Relaxed—on the Way Towards True Validation of Compound Documents[*]

Jirka Kosek
University of Economics, Prague
Dept. of Information and Knowledge Engineering
W. Churchill Sq. 4
130 67 Praha 3
Czech Republic

jirka@kosek.cz

Petr Nálevka
University of Economics, Prague
Dept. of Information and Knowledge Engineering
W. Churchill Sq. 4
130 67 Praha 3
Czech Republic

petr@nalevka.com

ABSTRACT

To maintain interoperability in the Web environment it is necessary to comply with Web standards. Current specifications of HTML and XHTML languages define conformance conditions both in specification prose and in a formalized way utilizing DTD. Unfortunately DTD is a very limited schema language and can not express many constraints that are specified in the free text parts of the specification. This means that a page which validates against DTD is not necessarily conforming to the specification. In this article we analyze features of modern schema languages that can improve validation of Web pages by covering more (X)HTML language constraints then DTD. Our schemas use combination of RELAX NG and Schematron to check not only the structure of the Web pages, but also datatypes of attributes and elements, more complex relations between elements and some WCAG checkpoints. A modular approach for schema composition is presented together with usage examples, including sample schemas for various compound documents (e.g. XHTML combined with MathML and SVG).

The second part of this article contains description of Relaxed validator application we have developed. Relaxed is an extensible and powerful validation engine offering a convenient Web interface, a Web-service API, Java API and command-line interface. Combined with our RELAX NG + Schematron schemas, Relaxed offers very valuable validation results that surpass W3C validator in many aspects.

Categories and Subject Descriptors

D.3.2 [**Software**]: Programming Languages—*Language Classifications*; I.7 [**Computing Methodologies**]: Document and Text Processing

General Terms

Standardization, Languages, Verification

Keywords

XML, XHTML, validation, RELAX NG, Schematron, compound documents

1. INTRODUCTION

The Web is a pervasive platform for sharing various kinds of documents and for providing remote access to applications. Since its birth in the beginning of 90s, Web has been developed as a heterogeneous system that should connect agents (browsers) and servers developed by different software vendors. In order to maintain interoperability, the Web was built around a set of standardized protocols and data formats which together define the Web environment. By adhering to standards, authors of Web pages should be assured that their pages will display correctly in browsers, because browsers should correctly implement support for Web standards. The situation here is not yet perfect—there are plenty of non-conforming pages available on the Web and at the same time not all features of the Web standards are correctly supported by all browsers.

But certainly there was a big movement for Web standards during the last years. However, the will of authors to adhere to Web standards is usually not sufficient—standards are complex and there are just few people around the world who know all details of (X)HTML language from a top of their head. The rest of the Web authors should check their pages using some sort of an automatic validation or a conformance testing tool. Several such tools exist, the most known is the W3C Markup Validation Service[1].

The problem of the W3C validator (and of many other similar services) is its sole dependency on a formal validation against DTD (Document Type Definition). But HTML and XHTML are not defined only in terms of DTD. Respective specifications [10][2] contain a lot of constraints which can not be expressed using DTD. This means that if some document passes validation against DTD it is not necessarily conforming to the (X)HTML standard.

To overcome limitations of DTD based validation we propose usage of more powerful schema languages that can cover more constraints than DTD. From the (X)HTML validation point of view the following features are the most missing ones:

---

[1]http://validator.w3.org/

- validation of element and attribute content against data types;

- support for validation of compound documents composed from XML fragments coming from various namespaces;

- emulation of SGML exclusions for XHTML document types;

- ability to express advanced constrains beyond structural validation.

In this paper we describe how can modern schema languages be utilized for this task (Section 2). Then we describe how we have reformulated XHTML constraints using RELAX NG and Schematron (Section 3). The Section 4 describes Relaxed validator that uses previously mentioned techniques to provide better validation service to Web content developers. In the last part of the article we outline directions for future development of our validation approach and the Relaxed application.

2. OVERVIEW OF THE MODERN SCHEMA LANGUAGES

Shortly after launching the XML 1.0 standard it became apparent that DTDs are lacking several critical features needed in many XML applications. The two most important and missing features were support for data types and namespaces. DTD does not have the concept of data types. Every element or attribute value is considered to be almost an arbitrary string. It is not possible to define content to look like a number, a date or a string with a given length.

Namespaces allow to combine several XML vocabularies in a single XML document. The resulting document is often called a compound document. Compound documents promote information reuse and encourage to invent new, previously unknown ways to process information stored inside documents.

Several new schema languages were created to overcome DTD limitations. Many of them were just prototypes or proprietary ones. Only two new schema languages get broader acceptance—W3C XML Schema [11] and RELAX NG [6]. Both of those languages have very good support for data typing and namespaces. At the same time there are also big differences between those languages. Formal comparison of DTD, W3C XML Schema and RELAX NG can be found in [9]. To summarize briefly, RELAX NG is the most expressive language and offers the greatest flexibility in modularizing and combining schemas. This is the reason why RELAX NG is very popular for creating complex document oriented schemas like TEI[2] or DocBook[3]. W3C XML Schemas are enforcing unambiguity of schema[4] and thus they are very popular in scenarios where unambiguous mapping from XML to object or database representation is required.

All previously mentioned schema languages are so-called *grammar based* languages. They define grammar of the

---

[2] http://www.tei-c.org/

[3] http://docbook.org

[4] This rule is called UPA (Unique Particle Attribution) in a W3C XML Schema terminology.

