Link to the Balisage Proceedings Home Page at http://www.balisage.net/Proceedings
Link to this paper (entry page with links) at http://www.balisage.net/Proceedings/vol24/html/Palmirani01/BalisageVol24-Palmirani01.html

Symposium on Markup Vocabulary Customization
Proceedings

Akoma Ntoso:

Flexibility and Customization to Meet Different Legal Traditions

Fabio Vitali
Professor, Department of Computer Science and Engineering
University of Bologna
<fabio.vtali@unibo.it>

Monica Palmirani
Professor, CIRSFID, School of Law
University of Bologna
<monica.palmirani@unibo.it>

Symposium on Markup Vocabulary Customization
July 29, 2019

How to cite this paper

Vitali, Fabio, and Monica Palmirani. “Akoma Ntoso: Flexibility and Customization to Meet Different Legal Traditions.” Presented at Symposium on Markup Vocabulary Customization, Washington, DC, July 29, 2019. In *Proceedings of the Symposium on Markup Vocabulary Customization*. Balisage Series on Markup Technologies, vol. 24 (2019). https://doi.org/10.4242/BalisageVol24.Palmirani01.

Abstract

We present different techniques to manage customization of Akoma Ntoso XSD, an OASIS XML vocabulary for legal documents, using native elements, like <foreign> or <proprietary>, general elements, modules or tools.

Table of Contents

1. Introduction

The Akoma Ntoso (AKN) [AKN-voc][1] standard is an XML vocabulary that represents in machine-readable format parliamentary, legislative, judicial, and soft law documents. Generally speaking, any deliberative document can be modelled in Akoma Ntoso. It was originally developed in the context of the United Nations Department of Economic and Social Affairs (UN-DESA) initiative to support African Parliaments in adopting ICT to foster transparency and accountability. In 2012 the UN handed over the language and activities to the OASIS LegalDocumentML Technical Committee [AKNOASIS] (LegalDocML), where it was ratified as an official standard in 2018 (the XSD schema) and 2019 (the naming convention of IDs and IRIs).

Akoma Ntoso has been designed to provide a unique XSD schema to manage all kinds of legal documents, in all legal

systems, while at the same time providing enough flexibility to adapt to all the variations of documents, languages and legal traditions. Akoma Ntoso is used in several parliaments (e.g., Senate of Italy [Senate Italy], European Parliament [Eu Parliament], European Commission [Leos]), gazette (e.g., Luxembourg Gazette [Luxembourg Gazette], The National Archives of UK [Archive Uk]), official institutions (e.g., USA Code Review [US Code]), associations (e.g., Nyaaya) and UN agencies (e.g., AKN4UN). The variety of documents managed by Akoma Ntoso makes this standard flexible for any needs and in the meantime, by using design patterns [Di Iorio 2012, Di Iorio 2014, Vitali 2005], it is sufficiently prescriptive to support interoperability. Akoma Ntoso manages bills, acts, debates and debate transcripts, reports and minutes, judgments and decisions, general documents and document collections such as gazettes or official publications.

The Akoma Ntoso standard aims to represent and annotate the structure and the semantics of digital legal documents modelling legal knowledge and making it accessible to machine-driven processes. Akoma Ntoso improves the quality of the legal documents, increases efficiency, accountability and transparency of deliberative institutions, supports interoperability between different bodies. It is a document-centric standard that preserves the integrity of authoritative texts and designed to be flexible, but also it supports exceptions [Vitali 2011, Palmirani 2011]. Since Akoma Ntoso vocabulary aims to support different legal traditions it has to be extensible to address the individual requirements of organizations or specific characteristics of different legal systems and languages. Particularly important is the temporal model of laws [Palmirani 2012], which can become quite complicated with issues such as retroactivity, ultractivity and their effects on laws modifying other laws, and for which Akoma Ntoso provides a simple and rigorous approach that fully tracks and describes the temporal evolution and timed events affecting a legal text.

2. Customization requirements

The design of the Akoma Ntoso standard natively includes different mechanisms to support interoperability and customization to adapt the schema to the specific needs of the adopter.

