

# e-learning Content Generation Tools: An XML-based Semantic Content Object Approach

---



**Amit Bhutoria**  
*IIT Kharagpur*

---

Research Interests: Adaptive Learning, Intelligent Tutoring Systems, e-Learning, networking, internet applications, e-Governance, smart structures. The author is presently working as an intern for the Centre for Education Technology, IIT Kharagpur and has served as Head of Linux and Web Applications Groups at Innovations Labs, IIT Kharagpur. The author is also currently working on an ontology based adaptive tutoring system under Prof. Sudehsna Sarkar, Computer Science Engineering Department, IIT Kharagpur. The author has successfully completed internships with Centre for Development of Advanced Computing, Hyderabad and ASI (American Supplier Institute -[www.asiusa.com](http://www.asiusa.com)).

---



**Akshay Mishra**  
*IIT Kharagpur*

---

Research interests:  
Working in the areas of real-time computer graphics and e-Learning. Web-team Member for Kshitij 2005. Designer and developer of Cs3D - an open source managed 3D graphics engine hosted at Sourceforge.net  
<<http://Sourceforge.net>>

---

## Abstract

This paper discusses an XML-based semantic object-oriented approach to content development wherein the content can be separated from its presentation logic and can be presented to the end-users in a platform independent way. This method also provides for powerful search capabilities within the content on the basis on meta-data published for individual objects at the time of creation.

## 1. Introduction

Content creation for e-Learning is a tedious and costly process. However, much time and financial resources can be saved if, at the time of creation, the parts of an e-Learning package are made reusable and embedded with metadata. Similarly, a huge amount of costs and inefficiencies encountered in user-specific package-deployment can be avoided if content is organized into reusable semantic-objects.

This paper explores an XML-based content-object document format, which attempts to resolve some of the common issues encountered during content creation and distribution. An XML-based content-object document format has the following features:

- i. A flexible metadata format
- ii. Reusable content-objects with embedded as well as referenced data
- iii. Hierarchical storage format
- iv. Portability for multiple platforms/device resources
- v. Portability according to user profile
- vi. Cost effective content deployment across platforms, minimizing the use of expensive transcoding services (proxies).

The paper also explores some common problems encountered during content-generation, indexing, and deployment and

illustrates the manner in which these are solved by the proposed document model.

## 2. Current Problems in Content-Creation and Deployment

To illustrate the common problems encountered during the process of content-generation, we consider an example.

To create an e-Learning package, an instructor might want to use the huge amount of free data available on the Internet and on other digital information repositories to save time in creating new content. But, since the normal documents available on such repositories contain no machine-readable intrinsic data about their content, search and categorization procedures for content – even when using search engines – prove to be inefficient and inaccurate.

The main cause is the current application base for content-generation – where traditional tools for content generation such as word-processors and graphics-editors do not allow the content-creator to embed organized and flexible content-describing data or “metadata” into the files themselves. Besides, many such tools embed styling and formatting data into the content itself, which leads to the undesired “mixing” of business logic and presentation logic. This, however, is a problem inherent with the document models used today on the Web and within many e-Learning packages.

Since styling data is also mixed with the actual content, it makes it a harder task for search and categorization algorithms to browse through content and index it for future use. Though CSS and XSL have resolved some of the issues for HTML as a document format, their application has not proved itself to be flexible and enough for e-Learning packages as they do not separate data into reusable objects. Also, since the use of XSL and XSLT functions is limited to porting of data from one XML format to another, they are not useful for exporting data to other popular formats such as PDF, RTF etc.

Another problem in the field of content creation occurs due to the continuous nature of the traditional document formats which

are not suitable for using their “sub-parts” or elements for use in content-creation leading to data that is not reusable in a natural manner.

Taking yet again the example of a course instructor, who needs to create redistributable e-Learning packages containing problems, derivations etc. among his/her students regarding a particular course. He/She also might desire that the content so generated must be reusable in other courses and updatable. Suppose, that in such a package, the instructor creates, after devoting much time, a MathML derivation which might also be useful in some other course. In this case, the MathML derivation acts as a content-object – a discrete, reusable and independent e-Learning unit.

