Tailoring Learning Management Systems and Learning Contents for the SCORM Model *

Xin Xiang, Ling Guo, Yuanchun Shi
State Key Lab of Intelligent Technology and Systems,
Computer Science Department, Tsinghua University, Beijing, 100084, P.R. China
{xiangx01@mails., guoling02@mails., shiyc@}tsinghua.edu.cn

Abstract

Standardized learning management systems and contents are becoming prevalent over time with the growing adoption of Web-based learning technologies. The complexity adherent in the SCORM run-time environment APIs and data model, however, makes it difficult for instructional designers, content developers and LMS vendors to tailor their contents and LMSs for the SCORM model. In this paper, our own practice of the SCORM model is presented. We firstly propose our architecture for E-Learning applications. Being LMS-centered, the architecture seeks to bridge the gap between conformant LMSs and diverse learning contents. To highlight our design, we present a comprehensive LMS implementation with extended data elements adapted to meet the needs of college education. Moreover, we detail a systematic method of turning existing HTML-based courseware into SCORM conformant contents. The last part of this paper discusses technical and pedagogical issues of concern regarding the learning scenarios of SCORM in different learning environments and the accordingly tuning of the LMSs.

1. Introduction

The recent approval of LOM (Learning Object Metadata) standard[1] by the IEEE-Standards Association and of Dublin Core Metadata Element Set[2] by ISO marked a new milestone in the field of metadata standardization. It is believed that in the near future, they are to be widely accepted by industry and academia alike.

Whereas metadata and content related E-Learning standards are intended to specify the description of metadata and content, the SCORM (Sharable Content Object Reference Model) model[3][4][5] extends these standards by further specifying the run-time environment of LMSs (Learning Management Systems), including APIs and data elements needed to launch the SCOs (Sharable Content Objects).

Based upon the AICC API specification[6], the IMS metadata and content package information models[7][8] and their XML binding specifications[9][10], the SCORM model has made a great progress in leading instructional designers, content developers and LMS vendors to develop their standardized products.

Further, the newly released SCORM 1.3 application profile working draft[11] complements the original SCORM model with the missing piece, namely sequencing, based on the simple sequencing specification published by IMS[12]. This supplement allows course designers to specify a learner's path through a number of learning objects depending on how they are doing, thereby making the SCORM model more flexible and easy to adapt for different instructional needs.

In this paper, we are to present our own practice of the SCORM model. In our viewpoint, LMSs play a central role in the Web-based E-Learning scenario. It connects learning contents and learners together in a standardized manner and is underscored in our E-Learning application architecture. Taking into account the maturity of the SCORM 1.2 specification and the complexity of the sequencing mechanism introduced in the SCORM 1.3 application profile, we choose SCORM 1.2 as the basic specification with which both the LMS and the contents comply. As a part of a campus-wide comprehensive Web-based E-Learning system, the LMS aims to provide an effective learning platform by customizing the SCORM run-time environment. It not only implements the basic run-time environment functionality, but also provides overall records on learner performance, which could be revisited in future evaluation. Besides, since conformant contents are also indispensable in fulfilling the instructional and technical function of the SCORM run-time environment, we demonstrate how to reconstruct a common HTML-based courseware to make it SCORM conformant, according to a programmable and systematic approach.

* This material is based upon work supported by the Ministry of Education and the National Natural Science Foundation in China.

The remainder of this paper is organized as follows: section 2 introduces some SCORM related work. Section 3 introduces the LMS-centered E-Learning application architecture, and section 4 covers the customization of the SCORM run-time environment. Section 5 gives a common method used to repurpose existing HTML-based courseware for SCORM. The process of launching SCOs from within run-time environment is detailed in section 6. Section 7 introduces technical and pedagogical issues of concern with respect to the differentiation of public and personal LMSs. Section 8 concludes this paper.

