schem@Doc: a web-based XML Schema visualizer

José Paulo Leal[1] and Ricardo Queirós[2],

[1] CRACS & DCC-FCUP, University of Porto, Portugal
zp@dcc.fc.up.pt

[2] CRACS & DI-ESEIG/IPP, Porto, Portugal
ricardo.queiros@eu.ipp.pt

Abstract. XML Schema is one of the most used specifications for defining types of XML documents. It provides an extensive set of primitive data types, ways to extend and reuse definitions and an XML syntax that simplifies automatic manipulation. However, many features that make XML Schema Definitions (XSD) so interesting also make them rather cumbersome to read. Several tools to visualize and browse schema definitions have been proposed to cope with this issue. The novel approach proposed in this paper is to base XSD visualization and navigation on the XML document itself, using solely the web browser, without requiring a pre-processing step or an intermediate representation. We present the design and implementation of a web-based XML Schema browser called schem@Doc that operates over the XSD file itself. With this approach, XSD visualization is synchronized with the source file and always reflects its current state. This tool fits well in the schema development process and is easy to integrate in web repositories containing large numbers of XSD files.

Keywords: Schema visualization, XML Schema, Transformation, Documentation, Interoperability.

1 Introduction

XML Schema is arguably the most used type definition language for XML documents. Thus, developing new XML documents types usually involves reading and understanding its definition in XML Schema (XSD). The teams that develop these definitions usually integrate persons with different backgrounds, including experts in the domain and experts in XML Schema. Typically, domain experts are not familiar with the complex features of this specification (e.g. reference bindings, extensions, restrictions and redefinitions). To make things harder for the all team, an XSD is serialized as an XML document, which makes it possible but difficult to read by humans. As postulated by XML design goals [1], although XML tags are usually self explanatory, and that is certainly the case with XML Schema, the resulting XML documents are usually anything but concise. This is not a problem for software processing XSD files, but is an extra burden for a person reading them in detail.

In this paper we present the design and implementation of a web-based tool for browsing XML Schema files called schem@Doc. Using this tool a human reader browses XSD files through a Web interface, using tree structures, tables and formatted text, rather than reading directly the source code. The schema structure and documentation are automatically generated from the document type definition, helping users to visualize the relationships among type definitions and reading documentation in plain natural language text formatted in XHTML. This tool does not require specialized knowledge of XML or XSD, so it can be used by novice users. Also, it does not require any pre-processing of XSD files. The user needs only to open an XSD file annotated with a special processing instruction on a conventional Web browser.

This tool was designed for Web based systems requiring the visualization of XSD files still under development, by users that may be unfamiliar with the XML Schema specification. Since it is a very light tool, it is well adapted for repositories with large numbers of XSD files. Although targeted primarily for interactive use, it also produces printed versions of XSD files that can be used as reference.

The remainder of this paper is organized as follows. The next section reviews basic concepts related to schema definition languages and existing approaches to browse schema definitions. Section 3 present an overview of schem@Doc and section 4 details its main features such as the navigation and visualization of the schema types, embedded Schematron support and example generation using instance documents. The last section draws conclusions from the presented work and outlines future developments.

2 Visualization of schema definitions

Document Type Definition (DTD) was the language inherited from SGML, to define types of documents in XML. Its many limitations [2] (e.g. insufficient data type support, lack of namespace awareness) lead to an official W3C recommendation for a schema language called XML Schema Definition language [3] (XSD) in 2001. XML Schemas are richer and more powerful than DTDs [4] and are written in XML. This new language overcame DTD limitations and provided several advanced features, such as the ability to build new types derived from basic ones [5], manage relationships between elements (similar to relational databases) and combine elements from several schemata.

In spite of its expressiveness, XSD lacks features to describe constraints on the XML document structure. For instance, there is no way to specify dependencies between attributes, or to select the content model based on the value of another element or attribute. To address these issues several schema languages were proposed, such as RELAX NG [6] (based on TREX [7] and RELAX [8]), DSD (Document Structure Description) [9] and Schematron [10]. The Schematron language provides a standard mechanism for making assertions about the validity of an XML document using XPath expressions and can be easily combined with W3C XML Schema documents.