Now, supposing the original material is in HTML format, copying the derivation will result in retention of unneeded formatting styles (that are useful only for that document) while searching for the MathML derivation itself might be cumbersome if the document is large enough. This illustrates the need for separation of the document into discrete, reusable learning-objects.

Another problem due to the lack of metadata-enabled content-objects and mixing of business and presentation logic is the difficulty in presenting the same content to different users in different formats. In the case of e-Learning, representing customized data for users (students) of different academic levels or caliber is an important issue. However, generating different documents from scratch having “beginner”, “intermediate” or “advanced” content for the corresponding levels of users is not economically feasible in many cases. The use of content-objects, as shall be shown, solves this issue to a great extent and retains the flexibility in presentation of content in different styling formats.

Thus the main problems in storing and generating learning-based documents in existing conventional formats are: *lack of reusability during content-creation, mixing of presentation and business logic, lack of portability to different media formats, lack of customized content presentation and*

*incompatibilities with search and categorization procedures.*

### 3. XML Content-Objects

XML has, over the years, become the de facto markup language for representing data over the web. Desktop-based database APIs based on XML are also emerging. Amongst its many positive points, such as DTD validation (useful for error-checking), custom tags etc., none is more useful than the strong document-object-model based-architecture that it provides. Not only does this help in effective parsing of XML documents but also helps in easy manipulation of XML-data which can be easily organized into trees, objects and database fields. XML, being a Unicode format, also ensures that culture-specific content is readable globally.

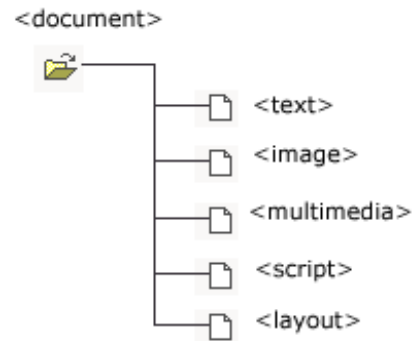
Since XML-marked data is machine-readable, search and categorization procedures become inherently simple for such a document format. Besides this, certain other features like support for flexible metadata helps to categorize objects easily. Thus, the key feature, which makes content-objects attractive candidates for generating e-Learning packages, is that, by using XML, they can be turned into “semantic” content-objects i.e. they contain embedded metadata that describes the content of that object. This can be done for objects at all levels of the document hierarchy.

### 4. The Document Model Hierarchy

According to the document format being proposed, every document exists as a tree-like hierarchical structure having content-objects as its nodes. The document itself also acts like a content-object which leads to inter-document parent-child relationships (which can be useful in content search procedures).

For an independent document, the hierarchy is as follows:

The document is split up into independent logical content pieces known as content-objects as described earlier. The lowest



**Figure 1** The document hierarchy

level content-objects are text-objects, image-objects and multimedia objects. The layout object is the parent node for these objects and acts as a container for these objects. A layout-object can contain other layout objects as well creating a tree-like document structure as depicted.

All the objects consist of two essential elements: metadata and content. The objects are represented in XML format with tags for metadata and content. Individual fields for the metadata section are represented as tags nested inside the metadata tags. Metadata and the content tags lie at the same hierarchical level. Each object is stored as a physically separate XML file.

For flexible parsing options, a separate “structure.xml” file is used to represent the structure of the document. This enables the document parser to know beforehand the expected metadata fields and the document hierarchy (that is, the object nesting rules, the type of data supported within each object, object categories etc.). The structure file is parser-specific and can be changed to include more object-types or change parameters for existing object- types.

*Metadata:* Metadata is embedded within each object. The metadata fields to be supported by each object can be changed through the structure file. Any number of metadata fields can be used for each object. This ensures that compliance with standard metadata formats (such as those developed by the Dublin Core project, IEEE LTSC, or the RDF metadata format) can be

maintained while also supporting custom metadata formats which may be useful in many cases (e.g. for firms which train their employees using e-Learning packages and use an internal metadata standard).

Some standard fields that may be supported are author information, document information, keywords, user-level, device compatibility etc.

