Managing Learning Objects in Large Scale Courseware Authoring Studio

Ivo Marinchev, Ivo Hristov

Institute of Information Technologies – Bulgarian Academy of Sciences,
Acad. G. Bonchev Str. Block 29A, Sofia 1113, Bulgaria
ivo@iinf.bas.bg, collin@iinf.bas.bg

Abstract. In the first part of this paper we represent architecture of distributed collaborative system for creation of reusable and modular learning content and our Description Tool for Learning Objects (DTLO) developed in the scope of LOGOS project. The basic functionality of the DTLO tool is to create learning objects (LOs) from sets of digital objects (DOs) that are physically located in a Digital Object Repository. DOs are annotated and indexed media objects or parts of media objects that are basic blocks for creation of learning content. These objects are aggregated (grouped) and annotated to produce coarse-grained learning entities named LOs. The latter are described with the help of Learning Object Metadata (LOM) format. The purpose of LOM is to support the reusability of LOs, to aid discoverability, and to facilitate their interoperability. In the second part of the paper we presented two possible extensions of the current architecture that can remove some rough edges of the current work process. The first proposition is the procedure for storing and sharing search filters created with DTLO search component as a communication channel for exchanging information between different user roles in a semi-formalized manner. The second proposition is employing UDDI registry as a facilitator of the integration between different tools that comprise the LOGOS framework.

Keywords: e-Learning, m-Learning, Ubiquitous Learning, Multimodal Environments, LOM, METS, SCORM, UDDI.

1 Learning Objects in Courseware Authoring Architecture

Today many learning management systems use Sharable Content Object Reference Model (SCORM) because it is widely acceptable as industry standard. SCORM define major aspects of usage of electronic content and there are lots of best practices of implementing SCORM content. SCORM editors like Reload [1], eXe [2] etc. are widely used for preparing and structuring e-Learning content. The SCORM tools are mainly targeting usage of fully functional browsers and desktop systems. Even there

---

1 This work was partially supported by project FP6-IST-027451 LOGOS “Knowledge on Demand for Ubiquitous Learning” and by project IIT-010061 “Technologies of the Information Society for Knowledge Processing and Management”
are dedicated tools for development and integration of e-Learning content for mobile learning [3], development of content suitable for multi-channel delivery has many challenges coming from technical characteristics of end user devices, channel properties, typical learner environment, and context of usage. Preparation of learning objects (LOs), suitable for delivery over different channels, is not a trivial task, because of many requirements, in some cases contradictory with their properties and lack of integrated courseware development tools. One of the goals of LOGOS Project [4] is to generate new cross-media learning content with specially developed authoring studios using existing digital archives. The simplified LOGOS Authoring Studio architecture is shown on the Figure 1 and is composted of following building blocks:

- **Repositories** – works as storage spaces for different kind of objects. Repositories are accessible via http protocol and presents functionality as web services.
- **Media Server** - contains actual archive data referenced by DOs.

![Fig. 1. General architecture of the LOGOS authoring studio.](image-url)
Tools for creation and editing of different types of objects. They are designed to support single atomic unit of work, specific for a given authoring studio user role. The tools on Figure 1 are:

- **Ontology Management Tool** – creates and maintains the core concepts of the domain specific ontologies.
- **Content Description Tool** – segments, annotates and semantically indexes multimedia content; supported by two search and retrieval tools - the Navigation-based Information Retrieval Tool and the Graphical Conceptual Graph Querying Tool.
- **Description tool for Learning Objects (DTLO)** – creates and maintains LOs.
- **Learning Design Editor** – creates abstract learning scenarios so that they could be used for the dynamic creation of personalized courseware.
- **Courseware Objects Editor** – creates static courseware or modifies dynamically created courseware.
- **Publishing Tool** – publishes static courseware as ready-to-be-delivered courses, thematic web folders, interactive videos, etc.

Authoring Studio user roles:

- **Knowledge Managers** – They create and maintain domain-specific ontologies, necessary for the semantic description of audiovisual content. They are domain-experts able to specify domain-specific concepts in the ontology language of LOGOS (Conceptual Graphs).
- **Annotation/Indexers** – They annotate, segment and semantically index the raw audiovisual material in order to create and maintain DOs.
- **Educationalists** – They create reusable LOs. Their work starts with the selection of appropriate DOs. These objects (and combinations of them) are then enriched with educational metadata for a given pedagogical or para-pedagogical use.
- **Learning Designers** – They use the Learning Designs Editor to create abstract learning scenarios so that they could be used for the dynamic creation of personalized courseware.
- **Courseware Developers** – They create, maintain and publish static courseware for learners. The creation of static courseware may exploit the facilities of the Courseware Repository to create dynamically courses and then modify the dynamically created courses. The Authoring Studio tools used by Courseware Developers are the Courseware Objects Editor and the Publishing Tool.

