Proceedings of Extreme Markup Languages®

Schema Languages & Internationalization Issues: A survey

Felix Sasaki
fsasaki@w3.org
Christian Lieske
christian.lieske@sap.com
Andreas Witt
andreas.witt@uni-bielefeld.de

Abstract

Many XML-related activities (e.g. the creation of a new schema) already address issues with different languages, scripts, and cultures. Nevertheless, a need exists for additional mechanisms and guidelines for more effective internationalization (i18n) and localization (l10n) in XML-related contents and processes. The W3C Internationalization Tag Set Working Group (W3C ITS WG) addresses this need and works on data categories, representation mechanisms and guidelines related to i18n and l10n support in the XML realm. This paper describes initial findings from the (W3C ITS WG). Furthermore, the paper discusses how these findings relate to specific schema languages, and complementary technologies like namespace sectioning, schema annotation and the description of processing chains. The paper exemplifies why certain requirements only can be met by a combination of technologies, and discusses these technologies.

Keywords: Schema Languages; Namespaces; Processing

Table of Contents

Felix Sasaki

Until 1999, Felix Sasaki has studied Japanese and Linguistics in Berlin, Germany. From 1999 until 2005 he worked in the Department for Computational Linguistics and 'Text Technology' in Bielefeld, Germany. As of 1 April 2005, he joined the W3C Internationalization Activity.

Christian Lieske

Christian Lieske is a member of SAP's group for MultiLingual Technology. He works on putting Natural Language Processing (NLP) and Content Engineering to use. Projects he works on include tools for quality assurance for terminology, terminology extraction and checking, style checking, and XML-related projects for various types of content. He is actively involved in standards activities driven by OASIS and the W3C, and enjoys internal consulting related to NLP, XML, and general authoring and localization issues. Before joining SAP, Christian worked for several NLP Research Institutes. He holds a degree in Computer Science with special focus on Natural Language Processing and Artificial Intelligence.

Andreas Witt

Since 1996, Andreas Witt has taught at Bielefeld University, Germany in the field of 'text technology'. His research interests include the combination of computational linguistics and markup technologies, schema languages, and corpus annotation.

Schema Languages & Internationalization Issues: A survey

Felix Sasaki [World Wide Web Consortium]
Christian Lieske [SAP AG]
Andreas Witt [Bielefeld University]

Extreme Markup Languages 2005® (Montréal, Québec)

Introduction

This paper addresses two topics. On the one hand, it introduces the work of the ITS WG [Internationalization Tag Set Working Group] of the World Wide Web Consortium[1]. On the other hand, the paper discusses various touch points between the ITS WG, schema languages like XML DTDs **[Bray et al. 2004]**, RELAX NG **[Clark and Murata 2001]** and XML Schema **[Thompson et al. 2001]**, and other XML related technologies.

Despite of its name, the ITS WG is not necessarily aiming at a "tag set". Rather, the focus is to gather requirements which are important for internationalization and localization. One important question with regard to these requirements is the following: How might they be considered in XML instances or their underlying schemas?

The touch points between the ITS requirements and schema languages are:

- ideally, the ITS data categories [2] can be used with existing XML instances or their underlying schemas; thus, the ITS WG should recommend an approach that is flexible enough to be used in combination with or as an extension of various schemas, which are possibly defined within different schema languages.

- some ITS data categories need specific mechanisms for their realization which are not present in some schema languages; thus, the ITS WG should recommend an approach that covers this requirement. The work of the ITS WG may relate to schemas also in the following ways: The ITS WG may create a separate, single schema, a schema module which is to be used with existing or new schemas, or it can be schema independent, i.e. a specification of information in instance documents.

The paper discusses various approaches for the realization of requirements gathered by the ITS WG. It sketches some findings of the W3C ITS WG related to capabilities / constraints of selected standards and technologies. In particular, the suitability of schema languages for ITS requirements is addressed. Furthermore ideas about additional or alternative mechanisms for realizing i18n/l10n requirements are summarized.

The paper is structured as follows. Section “Taking care of internationalization and localization” gives a general overview of the work of the ITS WG (section “General concerns of the W3C ITS Working Group”) and some specific requirements (section “Specific requirements of ITS”). Section “The need to go beyond schemas” explains the various touch points between the ITS WG and schema languages, and the need to go beyond schema languages. Section “Approaches for the realization of ITS” provides an overview of the capabilities of schema languages and related technologies from the viewpoint of ITS. Section “Summary and general discussion” summarizes the discussion.

Taking care of internationalization and localization

General concerns of the W3C ITS Working Group

Internationalization and localization related to XML

Content or software which is authored in one language (so-called original language) often has to be made available in additional languages. Translation and other activities related to this making available thus are an important factor in production processes related to content or software.

I18n [internationalization] is the process of generalizing a product so that it can handle multiple languages and cultural conventions without the need for redesign. Internationalization takes place at the level of program design and document development.

L10n [localization] is the process of taking a product and making it linguistically and culturally appropriate to a given target locale (country / region and language) where it will be used.

With XML entering the stage in both the content (e.g. the XML version of DocBook **[Walsh and Muellner 1999]**, a format related to software manuals) and the software world (e.g. XUL [eXtensible User Interface Language], cf. http://www.xulplanet.com/), new challenges and chances related to efficient and high-quality translation / internationalization arise, cf. **[Savourel (2001)]**. Some sample challenges are:

(1) For each XML vocabulary, possibly all people, processes and tools have to be made aware of specifics such as which attributes or elements contain translatable content, see figure 1.

Figure 1: Sample code with translatable text

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Sample XUL file -->
<window xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
 <box align="center">
  <button label="hello xFly" onclick="alert('Hello World');"/>
 </box>
```

```
</window>
```

(2) Translation may break certain types of coding. If for example an application requires titles of chapters to be unique, but translation does not adhere to this constraint, the application may fail to work.

Some sample chances are:

(1) XML-based material can be much more easily twisted than binary content. It is for example easy to extract the value of certain attributes by means of a very basic XSL stylesheet.

(2) Information which is helpful for translation / localization purposes often can be embedded or attached to XML-based material. This very often holds for both the specification of an XML vocabulary (e.g. in XML Schema) and for XML documents. Figure 2 gives an example.

Figure 2: Sample XML Schema with i18n / l10n information

```
<xsd:schema xmlns="http://example.com/myBook"
            xmlns:slim="http://www.example.com/slim"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified"
            attributeFormDefault="unqualified"
            targetNamespace="http://example.com/myBook">
 <xsd:element name="book">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element ref="chapter" maxOccurs="unbounded"/>
   </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="titleIdAndContent" slim:noteToLocEng="Please check
   that this constraint can be enforced in our target languages">
   <xsd:selector xpath="chapter"/>
   <xsd:field xpath="@title"/>
  </xsd:unique>
 </xsd:element>
 <xsd:element name="chapter">
  <xsd:complexType mixed="true">
   <xsd:attribute name="title" use="required"/>
   <xsd:attribute name="id" use="optional" type="xsd:ID"/>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

The figure contains a sample XML Schema with i18n / l10n Information. It provides a uniqueness constraint `slim:noteToLocEng` (see section “Features of schema languages and their relation to ITS”) for a localization engineer.

The need for additional i18n and l10n support in XML contexts

As indicated in the remarks above, a standard set of information / data categories that can be used with XML material (both schemas and XML instances) to support the internationalization and localization of documents would be helpful.

What kind of i180n- and l10n-related information is needed in a certain XML usage scenario, and how it can be captured depends on the scenario. Some vocabularies and processing pipelines

- center on prose
- focus on code (XUL, cf. figure 1)
- mix prose and code (e.g. DocBook)
- include presentational aspects (e.g. XHTML).

Examples are given in figure 3.

Figure 3: Example XML documents

```
DOCBOOK EXAMPLE:
<book lang="en-US">
 <title>Introduction to Python</title>
 <chapter>
  <title>Basics</title>
   <para>This is a simple example of the Python interpreter.</para>
   <example>
    <title>The Python interpreter</title>
    <programlisting format="linespecific">
     <![CDATA[>>> s = 'Hello, world.'
     >>> str(s)
     'Hello, world.'
     ]]>
    </programlisting>
   </example>
   <para>If you create a typing error like "strs(s)", you
    will get the message "NameError: name 'strs' is not defined".</para>
 </chapter>
</book>

TEI EXAMPLE:
<TEI.2>
 <teiHeader>
  <fileDesc>
   <titleStmt>
    <title>Japanese: An Introduction</title>
   </titleStmt>
   <publicationStmt>
    <date>5 May 2005</date>
   </publicationStmt>
   <sourceDesc><p>Initial Draft</p></sourceDesc>
  </fileDesc>
 </teiHeader>
 <text>
  <body>
   <div1>
    <head><title>Honorific Expressions</title></head>
    <p>The <term  id="honor-expr">Plain Style(普通語)</term> versus
     the <term id="polite-expr">Polite Style(丁寧語)</term> ...</p>
   </div1>
  </body>
 </text>
</TEI.2>

XHTML 1.0 EXAMPLE:
<html xmlns="http://www.w3.org/1999/xhtml"
      xml:lang="en-US" lang="en-US">
 <head>
  <title>World News - 2 May 2005</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <style type="text/css">
   h1 {color: blue; text-align:center}
  </style>
 </head>
 <body>
  <h1>World News - 2 May 2005</h1>
  <h2>Germany: The first Female Chancelor</h2>
  <p>...</p>
 </body>
</html>
```

DocBook is an example of a vocabulary that mixes prose and code. The code (see the content of the `programlisting` element) must not be translated in certain processing / production contexts (or only parts of it like the `Hello World` phrase may have to be translated). In the example from the TEI [Text Encoding Initiative] ([Sperberg-McQueen et al. 1994]), textual data is in the centre. Here the `term` element [3] might rely on an external data base for linguistic terminology.