One of the most important customizations of Akoma Ntoso was created for the United Nations. Between June 2016 and March 2017 the High Level Committee on Management Secretariat, in collaboration with FAO and UN/DGACM as co-leading entities, set up a Working Group on Document Standards (WGDS) to work towards the definition of a UN Semantic Interoperability Framework (hereinafter UNSIF). During this period a customization of Akoma Ntoso was done to model all official UN documents. The custom vocabulary AKN4UN was created and in late March 2017 the HLCM [HLCM-2017] adopted it as a fundamental component of the UNSIF for normative and parliamentary documents [UNSIF].

The requirements could be summarized as follow:

1. To restrict the use of some elements according to the drafting tradition of the institution adopter. For example, United Kingdom uses <section> as the basic normative units whereas in Italy <article> has that role, while <section> is a higher-level entity grouping <articles>. AKN regulates and differentiates the behaviour of elements such as<section>according to the legal tradition;
2. To expand the schema with external namespace such as MathML for mathematical formula or SVG for drawings;
3. To localize the XSD with new elements or attributes driven by special need, especially semantic inline elements (e.g., <inline refersTo="#english" name="originalLanguage">Original: English</inline>) or structural containers (<hcontainer eId="hcontainer_I" name="romanNumeral">);
4. To translate into other languages the whole vocabulary, in order to make the schema more understandable to end-users (e.g., LexML Brazil [Lexml-Brazil] is a derivation of Akoma Ntoso that uses <artigo> instead of <article>). This has been a frequent request of national governments across the globe, however we do not consider it a customization of AKN, but rather the creation of a new vocabulary that heavily affects existing tools, XSLT, applications and interoperability. We have been working to provide a narrowly defined path for translations of Akoma Ntoso that ease and foster alignment and translatability of the documents. We will not further discuss this problem in this paper.

Usually customization of an XML vocabulary needs two main competences. First of all, an expert of the domain that formalises the need to modify the schema according his/her perception of the existing vocabulary. With Akoma Ntoso these requests are usually resolvable using the existing structures of Akoma Ntoso and appropriate references to the metadata block. However, in case the problem is not resolvable with the current elements, a technical expert of Akoma Ntoso is necessary to find a technical solution considering both the technical design choices as well as the legal constraints and the legal analysis. This process can become fairly long and a good tuning between legal and technical AKN-XML experts takes time. In order to optimize the process, the language has adopted a number of mechanisms to make sure that customizations of the language can be done in a reasonable time and style and correctness.

In general, there are two basic varieties of customization options on any XML vocabulary:

- Restrictions, where derived rules are stricter than the base ones, and the set of valid documents for the derived rules is a subset of the set of valid documents according to the base rules.
- Extensions where derived rules add in a controlled way new features to base rules, and the set of valid documents

according to the derived rules is a controlled superset of the set of valid documents according to the base rules.

In the following sections we present some ways through which Akoma Ntoso can be customized via restrictions and extensions.

3 Restrictions

The first and most obvious type of restriction is simply to avoid using undesired elements and attributes. In fact, since restrictions basically means eliminating from actual document all unneeded optional elements and attributes, it is possible to obtain exactly the desired restriction by not using the optional elements that are not needed in the specific context, without affecting the schema at all. Self-constrained markup is thus the easiest way to do restrictions, and although it might sound trivial, it is by far the most effective and simple way to customize the language.

On the other hand, the main reason of using a schema is to have a mechanism that verifies whether the XML documents follow specified rules, and self-constrained markup has no supporting tool to verify whether self-constraint is actually working. Self-constrained markup makes sense either when the individual author of the XML markup is also the author of the self-constraint rules, or when it is possible to implement the self-constraint rules in a separate tool, e.g. an XML editor. In this case, the tool may prevent user to select and use undesired elements and attributes, and restrictions are automatic and transparent.

If we want to have more control over the markup generated by authors and editing tools are not reliable, we need to provide validation not just against the whole schema, but against the restrictions, as well. We created a web-based tool, called Akoma Ntoso subschema generator (anssg [xxx]), which creates restricted schemas in a simple and controlled environment by selecting and composing predefined modules.

