A Flexible Generator for Synthetic XML Documents

Chi-Chien Pan and Kai-Hsiang Yang and Tzao-Lin Lee
Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan, R.O.C.
E-mail: {d5526001, f6526004, tl_lee}@csie.ntu.edu.tw

Abstract. Many XML-related projects and programs have been highly developed under the age of information technology. In order to improve the accuracy and reliability of these XML programs, developers need a large number of valid data for testing. However, it's not easy to achieve. To solve this problem, we design a new XML generator that can easily generate the XML structures and contents for different needs. Furthermore, the new XML generator can accept different desiring parameters, such as the document size, style, and other properties under the generation process. We also consider different properties of XML document in our XML generator system. This paper gives an overview about the design of our new XML generator. Unlike the conventional ones, our XML generator can accept a few XML sample documents for the distribution model during the generation process, and according to the distribution model we can generate some relatively reasonable structures and data. These generating XML documents are similar to those documents in real world. In addition, our generator also has the mechanisms of the external enumeration and user-defined data-type, which can make it more extensible under different users' demand. Finally, these generated XML documents could be the best test data for the related researches, such as XML query, search engine, and database systems.

1 Introduction

For many research works, such as search engine and database system, the developers always need a large number of data for testing and evaluating their systems. However, it is not easy to obtain a great deal of data in the real world because of the access authority, confidentiality, and privacy of user data. Therefore, using the virtual synthetic data is becoming one reasonable solution for the problem.

On the other hand, using the virtual synthetic data has more distinctive advantages, including generating any document size and amount as we wish. These generated XML documents are almost all under the users' control. Furthermore, some special extreme data could be also generated to test those situations that happened infrequently. Accordingly that could help us find out some problems existing in unusual conditions. The testing operation is indispensable to development process of every mature system. It also enhances the efficiency and reliability of systems greatly.

For the XML field, the synthetic data is more important. There are many different XML-based applications today, and they require different documents with different complexities and sizes. For example, a benchmark for data-intensive applications might require a large number and relatively homogeneous documents, with many

references among elements. However, an adequate test suite for a parser might be one heterogeneous collection with thousands of documents of varying sizes. Therefore, no matter under any scenarios, we believe our tool for generating synthetic XML documents is very useful.

Although some XML document generators had been proposed, the generated documents are not suitable for testing very well. The main reason is that they just focus on the data transformation form between different database systems, thus the structures of generated documents are too simple.

By considering actual needs and practices, our goal for the XML generator is to conveniently and easily generate various structures and contents under the user's control. During the generation processes, user can assign some properties like document size and style to generate the desired data. Besides, user can input a few sample documents as the distribution model to adjust the structures of output documents.

This paper is organized as follows: section 2 lists the related projects and researches in this area. In section 3, some basic concepts are explained. Section4 presents the analysis and generation method for XML structure. Section 5 outlines the generation method for element and attribute. Finally, last section is the conclusion and suggestions for future studies.

2 Related Work

In the field of utilizing synthetic XML documents for systematic evaluations and testing, the most relevant market would be the database systems, among which the mainstream is the relational database.

In recent years, a major issue of the development of XML has been the investigations about how to store the XML data in the traditionally developed relational database system. The most primitive synthetic XML documents were used to evaluate the different strategies for storing XML in the relational database systems. [1] The XML documents used consisted of the elements at one level with no nesting.

In the studies of this issue, two benchmarks, XMach-1 [8,9] and XMark [6, 10], are proposed for evaluating the performance of XML data management systems. Both benchmarks use synthetic XML documents that model documents from high-level applications. The structures of the documents in both cases are fixed and simple, and there is very little opportunity for varying it. This kind of data may be adequate for a benchmark that serves as a standard yardstick for comparing the performance of XML data management systems. But if our goal is to test and evaluate a particular XML data management system or a particular feature or component of such a system and to gain insights into its performance, then using XML data with widely varying structure over which we have more control can be more helpful. [7]

IBM also provides a XML data generator [5]. It generates XML data that conforms to an input DTD (Document Type Definition). Like the previous approaches, the IBM data generator is limited in the control it provides over the data generation process. The limitations include using only uniform frequency distributions with no opportunity for generating skewed data.

