A Testing Approach for XML Schemas

Maria Claudia F. P. Emer
State University of Campinas
mcemer@dca.fee.unicamp.br

Silvia Regina Vergilio
Federal University of Paraná
silvia@inf.ufpr.br

Mario Jino
State University of Campinas
jino@dca.fee.unicamp.br

Abstract

XML is a language frequently used for data representation and interchange in Web-based applications. In most cases, the XML documents must conform to a schema that defines the type of data that is accepted by a web application. In this sense, an error in the schema or in the XML document can lead to failures in the application; the use of testing approaches, criteria and specific tools to ensure the reliability of data in the XML format is fundamental. We present a testing approach that helps to reveal faults in XML schemas. The test process involves generating XML documents with some modifications with respect to the original XML document and using queries to these documents to validate the schema. The XML documents and queries are generated according to a set of fault classes defined for the XML schemas. A case study applying the proposed approach is described and the results are presented.

1. Introduction

Web applications include information distribution, e-commerce, entertainment and other activities [1]. The growing demand for web applications is accompanied by the need for these applications to contemplate quality requirements as: reliability, usability and security [1].

Testing activity plays an important role in software development by contributing to generate reliable products and to evaluate their quality [2]. As it happens with traditional software, web applications can involve critical operations associated with high costs. Therefore, web applications need to be reliable and to have a high level of quality. One way to achieve that is the introduction of specific testing criteria and tools [3, 4, 5, 6, 7, 8, 9].

Web applications have as a fundamental requirement the ability to manipulate information from various sources and XML (eXtensible Markup Language) has been adopted as the common format for data exchange in the web [10]. XML was developed to improve data representation capabilities and data interchange facilities. However, there is the possibility of using XML documents to store data as a database [11].

XML documents are in general associated to a schema. A schema is a grammar that defines the structure of an XML document. Schemas are usually defined in a DTD (Document Type Definition) or in an XML Schema [12]. If an XML document conforms to a schema, it is called valid. However, valid documents can be incorrect because the schema may not conform to its specification.

If the schema has some fault in the definition of the restrictions to the data of the XML document, incorrect data can be considered valid or correct data can be considered invalid and a failure may occur in the application using XML documents.

A testing approach is presented here to reveal faults in schemas of XML documents through queries to valid XML documents associated to the schema under test. Faults detected in the schema are not related to syntactic issues; these issues are analyzed when validating the documents with respect to the schema. It can be applied to reveal faults in schemas of: XML documents in an XML database; XML messages used for information exchange between web components; queries results of relational databases in XML format and XML documents that have been subjected to changes (regression testing).

The test approach is fault-based. Therefore, classes of faults that may be present in an XML schema are identified. Faults in an XML schema are revealed through queries to XML documents, associated to the proposed classes of faults.

This paper is organized as follows: Section 2 provides a background on XML, XML schemas and XML query languages. Section 3 presents the formal model for XML documents used to generate queries. Section 4 proposes the testing approach for XML schemas including the fault classes and the test process. Section 5 presents the results of a case study. Section 6 describes related work. Section 7 contains the conclusion and future work.

2. Background

2.1. XML

XML (eXtensible Markup Language) is a markup language defined by W3C (World Wide Web Consortium) to ease the exchange of data between applications and heterogeneous hardware platforms [13]. In XML, information is represented as a hierarchy of elements that have names, optional attributes and optional content [14]. XML uses tags to indicate the logical structure of the document and to identify the data content [11]. Tags are textual descriptions of data delimited by the symbols '<' and '>'. The way the data is interpreted depends on the application using the data.

2.2. XML Schema

A well-formed XML document obeys XML syntax rules. A valid XML is a well-formed document that follows rules defined by a schema. Schema is a grammar that defines the logical structure of an XML document. Schemas are usually written using DTD (Document Type Definition) [13] or using XML Schema (XML Language Schema) [12]. The schema for XML defines elements, attributes, and rules, among other items. An XML Schema is written in XML. Example 1 presents an XML Schema.