XHTML 1.0 **[Pemperton et al. (2000)]** allows for the inclusion of presentational information. Accordingly, the style element might need adaptation, when the content is localized. An example for XML which contains only software code is XUL, cf. figure 1.

The examples show that each XML context (e.g. the underlying schema or the processing for a certain bit of content) raises specific issues related to i18n / l10n. Although i18n related information which is helpful for localization purposes often can be embedded or attached to XML-based material (a noteworthy example is the identification of a language via the `xml:lang` attribute), a need exists for a standard related to i18n and l10n facilitating information in XML. In lack of a standard tool support for i18n and l10n it might not be possible to realize these processes efficiently (since e.g. each XML vocabulary may choose its own way to represent information which a localizer should know about). The W3C ITS WG accordingly addresses this need and works on data categories, representation mechanisms and guidelines related to i18n and l10n support in XML contexts.

Specific requirements of ITS

Based on previous work from WG members **[Ishida and Savourel 2003]** the Working Group has progressed rapidly towards a preliminary list of detailed requirements. The requirements will be explained with reference to the examples in figure 3 and figure 1.

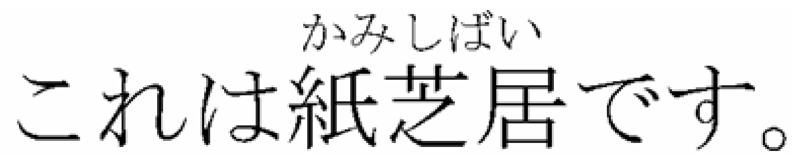
1. Direct or indirect identification of content that should not be localized. It must be possible to specify whether some content should be localized or not. For example the value of the `id` attributes at the `term` element in the TEI document, e.g. `honor-expr`, seems to be language specific. Nevertheless ID values must not be changed, because many processing steps might rely on them.
2. Selection of information with i18n and l10n related characteristics. The l10n processes encompasses not only translation, but also changing of styling or i18n specific data types like currencies or time related data, specification of the language of content **[Ishida (2005)]** and much more. The TEI document contains a `date` element those content might be relevant for such processes. L10n and i18n relevant information can be identified in various parts of a document or a schema, e.g. as the textual content of an element or the value of an attribute. In the DocBook example, program code is contained in the `programlisting` element. This might need specific treatment during the localization process, e.g. if one wants to translate the substring `Hello, world.`. The same is true for the substring `Hello, world` which is part of the value of the `onclick` attribute at the `button` element in the XUL example.
3. Specification of inheritance features and exceptions. This requirement is closely related to the ones mentioned above. E.g. for all elements in the XUL example, non translatability should be specified. An exception is the `onclick` attribute.
4. Requiring uniqueness of IDs across documents. For l10n related processes like terminology management, the unique identification of a term across a set of documents is necessary.
5. Provision of a SPAN-like element to mark up special properties for localization and internationalization. Such an element might be necessary if the XML based vocabulary does not provide a appropriate element. If possible, existing elements like the `span` element in XHTML 1.0 could be used for this purpose.
6. Provision of notes to localizers. For example, the content of the `p` element inside the `sourceDesc` element in the TEI example might need a note like `Please check with the author how to translate "inital draft".`.
7. Linking mechanisms for external information. This requirement has a specific status and is explained in detail below.
8. Description of emphasis & document conventions, handling of citations, specific rendering information. These requirements could be fulfilled by the `span` like element described above.
9. Markup to support international script features. For languages like Arabic or Hebrew, the directionality of writing has to be specified. This can depend on the data type of the string. For example in the XHTML 1.0 example. the `h1` element contains the string `World News - 2 May 2005`. The date needs a different handling of directionality than the rest of the string, see **[Ishida (2003)]**. For East Asian languages, complex markup like Ruby **[Suignard et al. (2001)]** might be necessary. Ruby

provides additional information about a string, which is rendered above or below the string (if the writing direction is from top to bottom, the position of ruby is left or right, respectively). The following example contains a string of Japanese characters in the `rb` element, with pronunciation information in the `rt` element:

```
<p>これは<ruby><rb>紙芝居</rb><rt>かみしばい</rt></ruby>です。</p>
```

A rendering of the example is visualized in 4.

Figure 4: Rendering of a ruby annotation



[Link to open this graphic in a separate page]

The list of requirements indicates that additional support for i18n and l10n in XML contexts may need more than a specific markup vocabulary (say an Internationalization Tag Set or Standard Localization Information Markup), since some requirements cannot easily be addressed by means of such a special purpose vocabulary. References to external information with corresponding processing expectations (e.g. fetching the translation for a term element from a term data base on the Web) go for example beyond the power of schema languages. Hence, one particular concern of the W3C ITS WG is to determine the possible interaction between additional support for i18n and l10n and the existing or forthcoming XML- related standards and technologies. The rest of the paper discusses some findings of the W3C ITS WG related to capabilities/constraints of selected standards and technologies. In particular, the power of schema languages is addressed. Furthermore ideas about additional or alternative mechanisms for realizing i18n/l10n requirements are summarized.

Requirements related to processing play an important role. Especially in the case of user interface documentation, links to external information (requirement 7), e.g. to predefined strings like error messages must be specified. An example is the string / the message `NameError: name 'strs' is not defined"`. In a localized user interface, the message might be different, hence the documentation has to reflect this difference. This requirement could be accomplished in the following way:

```
<para>If you create a typing error like "strs(s)",
 you will get the message
 <xref id="resfile.resx">
  <subst>
   <search>{0}</search>
   <replace>&lt;Filename&gt;</replace>
  </subst>
 </xref>.
<para>
```

A process could be described to retrieve the error message which is appropriate for the local respectively.

The need to go beyond schemas

Formalizing ITS data categories and requirements in general by means of a schema language is a natural choice. In this way, users of ITS can easily benefit from all aspects of schema-related processing (e.g. validation and transformation).

However, matching the list of requirements for the ITS in section “Specific requirements of ITS” against the capabilities of schema languages reveals that the requirements will not be satisfied if ITS is nothing but a markup vocabulary. Although some requirements can easily be encoded with a schema language

(Requirement 1 e.g. can be realized with a simple attribute `localize` with two alternative values yes or no), others cannot be captured easily by means of a schema language. This for example holds for requirements which are meta-information to markup or which represent processing expectations.

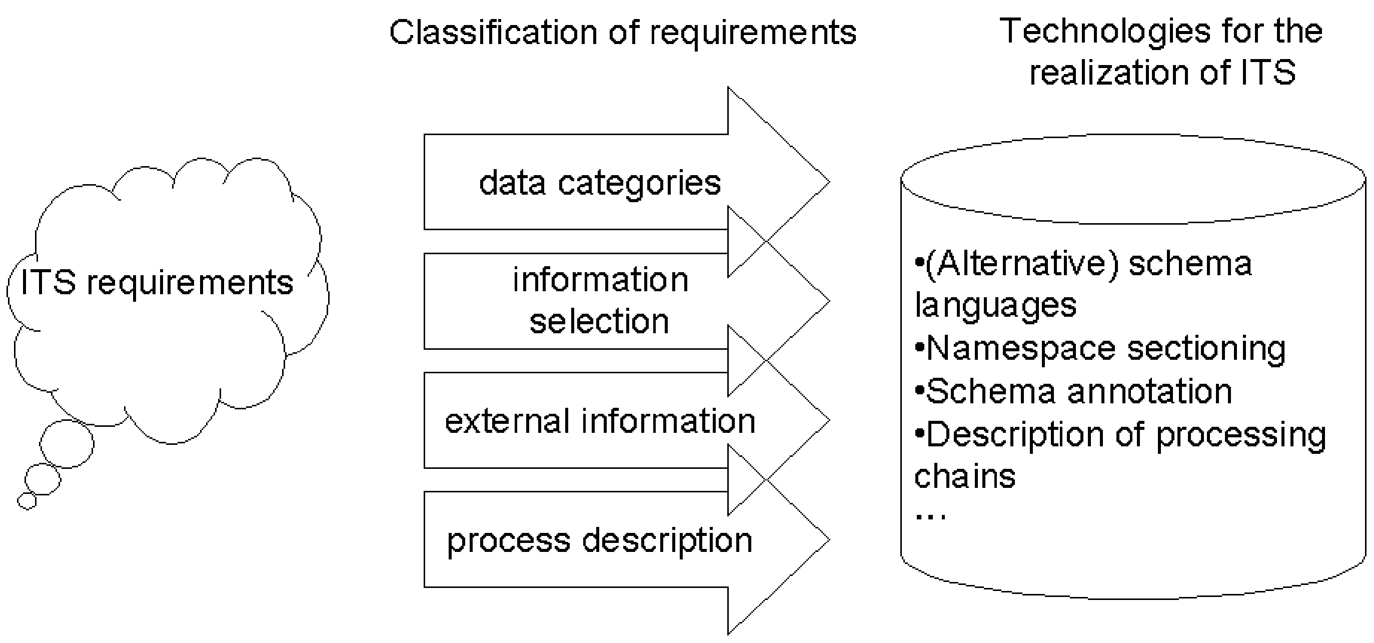
An additional indicator that the W3C ITS WG may have to go beyond the formalization of ITS data categories by means of a schema is the fact that it might not be possible to integrate ITS markup into the source markup (i.e. the markup, e.g. XHTML, which has to be equipped with i18n / l10n information). Other processing steps, e.g. rendering of XHTML documents, might be corrupted by the ITS markup.