3.1 The Akoma Ntoso SubSchema Generator

Modules are fragments of schema that can be selected and combined freely, always generating a valid subschema of the complete one. In Akoma Ntoso, a modular architecture of the schema has been developed, and a tool has been created that allows anyone to create custom schemas by selecting individual modules of the vocabulary.

The current version of the tool is aligned with the latest version of Akoma Ntoso and is composed of four independent but cooperating tools:

- The "Akoma Ntoso SubSchema Frontend" (AKNSSF), A test installation is available at the URI http://akn.web.cs.unibo.it/akgenerator/.
- The "Akoma Ntoso SubSchema Generator" (AKNSSG), A test installation is available at the URI http://akn.web.cs.unibo.it/akgenerator/schema-generator.php.
- The "Akoma Ntoso SubSchema Validator Frontend" (AKNSSVF), A test installation is available at the URI http://akn.web.cs.unibo.it/akgenerator/validator.html.
- The "Akoma Ntoso SubSchema Validator" (AKNSSV), A test installation is available at the URI http://akn.web.cs.unibo.it/akgenerator/schema-validator.php.

These tools introduce three types of customization mechanisms:

- Modules, i.e., collections of XML structures that can be freely selected and combined.
- Preset combinations, i.e., lists of the predefined modules that make sense together to handle specific sets of requirements.
- Rules: The subschema generator allows the specifications of co-constraints using XSD1.1 with conditional type assignments and assertions. These rules can be specified with a textual syntax and/or with an interactive interface for the most frequent cases.

Akoma Ntoso uses 34 modules, shown in fig. 1. Only one of them, the core set, is always required and cannot be deselected, but users can choose any of the other optional modules. They are organized firstly in document types (legislation, reports, amendments, judgments, collections) and then for optional features (specific elements such as titled blocks, tables of content, etc.).

Figure 1: Akoma Ntoso SubSchema Frontend

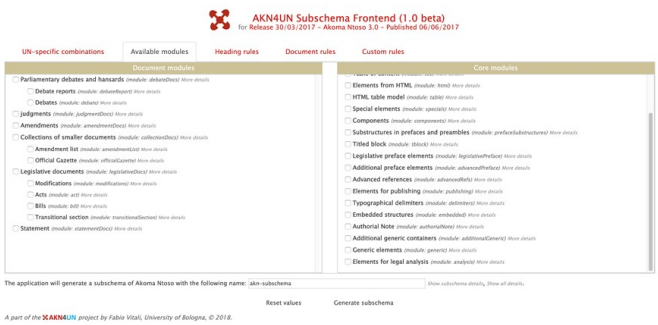

fig. 1: the modules' interface of the Akoma Ntoso SubSchema Frontend

To further simplify the creation of subschema, preset combinations have been created for the most common situations. They are available in a different page of the same application, shown in fig. 2.

Figure 2: Akoma Ntoso SubSchema Generator

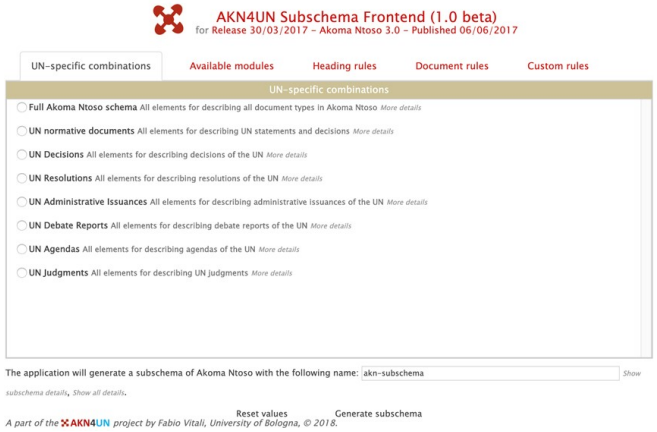

fig. 2: The combinations' interface of the Akoma Ntoso SubSchema Generator

The remaining panels are used to select, customize or create co-constraints on the schema. In particular, the third pane, "Heading rules", gives access to a mechanism to control the content model of the hierarchical containers of the schema (hcontainers).

