

Editing IMS Learning Design Explorative survey of editors

Citation for published version (APA):

Brouns, F. (2003). *Editing IMS Learning Design Explorative survey of editors*.

Document status and date:

Published: 14/02/2003

Document Version:

Peer reviewed version

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

<https://www.ou.nl/taverne-agreement>

Take down policy

If you believe that this document breaches copyright please contact us at:

pure-support@ou.nl

providing details and we will investigate your claim.

Downloaded from <https://research.ou.nl/> on date: 05 Oct. 2024

Open Universiteit
www.ou.nl



Educational Technology Expertise Centre OTEC
Open University of the Netherlands

Editing IMS Learning Design

Explorative survey of editors

COLOPHON

Title: Editing IMS Learning Design
Explorative survey of editors

Author(s): Francis Brouns

Project: WP 2, Development Programme

Project manager: ing H. Vogten

Project members: E.J. van den Berg
dr.ir. F.M.R. Brouns
drs. H.J.H. Hermans
H.G.H Martens
ir. G.W. van der Vegt
ing. M. Verhooren
ing. H.F. Vogten

Programme: Research and Development into
Learning Technologies

Programme chair: Prof.dr. E.J.R. Koper

Programme assistant: Mieke Haemers

Report type: Report

Publication date: 14 February 2003

Document Reference: U2003-3299 MMO

Distribution: OTEC

Educational Technology Expertise Centre (OTEC)
Open University of the Netherlands

**Editing IMS Learning Design
Explorative survey of editors**

OTEC report series

The Open University of the Netherlands develops higher distance education and is a central partner in a consortium for the renewal of higher education. Educational technological innovation is one of the main fields of interest. The educational and educational technological expertise of the Open University of the Netherlands are bundled in the Educational technology expertise centre (OTEC). This centre is involved with tasks concerning the development, innovation, research and evaluation of the Open University and its consortium partners. These tasks are performed in close collaboration with directorates and faculties of the Open University and/or its consortium partners.

Otec publishes a report series. This report is part of that series.

Development Programme reports and some documents can be obtained from:

Open University of the Netherlands
Secretary Development Programme P.O. Box 2960
6401 DL Heerlen
the Netherlands
Tel. +31 45 5762624
Fax. +31 45 5762800

Internet: <http://www.ou.nl/otec>

© 2003, 14 February
Educational Technology Expertise Centre,
Open University of the Netherlands

Save exceptions stated by the law no part of this publication may be reproduced in any form, by print, photoprint, microfilm or other means, included a complete or partial transcription, without the prior written permission of the publisher.

Table of contents

1.	Introduction	7
2.	Terminology and conventions	7
3.	Aims and constraints	8
3.1	Namespaces	8
3.2	Valid or well-formed	11
4.	Choice of editors	11
5.	Editing a learning design	12
5.1	Framemaker+SGML5.5.6 as IMS LD editor	12
5.2	XMetaL as IMS LD editor.....	14
5.3	XML Spy as IMS LD editor.....	15
5.4	TurboXML as IMS LD editor	18
6.	Editing imslcontent	18
6.1	Framemaker+SGML5.5.6 as editor of imslcontent	19
6.2	XMetaL as editor of imslcontent	19
6.3	XML Spy as editor of imslcontent.....	19
6.4	TurboXML as editor of imslcontent	20
7.	Imslcontent with schema	20
8.	Findings	21
9.	Conclusions	22
10.	Glossary	22
11.	References	22

1. Introduction

Learning Design is a specification for a semantic notation for education (units of learning) developed by the IMS Working Group on Learning Design. It can be used to describe didactical models and related processes. The Open Universiteit plays a major role in this IMS Working Group, as they provided EML as basis for the new specification. There were several reasons to start looking at XML editors as tool for creating and editing IMS Learning Design.

The members of the Development Programme of OTEC, Open Universiteit, needed to get acquainted with the IMS Learning Design, test the binding and create files. The team indicated that a text editor would not suffice and that a more sophisticated editor would be needed.

The main implementation of IMS Learning Design is an XML (eXtensible Markup Language) application in the form of a XML schema implementing many new XML constructs.

This document describes a short survey of available XML editors in their ability to create and edit IMS Learning Design files. The editor was intended solely for use by OU staff sufficiently experienced in XML. The assumptions were that no changes were made to the IMS Learning Design and that the editor would be used 'off-the-shelf', so without modification. However the suitability in an authoring environment was also considered.

