XML BASED DOCUMENT MANAGEMENT IN ESTONIAN LEGISLATIVE SYSTEM

Kristo Heero
Cybernetica
Lai 6, 51005 Tartu, Estonia
kristo@cyber.ee

Uuno Puus
Cybernetica
Lai 6, 51005 Tartu, Estonia
uuno@cyber.ee

Jan Willemson
Cybernetica
Lai 6, 51005 Tartu, Estonia
jan@cyber.ee

Abstract One of the major problems arising in the process of changing the document management infrastructure from paper-based to electronic in a large scale, is the question of standards. In order to provide a unified framework for different organizations that need to create and modify common documents, the solution must definitely include standardizing the document formats as well. An important example of joint document management is the process of legal acts creation in collaboration by several governmental institutions. The current paper is devoted to analyzing various aspects of this process and its standardization in Estonia. We discuss the problems that occurred when XML technology was introduced as the basis for new document types and highlight several possibilities provided by the technology, but unused for different reasons.

Keywords: XML, digital document management, legislative information systems

1. Introduction

Document management is a vital part of the work of every modern organization. Of course it is not always enough to consider the problems of document management in the context of one institution only. Often different types of documents are created as a joint effort of several establishments. A common example is given by the bill preparation in government. Typically, one ministry writes the first draft, then other ministries comment it and after discussing all the comments the second draft is sent to parliament. The parliament members and commissions can discuss and change the bill a few more times before it gets accepted as a law.

It is clear that such a process must be quite strictly regulated, because errors in the final product (i.e. the text of the legal act) are highly undesirable. With the history of law creation going back many centuries, every country has developed its own methods and traditions for ensuring the optimal outcome, and Estonia is no exception. On the other hand, evolution of information technology has provided us with powerful means that can ease and speed up the whole process considerably.

Before these means can be applied to the legislative process, several preconditions must be satisfied. One of the necessary building blocks is the unified format for data exchange, which in the case of document creation means an agreement on the document type definition.

In order to analyze the document flow in the public sector and to introduce necessary technology updates in Estonia, governmental Document Handling Project (DHP) was initiated in June, 1999. The legislative system was chosen to be one of the first areas in the DHP framework.

There are three major projects currently carried on within this framework:

- eLaw (*eÕigus*) project studies the process of legislative act and bill preparation;
- VIIS project builds an information system for the government's meetings;
- electronic Legal Acts (*elektrooniline Riigi Teataja*) project is targeted towards creating unified document structures and work-flows to ease the Legal Acts of Estonia (the official brochure for legislative acts) publishing process and support direct web publishing.

This paper covers the work done when developing suitable document type definitions for the third, electronic Legal Acts project.

The idea of introducing unified document types as the basis of a legislative information system is not new in the world scale. Historically the first widely accepted technology for defining suitable document types was Standard Generalized Markup Language (SGML) [5]. This technology has been applied successfully e.g. in Tasmania [3], Sweden [6] and Finland [9].

In 2000 a further development (or, to be more precise, a subset) of SGML, called eXtensible Markup Language (XML) was accepted as a standard by W3C [4]. XML framework was for example chosen to be the basis for legislative documents of United States Congress, currently the document type definitions (DTDs) for bills and resolutions are available [2]. As the popularity of XML has increased considerably since its standardization, the decision was made to use it as the basic technology for Estonian legislative system as well.

The structure of the paper is as follows. Section 2 highlights the differences between the points of view of lawyers and IT-specialists to the new information system. Section 3 describes several new possibilities enabled by XML and explains why they were not used in Estonian legislative system. Section 4 makes some conclusions and gives a couple of questions to be answered during the future work.

2. Conflict of paradigms – traditions *vs.* XML

2.1 WYSIWYG *vs.* structured documents

Legislative information system building requires collaboration between specialists in several fields. In Estonia the working group consisted of the representatives from Ministry of Justice, State Chancellery, Estonian Legal Acts publishing house and several IT-companies. Unfortunately participation from the side of the potential creators of legislative data (parliament, other ministries, local governments) was not very intensive. One of the probable reasons is that the future end users have not yet fully realized that XML documents differ substantially from what they have generally used to nowadays. This does not apply only to the information system level, but also the user interface look and feel has to change.

On the first sight the problem does not appear to be big – several major office productivity suites (Microsoft Office, StarOffice/OpenOffice) are heard being capable of handling XML data. Unfortunately, built to conform to the What-You-See-Is-What-You-Get (WYSIWYG) framework, these programs are not suitable for editing structured documents.

For example, consider the following fragment of an XML DTD:

```
<!ELEMENT section (title,(subsection | text)*)>
<!ELEMENT subsection (title,(text)*)>
```

If we now want create the following visual document:

Section title

Subsection title

My immortal text

it can correspond to two different XML-documents:

```
<section>
   <title>Section title</title>
   <subsection>
      <title>Subsection title</title>
      <text>My immortal text</text>
   </subsection>
</section>
```

and

```
<section>
   <title>Section title</title>
   <subsection>
         <title>Subsection title</title>
   </subsection>
   <text>My immortal text</text>
</section>
```

It is very hard to make a clear distinction whether the text belongs to the first or to the second level heading. Hence a good XML-editor must have tools for asking explicitly which level element is the user adding to the document tree, and this question is very difficult to ask in WYSIWYG context. On the other hand, users are currently not used to questions about document structure and introducing such an editor for everyday use means extensive training efforts.