Figure 3: Akoma Ntoso SubSchema Interface

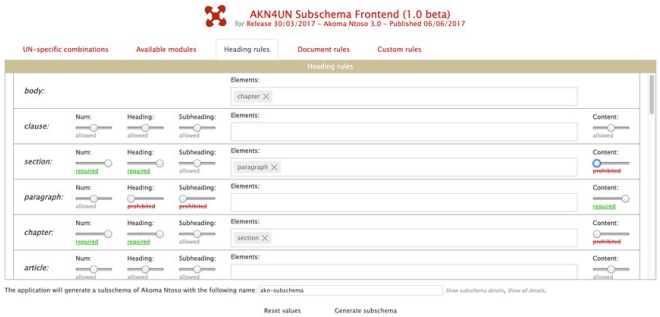

fig. 3: The heading rules interface of the Akoma Ntoso SubSchema Generator

The Heading Pane allows users to select, for each hierarchical element, whether num, heading, subheading and content are required, allowed or prohibited, and which other hierarchical containers are allowed.

3.2 The Akoma Ntoso SubSchema Generator and its co-constraints

The Subschema Generator tool is better accessed through the Akoma Ntoso Subschema Frontend, but in reality, this is an autonomous web service accessible by any other tool through a simple REST interface. The subschema accepts a list of module names, as specified in the actual XSD schema and chosen in the first two tabs of the Frontend, and of rules, detailed in the third, fourth and fifth tabs.

Rules are additional XSD Schema 1.1 co-constraints. Rules can specify further constraints to the content model of any element of the schema, according to a simple syntax. They allow to specify required elements, prohibited elements, or the full list of allowed elements.

As an example, the rule 11_hcm-chapter-reqnum-reqheading-section-nocontent is decomposed as follows:

- *11_hcm:* prefix for hierarchical content model using XSD 1.1 rules
- *chapter:* the element to which the rule applies is <section>
- *reqnum:* element <num> is required
- *reqheading:* element <heading> is required
- *section:* elements <section>are allowed.
- *nocontent:* element <content> (the one that contains any actual text) is forbidden

The SSG tool will generate modifications of the schema in a controlled way. In particular, the above-mentioned rule adds a new specification replacing the original one, where the

```
<xsd:assert>
```

blocks contain the additional constraint to be satisfied.

```
<xsd:element name="chapter">
   <xsd:complexType>
               <xsd:complexContent>
                       <xsd:extension base="hierarchy">
                               <xsd:assert test="every $x in (*) satisfies $x = (num | heading | subheading | section)"/>
                       <xsd:assert test="num"/>
                       <xsd:assert test="heading"/>
                       <xsd:assert test="not(content)"/>
                       </xsd:extension>
               </xsd:complexContent>
       </xsd:complexType>
</xsd:element>
```

Rules are orthogonal and independent of each other and of the content models specified in the rest of the schema. The actual validation corresponds therefore to the global intersection of the full list of constraints. Thus, if the basic complex type of an element allows an element, and the co-constraint rule forbids it, the element is prohibited, and vice-versa.

3.3 Validating Akoma Ntoso SubSchema using XSD1.1 rules

The Akoma Ntoso Subschema Validator Frontend (AKNSSVF), currently available at the URI: http://akn.web.cs.unibo.it /akgenerator/validator.html, is a web interface to activate and control the Schema Validator (AKNSSV) tool, by allowing users to upload Akoma Ntoso subschemas and XML files and verify the validation results of each. In reality, the Akoma Ntoso Schema Validator provides validation for any XML file against the main Akoma Ntoso schema or any subschema thereof.

The co-constraint rules of the subschema generator REQUIRE an XML Schema 1.1 validation engine with full XPath support. The XML Schema 1.1 standard mandates support only for an extremely simplified syntax of XPath for co-constraints (see section 3.11.6.3), and leaves to individual implementations the choice of providing the full XPath 2.0 language as an option. Unfortunately, the co-constraints used in Akoma Ntoso require full XPath 2.0 support.

At the moment, only two XML parsers provide XML Schema 1.1 validation with full XPath support: Saxon Enterprise Edition, which is a commercial application (the freeware Saxon Home Edition does not support it), and Apache Xerces, which is an open source tool. Thus, the schema validator is composed of a Java .jar module that loads and activates Xerces 2.12-beta-r1667115 with full XPath support and represents its error messages in JSON, plus a PHP web script that provides a REST interface to the Xerces engine.