Nor does this report specify any requirements for an authoring environment. When IMS Learning Design is used in a production environment where authors and academic staff are using it to create educational material, a much more complex and sophisticated authoring environment might be needed. Another project was foreseen to specify requirements for such an environment.

2. Terminology and conventions

IMS Learning Design is used to specify an instructional design. No binding, except the global elements, is provided for content. Content (i.e. actual text presented to users) is provided in resources. IMS Learning Design provides a mechanism to link the resources to the design. These resources can consist of so-called webcontent, i.e. those contents that can be rendered or handled by a web-browser, including binary documents, like Word documents etc. The other type of resource is the so-called imslcontent. Imsldcontent consist of a XHTML document combined with the IMS LD global elements. XHTML is HTML in XML notation.

In the remainder of this document the term IMS Learning Design (IMS LD) is used in several contexts. It can refer to the binding (XML Schema) or to documents conforming to the IMS LD binding. It is also used as synonym for learning design, unless explicitly stated otherwise. The context should be sufficient to determine the meaning.

The term IMS LD is never used for content; `imsldcontent` is used to refer to content in resources in which the global elements are used.

Tagnames are shown in italic between angular brackets, like `<learning-design>`. Attribute names are in italic. Examples are provided in Courier typeface, like `xsi:schemaLocation`.

3. Aims and constraints

The editor would serve multiple purposes. It could be used to create IMS LD and `imsldcontent` files to familiarise project members with the new specification. However, the survey was also intended to determine whether the editors were suitable for producing fully XML and IMS LD compliant files, which did not need any processing afterwards.

At the time of the survey the IMS LD schemas were not finalised yet, but a XML DTD existed. The assumption was made that it should be possible to create IMS LD and `imsldcontent` by using this DTD. It would be acceptable to produce an adapted version of the DTD as long as it entailed minor changes. This assumption proved wrong, as explained later.

It should also be possible to use the editor for creation of IMS LD without resorting to extensive adaptation of the editor, except for the normal configuration needed, like compiling a DTD or creating a basic template.

3.1 Namespaces

IMS LD already contains placeholders for and is set up to allow substitution of elements by other schemas. Because of this, it becomes important to be able to distinguish the schemas to which certain elements belong. It is even more important when there is a naming conflict. XML provides a mechanism for this: namespaces. Use of namespaces enables distinction of elements, prevents naming conflicts, enables context dependent processing etc.

A XML namespace is a collection of names in XML documents used as elements and attributes. The namespace is identified by an URI reference. The W3C recommendation on namespaces provides a method to associate elements and attributes with the namespace.

Namespaces are declared by the use of reserved attributes. Declaring the namespace by the use of the `xmlns` attribute is also referred to as default namespacing.

Example of default namespacing

```
<product xmlns="http://www.example.org/product">
  <table>
    <model>...</model>
  </table>
</product>
```

In this example the namespace has been declared on `<product>` and thus applies to all elements within `<product>`.

Another mechanism is to declare a namespace prefix (using *xmlns:*) and prefixing all relevant elements. The prefix can be declared at the root element of the XML document, or at the relevant element. All elements in the namespace need to receive the prefix. Advantage of declaring the prefix at the root element is that the prefix needs to be declared only once.

```
<prod:product xmlns:prod="http://www.example.org/product">
  <prod:table>
    <prod:model>...</prod:model>
  </prod:table>
</prod:product>
```

The namespace applies to the element in which the namespace is declared, and all its descendants (inherits down/inwards), until overridden. Multiple namespaces are allowed.

```
<product xmlns="http://www.example.org/product">
  <table>
    <model>
      <html:p xmlns:html="http://www.w3.org/1999/xhtml">A square ...
    </html:p>
    </model>
  </table>
</product>
```

In this way, the namespace prefix needs to be declared at every instance of an html element. The html namespace prefix could have been declared on the root element instead of at the relevant element. Then the namespace has to be declared only once and can be used on all relevant HTML elements.

Attributes are not necessarily in the namespace that applies to the elements, unless explicitly placed in the namespace. The authoring process becomes rather complex when attributes are placed in the namespace, because those attributes need to be prefixed. No default namespacing is possible with attributes.

