XML and Semantic Validation

Abstract

With XML as with SGML, we can have structural correctness, once they provide syntactic rules to state how to mark-up all the documents of the same family (of the same type); moreover, XML also imposes a working approach in which there is a complete separation between the structure of the data and the way it looks. So it is possible to avoid that someone will write a letter putting the ending before the body. Also, being purely declarative and completely independent of the processing, it is possible to swap documents between different systems without having to change them.
But even this way, there still is a lack of content validation. Therefore, as Ramalho et al [1] pointed out, if a document has the decrees published by some kings, and includes their birth dates, it is critical if there is a sentence in which a king is said to publish a decree before he was born.
In this paper we are concerned about reaching a way to automatically process a document in order to validate it semantically, avoiding this kind of incongruences that may spoil all a teams work.

Keywords:
XML, semantic validation, structured documents

1 Introduction

Like SGML, XML is a meta-language, for it allows the definition of new languages [2]. At first sight, SGML looks able to solve the problem, but it is complex and hard to learn. This will probably make it usable only by communities with an elevated technical level, and not by common users. This way, XML seems to be the appropriate choice. This is why we have chosen to

work with XML documents instead of SGML documents. In fact, XML offers a structured and consistent method to describe and transfer information. Maintaining the SGML characteristics, it separates the visual format from the information itself, allowing the independence from hardware and software platforms. The information within a XML document describes itself, allowing two XML applications to send and receive information without being concerned with the format of that information. Like a SGML document, a XML document has its structure definition within itself, which is called DTD. The need for structured information and valid facts, like the ones included in a History CDRom where everyone expects to find accurate data, motivates this work, in which we pretend to find and automatize a way of semantically validating documents.

Therefore, in section 2 we show what has been done in order to solve the problem; in section 3 we describe our approach; this approach is applied to a case study in section 4. Finally we present the ways in which our work may be improved.

2 State of the art

Ramalho et al [3] suggest two forms of dealing with the specification of a document semantics, which provides a way of document processing. They defend that, once SGML is totally independent of the further processing of the document, to do something with a certain document, we have to associate behaviour to it.

The first way of doing this, they said, is using a DAST (Decorated Abstract Syntax Tree); the approach consists in applying the technique traditionally used in the context of formal languages processing, to the documents' processing: the document semantics is represented by a DAST, which is formally specified by an attributes grammar. The first step in the development of such a semantic specification of documents is the design of the CFG (Context Free Grammar), which may be obtained directly from the DTD; after writing the CFG, it is necessary to associate attributes to its symbols and write the appropriate rules to evaluate the attributes in the context of each derivation rule; the attributes shall allow the declaration of the semantic conditions needed to restrict the set of valid sentences (static semantics) and transport all the information, directly or indirectly inferred from the original document

and necessary to make the translation (dynamic semantics).
The second form of doing this is using CAMILA or another model-based functinal approach (for instance, Haskell) —as a DTD is a type definition, it can be seen as a model from an algebraic point of view; this approach is good to prototyping semantics; in order to have semantics, we have to specify: models for the sorts; and definitions for the operators. CAMILA [4] is an attempt to make available, at the engineering level, the strategy of creating a mathematical mode , to reason on it and only then to calculate a solution. CAMILA uses formal methods and is currently being used both by Computer Science students and software engineers [5].
Ramalho et al [1] realized that in SGML it is not possible to implement the process in which the pre-conditions over the information are enforced in order to prevent the user from introducing erroneous data. Instead, a new SGML model should be built, extending the existing one, so that an adaptation of the SGML syntax would enable the expression of constraints, allowing some semantic validation. They propose an algebraic specification to implement both SGML and model extensions. In a way to warrant the preservation of some of the semantic characteristics of the documents, we have to associate restrictions to some of the DTD elements. There are two ways of enclosing invariants in the DTD:

- special comment sections —where invariants could be written, mixed with DTD declarations; new additions would not affect SGML syntax, granting the possibility of going on using the same SGML parsers in the structural validation process;

- invariants binded with an anchor —where the invariants will be written; the anchor will be placed in a special comment section; this alternative maintains the conciseness of the DTD and having a compiler that, given a DTD, generates a skeleton of the invariant file, the designer work becomes much more fast and safe.

Ramalho reformulated the first approach described above in his PhD Thesis [6], describing S4, which is a system that allows the integrated development of documents. This system consists of three editors: one for DTDs, another one for style and finally one specific, using SGen. In this work, Ramalho also describes a Constraint Language which syntax is a subset of the syntax of XQL [7]. A restriction's declaration will be like:

```
<--constraint CL-expression action-->
```

where:

constraint is the sintactic keyword.

CL-expression is a predicate applied to sub-areas of the document (nodes of the document's tree) . If its value is true, then the document is semantically correct.

action Describes the action to be performed if the boolean expression is not valid.

