Efficient Creation of Multi Media eLearning Modules

Hans-Martin Pohl, Patrycja Tulinska, and Jan-Torsten Milde

Fulda University of Applied Science
HCI Research Center
Heinrich-von-Bibra-Platz 3, 36037 Fulda, Germany
hans-martin.pohl@verw.hs-fulda.de, tulinska@aol.com,
jan-torsten.milde@informatik.hs-fulda.de

Abstract. ECampus is a project spanning all departments at the University of Fulda. It has been started to create a uniform learning environment at the university. The objective is to research and develop a user-friendly easy-to-use editor to generate SCORM 2004 conform eLearning modules. This editor is based on Open Source software and new technologies such as XSL transformations and the Google web toolkit. Some eLearning modules can be developed with the system immediately. These modules are now being used during the lessons with great success.

Keywords: SCORM 2004, XSLT, transformation, creation of content, eLearning, modules, lesson, user friendly, style sheet, LOM.

1 Introduction

Creating eLearning units in SCORM 2004 (see [8]) format is a complex task. SCORM, as a technical standard, combines a number of existing eLearning specifications and formats. A SCORM compliant eLearning unit may consist of arbitrary files containing the actual content of the unit. In most cases, the unit's textual content is stored in (X)HTML files, which can be displayed with a standard web browser. Multi media files, e.g. images, flash animations, audio and video files, are stored separately in their proprietary formats.

In addition to the content files, a SCORM 2004 module contains a set of XML files describing the structure of the unit ([12]). Metadata is added in order to facilitate the management of the unit within the Learning Management System (LMS). Sequencing information is also part of these external files, including a difficult to understand specification of conditional constraints for the sequence progression.

For the average author it is almost impossible to generate such a file structure by hand. A small number of editors for SCORM files exist (e.g. Reload, *http://www.reload.ac.uk/scormplayer.html*). Even with these systems, the user still has to have a good understanding of the SCORM file structure. This technical barrier is much too high for most tutors. As a result, standard compliant eLearning content is not produced.

M.J. Smith, G. Salvendy (Eds.): Human Interface, Part II, HCII 2007, LNCS 4558, pp. 457–465, 2007.

2 State of the Art

While there are a lot of different Learning Management Systems at present, the use is limited to but a few in Germany. Hereby 'moodle' (see *http://www.moodle.org*) and 'Ilias' (see *http://www.ilias.de/ios/*) play the greatest role.

In contrast to the LMS there are only a limited number of editors for creating an eLearning module. HTML editors are widely-used (HyperText Markup Language). But this doesn't provide adequate functionality. Furthermore eLearning modules generated by an editor can mostly only be used within a proprietary LMS.

Additionally a lot of effort is required to learn how to use these editors. Such a system fails entirely if there is to be consistent development and presentation of tests and learning success.

The advantage of these systems is the functionality for communication between users or between tutor and student. In our case this can be neglected because this functionality is captured by the LMS 'System2Teach' (see also *http://www.system2teach.de*) which is a proprietary development of the University of Applied Sciences Fulda. This paper focuses on easy and comfortable development of eLearning modules based on textual content by using the eCampus framework.

3 The eCampus Project

This research therefore focuses on the development of a user-friendly system for writing eLearning units, allowing authors to concentrate on the content and the didactic concept of the unit, instead of worrying about the underlying technology.

It is part of the eCampus project at the University of Applied Sciences Fulda. In this interdisciplinary project, content is being created for the faculties N*utritional Sciences*, F*ood and Consumer Sciences*, A*pplied Computer Science*, N*ursing & Health Care* as well as F*ood Technology*. Within the first year of the project nine eLearning modules were produced. All modules address prominent introductory