XML vocabulary by enumerating all elements and their content models. However, this approach is not sufficient in all situations. Some more complex constraints and relations between values in a document can not be captured in a grammar based approach. This problem can be solved by using *rule based* schema languages like Schematron. Schematron schema consists of a set of XPath expressions that are evaluated against the validated document.

Each schema language is good only in constraining of some document facets. In order to gain better validation results it is reasonable to combine several schema languages and validate document against all of them. Combination of Schematron with RELAX NG or W3C XML Schema is an example of such powerful constraint language. Moreover extensibility of both W3C XML Schema and RELAX NG allows to embed Schematron rules directly into principal, grammar based schema.

To prove the power of RELAX NG and Schematron for Web documents validation we created a validation application called Relaxed which utilizes RELAX NG and Schematron schemas for XHTML.

3. REFORMULATION OF XHTML IN RELAX NG AND SCHEMATRON

Relaxed is an open source project which aims to help authors of HTML documents to achieve better international standard compliance using an automated validation service. Modern state-of-the-art XML technologies allow Relaxed to provide better and more detailed validation results than other similar services including W3C validator. The project consists of two essential areas. The first is an HTML schema written in RELAX NG with embedded Schematron patterns which is further discussed in this section. The second but nevertheless important area is a validation engine which is discussed in Section 4.

The W3C's HTML specifications usually consist of two different parts. The first part is the specification text which is basically a set of restrictions, recommendations and explanations provided in a verbal form. The second part is a DTD which is another source of restrictions written in a formalized form. To write standard compliant documents authors need to follow both parts, but only the DTD based part may be validated automatically. This means that authors still need to be familiar with the verbal specification, but most of them aren't. This leads to a huge amount of non-standard documents in today's Web environment which causes many interoperability problems.

The aim of Relaxed is to express most of the verbally formulated restrictions in a form that can be automatically validated to reduce the authors' knowledge requirements and to achieve a better HTML specification compliance. To reach this goal, Relaxed uses one of today's most expressive combination of schema languages, RELAX NG and Schematron. The biggest advantage over other similar approaches (including W3C XML Schema combined with Schematron) is aside from expressivity also some sort of elegance of use, possibility to easily integrate both languages and a great support for modularity.

3.1 Modularity

Relaxed HTML schemas are derived from the work of James Clark "Modularization of XHTML in RELAX

NG" [7]. Those schemas demonstrate the power of RELAX NG modularity. If you look at XHTML modularization implementations using DTD or XML Schema, you see they need a specific model or driver schema for every used module combination. In RELAX NG, modularity is much more straightforward. The only thing that needs to be done is to include all desired modules and your new schema combination is done on the fly.

Suppose that a separate hypertext module should add possibility to use a linking element (a) everywhere in the Web page text. This can be accomplished by a simple definition that adds an a element into a list of elements which are permitted at the inline level.

```
<define name="Inline.class" combine="choice">
  <ref name="a"/>
</define>
```

There is no need to completely redefine the content model in which a occurs as is necessary in DTD and W3C XML Schema.

Example 1: Modularity in RELAX NG

HTML4.01/XHTML1.0 specification defines three language subsets (*strict*, *transitional* and *frameset*) and every of them has it's own monolithic schema. Those three schemas contain a huge number of duplicities. Most of the definition is basically repeated in an unchanged form in all of them. This approach is really error-prone. One small change in the shared language subset requires definition modifications across all three different schemas. It is difficult to keep such schemas consistent and the schemas are also hard to read and understand. For instance, to find out which elements are shared by all three subsets, you have to go through all of the schemas. It's also difficult to tell which aspects of the language are specific for a particular language subset.

Relaxed schemas solve all the previously outlined problems. They define the three language subsets just by including the right modules. Common modules are shared among all the subsets. There is no duplicity and separation into modules brings better readability and easier maintenance. With such modular architecture it is easy to fine-tune the level of restriction during the validation process. Whether you want to validate HTML including the WCAG specification [5] or you want to allow elements from different namespaces, the only thing you need to do is just to include the appropriate module.

3.2 Ability to easily extend schemas

Once we have basic schemas for XHTML ready, we can start to use them as a basis for compound documents schemas. This is a very challenging task especially if you only have experiences with DTD or W3C XML Schema. However, with a good initial schema organization it is quite easy to create such schema in RELAX NG.

Assume that we want to create schema for a XHTML+MathML validation. This means that proper MathML elements can be used anywhere inside the `body` element. Initial XHTML schemas used in Relaxed define a wildcard named pattern `otherNamespaceElement` which is used to allow non-XHTML elements and attributes almost

everywhere within XHTML documents (see Example 2). This is used to allow unrestricted use of foreign elements and attributes in XHTML.

```
<define name="otherNamespaceElement">
  <element>
    <anyName>
      <except>
        <nsName ns="http://www.w3.org/1999/xhtml"/>
      </except>
    </anyName>
    <zeroOrMore>
      <choice>
        <attribute>
          <anyName>
            <except>
              <nsName
                ns="http://www.w3.org/1999/xhtml"/>
            </except>
          </anyName>
        </attribute>
        <text/>
        <ref name="otherNamespaceElement"/>
      </choice>
    </zeroOrMore>
  </element>
</define>
```

Example 2: Wildcard named pattern

But an XHTML+MathML schema should be more restrictive. It should precisely prescribe where MathML elements may occur and at the same time the schema must validate MathML fragments against a MathML schema. This means that the wildcard pattern must be redefined to cover only the elements outside from XHTML and MathML as is shown in Example 3.

