications.dtd}} = -\sum_{i=1}^{5} \mathrm{P(C}_i) \log_2 \mathrm{P(C}_i) =$$
$$= -[(1/13)\cdot \log_2(1/13)+(1/13)\cdot \log_2(1/13) +(1/13)\cdot\log_2(1/13)+$$
$$+(1/13)\cdot\log_2(1/13)+ (9/13)\cdot\log_2(9/13)] = 1.50588$$

$$\mathrm{DSERS}(_{\mathrm{publications.dtd}}) = \sum_{i=1}^{5}\mathrm{de}_i^2 /13 = (1^2+1^2+1^2+1^2+9^2)/13 = 85/13 = 6.53846$$

The E(DTD) and DSERS(DTD) values for the schema document given in Listing 2 is:

$$\mathrm{E(DTD)}_{\mathrm{order.dtd}} = -\sum_{i=1}^{4} \mathrm{P(C}_i) \log_2 \mathrm{P(C}_i)=$$
$$= -[(1/13)\cdot\log_2(1/13)+(1/13)\cdot\log_2(1/13)+$$
$$+(2/13)\cdot\log_2(2/13) + (9/13)\cdot\log_2(9/13)] = 1.35203$$

$$\mathrm{DSERS}(_{\mathrm{order.dtd}}) = \sum_{i=1}^{4}\mathrm{de}_i^2 /13 = (1^2+1^2+2^2+9^2)/13 = 87/13 = 6.69231$$

The *E(DTD)* and *DSERS(DTD)* values for the schema document given in Listing 3 is:

$$\mathrm{E(DTD)}_{\mathrm{tvschedule.dtd}} = -\sum_{i=1}^{6} \mathrm{P(C}_i) \log_2 \mathrm{P(C}_i) =$$
$$= -[(1/13)\cdot\log_2(1/13)+(1/13)\cdot\log_2(1/13)+(1/13)\cdot\log_2(1/13)+$$
$$+(1/13)\cdot\log_2(1/13)+(1/13)\cdot\log_2(1/13)+ (8/13)\cdot\log_2(8/13)] = 1.85429$$

$$\mathrm{DSERS}(_{\mathrm{tvschedule.dtd}}) = \sum_{i=1}^{6}\mathrm{de}_i^2 /13 =$$
$$= (1^2+1^2+1^2+1^2+1^2+8^2)/13 = 69/13 = 5.30769$$

In order to compare the usefulness of E(DTD) and DSERS(DTD) metrics we also calculate the structural complexities of the three schema documents by using the element fanning [25] and tree impurity [33,40] metrics and these values are shown in Table 1. In the third column, the size metric [26] value that is evaluated by counting the number of elements and attributes of the schema document for each DTD file is shown for comparison. The fourth and fifth columns give fanning and tree impurity metrics values. The values residing in the sixth and last column, E(DTD) and DSERS(DTD) are calculated by considering the equivalence classes of the three DTDs.

The element fanning for a given DTD is:

$$\mathrm{Fanning} = e/n,$$

where e is the number of edges and n is the number of nodes in the directed graph of the schema. It can be interpreted that as the fanning value increases so does the complexity of a given schema document. The tree impurity of a given DTD can be calculated as:

$$\mathrm{TI} = [2(e - n + 1)/(n - 1)(n - 2)] \cdot 100\%.$$

where e is the number of edges and n is the number of nodes in the directed graph of the DTD. The tree impurity, (TI), metric is defined by Fenton *et al.* [33] and is used by Visser [40] as a structural complexity measure for the schema documents. The TI indicates the degree of deviation from a tree structure with the same number of nodes and the lower value is a better representation of complexity. From Table 1 it can be observed that the values of structural complexity measures, the fanning, tree impurity and size metric are consistent with each other.

Table 1. The structural complexity values of the example DTDs

DTD Listing No.	**Graph Figure No.**	**Size**	**Fanning**	**TI**	**E(DTD)**	**DSERS(DTD)**
1	1	13	12/13	0%	1.50588	6.53846
2	2	13	12/13	0%	1.35203	6.69231
3	3	13	12/13	0%	1.85429	5.30769

However, E(DTD) metric gives different values for the structural complexities of the three schema documents. It is due to the fact that the E(DTD) metric considers the structural complexities of each element of DTD by grouping these elements into the same equivalence classes. That is, the elements and attributes that have similar structures and which are grouped into the same equivalence classes can be interpreted as having the same complexities. As the diversity in structures increases, so does the number of equivalence classes, which results in more complex schema documents (since understanding and remembering the elements and attributes having different structures becomes more difficult).

