XML goes to School: Markup for Computer Assisted Learning and Teaching

Clemens H. Cap

Chair for Information and Communication Services, University of Rostock, Germany

Abstract

A basic introduction to the markup concept and *advanced markup techniques* is provided. It is pointed out where these techniques can be *used in computer based education*, and how a successful exploitation of markup ideas requires the development – and standardisation – of *content structuring and usage patterns*, similar to the design pattern of object oriented software engineering. Several conceptual examples of such patterns are provided. The use of the markup approach for developing educational documents is discussed.

Keyword: Tele-learning, markup language, XML, XSL, design patterns

1. Introduction

The last decade has seen dramatic innovations of computer based devices for the end user and an ever increasing number of new proprietary and standardised data formats. Whereas most of them are invented essentially to strengthen the market position of the software producer, some of them satisfy needs of the multimedia camp and an even smaller number provide true innovation. The *proliferation of formats* led to the unfortunate situation that the same contents is reprocessed an increasing number of times, not only to update the contents itself or to reformat it for a new brand of end devices or for new representation software, but also to port it to a new environment when the old one becomes obsolete. With the legal requirements to keep commercial bookkeeping and taxation documents for a time nearly twice as long than the innovation rate of computerised office equipment or the write-off period for tax deduction of the same equipment. Thus, *ensuring proper access to legacy document formats* is a well understood necessity.

In the field of *multimedia*, the innovation of the last decade has made this problem even worse, since virtually every format of importance has been (or is) developed during this period. The demand for user friendly interfaces to authoring systems is further increasing the variety of systems.

In the field of *computer assisted learning and teaching* the contents to be presented to the student changes at a fast pace only for a small number of subjects. The majority of topics, from elementary algebra to college level physics, from Spanish literature to English grammar, from bookkeeping to history are considerably stable – especially at those levels where a large number of learners increase the motivation to seek computer assistance throughout the teaching process.

The market, however, offers a *vast selection of authoring tools*, most of which not suited for the real needs of the content producer. First, the user interface is of a graphical click-and-drag-and-drop kind, requiring the author to produce content as the result of zillions of direct manipulative interactions with a machine, leaving no access to the deeper structure of his productions. Second, in the case of most tools, the content author has to produce the final (graphical, multimedia) representation of his contents, being a specialist for the contents but often not more than an *amateur in representation, graphics and layout*. Third, a change of the target device most likely requires (manual) reprocessing of the representation. Fourth, adapting or restructuring the contents to a new pedagogical setting requires fighting with the representation of the existing content.

Let us have a more detailed look at the last aspect: Contents for a pedagogical setting usually consists of a collection of information items such as motivations, definitions, statements, experiments, observations, questions, examples, exercises, tasks, hints, solutions and more. On the one hand, they are *coherent*, which means they must be well balanced, presented in the right order and are highly dependent on each other. On the other hand, even in a traditional, not computerised learning environment they frequently have to be *rearranged*: A beginner meeting new contents for the first time, preparations for the final exam, lecture notes for the teacher, collections of exercises (with or without hints or solutions), handbooks of possible questions for examinations for the hand of the teacher or handbooks of possible questions together with sample solutions for the hand of the student. All this calls for a methodology where content, structure, representation, and final compilation are all separate concepts. Such a methodology is inherent in the markup concepts, SGML and XML being the most prominent implementations thereof.

This paper analyses the *benefits of the markup concept and connected techniques* for developing and managing educational contents. Section 2 gives an *introduction* to the markup idea and the language XML[2]. Section 3 discusses those *advanced techniques* to which the markup concept readily lends itself and which have later been added to this concept. Section 4 points out where these techniques can be *used in computer based education*, it will become apparent that a successful exploitation of markup ideas requires the development – and standardisation – of *content structuring and usage patterns*, similarly as object orientation benefits enormously from the idea of the software design pattern [GOF]. The core concept of markup-based document production, i.e. separation of content from representation, has often been criticised for leading to inadequate representations or incoherent documents; Section 5 will discuss these issues. The final section will draw some conclusions and propose further development activities in the field.

2. Markup Concepts and Languages

The core idea of the markup concept is to separate the three aspects of a document: *Structure*, i.e. the information on the different parts of a document and the roles they play; *content*, i.e. the information conveyed by a certain part of a document of a specific structure; finally *representation*, i.e. the physical (graphical or multimedia) form in which the document is presented to its reader.

The decision on the structure of a document can be made without providing document contents at the same time, relating a document instance with its template in the same way as an object is related to the definition of its class.