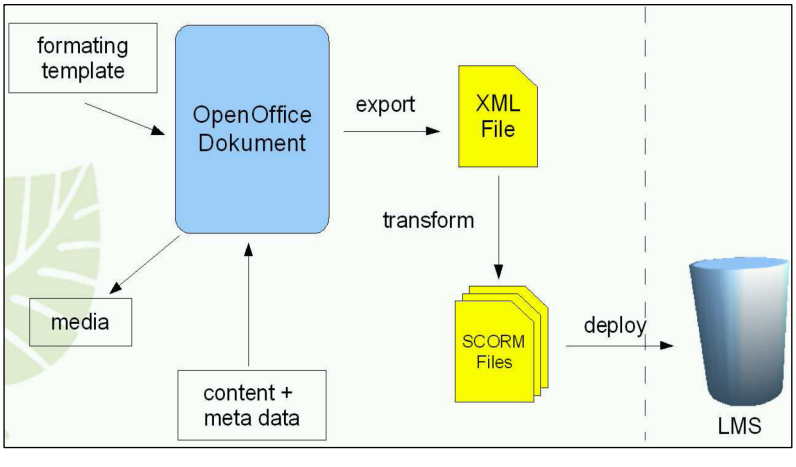

Fig. 1. Scheme of eCampus Framework

courses of the particular program of study. This was only possible by using the eCampus framework which is described below.

The central target of the project is the implementation of a process that decreases the complexity of the process of learning module production. The central objectives of the learning module production are an increase in the interactivity of the course, as well as an increase in the quality and quantity of self learning.

A secondary target of the project is the development of an LMS that is SCORM 2004 compliant. Here we would like to integrate the ADL reference implementation into our LMS System2Teach.

4 Overview eCampus Framework

The learning module production takes place in two phases (see Figure 1):

- the *structural* annotation of the textual content
- the *automatic* generation of the eLearning unit

In phase one, most of the (textual) content is created in a standard word processor (Open Office, [10]). Here the text is structurally annotated using a predefined formatting template. The text is stored and automatically converted into a simple XML file, which provides the basis for subsequent transformation steps. This approach works very well for static content and is used in a number of systems (e.g. eLAIX (see *http://www.boldt-media.de/*)).

In the second phase, the file prepared in phase one is transformed into a SCORM 2004 compliant eLearning module automatically. For this, the document is unpacked and analyzed. To get by on cascaded XSL transformations (EXtensible Stylesheet

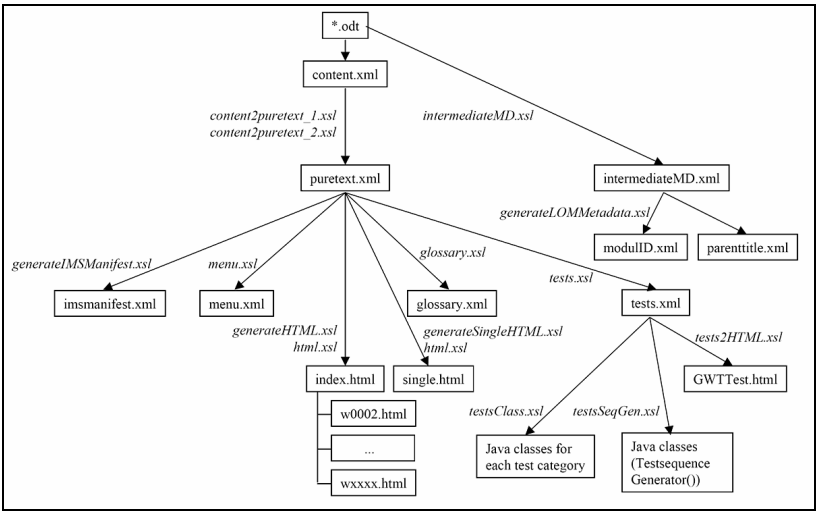

Fig. 2. Transformation Tree of eCampus Framework

Language) each element is identified and transformed with specific instructions (see Figure 2). Finally all contents are packed into an eLearning module and multiplexed with the necessary auxiliary files.

The available metadata is separated and attached to the module in a LOM (Learning Objects Metadata) compliant form.

The complete process is controlled by the Apache ant build tool and uses saxon8 (see *http://saxon.sourceforge.net/*) as a transformation engine ([4]).

5 Structure and Hierarchy

ECampus *modules* consist of *sessions* that are divided into *sections*. This hierarchy roughly corresponds to a *course*, that is organized as a sequence of *lectures*, each of them addressing a set of *topics*.

At a low level the s*ection* is implemented. The section is the least self contained element within an eLearning module. Sections can be arbitrarily nested. Therefore theoretically more than three levels are possible. The development of new sessions is possible by combining different existing sections. New modules arise by combining different sessions. Thus the hierarchical structure contributes to re-use of learning objects.