Example 4 shows the final schema for compound documents. We just need to include the standard XHTML schema, but during this inclusion we redefine the wildcard pattern to disallow also MathML elements. Then we add the MathML schema as a permitted element to all inline and block level contexts in XHTML.

We have used this approach to create schemas for XHTML+MathML, XHTML+SVG and XHTML+MathML+SVG. Every of those schemas comes in three variants—strict, transitional and frameset.

3.3 Support for datatypes

A big advantage of using RELAX NG over DTD are definitely datatypes. DTD's datatypes are very elementary and incompetent to fully express the complexity of HTML datatype requirements. By default RELAX NG contains just two built-in data types, but those are fully extensible using external datatype libraries. A good example is the W3C XML Schema datatype library implementation, which brings a set of thirty seven carefully selected types to be used within RELAX NG. This set may be further restricted by setting intervals or by using regular expressions.

Relaxed schemas reflect most of the HTML datatype requirements including lengths and multilengths, characters, pixels, targets, font sizes, colors and many more. Two representative datatype definitions are shown in Example 5.

Assume that this schema is saved in `ns-xhtml-mathml.rng`.

```
<element
    xmlns="http://relaxng.org/ns/structure/1.0">
  <anyName>
    <except>
      <choice>
        <nsName
          ns="http://www.w3.org/1999/xhtml"/>
        <nsName
          ns="http://www.w3.org/1998/Math/MathML"/>
      </choice>
    </except>
  </anyName>
  <zeroOrMore>
    <choice>
      <attribute>
        <anyName>
          <except>
            <choice>
              <nsName
                ns="http://www.w3.org/1999/xhtml"/>
              <nsName
                ns="http://www.w3.org/1998/
                                  Math/MathML"/>
            </choice>
          </except>
        </anyName>
      </attribute>
      <text/>
      <ref name="otherNamespaceElement"/>
    </choice>
  </zeroOrMore>
</element>
```

Example 3: Wildcard pattern with excluded XHTML and MathML elements

3.4 Using Schematron to enforce additional checks

There are several reasons why embedding Schematron patterns may help to enhance the schema definition. Some restrictions are simply inexpressible using RELAX NG. Those are usually complicated structural conditions. A nice example is the requirement that `select` elements with an absent `multiple` attribute can not have more `selected options` (see Example 6), another example is the WCAG requirement for a proper heading section order. Even a simple requirement to keep `id` and `name` of the same element equal, would be hard to express using RELAX NG only.

Another reason for using Schematron are situations when RELAX NG definition is possible but too complicated. Some of the restrictions may be expressed using Schematron with a simple XPath expression while RELAX NG implementation would involve many lines of code or changes across the schema modules. For instance, HTML specification requires a `form` element to have no other `form` elements nested. While RELAX NG definition would be quite complicated, possibly involving the introduction of a completely new module, by using Schematron we just need to add a very simple XPath rule shown in Example 7.

Combining RELAX NG with Schematron may even sim-

```
<grammar ns="http://www.w3.org/1999/xhtml">

<include href="xhtml-strict.rng">
  <define name="otherNamespaceElement">
    <externalRef
        href="exclude/ns-xhtml-mathml.rng"/>
  </define>
</include>

<define name="Block.class" combine="choice">
  <externalRef href="../mathml/mathml2.rng"
        ns="http://www.w3.org/1998/Math/MathML"/>
</define>

<define name="Inline.class" combine="choice">
  <externalRef href="../mathml/mathml2.rng"
        ns="http://www.w3.org/1998/Math/MathML"/>
</define>

</grammar>
```

Example 4: XHTML+MathML compound document schema

Interesting HTML datatypes defined in RELAX NG.

```
<define name="tabindexNumber.datatype">
  <data type="nonNegativeInteger">
    <param name="pattern">[0-9]+</param>
    <param name="minInclusive">0</param>
    <param name="maxInclusive">32767</param>
  </data>
</define>

<define name="Target.datatype">
  <data type="string">
    <param name="pattern">_(blank|self|parent|top)
                          |[A-Za-z].*</param>
  </data>
</define>
```

Example 5: HTML datatypes

plify the schema modularity design. In RELAX NG it is usually very simple to extend some elements by importing an additional module. You just need to specify the type of mutual combination and elements with same names are automatically merged. But element restrictions are more complicated. In this case you need to completely redefine some element within an import statement to create its more restrictive version. In some situations this may cause problems as imports usually take place outside modules in the main scheme where we specify all modules which shall be used. To keep this schema simple it is usually inconvenient for us to invade this part with any additional element definitions. A very convenient solution is to use Schematron for this additional restriction by including a simple rule into the module where it really belongs. Such restriction is automatically applied in case the module is imported.

As you can see RELAX NG and Schematron are very good fellows. Combining them together brings a strong value added. It may not only increase the expressiveness, but also simplify the schemas and make them more human-readable. Both languages have a very distinct philosophy

Sometimes grammar based languages cannot express what is otherwise easily expressible by a rule based language.

```
<sch:rule context="html:select">
  <sch:report test="not(@multiple) and
               count(html:option[@selected]) > 1">
    Select elements which aren't marked
    as multiple may not have more
    then one selected option.
  </sch:report>
</sch:rule>
```

Example 6: Selected options

In some situations, using grammar based languages may be an overhead. Schematron is used here to emulate SGML exclusions feature unavailable in XML DTD.

```
<sch:rule context="html:form">
  <sch:report test="descendant::html:form">
      form element can not have any nested
      form elements
  </sch:report>
</sch:rule>
```

Example 7: Disabling nested forms using Schematron

and their use is more or less suitable in different situations and for different purposes. Smart decisions about where to use which language can significantly improve the schemas by getting the best value of both.