The schema documents that exhibit less variety in structure of elements have a lower value of E(DTD) and a higher value of DSERS(DTD) than the schema documents that exhibit greater variety in structure of elements. Hence, higher DSERS(DTD) and lower E(DTD) can be interpreted as a suggestion that the developer makes less effort to understand and remember the structures of elements and dependency between them due to gained familiarity with elements having higher frequency of occurrences. From Table 1 it can easily be observed that, although the three schema DTDs have equal values for the size, fanning and tree impurity metrics, the complexities of them reflected by E(DTD) and DSERS(DTD) metrics are different. Therefore, the E(DTD) and DSERS(DTD) metrics are more realistic methods for measuring the structural complexity of a given schema than those of the others and in terms of the complexity, it can be useful in differentiating and ranking DTDs that have equal number of elements.

4. Validation of E(DTD) and DSERS(DTD) Metrics

In order to check the applicability of E(DTD) and DSERS(DTD) metrics through theoretical validation, we use Weyuker's properties [42]. For empirical validation of these two metrics, we analyze 50 real DTD files downloaded from the web. The statistics collected from these DTDs are shown in Table 2 and the graph based on these statistics is depicted in Figs. 4, 5, 6 and 7. In the following section, we evaluate our DTD metrics through Weyuker's properties.

4.1. Evaluation of E(DTD) and DSERS(DTD) by Weyuker's Properties

Property 1. $(\exists P)(\exists Q)(|P| \neq |Q|)$. *Where P and Q are program body.* This property states that a measure should not rank all programs as equally complex. This property is satisfied since it is obvious that for different schema different E and DSERS values can be calculated.

Property 2. Let c be a non-negative number then there is only a finite number of programs having complexity c. All schema documents consist of only finite number of chunks (i.e. elements and attributes). Now, as the E and DSERS measures depend upon the schema (graph) structure and are calculated by entropy value, some largest possible number can be assumed without harm to be an upper bound. Given that a schema contains only a finite number of elements, there are only finitely many schemas being equal to the upper bound. Hence, both E and DSERS hold this property.

Property 3. *There are distinct programs P and Q such that* $|P| = |Q|$.
This property states that a valid complexity measure allows different schema to have the same complexity value. As pointed out by Weyuker, the only measures that do not satisfy this property are those that assign a unique numerical name to each program and treat this name as a program's complexity. Therefore, it is clear that this property is held by E and DSERS.

Property 4. $(\exists P)(\exists Q)(P \equiv Q \;\&\; |P| \neq |Q|)$.
This property states that even though two programs compute the same function, their program complexities depend upon the implementation details. In order for two DTD documents to generate, the same XML documents should be designed in the same structure since we do not consider the number of occurrences of the elements in the resulting XML documents. That is, the implementation of DTDs should be the same for validation of the same XML document. Therefore, Property 4 cannot be satisfied by both E(DTD) and DSERS(DTD) metrics.

Property 5. $(\forall P)(\forall Q)(|P| \leq |P;Q|$ and $|Q| \leq |P;Q|)$.
Consider the example DTD documents given in Listing 4 and Listing 5. If we combine these two DTDs then the equivalence classes of the resulting DTD are:
C_1= {publications},
C_2= {book},
C_3= {author},

C_4= {conference, Customer, Shipment},
C_5= {article, publisher, edition, title, conftitle, year, city, Key, Tax, ExtendedPrice, Email, Shipdate, Address, ShipMode},
C_6= {Order},
C_7= {Part},
C_8= {Firstname, Lastname}.

Note that the elements Firstname and Lastname are common in both DTDs, hence the *fan-in, fan-out* and number of attributes of these two elements are 2, 0 and 0 respectively in the directed graph of combined DTD. The metrics values are:

$$\mathrm{E(DTD)}_{\mathrm{publications.dtd;order.dtd}} = -\sum_{i=1}^{8} \mathrm{P(C}_i)\ \log_2 \mathrm{P(C}_i) =$$
$$= -\ [(1/24)\cdot\log_2(1/24) + (1/24)\cdot\log_2(1/24) + (1/24)\cdot\log_2(1/24) +$$
$$+\ (3/24)\cdot\log_2(3/24) + (14/24)\cdot\log_2(14/24) + (1/24)\cdot\log_2(1/24) +$$
$$+\ (1/24)\cdot\log_2(1/24) + (2/24)\cdot\log_2(2/24)] = 2.08255$$

$$\mathrm{DSERS}(_{\mathrm{publications.dtd;order.dtd}}) = \sum_{i=1}^{8} \mathrm{de}_i^2\ /24 =$$
$$= (1^2+1^2+1^2+3^2+14^2+1^2+1^2+2^2)/24 = 8.91667$$