4. Extending Akoma Ntoso without a custom schema

In this section we introduce the two mechanisms for creating extensions to the Akoma Ntoso vocabulary that require no modifications or additions to the current set of schemas. Akoma Ntoso strongly differentiates metadata and content, both conceptually and physically: all metadata is contained within a section at the beginning of the document, separate from the markup of the document content. This is connected with the assumption, everywhere in Akoma Ntoso, that the content of the document is authorial and, under precise conditions, authoritative (i.e., coming from an authoritative source such as the legislator itself), while metadata are authored by someone else's than the author of the document and have no authoritative status on their own. Thus, there exist two very ways to provide extensions to the schema for metadata and content.

4.1 Adding new metadata

Akoma Ntoso does not allow existing, pre-defined metadata elements to be extended or used in different ways, but it provides a subsection of the metadata section of the document where any new metadata element can be added without constraints.

Consider the following Akoma Ntoso fragment:

```
<akomantoso>
  <act contains="OriginalVersion">
    <metagt;
      <identification source="#au1"> ... </identification>
     <publication … /gt;
      <proprietary source="#au1"
                xmlns:cirsfid="http://www.cirsfid.unibo.it/proprietary">
          <cirsfid:MissingInfo>
            <cirsfid:mActDate>1992-12-28</cirsfid:mActDate>
          </cirsfid:MissingInfo>
      </proprietary>
    </meta>
….
</akomantoso>
```

Akoma Ntoso makes no restrictions as to the vocabulary or containment constraints of the content of the proprietary block. The only requirement is that all new elements belong to a namespace different than Akoma Ntoso. This guarantees the immediate identification of the new elements. Some attributes should be used whenever possible even in proprietary elements:

1. id: an identification word unique within the whole document;
2. refersTo: a reference to the id of an ontological concept, person, organization, location etc. defined in the reference section of the metadata;
3. href: the address of another document, or the id of a section of the document referenced by the individual metadata element;
4. source: the id of a person or organization (placed in the reference section) providing the metadata element.

4.2 Generic elements

The vocabulary of Akoma Ntoso is very rich and aims at supporting precisely the descriptive needs of document authors. Nonetheless, it is clear that there are cases and situations that could not be foreseen when designing the vocabulary, and that would require specific descriptive elements that do not exist in the current version of the Akoma Ntoso language. For these situations there exists a workaround based on the notion of generic element.

A generic element is an element with a generic name and an attribute where its would-be name can be specified, that requires a very precise content model. Once it is determined that an element with the required name does not exist in the vocabulary, a generic element can be used instead, taking good care that the chosen generic element has the right content model.

Akoma Ntoso includes six patterns account for the content models: markers, inline, block, subflow, hierarchical container, container [xx]. For each of these patterns there exists a corresponding generic element called exactly as the pattern. The name attribute must be used to provide a description of the element, with the explicit requirement that the name attribute should be the name of the element if it were to be accepted in the language as an extension.

For instance, if there were the need for an element wrapping a small fragment of text within a sentence to identify, say, the language of the original version of the document, then the correct content model would be inline, and therefore the inline generic element should be used as showed below:

```
<inline refersTo="#english" name="originalLanguage">Original: English</inline>
```

Akoma Ntoso uses a simple and bare-bones ontological approach for fundamental entities mentioned or characterized in the content of the document, called Top Level Classes (person, organization, etc.). There are exactly 14 TLCs, each of which can be associated to better specified entities in external ontologies and connected to the body of the document through specific inline elements. Whenever there is the need to mark elements with a specific connection to an ontology, entities are the right elements to use.

4.3 The <foreign> element

Akoma Ntoso does not provide markup for situations that are very specific and for which better-suited vocabularies exist already. For instance, mathematical formulas or drawings have well-known standard XML vocabularies that should be used rather than inventing a new one. For these situations, the <foreign> element can be used to specify fragments of content that correspond to structures and data that are not currently managed by Akoma Ntoso. For instance, the following is a valid Akoma Ntoso fragment to show the equation ax2+c:

```
<foreign>
        <math xmlns="http://www.w3.org/1998/Math/MathML">
                <mrow>
                        <mi>a</mi>
```

```
                        <mo></mo>
                        <msup>
                                <mi>x</mi>
                                <mn>2</mn>
                        </msup>
                        <mo>+</mo>
                        <mi>c</mi>
                </mrow>
        </math>
</foreign>
```

The foreign element is a block-level element, which means that it is presumed that its content appears in a vertically isolated block. Furthermore, only fully qualified XML fragments can appear there, and they must belong to a different namespace than Akoma Ntoso.

