Metrics for XML Document Collections

Meike Klettke* Lars Schneider
Andreas Heuer

Database Research Group
University of Rostock
Germany
meike@informatik.uni-rostock.de

Abstract

In this paper, several metrics for XML document collections are enumerated and their applications are subsequently discussed.

1 Introduction

The quality of software products or of the software design process can be evaluated. Therefore, several characteristics are set up and metrics are established that can be applied to software.

In the field of XML, comparable methods don't exist. An evaluation of the quality is necessary because we often recognise an ad-hoc approach. The characteristics well-formed and valid are not sufficient to evaluate XML documents; these characteristics are only prerequisites for assessing the quality.

In this article, we develop five metrics that can be used to evaluate the schema of XML documents. Thereby, the characteristics usability and maintainability are evaluated.

The paper is organised as follows. We start with an overview on related work and then introduce methods for evaluating software in section 3. In section 4, metrics for DTDs are enumerated. Applications that can use the metrics are shown in section 5. Section 6 provides two detailed examples. In section 7, we give a conclusion.

2 Related work

Many publications and results on software metrics are available. A general overview of that research field is given in [Fen91]. An extensive collection of articles dealing with software metrics is available in [Dum02]. Horst Zuse

*Supported by the German ResearchAssociation (DFG), He 1768/7-1

and Katrin Drabe collected and classified more that 1600 different metrics in the (ZDMIS - measure information system) [ZD01].

More than 200 metrics developed for object-oriented systems exist. The metrics evaluate the class structure, the methods, and the inheritance hierarchy. [CK94] and [Hog97] give an overview of metrics for object-oriented systems.

The paper [GJP00] deals with metrics for the conceptual model of databases and the quality feature usability is evaluated. Size and complexity of Entity-Relationship models are thereby estimated. Several metrics are suggested in the article and a summing-up of all metrics to a total value is submitted.

Although the problem is relevant within the context of XML, only a few papers exist that deal with quality characteristics and metrics. Some work was done in the project DTD inquisitor. The paper "Everything You Ever Wanted To Know About DTDs But Were Afraid To Ask" from Arnaud Sahuguet [Sah00] was the first to describe typical characteristics of DTDs and to enumerate some typical measures. Thereby, the number of occurring elements, attributes, and entities and ID and IDREF attributes are counted. A graphical representation of DTDs is used to visualise the structure.

An extension of this paper is "DTD Mining Labor Day Report" from Byron Choi and Arnaud Sahuguet [CS00]. In this paper, you can find a more detailed description of 60 DTDs and their typical occurring patterns. Typical errors in DTDs are classified. The paper gives an overview of the number of attributes, elements and entities of the DTDs. Furthermore, it determines how often each content model of the element declaration (#PCDATA, EMPTY, ANY, sequence, mixed content, alternative, ...) occurs.

The article "A Few Tips for Good XML Design" from Byron Choi [Cho00] draws conclusions from the occurring DTD characteristics and suggests good designs for DTDs. This article enumerates for some typical measures how they influence the characteristics Size, Clarity, Parsing, Navigation, Mapping, Flexibility, and Modularisation. Thereby, the article shows that design decisions are not trivial because there are characteristics that are contradictory. For instance, using abbreviations on tags reduces the size but decreases the clarity. The decomposition of large expressions improves the clarity but makes the navigation through the documents more expensive.

These three articles analyse DTDs from an "XML point-of-view". In this paper, we want to follow another method. We will adapt common metrics used for software to determine characteristics of DTDs. Sometimes, we will come to the similar results; some measures and metrics are obvious. Additionally, we will derive further metrics for DTDs.

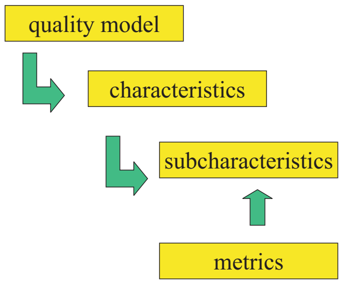

Figure 1: Correlation between software quality and metrics

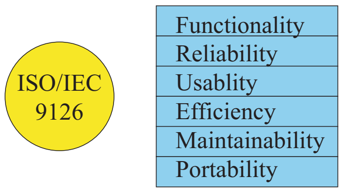

Figure 2: characteristics of the ISO 9126

3 Software quality and software metrics

The quality of a software product is defined through a set of characteristics that have to be fulfilled. These characteristics are determined in the ISO 9126 standard (figure 2). All characteristics enumerated in this standard contain subcharacteristics. These subcharacteristics are also specified in the ISO 9126. For example, the characteristic "'maintainability"' has four subcharacteristics: analysability, changeability, stability, and testability.