3.5 Example of improved validation

The power of RELAX NG and Schematron can be seen on Example 9. This page contains four HTML specification violations:

- `border` attribute contains value `10%`, but according to the specification it should contain only integer value[5], not percent value.

- `color` attribute has `nougat` value. According to the specification[6] this attribute can contain only numeric color value (sRGB) or one from the sixteen predefined color names[7].

- The second part of the page contains `form` element which is nested inside another `form` element. This is also prohibited[8].

- The last form has attributes `name` and `id` with distinct values (`form1` and `form2`). But according to the specification[9] these attributes must share same value when used on the single `form` element.

If you try to validate this page with W3C Markup Validation Service you are notified that this page is valid. But

---

[5]http://www.w3.org/TR/html4/struct/tables.html#adef-border-TABLE

[6]http://www.w3.org/TR/html4/present/graphics.html#adef-color-FONT

[7]http://www.w3.org/TR/html4/types.html#type-color

[8]http://www.w3.org/TR/html4/interact/forms.html#h-17.3

[9]http://www.w3.org/TR/html4/struct/links.html#anchors-with-id

The transitional subset of XHTML is with few exceptions less restrictive than the strict subset. One of the exceptions is the `param` element and it's `name` attribute which is optional in strict but mandatory in transitional. The following rule embedded in one of the transitional specific modules can easily solve this requirement.

```
<sch:rule context="html:param">
 <sch:assert test="@name">
  The name attribute of param element is mandatory.
 </sch:assert>
</sch:rule>
```

Example 8: Schematron is a good option to use in case of restrictive modularity

as you can see, this page contains four violations of HTML specification. In contrast validation against our RELAX NG and Schematron schema reveals all errors (see Figure 2).

```
<html>
  ...
  <body>
    ...
    <h1>Datatype tests</h1>
    <table border='10%'>
      <tbody>
        <tr>
         <td><font color='nougat'>B</font></td>
        </tr>
      </tbody>
    </table>
    ...
    <h1>Nested form</h1>
    <form action='process.form'>
      <div>
        <form action='process.subform'>
         <p>Somethings wrong</p>
        </form>
      </div>
    </form>
    ...
    <h1>Id and name are not the same</h1>
    <form name='name1' id='name2'
        action='process.form'>
      <p>Somethings wrong</p>
    </form>
    ...
  </body>
</html>
```

Example 9: Sample HTML document with errors

4. RELAXED APPLICATION

Relaxed application was designed to be fully configurable, extensible and to provide a transparent and easy-to-use validation interface. Beside the application programming interface (API), which may be used to embed Relaxed validation functionality into other applications or tools, Relaxed features a Web-based user interface and a command-line user interface as well. Where command-line interface is mainly intended for testing and administration, the Web-based interface may serve for two different purposes. An HTML

font-end may be rendered by a Web browser to allow a broad variety of users to interact with the validation engine, whereas an XML result output is intended for automated processing by another service or application.

4.1 Web-based user interface

Users may access Relaxed validation functionality through an HTML user interface with their Web browser. The only mandatory parameter to start a validation process is the URL of the source document. But the user can specify a number of additional validation parameters as well. Documents are implicitly validated against XHTML1.0/HTML4.01 specification, but as you can see on Figure 1 there are several other possibilities to choose from the "restriction" select box. For instance the user may include WCAG 1.0 [5] restrictions, choose to disallow elements from different namespaces, to validate compound documents etc...

By default the document type declaration is automatically detected by Relaxed. The "doctype" select box allows user to force Relaxed to handle his document as a different document type than declared. XHTML1.0/HTML4.0 specifies three language subsets with different level of strictness. One reasonable approach to use this validation parameter is to check your document's validity for a less or more restrictive language subset. In case your document is valid for a different document type than declared, you may think about changing your document type declaration. You may also use this parameter to bypass the Relaxed autodetection mechanism for some reason.

At the bottom in Figure 1 you see three check boxes. They represent another validation parameters which can be also specified. The check box labelled "view source" appends the complete document's source at the bottom of the validation output and links the error messages to the corresponding source lines. "Brief output" hides messages with a low severity level when checked. The last check box called "dirty parser" makes Relaxed convert the current document into well-formed XML before the validation takes place. This helps authors to focus on the structural problems rather than on syntactical ones. The purpose of those three parameter's is mainly to rearrange the validation output to make it more readable and easy to understand.

In Figure 2 you see an example of the validation output. The demonstrational document is considered to be invalid. There is one info message at the top telling the user that Relaxed has successfully detected the document type. This is followed by four error messages. The first one complains about a mismatched `name` and `id` attribute within the scope of the same element. The second error highlights a `form` element nested into another `form` element. And the third and fourth error reflects an HTML datatype violation. All of the mentioned errors are XHTML1.0/HTML4.0 specification violations and none of them is reported when using the current version of W3C validator (see Section 3.5).