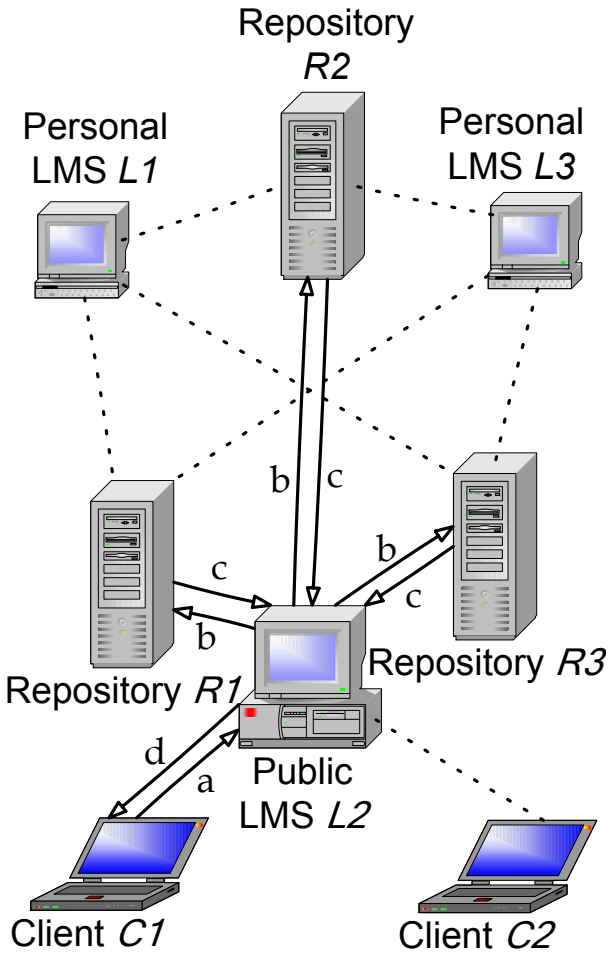
2. Related Work

Considering the complexity of the SCORM model, ADL released a SCORM implementation guide[13] covering the learner and context analysis, the instructional and content design, the development of SCOs and content packages, and the verification and validation of final products. This guideline provides a step-by-step and easy-to-follow approach for instructional designers and content developers to construct their SCORM conformant contents. Also, the Carnegie Mellon Learning Systems Architecture Lab released a SCORM best practices guide[14] being targeted at instructional designers and content developers. The guide gives a detailed description on the designing of SCOs, structuring of tests and determining of sequencing in the process of developing SCORM conformant contents. The accompanying SCORM Simple Sequencing Templates and Models[15] could be conveniently copied and revised to fit diverse instructional needs.

Aside from official SCORM guidelines, some issues of concern regarding the practice of SCORM have been proposed in the literature. [16] describes the design and implementation of library module components for WBT (Web-Based Training) systems conforming to SCORM and AICC CMI specifications. [17] proposes a method using Web service to build the SCORM run-time environment and LMS. [18] discusses several deficiencies of SCORM concerning reusable learning contents and WBT. [19] presents the design and implementation of a SCORM conformant CS courseware, and [20] narrates the SCORM conformant redesigning and upgrading of a collaborative courseware generating system. [21] proposes a method for separating the presentation of a SCO from its content, allowing multiple SCOs from multiple origins to be combined in a single unified learning experience. [22] gives a brief introduction of SCORM and its implications on engineering education.

3. Architecture

Our discussion is based on an LMS-centered E-Learning applications architecture depicted in Figure 1.



a. the requests of clients
b. the requests of LMSs asking for metadata or contents
c. the delivery of meta or contents from repositories
d. the delivery of metadata or contents to clients

Figure 1. The Architecture of LMS-Centered E-Learning Applications

Basically, there is no one-size-fits-all architecture or framework that could address all the problems in the field of Web-based learning. The proposed architecture focuses on the effective delivery of standardized contents between E-Learning systems and students in an LMS-centered approach.

The proposed architecture comprises three types of participants:

- *Learning Resource Repositories*

Standardized learning resource repositories provide massive storage for learning resources and metadata, and a uniform interface for query and delivery. The repositories deal with two types of requests: the query requests searching for specific metadata, and the delivery requests asking for the actual content. The content may be an asset,

a SCO, or a conformant content package, as described in section 4 and 5.