These schema languages have different syntaxes but most of them support XML serialization. It is a well known fact that the XML formalism was not designed for the convenience of human readers [1]. Although tag and attribute names are usually self explanatory and they are meaningful to an expert, large documents with complex structure are difficult to understand and require specialized viewers. Specially for XSD files, several tools are available for browsing and displaying schema definitions in a user-friendly way. These tools are usually part of commercial XML Integrated Development Environments (IDE) such as XML Spy, Stylus Studio and <oXygen/> Oxygen, or plug-ins of general programming IDEs such as Eclipse, NetBeans or even Visual Studio .NET. There are a few standalone tools [11, 12, 13] designed to browse and generate documentation from XSD files. These approaches require either a preprocessing of the XSD files (converting them to HTML files), or installing specific software for browsing them. On the other hand, the majority of these systems are not open source, which limits its use in non-commercial research projects.

3 Overview

XML documents play such an important role in publishing and exchanging data on the Web that they might be considered a universal language tool [14]. The design of the schem@Doc tool fits the ubiquity of XML documents and it is easy to install on a web server and use from any browser linked to the Internet. Simplicity is the key in the design of this tool, since it is the best way to ensure reliability and efficiency. Hence, it should be no surprise that our schema viewer is composed of the following three components:

a transformation for converting an XSD to a web layout;
a script for handling user interaction;
a stylesheet to configure the layout.

The interplay of these three components to produce an interactive visualization from a XSD file is represented schematically in Fig. 1.

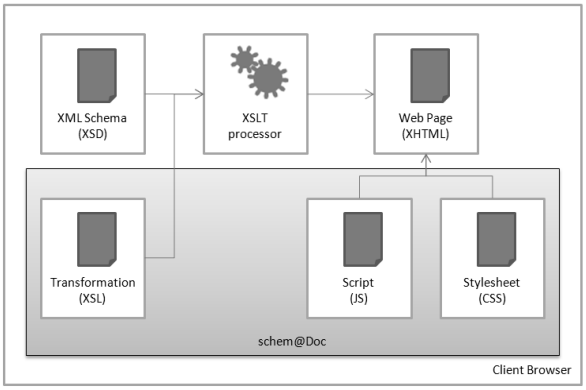

Fig. 1. Overall architecture of schem@Doc.

A Web browser is the only requirement for using the schem@Doc tool. The XSD file is opened from the Web browser and processed as a regular XML file. We rely on the XSLT processor, included in all recent versions of standard Web browsers, for the transformation of the XSD file into a web layout. The XSLT processor loads an XSLT transformation and produces a XHTML web layout that will in turn load the script and stylesheet files of the schem@Doc package.