A more detailed discussion of the issues of ITS requirements which cannot be resolved with schema languages looks as follows:

a. Requiring uniqueness of IDs across documents: Uniqueness of elements can be specified in XML only within a single document, not across documents.
b. Information about attributes, e.g. about their translatability: There is no mechanism to give additional information about attributes.
c. Linking to external information: This could be realized e.g. via XInclude **[Marsh and Orchard (2004)]**, but not with a schema language.
d. Incompatibility with the underlying schema: Consider for example that the schema for the TEI example in figure 3 defines the value of the `date` element as textual data. ITS then might impose a different requirement on this value, e.g. that it has the data type `date`.
e. Impact on processing chains: XPath expressions which rely on an element with only textual content might break if there are additional child elements. For example, if ruby markup is used for the Japanese characters 普通語 and 丁寧語 in the TEI example, XPath expressions which require the `term` elements to contain only characters might break.
f. Conflict between existing markup and ITS related markup: Take for example the `span` element described as requirement 5 in section “Specific requirements of ITS”. There might be a similar element in the existing schema already, and it is unclear whether to use that element or the one defined for ITS.
g. Versioning: The last problem has a strong relation to versioning. E.g. in the current Working Draft of XHTML 2.0 **[Axelsson et al. (2005)]** there are elements for international script features. Hence, an application of ITS as a part of a previous version of XHTML might rely on ITS specific markup for these features, whereas in a new version the ITS specific markup might use the XHTML 2.0 version. These differences impose problems for the stability of processing chains.
h. Specification of inheritance features: Some inheritance features which are described as a requirement 3 in section “Specific requirements of ITS” are foreseen in the XML specification itself. Important for i18n issues is the attribute `xml:lang`. Nevertheless, there are other ITS specific features which need inheritance, e.g. the specification of translatability. For these features, schema languages don't provide an inheritance processing mechanism.

It is now obvious that ITS requirements need more than schema languages to be realized. The following section discusses what the various schema languages offer (section “Realization with ITS based on schema languages”, and other or complementary mechanisms (section “Namespace sectioning” - section “ITS as a description of processing chains”). Figure 5 visualizes the various mechanisms for the realization mechanisms for ITS.

Figure 5: Overview of approaches for the realization of ITS


Classification of requirements
Technologies for the realization of ITS
ITS requirements
data categories
information selection
external information
process description
•(Alternative) schema languages
•Namespace sectioning
•Schema annotation
•Description of processing chains
...

[Link to open this graphic in a separate page]

Approaches for the realization of ITS

Realization with ITS based on schema languages

Existing schema languages

A few years ago, a lively discussion about various schema languages took place, and the variety of schema languages was immense. One reason for this discussion was unclearness about the formal properties of schema languages. The clearness needed was provided by **[Murata et al. 2001]**, who described schema languages in terms of formal language theory, relying on the framework of *regular tree grammars*. Today, mainly three alternative schema languages remain, i.e.

- XML-DTDs, a local tree grammar. It allows only one definition of an element with the same name and the same content model, i.e. it is forbidden to declare a `p` element several times.
- XML Schema, a single type tree grammar. It allows for the declaration of elements with the same name and different content models in different contexts, e.g. (1) a `p` element inside a `footnote` element; the `p` element has only textual content (2) a `p` element inside a `div` element; the `p` element has a content model with child elements. The UPA [unique particle attribution] constraint[4] disallows the definition of an element with the same name several times in the same context. E.g. XML schema would not allow (1) and (2) to be part of the same content model, e.g. of the same `div` element.
- RELAX NG, which can be described as a regular tree grammar. It has no constrains on the structure of content models. Elements with the same name but different content models are allowed in the same context, e.g. the variants of a `p` element mentioned above.

Beside these grammar-based schema languages, the so-called pattern-based schema language exist **[Lee and Chu]**. For the purposes of ITS, the pattern-based schema language Schematron (cf. http://www.schematron.com/), is highly valuable as well. Nonetheless, at the moment the ITS activities mainly deal with the grammar-based schema languages.

Figure 6 provides schemas for the documents in figure 3. As for the DocBook document , it is assumed that the schema is given as an XML DTD. For the TEI document, an RELAX NG Schema has been created. It is written here with the compact syntax of RELAX NG. The XHTML 1.0 example relies on XML Schema.

Figure 6: Schemas for the documents in figure 3

DTD for the DocBook document:
```
<!ELEMENT book (title,chapter)>
<!ATTLIST book
 lang NMTOKEN #REQUIRED>
<!ELEMENT chapter (title,(example|para)+)>
<!ELEMENT example (title,programlisting)>
<!ELEMENT para (#PCDATA)>
<!ELEMENT programlisting (#PCDATA)>
<!ATTLIST programlisting
 format CDATA #FIXED 'linespecific'>
<!ELEMENT title (#PCDATA)>
```

RELAX NG Schema in compact syntax for the TEI document:
```
start =
  element TEI.2 {
    element teiHeader {
      element fileDesc {
        element titleStmt { title },
        element publicationStmt {
          element date { text }
        },
        element sourceDesc { p }
      }
    },
    element text {
      element body { div1+ }
    }
  }
div1 =
  element div1 {
    element head { title },
    p
  }
title = element title { text }
p =
  element p {
    (text
     | element term {
         attribute id { xsd:NCName },
         text
       })+
  }
```

XML Schema for the XHTML 1.0 example document:
```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="http://www.w3.org/1999/xhtml"
           targetNamespace="http://www.w3.org/1999/xhtml"
           elementFormDefault="qualified">
 <xs:element name="html">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="head"/>
    <xs:element ref="body"/>
   </xs:sequence>
   <xs:attribute name="lang" type="xs:NMTOKEN" use="required"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="head">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="title" type="xs:string"/>
    <xs:element name="meta">
     <xs:complexType>
      <xs:attribute name="http-equiv" type="xs:NMTOKEN" use="optional"/>
      <xs:attribute name="content" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="style">
     <xs:complexType mixed="true">
      <xs:attribute name="type" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="body">
  <xs:complexType mixed="true">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
     <xs:element name="h1" type="xs:string"/>
     <xs:element name="h2" type="xs:string"/>
     <xs:element name="p" type="xs:string"/>
    </xs:choice>
   </xs:complexType>
 </xs:element>
</xs:schema>
```

It is assumed that the reader is familiar with the syntax of XML DTDs. In the RELAX NG schema, the root element of the document is defined via the keyword `start`, e.g. `start = element TEI.2`. The `element` keyword is used for element declarations. Content models are given in curly brackets, e.g. `element TEI.2 { ...}`. Attributes are declared with the keyword `attribute`, e.g. `attribute id { ... }`. As for the content of elements and attributes, the keyword `text` signals textual data. Other data types are possible as well, e.g. the data type `xsd:NCName`. RELAX NG allows for the definition of named patterns, which can be referenced within other declarations. An example is the pattern `div1`, which is referenced in the content model of the `body` element.

XML Schema is defined in an XML based Syntax. The example schema contains element declarations, e.g. `<xs:element name="html"> ...`, attribute declarations, e.g. `<xs:attribute name="lang" ... > ...`, and type definitions, e.g. with the `xs:complexType` element. Types will be discussed later in section “Features of schema languages and their relation to ITS”. Each element or attribute needs to have a non-ambiguous type. Types can have names, e.g. `xs:NMTOKEN` for the attribute `lang`, or they can be defined without a name. The elements in the schema are defined with complex types. If the elements should contain textual data and child elements or attributes, the complex type definition has an attribute `mixed="true"`. Global declarations of elements and attributes, i.e. declarations under the `xs:schema` element, can be referenced. In the example, the content model of the `html` element makes references to the elements `head` and `body`.

Features of schema languages and their relation to ITS

For the realization of ITS, the features of schema languages which are summarized in the following table are of importance. The table also describes the position of the schema languages which respect to the framework of regular tree grammars mentioned above.

Table 1: Features of Schema languages for the purposes of ITS

Feature relevant for ITS	Schematron	DTD	XML Schema	RELAX NG
Pattern based descriptions	+	-	-	-
XML Syntax	+	-	+	+
Non-XML Syntax	-	+	-	+
Namespaces	+	(+)	+	+
Datatyping for attribute values	-	+	+	+
Datatyping for simple content in elements	-	-	+	+
Datatyping for complex content in elements	-	-	+	-
Subtyping mechanism	-	-	+	-
XML-based grouping mechanism	-	-	+	+
Local tree grammar constraints	-	+	+	+

Feature relevant for ITS	Schematron	DTD	XML Schema	RELAX NG
Single-type tree grammar constraints	-	-	+	+
Regular tree grammar constraints	-	-	-	+
Uniqueness constraints, KEY / KEYREF constraints	-	-	+	-

Pattern based descriptions are only possible with Schematron, which relies on XPath. Using XPath 2.0 **[Berglund et al. (2005)]** which allows for selection of information across documents, the requirement of uniqueness from various documents (see requirement 1 in section "The need to go beyond schemas") can be fulfilled. The XML Syntax which is provided by all schema languages except XML DTDs[5] is an important means for the processing of ITS data categories in schemas in general. If for example ITS data categories / requirements are incompatible with an existing schema, an XSLT based filter can be used to create two versions of the schema: One with ITS specific additions, the other as the basic schema. A similar filter can be applied to documents. In this way, existing processing chains (requirement 5 in section "The need to go beyond schemas") like validation can be maintained.

Namespaces are provided by all schema languages, with a limited support in XML DTDs[6]. To avoid conflicts between existing schemas and ITS related markup (requirement 6 in section "The need to go beyond schemas"), they are an important means. Nevertheless they do not solve all problems: If a version of ITS relies on markup from one scheme, e.g. XHTML 1.0, that markup might not be available in a different version of the scheme (requirement 7 in section "The need to go beyond schemas").

