

XLML: An ILIAS Compatible Authoring Tool

Fred Neumann¹

FIM-NeuesLernen²

University of Erlangen-Nuremberg

XLML³ (eXtended Logical Markup Language) is an open source tool for the creation of eLearning contents. It consists of an author-friendly XML document type and a transforming tool creating a set of HTML pages for offline viewing or import to a learning management system. For ILIAS 2 a patch is available providing an easy upload and seamless use of the imported modules. For ILIAS 3 four migration strategies are discussed.

1 Motivation

From a technical perspective every eLearning project producing own content has to come up with two major decisions: one for the learning management system and one for the authoring tool. If the decision for ILIAS has fallen there are still several ways of getting content into it:

The **built-in editor** of ILIAS is the most common way. It is well documented, allows for a distributed and ongoing maintenance of content without further requirements than a web browser and provides the best integration with ILIAS functionality. However, putting the content together through web-based forms is cumbersome and the copy/paste of prepared HTML code requires an additional upload of pictures and the adaptation of relative hyperlinks. If the content has to be reused outside of ILIAS, an offline version can be created but its code is not the best basis for a further revision.

The **XML based import and export** of learning modules can be used for exchanging editable content, but an authoring tool has to provide a conversion to that format which has gone through changes in the past and seemed to be more appropriate for an exchange between ILIAS installations than for external tools.

The **Import of HTML modules**, a feature introduced in version 2.3 is an easy way to make pre-produced content available for ILIAS. But in contrast to similar functions in other systems like WebCT it just calls the whole module as a unit in an independent frame and can't provide the typical content oriented features like remembering the last visited page or the notes function.

All of these options have their pros and cons. Our own solution is a separate way and a by-product of eL3 (eLearning and eTeaching in initial and further teachers education)⁴, a compound project of the Universities of Erlangen and Oldenburg that is funded by the BMBF⁵ and mainly focussed on teacher training in ICT skills. In eL3 the following requirements have been identified for an authoring tool:

- A considerable amount of learning modules should be produced at both locations, exchanged and adapted to local needs.
- The authors should create the final content directly without post-production by web developers but the content should be web oriented and not just a text conversion.
- Special didactical requirements coming up during the project should be able to be realized.

¹ fred.neumann@fim.uni-erlangen.de

² <http://www.fim.uni-erlangen.de>

³ <http://www.xml.org>

⁴ <http://www.el3.de>

⁵ <http://www.bmbf.de>

- The content should be used in two different systems: The Hyperwave eLearning Suite in Oldenburg and ILIAS open source in Erlangen.
- The content's sources should have a vendor independent format, being open for further conversions.

Fortunately we could build on an earlier development of the University of Oldenburg called Lml (Logical Markup Language). Lml extends classical HTML with structuring tags in a special syntax and a generating tool produces a set of automatically layouted HTML pages. To meet the last requirement, our first effort was to redefine the source format of Lml on the basis of XML and so we created XLML (eXtended Logical Markup Language).

2 The Authoring Tool

XLML is not a editor but an XML document type (DTD). To create the content, a convenient XML editor should be used.

The **XLML DTD** is formulated in the modular framework of XHTML 1.1⁶. This architecture makes it easy to avoid unnecessary elements or to include newer developments. The content of a whole module may be written in one file or distributed to several files. Additional elements provide the special behaviour:

- Structuring in flows, sections and pages
- Definition of a bibliography and glossary
- Specially treated pictures, multimedia and side column elements etc
- Name-based linking with namespaces for links between learning modules

From an authors view the DTD is designed to be easily written by hand with focus on the content than on the layout. However, it is based on familiar HTML elements and not as complex as for example DocBook.

The XSLT and Java based generator tool **XlmlGen** takes an XLML document with all related files and produces a set of HTML pages with a slight but not necessary use of JavaScript. Some of the features are:

- Page layout with optional side columns for texts or icons
- Multiple flows of pages linked by a navigation bar
- Table of contents for chapter, images and other language elements
- Automatically generated tool tips for glossary and bibliography entries
- Automatic image size recognition with layout accommodation
- Video and audio embedding

Many features are configurable by an ini file, e.g. the column layout or the link processing and checking. Normally a module is generated for offline viewing or simple web delivery. However it can also be generated to be used in ILIAS:

- Modifying the relative links to be prepared for the dedicated ILIAS server
- Providing the page and section structure in a CVS file
- Creating a ZIP archive with all the files needed

3 ILIAS Integration

To use the XLML generated content in ILIAS, the system was modified and extended with an import function. The extension was designed to meet the following requirements:

- Easy to use upload, reload and deletion of modules

⁶ <http://www.w3.org/TR/xhtml-modularization>

- Preserving the generated page layout and providing the same look & feel as for the offline version
- Support for all content oriented ILIAS features like searching, recognition of the last visited page, editing of notes and printing of notes and content
- Keeping modifications of existing ILIAS structures as minimal as possible

The final solution uses three new database tables, an import script, an include file, two special course templates and does some smaller changes in six ILIAS scripts.