6 Installation of the eCampus Framework

By running the installation file the framework is installed on the machine. Hereby the necessary programs and styles are copied onto a local disk drive. Furthermore the java development kit is installed if not yet available. Additionally the environment variables are adapted and the path variable enhanced. After installation all parts of the eCampus framework needed are in a user definable folder. Similarly the user guide and some examples can also be found there.

7 Process of Content Compilation

After starting the writer application from the open source office suite ‚Open Office', executing macros must be allowed. The new macros are especially needed for collecting the metadata.

The process of content compilation begins with the creation of a new document by using the eCampus template. All content and metadata can now be collected in this new document.

The menu entry for creating the metadata for the whole module is first chosen. Then a form mask opens in which the necessary metadata is requested. If a choice is to be made possible a list box is shown with possible entries. Fixed data is either superimposed or compiled in the document in the background. This top hierarchy level entry is allowed only once per module.

After this the first session can be generated by choosing the appropriate menu entry. After filling in the metadata form it can create the sections. Once the metadata

is acquired the author is able to start writing the content. The author can add further sections to structure the lesson. For more substantial contexts further sessions are also possible. The pure content can be written into the document directly or can be integrated by copy and paste from existing documents.

All common media types can be used for content. Beside text these can be images, pictures, tables, lists, definitions, links, glossary entries or arbitrary references to external media files.

The most important task is the subsequent annotation task. Hereby the content is annotated using predefined style templates. Depending on the specific element the transformation can identify text, pictures, headlines, etc. There are therefore more than 20 styles available. These styles are stored and linked to the content, so the transformation recognizes each specific element and can transform it.

By saving the document the process of content collection is completed. The document is now ready for transformation.

8 Process of Transformation

The complete process is controlled by the Apache ant build tool and uses saxon8 (see *http://saxon.sourceforge.net/*) as a transformation engine ([4]) supported by a graphical user interface (GUI).

After starting transformation the written Open Office document is unpacked. After that the file ‘content.xml’ which contains the content is accessible and can now be transformed (see Figure 3).

The first step of the transformation is to transform this content.xml into a new file called ‘puretext.xml’. This new file is in an intermediate format. Thereby the annotated elements are assigned an explicit format template. Furthermore all elements without annotation are deleted. The result of this transformation is a universal intermediate format without direct correlation to the producer application.

The transformation is controlled by the XSLT stylesheets, content2pure_1.xsl’ and, content2pure.xsl’.

The advantage of this intermediate format is that the resulting transformations are totally independent of the content collecting process or application. As a result a content collecting process using other applications is also possible in the future, e.g. Microsoft Word. Only the transformation into the intermediate format has to be adapted.

This file in intermediate format can be transformed into the output format (here HTML) by applying the XSLT stylesheets ‚generateHTML.xsl’ and ‚html.xsl’

For this purpose each element from ‚puretext.xml’ is analyzed and the corresponding transformation instruction is searched. If the correct instruction is found, it is applied to the specific element. The transformation result is stored into the HTML file.

To print out the content easily a further version is generated. In contrast to the first form all learning objects, sessions and sections, are stored within one page. This page can easily be printed by using the print functionality of the browser.

```
<text:p text:style-name="EC_5f_Page" />
<text:p text:style-name="EC_5f_Text">Food law, as with all law, is
  enacted by states. Today the world consists of about 200
  independent states. This leads to more than 200 food law systems,
  which follow different trade theories.</text:p>
<text:p text:style-name="EC_5f_Text">
  <text:span text:style-name="T1">Centuries ago, the absolutistic
    states created a mercantile system. They cared only for the
    advantages of domestic production and used deep government
    interventions and control over the economy. Mercantilism is not
    the leading economic theory anymore. The liberalisation during
    the 19</text:span>
  <text:span text:style-name="T2">th</text:span>
  <text:span text:style-name="T1">century brought more and more
    freedom to private ventures. The states restricted themselves
    to setting general rules (e.g. contract rules) and controlling the
    conditions for foreign products (e.g. via customs and standards
    for imported goods). In times of protectionism, these rules
    were strict to give maximum room and better possibilities for
    the state owned products. Socialistic state law brought about
    similar results. At present, socialism and open protectionism is
    reduced to only a handful of states. Neoliberalism and the free
    trade of goods has become the leading economic theory in all
    continents. Economists believe in the power of worldwide
    liberalised markets.</text:span>
</text:p>
```

Fig. 3. Abstract of the unpacked *.odt

An overall glossary is generated by applying the stylesheet, glossary.xsl'. Hereby all elements which are annotated as glossary entries are transformed into a separate glossary file.