Among the existing synthetic XML document generators, the ToXgene [4] may be the one that has developed much more comprehensive functions. ToXgene is a template-based generator for large collections of synthetic XML documents. It is under

the development at the University of Toronto as part of ToX (Toronto XML Engine) Project. They define one language to describe the XML document content. The Template Specification Language (TSL) is essentially the XML Schema with the probability distributions and CDATA content descriptors, and is used for generating both attributes and elements. Like XML Schema specifications, TSL templates are also XML documents.

3 Basic Concepts

An XML document will be identified as validate if only defined by its DTD or XSD. As a result, the data and structures of a DTD and a XSD are the keys to generate an effective XML document.

The purpose of a DTD is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements. For example: one rule, <!ELEMENT name (#PCDATA | family | given)>, define the "name" element to be composed of one "PCDATA", one "family" element, and one "given" element.

DTD was one of the pioneering forms to define XML. Its major limit is that it was unable to comprehensively demonstrate the types of data and the magnitude of numerical values, and thus causing an inaccurate data analysis. To derive the data outputs, the DTD would require the users to provide extra definitions in order to obtain reasonable contents. Moreover, the syntax of DTD also differs from the XML document and therefore requires an additional DTD parser as it strives to process the data. Other limitations of DTD include inextensibility and incompatibility with the namespace of XML.

The major components of XML Schema include: (a) Part 0: Primer, (b) Part 1: Structure, and (c) Part 2: Data Type. The XML Schema describes the data type and restrictions of each element and attribute in detail, thus reducing the inconvenience of stepwise definitions appointed by the users and automatically yielding the XML documents. See Figure 1 for the operation procedure of the present system.

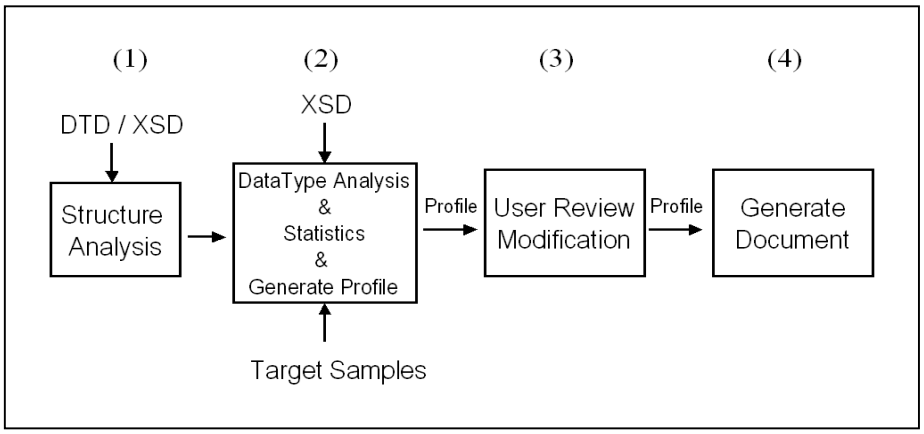

Figure 1. System Operation Procedure

The operation procedures are as follows. First, indicate the target DTD/XSD and construct the syntax structure by parsing it.

Second, conduct the Data Type parsing analysis if using an XSD file and assign each related data type directly. Third, if some target samples are provided, conduct a data analysis with the foregoing syntax, yielding the distribution of probabilities of each element and attribute and determining the influence to the whole structure.

If the current file contains the contents of the elements and attributes that are not defined by the DTD or XSD, an "assumption" of the Data Type should be made according to the sample value and its name. The names associated with the specific Data Type are derived from the simple correlations and are saved as default files in the system. These names are made according to the data's nominal senses in our daily life, for example, a "name" is defined to be a string, and an "age" is defined as an integer. The determination of data type by the sample values is based on type priority (1. numerical orders, 2. dates, and 3. characters.

The last stage is to generate the Profiles. A Profile is a simple plain text file, which contains the processes and results of all the previous analyses.

Because an "assumption" is more likely to be error prone, and in addition, the users may attempt to define some certain detail parameters on their own (e.g., yielding a probability distribution model), the users then will be required to conduct the Review and Modification. Given that the system has automatically generated all syntax structures, the data used in the Review and Modification stage have already existed and there is no need for user to assign parameters. The creation of this stage has also increased the applicability of the present system. The last stage is generating the document.