5. Creating custom schemas

Finally, some customizations to the general Akoma Ntoso schema imply performing actual modifications to the whole schema. Once we begin editing the actual Akoma Ntoso schema, it becomes possible to add here, remove there, and still come out with a valid XML Schema. Of course, even if the majority of the content in the edited schema derives from the original schema, this is not enough to guarantee that the result is a valid customization of Akoma Ntoso, which has strong requirements for compliance. For this reason, it is very easy to generate either an incorrect XML Schema or a correct XML Schema for a language that is not compliant with Akoma Ntoso.

In general and in absolute, the fundamental rule for the customization of Akoma Ntoso is that any document that is correct with regard to the custom rules must be also correct (in validity and spirit) with regard to the full Akoma Ntoso schema.

Concretely, the customization possibilities are limited to what was described in the two previous sections (paragraphs xxx, xxx).

In all cases, although opinions may differ on the most elegant way to proceed, the kinds of modifications that are possible are pretty much the same: you can derive types (most often to restrict rather than extend, as there are many more constraints in Akoma Ntoso with regard to extensions), you can define new attributes (in a different namespace) for any existing elements, or you can define new elements (in a different namespace) within the existing elements that allow them, such as proprietary or foreign.

5.1 Custom types

The first and most evident customization of the schema is the redefinition of existing types. Regardless of whether one creates a new type, restricts an existing type or edits the main Akoma Ntoso definition of the existing type, the basic requirement is that the fundamental rule for the customization of Akoma Ntos holds: the resulting type must make sure that valid documents for the custom schema are valid documents for the main schema.

5.2 Custom attributes

While being very rigid about new elements, the Akoma Ntoso schema is much more flexible about custom attributes to existing elements. The rule is that it is possible to create new attributes and assign them to any of the existing elements, as long as these attributes are assigned to a different namespace than Akoma Ntoso.

XML Schema requires that each XSD file is associated to only one namespace (its target namespace), so that the definition of the new attributes must be in a different namespace than the schema. This means that we need two separate schemas if we edit the Akoma Ntoso schema directly, and three files if we redefine the schema: the original Akoma Ntoso schema, the schema containing the new attributes, and the pivot schema that redefines the Akoma Ntoso structures.

Although XML Schema would allow custom attributes to be specified in the Akoma Ntoso namespace, this would go against the rule of the Akoma Ntoso language and should not be performed.

5.3 Custom elements

A shown in the previous section, defining new attributes is easy, and there is no requirement for sequence and organization of the text content of the document. On the other hand, new elements are a completely different problem, because Akoma Ntoso expects a specific sequence of containment when dealing with the actual text content of a document. Custom elements, therefore are only indirectly introduced as generic elements or within special contexts, i.e. those whose type is defined as anyOtherType, a list which includes only one content-oriented element (i.e., <foreign>) and several metadata element.

6. Governance of Akoma Ntoso

Akoma Ntoso is managed by the OASIS LegalDocML Technical Committee. Future versions of the Akoma Ntoso XSD schema will not change the namespace and an optional attribute in the root element will be used to identify the version of the vocabulary. Backward compatibility is very recommended.

7. Conclusions

Akoma Ntoso is a mature and robust XML vocabulary for modelling legal documents. Because the legal traditions could have peculiarities, extensibility is managed in Akoma Ntoso with three main mechanisms: restriction, extension, modifications. Those mechanisms permits to adapt Akoma Ntoso to each specific need of the adopter and in meantime to maintain the basic levels of interoperability thanks the patterns.