To check the fulfillment of the subcharacteristics, metrics can be developed and applied. The aim of the metrics is to measure and evaluate the software quality. Figure 1 visualise the cooperation between the components.

The correlation between metrics and subcharacteristics is not a 1:1 relationship. Some metrics can be used to determine several subcharacteristics, for instance, metrics that evaluate the complexity of software influence the characteristics usability, testability, and analysability, etc.

The metrics that have been developed for software can be classified as follows:

- product metrics
  (the metrics evaluate a software product)
- process metrics
  (the metrics evaluate the design process)
- resources metrics
  (the metrics appraise the required resources).

If we want to adapt the metrics for XML, then we have to concentrate on product metrics. These metrics can be applied to evaluate a static DTD.

4 Metrics for XML documents

We will not develop our own quality model for evaluating DTDs, but rather use the quality model for software and the characteristics and subcharacteristics defined in the ISO 9126. Therefore, in this article we concentrate on metrics for evaluating the characteristics *usability* and *maintainability* and use product metrics.

Efficiency and *functionality* can be proven by applying resources metrics. Therefore, we evaluate document size, and determine the time that is necessary to process documents with XPath, and analyse XML documents with DOM or SAX.

In this section, we will concentrate on product metrics. We will adapt metrics normally applied to software to DTDs. Although XML schema is a recommendation of the W3C, we use DTDs because many XML documents are defined with a DTD. Furthermore, all element and attribute declarations of a DTD can also be represented with XML schema.

We now introduce a DTD for storing publications (figure 3) as a running example for illustrating all metrics. This sample DTD is compact and yet quite complex. Figure 4 shows a graph representing the DTD from figure 3. You can see in this visualisation that the textual representation of the sample DTD doesn't look difficult, but the representation as a graph looks a bit more complex.

In this section, we develop five metrics. All metrics exploit the graph representation.

Size A widely accepted and applied software metric is the LOC (lines of code). There exist different facilities to determine this metric. It is possible to count the lines with

- executable code
- executable code and data definitions or
- executable code, data definitions, and comments.

We want to adapt this metric for DTDs. It would thereby be very simple to count only the number of lines of a DTD or XML Schema Description. However, a metric that counts all relevant declarations is a bit more reliable. Different facilities exist according to the LOC metric. We can evaluate the

- number of elements and attributes,
- number of elements, attributes, and entities or
- number of elements, attributes, entities, and notations

in the metric. The following calculation is the simplest facility to determine the size of a DTD:

$$Size(DTD) = n_{EL} + n_A$$

```
<!ELEMENT publications
  (book | article | conference)*>
<!-- book -->
<!ELEMENT book (front, body, references)>
<!ELEMENT front
  (title, author+, edition, publisher)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (first, second, email?)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT second (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT edition (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT body (part+ | chapter+)>
<!ELEMENT part (ptitle, chapter+)>
<!ATTLIST part id ID #REQUIRED>
<!ELEMENT ptitle (#PCDATA)>
<!ELEMENT chapter (ctitle, section+)>
<!ATTLIST chapter id ID #REQUIRED>
<!ELEMENT ctitle (#PCDATA)>
<!ELEMENT section (stitle, paragraph+)>
<!ATTLIST section id ID #REQUIRED>
<!ELEMENT stitle (#PCDATA)>
<!ELEMENT paragraph (#PCDATA)>
<!ELEMENT references (publications*)>
<!ATTLIST references reftype (book | article
  | conferences | wwwaddress) "article">
<!-- article -->
<!ELEMENT article (meta, body, references)>
<!ELEMENT meta (author+, title, conference)>
<!-- conference -->
<!ELEMENT conference
  (editor+, conftitle, city, year)>
<!ELEMENT editor (first, second, email?)>
<!ELEMENT conftitle (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT year (#PCDATA)>
```