The **Import script** provides all functions to maintain the XML modules. When a module is imported a project has to be chosen that defines the import defaults (e.g. meta data) and the target web directory. An uploaded zip file is extracted to a folder within the project directory. Now the structure definition is read from the enclosed CSV file and analysed. The according ILIAS structures (table of contents, pages and elements) are created and a table mapping the generated URL parameters to the internal ILIAS IDs is filled. Each page is divided into header, content and footer and the content is stored in the database as a text element to be available for searching and printing.

The **Include file** is used for content related ILIAS scripts. It recognizes the URLs provided by XmlGen, makes a lookup in the mapping table and provides the according parameters to the ILIAS script as global variables. For the presentation of an XML module two templates can be used, one showing the content navigation in an extra frame and one having it combined with the page. In both cases the page content and all related files are taken from the module directory.

The **update** of an existing module is possible as long as the section/page relationship isn't changed by the authors. This is automatically checked when a module is reloaded and ensures that the learners' notes and reading states are kept in sync with the content. If the structure is changed, a module has to be deleted and imported again.

4 Experiences

The development of XML started in autumn 2001 and the ILIAS Integration was firstly tested in Summer 2002 with the eL3 pilot course. Since then we used it in three course cycles with content for around 100 learning hours adapted to 7 school subjects. A handful of other eLearning projects are now using the combination of XML and ILIAS, too.

Maybe the most impressive result is that creation of XML content is accepted by our non-technical authors. However that depends on two key factors:

- An initial training should be given and a technical support should be available
- A comfortable validating XML editor should be used

We chose XMetaL as XML editor, which provides a WYSIWIG-like editing mode and an adaptable, user-friendly interface. However, the tool is relatively expensive, so we're looking forward to a free alternative in the future, maybe based on Open Office.

Another success factor is that XML can be used by the authors as an offline tool without connection to a content management system or an XML repository. At every time a module can be generated in a few seconds giving exactly the output that will be seen in ILIAS afterwards. At last we decided not to claim meta data from our authors.

The ILIAS Integration has been developed so far that it can be used without problems if some basic rules are followed. Currently only administrators have access to the import function and the chapter and page editor of ILIAS shouldn't be used for an XML module. From a learner's perspective the only restriction is that cookies have to be activated.

4.1 Current State

Since October 2002 XLML is published as an open source tool at www.xmlml.org. The tool can be downloaded as a binary distribution for authors or by anonymous CVS access for interested developers. The development of XLML is going on - our current activities include the conversion from and to Open Office and creation of a SCORM compliant content packages.

The ILIAS integration is currently provided as patch for ILIAS 2.2.2 to 2.2.8. Further versions of ILIAS 2 may include the XLML support as an optional module but for that some corrections should be considered, especially for switching on/off the XLML support and for preventing the ILIAS editor from being used with imported modules.

5 Forecast: ILIAS 3

The combination of XLML and ILIAS 2 has been proved as a stable basis for our content creation and delivery. But ILIAS 2 comes to the end of its life cycle and the question is how to provide the same for ILIAS 3 in the future. There are four possible ways:

1. Upgrading the current solution

This should give the quickest results because the issues are well known and solved for ILIAS 2 that has lesser consistency than the new version. However, many parts of the adaptations would have to be rewritten and again a special solution would be created that must be maintained in the future.

2. Conversion to the ILIAS3 import format

This seems to be the most straightforward solution if the DTD for ILIAS 3 has reached a stable stadium. The conversion could be done from the source files or from a later stadium of XmlGen's processing when the final layout has already been produced. The question is whether the content should be editable afterwards in ILIAS or how the layout can be completely preserved. Problems concerning the mapping of XLML features to the DTD structure and the handling of our content in ILIAS 3 can't be foreseen at the moment.

3. Import as a SCORM package

Currently we have an alpha solution to create SCORM content packages by XmlGen. We still experiment on how to map our structure and navigation concept to the content objects and assets required by SCORM. We also have to experiment with the SCORM support of ILIAS 3 on how it recognizes our packages and how it presents the content to the learners. Maybe the standard will show limitations compared to a proprietary XLML support.

4. Using HTML import

An HTML import like the one provided for ILIAS 2 would be a reasonable feature not only for the inclusion of XLML modules but for every existing web based content. The table of contents may be delivered in an additional XML file or manually created from the imported page list.

The core problem for that solution is how the imported pages link each other in a way the platform can recognize and control. XmlGen currently uses templates to create ILIAS-prepared links for that purpose but that's not a common solution. Parsing the pages and changing all URLs at import is tricky and may fail with JavaScript.

The easiest solution would require a URL rewriting mechanism, which is used by many content management systems to get their content indexed by search engines. In our case a relative link from one content page to another can be rewritten to an ILIAS script that checks the access rights, does the necessary logging and delivers the desired file. The files even don't need to be in a physical directory of the web server's document root. The disadvantage of the solution is that ILIAS has to rely on a well-configured web server module: `mod_rewrite` for Apache or a third-party product like ISAPI Rewrite for the IIS.