To use the learning module without an LMS there must be extra navigation. In this case all headlines from sections and sessions are transformed into the file 'menu.xml' supported by 'menu.xsl'

The metadata which is stored within the ‚content.xml' is read and transformed into a further intermediate format, intermediateMD.xml'. From this the , generateLOMMetadta.xsl' generates the ‚modulID.xml' and parenttitle.xml' which conforms to the LOM standard. Subsequently the complete content and all metadata are transformed into XML files.

9 Process of Test Transformation

Creating the static content of the eLearning module has become relatively simple. A much harder problem is the XML-based generation of dynamic tests. Here online forms have to be created, JavaScript code is needed for data pre-processing and data submission to the learning management system. The tests should work offline and online and the student should not be able to manipulate the results or to cheat by looking at the source code of the online test. Finally, the tests should run on as many platforms as possible.

In order to meet these requirements, we chose to use the GWT (Google Web Toolkit, see [5]), a new free software technology provided by Google for creating AJAX-based online applications. It centres around a powerful Java to JavaScript cross compiler. Using this approach the programmer, instead of coding in JavaScript directly, implements the online application in Java, compiles it and embeds the dynamic components into XHTML host pages. The GWT automatically generates

JavaScript code for the four most prominent web browsers. The generated code is obfuscated by the compiler, thus hiding the internal structure of the online application.

With this technology we have implemented a widget library framework for online tests. The widgets provided are flexible, configurable and easy to use UI components for multiple choice tests, fill in forms and free form questionnaires. Special forms have been designed for tests, where a set of given answers has to be ordered by the student dynamically. Multimedia elements, e.g. images or video sequences, may be integrated into all forms. Additional widgets for online help, user authentication and user guidance have been developed and can be combined with the test forms. The widgets are placed in XHTML host pages, where anchor elements define their positions (an example is shown in figure 3). Cascading style sheets are used to specify the visual appearance and the layout of the widgets. As a result the user interface design of the online tests is not fixed and can be adapted to user specific requirements.

In order to create SCORM 2004 compliant online tests, we need to be able to submit the results to the server. The ADL reference implementation of SCORM 2004 provides a simple JavaScript API for data persistence. When loaded, the eLearning unit registers itself with the LMS and initializes the API. In our framework, we used the JavaScript native interface (JSNI) of the GWT to connect to this API. This interface makes it possible to call arbitrary external JavaScript code from the generated application. The form data can be serialized into JSON (see [1]) and XML (see [11]). We also provide functions to encrypt and decrypt the form data.

10 A Preparatory Module in Microbiology

As an example we would like to give a short overview of one of the modules created using the eCampus approach. Figure 4 shows the module as it is displayed in the ADL reference implementation of SCORM 2004.

For the faculty of Nutritional Sciences a preparatory module for a biochemical practical course was developed. Up to now, a two week course had to be taken by the students introducing them to the biochemical laboratory of the faculty. The course has now been reduced to a single week. Beforehand the students receive a copy of the new module. It explains the laboratory setup and the basic working methods in microbiology. The module's content is based on an existing written handout. The handout was annotated with Open Office. Images were taken and integrated into the text. In addition multi media material was produced. A number of video sequences show the central steps of the working methods (e.g. how to produce a sterile culture medium). Students are requested to conduct a number of statistical analyses. Online simulations have been developed, making it possible, to gain some experience in virtual experiments. In order to control the students learning progress, a set of 100 questions have been annotated and integrated into the module. These questions are automatically transformed into online quizzes.