The decision on the content of a document requires an encoding of information, often but not necessarily in the form of text strings, and its linkage to the respective structural elements.

The decision on the representation of a document can comprise a wide range, from the choice of the font size in which to type specific parts of the document over the choice to print the document in graphical form or to have it read by a text to speech engine up to a restructuring and compilation of the document before its final rendition, i.e. a choice, to drop certain parts of the document or to reorder other parts according to given rules.

In the case of a book, the structure usually consists of information on the author (consisting itself of the first name and the last name, an optional middle initial and a short curriculum), of chapters (consisting themselves of sections, subsections, paragraphs, chapter headings etc.), a title of the book, an ISBN number and so on. Several aspects of this information are *data centric*, i.e. consist of (unordered) attributes belonging to entities or related entities (e.g. a book has an attribute "author" and an author has an attribute "last name"), others are *document centric*, i.e. contain elements in a specific order (e.g. first section one, then section two).

The language most commonly used today for *encoding the content* of a document together with structural information is XML[2] (see Figure 1 for an example). The X in XML stands for *extensible* and points out that XML can encode structure and content for a wide range of structures and is not restricted to specific application domains (such as books, for example). In this sense, XML is more a family of document specification languages than a specific language. For the *specification of the structure* of a document alone, so called DTDs were used, specifications written in a language employing a different kind of syntax than XML. Today, there are several XML-based languages for the description of the structure of a document, and XML parsing technology can be used for them: Some of these languages provide additional semantic concepts to XML, such as data types, inheritance structures and restrictions for the values of certain elements. XML itself does not provide facilities for transforming documents into its *final representation*.

```
<?xml version='1.0' encoding='us-ascii' ?>
 <BOOK>
   <TITLE> Java XML Programming</TITLE>
   <AUTHOR>
   <FIRSTNAME>Alexander</FIRSTNAME>
   <LASTNAME>Nakhimovsky</LASTNAME>
   </AUTHOR>
   <AUTHOR>
   <FIRSTNAME>Tom</FIRSTNAME>
   <LASTNAME>Myers</LASTNAME>
   </AUTHOR>

   . . .

   <CHAPTER TITLE="Three-Tier Web Applications">
     This chapter and the next serve as a preparation for the rest of the
     book. We will need two things:
   <LIST>
   <ITEM>a familiarity with Java</ITEM>
   <ITEM>a familiarity with JDBC<7ITEM>
   </LIST>

   . . . and so on . . .

  </BOOK>
```

Figure 1: *A basic XML example.*

3. Advanced Techniques

In order to qualify for the requirements of content management, many add-ons to XML are required. Whereas XML provides a rich structure for a specific content it lacks concepts for its representation: It is not clear, how the first name of an author, encoded as a <FIRSTNAME> subelement of an <AUTHOR> element of a <BOOK> shall be represented, when said book is printed, distributed as electronic book on CD or in the Web or read by a voice processor. To keep the learning curve flat and the costs for parsers low, extensions of the XML concept should themselves be presented in XML-like languages where the individual tags have a specific semantics.

This concept of *assigning a specific meaning* to the tags of an XML-type language could also be employed to solve the representation problem. In fact, this path had been followed partially in HTML. Here, some tags used for structuring such as <H2> or <LI> have a strong connotation on their preferred (although browser- or stylesheet-dependent) graphical representation, other tags such as <STRONG> or <A HREF=' some URL '> have a well defined global semantics. This approach, however, compromises the fundamental idea of markup; more especially, it does not allow the restructuring of a document throughout a transformation process.

To avoid this limitation, in contemporary XML-extensions the following approach is used: There are several languages with a *fixed semantics* either for *representation purposes* (e.g. HTML 4.0, WML, VoxML, HDML, FO, etc.) or for *specific tasks* (e.g. XPath, XPointer, XLink, XQL, XSpan, XForms, etc.). Second, there is a *transformation language* XSL(T) for expressing arbitrary conversions between documents content encoded within the XML-language family. Thus, a book with its content encoded as a <BOOK> is transformed into a document in a representation language (HTML 4.0, FO, etc.) for suitable display to the reader. This transformation process may contain the selection of specific parts of the book, a reordering and total recompilation up to a complex linkage to other documents (a Spanish text for the English learner

of this language could be linked to its English translation, to a Spanish-English dictionary or an explanation of difficult or new grammatical concepts of Spanish). Most of these new languages obey the syntactical rules of XML (although determining the specific semantics of certain expressions, e.g. in the case of XPointer, might require a more thorough syntactical analysis than present in a XML-parser).