In Section 3.2 we found that

$$\mathrm{E(DTD)}_{\mathrm{publications.dtd}} = 1.50588 \ \text{ and } \ \mathrm{E(DTD)}_{\mathrm{order.dtd}} = 1.35203.$$

Since

$$\mathrm{E(DTD)}_{\mathrm{publications.dtd}} < \mathrm{E(DTD)}_{\mathrm{publications.dtd;order.dtd}}$$
$$\text{and}$$
$$\mathrm{E(DTD)}_{\mathrm{order.dtd}} < \mathrm{E(DTD)}_{\mathrm{publications.dtd;order.dtd}}$$

our E(DTD) metric satisfies this property.

In Section 3.2 we found that

$$\mathrm{DSERS}(_{\mathrm{publications.dtd}}) = 6.53846 \ \text{ and } \ \mathrm{DSERS}(_{\mathrm{order.dtd}}) = 6.69231.$$

Since

$$\mathrm{DSERS}(_{\mathrm{publications.dtd}}\ ) < \mathrm{DSERS}(_{\mathrm{publications.dtd;order.dtd}})$$
$$\text{and}$$
$$\mathrm{DSERS}(_{\mathrm{order.dtd}}) < \mathrm{DSERS}(_{\mathrm{publications.dtd;order.dtd}})$$

this property is also satisfied by DSERS(DTD) metric.

Property 6. $(\exists P)(\exists Q)(\exists R)(|P| = |Q|)\&(|P;R| \neq |Q;R|)$.

This property asserts that we can find two DTDs of equal E(DTD) and DSERS(DTD) values which when separately concatenated to a third DTD yield different E(DTD) and DSERS(DTD) values. Let us assume three DTD documents P, Q and R. The first DTD document P.dtd having 4 equivalence classes consisting of similar structured elements and the number of elements in each classes $C_1 = 3$, $C_2 = 3$, $C_3 = 1$, $C_4 = 1$. The second DTD, Q.dtd has elements completely different in structure from the elements of P.dtd. However, the number of equivalent classes and the number of member elements are $C_5 = 3$, $C_6 = 3$, $C_7 = 1$, $C_8 = 1$, which are similar to P.dtd. Please note that the graph representation of Q.dtd and P.dtd are different to each other, but due to similar number of equivalence classes and similar number of member elements, both P.dtd and Q.dtd have same E(DTD) and DSERS(DTD) values. The third schema R.dtd has same graph representation as P.dtd and its classes are: $C_9 = 3$, $C_{10} = 3$, $C_{11} = 1$, $C_{12} = 1$. Note that all three DTD documents; P, Q and R do not have any elements in common *i.e.* their component definitions for their namespaces [23] are different. Now to prove this property, we will examine how complexity values are affected after adding R.dtd to P.dtd and Q.dtd.

The E(DTD) and DSERS(DTD) values for these three documents (P.dtd, Q.dtd and R.dtd) are equal and found as:

$$\mathrm{E(DTD)_P} = \mathrm{E(DTD)_Q} = \mathrm{E(DTD)_R} = -\sum_{i=1}^{4} \mathrm{P}(\mathrm{C}_i)\ \log_2 \mathrm{P}(\mathrm{C}_i) =$$
$$= -\ [(3/8)\cdot \log_2(3/8) + (3/8)\cdot \log_2(3/8) +$$
$$+(1/8)\cdot \log_2(1/8) + (1/8)\cdot \log_2(1/8)] = 1.81128$$
$$\mathrm{DSERS(DTD)_P} = \mathrm{DSERS(DTD)_Q} = \mathrm{DSERS(DTD)_R} = \sum_{i=1}^{4} \mathrm{de}_i^2\ /8 =$$
$$= (3^2+3^2+1^2+1^2)/8 = 2.5$$

If the DTD documents P and R are combined then the combined DTD has four classes since these two DTDs have the same graph representation with no common nodes. Hence, the number of resulting equivalence classes with their member elements will be $C_1 = 6$, $C_2 = 6$, $C_3 = 2$, $C_4 = 2$ and:

$$\mathrm{E(DTD)_{(P;R)}} = -\sum_{i=1}^{4} \mathrm{P}(\mathrm{C}_i)\ \log_2 \mathrm{P}(\mathrm{C}_i) =$$
$$= -\ [(6/16)\cdot \log_2(6/16) + (6/16)\cdot \log_2(6/16) +$$
$$+\ (2/16)\cdot \log_2(2/16) + (2/16)\cdot \log_2(2/16)] = 1.81128$$
$$\mathrm{DSERS(DTD)_{(P;R)}} = \sum_{i=1}^{4} \mathrm{de}_i^2\ /16 = (6^2+6^2+2^2+2^2)/16 = 5$$