- *Learning Management Systems*

LMSs play the role of clients and application servers simultaneously. As clients, they request metadata and contents from the repositories; and as application servers, they forward the clients' query and delivery requests to repositories, and prepare the returned metadata and contents for clients' browsing.

- *Clients*

Clients use common Web browsers to view the metadata information and the launched SCOs through HTTP sessions.

It is possible for a client and an LMS to reside in the same host, e.g., a PC may have a lightweight personal LMS (such as L1 or L3 in Figure 1) capable of requesting contents, launching SCOs, and providing application service to a local browser.

A typical learning scenario taking place in this architecture is described as below:

1. Client C1 logs on to LMS L2;
2. Client C1 issues a request for conformant contents on *data structure in computer science*;
3. LMS L2 forwards the request to repository R1, R2 and R3 in turn;
4. Metadata records satisfying the request are returned from the repositories. After being returned, they are cached in LMS L2;
5. Client C1 looks through the resulting metadata of contents and issues a request to download one of them;
6. LMS L2 forwards the request to the repository storing that content;
7. The repository delivers the requested content to LMS L2;
8. LMS L2 launches the SCOs packaged in the retrieved content from within its SCORM compliant run-time environment and delivers them to client C1.

This architecture to some extent integrates existing heterogeneous learning resource repositories and learning management systems, and connects the students and learning contents in a distributed environment.

The LMSs play a key role in the scenario. It is the LMSs where most of the application logics of learning activities are performed, including search, evaluation, delivery and launch. It is also the LMSs that bridge the gap between a variety of contents and learners who are unaware of the internal mechanism of run-time environment when browsing learning materials.

The LMSs may vary in terms of size, performance, functionality and scalability. The lightweight LMSs, such as LMS L1 and LMS L3 in Figure 1, could be referred to as *Personal Learning Management Systems*. They are for personal use, functionally and instructionally compact, and could be easily deployed on a home PC. On the other hand, *Public Learning Management Systems*, such as LMS L2 in Figure 1, have the full functionality of an LMS and are able to serve learners in an organization ranging from a lab to a college.

4. Customization of the SCORM Run-Time Environment

Incorporating the AICC CMI/Lesson communication data model and IMS metadata and content packaging specifications, SCORM seeks to equip developers with a standard and practicable application profile in developing conformant contents and learning management systems. The API and data model defined in SCORM, however, is rather complex and all-inclusive. It is the responsibility of instructional designers, content developers and LMS vendors to customize and tailor the SCORM model to fit their instructional and technical needs.

A public LMS is needed for college education. It should provide the basic functionality of the run-time environment as well as overall records on learner performance, which could be revisited in future evaluation.

4.1 The Components

As depicted in Figure 2, four components are involved in the run-time environment:

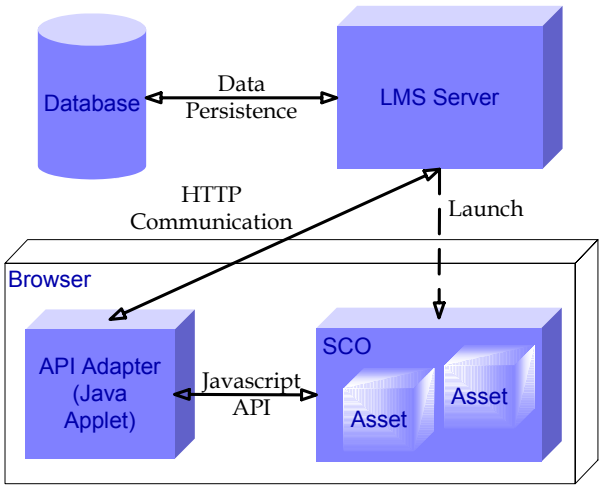


Figure 2. The SCORM Run-Time Environment

- *LMS Server*

Strictly speaking, an LMS server is part of an LMS. Besides the LMS server, the LMS also provides the data persistence and the API adapter described below. Hereafter, an LMS server is referred to as the component of an LMS

that is responsible for communication with the API adapter.