To prepare an XSD file for schem@Doc it is only necessary to annotate it with a Processing Instruction (PI) in its prologue, to activate the XSLT processor and load the transformation file. The following code shows a PI in a schema file prologue:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="schemaDoc.xsl"?>
<xsd:schema …>
…
</xsd:schema>
```

The transformation file, as well as the script and stylesheet files it requires, are all part of the schem@Doc distribution package available for download from the following URL http://www.dcc.fc.up.pt/schemaDoc. In a typical installation these files are placed on web published directory and shared by several XSD files.

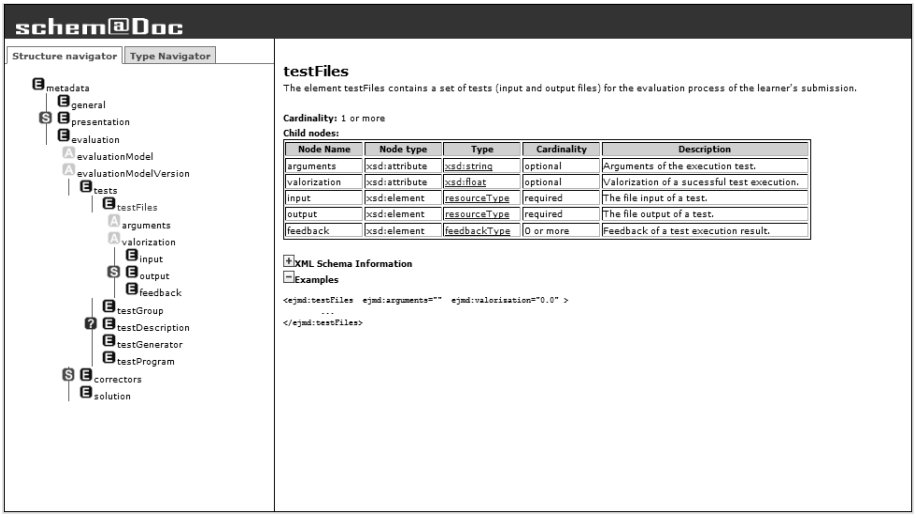

Fig. 2. XSD displayed by schem@Doc.

When an XSD file with the above PI is opened on a Web browser the user will be presented with an interface similar to the one depicted in Fig. 2. Of the left part the reader finds a navigation area with two tabs, each proving its own way to browse and select types defined on the XSD. Documentation on selected types is displayed on the right part. These features of schem@Doc are described in greater detail in the next section.

4 Main features

Schem@Doc is a Web-based tool to navigate through the types defined on a W3C XML schema documents and to view their documentation. The remainder of this section details its navigation, visualization and example generation features.

4.1 Navigation

When designing the web interface of schem@Doc we considered two distinct usage profiles. Novice or first-time users are expected to use schem@Doc to understand the structure of an XSD and identify its main types. Expert and recurrent users, that already know the basic structure of the XSD, are expected to search for documentation on specific schema type, to focus on technical details and may need to print a reference for off-line usage. To address this range of needs this tool includes two navigation modes, depicted side by side in Fig. 3.

> **Structural**: browse the schema file based on the structure of instances;
> **Reference**: list of complex and simple types in alphabetic order.

The former is the default and displays the structure of a valid XML instance document, with elements, attributes and respective types as tree nodes. The roots of these structures are element definitions on the XSD. Type definition referred by other types (using attributes `type`, `ref` or `base`) will appear as a child of the referring type in the hierarchy. These nodes can be expanded and collapsed to control visualization.

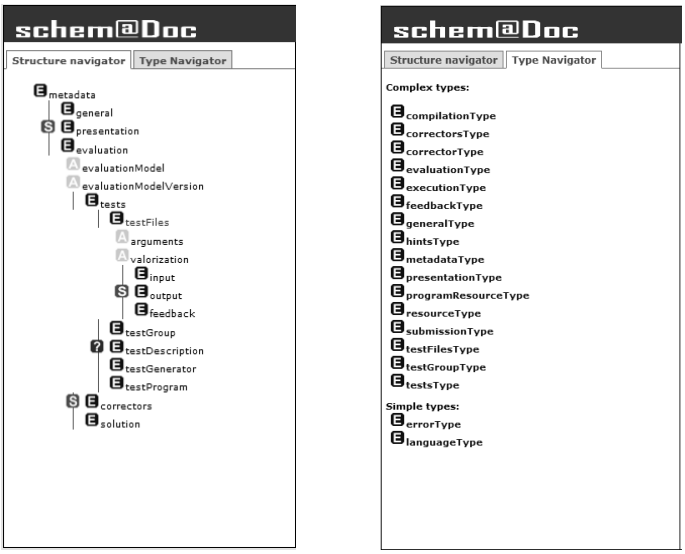

Fig. 3. Structural and Reference navigation.

The Reference navigation provides direct access to specific types and their documentation. This list is split up in complex and simples types and sorted in alphabetic order within each group.

Both navigation modes support the visualization of schema definitions loaded using the W3C XML Schema elements: import and include.

4.2 Views

After selecting an element/attribute in the navigation area (left side), the user visualizes automatic generated documentation related to that type on the viewing area in the right part of the screen. Fig. 4 shows the visualization presented to the user after selection of the `metadata` element on the structure navigator.

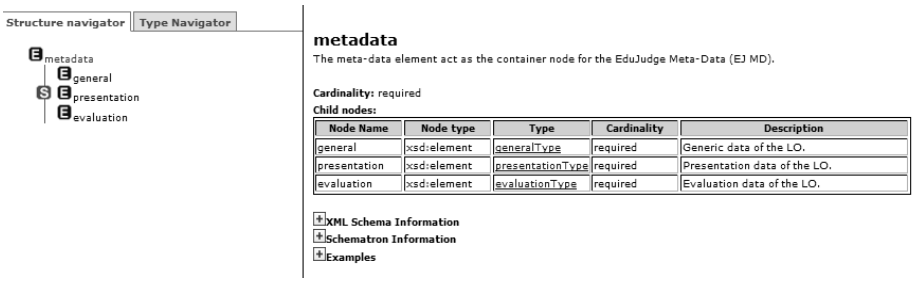

Fig. 4. Visualization of type documentation.

The generated documentation contains several sections, namely: element/attribute name, description, cardinality, a table with summary on child nodes information. It contains also technical information on schemata (both XSD and embed Schematron, presented in following sub-section) and usage examples. These last sections are meant for the expert user thus are initially presented collapsed.

As mentioned before, XSD files can be hard to understand and undocumented ones are even harder. To mitigate this problem the W3C specification includes the `annotation` element for adding complementary information. This element can annotate any XSD elements and has two child elements: `appinfo` and `documentation`. These elements can occur zero or more times inside the `annotation` element. The schem@Doc tool uses the `documentation` element to generate description for types.

The annotations allowed by the W3C specification are not enough for the needs of schem@Doc. For that purpose, we created a new document type, with a target namespace that separates new elements and attributes from the schema itself. It includes elements and attributes to define and control the automatic generation of the schema documentation. The following code illustrates the use of the type attribute to classify the descriptions found in the documentation element either as “extensive” or “summary”. Note that these examples assume a prior namespace declaration associated with “doc” prefix.