If the DTD documents Q and R are combined then the number of resulting classes will be eight since these DTDs also have fully different structured elements. The classes and member elements' counts of combination of Q and R will be $C_1 = 3$, $C_2 = 3$, $C_3 = 1$, $C_4 = 1$, $C_5 = 3$, $C_6 = 3$, $C_7 = 1$, $C_8 = 1$ and:

$$\mathrm{E(DTD)_{(Q;R)}} = -\sum_{i=1}^{8} \mathrm{P}(\mathrm{C}_i)\ \log_2 \mathrm{P}(\mathrm{C}_i) =$$
$$= -\ [(3/16)\cdot \log_2(3/16) + (3/16)\cdot \log_2(3/16) + (1/16)\cdot \log_2(1/16) +$$
$$+\ (1/16)\cdot \log_2(1/16) + (3/16)\cdot \log_2(3/16) + (3/16)\cdot \log_2(3/16) +$$
$$+\ (1/16)\cdot \log_2(1/16) + (1/16)\cdot \log_2(1/16)] = 2.81128$$

$$\mathrm{DSERS(DTD)}_{(\mathrm{Q;R})} = \sum_{i=1}^{8} \mathrm{de}_i^2 \,/16 = (3^2{+}3^2{+}1^2{+}1^2{+}3^2{+}3^2{+}1^2{+}1^2)/16 = 2.5$$

Since

$$\mathrm{E(DTD)}_{(\mathrm{P;R})} \neq \mathrm{E(DTD)}_{(\mathrm{Q;R})}$$
and
$$\mathrm{DSERS\ (DTD)}_{(\mathrm{P;R})} \neq \mathrm{DSERS(DTD)}_{(\mathrm{Q;R})}$$

this property is satisfied by our E(DTD) and DSERS(DTD) metrics.

Property 7. There are program bodies P and Q such that Q is formed by permuting the order of the statement of P and $(|P| \neq |Q|)$.

If we change the place of the definitions of elements and attributes in the DTD document E(DTD) and DSERS (DTD) metrics values for the resulting DTD will not change. Although SE and DSERS metrics for XSD satisfy this property E(DTD) and DSERS(DTD) do not. This is due to the fact that in XSD we can define any local element or attribute as global element or attribute and make them reusable components which results in different *fan-in* and *fan-out* values in the directed graph representation of XSD. However, this is not the case for DTD. In order to confirm the same XML document, DTD documents should be designed in the same manner and hence element's design style in DTD cannot be changed. As a result, both DSERS(DTD) and E(DTD) metrics do not satisfy this property.

Property 8. The values of E(DTD) and DSERS(DTD) are real numbers so renaming of DTD cannot change the values of both metrics. Hence, these two metrics clearly do adhere to this property.

Property 9. $(\exists P)(\exists Q)(|P| + |Q|) < (|P;Q|)$. By using the same example given for Property 5 this property is not satisfied by E(DTD) metric since we find that:

$$\mathrm{E(DTD)}_{\mathrm{publications.dtd;order.dtd}} < \mathrm{E(DTD)}_{\mathrm{publications.dtd}} + \mathrm{E(DTD)}_{order.dtd}$$
$$2.08255 < 1.50588 + 1.35203$$

DSERS(DTD) metric does also not satisfy this property since :

$$\mathrm{DSERS}(_{\mathrm{publications.dtd;order.dtd}}) < \mathrm{DSERS}(_{\mathrm{publications.dtd}}) + \mathrm{DSERS}(_{\mathrm{order.dtd}})$$
$$8.91667 < 6.53846 + 6.69231$$

In this section, we have validated E(DTD) and DSERS(DTD) metrics through Weyuker's properties and found that the proposed measures satisfy six Weyuker's properties out of nine, which established E(DTD) and DSERS(DTD) measures as well structured ones.

4.2. Empirical Validations of E(DTD) and DSERS(DTD) Metrics

For empirical validation of these two metrics, we analyze 50 real DTD files from web. Data evaluated after analyzing these files is given in Table 2.