Figure 3: Book-DTD Example

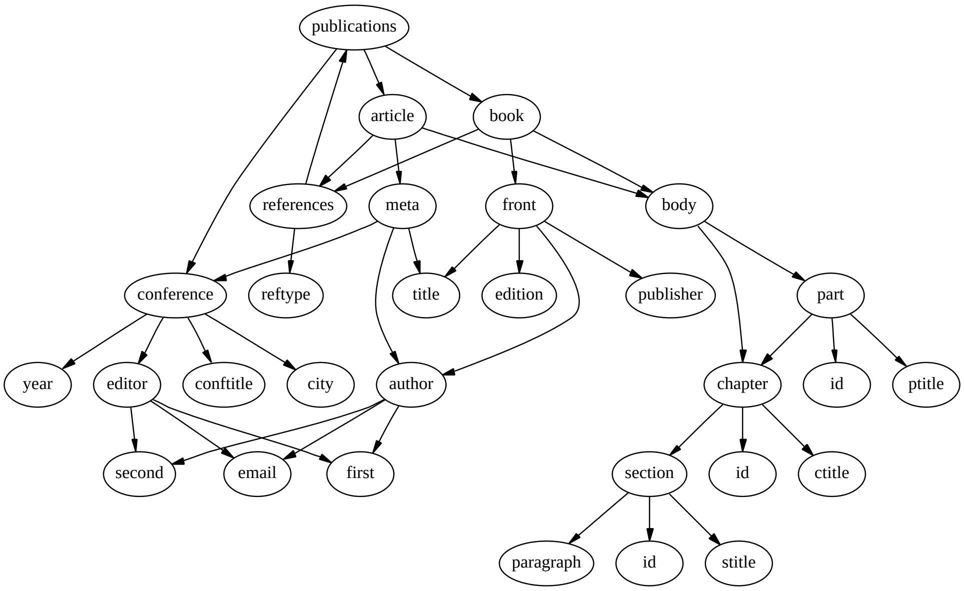

Figure 4: DTD graph for the sample DTD

n_{EL} – number of elements
n_A – number of attributes

For our running example, we can calculate the size of a DTD corresponding to this metric. The result is 30 (26 elements and 4 attributes). It is easy to recognise that the metric delivers the number of nodes in the DTD graph as a result.

Structure complexity In the previous paragraph, we introduced the metric *Size* that is based on the number of elements and attributes of the DTD. Another metric evaluates the complexity of the DTD structure. Subsequently, we adapt the McCabe measure [McC76] developed for evaluating the control flow of software. Therefore, we add additional edges in our DTD graph if quantifiers + and * occur and if mixed content elements (but not #PCDATA) exist. These extensions of the graph are according to the treatment of repetitions in software modules. Figure 5 shows the DTD graph for determining the complexity. All additional edges and all same content elements are marked by light blue colored edges in figure 5.

The complexity of the DTD structure is derived from the graph representing the DTD.

