AUTHORING REUSABLE SLIDE PRESENTATIONS

Alberto González Téllez
Dept. of Computer Engineering, Polytechnic University of Valencia, Spain

Keywords: Content reuse, Slides, Docbook, SMIL, XML, XSLT.

Abstract: Slide presentations have become a common method used by lecturers to prepare and deliver teaching content. The amount of effort invested in authoring this material is usually very high and then it is desirable to be able to reuse it. Unfortunately commonly used slide editors offer very little reuse capabilities. In this paper we present a method of authoring teaching slide presentations that has the advantage of being able to automatically port slide contents into other delivering formats (textbooks, synchronized multimedia presentations, e-learning platforms, etc). The technique is based on using an XML compliant language with available tools that are user friendly and free.

1 INTRODUCTION

Slide presentations have become a common method to make lectures more effective and efficient (Alley, 2005). To produce good slide presentations requires carefull design and working time (Alley et al., 2005), in order to avoid countereffects like those exposed in (Parker, 2001). The amount of effort invested by lecturers producing their presentations is, after some years, quite big. But as a result they find themselves with hundreds of slides packed in proprietary format. In order to take the maximum profit from this investment it would be desirable to be able to automatically reuse this material as much as posible (several final formats, topic units, exercise collections, multimedia presentations, etc).

A way to separate content from format is to use XML compliant markup languages (W3C rec 04, 2004). This need appeared for instance on the web and this was the reason why the W3C developed XML specification. A widespread XML language oriented to book production is Docbook (Docbook, 2006, Walsh, 2003, Stayton, 2005). It was born in the seventies under the SGML domain with the aim of being used to document computer products, particularly the UNIX operating system. Nowadays Docbook is available in both SGML and XML, and it is being used as a general documentation tool. A dialect from Docbook is Slides that is oriented to slide presentations authoring. Slides markup reuses some of Docbook content elements and it basically changes the hierarchical structure. There are also specific XSLT style sheets (Docbook, 2006) to generate HTML and printing format from Slides content.

Besides of being technically feasible to automatically reuse content it is important to have productive and easy to use authoring tools. XML editors tend to be much less friendly than slide presentation editors because they usually show the text markup instead of a WYSIWYG preview. XMLMind XML Editor or XXE (XXE site, 2006) does not follow this line. This editor is intended to be used by non technical authors that are interested in XML capabilities that are not available in proprietary formats.

The content of the paper is organized as follows: first we describe briefly Slides markup and style sheets. Then we propose some authoring and format generation tools we have found very effective. Finally we describe two slides reuse applications: textbooks and multimedia presentations.

2 SLIDE PRESENTATIONS WITH SLIDES

As stated previously our proposal to author reusable slides, is based on XML, particularly on the Docbook dialect Slides. Norman Walsh defined this language to be used for authoring slide presentation in a similar style as other proprietary tools.

Slides markup language is specified by a public DTD and it includes a subset of Docbook content elements. The hierarchical elements differ from Docbook to make them suitable for presentations.

González Téllez A. (2007).
AUTHORING REUSABLE SLIDE PRESENTATIONS.
In *Proceedings of the Third International Conference on Web Information Systems and Technologies - Society, e-Business and e-Government / e-Learning*, pages 366-371
DOI: 10.5220/0001263703660371

The root element is *slides* and it has as children: *slidesinfo* and a sequence of one or more *foilgroup*.

The *slidesinfo* element includes the next element types:

- Title and subtitle.
- Author information.
- Meta information (release, revision, etc).

Some of *slidesinfo* elements are generated by default by the style sheets (i.e. title and author). To show other elements style sheets customizations is required [6].

A *foilgroup* element contains a sequence of one or more *foil* elements. A *foil* element corresponds to a slide and then can contain a title and a sequence of one or more content elements.

The set of content elements available inside a foil is a small subset of content elements defined inside a section element in Docbook. These content elements can be classified as:

- Text block and text inline.
- Lists of items (ordered, unordered and variable)
- Figures and media objects.
- Tables.
- Links (local, external and web).

A *speakernotes* element is also available to include lecturer guide notes that are not intended to appear in the presentation.

When a Slides document is completed it has to be transformed into presentation format. To accomplish this there are available XSLT style sheets (Docbook, 2006) to generate chunked HTML and XSL-FO. This later format can be transformed into WordML, OpenDocument, PDF or RTF in order to get a final printed format. Slides style sheets can also be customized to get a satisfactory presentation format in combination with CSS.

To apply the style sheets to the slides document we need an XSLT processor. There are many publicly available in both native code like libXML (Libxlst site, 2006) and Java like Saxon (Saxon site, 2006).