*Text-objects:* These objects contain metadata about the object such as content category, object type, keywords, date/time stamp and other data as specified by the content creator.

The content section of the text-object contains Unicode-based text which retains elementary formatting such as paragraphs and white-space.

*Image-Objects:* Image-objects contain the same metadata structure as text-objects. However, image-objects can contain several alternate entries (image-file paths) for the content section. This can aid in sending data according to target device characteristics. An example might be choosing the lower-resolution version of an image to be sent to a PDA device while the higher-quality version may be sent to a desktop PC client. Since the qualitative content of the images remains the same, therefore the metadata and the image objects are also the same.

A similar structure exists for the multimedia object which allows for multiple entries of audio or video files with varying bit-rates targeted at different platforms. Other such low-level objects can easily be incorporated into the document hierarchy using the structure file.

An instance of this might be an “equation-object” whose content part contains MathML data. Implementing an equation as an “equation-object” rather than simple MathML has obvious advantages. For example, using the same equation-object and multiple object-categories, Maxwell’s equations may be represented in integral-form for a “beginner” user while they may be represented in another notation using “del” operators for “advanced” users who has covered vector calculus before.

*Layout-Objects:* These objects act as container objects for image, text, multimedia and other layout objects. These also take care of the sequencing of the objects within a document. The metadata structure is same as that of the other objects. The content section can contain the child-object data in two formats: inline and referential. If objects are specified as inline then these objects are contained fully along with their content in the layout file itself. If the objects are stored as referential objects then the URI (Uniform Resource Indicator) of the object is stored in the layout file like so:

```
<textobj>http://myserver/myexternaltextobject</textobj>
```

Inline content-objects embedded within other layout-objects can be also be included by including a reference to the object name in the URI like so:

```
<imgobj>http://myserver/myexternallayoutobject#myembeddedobjectname</imgobj>
```

This, however, leads to the restriction that child-object names must be unique within a layout-object.

## 5. Content - Creation and Packaging

The above document format stores the document as physically separate files for each object. Therefore, object-specific editors for each of the object types can be constructed for ease of creation and editing of individual objects. These can be combined into a complete document in a WYSIWYG environment in a layout-object editor.

### *Reusability and Search & Categorization*

The objects so created are reusable and each object can be inserted into any number of layout files. A single physical copy of the content-object can serve many layout objects.

With a proper metadata section structure, even differential versioning of the object can be realized. This ensures reusability of the content-object.

Moreover, each object has metadata associated with it which can be easily parsed and stored for indexing. Also, metadata fields can be added or reduced giving great flexibility to the content-creator to describe the content within the object. This also ensures that objects can be categorized easily, thus making search procedures faster and accurate.

A tree-like structure of the document-in-memory also helps to reduce search time. Therefore, unlike HTML documents, the entire file does not need to be scanned for each word of the search query.

#### *Packaging*

The Layout objects along with their constituent text, image and other objects can all be combined to form a single XML file which is known as a package. This package can be stored on a content-serving repository.

However, all components of a package need not be stored locally – individual objects can reside on remote servers and can be referenced appropriately through the Layout object. The package then acts as an information pool through which content-objects can be extracted selectively based on the client's request.

The Layout objects can also be parsed and exported to web-standard formats such as HTML and XHTML (with styling using CSS templates or XSLT). Wireless protocols such as WAP etc. can also be implemented in this way. Transformation to other XML formats for portability to custom database systems is also achieved through this.

Custom parsers need to be written for other standard formats such as PDF, RTF etc. This, again proves, that separating content from protocol is desirable as the same content-package can, after suitable formatting, be deployed on any number of platforms.