$$Compl(DTD) = e - n + 1$$

e – number of edges in the DTD graph
n – number of nodes in the DTD graph

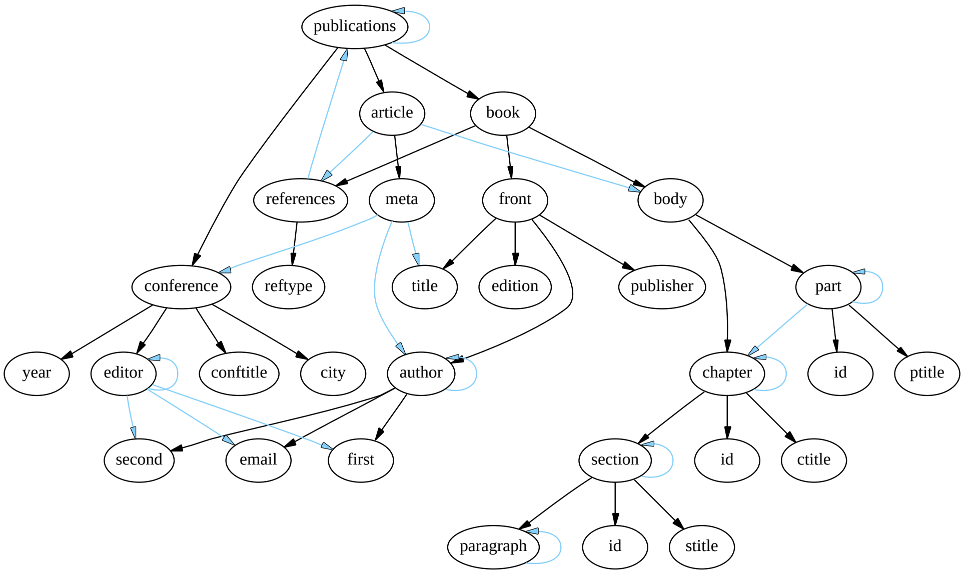

Figure 5: Structure complexity of the DTD

If we determine this metric for the running example, then the result is that the number of nodes in the DTD graph is 30, and 46 edges exist in the graph. Therefore, the structure complexity of the DTD is 17. This value is equal to the number of colored edges in the DTD graph.

Other factors increasing the complexity are the attributes of type IDREF and IDREFS. This information cannot be derived from the graph because you can't derive which ID occurs in the IDREF attribute from a DTD. Therefore, we have to extend the calculation of the structure complexity as follows:

$$Compl(DTD) = e - n + 1 + n_{IDREF}$$

e – number of edges in the DTD graph
n – number of nodes in the DTD graph
n_{IDREF} – number of IDREF and IDREFS attributes

If the DTD describes a tree (which always has n nodes and $n-1$ edges), then we get the value 0 as a result of the structure complexity. Every recursion, every same content, all quantifiers + and *, and all IDREF attributes increase the complexity.

Structure depth For determining the size and the complexity of a DTD, we used a graph representation of the DTD. From the same representation, we can derive further metrics.

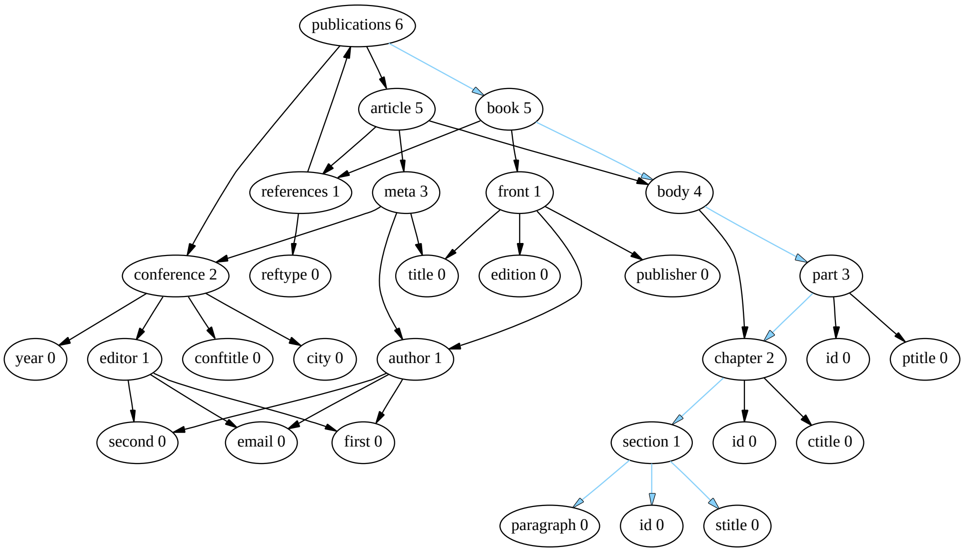

Figure 6: Maximal depth of the DTD

The depth of the DTD graph is another metric that also delivers information about the complexity of the schema. This metric is also applied for the class structure of object-oriented systems [CK94].

For our sample DTD in figure 6, we have the structure depth 6.

How can we determine the depth of the graph? First of all, we have to eliminate recursions, or the result for the depth would be infinite. Then, we determine the depth of each node as follows: The depth of the leaf nodes on the graph is 0. The depth of all other nodes is the maximal depth of all child nodes +1. The depth of the whole graph is equal to the depth of the root node.

The following calculation can be applied for determining the depth of the nodes:

$$Depth(n) = \begin{cases} 0 & : \; n \text{ is leaf node} \\ max(Depth(n_i)) + 1 & : \; n_i \text{ is child node of } n \end{cases}$$

In figure 6, the depth for each element and attribute in the graph is visualized. Additionally, the path in the graph that causes the maximal depth of the root node is marked with blue colored edges.