4.2 Relaxed internals

Relaxed validation engine is written in Java and uses a number of third-party libraries. One of those libraries is the Sun's Multi-Schema Validator (MSV), which is used to validate XML against RELAX NG schemas. Another reason for using MSV is that it implements a common validation interface called JARV. This interface allows Relaxed to eas-

ily switch to a different validation library or to configure Relaxed to use schemas written in several different schema languages and combine them.

Relaxed was also designed to validate Schematron patterns embedded in RELAX NG schemas. This step is achieved through several XSLT transformations. Those are processed using the Saxon[10] library. One of the reasons to use Saxon are its extension capabilities. As the standard transformation mechanism doesn't provide information about current source file line numbers, one of the Saxon's extensions is used to provide this information, which is essential for validation results usability and convenience.

Using current XML technologies gives Relaxed the power to validate more than the others, but not all real-life HTML documents are XML instances. Despite XML brings many advantages to HTML, there is still a significant number of documents adhering to the HTML 4.x specification, which is an application of SGML. As SGML is less restrictive than XML, those documents mostly don't meet the requirements for a well-formed XML which makes processing of those documents using common XML tools impossible. Enabling Relaxed to validate XHTML documents only would make its applicability in current heterogeneous Web environment very limited. That's a reason why Relaxed uses another library called TagSoup. A specially modified version of TagSoup brings the possibility to convert HTML documents to well-formed XML (basically to XHTML) and to keep users informed about most SGML syntax violations at the same time. TagSoup architecture always guarantees a well-formed output which makes possible to validate almost all of the current real-life HTML documents using Relaxed validation engine.

4.2.1 Application architecture

Relaxed architecture consists of several integrated components. The centerpiece of the application is a validator container, which contains a set of different validation components. The container basically manages the validation process. It invokes all registered validation components in a sequence and finally aggregates their validation results.

A validation component is designed to handle a large number of validation requests against a small set of prearranged schemas. Preparing a schema for validation may be a quite expensive operation. For that reason the component caches every prepared schema when it is first requested in order to make it ready for future immediate use.

Currently there are two different validation components implemented in the Relaxed project. One of them is basically a wrapper around the JARV validation interface. The component can be be configured to use any schema language supported by any validator which implements the JARV interface. This validation component is used in the Relaxed project to validate the modular RELAX NG schemas.

The second component does the embedded Schematron pattern validation. The schema preparation process is a series of three XSLT transformations. First transformation extracts the embedded Schematron patterns from RELAX NG schema modules. After that, Schematron is transformed into an XSLT stylesheet. Finally the third transformation adds Saxon line number extension elements. The outcome of those transformations is again an XSLT stylesheet which can be directly used for validation of the individual documents.