Table 2. Size, Fanning, TI, E(DTD), DSERS(DTD) metrics values for analyzed DTD files. Last column references web addresses of analyzed DTDs

ID	Size	Fanning	TI	E(DTD)	DSERS(DTD)	REF
1	4	0.75	0	0.81	2.50	[17]
2	4	0.75	0	0.75	1.50	[24]
3	4	0.75	0	1.00	1.00	[24]
4	6	1.50	0.4	2.58	1.00	[17]
5	6	0.83	0	2.25	1.33	[24]
6	7	1.14	0.13	2.81	1.00	[24]
7	7	0.86	0	2.52	1.29	[16]
8	7	1.00	0.06	1.66	2.71	[17]
9	7	1.43	0.27	1.66	2.71	[18]
10	8	0.88	0	1.75	2.75	[20]
11	8	1.00	0.02	2.25	1.75	[24]
12	8	1.13	0.095	2.41	1.75	[24]
13	8	1.00	0.04	1.55	3.50	[16]
14	8	1.75	0.33	3.00	1.00	[17]
15	10	0.90	0	2.72	1.60	[24]
16	10	0.90	0	1.57	4.20	[20]
17	10	0.90	0	1.72	3.40	[20]
18	10	0.90	0	1.57	4.20	[20]
19	11	0.91	0	2.04	3.18	[24]
20	11	0.91	0	1.68	3.91	[20]
21	11	0.91	0	0.78	6.27	[20]
22	11	1.09	0.04	1.67	4.82	[17]
23	11	2.09	0.03	2.55	2.27	[18]
24	11	1.09	0.04	1.62	4.27	[19]
25	12	1.08	0.04	1.58	5.67	[17]
26	12	0.92	0	1.21	7.00	[20]
27	13	1.00	0.02	1.61	4.85	[20]
28	13	0.92	0	1.57	5.15	[17]
29	13	0.85	– 0.02	2.65	2.69	[24]
30	13	1.08	0.03	1.82	4.69	[19]
31	13	0.92	0	2.51	3.23	[19]
32	13	1.07	0.03	1.51	6.54	[17]
33	14	0.93	0	1.73	4.71	[20]
34	14	1.00	0.01	3.09	2.00	[18]
35	15	0.93	0	2.87	2.60	[24]
36	15	0.93	0	1.24	8.47	[24]
37	16	0.88	– 0.01	2.35	3.50	[20]
38	16	1.00	0.09	1.63	7.00	[17]
39	16	0.94	0	1.19	9.38	[24]
40	17	0.94	0	1.57	7.00	[20]
41	17	0.94	0	0.95	11.71	[24]
42	17	1.12	0.03	1.85	6.65	[17]
43	17	1.24	0.04	2.13	5.59	[17]
44	18	0.94	0	0.91	12.67	[24]
45	18	1.39	0.06	3.17	2.22	[24]
46	20	2.10	0.13	3.42	1.90	[21]
47	20	1.35	0.04	4.01	1.60	[18]
48	32	1.34	0.03	2.18	12.06	[19]
49	32	1.25	0.02	2.32	9.88	[17]
50	32	1.19	0.02	2.52	7.63	[22]

Note that the data given in Table 2 is ordered by size metric values. The graphs that depict the comparison results of E(DTD) metric with size, fanning and TI metrics are shown in Figs. 4, 5 and 6 respectively.

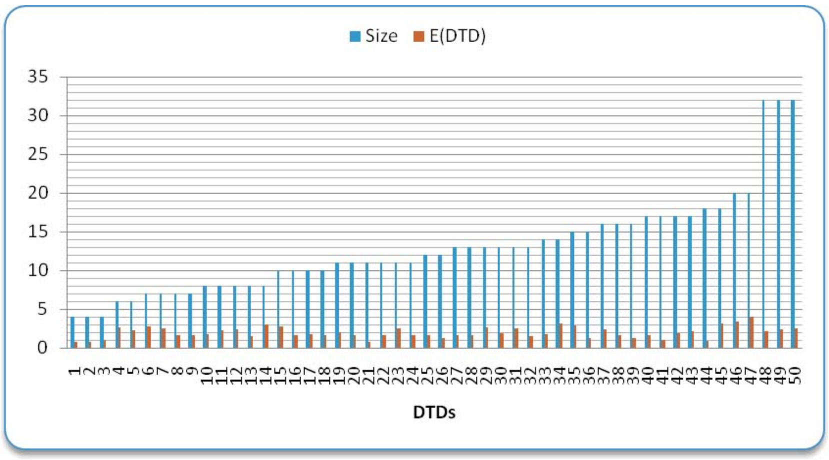

Fig. 4. E(DTD) vs. Size metrics results. The data is ordered by Size metric values for analyzed DTDs.