The authoring studio architecture is build on logically separated layers, that helps breaking down complexity and provide appropriate tools for various user roles so that each user can concentrate on his/hers specific responsibilities with respect to the authoring process. That enables in some cases parallel working and work specialization.

In the above architecture LOs stay at the middle between DOs and courseware objects (COs). LOs consist of one or more appropriate DOs and they are used as building blocks for the creation of COs. LOs contain metadata that follow the LOM Standard [5]. Learning object definitions are stored in the Learning Objects
Repository. LOs are managed by Educationalist user role. He or she composes objects by adding references to DOs and entering metadata descriptions. LOs are structured according to METS standard [6]. dmdSec section of the document contains LOM description, fileSec contains links to DOs being used and structMap outlines a hierarchical structure for the original DO.

2 Description Tool for Learning Objects

The Description Tool for Learning Object is used for the pre-selection and organization of sets of relevant DOs into LOs for a given pedagogical or para-pedagogical use. This tool essentially provides the means to add educational metadata to a particular combination of DOs so that reusable LOs is created. The latter could be exploited for courseware creation or as elements facilitating learning processes (e.g. material that a teacher can use in the classroom).

The high level functional requirements of the DTLO are the following:

- Support creation, management and retrieval of reusable LOs consisting of DOs – The DTLO should provide functionality for the creation, management and retrieval of reusable LOs consisting of DOs. Internally it uses the appropriate web services for the creation, management and retrieval of LOs provided by the Learning Objects Repository.
- Ensure authorized access – the DTLO should have an authentication process to ensure that only authorized users have access to it.
- Work with Digital Object Repositories – the DTLO should provide a list of available Digital Objects Repositories so that the user may select one in order to create LOs.
- Create new LO – the DTLO should provide functionality to select DOs and combine them with educational metadata in order to create a LO. It uses the service to submit/store a new LO provided by the Learning Objects Repository.
- Create educational metadata for LOs that facilitate their retrieval and integration in COs, using LOM specification standard – the DTLO should be able to create LOs from DOs and specify the corresponding educational metadata based on the LOM specification. These LOs can be further used to create COs through the functionality offered by the Courseware Objects Editor or through the middleware for the automatic construction of personalized courseware.
- Enable editing LO metadata. It should use the services to create a new LO by copying an existing one and to submit/update a LO provided by Learning Objects Repository
- Allow search through learning object repository - the DTLO should provide functionality for searching LOs residing at the Learning Objects Repository including functionality to browse the search results. The Learning Objects Repository provides services to search LOs using certain criteria.

DTLO is implemented in Java programming language [7]. Basic application framework that we use is Eclipse Rich Client Platform (RCP) [8]. The latter was selected because it allows building the tool as a collection of modules (plug-ins in
Eclipse terminology) that can be reused (if it is needed) in other parts of the LOGOS framework. Also the standardized component framework as Eclipse RCP prescribes predefined ways of integration between the modules developed from different partners in the scope of the LOGOS project. The implementation follows Model-View-Controller design pattern. The interface is bound to the model by Eclipse Data-binding Framework. A custom DataGrid control was developed (because SWT library does not provide stable and customizable DataGrid control at the moment) to facilitate editing of certain data in more intuitive tabular manner.

It was decided that the DTLO tool should additionally use a local storage facility for storing a working copy of all edited documents. Some of the reasons/advantages for using local storage in our tool are:

- Network connection is not always reliable.
- Avoids latency in the interface if only remote repository is used.
- Reliability in case of system or program failure.
- In practice it is faster than using in memory XML models (because of indexing).
- Keep template document in the local repository.
- Allows offline work.

As all documents processed and exchanged in the LOGOS authoring studio are in XML format in practice the best suited for the local storage is embedded XML database that support XPath, XQuery, and XML Validation. For our current implementation the eXist XML database [9] was selected based on evaluation of its stability, feature set (mainly complete XQuery and XUpdate support) and active development.