It is not easy to evaluate the results of this metric. One way would be to state that DTDs with a depth much higher than seven are complex. This evaluation is carried out according to the often cited study [Mil56] which specifies that humans have difficulties dealing with more than seven plus/minus two objects.

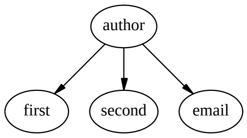

Figure 7: Fan-In metric for the element `author`

Fan-In metric Another metric that can be derived from the graph presentation is the Fan-In metric. This metric doesn't describe the depth of an element declaration but it determines the "width" of each element declaration. For each element declaration this metric expresses how many child elements and attributes it has. This value correlates with the number of child nodes in the DTD graph.

The metric is similar to the metric developed from Henry/Kafura [HK81] for evaluating the class hierarchies of object-oriented systems.

We can determine the metric for each node of the DTD graph as follows:

$$Fan - In(n) = |\{n_i | n_i \text{ is child node of } n\}|$$

Figure 7 demonstrates the metric Fan-In for the node `author`. The result for this metric is 3.

What does this metric express?
Elements with a high Fan-In value are more complex than other elements. Here, we can also apply the "magic seven" introduced from George A. Miller in [Mil56] for evaluating the metric. Elements which have a Fan-In value much higher than seven are difficult to understand and to use.

The Fan-In metric for a graph is equal to the maximal Fan-In value for all nodes. A high Fan-In value specifies that the schema description is complex.

Fan-Out metric The counterpart of the Fan-In metric is the Fan-Out metric. This metric counts how often every element occurs in the DTD declaration, or how often an element declaration is reused. This measure can be derived from the DTD graph, too. Subsequently, we determine the number of the direct parent nodes of every node in the graph according to the following formula:

$$Fan - Out(n) = |\{n_i | n_i \text{ is parent node of } n\}|$$

Figure 8 introduces this measure.

For the element `author` the Fan-Out metric has the value 2.

Both metrics (Fan-In and Fan-Out) can be calculated using an adjacency matrix. This matrix represents the graph as follows. We create a $n*n$ matrix whereby n is the number of nodes in the graph. $m(i, j) = 1$, if an edge exist

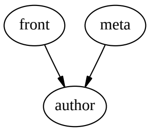

Figure 8: Fan-Out metric for the element `author`

between the node i and j, otherwise $m(i,j) = 0$. Figure 9 shows the matrix for the graph.

ID	1	2	3	4	5	6	7	..	FI
1	0	1	1	1	0	0	0	..	3
2	0	0	0	0	1	1	1	..	3
3	0	0	0	0	0	0	0	..	..
4	0	0	0	0	0	0	0	..	..
5	0	0	0	0	0	0	0	..	..
6	0	0	0	0	0	0	0	..	..
7	0	0	0	0	0	0	0	..	..
..	..	..	..	..	..	..	..	..	..
FO	0	1	1	1	1	1	1	..	..

1 - publications
2 - book
3 - article
4 - conference
5 - front
6 - body
7 - references

Figure 9: Adjacency matrix for calculating Fan-in and Fan-out

The Fan-In value is the sum of a row, the Fan-Out is the sum of a column derived from the adjacency matrix. This matrix is very descriptive but not the most efficient way to calculate both metrics.

Analysis of the metrics In the previous paragraphs, we introduced five different metrics for DTDs. These metrics are

- $Size$
- $Compl$
- $Depth$
- $Fan - In$ and
- $Fan - Out$.

The metrics $Size$ and $Compl$ can be applied to complete DTDs. The metrics $Depth$, $Fan-In$ and $Fan-Out$ can be determined for each element and attribute of a DTD and for the complete graph.

The metrics $Size$ and $Compl$ influence the characteristics usability and maintainability.

The metrics $Depth$ and $Fan - In$ determine which elements of the DTD are complex. DTD attributes always have the value $Depth = 0$ and $Fan - In = 0$. The complexity of elements can be evaluated using both metrics.

The metric $Fan - Out$ specifies which elements are often reused in the element declarations. Elements that have a high Fan-Out value require much effort if the DTD undergoes changes.