The data typing facilities provided by the various schema languages lead to many differences between them. Data typing for attribute values is provided by all schema languages, except Schematron[7]. Only RELAX NG and XML Schema allow for typing for simple content in elements. In this way it is possible to state that the content of the `date` element in the TEI document in Figure 3 has to be of the type `date`. This makes it possible to provide date and time related information for the localization process. Datatyping for complex content, i.e. for elements with attributes and / or child elements, is only possible with XML Schema. Together with the subtyping mechanism of XML Schema, this is a powerful mean to avoid incompatibility with existing schemas (requirement 4 in section "The need to go beyond schemas"). The following example, which is based on the XHTML example in Figure 3, demonstrates how subtyping can be used to add ITS specific information to a schema:

```
<xs:complexType name="paraContent" mixed="true">
 <xs:attribute name="id" type="xs:ID" use="optional"/>
</xs:complexType>
<xs:complexType name="itsParaContent">
 <xs:complexContent>
  <xs:extension base="paraContent">
   <xs:attribute name="locinfo" type="xs:string"/>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
...
<xs:element name="p" type="itsParaContent"/>
```

The - here simplified - definition of the complex type `paraContent` provides a content model with an `id` attribute and textual content in the element. For the purposes of ITS, a type `itsParaContent` is derived from `paraContent` which provides an additional attribute `locinfo`. The `p` element is then defined in terms of this derived type.

A mechanism which is different, but yet in some respect similar to these typing mechanisms, are grouping and modularization facilities. These are provided by all schema languages except Schematron. Nevertheless, the details of these facilities differ between each schema language. Some examples are introduced here for XML DTDs, and RELAX NG, and then compared to data typing.

With the combined application of parameter entities and marked sections in XML DTDs, definitions like the following are possible:

```
<!ENTITY % para "INCLUDE">
<![%para;[
   <!ELEMENT p (#PCDATA)>
   <!ATTLIST p id ID #IMPLIED>
   <!ENTITY % i18n.para "IGNORE">
]]>
<![%i18n.para;[
   <!ELEMENT p (#PCDATA)>
   <!ATTLIST p
     id ID #IMPLIED
     locinfo CDATA #IMPLIED>
                                                ]]>
```

There are two alternative definitions of the `p` element. `%para` is a basic definition of a paragraph, `%i18n.para` contains an i18n related extension. Since `%i18n.para` is set as `IGNORED` as part of the `%para` definition, a common application of this DTD does not encompass the i18n related extension. Depending on which parameter entity is selected in the declaration subset in an XML document, the validation can make use of the extensions or not. XML Schema and RELAX NG do not provide such grouping mechanisms which depend on the processing of the XML document. On the other hand, they provide XML based grouping mechanisms which can be manipulated e.g. by an XSLT stylesheet. The following examples demonstrate the extension of the content model of the `p` element in RELAX NG:

```
p = p-common | p-i18n
p-common =
  element p {
    attribute id { xsd:ID },
    text
  }
p-i18n =
  p-common,
  attribute locinfo { text }
element div = p+
```

XML Schema provides a similar grouping mechanism. The difference is that in the case of XML Schema, the groups must not lead to grammar patterns which exceed the power of a single type tree grammar. For example, the definition `p = p-common | p-i18n` from the RELAX NG example leads to two `p` elements with different content models. These are then used in the same context, i.e. in the content model of the `div` element. As mentioned before, this is not possible in XML Schema.

The following table summarizes the differences between marked sections in XML DTDs, type derivations in XML Schema and grouping with named patterns in RELAX NG.

Table 2: Schema language specific grouping and typing mechanisms

XML DTD marked sections	XML Schema sub typing	RELAX NG named patterns
Defined in schema or declaration subset, evoked in declaration subset of the XML document	Defined in schema	Defined in schema
Allows for alternative, unrelated declarations	Allows for elements with the same name and different content models, but with restrictions on derived types	Allows for elements with the same name and different content models, without restrictions on ambiguity
Information about the modularization is not given in the XML document	The XML document is enhanced with typing information	Information about the modularization is not given in the XML document

The key difference between the grouping or modularization mechanisms and subtyping is that grouping provides only information in the schema. As for the validation of XML documents against the schema, this information is just "syntactic sugar". It is not accessible for or after the validation process. This is the crucial difference to the typing approach of XML Schema: After validation, typing information is available in the PSVI [Post Schema Validation Infoset] [8]. For example, the PSVI provides information about the type of a `p` element, e.g. whether it is of the complex type `itsParaContent`. This information can be used for further processing. A query in XQuery [Boag et al 2003] which retrieves all elements with the type `itsParaContent` would be

```
for $itsPara in doc("mydoc.xml")//element(*, itsParaContent?) return $itsPara
```

The question yet still to be answered is whether such information is useful for ITS related processing. As for the grouping mechanisms of XML DTDs, RELAX NG and XML Schema, they could be used to create hard-wired schemas which encompass for example XHTML and ITS, or XHTML, ITS and SVG. Nevertheless, it would be difficult to assure that such hard-wired combinations are used in widespread i18n and l10n sensitive applications.

Uniqueness constraints, the last property of schema languages to be discussed in this section are uniqueness constraints. They are only available in XML Schema. They allow to specify e.g. that during the localization of an XUL document, the values of the `label` and `onclick` attributes always form a unique combination:

```
<xsd:unique name="uniqueButton">
 <xsd:selector xpath="button"/>
 <xsd:field xpath="@label"/>
 <xsd:field xpath="@onclick"/>
</xsd:unique>
```

However, because XML Schema uses only a subset of XPath, XML Schema is not sufficient for many constellations of values in documents. An alternative approach is to use pattern based descriptions via Schematron:

```
<schema xmlns="http://www.ascc.net/xml/schematron" >
 <pattern name="uniqueButton">
  <rule context="window">
   <assert test="
exists(
 if (count(distinct-values(
  for $i in .//button
  return (concat($i/@label, concat(' ',$i/@onclick)))) &lt; count (for $i in $input//a
  return $i))
 then 'error found' else ''
)"
   >The values of the label attribute and
   the onclick attribute are not in a unique combination.</assert>
  </rule>
 </pattern>
</schema>
```

This Schematron document tests for each `window` element if the attributes `onclick` and `label` on the nested `button` elements have unique values.

Namespace sectioning

Schema languages allow only for hard-wired combinations of markup vocabularies. An approach to integrate ITS into markup vocabularies without such a hard-wired combination is discussed in this section. The approach relies on the separation of documents in namespace specific sections before the validation phase.

The NRL [Namespace Routing Language] [Clark (2003)], has been designed to fulfill this purpose. It is the main input to part 4 NVDL [Namespace-based Validation Dispatching Language] of the ISO / IEC 19757 proposal DSDL [Document Schema Definition Languages] [9]. An example of the application of NRL is given in Figure 7.

Figure 7: Example for an NRL document

```
XHTML document:
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:its="http://www.example.org/its"
      xml:lang="en" lang="en"
      its:translate="yes">
 <head>
  <title its:translate="no">World News - 2 May 2005</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
 </head>
 <body>
  <h1 its:translate="no">World News - 2 May 2005</h1> ...
 </body>
</html>

RELAX NG Schema for ITS its.rnc:
namespace its = "http://www.example.org/its"
attribute its:translate { "yes" | "no" }?

NRL document:
<rules xmlns="http://www.thaiopensource.com/validate/nrl">
 <namespace ns="http://www.w3.org/1999/xhtml" match="elements attributes">
  <validate schema="xhtml.rng"/>
 </namespace>
 <namespace ns="http://www.example.org/its" match="attributes">
  <validate schema="its.rnc" schemaType="application/x-rnc"/>
 </namespace>
</rules>
```

The NRL document is used for the validation of an XHTML document with attributes `translate` from an fictive ITS namespace http://www.example.org/its. This attribute is defined in the schema `its.rnc`. In the NRL document, the first `namespace` element with the attributes `ns="http://www.w3.org/1999/xhtml"` and `match="elements attributes"` states that all elements and attributes from the XHTML namespace should be validated with the schema `xhtml.rng`. The second `namespace` element and the respective attributes state that all attributes from the ITS namespace should be validated with the schema `its.rnc`.

There are many other facilities of NRL. It allows giving arguments to the validator for a schema, for example to select strict or lax validation levels. Wildcards can be used for schemas which can contain elements from arbitrary namespaces. And a subset of XPath can be used to select sub sections from a namespace, e.g. all XHTML elements which are children of the `body` element. As for the issues mentioned in section "The need to go beyond schemas", NRL could be used to resolve the issues 4 (incompatibility with existing schemas), 6 (conflicts with between existing markup and ITS markup) and 7 (versioning). Nevertheless, ITS markup still might break processing steps other than validation, which are created for a single markup vocabulary, e.g. XHTML (issue 5).

ITS as schema annotation

Schema annotation is defined in the sense of **[Sperberg-McQueen (2002)]**. It is a means to provide additional information to the schema. The schema itself provides *only* a vocabulary of XML elements, attributes, entities, notations and processing instructions. Depending on the schema language, where might be other components like user defined types in XML Schema. Schema annotation should provide information about the *meaning* of the components of a schema. *Providing Meaning* can be described as a problem of *mapping* to a data structure. What kind of data structure this is, depends on the purpose of the meaning description. E.g. one way of meaning description could be a mapping in tables from a relational database, or into a form of first-order predicate calculus, or into and object-oriented data structure.

In the case of ITS, markup is mapped to other markup or integrated into it, e.g. the ITS vocabulary is mapped to XHTML. That is, there are two separate schemas, for example an XHTML and an ITS schema.