Example 1: XML Schema for courses XML document

```
<?xml version="1.0" ?>
<xs:scheman
XMLns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="courses">
  <xs:complexType>
  <xs:sequence>
    <xs:element name="course" minOccurs="0"
maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
        <xs:element name="prerequisite" type="xs:string"
minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="name_course" type="xs:string"
use="required" />
        <xs:attribute name="term" type="xs:integer"
use="required" />
    </xs:complexType>
    </xs:element>
  </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

The XML Schema of Example 1 defines: the elements and attributes of XML document for courses, the data type of the elements and attributes, the maximum and minimum number of times that an element can occur and, the attributes as required. The valid XML document, for this schema, stores data of courses as: name, academic term and prerequisite. The name of the course and the term are attributes of the element course, and the prerequisite is a child element of the element course.

2.3. XML Query Language

XQuery [15] were created to access data directly from XML documents. XQuery was proposed by W3C as an attempt to establish a standard query language for XML documents [11]. XQuery is a functional language (expressions can be nested), strongly typed (operands, operators and functions must conform to the expected types) and case-sensitive (keywords use lower-case characters) [15]. Every query in XQuery is an expression that is evaluated by generating a value. Path expressions and FLWOR expressions are examples of expressions of XQuery. Path expressions are used to locate nodes in XML. FLWOR expressions, formed by clauses “For”, “Let”, “Where”, “Order by” and “Return”, are used to combine and restructure information from XML. XQuery allows several XML documents to be queried simultaneously.

3. Formal Model

A formal model for XML documents is necessary to formalize the testing schema through queries on XML documents. Several formal models have been created for different purposes [8]. In this work, the presented formal model is derived from the formal models proposed by Murata [16], Chidlovskii [17] and Offutt [8]. The formal model is an extended Regular Tree Grammar (RTG). Extended RTG is used to represent an XML schema. Extended RTG is a 7-tuple $<\Sigma, A, D, R, N, P, n_s>$, where:

- Σ is a finite set of elements
- A is a finite set of attributes
- D is a finite set of data types
- R is a finite set of restrictions
- N is a finite set of non-terminals
- P is a finite set of production rules, which can take one of the three forms:
  - o $n \rightarrow a<d>$, where n is non-terminal in N, a is an attribute in A or an element in Σ and d is a data type in D.
  - o $n \rightarrow a<d\ r>$, where n is non-terminal in N, a is an attribute in A or an element in Σ, d is a data type in D and r is a restriction in R for a.
  - o $n \rightarrow e<m\ r>$, where n is non-terminal in N, e is an element in Σ, m is a regular expression comprising non-terminals and r is a restriction in R for e.
- n_s is the starting non-terminal, $n_s \in N$.

An extended RTG model can be generated from a DTD or an XML Schema. Example 2 presents an extended RTG $G_1 = <\Sigma, A, D, R, N, P, n_s>$ for the schema of Example 1.

Example 2: The extended RTG that represents the schema of Example 1.

Σ = {courses, course, prerequisite}
A = {name_course, term}
D = {xs:string, xs:integer}
R = {complexType, minOccurs, maxOccurs, use}
N = {n_s, n_1, n_2, n_3, n_4}
P = { $n_s \rightarrow$ courses<n_1, complexType>,
$n_1 \rightarrow$ course<n_2 n_3 n_4, complexType, minOccurs, maxOccurs>,
$n_2 \rightarrow$ prerequisite<xs:string, minOccurs, maxOccurs>,
$n_3 \rightarrow$name_course<xs:string, use>,
$n_4 \rightarrow$ term<xs:integer, use>}

4. Testing XML Schemas

The goal is to reveal faults in the XML schema with respect to its specification. Figure 1 illustrates a possible testing context, in which the database could be in the XML format and the application should execute queries and update the XML database. The communication is performed with http and XML.

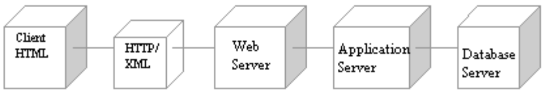

Figure 1. Application model [18]

We consider that the documents and schemas to be tested are well formed and valid. The idea is to evaluate semantic issues, faults in the schema of an XML document that can affect the integrity of the data in the XML document and, as a consequence, can cause failures in the web application that manipulates the data of this document. The approach is fault-based. Next a fault model composed of fault classes in XML schemas is presented and a fault-based process is described.