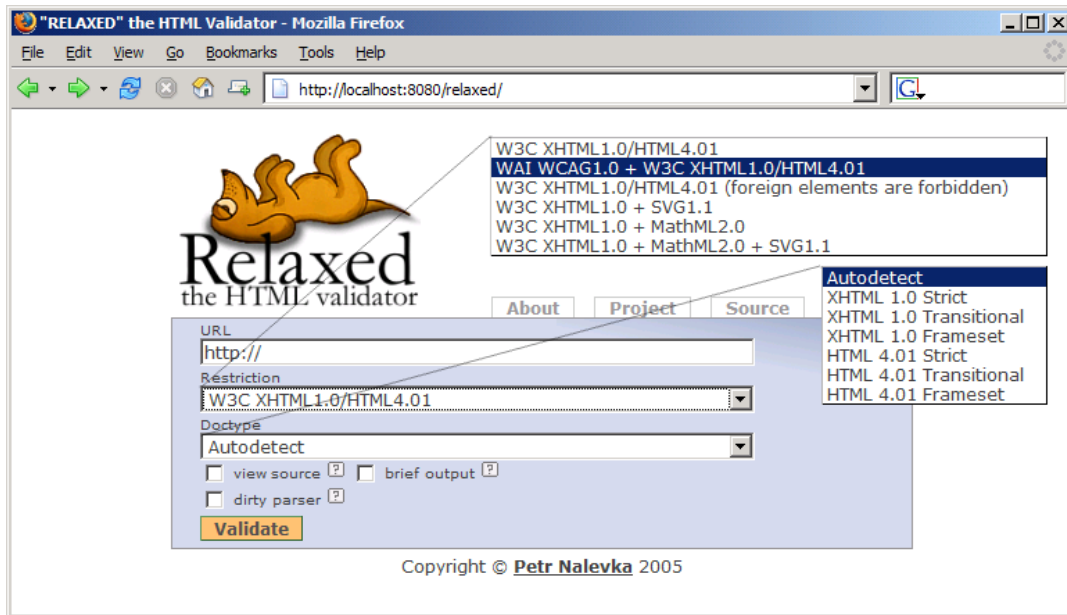[10]`http://saxon.sf.net`

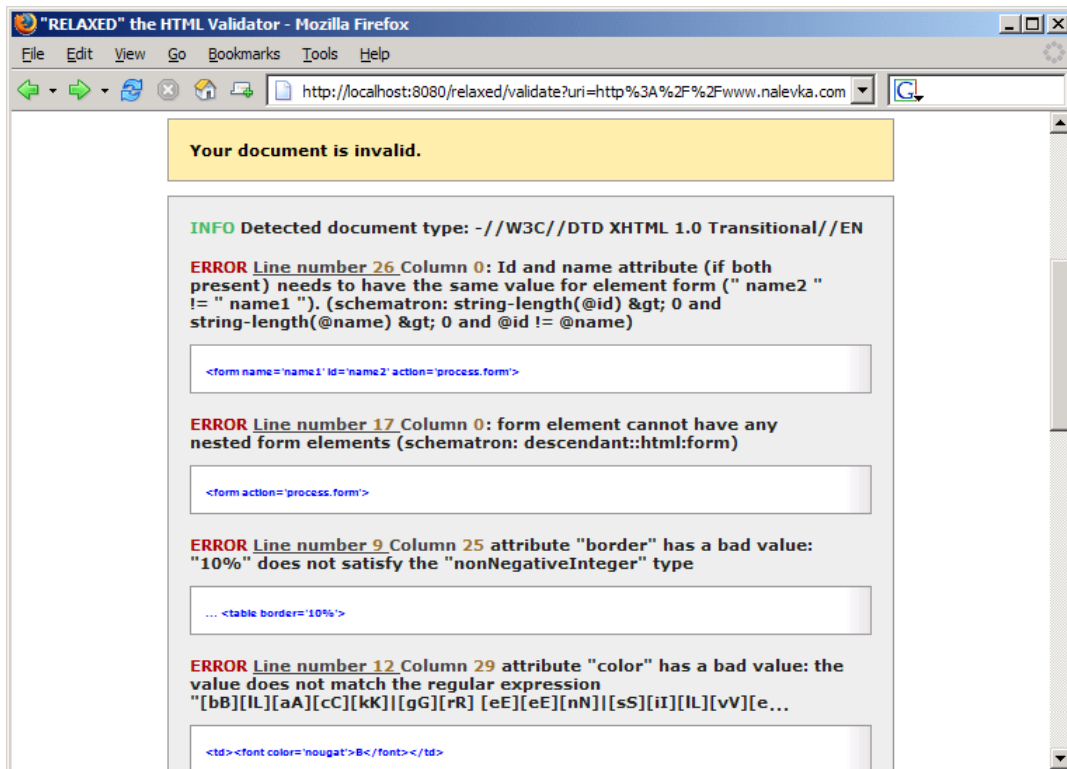Figure 1: Relaxed user interface

Figure 2: Validation output

Such validation process is again nothing else than another XSLT transformation.

Another part of the architecture is a filter chain mechanism. Before any document is processed by the validator container, it is first passed into a filter chain. The chain can consist of a number of filters. Filters may modify the validated document content with respect to the current validation properties. As well as the validation components, filters also have the ability to issue validation errors. One example of a filter
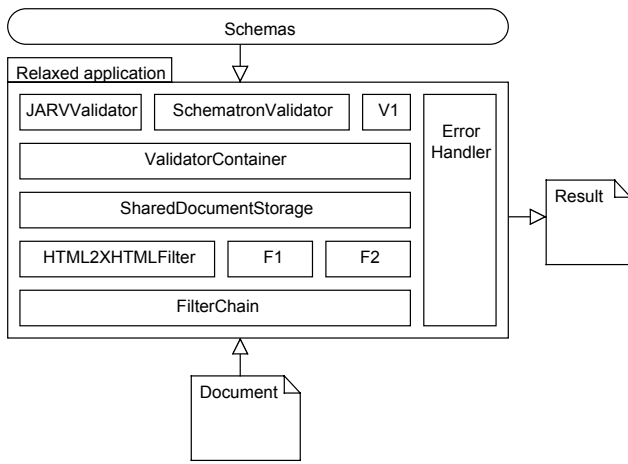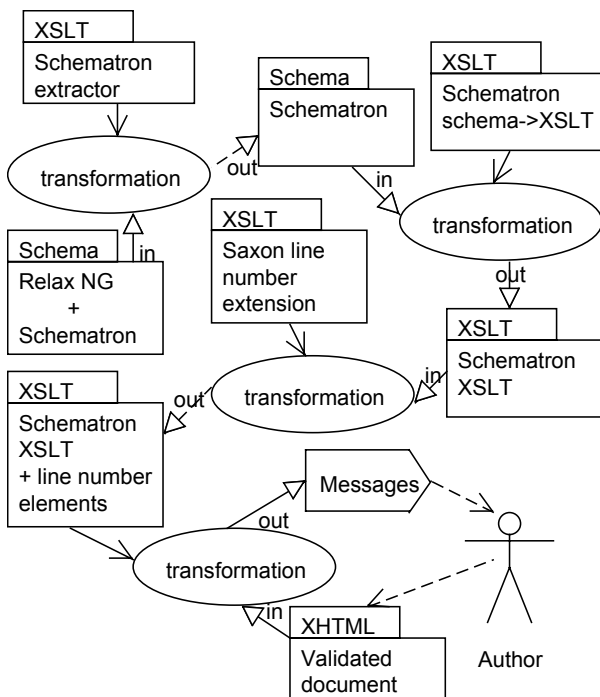
Figure 3: Application architecture

Figure 4: Schematron validation

implementation is the HTML to XHTML conversion filter.

4.2.2 Schema selection

As there are several types of HTML documents and every type needs a special schema to be validated, Relaxed validator needs some mechanism to map document types to relevant schemas. An eligible unique document type identifier is the public identifier used in the document type declaration. Occurrence of such declaration in HTML documents is required by the W3C HTML specification.

Relaxed validation components use a special configuration file to map document types to schemas. Such mapping may be further structured into different validation options which are basically different mapping subsets. It is possible to specify which option to use at each validation request. For instance a user may specify such an option, using a select box in a user interface. Grouping mappings into options allows Relaxed users to select schemas or a group of schema modules involved in the particular validation process and thereby for instance adjust the level of validation strictness.