3 Our approach

Our approach to speciy the constraints is directly related to the last one described above. This way, we will give attributes to the key elements, according to the DTD nomenclature, and then use the Constraint Language to specify the constraints. Therefore, embedded in the DTD, which is specified as usual, we will have some comments, written in this particular language, which are describing the constraints that we want to enforce.

Afterwards, we will use a parser to generate an abstract tree with the information of the document. With this abstract tree, it will be possible to compare the attributes, taking into account the objective , which will complete the semantic validation of the document.

Parallelly, prototypes both based in compiler tools supported in Attribute Grammars and in Algebraic-functional environments are being done. These prototypes are to execute the predicate's tests over the Decorated Tree, and its details will be written elsewhere.

We are at present working in several examples. One of them is presented in the next section.

4 Case-study

The world of laws is very critical to erroneous data or incongruences.

For instance, if a lawyer goes to a finance department to ask for a certificate of something, it is advisable that all data is consistent. Let us suppose that

someone asks for a fiscal certificate, which is a certificate of the goods declared by someone's relatives by the time of his or her death (this is compulsory so that the goods can be inherited). That document should include the identification of the dead one and both the dates of that person's death and the current one, once this are required fields. The correspondent DTD for such a document, could be:

<!– document to ask for a fiscal certificate –>
<!– !DOCTYPE fiscal_certificate [–>
<!ELEMENT fiscal_certificate (header,body,ending)>
<!ELEMENT header (#PCDATA,department)>
<!ELEMENT department (#PCDATA)>
<!ATTLIST department place CDATA "Braga">
<!ELEMENT body (requester,request)>
<!ELEMENT requester (name,#PCDATA,CF,#PCDATA,address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT CF (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT request (#PCDATA,affinity,name,#PCDATA,data,#PCDATA,
village,#PCDATA,parish, #PCDATA,name,#PCDATA)>
<!ELEMENT affinity (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ATTLIST date value CDATA "19000101">
<!ELEMENT village (#PCDATA)>
<!ELEMENT parish(#PCDATA)>
<!ATTLIST parish place CDATA "Braga">
<!ELEMENT ending (#PCDATA,local,date,#PCDATA)>
<!ELEMENT local (#PCDATA)>

It is easy to realize that if the current date has occurred before the death date, we can not ask for that certificate. Therefore, we added an attribute *value* to the ELEMENT *date*. But this is not enough, for we still have to compare the dates. If we add the following constraint:

```
<--constraint /fiscal_certificate/body/request/date/@value <
                        /certidao_fiscal/ending/date/@value-->
```

the only thing we will lack is a way of automatically verifying this validity. Besides, people must ask for this kind of certificate in the finance depart-

ment of the last residence of the dead one area , so we should enforce that congruence too. The following constraint will do it, but the attribute *place* had to be added to the *department* and *parish* ELEMENTS.

```
<--constraint /fiscal_certificate/header/department/@place <
                          /certidao_fiscal/body/request/parish/@place-->
```

In the final version of the paper, we will present the complete example of this certificate, with the remaining alterations that shall be done in order to allow the semantic validation.

5 Conclusion

In the paper we have discussed a master thesis work aiming at the inclusion of contextual conditions in the XML DTDs with the purpose of allowing a static semantics validation of the XML documents according to that DTD. Checking for semantic correctness, completes the structural check traditional with mark-up languages and is a step forward towards the quality assurance in documents processing.

It still remains deciding about which programming language is the best one to use in order to validate the abstract tree obtained from the complete DTD (the original DTD plus the attributes of the key elements), but at moment there are clear options, trying to reuse existing languages for document handling.

This choice and the development of a checker prototype is the under development work involved in this thesis.

References

[**1**] Ramalho, Rocha, Almeida, Henriques, *SGML Documents: Where does quality go?*, GCA SGML/XML'97, 1997

[**3**] Ramalho, Almeida, Henriques, *SGML Documents: Two approaches*, GCA SGML'96, 1996

[**2**] Bradley, *The XML companion*, Addison-Wesley, 2nd Ed., 2000

[**6**] Ramalho, *Anotação Estrutural de Documentos e sua Semântica*, PhD Thesis, Universidade do Minho, 2000 (in Portuguese)

[**4**] Barbosa, Almeida, *CAMILA: A reference Manual*, Technical Report DI-CAM-95:11:2, DI (U. Minho), 1995.

[**5**] Almeida, Barbosa, Neves, Oliveira, *CAMILA: Formal Software Engineering Supported by Functional Programming*, Proc. II Conf. Latino Americana de Programacion Funcional (CLaPF97), La Plata, Argentina, October 1997.

[**7**] Robie, Lapp, Schach, *XML Query Language (XQL)*, QL'98 - The Query Languages Workshop, December 5, 1998.
Prentice-Hall, 1998.