5 Applications

The metrics developed in the last section can be used in many ways:

1. If a user develops a DTD then the metric can support estimating how easy it is to understand and use the DTD. The metrics can also find out which element declarations are difficult to understand. This information is the prerequisite for simplifying and improving DTDs.

2. Often several schema descriptions for the same field of applications exist. One task is to choose one of these descriptions. The five metrics suggested in this article can support users during this task. The metrics allow for comparing the size and complexity of the schema descriptions.

   This problem is relevant if users have to choose a DTD for their application. The same problem is relevant in federated information systems where XML documents with different schemas typically occur.

3. Evaluation is also necessary if a schema description shall be derived from a collection of XML documents. Tools that realise this task always have different alternatives for creating a schema. The metrics can support choosing one of the alternatives.

6 Examples

In this section we give two complete examples. The DTDs are thereby given and all metrics described in this article are calculated.

6.1 Periodic table ([HE00])

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT PERIODIC_TABLE (ATOM)*>
    <!ELEMENT ATOM (NAME, ATOMIC_WEIGHT, ATOMIC_NUMBER, OXIDATION_STATES?,
        BOILING_POINT?, MELTING_POINT?, SYMBOL, DENSITY?, ELECTRON_CONFIGURATION,
        COVALENT_RADIUS?, ELECTRONEGATIVITY?, ATOMIC_RADIUS?, HEAT_OF_VAPORIZATION?,
        ATOMIC_VOLUME?, HEAT_OF_FUSION?, IONIZATION_POTENTIAL?,
         SPECIFIC_HEAT_CAPACITY?, THERMAL_CONDUCTIVITY?)>
        <!ATTLIST ATOM STATE (GAS | SOLID) #IMPLIED>
        <!ELEMENT NAME (#PCDATA)>
        <!ELEMENT ATOMIC_WEIGHT (#PCDATA)>
        <!ELEMENT ATOMIC_NUMBER (#PCDATA)>
        <!ELEMENT OXIDATION_STATES (#PCDATA)>
        <!ELEMENT BOILING_POINT (#PCDATA)>
            <!ATTLIST BOILING_POINT UNITS CDATA #FIXED "Kelvin">
```

```
15          <!ELEMENT MELTING_POINT (#PCDATA)>
16              <!ATTLIST MELTING_POINT UNITS CDATA #FIXED "Kelvin">
17          <!ELEMENT SYMBOL (#PCDATA)>
18          <!ELEMENT DENSITY (#PCDATA)>
19              <!ATTLIST DENSITY UNITS CDATA #FIXED "grams/cubic centimeter">
20          <!ELEMENT ELECTRON_CONFIGURATION (#PCDATA)>
21          <!ELEMENT COVALENT_RADIUS (#PCDATA)>
22              <!ATTLIST COVALENT_RADIUS UNITS CDATA #FIXED "Angstroms">
23          <!ELEMENT ELECTRONEGATIVITY (#PCDATA)>
24          <!ELEMENT ATOMIC_RADIUS (#PCDATA)>
25              <!ATTLIST ATOMIC_RADIUS UNITS CDATA #FIXED "Angstroms">
26          <!ELEMENT HEAT_OF_VAPORIZATION (#PCDATA)>
27              <!ATTLIST HEAT_OF_VAPORIZATION UNITS CDATA #FIXED "kilojoules/mole">
28          <!ELEMENT ATOMIC_VOLUME (#PCDATA)>
29              <!ATTLIST ATOMIC_VOLUME UNITS CDATA #FIXED "cubic centimeters/mole">
30          <!ELEMENT HEAT_OF_FUSION (#PCDATA)>
31              <!ATTLIST HEAT_OF_FUSION UNITS CDATA #FIXED "kilojoules/mole">
32          <!ELEMENT IONIZATION_POTENTIAL (#PCDATA)>
33          <!ELEMENT SPECIFIC_HEAT_CAPACITY (#PCDATA)>
34              <!ATTLIST SPECIFIC_HEAT_CAPACITY UNITS CDATA #FIXED
                                                 "Joules/gram/degree Kelvin">
35          <!ELEMENT THERMAL_CONDUCTIVITY (#PCDATA)>
36              <!ATTLIST THERMAL_CONDUCTIVITY UNITS CDATA #FIXED
                                               "Watts/meter/degree Kelvin">
```

Evaluation of the DTD

$Size$	31
$Compl$	1
$max.\ Depth$	2
$max.\ Fan-In$	18
$max.\ Fan-Out$	1

Evaluation of the elements/attributes

Line-ID	2	3	9	10	11	12	13	15	17	18	20	21	23	24	26	28	30	32	33	35
Element-ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$Fan-In$	1	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$Fan-Out$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$Depth$	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.2 DTD for Shakespeare