One schema is augmented with information about the relation to the other schema. For example, the `html:span` element can be described to have the same meaning as an `its:span` element. Or two views of the same schema can be described, one with ITS specific information, and one without. This description then can be used to automatically create ITS markup in documents or to filter it.

In the following sections, approaches to schema annotation are discussed which might resolve some issues of ITS.

Architectural forms

Architectural forms are defined in Annex A of the HyTime standard **[ISO/IEC10744]** [10]. Their purpose is to describe relations between schemas in the format of SGML or XML DTDs. The relations are sets of mapping rules. Each set of rules is an archform with a name, e.g. the archfom `its-arch`. Figure 8 shows an architectural form.

Figure 8

```
XML DOCUMENT:
<!DOCTYPE html SYSTEM "xhtml-plus-its.dtd">
<html>
 <head>...</head>
 <body>...
  <its-span title="This needs special handeling"></its-span>
 </body>
</html>

CLIENT DTD "xhtml-plus-its.dtd":
<?IS10744:arch name="its-arch"
               bridge-form="archbridge"
               renamer-att="its-arch.atts"
               dtd-system-id="xhtml1-transitional.dtd"?>
<!ELEMENT html (head, body)>
<!ATTLIST html
  its-arch NAME #FIXED "archbridge">
...

META DTD "xhtml1-transitional.dtd":
<!ELEMENT html (head, body)>
...
<!ELEMENT span (#PCDATA)>
<!ATTLIST span title CDATA #IMPLIED>
...
<!ELEMENT its-span (#PCDATA)>
<!ATTLIST its-span
  its-arch NAME #FIXED "span"
  its-arch.atts CDATA #FIXED "locinfo title">
...
```

The XML document, to which the architecture should be applied to, refers to the DTD `xhtml-plus-its.dtd`. In the terminology of architectural forms, this is the client dtd. The DTD to which the client dtd is being related, is called meta dtd. In the client DTD, the architecture is identified by the processing instruction `ISO10744` which defines the name of the architecture. The architecture is applied via attribute declarations in the client dtd. In the case of the `its-span` element, there are two attributes:

```
<!ATTLIST its-span
 its-arch NAME #FIXED "span"
 arch.atts CDATA #FIXED "locinfo title">
```

The attribute `its-arch` with the fixed value `span` states that `its-span` should be mapped to `span`. The second attribute `arch.atts` is defined in the processing instruction as a renamer attribute. Its value are pairs of values which denote the attribute name in the client dtd and in the meta dtd. In the example, the `locinfo` attribute of `xhtml-plus-its.dtd` is mapped to the `title` attribute of `xhtml1-transitional.dtd`

Architectural forms could resolve many issues of ITS for two reasons. First, the mapping rules are separated by names, i.e. the names of the architectures. It is possible to have several rules, i.e. several architectures for one client dtd. This can be achieved by an additional processing instruction, e.g. `?IS10744:arch name="its-i18n-tool"....` This architecture could map `xhtml-plus-its.dtd` to a format which is important for a specific i18n tool. [Lobin 2000] describes how a whole network of architectures can be used to create several interpretations of the same schemas. Second, the mappings can be used for validation of XML documents and their transformation respectively. Through a network of mappings, an XHTML document with ITS specific markup could be transformed into documents which are useful for specific processing steps.

These facilities of architectural forms resolve many issues of ITS which have been mentioned in section "The need to go beyond schemas": the resolution of the issues 2 (information about the content of attribute), 3 (linking to external information), 4 (incompatibility with existing schemas), 5 (impact on the processing chain), 6 (conflict between existing markup and ITS markup) and 7 (versioning) can benefit from architectural forms. Unfortunately, architectural forms have not been successful for various reasons. The original syntax used for declaring architectural forms was based on SGML-features not available in XML. Probably, the most important reason for the non-success of architectural forms was the limited demand from the praxis. One reason for the definition of architectural forms was the provision of a special purpose, document grammar-driven document transformation mechanism. But for this purpose most users preferred XSLT, or even Perl. Consequently, software support for AFDR is rare.

An alternative for AFDR could be the introduction of an XML based format with similar functionality:

```
<its:purposeSpec>
 <servesPurpose origVoc="span" its="its-span"/>
</its:purposeSpec>
```

The `servesPurpose` element has 2 attributes. `origVoc` denotes an element from the original vocabulary, `its` adds an ITS specific interpretation to this element. XML based formats with similar functionality are currently under development. Part 8 of DSDL (ISO / IEC 19757), DSRL [Document Schema Renaming Language], is an example:

```
<element name="span-its">
 <dsrl:name-map>span</dsrl:name-map>
 <text/>
</element>
```

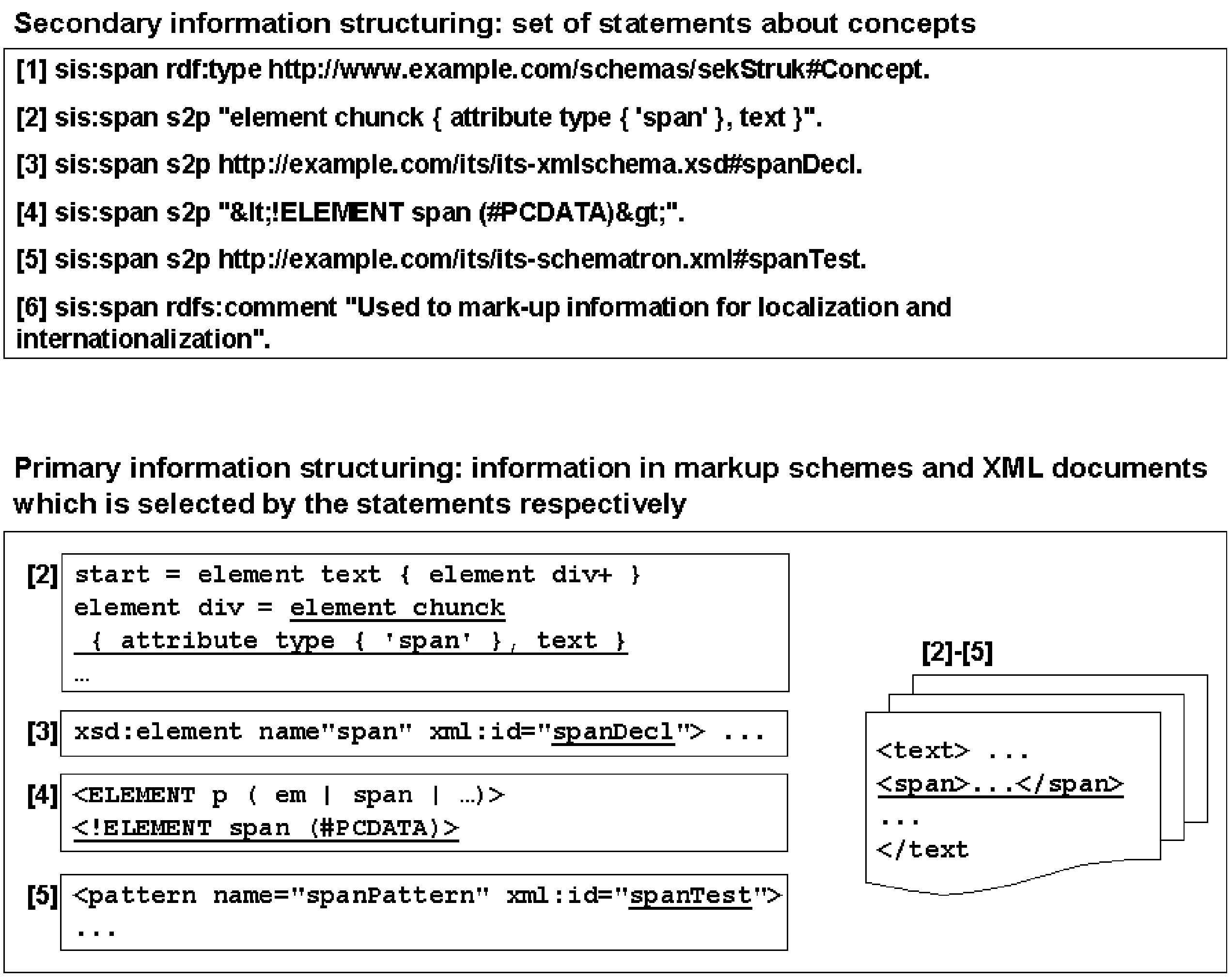
This partial RELAX NG schema declares a `span-its` element. The declaration is enhanced with an `dsrl:name-map` element. It maps the `span-its` element to the `span` element. The problem with DSRL or an XML format for ITS purposes is that they are not yet standards, and it is hard to foresee their future development. On the other hand, although architectural forms are standardized, they are used rarely.

An RDF based approach

Architectural forms describe sets of rules of the form *A maps to B*. A is a markup construct in the client dtd, B is a markup construct in the meta dtd. This approach of schema annotation can be described as rule-based. Another approach, which will be described in this section, is knowledge-based. The approach has been described in detail by [Sasaki 2004]. It is based on work conducted within the research group Text-technological modeling of information, which is funded by the German Research Foundation. [Sasaki et al. (2003)], [Bayerl et al. 1999] and [Witt (2005)] provide further information.

In this approach, schemas and XML documents are an information resource which is named as primary information structuring. The knowledge-based schema annotation approach is conceived as a different level of information structuring, which is called secondary information structuring. The schemas and documents exist independently of secondary information structuring. The approach is called knowledge-based, because RDF [Resource Description Framework] [Manola and Miller 2004] is used to realize secondary information structuring. An example of these statements is given in Figure 9. It shows the description of a **span** in secondary information structuring, and the information in schemas and XML documents to which this description refers to.