2.1 Theory and Exercises

Contents in slides can be classified as theory and exercises. In order to be able to clearly distinguish and manage both types of elements it is convenient to be able to identify them. This can be accomplished with the next slides markup customization:

- The *role* attribute of *foil* elements is restricted to the values: theory, exercise.
- Inside a *foil*, with its *role* attribute set to "exercise", enunciate elements are marked setting its *role* attribute to "enunciate". This has to be done only to direct foil descendants.

With this simple customization we can easily separate theory from exercises. Applications of this capability are for example: publishing an exercise book, create an exercise repository, etc. In order to make exercises more effective solutions can be filtered when the presentation format is generated.

3 AUTHORING TOOLS

The slides DTD and XSLT style sheets are the technical basement of the proposal. In order to make it attractive to final users it is required the availability of user friendly tools that hide as much as possible XML related details.

The tools required are an editor and a format generator. After looking for sometime among the many available free and commercial products we have chosen XXE editor (XXE site, 2006) and XFC XSL utility (XFC site, 2006). They are commercial products from XMLMind but there is a free license for both tools that offers enough functionality to fulfill our requirements.

3.1 Proposed Editor

XXE stands for XMLMind XML Editor, it is Java written general purpose and friendly XML editor. XXE is specially customized to Docbook, Slides and XHTML. There are two views of document content, one shows the XML tree structure and the other is a pseudo WISYWIG view, based on CSS, that is proven to be very comfortable and productive. This friendlier preview is shown in figure 1.

The application has the common main menu and two toolbars. The one below the main menu gives quick access to the most common operations. Below the quick access toolbar there is the navigating toolbar that permits quick location of:

- XIncluded elements.
- Elements in the document element subtree of the active element.
- Sibling elements and ancestor/descendant elements.

The main panel is divided into the editor panel and the tools panel. This later panel permits:

- Attribute setting.
- Spell checking.
- Searching and replacing.

- Selecting especial characters.
- Locating validity errors.

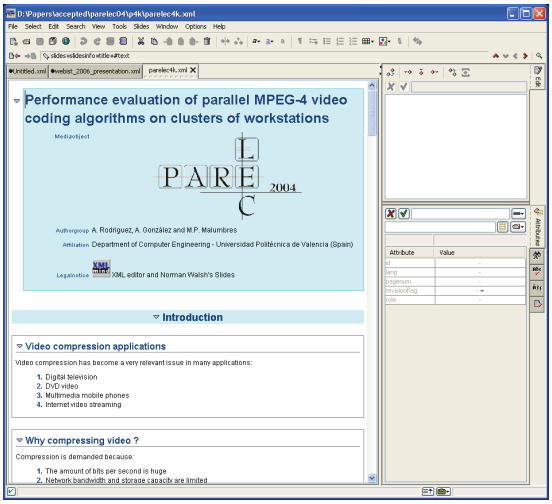

Figure 1: WISYWIG like preview in XXE.

There are two license modalities for XXE: standard and professional. The standard edition is free and it has been artificially limited in functionality. The most relevant features not available are:

- Schema based validation.
- Documents can not be saved to FTP and WebDAV servers.
- It could not be deployed using Java Web Start.
- XSL-FO plug-ins do not work inside XXE.

None of these missing features is relevant for the proposed application because we use DTD validation, our content files are stored in the local file system and we generate format from another application. Then standard edition is the sensible choice.

3.2 Proposed Format Generator

The tool we propose to generate presentation format is XFC (XMLMind Format Converter) XSL utility. It is also written in Java and it has a free personal license.

XFC is in fact an XSL-FO processor that generates WordML, OpenDocument and RTF. The XSL utility also allows generating other formats like HTML and PDF. It includes also Saxon XSLT processor and FOP converter from XSL-FO to PDF.

XFC XSL utility is customized to generate HTML, chunked HTML and printing formats from Docbook and Slides documents. The corresponding XSLT style sheets are included in the distribution. The transformation process is described in figure 2.

The tool is completely customizable the only restriction is that Saxon can not be replaced. New transformations can be defined by setting the style sheet, defining its parameter values, passing the transformation output to a external program (i.e. an XSL-FO processor or a script) and setting an output format viewer (i.e. web client, PDF viewer, etc).

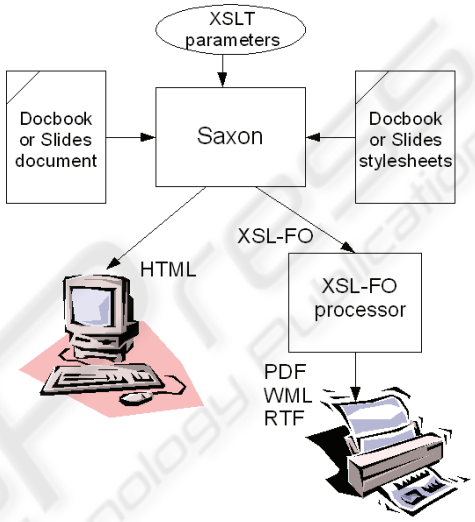

Figure 2: Format generation with XFC XSL utility.