```
1   <!-- DTD for Shakespeare    J. Bosak    1994.03.01, 1997.01.02 -->
2   <!-- Revised for case sensitivity 1997.09.10 -->
3   <!-- Revised for XML 1.0 conformity 1998.01.27 -->
4
5   <!ENTITY amp "&#38;#38;">
6
7   <!ELEMENT PLAY (TITLE, FM, PERSONAE, SCNDESCR, PLAYSUBT, INDUCT?, PROLOGUE?,
                  ACT+, EPILOGUE?)>
8       <!ELEMENT TITLE (#PCDATA)>
9       <!ELEMENT FM (P+)>
10      <!ELEMENT P (#PCDATA)>
11      <!ELEMENT PERSONAE (TITLE, (PERSONA | PGROUP)+)>
12      <!ELEMENT PGROUP (PERSONA+, GRPDESCR)>
13      <!ELEMENT PERSONA (#PCDATA)>
14      <!ELEMENT GRPDESCR (#PCDATA)>
15      <!ELEMENT SCNDESCR (#PCDATA)>
```

```
16      <!ELEMENT PLAYSUBT (#PCDATA)>
17      <!ELEMENT INDUCT (TITLE, SUBTITLE*, (SCENE+ | (SPEECH | STAGEDIR | SUBHEAD)+))>
18      <!ELEMENT ACT (TITLE, SUBTITLE*, PROLOGUE?, SCENE+, EPILOGUE?)>
19      <!ELEMENT SCENE (TITLE, SUBTITLE*, (SPEECH | STAGEDIR | SUBHEAD)+)>
20      <!ELEMENT PROLOGUE (TITLE, SUBTITLE*, (STAGEDIR | SPEECH)+)>
21      <!ELEMENT EPILOGUE (TITLE, SUBTITLE*, (STAGEDIR | SPEECH)+)>
22      <!ELEMENT SPEECH (SPEAKER+, (LINE | STAGEDIR | SUBHEAD)+)>
23      <!ELEMENT SPEAKER (#PCDATA)>
24      <!ELEMENT LINE (#PCDATA | STAGEDIR)*>
25      <!ELEMENT STAGEDIR (#PCDATA)>
26      <!ELEMENT SUBTITLE (#PCDATA)>
27      <!ELEMENT SUBHEAD (#PCDATA)>
```

Evaluation of the DTD

$Size$	22
$Compl$	42
$max.\ Depth$	5
$max.\ Fan-In$	9
$max.\ Fan-Out$	7

Evaluation of the elements/attributes

Line-ID	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Element-ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$Fan-In$	9	0	1	0	3	2	0	0	0	0	6	5	5	4	4	4	0	1	0	0	0
$Fan-Out$	0	7	1	1	1	1	2	1	1	1	1	1	2	2	2	4	1	1	6	5	3
$Depth$	5	0	1	0	2	1	0	0	0	0	4	4	3	3	3	2	0	1	0	0	0

The metric for both examples reflect that the size of the DTDs is comparable. The complexity of the first examples is very low, only the declaration of the element `author` is complex. The second example contains a lot of complex elements and the structure complexity of the entire DTD is high.

7 Conclusion

A more detailed representation of the metrics given in this article can be found in [Sch02]. Algorithms that determine all metrics are implemented. The program uses the DTD parser developed by Mark Wutka.

If you look at the resulting metrics, you will recognise them as being very simple. The reason for this is that we have not tried to develop metrics that can be applied for only one special application. We tried to find general metrics that can be used for several applications. Here, you can see a parallel to the software metrics. Those metrics that are widely accepted are also very simple ones. For instance, the metric lines of code (LOC) at first glance are a very inaccurate measure, nevertheless, it is widely used.

All metrics employ characteristics of a graph representing the DTD. That means that the metrics are based on graph theory and have an underlying theoretical model.

Although we believe that metrics for XML document collections can be applied in many ways, we want to conclude with the following citation.

"Not everything that counts can be counted,
and not everything that can be counted counts."

Albert Einstein

8 Future Work

The development of specialized metrics that evaluate if a DTD is data centric or document centric is one of the future tasks. It is widely accepted that this characteristic influences the storage of the XML documents.