```
<xsd:element name="general" type="generalType">
  <xsd:annotation>
    <xsd:documentation doc:type="description-extensive">
      Extensive description here.<xhtml:br/>
    </xsd:documentation>
    <xsd:documentation doc:type="description-summary">
      Summary description here.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
```

The previous example shows also how XHTML tags can be used to format the schema elements descriptions, thus enriching the readability of the documentation. According to the W3C XML Schema specification `documentation` element may contain children from any namespaces. Since the content of this element is injected in a XHTML template, elements of this namespace are particularly recommended when using a XSD with schem@Doc. Typical XHTML elements used in this context are hypertext references to HTML pages with extended documentation, images containing diagrams and even tables to improve the documentation content layout.

4.3 Support for embedded Schematron

In spite of its expressiveness, there are cases where it is impossible to constraint an XML document instance using XML Schema, as in the following examples:

- Specify incompatible/complementary attributes;
- Force an element or attribute value to be greater/less than other;
- Make the content model depend on the value of an element or attribute.

To validate this type of constraints schema authors usually extend XSD files using at least one of these options: 1) combine XML Schema with others schema languages; 2) write code in a programming language to express the additional constraints; 3) use an XSLT/XPath stylesheet. To avoid ad-hoc solutions in schem@Doc we opted for combining XSD with other standard schema languages.

A usual candidate to perform this “second level of validation” is the rule-based validation language Schematron. The combination of the Schematron rules with the W3C schema can be done in two ways: as separate files, using pipeline validation languages such as the DSDL (Document Schema Definition Language) or Schemachine language, or as a single file, in this case embedding Schematron rules in the XML Schema using the `appinfo` element within the `annotation` element of a particular XSD element. This last approach fits our processing model best and thus was selected.

The following example shows an XSD that includes Schematron rules to define constraints that are beyond the capabilities of XML Schema. Specifically the pattern enforces the existence of a `time-submit` element if a `time-solve` element also exists.