Example where the elements are but none of the attributes is placed in the namespace:

```
<product xmlns="http://www.example.org/product" id="p1" name="oak">
  <table>
    <model>
      <html:p xmlns:html="http://www.w3.org/1999/xhtml">A square ...
    </html:p>
    </model>
  </table>
</product>
```

Example where the attribute 'id' is placed in the namespace, but the 'name' attribute not:

```
<product xmlns="http://www.example.org/product"
xmlns:prod="http://www.example.org" prod:id="p1" name="oak">
  <table>
    <model>
      <html:p xmlns:html="http://www.w3.org/1999/xhtml">A square ...
    </html:p>
```

```

    </model>
  </table>
</product>

```

Namespaces can be used in a DTD, but this is rather complicated. It is even more complex to combine different namespace in a DTD. According to W3C specs (Modularization of XHTML) it is possible when designing a DTD in a modular way. This is a rather complex method, for which no tooling was available in the project. That would make the creation and maintenance of the DTD very cumbersome. The DTD editor Near&Far Designer in use by the project is not suited for this. Whether XML editors are able to parse such modular DTDs was unknown. Therefore this approach was not taken. Instead an alternative was tried.

Namespaces in a DTD can be indicated by the use of the *xmlns* attribute, or by naming the element including the prefix. This prefix then is not seen as the namespace prefix, but as part of the XML name. However this can result in an unnecessary complex authoring environment, depending on the editor used. Editors that are not namespace aware will allow the use of the *xmlns* attribute as normal attribute, depending on the author to provide the value for the attribute. Namespace aware editors however, will handle *xmlns* attributes in a special manner; often preventing it's use as 'normal' attribute.

Additional problems arise with default namespacing for a schema combining multiple namespaces, because the namespace inherits onto descendants until overridden. This means that default namespacing (in a DTD) can only be used when the other vocabulary has provided the *xmlns* attribute for all elements (this is not the case for XHTML). Several parsers (like MSXML) do not allow default namespacing for documents based on a DTD.

For testing purposes, a DTD for *imsldcontent* was created, which used prefixes for the IMS LD global elements; addition of the *xmlns* attribute to all elements was seen as a too large deviation from the IMS LD specification. The document instance requires a default namespace declaration for the XHTML elements and a prefix declaration for the prefixed IMS LD elements, and requires the prefix for all relevant elements.

```

<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:imsld="http://www.imslobal.org/xsd/imsld_v1_p0">
  <head>
    <title></title>
  </head>
  <body>
    <p>Some text here</p>
    <imsld:view-property/>
    <p>Some more lines </p>
  </body>
</html>

```

Because IMS LD contains elements from other namespaces, and even elements which in turn can contain elements from other namespaces, there must be a mechanism to reset or override the namespace for descendants. The DTD does not provide this, unless the *xmlns* attribute is added to all elements in all namespaces, or prefixes are used for all namespaces.

Not all editors are 'namespace aware'. Even namespace aware editors often still depend on the author to set the namespace declaration on the relevant elements.

The editor should be able to handle documents with prefixed elements, default namespacing and multiple nested namespaces.

3.2 Valid or well-formed

A learning design should be a valid IMS LD and a valid XML document. A valid XML document should conform to a DTD or a XML schema. A valid XML document usually contains a public or system identifier in a doctype declaration (`<!DOCTYPE html PUBLIC "-//OUNL//DTD XHTML imslldcontent//EN" "imslldcontent.dtd"]>`) when using a DTD or a *XMLSchema-instance* plus *schemaLocation* processing instruction (`<html xmlns="http://www.w3.org/1999/xhtml" xmlns:imslld="http://www.imslobal.org/xsd/imslld_v1_p0">`) when validating against a schema.

Documents presenting imslldcontent should at least be well formed. However, the authoring process is much easier when the document is based on a DTD or schema, because it guides the author in constructing the IMS LD global elements and also the XHTML document. So, also for imslldcontent the test was conducted with a DTD or schema, in order to create well-formed and valid documents.

The use of both the doctype declaration and the schemalocation attributes is not advisable. The doctype declaration can only point to a DTD, while the schemalocation points to a schema. This might confuse implementations.

Most editors will provide an absolute system path or an absolute file URI. This presents interoperability problems, complicating re-use. It also presents problems with those validating parsers that expect a correct URI. The XML schema standard specifies that a URI should be used to point to the schema location, not a file path.

4. Choice of editors

This survey was not intended as a full-scale investigation of all available editors. A few criteria were taken into account when selecting editors. The editor should support namespaces, preferably support XML schema, provide a WYSIWYG interface. Possibilities to adapt and configure the editor via scripting and APIs were considered an advantage, although for this survey the editor was used 'off-the-shelf'. A demonstration version of the editor should be available, because there was no funding for this survey.