In our system, the LMS server is implemented as a Web application deployed on a WebLogic application server. It manages all the database connections and http sessions through the internal mechanism provided by the application server.

- *API Adapter*

The role of the API adapter is to connect the client-side SCOs with the server-side learning management system server. It implements the required API functionality defined in the SCORM run-time environment specification.

We built a fully functional API adapter as a Java applet without user interface (It can also be implemented in C++ and loaded as a browser plug-in.). It is delivered to the browser when the learner logs on to the LMS. To maintain the robustness of the API adapter, all the required APIs and an adaptive debug mechanism are implemented.

- *SCO*

A *SCO* is defined in the SCORM content aggregation specification as a collection of one or more assets including a specific asset that is launchable and utilizes the SCORM run-time environment to communicate with the LMS server. *Assets*, as defined in SCORM, are electronic representations of media, text, images, sound, Web pages, assessment objects or other pieces of data that can be delivered to a Web client.

It is the responsibility of the SCO to locate the API adapter and issue the required API calls to communicate with LMS.

Section 5 details the customization of SCOs.

- *Database*

Although not explicitly specified in the SCORM model, databases play a key role in maintaining the persistence of data model elements transmitted between the LMS and the client browser, especially for a comprehensive LMS designed to serve a large group of users.

In our system, an Oracle 8i database management system is used to store all the implemented run-time environment data elements in conjunction with some educational parameters useful in future evaluation.

4.2 The Data Elements

All the data elements defined in the SCORM run-time environment data model are broken up into nine categories: core, suspend data, launch data, comments, objectives, student data, student preference, interactions and comments from LMS, as depicted in Figure 3 (reproduced from [13]).

Figure 3. Data Model Categories Available for Use by the SCO (Reproduced from [13])

Some of them are mandatory for an LMS to be SCORM 1.2 conformant[23] (To increase interoperability, the SCORM version 1.3 will make all run-time environment data model elements mandatory, but presently we prefer SCORM 1.2 and do not follow that obligation.), hence should be implemented in any SCORM conformant LMS. These mandatory data elements include: student id, student name, lesson location, credit, lesson status, entry, score, total time, exit, session time, suspend data, and launch data. All of them are indispensable for an LMS to track the information contained in a SCO and the learner performance.

To keep track of the exact time a user requests to launch and exits a SCO, two data elements are added to the SCORM run-time environment data model. One is *start_time*, representing the time when a learner requests to launch a SCO, the other is *end_time*, representing the time when a SCO is exited.

Both our LMS server and API adapter provided by the LMS have been adapted in order to effectively accomplish the addition of data elements.

To comprehensively record the information pertaining to learner performance, several database tables are created to maintain the information about students, courses and SCOs.

5. Making an HTML-Based Courseware SCORM Conformant

One initiative of the SCORM model is to construct learning material as SCOs that could then be seamlessly reused within different learning environments in a context-independent fashion. The concept of integrating

existing sharable content objects, however, has been doubted by some instructional and pedagogic experts who claim that a sequence of such de-contextualized learning objects may not truly convey a unified experience for the learner[24].

We thereby drop the idea of developing a comprehensive courseware on the basis of a series of reusable learning objects, and resort to a technically and pedagogically more sound approach, that is, repurposing existing learning materials for SCORM model.

The existing material is an HTML-based courseware on *The C Programming Language Course* in computer science with built-in navigation between chapters and sections.

In the process of making an existing HTML-based material SCORM conformant, the following steps are required:

1. *Remove all the frame structures in the HTML pages.*

In a SCORM conformant learning management system, frames are used to provide a tree-based navigation mechanism. If an HTML page itself contains frames, the Web interface provided by the LMS will be distorted, hence is unacceptable. By removing the frame structures, the HTML pages become clean, compact and easy to tailor for the SCORM model.

2. *Identify SCOs.*

Although there are many different instructional designs that could affect the identification of SCOs, we adopt a simple and easy-to-follow method, that is, identify each HTML page in the courseware as a SCO.