There is another problem with using XML (or in fact any structurally limited document format) as a basis for document management. On one hand one can not do anything with an XML document unless it is explicitly allowed in DTD (e.g. tables, slanted text etc.). On the other hand, current users, including law-writers, have used to express their ideas in quite a free form and stating strict rules means restricting their freedom, which they need not necessarily accept.

As the bottom line we note that in a democratic country like Estonia the parliament has the right to take any decisions it wants to, including passing a law that does not correspond to any known structure.

2.2 The definition of a document

Another area requiring paradigm shift from the legislative community is the notion of a document. Whereas the inability to distinguish the original of a document from its copy is slowly accepted, it still takes some work to understand that a document is not necessarily an integral object. E.g. in the solution proposed for Estonian legislative information system not all of the displayed elements are parts of the XML-document, but are generated with the help of a style sheet. The examples include the sign "§" before the paragraph numbers, the text "Announced by the President of the Republic" and other texts that are standard for all the laws or other types of legislative documents.

In such a way we loose the common paradigm of comparing documents with files; in our case a document rather consists of several files. Of course we can always restore the familiar analogy by `tar`ing the files together or putting them into some other container file. It is also possible to include style information directly into the XML-document, but this generally results in ugly XML files (like the ones Microsoft Office exports for HTML). Additionally, as the original idea of XML about separating the document structure from visual information is lost with this approach, it was not favored for Estonian legislative information system design.

On the other hand, the concept of having the structured XML-file and style information separately caused a major misunderstanding from Estonian State Chancellery. The main question posed to the developers of the system was the following: what happens if someone alters the style sheet in a way that changes the visual representation of the text?

We see that it is in fact an integrity problem and as such can be answered in several ways.

1 Why should we fear only the compromise of the style sheet? The basic XML-document can be a target for a malicious attack as well.

2 Integrity of an XML-document can be established by using cryptographic techniques – digital signatures, server authentication etc. If so, why can't we protect the style sheet the by using the same means?

Consequently, having the structural information separated from the visual one is not a security threat by itself. Still, when applied wrongly, such a threat can occur. For instance if digital signatures are used in order to ensure the integrity, but only the XML-sources are signed without the style sheets, the above-described attack is realistic.

2.3 Tag semantics

When an XML DTD is prepared, descriptive names like `date` or `paragraph` are often given to the tags. Still, the users mostly do not write XML-source by hand, but rather use some nice visual software tools. As a result of a software bug, malicious modification of a style sheet or some other system misbehavior, it is possible that some tags are displayed inadequately. This in turn can cause different interpretations which is very undesirable for legislative texts.

A good solution to this problem is very difficult to propose as the possibility of several visual interpretations of one digital document is a more general one than just applying to XML. The solution can obviously not do without clear definition of the semantics for all the tags used, but this definition itself does not solve the problem. We still need assurance that the software in use reflects the intended meaning of the tags correctly and this can only be achieved with some kind of a software certification. The exact scenarios for this procedure have not been specified yet; depending on the security requirements different techniques like testing and source code review can be applied.

3. Features enabled by XML – but not used

3.1 Style sheet based enumeration

Analyzing the structure of a law or some other legislative text we find that all the main elements of the corresponding documents (parts, chapters, paragraphs etc) are numbered. The enumeration of chapters and paragraphs is continuous, but the numbers of all the other elements are reset to 1 with the start of a higher element.

There are two possible ways of displaying the element numbers.

1 **The manual way.** DTD has special number fields for elements of all levels and when typing in the document the user is required to fill these fields in manually.

2 **The automatic way.** DTD has no fields for numbers and they are generated automatically with the style sheet. This functionality is standardized for instance with W3C's CSS2 style sheets [1].

The second approach is clearly preferable for the people who type in the bills of legislative acts, as with the first solution one has to do a lots of renumbering work when inserting some new elements to the beginning.

Nevertheless the first approach was taken in Estonian electronic Legal Acts project for several reasons.

- Despite the fact that nice enumeration facilities are stated by W3C, the software supporting these facilities is not yet out there. The only enumeration-compliant piece of software currently known to the authors is Opera web browser and this is merely just a display engine, not an editor. Private communication with the XMLMind editor development team revealed that the W3C's recommendation [1] is not very suitable for implementation in visual editors overall.

- Quite often it happens that enumeration of the elements is not continuous. For example, if new paragraphs are added to a law between the paragraphs 13 and 14, they are numbered 13^1, 13^2 etc. It is also possible that a chapter that suits logically between the chapters 6 and 7 is numbered 6^1, but set to be the last one after chapter 12. Hence it must at lest be possible to override the automatic enumeration.

- It is common to set references from one act to another. Sometimes it is necessary to refer not just to the whole act, but e.g. to some paragraph. In order to enable such a reference, one needs to put a label to the paragraph to be referred to. What would be a more natural label than the paragraph's number?