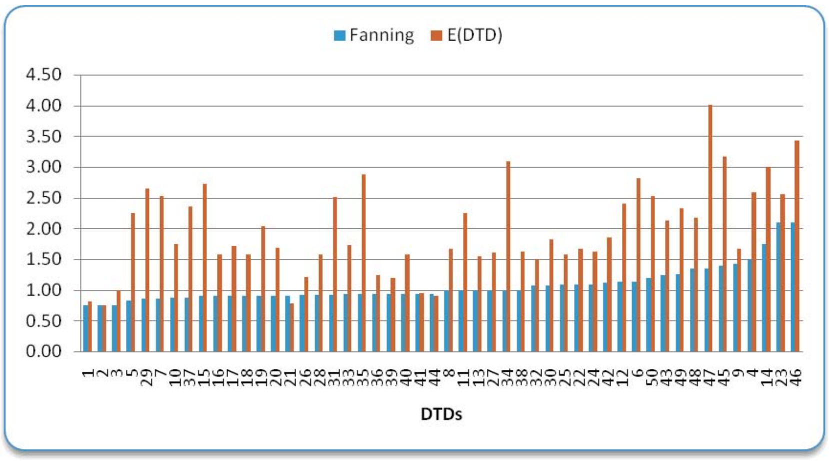

Fig. 5. E(DTD) vs. Fanning metrics results. The data is ordered by Fanning metric values for analyzed DTDs.

From Table 2 and Fig. 4 it can be observed that E(DTD) metric evaluates different values for analyzed DTDs that have the same Size metric values. This is due to the fact that the Size metric ignores the complexity of DTD caused by frequencies of similar structured elements defined in DTD. The same point is also missed by TI and Fanning metrics. Furthermore, having a negative value for TI is not an indicator of a good metric. As it can be seen from Fig. 6, most of the DTDs have 0% of TI values whatever their sizes. From this point of view, TI metric fails in differentiating

DTDs in terms of their complexities. The last graph shown in Fig. 7 depicts the comparison result between E(DTD) and DSERS(DTD) metrics. Since these two metrics have inverse relation, lower E(DTD) and higher DSERS(DTD) values have the same meaning: lower psychological complexity of DTD. The values of these two metrics are not in contradiction as can be observed from Fig. 7.

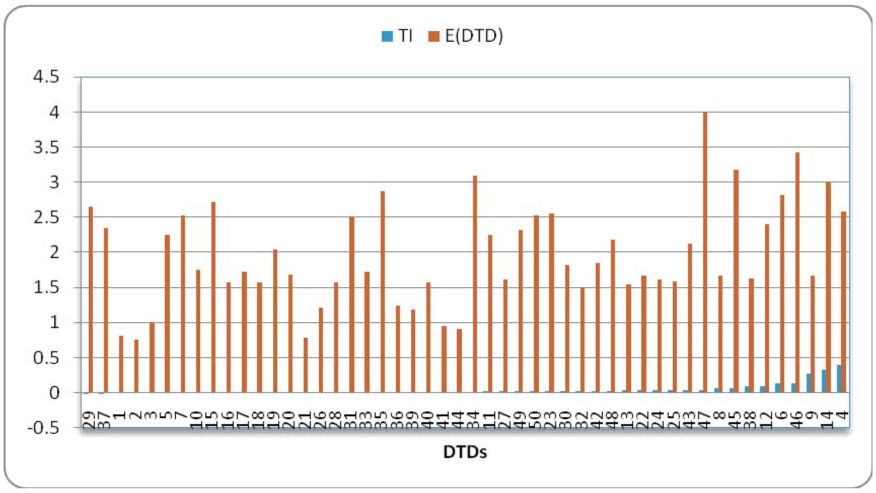

Fig. 6. E(DTD) vs. TI metrics results. The data is ordered by TI metric values for analyzed DTDs.

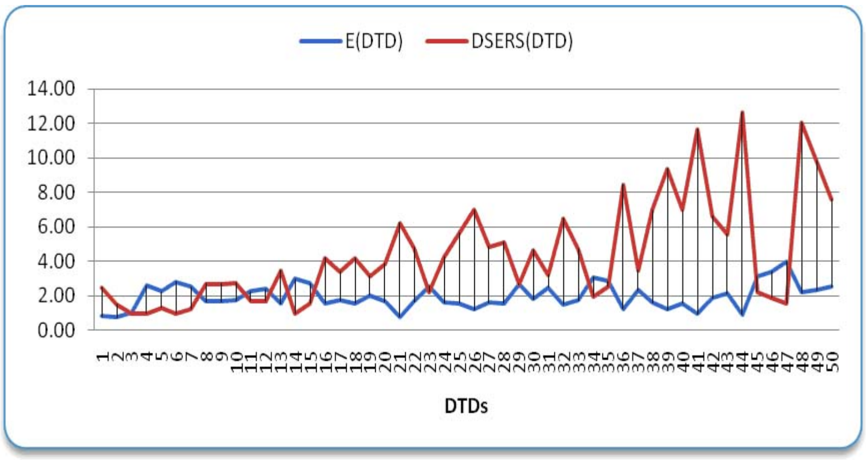

Fig. 7. DTD vs. DSERS(DTD) metrics results.