4.1. Fault Classes

Faults that will be detected in a schema are related to mistakes commonly made during the development of the schema or during its evolution.

In this paper, the schema for XML that is being used, initially, is the XML Schema. However, the approach can also be applied to DTDs. XML Schema is now more used than DTD. It is written in XML, supports data types and namespaces. Therefore, some kind of faults can be specific for XML Schema.

Three classes of faults found in XML schemas are presented next. Each class is subdivided into fault types.

Class 1 – Elements $\rightarrow$ faults related to the element definition in the schema.

1.1 Incorrect Order - order in that the child element of a complex element in an XML document should appear is defined in an XML Schema through the words *all*, *sequence* and *choice*, that mean respectively: the child elements can be appear in any order, the child elements should obey the order of its definition in the schema and only one of the child elements can occur in the XML document.

1.2 Incorrect Value - an element may be associated to a value, default or fixed. The value is defined as default if it should be automatically assigned when no other value is provided. The value is defined as fixed if it is the only value for the element.

1.3 Incorrect Occurrence - the number of times a same element may occur is defined by a minimum and maximum number of occurrences for the element. The default value for minimum and maximum is 1.

Class 2 - Attributes $\rightarrow$ faults related to the definition of the attributes of an element in the schema.

2.1 Incorrect Use - the attribute may be defined as optional or required.

2.2 Incorrect Value - an attribute may contain a default or fixed value, in the same way that happens with the value of an element.

Class 3 - Restrictions on data $\rightarrow$ faults associated to the established restrictions for the data that can be assumed by an element.

3.1 Incorrect Data types - the most common data types in XML Schema are: string, decimal, integer, boolean, date and time.

3.2 Incorrect Enumerated Values – incorrect list of acceptable values for an element.

3.3 Incorrect Maximum and Minimum Values – incorrect upper and lower bounds for numeric values

3.4 Incorrect Pattern – incorrect sequence of characters for an element.

3.5 Incorrect White Space characters – specify incorrectly how the XML processor should handle white space characters, that is, it should be preserved, removed or reduced.

3.6 Incorrect Length - size of an element with respect to the number of characters.

3.7 Incorrect Digits - the total amount of digits that a number may assume or the total of decimal digits that a number may contain.

These classes are used in the testing approach to guide the execution of queries on XML documents. The goal is to detect faults, which are related to the absence of necessary restrictions or to an incorrect definition of a restriction for an element. The approach proposes the execution of such queries in valid XML documents to detect faults in the schema of the document that can generate failures in the web application. The queries are defined in agreement with the fault classes, i. e., each kind of fault is related to a query capable to reveal that fault.

4.2. Testing Process

In this section, the steps that compose the test process for XML schemas are described. The process is illustrated in Figure 2.

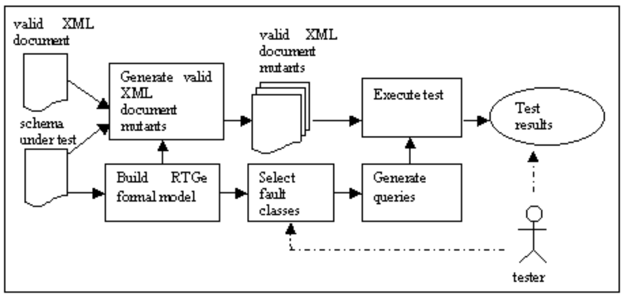

Figure 2. Process for Testing XML schemas

Initially, the schema to be tested and one or more XML documents are extracted from the web application or provided by the tester. The extended RTG model is built and, based on this model, the classes of faults that are present in the XML schema being tested are generated. Elements and attributes of the schema are associated to the fault classes using the formal model. Fault classes are then used to insert modifications in the valid XML documents, generating diverse XML document mutants. In the same way of traditional mutation testing, each XML document mutant differs from the original document by only one simple modification described by the fault classes introduced previously. An example mutation is the modification of the value assumed by certain elements of the XML document that has the restriction of maximum value so that the defined limit value in the schema is tested according to the specification.