Depending on the intended audience, purpose and representation of a document, an XML-encoded content can be transformed into its final version by applying a specific XSL-transformation to this document. Thus, content is separated from representation. See Figure 2 for a rudimentary XSL-transformation program.

Once content and representation are separated, both can be *designed independently* from each other. In the (very elementary) example of the book, the author provides the text for the individual paragraphs and the typographical specialist determines font-sizes and font-styles for the chapter headings and the footnotes.

```
<xsl:stylesheet version='1.0'
     xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
     <xsl:template match='BOOK/TITLE'>
     <HTML>
     <HEAD> <TITLE> <xsl:value-of select='.'> </TITLE> </HEAD>
     <BODY> <xsl:apply-templates/> </BODY>
     </HTML>
     </xsl:template>
     <xsl:template match='CHAPTER'>
     <H1> <xsl:value-of select='@TITLE'> </H1>
     </xsl:template>
     . . . and so on . . .
     </BOOK>
```

Figure 2: *A basic XSL example for a HTML formatting of a book.*

Furthermore, this separation allows some kind of *reuse*: A specific contents can be formatted for different representations and in varying compilations, a specific layout can be used for many documents – and can guarantee a consistent look-and-feel across the border of different contents and authors. This reuse is further increased by the possibility to include documents or specific parts of them in other documents or to provide various kinds of relationships between documents [CAP1]

4. Application Patterns for Computer Based Education

Structure, as the third independent element of a document, can be prepared by an according specialist. In case of a book – a well familiar document – this structure is widely known. A pedagogical document, however, even if it is a book, consists of a much richer structure, not sufficiently explicit in the traditional structure of a book. In *mathematics,* there a motivations, definitions, theorems, proofs, corollaries, lemmas, examples, exercises, hints to the exercises and solutions to the exercises. In *medicine,* there is the patient history, the health background of the family, the diagnosis, possible differential diagnoses, the therapy, contraindications to certain therapies and more. The specialist teaching a certain subject has developed an approach to this structuring, highly dependent on the field he teaches and expression of his personal teaching style.

This structure comprises in teaching what *design patterns* [GOF] comprise in software engineering: An abstract template of a *well known best practice situation*, without specific contents but of a sufficiently generic and parametrisable architecture allowing those skilled in the art to fill it with their own content and benefit at the same time from the guidelines set forth in the design pattern or document structure.

In a most simple example, a letter in international business transactions contains address information on the sender and the receiver, the date of the letter, an initial greeting, a final salutation and more, depending on the corporate and national culture. This culture dictates a certain sequence and layout of elements, different even within European countries. The structural information of which parts a letter should consist as well as a localized transformation producing the expected national layout provide enormous help to the writer.

In a *pedagogical example*, teachers of calculus or of surgery would develop a corpus of information and recompile it according to their specific needs. Thus, a growing collection of high school mathematical problems together with hints, solutions, alternate solutions, typical student errors and suggestions for grading these errors can evolve and be readily transformed into task sheets for an examination, a collection of examples (without solutions) for the hand of the pupil, a collection of examples (with solutions) for the hand of the experienced teacher, a collection of examples (with typical student errors) for the hand of the instructor at a teachers college, etc.

Above example gives a first idea of the variety of applications possible by this approach. It further illustrates the needs for *pedagogical structures and design patterns*. They will highly depend on the subject taught and on the instructional method used, being well beyond the scope of this computer science paper.

- *Omission or inclusion* of certain parts of a document. Example: Include hints or omit them in compilations of student exercises. This pattern is a very simple one but, according to the teaching experience of the author, a very frequently used one, occurring every time a new seminar or lab task or examination sheet has to be prepared from the repository of exercises.

- *Linkage* of certain parts of the representation to further information elements. Example: Every occurrence of a newly defined technical term is linked to its definition or to an illustrating example. The target of the link could depend on attributes of the user or on specific selection actions made by the user. Such links transcend the capabilities of a simple HTML link but can easily be specified using the XML link concept XLink [XLINK].

- *Restructuring* of the presented information, its indexing structure and the access paths for the user. Example: Information on diseases can be accessed by symptoms, by the results of performed lab tests or by therapeutic methods.
- *Annotation* of the document, as links to additional information. Annotations can belong to the document itself or to a specific user of the document. Examples: Annotation by the lecturer for his own preparation, by the student as a study aid, by a teaching assistant to remember steps for a

laboratory exercise. XLink [XLINK] can store link information separate from the document itself thus allowing an annotation of documents by persons who do not have write permission to the document itself. Furthermore individual access permissions can be set to the annotation text by its owner.