4.2.3 Validation dispatching and validation result aggregation

When a validation request is retrieved, the validation container component performs a sequence of predefined steps. At first the specified document is retrieved, filtered by the filter chain and stored in a reusable form to be accessible among several validation components. After that the validator container triggers all registered validation components to start their individual validation tasks.

During this process, the validator container propagates a shared error handler instance into all validation components and even into all chained filters. The error handler collects all messages issued by different components. It may also perform additional operations e.g. message sorting or grouping on them.

A message contains an explanatory text, severity level and when it makes sense also a position of the described issue in the source document (usually specified by a line number and possibly by a column). The info severity level is used to inform the user about the validation process stages. Warnings are used in case some observed issue is not directly violating the specification, but instead its use in such a context is not recommended or questionable. The validated document is labelled as invalid in a case that the error handler retrieves any error or fatal error messages. The error severity level is used in case of a recoverable schema violation, while fatal error announces a serious problem, which makes further document processing impossible (e.g. document isn't well-formed).

4.2.4 Legacy HTML parser

As discussed in Section 4.2, for Relaxed it is very important to keep backward compatibility with HTML4.x documents. For that reason the TagSoup[11] library has been integrated into the Relaxed project. TagSoup is a SAX-compliant parser which allows standard XML tools to be applied to the real-life HTML documents. What's important, TagSoup architecture guarantees a well-formed output under all circumstances without any syntax error thrown. The TagSoup motto is "Just Keep On Trucking". This means that from any HTML4.x document Relaxed gets always a well-formed input.[12]

From the Relaxed point of view the TagSoupes philosophy is both, its biggest advantage and its biggest disadvantage. Relaxed requirement is to convert the SGML syntax into the more strict XML syntax but to stay notified about any SGML violations at the

---

[11] http://home.ccil.org/~cowan/XML/tagsoup/

[12] We also tested Java version of HTML Tidy (http://sourceforge.net/projects/jtidy), but this library was not able to correctly load and parse many real world documents.

same time. A TagSoup patch was necessary to achieve that.

TagSoup is basically a simple state-machine. When parsing a document, every important unit of the document transits TagSoup into a different state by invoking an action. A simple example of such behavior is an end tag without the start tag. This transits TagSoup into a special state where a repair action is invoked and the omitted tag is set right. Other violations e.g. missing end tag, unknown entities, attribute minimization, overlapping tags (see Example 10) etc. are repaired in a similar manner. The problem is that fixed SGML violations stay hidden to Relaxed and the user doesn't get notified about them. Overlapping tags are a fatal violation of XML as well as SGML. If TagSoup wouldn't fix such a problem the XML parser used in Relaxed would throw a fatal error, the validation process stops and the user gets finally notified. As you can see one solution is to modify TagSoup to fix just those errors which violate XML but not SGML.

```
before TagSoup:

<p> <i> Hello,</p> world! </i>

after TagSoup:

<p> <i> Hello,</i></p><i> world! </i>
```
Example 10: Overlapping elements

Another approach currently used in Relaxed is to let TagSoup fix all problems, but force him to report those, which are an SGML as well as an XML violation, with a specific error message. This approach is more comfortable for the user. As the validation process doesn't stop, the user gets most of the errors during a single validator run.

4.2.5 REST Interface

Representational State Transfer (REST) is a model for building Web-services based solely on the HTTP protocol and the URL specification. The client accesses a URL (a resource identifier) and gets a response (a resource representation) in some understandable form, for instance XML, HTML, PNG etc... REST is a light-weight alternative to SOAP, XML-RPC and other protocols which may mean an overhead in some simple situations. This is exactly the case of Relaxed as it features just one method with a small number of parameters.

The general URL for Relaxed REST Interface looks like this:

```
http://servername/context/method?parameter1=value&
       parameter2=value&...&parameterN=value.
```

The validation method is called `validate` and the context name depends on the particular server deployment. The `validate` method has a set of parameters. They basically correspond to the validation parameters introduced in Section 4.1.

VALIDATION PARAMETERS

uri the document to be validated (mandatory to start the validation process)

option name of the validation option, specified in the document type to schema mapping, see Section 4.2.2 (if not specified a default option is used)

source (true / false − default) specifies whether to include the full document's source within the validation output

severity (true / false − default) filters low severity level messages from the output

filterForcedDoctype (... / autodetect − default) forces an explicit document type, if not default

filterDirtyParser (true / false − default) if true, document is converted to well-formed XML before validation

xml (true / false − default) choose between HTML and XML output

The `xml` parameter determines the output format. If not specified the output is by default HTML which can be rendered by a Web browser. If set to true, the response is an XML output which is tailored for further automated processing by some other tools, applications or different services (see Example 11).

```
<relaxed>
  <source
   url="http://nalevka.com/resources/
 relaxed/poc.html"/>
  <output result="Your document is invalid.">
    <message severity="INFO">
      <text>Forced document type:
          -//W3C//DTD XHTML 1.0 Strict//EN</text>
    </message>
    <message severity="ERROR">
      <locator line="9" column="25" />
      <text>attribute "border" has a bad value:
          "10%" does not satisfy
          the "nonNegativeInteger" type</text>
      <source>...&lt;table border='10%'&gt;</source>
    </message>
    ... more messages ...
  </output>
</relaxed>
```
Example 11: Relaxed XML output

5. FUTURE WORK

There is no doubt that compound documents are the future of the Web. Compound documents combine elements from several vocabularies in one XML document. In the Web paradigm, compound document is usually a combination of XHTML with SVG, MathML, XForms, SMIL or VoiceXML fragments. In Section 3.2 we have shown how to manually create a schema which combines several vocabularies together and which can be used for validation of compound documents. We plan to create schemas for additional vocabulary combinations in the future. However, this approach does not scale well because there are simply too many possible vocabulary combinations.