It is important to observe that the generated mutants are well-formed and valid documents with respect to the schema.

The associations between the classes of faults and the elements and attributes present in the schema under test are also used in the next step. This step includes a selection of associations that can be made automatically or by the tester. For each selected association, a query is generated. It will be executed on the valid XML document mutants and the results of these queries are confronted with the specification of the expected results.

Test data, in this approach, are formed by a valid XML document for the schema under test, for which different valid XML documents are generated, and of the queries associated to each class of faults defined previously. The specification of the expected result for the test data can be obtained from the specification of the schema or from the web application that deals with the XML documents.

5. Case Study

This section presents the application of the proposed approach in two web-based systems: library and registration system. The systems were developed by two groups of students enrolled in a graduate course. The goal was the development of a web application using UML models. Specifications of the systems and of the schemas were given to each group.

In this case study, the steps of the test process were executed manually, except for the execution of the test data.

Registered users can access the library system through login and password, to consult available titles in the collection. Each user can access at most three titles a month if its monthly fee has been paid. The XML database contains data referring to the registered users and on the available titles in the collection. XML documents are based on two XML Schemas: *user.xsd* and *title.xsd*.

A student may access the registration system with his login and password. The student can enroll only in courses referring to his or her term in courses, whose prerequisites have been met. XML database has data referring to the students and the available courses in the program. The database is based on two XML Schemas: *student.xsd* and *course.xsd*.

Initially, the XML documents and corresponding schemas of the database for each system, identified above, were accessed. The extended RTG formal model, presented in Section 3, was obtained to represent each XML Schema under test. Based on the formal model, classes of faults that could be applied to the elements present in the schema were generated. Table 1 presents the number of associations (fault classes with elements) identified for each XML document of the databases.

Table 1. Identified associations for the XML documents of the databases

Database XML	Number of associations
User	22
Title	20
Student	30
Course	26

XML documents were generated with simple modification, such as: number and order of elements in the document, number of characters or digits in the value of an element and data type that an element can assume. The mutated XML documents were validated, in other words, evaluated with respect to being well formed and to be in conformity with the schema associated to the document. The mutated XML documents that were not valid for the respective schema were discarded. This step was necessary because the mutants were manually generated. Table 2 shows the number of mutated documents and the number of valid mutated XML documents that stayed in the test process.

The queries were generated with the XML language query XQuery. Qexo [19] was used to execute the expressions in XQuery. Qexo is free software, executed on the Java platform. Table 3 shows the list of generated queries for the generated XML documents for the *user* database.

Table 2. Number of generated mutants

Database XML	Number of mutated documents	Number of valid mutated documents
User	45	26
Title	62	34
Student	59	35
Course	59	35

Table 3. List of generated queries for the XML documents of *user*

Element of the XML document *user*	Generated queries	Number of generated queries
user	order, occurrence	2
code	occurrence, data type, pattern, size	4
name	occurrence, data type	2
login	occurrence, data type, pattern, size	4
password	occurrence, data type, pattern, size	4
payment monthly fee	occurrence, data type, size	3
amount of access	occurrence, data type, minimum and maximum value	3
Total		22