From these observations, we can conclude that our proposed DTD metrics are able to differentiate DTDs that have equal number of elements and can be useful to provide feedback about psychological complexity of DTD.

5. Concluding Remark on E(DTD) and DSERS(DTD) Metrics

We have presented the complexity metrics for measuring the structural complexity of a given XML schema document written in W3C Document Type Definition language. It is found that measuring the complexity of the XML schema documents by entropy metric E(DTD) and DSERS(DTD) is more realistic. The E(DTD) and DSERS(DTD) metrics calculate the structural complexity of DTDs by considering the fact that the schema documents, which have less diversity in elements, are less complex in comparison to the others that have greater diversity. In other words, the presented approach has exhibited a better representation of structural complexity of a given schema document. In order for the proposed metric to be reliably applied for the assessment of the XML schema documents written in DTD, it should also be empirically validated. The empirical validations of the proposed metrics have been carried out by collecting 50 DTD documents from the web and comparing the values of the newly proposed metrics with the values of other metrics for these DTDs. From empirical validation results, we observed that newly proposed DTD metrics evaluate different complexity values for DTDs that have equal element declarations measured by Size metric. Hence, we can conclude that our metrics can be useful in differentiating DTDs, which have the same size.

6. Future Work

As future work, we aim to develop metrics based on the grammatical context of the XML schema documents written in W3C XML Schema and DTD. Another work to be handled in the future can be the evaluation of the cognitive complexity of the XML schema documents for XSD and DTD. Since XML has been also used by databases, we are planning to develop a criterion to evaluate and maintain the quality of the XML enabled database in future.

References

[1] Bansiya J., Davis C., Etzkorn L., *An Entropy Based Complexity Measure for Object-Oriented Designs*, Theory and Practice of Object Systems, **5**, 2, pp. 1–9 , 1999.

[2] Boxall M., Araban S., *Interface Metrics for Reusability Analysis of Components, Proceedings of the 2004 Australian Software Engineering Conference (ASWEC'04)*, Melbourne, Australia, pp. 1–40, 2004.

[3] Basci D., Misra S., *Entropy Metric for XML DTD Document*, ACM SIGSOFT Software Engineering Notes, **33**, 4, pp. 1–6, 2008.

[4] Basci D., Misra S., *A W3C Document Type Definition (DTD) Metric*, *Proceedings of Sixth International Workshop on SOA and Web Practices, OOPSLA 2008*, pp. 1–10, 2008.

[5] Basci D., Misra S., *Measuring and Evaluating a Design Complexity Metric for XML Schema Documents*, Journal of Information Science and Engineering, **25**, 5, pp. 1405–1425, 2009.

[6] Cerami E., *Web Services Essentials, Distributed Applications with XML-RPC, SOAP, UDDI & WSDL*, O'Reilly Publishers, 2002.

[7] Chapin N., *An Entropy Metric For Software Maintainability System Sciences*, *Proceedings of the Twenty-Second Annual Hawaii International Conference Vol. II: Software Track*, pp. 522–523, 1989.

[8] Choi B., *A Few Tips for Good XML Design. Technical report, University of Pennsylvania*, available at: `http://db.cis.upenn.edu/~kkchoi/DTDI2/`

[9] Davis J., Leblanc R., *A Study of the Applicability of Complexity Measures*, IEEE Transactions on Software Engineering, **14**, 9, pp. 1366–1372, 1998.

[10] Edward B. A., Taghi M. K., Ye C., *Measuring Coupling and Cohesion of Software Modules: An Information-Theory Approach*, *Proceedings of Seventh International Software Metrics Symposium* (metrics'01), pp. 124–137, 2001.

[11] Edward B. A., Sampath G., Govindarajan R., *Measuring Size, Complexity, and Coupling of Hypergraph Abstractions of Software: An Information-Theory Approach*, Software Quality Control, **15**, 2, pp. 179–212, 2007.

[12] Erl T., *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*, Prentice Hall Publishers, 2004.

[13] Etzkorn L., Gholston S., Hughes W. E. Jr., *A Semantic Entropy Metric*, Journal of Software Maintenance and Evolution, **14**, 4, pp. 293–310, 2002.

[14] Gaffney J., *Instruction Entropy, a Possible Measure of Program/Architecture Compatibility*, ACM SIGMETRICS Performance Evaluation Review, **12**, 4, pp. 13–18, 1984.