Figure 9: Examples of RDF statements for secondary information structuring



Secondary information structuring: set of statements about concepts

[1] sis:span rdf:type http://www.example.com/schemas/sekStruk#Concept.

[2] sis:span s2p "element chunck { attribute type { 'span' }, text }".

[3] sis:span s2p http://example.com/its/its-xmlschema.xsd#spanDecl.

[4] sis:span s2p "<!ELEMENT span (#PCDATA)>".

[5] sis:span s2p http://example.com/its/its-schematron.xml#spanTest.

[6] sis:span rdfs:comment "Used to mark-up information for localization and internationalization".

Primary information structuring: information in markup schemes and XML documents which is selected by the statements respectively

[2]
```
start = element text { element div+ }
element div = element chunck
 { attribute type { 'span' }, text }
...
```

[3]
```
xsd:element name"span" xml:id="spanDecl"> ...
```

[4]
```
<ELEMENT p ( em | span | …)>
<!ELEMENT span (#PCDATA)>
```

[5]
```
<pattern name="spanPattern" xml:id="spanTest">
...
```

[2]-[5]
```
<text> ...
<span>...</span>
...
</text
```

[Link to open this graphic in a separate page]

Six RDF-statements are given. Each statement consists of a subject, e.g. **sis:span**, a predicate, e.g. **s2p**, and an object, e.g. **element chunck { attribute type { 'span' }, text }**. For readability, numbers are assigned to the statements. Statement [1] says that **sis:span** is an instance of the class **Concept**, which is defined in an RDF Schema [Brickley et al. 2004] for the development of secondary information structuring[11]. That is, **sis:span** does not denote an element in a document or an element declaration in a schema, but an abstract unit which can be realized as markup - e.g. as an element `span`, as an element `chunck` with an attribute `type="span"`, and so on. Also, **sis:span** is defined in secondary information structuring independently of a specific schema language. It can be realized in XML DTDs, RELAX NG, and so on.

The realization of **sis:span** is given in primary information structuring. Statements are used to describe the mapping of **sis:span** to one way of realization, i.e. the mapping from secondary information structuring to primary information structuring. The statements [2]-[5] describe mappings to element declarations within various schema languages. The mapping is realized with the predicate **s2p**. This predicate is also defined in the RDF Schema mentioned above. Examples of its usage are given for RELAX NG in its compact syntax [2], an XML Schema document [3], an XML-DTD [4], and a Schematron document [5]. Statement [6] provides a human readable comment about the meaning of **sis:span**.

Three things are to be mentioned: about the mechanism for selecting information in primary information structuring, what is being selected, and what other predicates except **s2p** are available in secondary information structuring.

First, the selection mechanism currently uses URI references[12] to select information encoded in XML based schema languages. Statements with the predicate **s2p** then refer to the unique identifier of an element in a schema. An example is an element with the ID `spanDecl` in an XML Schema, cf. statement [3]. Another way of selection is using the compact syntax of RELAX NG. The declaration is then given as the literal value in the object of the statement, see statement [2]. But it is also possible to use other selection mechanisms which are specific to a schema language. For example, **[Holstege and Vedamuthu (2005)]** define component designators as a format to identify components of an XML schema. They consist of an URI which identifies the schema itself, and a fragment part to specify the component path. The fragment part is an XPointer Scheme, cf. **[Grosso et al. (2003)]**, that uses a schema component path. As an example, the following two statements could replace statement [3]. The first statement maps **span** to a globally defined declaration of a `span` element. The second statement selects a local declaration which is part of a named complex type `paraType`:

```
sis:span s2p http://www.mySchema2.xsd#(/~element::span)
sis:span s2p http://www.mySchema2.xsd#(/~type::paraType/model::choice/element::span)
```

Second, it is also possible to select information in XML documents. For this purpose a path language called caterpillar expressions which is defined by **[Brüggemann-Klein and Wood (2000)]** is being used[13]. For example, to identify all `span` elements which are child elements of a `p` element, the following statement can be used:

```
(sis:-span-in-para
s2p "pathEx 'span' up 'p'".)
```

The pre-defined string `pathEx` identifies the object of the statement as a path expression. The path expression itself consists of a name-test for the element name `span`, a move to the parent of the current node, and a test for the element name `p`. This path expression matches only for - possibly a subset of - `span` elements as the initial node.

Third, other predicates and pre-defined constructs are available to make further statements in secondary information structuring. The following list introduces only a subset of these predicates.

- *sis*: the namespace prefix for concepts in secondary information structuring, for example **sis:span**. Currently, just a dummy-namespace is being used: http://example.com/sekStruk.
- **subConceptOf**: predicate for the creation of a conceptual hierarchy. For example **(sis:span-in-para subConceptOf sis:span.)** expresses that **sis:span-in-para** is sub ordinate to **sis:span**. If the concepts are mapped to markup with the predicate **s2p**, the relation is interpreted as a subset relation respectively. E.g., a `span` element inside a `p` element is a subset of all `span` elements.
- **componentOf**: predicate which is used to describe component relations between concepts. An example is **(sis:locinfo componentOf sis:span.)**. This statement describes that a **sis:locinfo** concept is part of a **sis:span** concept. If the concepts are mapped to markup with the predicate **s2p**, the component relation holds for the markup declarations respectively. E.g., a declaration of a `locinfo` attribute might be a component of a declaration for a `span` element.
- **equal**: predicate for the description of relations between concepts in secondary information structuring, e.g. **(sis:span equal sis:inline.)**. Mapped to markup, this is interpreted as an identity relation between the markup declarations respectively, e.g. a `span` element is equal to an inline element in general. **sis:inline** can be mapped to several elements with several statements using **s2p**, e.g. `html:span` or `html:em`.
- **s2c**: predicate for the mapping between secondary information structuring and another separate conceptual level. For example, **(sis:para s2c "http://www.cogsci.princeton.edu /~wn/concept#104824115".)** describes the mapping to the **paragraph** concept from the WordNet database **[Fellbaum 1998]**. Here, the RDF representation of WordNet developed by Sergey Melnik is being used, see http://www.semanticweb.org/library/.

[Sasaki 2004] describes various processes which could be applied to secondary information structuring. Conceptual queries could be used to retrieve markup independently of element and attribute names, making use of their conceptual description in secondary information structuring. A test whether markup really has the properties which are described in secondary information structuring could be used as a kind of conceptual validation for XML documents. And the statements with the predicate **equal** could be a base for the transformation of primary information structuring, i.e. of marked-up documents or schemas.

Compared to the rule-based approaches of schema annotation in section “Architectural forms”, this knowledge-based approach has three benefits. First, there is no direct interrelation of markup constructs, i.e. their description in secondary information structuring is separated from their realization in primary information structuring. This separation is useful e.g. for the description of relations between schemas, without the need to combine the rules with a given set of element and attribute names in the schemas. Second, secondary information structuring allows to apply inferences, e.g. concerning the properties of concepts which have the **subClassOf** relation. And third, a mapping to further knowledge bases is possible, e.g. to WordNet, terminological data bases etc. Again this mapping is separated from the markup itself.

With secondary information structuring, all the issues of ITS which where mentioned during the discussion of architectural forms in section “Architectural forms” can be resolved as well. In addition, the inference mechanism of **subConceptOf** allows for the specification of inheritance features (issue 8). The disadvantage of secondary information structuring is that it is not part of a standardized format and so far there is no stable implementation available. Nevertheless, its knowledge-based characteristics could be an input for the development of such a format.

ITS as a description of processing chains

The approaches described so far concentrate on two kinds of resources: XML documents and schema languages. Various additional resources have been described, like schema annotation or descriptions namespace sections. We question when is how all these resources could be processed. For this purpose, processing chains could be specified. Hence, this section discusses approaches for the standardized description of processing chains.

Currently various XML based formats for the description of processing chains are being developed. Examples are **[Walsh and Maler (2002)]** or **[Thompson (2005)]** [14]. The following tentative description of a processing chain (see figure 10) relies on XPL [XML Pipeline Language] **[Bruchez and Vernet (2005)]**. This choice of XPL is rather arbitrary, since none of the proposed XML based formats for descriptions of processing chains has reached the status of a widely adopted standard.