```
<xsd:appinfo>
 <sch:pattern name="p2">
  <sch:rule context="time-solve">
   <sch:assert test="following-sibling::*[1]= time-submit">
If the element time-solve appears in the document, then the
timesubmit must appear after.</sch:assert>
  </sch:rule>
 </sch:pattern>
</xsd:appinfo>
```

Fig. 5 shows how schem@Doc renders this particular rule in the viewing part. It should be noted that Schematron information is placed inside a special section that is initially collapsed. As for XSD, Schematron documentation is automatically generated from its XML source.

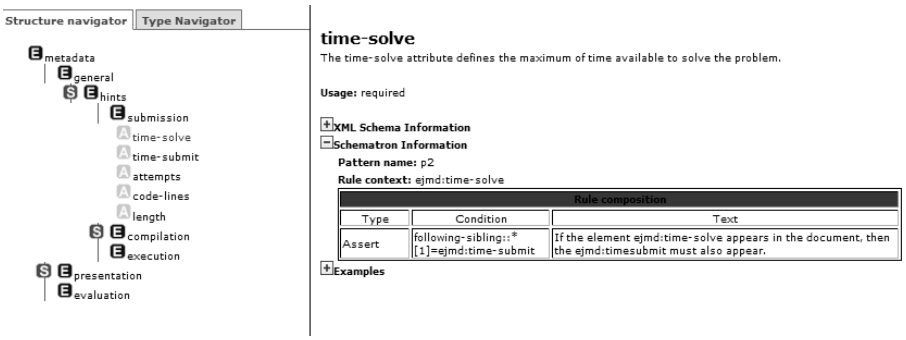

Fig. 5. Visualization of Schematron rules.

4.4 Visualization of instance examples

The automatically generated documentation includes an examples section to illustrate the use of a type with a fragment of a document instance. Surely, the XSD author can include these fragments in XSD `documentation` elements and even format them using XHTML. This special type of documentation could have been just another value in the `type` attribute introduced in sub-section 4.2. However, producing examples this way would be time-consuming and error-prone.

To produce document instance fragments as example for particular elements or attributes, schem@Doc uses a set of instance documents defined by the XSD author. To generate an example fragment for documenting an element definition, the tool simply searches the document instances for occurrences of those elements. These instance files may be created from the XSD file itself using an IDE (e.g. Eclipse) or a XML specialized editor (e.g. XESB [15]). The visualization of the fragment example is controlled by using the schem@Doc namespace at two levels. At the root level it uses the `href` attribute to identify document instance(s). At the element level it uses the `order`, `childDepth` and `parentDepth` attributes to control, respectively,

what document instance should be used and the depth level visualization of the example. The following example illustrates the use of these attributes to control the visualization of examples for a particular element named `presentation`.