A quick Internet search was conducted, and several editors were downloaded. Some of them were rejected, because either the interface was too crude, or they failed to read the IMS LD DTD.

The authoring environment currently in use by the Open Universiteit Nederland for creation of EML 1.0 is based on an SGML editor, Framemaker+SGML 5.5.6. A first attempt in configuring the current authoring environment for IMS Learning Design indicated that Framemaker+SGML 5.5.6 would not suffice and that XML editors would make a better choice.

Some experience with XMetaL was gained in a previous test and therefore XMetaL was included in the test.

Two new editors were added: XML Spy and TurboXML. TurboXML was added at a later stage and was only tested for imslcontent.

Most available editors are intended for creation of documents and are designed for experienced users or users wanting to learn or work with XML. These kinds of users are familiar with XML concepts. The target audience for IMS LD is not likely to be familiar with those concepts, nor would there be any need to. For these authors it would be important to use an editor that would allow the user to work in a manner they are used to. A WYSIWYG interface and minimal training are among the required features. These aspects were taken into account while evaluating the editors, but were not leading. It was less important for the current survey, because it was intended for experienced users familiar to XML.

5. Editing a learning design

Two problems existed with the current authoring environment of the Open Universiteit: it works only with a DTD and it can only import SGML no XML, although XML export is possible.

The first problem, use of a DTD, is no real problem. The document instance remains the same when creating SGML or XML except for the first few lines of the document, which optionally contains a XML declaration. Most editors insert the XML declaration (`<?xml version="1.0" ?>`) when creating XML files; some only provide a doctype declaration. This is allowed because the XML declaration is not mandatory.

The main problem is the addition of the *xmlns:xsi* and *xsi:schemaLocation* attributes to the root element. These attributes are not present in the DTD. They either need to be added to the DTD, or to the internal subset in the document instance. In either case, this presents interoperability problems. In some editors this might even result in conflicts, as they will recognise the xsi parameters, but not in combination with a DTD. No tests have been done where the attributes have been added either to the DTD or to the internal subset. This is rather complicated, because most editors do not recognise the prefixed attribute name as valid.

Other problems lie in the *xmlns* attributes. These attributes are present in the DTD as #FIXED or read-only attributes. Most XML editors will not insert these attributes into the document instance, unless the author manually inserts the attribute (actually resulting in default namespacing for the relevant elements). Several editors/parsers fail to validate document instances that contained both the namespace declaration at the root and this default namespace declaration at respective elements.

5.1 Framemaker+SGML5.5.6 as IMS LD editor

Framemaker+SGML 5.5.6 is a product of Adobe (<http://www.adobe.com>). It is a product mainly used by technical writers to produce and format complex documents, both structured (SGML) and unstructured.

Advantages

Framemaker+SGML5.5.6 (referred to as FM) is currently in use by the Open Universiteit.

It is a multi-platform application.

FM has a graphical WYSIWYG interface, with extensive formatting capabilities.

Authors and developers of the Open Universiteit are familiar with the product.

Adaptation of the product is possible in several ways; some require programming via an API, others rely on creation of templates, or configuration files.

Support is provided by Adobe, but the large developer and user community provides more effective and efficient support.

Disadvantages

FM is an SGML editor, but is capable of exporting to XML.

Although FM can export to XML, FM can not import XML files, so round tripping was not possible.

Another major problem is that Framemaker+SGML5.5.6 does not validate the document when exporting to XML. FM provides a validation mechanism for the document while in FM format. This validation is not as strict as needed for SGML (or XML). For example, FM allows a less strict format for the value of ID attributes than is allowed in SGML. A second, stricter SGML validation is applied when exporting to SGML. Because the second validation is not applied when exporting to XML, this might result in invalid IMS LD files.

FM creates the XML declaration, but omits the doctype declaration (because it creates only well formed XML) and omits the xsi parameters. This is a disadvantage, because the document can no longer be validated automatically. These documents might require additional processing after FM to promote interoperability.

Because FM creates only well-formed XML documents, it will insert all attributes with default or fixed value into the document instances. This results in elements with the *xmlns* attribute present in the instance, i.e. a default namespace declaration. (Most other editors will omit these attributes, because they assume the presence of the DTD). Some validators, like MSXML, which is based on a schema, flag this as errors, because the *xmlns* attribute is not allowed on descendants. This is probably a bug in the validator.