Figure 10: Example for a processing chain for ITS related resources

```
<p:config xmlns:p="http://www.orbeon.com/oxf/pipeline"
          xmlns:oxf="http://www.orbeon.com/oxf/processors">

 <p:param name="source-document" type="input"/>
 <p:param name="result-document" type="output"/>

 <p:processor name="oxf:nrl">
  <p:input name="source-document" href="#source-document"/>
  <p:input name="rules" href="nrl-rules.xml"/>
  <p:output name="data" id="xhtml-stream"/>
  <p:output name="data" id="its-stream"/>
 </p:processor>

 <p:processor name="oxf:validation">
  <p:input name="data" href="#xhtml-stream"/>
  <p:input name="schema" href="xhtml.rng"/>
  <p:output name="data" id="xhtml-stream-valid"/>
 </p:processor>

 <p:processor name="oxf:validation">
  <p:input name="data" href="#its-stream"/>
  <p:input name="schema" href="its.xsd"/>
  <p:output name="data" id="its-stream-valid"/>
 </p:processor>

 <p:processor name="oxf:xslt">
```

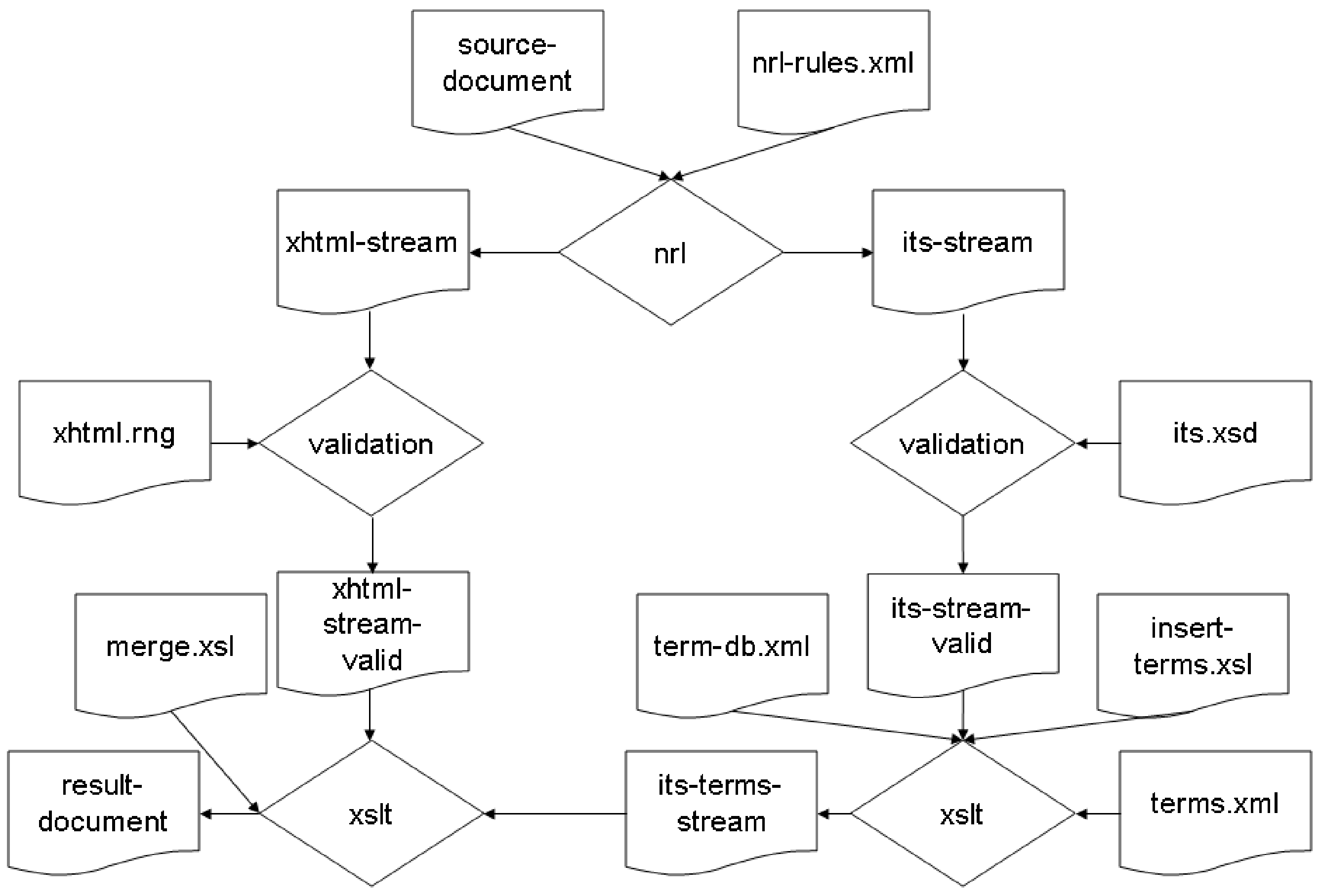
```
  <p:input name="data" href="#its-stream-valid"/>
  <p:input name="data" href="term-db.xml"/>
  <p:input name="config" href="insert-terms.xsl"/>
  <p:output name="data" id="its-terms-stream"/>
 </p:processor>

 <p:processor name="oxf:xslt">
  <p:input name="data" href="#its-terms-stream"/>
  <p:input name="data" href="#xhtml-stream-valid"/>
  <p:input name="config" href="merge.xsl"/>
  <p:output name="data" ref="result-document"/>
 </p:processor>

</p:config>
```

The process chain of the XPL example is visualized in figure 11. The XPL document [15] consists of three processes: `nrl`, `validation` and `xslt`. The processes are evoked via the `p:processor` element. Some of the processes are evoked several times. The `nrl` process has as an input a source document and a description of how to create namespace sections. The rules for the sectioning are given in the file `nrl-rules.xml`. It is assumed that two sections are created, i.e. for the XHTML namespace and for the ITS namespace. The output of this process is described as data streams. The two streams being created are called `xhtml-stream` and `its-stream` respectively.

The `validation` process is evoked two times in parallel, i.e. for the XHTML stream and for the ITS stream. Besides the data to be validated, `validation` has a schema as an input. In the case of XHTML, this is an RELAX NG schema called `xhtml.rng`. In the case of ITS, the schema is an XML Schema called `its.xsd`. The output of the validation processes are possible identical to the input. Nevertheless, in the case of `its.xsd`, the output might be a PSVI (cf. section “Features of schema languages and their relation to ITS”).

The `xslt` process is evoked two times. First, it is evoked with the stream of validated ITS data, and a file `term-db.xml` which could contain a term database. The stylesheet `insert-terms.xsl` is then used for the automatic insertion of terms into the ITS data stream. The output of this process is a data stream `its-terms-streams`. This is an input for another `xslt` process. It merges `its-terms-stream` with `xhtml-stream-valid`, using the stylesheet `merge.xsl`.

Figure 11: Visualization of the processing chain example from figure 10



[Link to open this graphic in a separate page]

Processing pipelines might contribute to the resolution of the following issues in section "The need to go beyond schemas": 1 (requiring uniqueness across documents), 3 (linking to external information), 4 (incompatibility with existing schemas), 5 (impact on processing chains) and 6 (conflict between existing markup and ITS). In general, processing chains could be a means to combine the various approaches for the realization of ITS. The processing chain in the example combines the Namespace Routing Language, the schema languages XML Schema and Relax NG, and general XSLT processing. Nevertheless, some questions remain: Will there be a standardized format for the description of processing chains? Without such a format, reliable implementations might not be available. Another question is concerned with ITS in general: Should ITS really specify more than resources, i.e. XML documents, schemas etc? Or should ITS leave process descriptions to the individual implementations? This crucial decision is still to be made.

Summary and general discussion

The following table repeats the ITS requirements from section "Specific requirements of ITS", relates them to the issues which were raised by their realization (cf. section "The need to go beyond schemas"), and shows possible resolutions which were gained through the discussion in section "Approaches for the realization of ITS".

Table 3: Summary of the discussion

ITS requirement	Classification of requirement	Issues raised by the requirement	Possible resolution(s) for the issues
Specification of	data category	2, 4, 5, 6, 7	Schema languages,

ITS requirement	Classification of requirement	Issues raised by the requirement	Possible resolution(s) for the issues
content that should not be localized			namespace sectioning, schema annotation
Selection of i18n and l10n related information	data category and / or selection mechanism	2, 8	Schema languages, schema annotation, external information (integrated via processing chains)
Specification of inheritance and exceptions	data category and general processing	8	Schema annotation, general processing
Uniqueness across documents	data category	1	General processing, processing chains
SPAN-like element	data category	4, 5, 6, 7	Schema languages, schema annotation, namespace sectioning
Notes to localizers	data category / external information	3, 4, 5, 6, 7	Schema languages, schema annotation, namespace sectioning, general processing
Linking to external information	data category; general processing	3	Schema annotation, processing chains
Emphasis and document convention	data category	4, 5, 6, 7	Schema languages, schema annotation, namespace sectioning
Markup for international script features	data category	4, 5, 6, 7	Schema languages, schema annotation, namespace sectioning

The second column contains a general classification of the requirement: as a data category, as a mechanism for selecting information, as external information, or as a process description. Multiple classifications are possible and happen quite often. Hence, some requirements may rely on alternative or combined realization mechanisms. E.g. the specification of content that should not be localized can be realized with markup, e.g. an attribute `locinfo`, or it can be realized as external information.

It is for sure that this classification is not stable and will change during the development of ITS. Nevertheless it is also clear - and that is the good news - that most of the requirements can be fulfilled with schema languages. It is hard and at the current state of the development of ITS not useful, to compare the various approaches for the realization of ITS. And one has to keep in mind that a lot of XML documents are being created and distributed without a schema at all. Hence, one has to be careful with the combined application of various technologies. An overload of new technologies might not help, but prevent the widespread adoption of ITS.

Notes

1. Any statements contained in this document that are not historical facts are forward-looking statements. The URIs and prefixes used with respect to namespacing mechanism etc. only serve exemplary purposes. Words such as "anticipate," "believe," "estimate," "expect," "forecast," "intend,", "may," "plan," "project," "predict," "should" and "will" and similar expressions as they relate to the W3C ITS WG are intended to identify such forward-looking statements. The W3C ITS WG undertakes no obligation to publicly update or revise any forward-looking statements. All forward-looking statements are subject to various risks and uncertainties that

could cause actual results to differ materially from expectations.