References

[Di Iorio 2012] Di Iorio A., Peroni S., Poggi F., Vitali F. (2012) “A first approach to the automatic recognition of structural patterns in XML documents”. *Proceedings of the 2012 ACM Symposium on Document Engineering*, pp. 85-94. doi:https://doi.org/10.1145/2361354.2361374.

[Di Iorio 2014] Di Iorio A., Peroni S., Poggi F., Vitali F. (2014) “Dealing with structural patterns of XML documents”. *JASIST* 65(9), pp. 1884-1900. doi:https://doi.org/10.1002/asi.23088.

[Palmirani 2012] Palmirani M. (2012) “Legislative XML: Principles and technical tools”. ROMA, Aracne, 2012, pp. 161.

[Palmirani 2011] Palmirani M., Vitali F. (2011) “Akoma-Ntoso for Legal Documents”. In: Sartor G., Palmirani M., Francesconi E., Biasiotti M. (eds) *Legislative XML for the Semantic Web*. Law, Governance and Technology Series, vol 4. Springer, Dordrecht, pp. 75-100.

[Vitali 2005] Vitali F., Di Iorio A., Gubellini D. (2005)“Design patterns for descriptive document substructures”. Extreme Markup Languages Conference, Montréal, Canada. https://www.academia.edu/2730273/Design_patterns_for_descriptive_document_substructures.

[Vitali 2011] Vitali F. (2011) “A Standard-Based Approach for the Management of Legislative Documents”. In: Sartor G., Palmirani M., Francesconi E., Biasiotti M. (eds) *Legislative XML for the Semantic Web*. Law, Governance and Technology Series, vol 4. Springer, Dordrecht, pp. 35 – 47.

[Archive Uk] “The National Achives of UK”. http://leggovuk.s3-website-eu-west-1.amazonaws.com/texts/enacted-epublished/akn/ukpga/index.htm.

[Lexml-Brazil] “LexML-Brazil”. http://projeto.lexml.gov.br/documentacao/Parte-3-XML-Schema.pdf.

[US Code] “United States Code of Office of the Law Revision Counsel”. http://uscode.house.gov/download/download.shtml.

[Nyaaya] “Nyaaya initative”. https://github.com/nyaayaIN/laws-of-india.

[Senate Italy] “Senate of Italy bulk of Akoma Ntoso XML files”. https://github.com/SenatoDellaRepubblica/AkomaNtosoBulkData.

[Leos] “Leos project of the European Commission”. https://joinup.ec.europa.eu/sites/default/files/document/2012-05/ISA_LEOS_Final_Results_Final_Version.pdf.

[AKN4UN] “AKN4UN”. https://unsceb-hlcm.github.io/part1/.

[AKNOASIS] “Akoma Ntoso OASIS Technical Committee”. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=legaldocml.

[AKN-voc] “Akoma Ntoso OASIS Vocabulary”. http://docs.oasis-open.org/legaldocml/akn-core/v1.0/akn-core-v1.0-part1-vocabulary.html.

[HLCM-2017] “Delibertion of the High-Level Committee on Management March 2017”. https://www.unsystem.org/content/report-33rd-session-march-2017-budapest.

[UNSIF] “United Nations Semantic Interoperability Framework”. https://www.unsystem.org/content/unsif?page=3.

[Eu Parliament] “European Parliament AT4AM”. https://joinup.ec.europa.eu/solution/at4am-all/about.

[Luxembourg Gazette] “Luxembourg Gazette example of Akoma Ntoso file”. http://data.legilux.public.lu/eli/etat/adm/pa/2019/07/12/b2143/jo/fr/xml.

---

[1] Akoma Ntoso means "linked hearts" in the Akan language of West Africa.

Author's keywords for this paper: Legal XML vocabulary; markup language; modules; Akoma Ntoso

Fabio Vitali
Professor, Department of Computer Science and Engineering
University of Bologna

<fabio.vtali@unibo.it>

Fabio Vitali is full professor of Computer Science at the University of Bologna, co-chair of the LegalDocML technical Committee in OASIS.

Monica Palmirani
Professor, CIRSFID, School of Law
University of Bologna
<monica.palmirani@unibo.it>

Monica Palmirani is full professor of Legal Informatics, she is co-chair of LegalDocML and LegalRuleML OASIS technical committees.

Balisage Series on Markup Technologies