The test data composed of the valid XML document mutants and of the corresponding queries were executed. Results obtained with the execution of the test data were confronted with the specification of the expected results according to the specification of each schema for the XML database of the library and registration systems.

The results of the queries have shown the presence of faults in the schemas of the XML documents. The faults are related to the absence of restrictions in the definition of some elements with respect to the size, pattern, maximum and minimum values and the incorrect definition of the data type and of the occurrence of some elements. Table 4 presents the faults revealed by the test in the schema regarding the XML database for *user*.

Table 4. Faults revealed in the schema of *user*

Element of the XML document *user*	Revealed Fault
code	pattern and size
login	pattern and size
password	pattern and size
payment monthly fee	data type
amount of access	data type and maximum value

The number of defects found in the schemas of the XML databases is summarized in Table 5.

Table 5. Test results

XML Schema	Number of detected faults
user.xsd	9
title.xsd	3
student.xsd	12
course.xsd	8

6. Related Work

Lee et al. [6] propose an approach that applies mutation analysis for the validation of the interaction of data through XML messages among web-based software system components. Mutant XML messages are generated and executed so that the tester analyzes the results of the interaction among the web components. In the same way of the mutation analysis [20] for traditional software, in this approach the behavior of the interaction mutants is observed and the mutants can be considered dead, alive or equivalent. The tester should still examine the live mutants, after the execution of all the test cases, to verify if equivalent mutants exist or to decide if new test cases should be created or, still, if the results are satisfactory.

Offutt et al. [8] explore the use of data perturbation in XML to generate test cases for web services. This approach focuses on testing peer-to-peer web services interaction, altering request messages and analyzing the response obtained for the correct behavior. Data perturbation includes data value perturbation and interaction perturbation. Data value perturbation modifies values in SOAP (Simple Object Access Protocol – message in format XML) messages in terms of the types of the data. Interaction perturbation alters messages in RPC (Remote Procedure Calls – messages with values for the arguments of remote procedure functions) and in data communication (messages to transfer data).

The approaches proposed by Lee et al. [6] and by Offutt et al. [8] explore the use of mutated XML documents to test the interaction among components (or web services) in a web application. The approach proposed in this paper also uses XML mutants and, in addition, uses queries to XML documents to test XML schemas and evaluate the results. In the testing context, schemas of the following documents can be considered: 1) XML documents in an XML database; 2) XML messages used for information exchange in a web application, for instance, exchange of messages among web services of an application; 3) results in XML format of queries in relational database. Moreover, the test in XML schema can happen when an XML document is updated and, as a consequence, the schema associated to the document is also updated [21], i.e., schemas are changed. This is a very common problem, mainly in applications that use DTD and are now adopting XML Schema.

The advantage of using queries is that if the specification is available it is possible to test the schemas or XML documents independently of the application. The kind of mutations proposed by the above mentioned authors could be applied to generate the XML mutants in our approach.

7. Conclusion and Future Work

This paper presents a fault-based approach to test schemas of XML documents. This approach aims to ensure the quality of the data in an XML document associated to the schema under test. It is fundamental that data in XML format are reliable so that the incorrect data processing, that could generate failure in the web application, is avoided.

The proposed test approach includes a test process that is based on a set of XML mutants and queries in the XML query languages that describes some fault classes common to the XML schemas.