In the article [KM00], we developed a method for choosing an optimal storage facility for XML documents. Therefore, the first step of the design algorithm is to determine whether each element and attribute of a given DTD contains data or full text. For this decision, the article uses information from the DTD, furthermore statistics derived from sample XML document collections and sample queries. The developed metric controlled implicit the design algorithm. One of the future tasks is to make this metric explicit and to generalize the metric so that it can be used by several applications.

Another task is to combine the evaluation of DTDs with a method for extracting an XML schema from an XML document collection which was developed in [Rom01]. We want to achieve thereby an improvement in the quality of the resulting XML schema descriptions. For this task, it is necessary to develop metrics that evaluate XML schema.

References

[Cho00] Byron Choi. A Few Tips for Good XML Design. Technical report, University of Pennsylvania, http://db.cis.upenn.edu/˜kkchoi/DTDI2/, November 2000.

[CK94] Shyam R. Chidamber and Chris F. Kemerer. A Metric Suite for Object-oriented Design. In *IEEE Transaction on Software Engineering*, volume 20(6), pages 476–493, June 1994.

[CS00] Byron Choi and Arnaud Sahuguet. DTD Mininig Labor Day Report. Technical report, University of Pennsylvania, http://db.cis.upenn.edu/˜kkchoi/DTDI2/, September 2000.

[Dum02] R. Dumke. Software Metrics Classification, University of Magdeburg, Germany, "http://ivs.cs.uni-magdeburg.de/sw-eng/us/metclas/index.shtml", 2002.

[Fen91] Norman E. Fenton. *Software Metrics – A Rigorous Approach.* Chapman & Hall, London, 1991.

[GJP00] Marcela Genero, Luis Jimnez, and Mario Piattini. Measuring the quality of entity relationship diagrams. In *Conceptual Modeling - ER 2000*, volume 1920 of *Lecture Notes in Computer Sience*. Springer, 2000.

[HE00] Eliotte Rusty Harold and Engel, Reinhard (bers.). *Die XML-Bibel*. MITP-Verlag, Bonn, 1. auflage edition, 2000. CD-ROM.

[HK81] Sallie M. Henry and Dennis G. Kafura. Software structure metrics based on information flow. In *IEEE Transactions on Software Engineering*, volume SE7(5), pages 510–518, September 1981.

[Hog97] Jer Hogan. An Analysis of OO Software Metrics. Technical report, University of Warwick, May 1997.

[KM00] Meike Klettke and Holger Meyer. XML and Object-Relational Databases - Enhancing Structural Mappings Based on Statistics. In *Third International Workshop WEBDB2000*, volume 1997 of *Lecture Notes in Computer Sience*, pages 151–170. Springer, May 2000.

[McC76] Thomas J. McCabe. A complexity measure. In *IEEE Transactions on Software Engineering*, volume SE2(4), pages 308–320, December 1976.

[Mil56] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. In *The Psychological Review*, volume 63, pages 81–97, 1956.

[Rom01] Christian Romberg. Untersuchung zur automatischen XML-Schema-Ableitung. Master's thesis, University of Rostock, 2001. (in German).

[Sah00] Arnaud Sahuguet. Everythink You Ever Wanted to Know About DTDs, But Were Afraid to Ask. In *Third International Workshop WEBDB2000*, volume 1997 of *Lecture Notes in Computer Sience*, pages 171–183. Springer, May 2000.

[Sch02] Lars Schneider. Entwicklung von Metriken für XML-Dokumentkollektionen. Master's thesis, University of Rostock, 2002. (in German).

[Wes96] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, Inc., 1996.

[ZD01] Horst Zuse and Karin Drabe. *ZD-MIS: Zuse/Drabe - Measurement Information System*. URL http://home.t-online.de/home/horst.zuse/zdmis.html, February 2001.