Finally, FM creates a XML stylesheet processing instruction and the corresponding CSS stylesheet. This might impose problems for IMS LD players/implementations. So, this XML stylesheet processing instruction (PI) should be removed. As FM does not provide a XML API, documents have to be processed by an external application to remove the stylesheet PI.

Documents can only be created and edited as SGML documents. Therefore the use of prefixes is prohibited; the colon is not allowed in SGML element-names.

As FM does not import XML documents, a much stricter version control and management procedures need to be in place.

Another option was explored, in which FM was only used to create SGML files. These were then converted to XML using either XMetaL version 2.1 or 3.0. Both editors however, failed to properly read the SGML documents created by FM, due to bugs in the XMetaL implementation.

Conclusion

In short, although FM produces documents that are IMS LD valid, the resulting XML files need post-processing to insert proper XML processing instructions. Framemaker supplies a standard API and an SGML API, but not for XML events. Any post-processing therefore need to occur outside Framemaker, using additional tooling.

5.2 XMetaL as IMS LD editor

Corel recently acquired XMetaL (<http://www.xmetal.com>) (originally from Softquad).

XMetaL is a SGML/XML editor that comes in several flavours. Functionality partly depends on operating system used (Win9x vs. NT, Win2000).

Two versions have been evaluated. XMetaL 2.1 can edit documents based on a DTD. XMetaL 3 should also support XML Schema.

Advantages

According to the specifications, XMetaL provides many scripting possibilities. There was little prior experience with XMetaL. Adaptations and configuration would require time to explore these possibilities.

XMetaL can be used without any prior configuration by advanced authors. It then relies on the author to indicate the DTD to base the document on. XMetaL then automatically compiles this DTD in a proprietary format. It is also possible to distribute only the compiled DTD. This prevents changes to the DTD. However, XMetaL is rather unstable, even with compiled DTDs, in that it does not always read the compiled DTD properly.

XMetaL provides multiple views of a XML document, amongst which a graphical WYSIWYG interface. This graphical interface can be configured via XMetaL specific instructions in combination with CSS stylesheet instructions. This interface is not very intuitive.

Disadvantages

Although XMetaL should support SGML, the problems are such that XMetaL for SGML is not advisable.

XMetaL is available for Windows only. Unicode support depends on Windows version.

When XMetaL is not configured, authors need to indicate the DTD on which the document must be based. This assumes that authors are familiar with XML concepts, which usually is not the case.

The graphical interface is hard to navigate and not intuitive.

It is very hard to navigate in documents and edit without using the mouse. This is a disadvantage, because it increases rsi problems.

XMetaL generates a doctype declaration with a system identifier that is an absolute file path. When a template is provided, a relative path can be specified, but it remains a file path, instead of an URI. This restricts interoperability.

Implied attributes with default or fixed value are not entered into the document instance, unless explicitly entered by the author. This applies to the *xmlns* attribute as well, thus relying on the author to insert this attribute at the proper places to declare the correct namespace.

It is not possible to indicate the required order of elements in complex content models. The author needs extensive knowledge of the DTD.

XMetaL 2.1 is not namespace aware.

XMetaL 3 supports XML Schema partly. It does not support import of other namespaces. Any schema for IMS LD requires import of other namespaces, in particular for *imsldcontent*.

XMetaL is unstable and seems to 'forget' configurations.

XMetaL is unable to open a document instance where the required additional schemalocation and namespace attributes have been added to the root element.

XMetaL 3 can not add the namespace declaration to the root element, because the generated schema does not provide a target namespace, or a default namespace. The schema would need adaptation.

Support is available during the first 30 days after purchase. Additional support can be purchased. Since the survey, XMetaL has been taken over by Corel. No information about quality of support is gathered.

Conclusion

Maybe it would be possible to configure XMetaL via scripting so that it could be used to create IMS LD documents. Due to its instability, the first impression is against XMetaL.

5.3 XML Spy as IMS LD editor

XML Spy is a product of Altova (<http://www.altova.com>). The test was conducted with XML Spy 4.3 IDE.

The XML Spy 4 IDE is intended for developers. It is more than an XML editor, providing much functionality like DTD and schema creation, and XSLT.

There is also a document editor, intended for authors. This client however needs to be configured extensively, requiring XSLT scripts to be developed in the IDE, before the client can be used, and is available as plug-in only. This client was not evaluated.