3.2 Automatic change tracking

Quite surprisingly (maybe not for a lawyer, but for an IT-specialist for sure), the most common legislative acts are not the newly produced ones, but rather the acts of changing some old acts. It is possible that a law gets modified several dozens of times and for simple people it is really very difficult to keep up with all the changes (anyway, the business idea of being a lawyer is based on this observation).

One of the main motivations behind the electronic Legal Acts project was to make up-to-date and usable legislative information available for simple users as well. This means keeping all the current full texts of the laws in the database. There are basically two approaches possible for doing it.

1 **The manual way.** Every time a new change is accepted in the parliament, some official in the Ministry of Justice takes the old text, reads the change and modifies the file manually. This is the way full texts have been produced this far and it has not been done after every change, but only when the number of modifications has become too large even for the lawyers to handle.

2 **The automatic way.** If a new law changing some old one is prepared, special XML-tagging can be used to mark the change. For example, it could look something like this:

```
<change>
   <actID>123-XYZ</actID>
   <paragraphnumber>17</paragraphnumber>
   <newformulation>
         <paragraph>
         <number>17</number>
         <text>This paragraph is canceled</text>
      </paragraph>
   </newformulation>
</change>
```

The last piece of code can be analyzed by an XML-parser and used by the information system to replace the 17th paragraph of the law having the ID 123-XYZ with the new content. Before publishing the new full text it is of course necessary to verify the correctness of the automatic change by a human, but we would still save a lots of manual work.

Nevertheless, in Estonian electronic Legal Acts project the first approach was taken. Why? The main reason was much greater complexity needed from the information system than it seems to be necessary in the first sight. Namely, the logical structure of possible changes is rather complicated:

- in principle, it is possible to change any element of the initial XML-document;
- the change can be either canceling an element, modifying its content or adding new elements;
- the way the changes are written down does not follow any particular logic. For instance, it is possible have a paragraph stating the changes in some law, with one subparagraph changing one paragraph, but another subparagraph changing the whole chapter of that law.

Hence a structure for expressing all the necessary modifications would inevitably become too complex for a simple official to handle. It would also be difficult to create a reasonable yet exhaustive visual interface for editing such a structure. Nevertheless, the idea of having the possibility of updating the full texts automatically is not buried yet and it is one possible further development of the Legal Acts information system.

4. Further work

As mentioned in Section 1, SGML and XML based legislative document management has been developed in the world for several years already. During this period not only many DTDs have been written, but also the methodology of document structure analysis has been the object of intensive research [8, 7, 9].

Generally, DTD development is seen as a part of a long iterative process consisting of analysis, DTD and work process redesign, implementation, training and evaluation [9]. Out of these stages, only the primary requirement analysis and the first design round have been completed in Estonia. Implementation of the software necessary for electronic Legal Acts is going on at the time of this writing and the concrete steps for training are not even planned.

Already on the current level of development the need for next iterations can clearly be seen. As noted in Section 2, it takes some time from the future users to get familiar with the concept of structured document handling in the first place, and one can not expect any real feedback before someone actually tries to use the system. This will most probably imply the need for making changes both in DTDs and work processes.

Another important future development is connected to the eLaw project briefly mentioned in Section 1. In a way it can be seen as a continuation project for electronic Legal Acts as most of the document structure of a law is contained in the structure of a bill. Of course, several additional data fields (comments, alternative proposals, various meta-data elements) must be included into a bill in order to enable convenient processing. It is yet unclear which end user software will be used in Estonian legislative system, so this question together with the abovementioned ones remains the subject of future research as well.

References

[1] Cascading Style Sheets, level 2 CSS2 Specification. W3C Recommendation, May 1998.

[2] United States Congress: XML and Legislative Documents, 2001. Project webpage available at `http://xml.house.gov/`.

[3] Timothy Arnold-Moore and Jane Clemes. Connected to the Law: Tasmanian Legislation Using EnAct. *The Journal of Information, Law and Technology*, 1, February 2000.

[4] Tim Bray, Jean Paoli, C. M. Sperberg, and Eve Maler. Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation, October 2000.

[5] C.F. Golrfarb. *The SGML Handbook*. Oxford University Press, 1990.

[6] Cecilia Magnusson-Sjöberg. DTD-development for the Legal Domain, March 1997. "SGML Sweden 97" workshop.

[7] A. Salminen. *Methodology for document analysis*, volume 67 of *Encyclopedia of Library and Information Science*, pages 299–320. Marcel Dekker, Inc., 2000.

[8] A. Salminen, K. Kauppinen, and M. Lehtovaara. Towards a methodology for document analysis. *Journal of the American Society for Information Science, Special Issue on Structured Information/Standards for Document Architectures*, 48(7):644–655, 1997.

[9] Airi Salminen, Virpi Lyytikäinen, Pasi Tiitinen, and Olli Mustajärvi. Experiences of SGML Standardization: The Case of the Finnish Legislative Documents. In *Proceedings of the 34th Hawaii International Conference on System Sciences*, 2001.