```
<xsd:schema doc:href="ex1.xml ex2.xml" …>
   <xsd:element name="presentation" doc:childDepth="0"
   doc:parentDepth="1">
      <xsd:annotation>
        <xsd:documentation>…</xsd:documentation>
      </xsd:annotation>
   </xsd:element>
</xsd:schema>
```

The `childDepth`/`parentDepth` attributes defines how many ancestors and descendants levels should be shown in the example. The absence of the `order` attribute, at the element level, will make the tool to iterate over all instance documents included in the root level `href` attribute, until it finds the first occurrence of the `presentation` element. Fig. 6 shows how schem@Doc renders the example fragment.

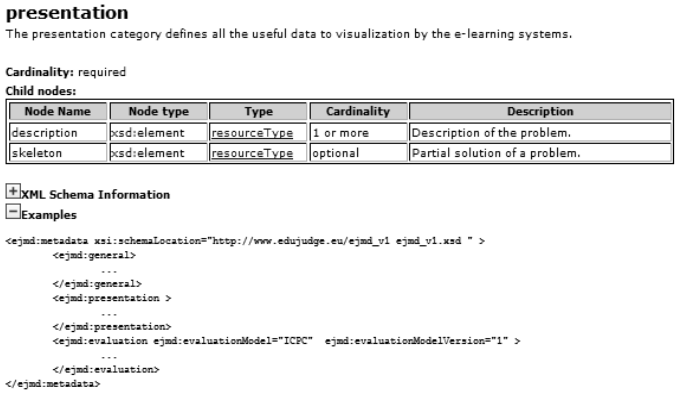

Fig. 6. Visualization of fragment examples.

5 Conclusions and future work

In this paper we described the design and implementation of a Web-based XML Schema documentation generator called schem@Doc. The main contributions of this work are a new approach to the visualization of schema definitions and a tool to present XML Schema files on the Web. The schem@Doc tool provides direct visualization of the XSD files, generating documentation on-the-fly from the XML schema file itself, without requiring the installation of add-ons or plug-ins at the client side. The tool is being distributed in open source, and is available for testing and download at the following URL `http://www.dcc.fc.up.pt/schemaDoc`.

Although the current implementation is already being used for documenting XSD files developed as part of an eLearning project, it has some limitations that we will address in the near future:

- Ensure full support for the XML Schema specification (e.g. field, key, keyref, redefine, selector, unique elements and abstract, substitutionGroup attributtes);
- Handle recursive definitions in the navigation view;
- Improve the initial responsiveness by generating type information on a lazy basis .

Make extensions of schem@Doc compatible with existing editors to avoid the need to edit the XSD source code. After addressing these limitations we plan to evaluate the this tool experimentally, testing both its usability and utility with users with different knowledge levels of XML Schema . We plan also to improve the support to printed versions of XSD files that can be used as reference and develop a search feature based on XPath expressions.

Acknowledgements

This work is part of the project entitled “Integrating Online Judge into effective elearning”, with project number 135221-LLP-1-2007-1-ES-KA3-KA3MP. This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

References

1. Birbeck, M. And others: Professional XML, Wrox, 2ª edição, 2001.
2. Harold, E.R.: The XML Bible, 3rd edition, Hungry Minds, 2004.
3. Fallside, D.C. (Ed.): XML Schema Part 0: Primer Second Edition. World Wide Web Consortium, Recommendation, 28 October 2004.
4. Song, G., Zhang, K.: "Visual XML Schemas Based on Reserved Graph Grammars," itcc, vol. 1, pp.687, International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 1, 2004.
5. Biron, P. V. and Malhotra, A. (ed.). XML Schema Part2: Datatypes Second Edition. W3C Document, October 2004.
6. Clark, J, Murata, M.: RELAX NG Specification, OASIS Committee Specification, December 2001, http://relaxng.org/spec-20011203.html
7. Clark, J.: TREX - Tree Regular Expressions for XML. Thai Open Source Software Center, 2001, http://www.thaiopensource.com/trex/.
8. Murata, M.: RELAX (Regular Language description for XML). INSTAC (Information Technology Research and Standardization Center), 2001, http://www.xml.gr.jp/relax/.
9. Moller, A.: Document Structure Description 2.0, BRICS, 2002, http://www.brics.dk/DSD/dsd2.html
10. The Schematron, An XML Structure Validation Language using Patterns in Trees, http://www.ascc.net/xml/resource/schematron/schematron.html
11. DocFlex/XSD Home Page: http://www.filigris.com/products/docflex_xsd/

12. Kilkelly, F.: SchemaViewer1.0, Internet Document, Nov. 2002. http://xml.coverpages.org/SchemaViewer10-Ann.html.
13. VisualSchema Home Page: http://www.visualschema.com/
14. Mignet, L., Barbosa, D., Veltri P.: The XML Web: a First Study, International World Wide Web Conference. Proceedings of the 12th international conference on World Wide Web, 2003.
15. Queirós, R., Leal, J.P.: Xesb: um editor XML para as massas. In José Carlos Ramalho, Alberto Simões e João Correira Lopes (eds.), In Actas da 3a Conferência Nacional XML: Aplicações e Tecnologias Associadas, (XATA 2005), Universidade do Minho, Braga, 2005.