Since the test XML Spy 4.4 and XML Spy 5 has been released. These new releases contained several bug fixes, but are still unable to provide solutions for the problems encountered. In particular the latest release seemed to re-introduce previous bugs and introduce new ones.

The new release also contained a separate document editor. This document editor still relies on extensive configuration and XSLT scripting, which needs to be created in another application.

Advantages

According to the specification, XML Spy can be scripted and adapted.

XML Spy can be used to create XML documents without configuration, when the author indicates the DTD or schema to be used. This is an advantage for experienced authors, but not for the intended target audience of IMS LD users.

Response of support, at least during the first 30 days of the license, is quick. Resolution of problems might not be as quick.

Disadvantages

There was no prior experience with XML Spy, so time had to be spent to explore the application.

XML Spy is a suite combining much functionality, targeted mainly at developers. This seemed to compromise quality and stability.

XML Spy is a XML editor, not able to read SGML files.

XML Spy is only available for the Windows platform.

XML Spy IDE does not have a WYSIWYG interface. A WYSIWYG interface can be created for the document editor (separate tool) by creating a XSLT template. This is rather cumbersome because IMS LD is very complex and not document-based.

Document instances can be created in XML Spy IDE in the 'enhanced grid view'. This is a tabular view. This view is very limited, because the fields to enter elements and text are limited in size. Navigation in this grid view is not intuitive and relies heavily on use of the mouse.

Editing requires the use of the mouse. It is very hard to impossible to enter text, elements or attributes without the use of the mouse.

The interface is not intuitive. The author need to indicate whether elements/attributes need to be appended, inserted, or are children. This assumes extensive knowledge of the DTD/schema.

XML Spy has a very strict validation, when explicitly calling the validation tool, but not while creating and editing documents. While editing, invalid documents are easily created. XML Spy does not warn the author about this. Errors are reported only when explicitly validating. In addition, XML Spy does not provide a full overview of all validation errors, but stops the validation at the first error. It is not possible to skip some errors and validate the rest of the document. There is no indication whether it is the only validation error, or that other errors are present.

The strict validation prevents the use of XML Spy by educational designers. They knowingly create (temporarily) invalid templates as basis for authors. It is not possible for them to check the remainder of the template.

Implied attributes with fixed or default values are not entered into the document instance.

Required attributes are automatically inserted with empty string "". This empty string remains in the document until the author provides a value. The validation does not always reports validation errors on this, because empty strings are allowed for some attributes.

XML Spy generates a doctype declaration with a system identifier that is an absolute file path. This prevents interoperability.

The elements are not listed in the correct order. Authors need extensive knowledge of the DTD/schema to know which element to insert.

The same problems, inserting the required additional schema attributes to the root element, exist as reported with the other editors, when basing documents on a DTD.

Features of XML Spy seem to vary with the view to such extend that what is functioning in one view is not available or reporting errors in another view.

Support is free during the first 30 days of purchase. After this period, support is available only when a support and maintenance contract has been bought. New release and upgrades are available only under the support and maintenance contract, but not under the regular license.

Conclusion

Again the basic problem was the addition of the schemalocation attributes to the root element.

The other aspects seem to preclude its use for beginning authors and IMS LD users.

5.4 TurboXML as IMS LD editor

TurboXML is a product of Tibco (<http://www.tibco.com>). It is a Java-based multi-platform IDE for XML, providing facilities for creation of XML schemas, XML documents and DTDs.

TurboXML 2.3.1 has been tested.

Advantages

TurboXML provides the user with multiple views. Although the graphical view is not really WYSIWYG, this view is more intuitive than the enhanced grid view of XML Spy.

The graphical view can be customised.

TurboXML provides a good graphical view of DTD and schema files.

Disadvantages

Turbo XML generated a doctype declaration with absolute file URI when assigning a DTD to the instance. This prevents interoperability.

No information about support could be found.

Conclusion

TurboXML was not really tested in the creation of IMS LD documents. The same problems with the schemalocation attributes exist with TurboXML as with the other editors.

6. Editing imslcontent

Actual content for IMS LD can be of two types: either webcontent or imslcontent. Webcontent is all content that can be handled by a web browser, either natively, via plug-in, or via the shell. Webcontent can consist of native web pages, HTML, XHTML or XML, text, or can consist of binary documents, etc.