4 Generating Structure

The XML documents yielded include the "Structure" and the "Content." The Structure determines the overall complexity of the structure. The major processes start from the analysis of input DTD/XSD, and then derive the Profile by the statistical results of the target samples, and finally come to the users' review and modification.

To provide a detailed illustration of the process of data analysis, the following example is a part of the DTD that is to transform into graphs.

```
<!ELEMENT name (#PCDATA | family | given)*>
<!ELEMENT family #PCDATA)>
<!ELEMENT given #PCDATA)>
```

The DTD is composed of six components. E (one element), and E? (none or one occurrence of E), E+ (minimum one occurrence of E), E* (none or more occurrences), E | F (one E element or one F element), and EF (one E element followed by one element F).

According to the definition of DTD, the basic elements of the graphic units are first defined, as shown in Figure 2.

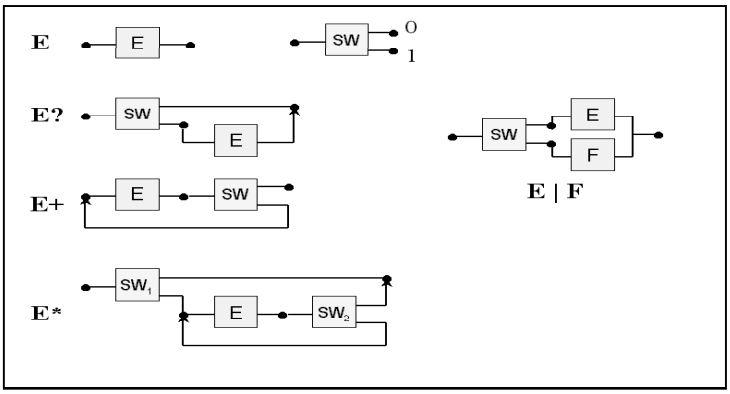

Figure 2. Basic Graph Units

The "E" represents "Element," with each element having two connection points (i.e., in and out). The "SW" presents "switch," which randomly selects the path 0 or 1 to connect with "E."

The "E?" means "zero or one occurrences." The "E+" means "minimum one occurrence." The "E*" means "zero or more occurrences." The parallel connections represent the sequential connections, and those using SW to connect represent OR.

Following the procedure of system generation, the aforementioned simple DTD structure can be demonstrated by the graphs in Figure 3.

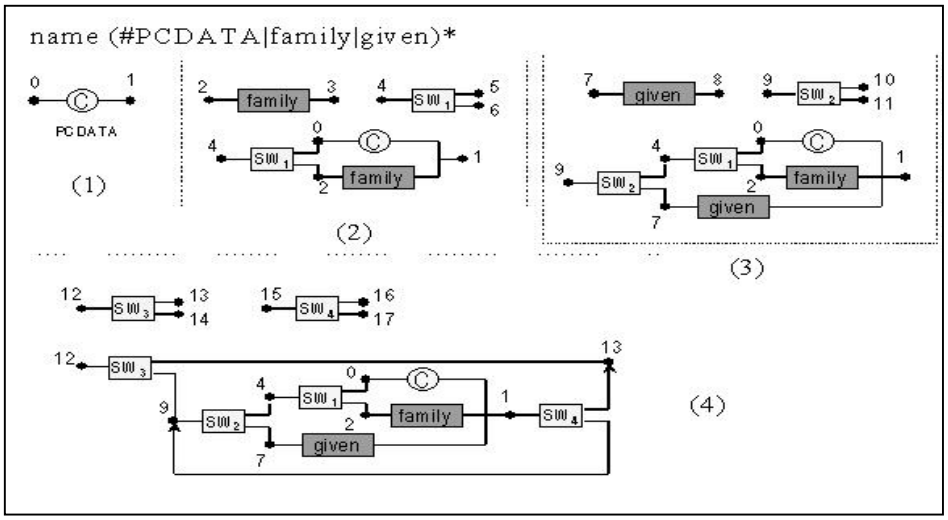

Figure 3. The DTD name element

As can be seen, the numbers in Figure 3 represent the orders that each connection points proceed. When two points are connected, the point with a smaller number will be maintained by the system as it develops a table to document the connections between points. When the generator processes the "#PCDATA," it generates the corresponding unit (step 1). Then the generator processes the "|family," it generates one switch and one "family" element (step 2). When the "|given" is seen, another switch

and “given” element are constructed. Finally, when the * is seen, the generator constructs the whole syntax structure as step 4.