- *Interaction* of the student with the system within reasonable limits. Of course, not every interactive computer based training system – e.g. a flight simulator – can be described by this approach. However, certain general interaction patterns can be prepared. Example: The student is presented a question and several offers for an answer. Depending on his answer the system navigates to further information, such as alternative solutions, hints or further explanations. At the contents level, the XML document specifies the prompt, the possible and correct answers and further action to be taken. The XSL transformation then determines, whether a naive, text based multiple choice presentation shall be used or whether more elaborate drag-and-drop or graphical interfaces shall be used. This also may depend on whether the required graphical information (icons, images etc.) are available as part of the contents specification.

5. A Critical Analysis

Many objections can be made against a broad and indiscriminate use of the suggested approach. First, even content production systems as simple as *the generic naive text processors* provide everyday examples *that contents and representation can never be completely separated*: Page breaks one line after the beginning or before the end of a paragraph, badly placed figures or the German word Analphabet, with its hyphenation Anal-phabet, which is grammatically correct but could, occurring at the very end of a page, produce embarrassing associations before the page is turned. The problem becomes worse if a longer document containing equations, photographs and figures is produced by transformation from an XML-document. Second, these difficulties and not reduced to the layout level but also show up with regard to the *correct ordering of document fragments*: Especially in a pedagogical setting information elements are highly dependent on each other, requiring the right path to a new concept. Third, as soon as multimedia or interactive elements are used as opposed to pure text documents, there is the real risk that transformations and automatic compilations produce results which are *not adequate for the employed media*. Fourth, if the transformation approach is used broadly to produce a large variety of different compilations, it is no longer possible to proofread every possible variant which might ever be generated. With implicit, not fully documented or even not fully understood requirements on the XML document or the XSL transformation for a reasonable result it is possible that the mentioned approach occasionally produces garbage.

As a consequence, the proposed method should be used *with care*. Given the present lack of precise empirical studies on the efficiency of multimedia based learning systems this conclusion is hard to detail yet and must be part of further studies.

6. Conclusions and Future Work

First analysis of XML concepts indicate advantages when markup techniques are used for the encoding of teaching material. At present, we are developing toolkits to support the entering and editing of XML documents as well as XSL stylesheets. Furthermore, existing web server concepts are adapted to host the XSL transformation process, programmed in C++, since the available Java-based parsing and transformation tools lack the required execution speed. At the same time we are developing first course elements for an introductory lecture to computer systems to gain more experience with XML usage patterns. Much work still is needed to obtain a detailed catalogue for contents structuring and usage patterns, similar as provided by [GOF] for software design patterns. An empirical analysis of the obtained result would provide important information but will be difficult to conduct on a broad scale.

References

[CAP1] Clemens H. Cap, Reusability for Document Markup: XML Design Patterns Using XLink. Technical Report, Chair of Information and Communication Services, University of Rostock, August 2000, mailto:cap@informatik.uni-rostock.de.

[CAP2] Clemens H. Cap, Virtuelle Lehrveranstaltungen: Möglichkeiten und Grenzen, 2. IuK Tage Mecklenburg-Vorpommern, Rostock, June 1999.

[GOF] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

[SGML] Charles F. Goldfarb and Yuri Rubinsky. The SGML Handbook. Clarendon Press, Oxford, UK, 1990.

[XLINK] World Wide Web Consortium. XML Linking Language (XLink) Version 1.0 W3C candidate recommendation. http://www.w3.org/TR/2000/CR-xlink-20000703.

[XML] World Wide Web Consortium. Extensible Markup Language (xml) 1.0 W3C recommendation. http://www.w3.org/TR/1998/REC-xml-19980210

[XSL] World Wide Web Consortium. Extensible style sheet language (xsl) version 1.0 working draft. http://www.w3.org/TR/xsl, March 2000.

[VOXML] Motorola. Voice Markup Language Voxml. http://www.voxml.com.

[WML] Wireless Application Protocol Forum. Wireless markup language specification. http://www.wapforum.org, 2000.

Author

Clemens Cap, Prof. Dr.
University of Rostock, Department of Computer Science, Chair for Information and Communication

Services
Albert Einstein Straße 21, D-18059 Rostock
mailto:cap@informatik.uni-rostock.de