[15] Harrison W., *An Entropy-Based Measure Of Software Complexity*, IEEE Transactions on Software Engineering, **18**, 11, pp. 1025–1029, 1992.

[16] `http://docbook.sourceforge.net/release/dsssl/current/dtds/`

[17] `http:// java.sun.com/dtd/`

[18] `http://struts.apache.org/dtds/`

[19] `http://jonas.objectweb.org/dtds/`

[20] `http://www.ncbi.nlm.nih.gov/dtd/`

[21] `http://www.cs.helsinki.fi/group/doremi/publications/XMLSCA2000.html`

[22] `http://www.pramati.com/dtd/`

[23] `http://www.w3.org/TR/REC-xml-names/`

[24] `http://www.omegahat.org/XML/DTDs/`
`http://www.openmobilealliance.org/Technical/dtd.aspx`
`http://fisheye5.cenqua.com/browse/glassfish/update-center/dtds/`
`http://www.python.org/topics/xml/dtds/`
`http://www.okiproject.org/polyphony/docs/raw/dtds/`

[25] Henry S., Kafura D., *The Evaluation of Software Systems' Structure Using Quantitative Software Metrics*, Software Practice and Experience, **14**, 6, pp. 561–573, 1984.

[26] Klettke M., Scneider L., Heuer A., *Metrics for XML Document Collections*, *Proceeding of XMLDM Workshop*, Czech Republic, pp. 162–176, 2002.

[27] Kim K., Shin Y., Wu C., *Complexity Measures for Object-Oriented Program Based on the Entropy*, *Proceedings of Second Asia-Pacific Software Engineering Conference* (apsec'95), pp. 127–136, 1995.

[28] Lee D., Chu W., *Comparative Analysis of Six XML Schema Language*, ACM SIGMOD Record, **29**, 3, pp. 1–12, 2000.

[29] Lin Z., He B., Choi B., *A Quantitative Summary of XML Structures*, Lecture Notes in Computer Science, **4215**, Springer-Verlag, pp. 228–240, 2006.

[30] McDowell A., Schmidt C., Yue K., *Analysis and Metrics of XML Schema*, *Proceedings of the International Conference on Software Engineering Research and Practice*, CSREA Press, pp. 538–544, 2004.

[31] McCabe T. J., *A Complexity Measure*, IEEE Transactions on Software Engineering, **2**, 1, pp. 308–320, 1976.

[32] Mohanty S. N., *Entropy Metrics for Software Design Evaluation*, The Journal of Systems and Software, **2**, pp. 39–46, 1981.

[33] Fenton N. E., *Software Metrics – A Rigorous Approach*, Chapman & Hall, London, 1991.

[34] Newcomer E., Lomow G., *Understanding SOA with Web Services*, Addison Wesley Professional, 2004.

[35] Mustafa H. Q., Smadzadeh M. H., *Determining the Complexity of XML Documents*, *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)*, Vol. **2**, pp. 416–421, 2005.

[36] Salwa K., Hafiz A. E., *Entropies as Measures of Software Information*, *Proceedings of 17th IEEE International Conference on Software Maintenance* (ICSM'01), pp. 110–121, 2001.

[37] Sahuguet A., *Everything You Ever Wanted to Know About DTDs, But Were Afraid to Ask*, Lecture Notes in Computer Science, 1997, pp. 171–183, 2000.

[38] Shannon C. E., Weaver W., *The Mathematical Theory of Communication*, Urbana, IL: University of Illinois Press, 1949.

[39] Torres W. R., Samadzadeh M. H., *Software Reuse and Information Theory Based Metrics*, *Proceedings of Applied Computing*, IEEE CS Press, pp. 336–345, 1991.

[40] Visser J., *Structure Metrics for XML Schema*, *Proceedings of XATA:* 2006, pp. 1–10, 2006.

[41] Weerawarana S., Curbera F., Leymann F., Storey T., Ferguson D. F., *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*, Prentice Hall Publishers, 2005.

[42] Weyuker E., *Evaluating Software Complexity Measures*, IEEE Transactions on Software Engineering, **14**, 9, pp. 1357–1365, 1998.

[43] Yuming Z., Baowen X., *Measuring Structural Complexity For Class Diagrams: An Information Theory Approach*, *Proceedings of the 2005 ACM symposium on Applied computing SAC'05*, pp. 1679–1683, 2005.

[44] Zweben S., Halstead M., *The Frequency Distribution of Operators in PL/1 programs*, IEEE Transactions on Software Engineering, **3**, 2, pp. 91–95, 1979.