### 3 Managing Top-Down Data Workflow

LOGOS authoring studio supports a bottom-up approach for courseware creation. Figure 2 depicts the data flow in the process of creation of a CO that corresponds on the general architecture discussed in the previous section. The Annotator (see the user roles in the Figure 1) creates DOs in the Digital Object Repository. The Annotator uses pre-existing media files from the media archives. On the next step the Educationalist using DTLO tool creates LOs from the existing DOs available in the DOs repository. Finally Learning Designer and Courseware Developer use completed LOs to create courseware content.

Important question in this setting is how the users from the different roles that usually work completely separate in space and time can communicate their needs in this environment. Users from the “higher” levels usually use already available resources in the repository to do their work. They search through the repository using the so named search filters that return all available objects satisfying certain structured description. But if there are no suitable objects available they need to communicate their needs with the "lower" level user roles. To address this need they may store a search filter as a semantic description of the object(s) that the "lower" level must provide.
In the current implementation of the LOGOS studio, the search filters are part of the DTLO tool user interface. These filters are implemented in the LOs repository and exposed as SOAP web service. The filters specify the metadata parameters to be examined in order to return the corresponding LOs. Figure 3 shows a screenshot of the current LOs search filter interface of our tool. In the search filter dialog, different sections of the LOM specification are represented as separate tabs. This dialog is used
to collect the data needed to build a XML query document that is send to the repository. The results of the query (matching LOs) are shown in the table at the bottom of the dialog.

Based on the above introduction we propose the following top-down data workflow (Figure 4) that channels the request of higher to lower levels in the studio environment. If the user from the upper level can not find the information he/she needs in the repository that corresponds to his/her functional level he/she stores the most specific filter in the repository as a request for fulfillment. Then the lower level users can review the pending request fulfill them and then notify the originator of the request that the new corresponding objects are already available.

![Diagram](image.png)

**Fig.4.** Workflow when higher levels request specific objects from the lower levels.

We think that the approach proposed in this section will create semi-formalized process in managing, tracking and auditing the information needs of the users of the authoring studio and it is better than the free form plain text communication that can occur by email or some instant messaging system.

### 3 Central vs. Distributed Repositories

The second extension that we propose deals with the integration points between different tools that comprise the LOGOS framework. Interactions between separate tools are implemented with the help of predefined SOAP web services. Currently, each tool contains hard-coded (or as a configuration parameter) the connection points
of all of the web services that it consumes. This architecture is inflexible and hard to maintain because different partners can reconfigure and move their web service, can create clusters or introduce additional repositories over the time.

A possible solution to these problems is the introduction of centralized (or distributed) UDDI registry. A UDDI registry's functional purpose is the representation of data and metadata about web services. A registry, either for use on a public network or within an organization’s internal infrastructure, offers a standards-based mechanism to classify, catalog, and manage web services, so that they can be discovered and consumed by other applications. As part of a generalized strategy of indirection among services based applications, UDDI offers several benefits to IT managers at both design-time and run-time, including increasing code reuse and improving infrastructure management by:

- Publishing information about web services and categorization rules specific to an organization.
- Finding web services (within an organization or across organizational boundaries) that meet given criteria.
- Determining the security and transport protocols supported by a given web service and the parameters necessary to invoke the service.
- Providing a means to insulate applications (and providing fail-over and intelligent routing) from failures or changes in invoked services.

Instead of hard-coded web service connection points in the client programs, the tools can get them (for example at startup time or upon the first use) from this registry. Figure 5 shows the general work process that employs UDDI registry indirection in the scope of the LOGOS framework represented as a use case of our DTLO. Before the first communication with the web services of the LO repository (or repositories) is initiated, the DTLO can initially get the current connection points and
invocation protocol (i.e. discovery of the web services) from the UUDI registry and then consumes the corresponding services. This scenario allows the developers and/or administrators of the LO repository to be able to freely move, cluster, replace cluster nodes, etc of their software provided they reflect their changes in the UDDI registry without the need to communicate their changes with the rest of the partners. The same web service discovery approach can be utilized not just be DTLO but all LOGOS framework tools that consume web services.

4 Conclusions

In this paper we have presented architecture of distributed large-scale courseware-authoring studio. It comprises logically separated layers, using 3 type of entities (DOs, LOs and COs) that helps to break down complexity. We focus on LOs creation and manipulation process (and tools) as they are the basic entities that facilitate learning content sharing and reuse. Finally we describe possible improvements in the Authoring Studio architecture and implementation based on our experience gained during the implementation of our DTLO tool, that can benefit all of the partners involved in the creation of the LOGOS studio and its future users.

References

1. Reload Project, Reusable eLearning Object Authoring and Delivery, http://www.reload.ac.uk/editor.html
4. LOGOS Project, www.logosproject.com