2. The term "data categories" is used in the sense of [ISO/IEC 1087-2:2000], i.e., a type of data field, such as "definition". ISO 12620-2 is an inventory of data categories for use in terminology resources. The term "data category" used in ISO TC 37 corresponds directly to data elements in the environment of ISO/IEC 11179.
3. The element `term` is a user defined extension of the TEI-DTD.
4. See http://www.w3.org/TR/xmlschema-1/#non-ambig for further explanations.
5. XML-based formats and automatic conversion tools for DTDs exist, although they are not part of a standardized format.
6. Proposals exist for enhancing DTDs with Namespaces, e.g. DTD++[Vitali et al. (2003)] .
7. This is only partially true, since - as has been described before - Schematron relies on XPath. In the version 2.0, XPath provides functions which rely heavily on built in data types or data types which are defined in an XML Schema.
8. http://www.w3.org/People/cmsmcq/doclist.html#psvi provides an XSLT stylesheet to create a visualization of the PSVI.
9. See http://dsdl.org/ for further information.
10. A (German) introduction to the usage of AFDR and application scenarios can be found in [Lobin 2000].
11. The complete RDF Schema can be found at http://coli.lili.uni-bielefeld.de/~felix/phd/sekStruk/schemas/sekStruk.rdfs.
12. To be more precise: The selection of some schemas is realized with IRIs (Internationalized Resource Identifiers, cf. http://www.ietf.org/rfc/rfc3987), which identify the respective declarations. In the case of the compact syntax of RELAX NG and XML-DTDs, the declarations are given as literal values respectively.
13. The benefits of using caterpillar expressions instead of XPath are explained in detail in [Sasaki 2004].
14. The current draft of part 10 of DSDL (ISO / IEC 19757) on validation management gives an overview on current approaches. The example in this section is also derived from this draft.
15. The syntax of the XPL document is derived from the example in DSDL part 10.

Bibliography

[Axelsson et al. (2005)] Axelsson, John, Mark Birbeck, Micah Dubinko and others (2005). XHTML 2.0. W3C Working Draft 27 May 2005. http://www.w3.org/TR/2005/WD-xhtml2-20050527/

[Bayerl et al. 1999] Bayerl, Petra Saskia, Harald Lüngen, Daniela Goecke, Andreas Witt, and Daniel Naber: Methods for the semantic analysis of document markup. In: Roisin, C., E. Munson and C. Vanoirbeek (Ed.): Proceedings of the ACM Symposium on Document Engineering (DocEng 2003). pp. 161 - 170

[Berglund et al. (2005)] Berglund, Anders, Scott Boag, Don Chamberlin, Mary F. Fernández, Michael Kay, Jonathan Robie and Jérôme Siméon. XML Path Language (XPath) 2.0. W3C Working Draft 04 April 2005. http://www.w3.org/TR/2005/WD-xpath20-20050404/

[Boag et al 2003] Boag, Scott, Don Chamberlin, Mary F Fernandez, Daniela Florescu, Jonathan Robie and Jérôme Simeon. XQuery 1.0: An XML Query Language. W3C Working Draft 12 November 2003. http://www.w3.org/TR/2003/WD-xquery-20031112/

[Bray et al. 2004] Bray, Tim, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler and François Yergeau. Extensible Markup Language 1.0 (Third Edition). W3C Recommendation 04 February 2004. http://www.w3.org/TR/2004/REC-xml-20040204/

[Brickley et al. 2004] Brickley, Dan and R. V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 Februar 2004. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

[Bruchez and Vernet (2005)] Bruchez, Eric and Alessandro Vernet. XML Pipeline Language (XPL) Version 1.0 (Draft). W3C Member Submisson 11 April 2005. http://www.w3.org/Submission/2005/SUBM-xpl-20050411/

[Brüggemann-Klein and Wood (2000)] Brüggemann-Klein, Anne and Derick Wood. Caterpillars: A Context Specification Technique. *Markup Languages: Theory and Practice 2.1* (2000): pp. 313-331.

[Clark (2003)] Clark, James. Namespace Routing Language (NRL). *Proceedings of XML Conference 2003.* Pennsylvania, 2003.

[Clark and Murata 2001] Clark, James and Makoto Murata. RELAX NG Specification. OASIS Committee Specification 3 December 2001. http://www.oasis-open.org/committees/relax-ng/spec-20011203.html

[Fellbaum 1998] Fellbaum, Christiane (ed.). WordNet. An Electronic Lexical Database. MIT Press, Cambridge, Mass., 1998.

[Grosso et al. (2003)] Grosso, Paul, Eve Maler, Jonathan Marsh and Norman Walsh. XPointer Framework. W3C Recommendation 25 March 2003. http://www.w3.org/TR/2003/REC-xptr-framework-20030325/

[Holstege and Vedamuthu (2005)] Holstege, Mary and Asir S. Vedamuthu. XML Schema: Component Designators. W3C Working Draft 29 March 2005. http://www.w3.org/TR/2005/WD-xmlschema-ref-20050329/

[Ishida (2003)] Ishida, Richard (2003). What you need to know about the bidi algorithm and inline markup. Article, 2003. http://www.w3.org/International/articles/inline-bidi-markup/

[Ishida (2005)] Ishida, Richard. Authoring Techniques for XHTML & HTML Internationalization: Specifying the language of content 1.0. W3C Working Draft 24 February 2005. http://www.w3.org/TR/2005/WD-i18n-html-tech-lang-20050224/

[Ishida and Savourel 2003] Ishida, Richard and Ives Savourel. Requirements for Localizable DTD Design. Working Draft, 2003. Cf. http://people.w3.org/rishida/localizable-dtds/localisable-dtds-030704.html

[ISO/IEC 1087-2:2000] ISO 1087-2:2000. Terminology work - Vocabulary - Part 2: Computer applications

[ISO/IEC10744] Information Technology - Hypermedia/Time-based Structuring Language (HyTime). International Organization for Standardization, 1997.

[Lee and Chu] Lee, Dongwon and Wesley Chu. Comparative Analysis of Six XML Schema Languages. Cf. http://www.cobase.cs.ucla.edu/tech-docs/dongwon/ucla-200008.html.

[Lobin 2000] Lobin, Henning. Informationsmodellierung in XML und SGML. Springer, Berlin, 2000.

[Manola and Miller 2004] Manola, Frank and Eric Miller. RDF Primer. W3C Recommendation 10 February 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

[Marsh and Orchard (2004)] Marsh, Jonathan and David Orchard. XML Inclusion (XInclude) Version 1.0. W3C Recommendation 20 December 2004. http://www.w3.org/TR/2004/REC-xinclude-20041220/

[Murata et al. 2001] Murata, Makoto, Dongwon Lee and Murali Mani. Taxonomy of XML Schema Languages using Formal Language Theory. *Proceedings of Extreme Markup Languages 2001.* Montreal, Canada, 2001.

[Pemperton et al. (2000)] Pemerton, Steven and others. XHTML 1.0. The Extensible HyperText Markup Language (Second Edition). W3C Recommendation 26 January 2000, revised 1 August 2002. http://www.w3.org/TR/xhtml1/..

[Sasaki 2004] Sasaki, Felix. Secondary Information Structuring - A Methodology for the Vertical Interrelation of Information Resources *Proceedings of Extreme Markup Languages 2004.* Montreal, Canada, 2004.

[Sasaki et al. (2003)] Sasaki, Felix, Andreas Witt, and Dieter Metzing. Declarations of Relations, Differences and Transformations between Theory-specific Treebanks: A New Methodology In: Nivre, Joakim (Ed.): *Proceedings of the The Second Workshop on Treebanks and Linguistic Theories* Växjö, pp. 141 - 152

[Savourel (2001)] Savourel, Yves. XML Internationalization and Localization. Sams, 2001.

[Sperberg-McQueen (2002)] Sperberg-McQueen, C. Michael. Notes on Schema Annotation. Draft Paper, 2002. http://www.w3.org/People/cmsmcq/2002/schema-annotation.html

[Sperberg-McQueen et al. 1994] Sperberg-McQueen, C. Michael and Lou Burnard, eds. Guidelines for Electronic Text Encoding and Interchange (TEI P3). ACH / ALLC / ACL Text Encoding Initiative, Chicago, Oxford, 1994.

[Suignard et al. (2001)] Suignard, Michel, Masayasu Ishikawa, Martin Dürst, Tex Texin. Ruby Annotation. W3C Recommendation 31 May 2001. http://www.w3.org/TR/ruby/

[Thompson (2005)] Thompson, Henry. MT Pipeline: A Public Service and a New Product. Presentation at XTech 2005. Amsterdam, Netherlands, 2005.

[Thompson et al. 2001] Thompson, Henry, S., David Beech, Murray Maloney and Noah Mendelsohn. XML Schema Part 1: Structures. W3C Recommendation 2 May 2001. http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[Vitali et al. (2003)] Vitali, Fabio, Nicola Amorosi and Nicola Gessa. Datatype- and namespace-aware DTDs. *Proceedings of Extreme Markup Languages 2003.* Montreal, Canada, 2003.

[Walsh and Maler (2002)] Walsh, Norman and Eve Maler (2002). XML Pipeline Definition Language Version 1.0. W3C Note 28 Februar 2002. http://www.w3.org/TR/2002/NOTE-xml-pipeline-20020228/

[Walsh and Muellner 1999] Walsh, Norman and Leonard Muellner. Docbook: The Definitve Guide. O'Reilly, Sebastopol, California, 1999.

[Witt (2005)] Andreas Witt (2005). Multiple Hierarchies: New Aspects of an Old Solution. In: Stefanie Dipper, Michael Götze, and Manfred Stede (eds.). 2005. Heterogeneity in Focus: Creating and and Using Linguistic Databases", volume 2 of "Interdisciplinary Studies on Information Structure (ISIS), Working Papers of the SFB 632. University of Potsdam, Germany. (Corrected reprint of an Extreme Markup 2004 paper) http://www.sfb632.uni-potsdam.de/isis.php

---

Schema Languages & Internationalization Issues: A survey

Felix Sasaki [World Wide Web Consortium]
fsasaki@w3.org
Christian Lieske [SAP AG]
christian.lieske@sap.com
Andreas Witt [Bielefeld University]
andreas.witt@uni-bielefeld.de