3.3 Collaborative Authoring

Docbook and Slides documents are plaintext, this allows to use concurrent versioning systems like CVS to collaborative authoring teaching material. We use a CVS server on Linux and the java client SmartCVS which is very user friendly and free for non commercial use.

4 SLIDES REUSE EXAMPLES

Next we are going to describe two interesting applications of the proposal capability of flexible reuse. First we will port slides content into textbook by means of transforming slide presentations into Docbook. The second application consists of including slides content into synchronized multimedia presentations based on SMIL (W3C rec 13, 2005).

4.1 Reusing Slides Into Textbook

When a lecturer begins with a subject he or she needs to write a sketch of content in slide format normally based on preexisting information sources. In this first approach we propose to use Slides language as the source.

From the Slides source a chunked HTML format can be generated directly just applying Slides HTML style sheets. A printed format, more convenient for students can be obtained translating Slides to Docbook and then applying Docbook XSL-FO styles sheets and an XSL-FO processor to get final printable format. The XSLT style sheet required to convert Slides to Docbook is very simple because it has to change only the document structure. We can also take advantage of the *role* attribute that is always present in Slides and Docbook elements and does not have predefined values.

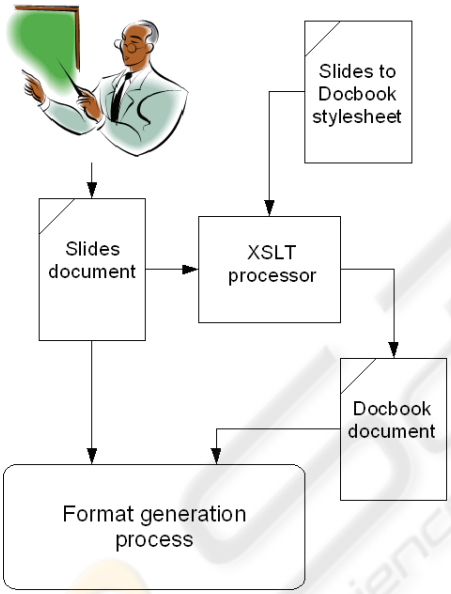

Figure 3: Slides to textbook conversion.

A special attention is paid to exercises. We implement exercises in Docbook by means of *qandaentry* elements. Originally this element was defined to write FAQ collections but it is very well suited to exercises also. *Qandaentry* elements are wrapped in *qandaset* elements. On the other side *qandaentry* elements have as children one *question* element and one or several *answer* elements.

Then the conversion from Slides to Docbook is performed in the next way:

- Slides root *slides* is converted to Docbook root *chapter*.
- Slides *foilgroup* elements are converted to Docbook *section* elements.
- Slides *foil* elements with its role attribute set to "theory" are converted to Docbook *simplesect* elements.
- Slides *foil* elements with its *role* attribute set to "exercise" are folded in a *qandaset* element and converted to Docbook *qandaentry* elements:
  - Foil children with attribute *role* set to "enunciate" are folded inside a *question* element.
  - Other foil children are folded inside an *answer* element.

The conversion process is shown graphically in figure 3.

When the authored material has reached a critical mass then the source can be changed to Docbook Focus now will not be on the presentation summary but on producing a complete textbook. From that moment the Slides document will be generated from the Docbook document by means of custom XSLT style sheets. The Slides document content will be a subset of Docbook content and the lecturer has to be able to define this subset. Content selection will follow the next criteria:

- Text selected from:
  - Inline marked text (i.e. emphasis).
  - <literallayout> elements
  - <programlisting> elements.
- Tables.
- Media as block and inline elements.

Common block text elements (i.e. paragraphs) are filtered in such a way that only inline marked text is selected. The most common inline text element is <emphasis>. Also block elements inside a literal layout or program listing elements are included in the Slides document.

Tables content and media elements are not filtered as they are supposed to be relevant in the presentation format.

As before exercises are specially considered. The textbook can be used as exercises repository then a mechanism is required to select what exercises will be included in the presentation format (i.e. as examples), this can be accomplished by using the *role* attribute of *qandaentry* elements. Selected exercises will generate a foil element according to the previously described design.

XLST style sheets required to perform Slides to Docbook and Docbook to Slides conversions are available at (Slides2Docbook site, 2006) and Docbook2Slides site, 2006), respectively.

4.2 Porting Slides to Multimedia Presentations

In order to keep the student attention alive it is convenient to make class presentations as much dynamic as possible. A way to achieve this is to combine several media elements, not only text and graphics, into a timeline. An open XML standard designed specifically with this goal in mind is SMIL that stands for Synchronized Multimedia Integration Language.