Imslcontent consists of the combination of the IMS LD global elements with another vocabulary. The most obvious choice for this is XHTML. To create imslcontent at least 2 bindings need to be combined. In a DTD this is difficult, XML schema provides mechanisms for this. However, there is not yet a schema for XHTML.

Several DTDs were created to conduct the tests. The basis was the XHTML strict or transitional DTD. The IMS LD global elements were added to the content models of body, all block-level elements and all inline elements.

There were two variants of the DTDs; one without prefixed element names and one with prefixed element names.

6.1 Framemaker+SGML5.5.6 as editor of imslidcontent

Framemaker could not be used to create imslidcontent with prefixed element names, because Framemaker is an SGML editor. In SGML no colon is allowed in the element name.

The use of default namespacing was not possible without changing the DTD by adding a *xmlns* attribute to all elements, including the XHTML elements. It would also require the author to manually insert the *xmlns* attributes in all elements.

Framemaker 7 was acquired after the test. Framemaker 7 is capable of editing XML documents, based on a DTD. According to the specification, there is no support for XML Schema, but it supports namespaces. Additional tests would be required to investigate its suitability as IMS LD editor.

6.2 XMetaL as editor of imslidcontent

Namespaces had to be declared in the instance. When creating new documents, these either had to be added to the template, or had to be added manually afterwards. Upon editing, XMetaL considered these as attributes not present in the DTD and reported an error. Consequently, the instance could not be validated and only be opened as well formed. For well-formed documents there is no guidance about allowed elements and documents.

When the DTD to be used was indicated (manually), many problems remained. XMetaL failed to indicate all allowable elements: sometimes only the elements already present in the document were shown, sometimes only some of the IMS LD global elements.

XMetaL added a doctype declaration to all documents. This might be a problem for other implementations.

Conclusion

Both versions of XMetaL failed to provide the author with the options to create valid imslidcontent files.

6.3 XML Spy as editor of imslidcontent

XML Spy generated a XML declaration and doctype declaration including an absolute file path when creating new documents based on the DTD. However, it failed to add the *xmlns* attribute to the instance, unless explicitly (manually) added to the element. This is not allowed in combination with prefixed elements; in this case the declaration of the prefix is required, not the default namespace. Default namespacing is also unwanted, because it inherits to descendants.

XML Spy did not add namespace declarations to the document.

XML Spy had many problems with those elements consisting of the ANY content model.

XML Spy could handle nesting the IMS LD global elements in an HTML element, but crashed when multiple nested namespaces are required.

Conclusion

No imslcontent documents, except really simple ones, could be created in XML Spy.

6.4 TurboXML as editor of imslcontent

Advantages

TurboXML has a rather intuitive interface.

Preliminary testing was very promising.

Disadvantages

Problems arose with more complex documents, in particular with those elements consisting of the ANY content model and those elements that required nesting of namespaces.

Nesting of namespaces was not handled properly.

There was no guidance on inserting valid elements when namespaces were nested.

Conclusion

Many problems, mainly related to namespaces remained, to such extent that no valid imslcontent instances could be created.

7. Imslcontent with schema

Whenever namespaces are required XML schema is a better choice than a DTD. Several attempts were conducted to create a schema for imslcontent that allowed creation of valid documents. There is no acknowledged schema for XHTML, but a preliminary version was available. In addition the XHTML Strict DTD was converted to a schema with Turbo XML.

Two editors, capable of creating documents based on schemas were used to test the schemas: XML Spy 4.3 and TurboXML 2.3.1.

The schema needed adaptation to TurboXML because it had specific requirements for the XML namespace.

Many problems remained with these schemas. Both editors (XML Spy and TurboXML) failed to create valid instances. There were problems with the elements consisting of the ANY content model, but also with choice groups. Sometimes the problems were

restricted to the graphical interface. The textual view sometimes allowed elements to be inserted, which could not be inserted in the graphical interface. However the textual view did not offer any guidance on allowed elements.

Both editors would finally crash or hang when creating imslcontent instances.

8. Findings

None of the available editors could be used, either to create IMS LD or imslcontent. Considerable adaptations would have been required, both of the bindings (DTD or schema) and of the tooling.

In particular the use of namespaces introduced many problems. Then there were still many bugs in available tooling that prevented creation of documents in a graphical view. Working in a textual view could circumvent some of these. However the textual views did not offer any author guidance, thus requiring that authors have extensive knowledge of the binding.

Nesting of multiple namespaces caused considerable problems.