After the connection model of the whole system is established, the next step is to input the target samples. Given that the parsing procedure acts as the flow vectors within the connected graphic units, by the positions of the elements, the system will be able to search the lookup table and to determine the points that fall within the current element and the next. See Table 1.

Table 1. Sample document flow record.

Node \ Next	End of Element	PCDATA	family	given
12	13	9,4,0,1	9,4,2,1	9,7,1
1	13	9,4,0,1	9,4,2,1	9,7,1

As shown in table 1, the first record means the flow from node 12 to 13, if one “PCDATA” occurs, will pass the node (9,4,0,1). If one “family” element occurs, the flow will pass the node (9,4,2,1).

The vector between points may be used as the counting units, for example, V12,9 and V12,13 represent the vectors of the two selected directions for SW3. As the result, the system will be able to generate the distribution of probabilities of every switch based on the amount of vector of each switch. After the completion of the previous processes, the Profile is derived for the users’ review and modification.

After modifying the Profile, the whole structure is generated through the control of random number generator and switch assignment.

The yielding of the attributes is a less complicated part compared to the Content (e.g., Data Type) and the Structure, as it only contains two options (e.g., demonstrate or not), this part can be calculated through the parsing procedure.

5 Generating Content

A complete XML requires an XML Structure and its Content. The Content includes “element” and “attribute.”

The objective of the content generation is to extend its applicability to the most extent so that accurate data can be derived by simple operation procedure. Although some system design methods contain more sophisticated analyses, including grammars and semantic meanings, and induce complex formula or pattern, they are not commonly used. This is mainly because that the goal of the present study is to generate testing data for the further investigations, and given that almost all users should have already decided what their desired data are, therefore, the Data Type has been defined in the profile and yielded after the user’s review.

See Figure 4 for the procedure of "Generate" of data elements and attributes.

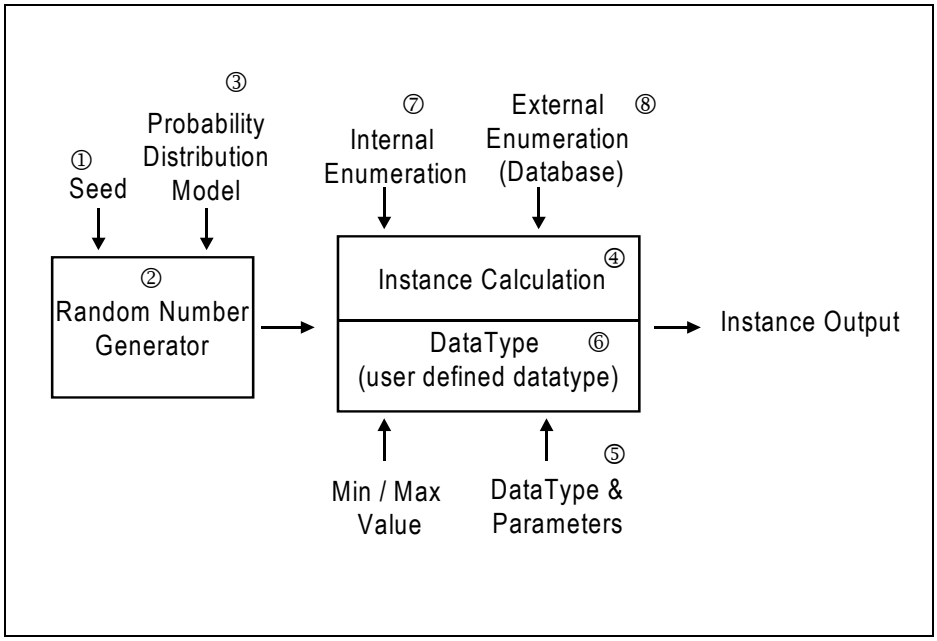

Figure 4. The content generation for elements or attributes

Within the procedure of yielding data, "Seed," "Probability Distribution Model," "Min/Max Value," and "Data type & Parameters" are given specific definitions in the profile based on their desired element or attribute.