The queries associated to the fault classes are generated and executed with the mutated XML documents. The results obtained with the execution of the queries are compared with the specification of the expected results, established in the specification of the schema.

A case study of application of the approach is presented. This preliminary study shows that the approach can detect faults in XML schemas, which could imply failures in the web application.

The approach in this paper has been illustrated with XML Schema only. However, it can be also used to test other kind of schema, such as DTD.

An algorithm to automatically derive the extended RTG model, and also the associations between fault classes and elements in the XML schemas is now being developed. With these associations the queries and XML mutants will be automatically generated. A testing tool to support the introduced approach will be developed and experiments will be conducted.

8. References

[1] Offutt, J. *Quality attributes of web software applications*. IEEE Software, Volume: 19, Issue: 2, Pages: 25–32, March-April 2002.

[2] Pressman, Roger S.. *Software Engineering - Practitioner's Approach*. 5th ed., McGraw-Hill, New York, 2000.

[3] Di Lucca, G.A.; Di Penta, M. *Considering browser interaction in web application testing*. Web Site Evolution, 2003. Theme: Architecture. Proceedings. Fifth IEEE International Workshop on, Pages:74 – 81, 22 Sept. 2003.

[4] Di Lucca, G.A.; Fasolino, A.R.; Faralli, F.; De Carlini, U. *Testing Web Applications*. Software Maintenance, 2002. Proceedings. International Conference on , 3-6 Oct. 2002 Page(s): 310 –319.

[5] Kung, D. C; Liu, C. H. and Hsia, P. *An Object-oriented Web test model for testing Web applications*. In The 24th Annual International Computer Software and Applications Conference, COMPSAC 2000, pages 537-542. IEEE Press, 2000.

[6] Lee, S. C.; Offutt, J.; *Generating test cases for XML-based web component interactions using mutation analysis*. Software Reliability Engineering, 2001. ISSRE 2001. Proceedings. 12th International Symposium on , 27-30 Nov 2001 Page(s): 200 –209.

[7] Liu, C. H.; Kung, D. C; Hsia, P. and Hsu, C. T. *Structural Testing of Web Applications*. In 11th International Symposium on Software Reliability Engineering, pages 84-96. IEEE Press, 2000.

[8] Offutt, J.; Xu, W. *Generating Test Cases for Web Services Using Data Perturbation*. In TAV-WEB Proceedings/ACM SIGSOFT SEN, vol. 29, number 5, September, 2004.

[9] Ricca, F.; Tonella, P. *Analysis and Testing of Web Applications*. Software Engineering, 2001. ICSE 2001. Proceedings of the 23rd International Conference on , 12-19 May 2001 Page(s): 25 –34.

[10] Chitic, C. and Rosu, D. *On Validation of XML Streams using finite state machines*. Technical Report CSRG-489, University of Toronto, 2004. http://citeseer.ist.psu.edu / chitic04validation.html (accessed in November 2004).

[11] Sousa, A. A.; Pereira, J. L.; Carvalho, J. A. *Querying XML Databases*. Computer Science Society, 2002. SCCC 2002. Proceedings. XXII International Conference of the Chilean, Pages:142–150, 6-8 Nov. 2002.

[12] W3C. XML Schema, Working Draft, July. 2004.

[13] W3C. Extensible Markup Language (XML) 1.0 (Third Edition)–W3C recommendation, February 2004.

[14] Katz, H. (Ed.). *XQuery from the Experts–Guide to the W3C XML Query Language*, Addison-Wesley, 2003.

[15] Boag, S.; Chamberlin, D.; Fernandez, M. F. et al. *XQuery 1.0: An XML Query Language*. Working Draft, W3C World Wide Web Consortium, November, 2003. www.w3.org/TR/2003/WD-xquery -20031112/ (accessed in May 2004).

[16] Murata, M.. *Hedge Automata: A Formal Model for XML schemata*, 1999. http://citeseer.ist.psu.edu/murata99hedge.html (accessed in October 2004).

[17] Chidlovskii, B. *Using Regular Tree Automata as XML Schemas*. In Proceedings of the IEEE Advances in Digital Libraries 2000 (ADL 2000), Washington, D. C., May 2000. http://citeseer.ist.psu.edu/chidlovskii99using.html (accessed in October 2004).

[18] Conallen, J. *Building Web Application with UML*. 2nd ed, The Addison-Wesley object-technology series, 2002.

[19] Qexo - The GNU Kawa implementation of XQuery. http://www.gnu.org/software/qexo/ (accessed in October 2004).

[20] DeMillo, R. A.; Lipton, R. J.; Sayward, F. G. *Hints on Test Data Selection: Help for the Practicing Programmer*. IEEE Computer, 11(4):34-41, April, 1978.

[21] Bouchou, B.; Alves, M. H. F. *Updates and Incremental Validation of XML Documents*. Proceedings of the 9th International Conference on Data Base Programming Languages (DBPL), Potsdam, Germany, September 6-8, 2003.