There are several SMIL clients (RealPlayer, Ambulant, QuickTime, etc) and authoring tools (LimSee2, GoLive, GiNS, etc). The fact that SMIL is a compliant XML language allows other alternatives to create SMIL presentations: a plain text editor or automatic generation through XSLT. The later alternative is particularly attractive in our case because we can use as input to the XSLT transformation our slides documents.

Slides language is capable of including media objects. An example is to add to every foil element an audio stream file that will expand the slide content. Other audio and video object can be added in order to make the presentation more attractive.

All the previous objects have to be located in the screen and scheduled in time. SMIL allows to define a layout on the screen, decomposing it into regions, and to schedule media objects in time in sequence and/or in parallel.

In spite of SMIL being a standard, there are many details that have to be considered to produce SMIL presentations: the way text is included and formatted, the media formats supported, the types of linking mechanisms, etc. This means that a particular client has to be selected in order to guaranty that everything will work. We choose RealPlayer because it is a widely spread client and it has a considerable SMIL 2.0 support (Real Networks, 2004). Other interesting features of this client are: it is free, it is multiplatform and it has complementary authoring tools also free and multiplatform like RealProducer Basic.

An institution that has choosen SMIL as their presentation tool is INRIA (Institut National de Recherche en Informatique et en Automatique) in Grenoble, France. Their interest in SMIL is proven by the fact that they have developed an open source SMIL authoring tool called Limsee. A good example of the SMIL capabilities can be seen at the INRIA presentations site (INRIA, 2006)

5 CONCLUSIONS AND FUTURE WORK

We have proposed a method to author teaching slide presentations based on XML, particularly on the Slides language. Our experience using this method has shown that losing some user friendly features, present in more conventional slide editors, is paid off by the ability of reusing content. In the context of teaching topics at graduate level and above, this benefit is particularly relevant.

An important aspect of the proposed method is the authoring tools, particularly the XXE editor that has proven to be very productive to non XML experts.

As present and future work we are developping Docbook and Slides customizations, at markup and style sheets levels (González, 2006), in order to better fulfill teaching material authoring requirements. We also are working on porting Slides document to SMIL as much automatically as possible using RealPlayer as the target client.

Other interesting areas for future development will try to take advantage of other XML related capabilities like content repository management (Dhraief, 2001) and to port content to e-learning platforms (González, 2005, Mengod, 2006).

REFERENCES

Alley, M., 2005. The Craft of Scientific Presentations. Critical Steps to Succeed and Critical Errors to Avoid 4th edition, Springer.

Alley, M., et al. 2005. Pilot Testing of a New Design for resentation Slides to Teach Science and Engineering. 35th ASEE/IEEE Frontiers in Education Conference. Indianapolis, IN, 2005

Parker, I., 2001, Absolute PowerPoint, The New Yorker, 28 May.

W3C rec 04, 2004. XML specs. http://www.w3.org/TR/REC-xml/

Docbook, 2006. Docbook main sites. http://www.docbook.org http://docbook.sourceforge.net/

Walsh, N., 2003. DocBook: The Definitive Guide. O'Reilly

Stayton, B., 2005. DocBook XSL: The Complete Guide. Sagehill Enterprises.

XXE site, 2006. http://www.xmlmind.com/xmleditor/

XFC site. 2006, http://www.xmlmind.com/foconverter/

Libxslt site. 2006, http://xmlsoft.org/XSLT/

Saxon site. 2006, http://saxon.sourceforge.net/

González, A, 2005. Entorno web para la generación y corrección automatizada de exámenes basado en XML

y Java, I Congreso Español de Informática, Granada, Spain.

González, A, 2006. Teaching Document production and management with docbook. WEBIST 2006. Setúbal, Portugal.

INRIA, 2006. Multimedia presentations site http://www.inria.fr/multimedia/exposes

W3C rec 13, 2005. SMIL 2.1 specs http://www.w3.org/TR/2005/REC-SMIL2-20051213/

Rutledge, L., et al., 1999. Anticipating SMIL 2.0: The Developing Cooperative Infrastructure for Multimedia on the Web, in Proc. The Eighth International World WideWeb Conference (WWW8), Toronto, Canada.

Real Networks, 2004, Real Networks Production guide, http://service.real.com/help/library/guides/ProductionGuide/prodguide/realpgd.htm

Dhraief, H., 2001. Open Learning Repositories and Metadata Modelling, in Proc. of Int. Semantic Web Working Symposium, Stanford, USA

Mengod, R. PoliformaT, the Sakai-based on-line campus for UPV - history of a success. 5th Sakai Conference, Vancouver, BC, Canada 30 May - 2 June 2006

Slides2Docbook site, 2006, http://www.disca.upv.es/teachdb/xsl/slides2docbook.xsl

Docbook2Slides site, 2006, http://www.disca.upv.es/teachdb/xsl/docbook2slides.xsl