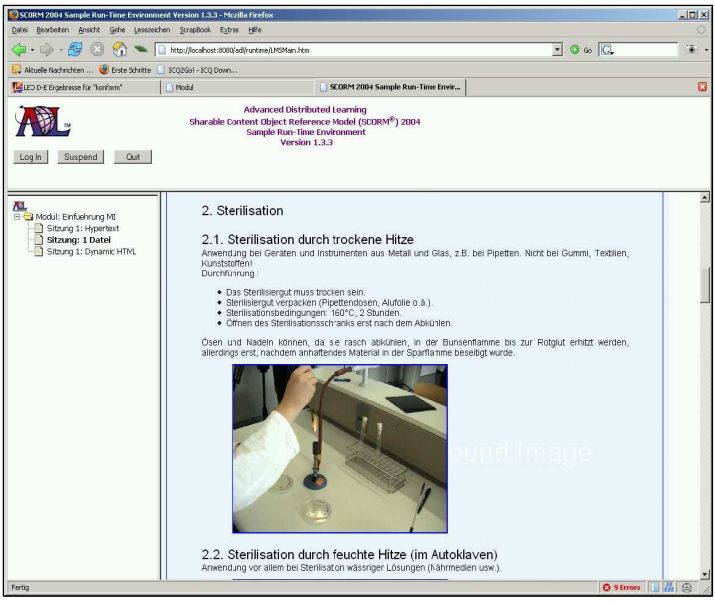

Fig. 4. Example of an eLearning Module within the ADL Reference Server Implementation

11 Advantages and Disadvantages of the eCampus Framework

The advantage of the eCampus framework is the easy production of SCORM 2004 conform eLearning modules.

Most of the learning material is drafted and written with the support of a word processor. With the eCampus framework there is no need of change the working environment. Even for beginners producing eLearning modules, using the templates to annotate the content is simple. All users with experience with the framework confirm this.

A further advantage is the independent intermediated format. Alternative applications for collecting content can be used in the future.

In contrast to most other editors the most important advantage is the conformity to the standard SCORM 2004. Through consequent implementation the eLearning modules can be used within arbitrary LMS which can administrate SCORM 2004 conform eLearning modules. Therefore a high re-use is possible. Also the reorganisation or editing with any standard conform editor is feasible. Especially adding constraints or information about sequencing is possible.

As well as small problems with the robustness of the framework which is due to the early prototype state of the software, the fundamental problem is the lacking route back from the module to the editor. The generated modules can not be transform back to the original document. Necessary editing of the content requires transformation once again.

Small corrections within a section are not problematic. Changes of the structure of the module are not possible because the storage of the state and results of tests can not be kept consistent.

With the eCampus project a new kind of teaching can be offered at Fulda University. ELearning modules can be created efficiently even by non-experts.

Already existing resources can be integrated into the curriculum and will be used in online, offline and blended learning situations. The use of SCORM 2004 compliant eLearning modules allows more complex student to LMS interactions, thus facilitating the design of electronically supported courses of a higher didactic complexity ([9], [7]).

References

1. Crockford, D.: RFC 4627. The application/json Media Type for JavaScript Object Notation (JSON). Online, accessed (17-11-2006) http: //www.json.org/
2. Eisenberg, J.D.: OASIS OpenDocument Essentials. LuLu.com (2004)
3. Hodgins, W., Duval, E., et al.: Draft Standard for Learning Object Metadata. Report, Learning Technology Standards Committee. IEEE, Washington (2002)
4. Holzner, S.: Ant - The Definitive Guide. O'Reilly Media (2005)
5. Google Inc. GWT: the Google Web Toolkit. Online, accessed (17-11-2006) http://code.google.com/webtoolkit/
6. Kay, M.: XSLT 2.0 programmer's reference, Indianapolis, IN. [u.a.] Wiley, Chichester (2004)
7. Koper, R.: Modeling units of study from a pedagogical perspective (2001) http://eml.ou.nl/introduction/articles.html
8. Advanced Distributed Learning. SCORM 2004. Online, accessed (17-11-2006) http://www.adlnet.gov/scorm/index.cfm
9. Schulmeister, R.: Grundlagen hypermedialer Lernsysteme: Theorie - Didaktik - Design. München [u.a.]: Oldenbourg Verlag (2002)
10. OpenOffice.org team. OpenOffice. Online, accessed (17-11-2006) http://www.openoffice.org/
11. W3C. XML. Online, accessed (17-11-2006) http://www.w3.org/MarkUp/
12. Wilde, E., Lowe, D.: XPath, XLink,XPointer, and XML. Addison-Wesley Professional, London (2002)