Although all editors seemed to have problems with the same issues, in addition to aspects of user-friendliness, no common denominator could be found. Most tools claimed to use their own XML parser, but no details were provided.

Use of a DTD binding, in particular for imslcontent is prohibited, because:

- will create a doctype declaration instead of a XML PI
- will have to add xmlns:xsi attribute declaration to internal DTD subset in document instance. However, most editors can not handle the prefixed attribute name in a DTD.
- will have to create new DTDs, either to add the xmlns attribute to all elements, or to add prefixes to all elements
- use of xmlns attribute will only work in editors, which are not namespace aware. Then it will be an author action to set the attributes, which is not advisable. When this attribute would be fixed, most editors will not write the attribute value into the document instance (which is valid).
- when prefixing all elements, a global namespace declaration is needed in the document instance. The editors do not create this XML PI, instead they create a doctype declaration.

In order to create IMS LD and imslcontent with current tooling a schema needs to be developed that specifies exact models at the placeholders, instead of the many open models currently present in IMS LD.

There are additional problems not raised here that need solving before any of the editors could be applied in an authoring environment. One of these is the extensive cross-referencing mechanism for items and resources. This can not be solved in a simple editor, but requires a sophisticated authoring environment.

9. Conclusions

It soon appeared that none of the major editors were suitable for editing IMS LD and imslldcontent.

Major problems seem to lie in the use of namespaces with DTDs, and nesting of namespaces. Other problems lie in open content models, like the ANY element. For some of these problems workarounds might be used by creating a specific binding and implementing closed schemas for those constructions that are now open.

Even when a valid schema could be constructed, many problems remain with the tooling. All tooling would need adaptation, either via configuration or via scripting to pre-process or post-process the files.

Then, there still remain bugs in the current tooling that probably can not be circumvented by additional programming or changes in the binding.

10. Glossary

design	specifies the instructional design of the unit of learning
DTD	Document Type Definition Declarations laying down vocabulary and syntax, or grammar of documents.
IMS LD	IMS Learning Design A specification to model pedagogies and related processes. http://www.imsglobal.org
HTML	Hyper Text Markup Language
SGML	Standard Generalized Markup Language (SGML) A text format to structure documents. A 'meta' language to describe markup languages. A markup language contains a vocabulary and syntax. ISO Standard 8879: 1986
XHTML	Extensible Hyper Text Markup Language A reformulation of HTML 4.0 in XML.
XML	Extensible Markup Language A simple, very flexible text format derived from SGML (ISO 8879). It provides a universal format for structured documents and data. It is a 'meta' language to describe markup languages.
learning design	root element of IMS LD document, consisting of the design containing the didactical scenario, resources, and metadata

11. References

HTML. <http://www.w3.org/MarkUp/>

IMS Global Learning Consortium, Inc. <http://www.imsglobal.org>

IMS Learning Design Specification.

<http://www.imsglobal.org/learningdesign/index.cfm>

IMS Metadata Specification. <http://www.imsglobal.org/metadata/index.cfm>

IMS Content Packaging Specification.

<http://www.imsglobal.org/content/packaging/index.cfm>

Modularization of XHTML. [http://www.w3.org/TR/2001/REC-xhtml-modularization-](http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/)

[20010410/; http://www.w3.org/TR/2001/REC-xhtml-modularization-](http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/dtd_module_rules.html#s_dtd_module_rules)

[20010410/dtd_module_rules.html#s_dtd_module_rules](http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/dtd_module_rules.html#s_dtd_module_rules)

Namespaces in XML. World Wide Web Consortium 14-January-1999.

<http://www.w3.org/TR/REC-xml-names>

RFC2396. IETF (Internet Engineering Task Force) RFC 2396: Uniform Resource

Identifiers (URI): Generic Syntax, eds. T. Berners-Lee, R. Fielding, L. Masinter.

August 1998.

SGML. [http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=](http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=16387)

[16387](http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=16387)

XHTML. <http://www.w3.org/Markup/#xhtml1>

XML. <http://www.w3.org/XML/>

XML Schema. <http://www.w3.org/XML/Schema>

XML Schema Part 0: Primer W3C Recommendation, 2 May 2001

<http://www.w3.org/TR/xmlschema-0/>

XML Schema Part 1: Structures. W3C Recommendation 2 May 2001.

<http://www.w3.org/TR/xmlschema-1/>

XML Schema Part 2: Datatypes. W3C Recommendation 2 May 2001.

<http://www.w3.org/TR/xmlschema-2/>