At the "Random Number Generator," (2) the main inputs include "Seed" and "Probability Distribution Model." "Seed" is the default number to control and yield the random number, aiming to reduce the stepwise definition set-ups for users. The present study sets up a Global seed to pre-determine a random formula in order to derive each seed to reduce the labor of users.

The "Probability Distribution Model" (3) may be referred to some common probability models and be set up as these distributions: "uniform," "exponential," "normal," "geometric," "lognormal," or "none." The "none" refers to a non-randomized number and an output of sequential number.

The major inputs of "Instance Calculation" (4) include "Min/Max Value," and "Data type and Parameters." The external references include "Internal Enumeration" and "External Enumeration." This part is the core of data generated, including using the numerical values derived by the random number generator to refer to the data type that is associated with the min/max value and to refer to the enumeration.

The present system uses the definitions of the common data types in the XML Schema, including string, decimal, integer, Boolean, data, and time for the Data Type definitions. Associated with the parameters and user defined types, the external enumeration can be stored and retrieved. The data type of "string" can be further classified into multiple subtypes for the external enumeration, such as word, phrase, sentence, and paragraph.

The present system uses the user-defined types. The users can define their own data types to associate their data and the output. Doing so makes the system more flexible to process more complicated pattern and formula output, and thus not being restricted by the existing data types in the system.

The reference data include "Internal Enumeration" and "External Enumeration." The Internal Enumeration is derived from the primitive definitions of the DTD/XSD. Because the magnitude of enumeration defined is usually restricted, in an attempt to accelerate the processing, the Internal Enumeration is stored in the main memory in order to yield the reference.

The External Enumeration (8) is the critical part where the data is generated. Given that the profound data are derived from the real world, the authenticity of the data is therefore enhanced.

For example, yielding a name string data requires meaningful characters but rather randomized selection of letters (i.e., a-z), such as "Kevin" or "James."

6 Conclusion and Suggestions for Future Studies

The present study demonstrated its design of our XML Document Generator specifically for data testing. Its specialty is using the existing few samples for data type analysis, which generates profound testing data easily and with less difficulty and makes the data resemble to the real population.

These testing data will be used for the further developments of XML search engines and XML databases. Therefore, this generator will be revised continually.

References

[1] Daniela Florescu and Donald Kossmann. Storing and Querying XML Data Using an RDBMS. IEEE Data Engineering Bulletin, 22(3):27–34, September 1999.

[2] J. Edvarsson. A Survey on Automatic Test Data Generation. In Proceedings of the Second Conference on Computer Science and Engineering in Linkoping, pages 21-28, ECSEL, October 1999.

[3] D. Barbosa, A. Mendelzon, J. Keenleyside and K. Lyons. ToXgene: a template-based data generator for XML. In Proceedings of the Fifth International Workshop on the Web and Databases (WebDB 2002). Madison, Wisconsin - June 6-7, 2002.

[4] ToXgene web site, http://www.cs.toronto.edu/tox/toxgene

[5] IBM XML Generator, http://www.alphaworks.ibm.com/tech/xmlgenerator

[6] The XML Benchmark Project, http://monetdb.cwi.nl/xml

[7] Aboulnaga, J. Naughton and C. Zhang. Generating Synthetic Complex-structured XML Data. WebDB'00.

[8] Rahm, E., Böhme, T.: XMach-1: A Multi-User Benchmark for XML Data Management. Proc. VLDB workshop Efficiency and Effectiveness of XML Tools, and Techniques (EEXTT2002), Hongkong, Aug. 2002 (Invited Talk)

[9] Böhme, T., Rahm, E.:XMach-1: A Benchmark for XML Data Management. Proc. of BTW01 (Datenbanksysteme für Büro, Technik und Wissenschaft), Oldenburg, March 2001.

[10] A. Schmidt, F. Wass, M.Kersten, M.J. Carey, I.Manolescu, R.Russe, Xmark: A Benchmark for XML Data Management, Processdings of the 28th VLDB conference, Hong Kong, China, 2002.Appendix: Springer-Author Discount