#### *Sample Image-Object*

```
<?xml version="1.0"?>
<object id="OBJECTIMG23222-KJHGDH-2004-37373">
  <metadata>
    <type>image</type>
    <name>logo</name>
    <author>S K Som</author>
    <datecreated>23/12/2004</datecreated>
    <timecreated>18:38:45.2164523</timecreated>
    <subject>Content Creator</subject>
    <predicate>Development</predicate>
    <keyword>Content Creator, testing, development</keyword>
  </metadata>
  <imgpath category="high-res" width="800" height="600">test.jpg</imgpath>
  <imgpath category="low-res" width="320" height="240">test2.jpg</imgpath>
</object>
```

#### *Sample Layout-Object (Child objects are referential)*

```
<?xml version="1.0"?>
<object id="OBJECTIMG23222-KJHGDH-2004-37373">
  <metadata>
    <type>layout</type>
    <name>TestDoc2</name>
    <author>S K Som</author>
    <datecreated>28/12/2004</datecreated>
    <timecreated>20:01:12.6723121</timecreated>
    <subject>Content Creator</subject>
    <predicate>Development</predicate>
    <keyword>Content Creator, testing, development</keyword>
```

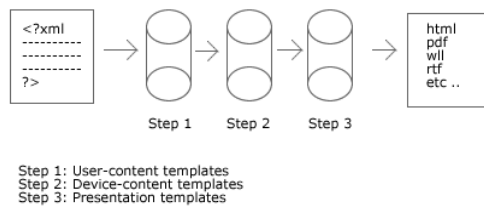
```

</metadata>
<textobj>http://localhost/ContentXML/text_test.xml</textobj>
<imgobj> http://localhost/ContentXML/img_test.xml </imgobj>
<imgobj> http://localhost/ContentXML/TestDoc1.xml#logo </imgobj>
<layoutobj> http://localhost/ContentXML/TestDoc3.xml</layoutobj>
</object>

```

## 6. Deployment

Packages for an e-Learning module may reside on a server. The deployment of the package, on a client's request, involves information processing through three stages.



**Figure 2** Content delivery

The first is the filtering and sequencing of the information according to user-content-templates. Content templates are standard XML files which store information about which content-objects are to be deployed to a client program according to user-level and device capabilities.

All the content objects can be categorized as fit for different levels of users. This information can be stored in the metadata section of the objects. When the request from a client arrives, the client's user-level is ascertained and, accordingly, the package is filtered against the *user-content-template* which, in an e-Learning scenario, can be used for deploying "beginner", "intermediate" or "advanced" content objects. User-profiles describing the clients reside on the server in this model.

The second stage involves filtering of the data against the *device-content-template*. This involves finding out the device capabilities of the system on which the client is being executed and then selecting the appropriate data (such as "low-res" images for PDAs/Tablet PCs, "low-bit-rate" audio for dial-up connections) embedded in the

content-objects as mentioned in the template. Protocol-specific formatting takes place at this stage – for example, for PDAs or mobile phones, the content might be parsed to WAP before the last stage. The information-selection can be done using many standards as the document structure and content-object categories are flexible (for instance, the InfoPyramid Transcoding Model from IBM is easily implemented using the referential layout-object).

Device capabilities can be stored on the server according to the CC/PP framework from W3C or using the WBXML format (preferred).

The final stage involves formatting the data using a *presentation-template* that stores and uses styling information according to client-specifications and content-object categories. The final data is then dispatched to the client.

All the three filters can be combined into a single trans-coding application residing on the server, which handles client requests.

## 7. Conclusion

In this paper we have outlined the basic methodology behind the object oriented content development approach. Although the schemas for various objects have been discussed, they are far from complete. More research will have to be done to expand their schema definitions so that users can be presented with custom content in any format and on any platform they want. Some work is also being done to expand the object-types being supported currently to be able to cover all existing object types. A WYSIWYG object-oriented content authoring environment is also being developed so that objects can be created and achieved for latter user.

## References

[1] R. Mohan, J. Smith, C.-S. Li, "Adapting Multimedia Internet Content For Universal Access," IEEE Transactions on Multimedia, March 1999, pp. 104-114

[2] J. R. Smith, R. Mohan and C.-S. Li, "Transcoding Internet Content for Heterogenous Client Devices," Proc. IEEE Inter. Symp. on Circuits, Syst. (ISCAS), Special session on Next Generation Internet, June, 1998.

[3] Dublin Core Metadata Standard (<http://dublincore.org/>)

[4] IEEE LTSC Learning Object Metadata (LOM) <http://ltsc.ieee.org/wg12/>

[5] Extensible Markup Language (XML) <http://www.w3.org/XML/>