In order to create even better support for compound documents in Relaxed we plan to implement our own NVDL based validator. NVDL (Namespace Validation and Dispatching Language) [1] is an upcoming ISO standard for validation of compound documents. In NVDL you can specify

how to split compound document into several fragments and how to validate these fragments against separate schemas.

```
<rules xmlns="http://purl.oclc.org/dsdl/
                  nvdl/ns/structure/1.0">
  <namespace ns="http://www.w3.org/1999/xhtml">
    <validate schema="xhtml.rng">
      <mode>
        <namespace ns="http://www.w3.org/1999/
                        02/22-rdf-syntax-ns#">
          <validate schema="rdfxml.rng">
            <mode>
              <anyNamespace>
                <attach/>
              </anyNamespace>
            </mode>
          </validate>
        </namespace>
      </mode>
    </validate>
  </namespace>
</rules>
```

Example 12: NVDL script for XHTML+RDF

We do not only plan to improve Relaxed internals, but we also plan to create usable tools for end users on top of Relaxed. One example of such tool is a validator browser plugin that lets you automatically validate a currently loaded page.

6. RELATED WORK

As far as we know Relaxed is the only validator that provides validation of compound documents.[13] There are validators like XML schema validator[14] and Validation Service for RELAX NG[15] which are based on W3C XML Schema or RELAX NG and are able to check data types better then DTD validators. However, none of those validators is supporting embedded Schematron rules and thus it is far beyond Relaxed validation capabilities. This is also supported by the fact that our RELAX NG schemas for XHTML are being used by another validation services, not only by Relaxed.

Currently Relaxed is also used for evaluating Web page accessibility in European Internet Accessibility Observatory[16] project. We cooperate with MedIEQ[17] project. The aim of this project is to develop tools for quality labelling of medical Web sites. Relaxed will be used there to check validity and some aspects of accessibility.

7. CONCLUSIONS

This article has shown that our approach of using RELAX NG combined with Schematron for Web documents validation is in all aspects superior to currently used DTD based validation. Implementation of our validation approach—Relaxed validator—is mature enough for production use. Significance of Relaxed will grow up with adoption of compound documents on the Web because currently there are no other validators able to validate compound documents.

Relaxed is an open-source project hosted on SourceForge[18]. Relaxed on-line validation service is available at `http://badame.vse.cz/validator/`.

== REFERENCES
8. REFERENCES

[1] *Document Schema Definition Languages (DSDL) — Part 4: Namespace-based Validation Dispatching Language — NVDL*. ISO/IEC FCD 19757-4. 2005.

[2] *XHTML^{TM} 1.0 The Extensible HyperText Markup Language (Second Edition)*. W3C, 2002.
WWW: `http://www.w3.org/TR/2002/REC-xhtml1-20020801/`

[3] Altheim, M., McCarron, S., Boumphrey, F., Dooley, S., Schnitzenbaumer, S., Wugofski, T.: *Modularization of XHTML^{TM}*. W3C, 2001.
WWW: `http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/`

[4] Biron, P., Malhotra, A.: *XML Schema Part 2: Datatypes Second Edition*. W3C, 2004.
WWW: `http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/`

[5] Chisholm, W., Vanderheiden, G., Jacobs, I.: *Web Content Accessibility Guidelines 1.0*. W3C WAI, 1999.
WWW: `http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/`

[6] Clark, J., Murata, M.: *RELAX NG Specification*. OASIS Committee Specification, 2001.
WWW: `http://www.relaxng.org/spec-20011203.html`

[7] Clark, J.: *Modularization of XHTML in RELAX NG*. Thai Open Source Software Center Ltd, 2003.
WWW: `http://www.thaiopensource.com/relaxng/xhtml/`

[8] Jelliffe, R.: *The Schematron Assertion Language 1.5*. Academia Sinica Computing Centre, 2002.
WWW: `http://xml.ascc.net/resource/schematron/Schematron2000.html`

[9] Murata, M., Dongwon, L., Murali, M., Kawaguchi, K.: *Taxonomy of XML Schema Languages using Formal Language Theory*. 2004.
WWW: `http://web.cs.wpi.edu/~mmani/toit/taxonomy/new/taxonomy.pdf`

[10] Ragget, D., Le Hors, A., Jacobs, I.: *HTML 4.01 Specification*. W3C, 1999.
WWW: `http://www.w3.org/TR/1999/REC-html401-19991224/`

[11] Thompson, H.S., Beech, D., Maloney, M., Mendelsohn, N.: *XML Schema Part 1: Structures Second Edition*. W3C, 2004.
WWW: `http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/`

---

[13]Strictly speaking W3C validator can validate XHTML+MathML. But as this validator is DTD based you are forced to use specific namespace prefixes in your document.

[14]`http://schneegans.de/sv/`

[15]`http://hsivonen.iki.fi/validator/`

[16]`http://www.eiao.net/`

[17]`http://zeus.iit.demokritos.gr/medieq`

---

[18]`http://relaxed.sourceforge.net`