3. *Obliterate all the navigation structures contained in the SCOs (HTML pages).*

All the navigation mechanisms, i.e., all the buttons such as “previous page”, “next page”, “first page”, “last page”, etc, are to be provided by the LMS as a sequencing engine defined in SCORM as the components of an LMS used to interpret sequencing information and execute the specified sequencing behaviors. It may sequence the contents based upon the sequencing information provided in the content package, or in accordance with its hard-coded sequencing mechanism.

4. *Remove all the inter-SCO links.*

In SCORM model, only an LMS can launch SCOs. A SCO can not launch another SCO. In this step, the previous rule applies, that is, all the navigation mechanisms are to be provided by the LMS.

5. *Tailor the SCOs for the SCORM model.*

A SCO is required to adhere to the SCORM run-time environment. The SCO must have a means to locate an LMS’s API Adapter and must contain minimum API calls, i.e., *LMSInitialize* and *LMSFinish*. It can be achieved by calling common methods responsible for finding an adapter and calling APIs, thus obviating the need for each SCO to execute the same code fragment. The file containing these common functions can be identified as an asset.

6. *Create metadata for the learning objects in the courseware.*

Although optional in the development of SCORM conformant content, metadata is used here to increase the reusability and discoverability of the contents. The metadata pertaining to the courseware include the metadata for assets, SCOs and the whole package.

7. *Create manifest file.*

The manifest file contains information relative to the contents (including assets and SCOs) in the courseware, the course structure, the location of metadata, and if available, the sequencing method. The manifest file is indispensable for the LMS to extract information from the courseware for future launch.

8. *Package the courseware in a PIF (Package Interchange File) format using the PKZIP Version 2.04g archive format (zip).*

The PIF provides a concise Web delivery format that can be used to transport content packages between systems (zipped). The LMS could then import the packaged courseware and extract all the needed information.

A launched SCO in the standardized courseware is shown in Figure 4:

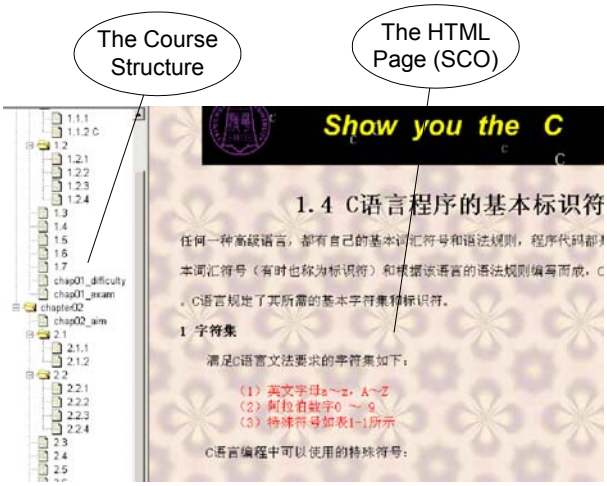


Figure 4. A Launched SCO in the LMS

Since there is no special requirement on the sequencing method utilized by the LMS in launching this course (and it is hard for instructional designers to impose lots of sequencing requirements in a non-traditional course and comply with the newly published SCORM 1.3 draft specification), the LMS could launch the course in a simple *flow* or *choice* sequencing method, that is, launch the SCOs one by one or launch them upon the unconstrained choice of the learner.

As far as the navigation interface is concerned, the

AICC's icon standards: user interface[25] gives a good recommendation on functions of the student/user interface to CBT material and delivery systems.

The procedure described above is rather straightforward and could be employed to mass-produce SCORM conformant contents based on existing HTML-based courseware. These courseware are considerably popular in current E-Learning environment.

[19] also gives an instructive narration of SCORM conformant redesigning of an existing computer science courseware. It focuses on the identification and management of assets, SCOs and content aggregations.

6. Launching of SCOs from within Run-Time Environment

At the time of courseware importing, a persistence object (database or disk file) is built to maintain the persistence of run-time environment data elements implemented by the LMS. Meanwhile, the LMS extracts the course structure by parsing the standard manifest file contained in the content package, and the sequencing method is determined depending on either the sequencing mechanism hard-coded in the LMS or that specified (by instructional designers and programmers) in the manifest file.

On completion of the actions above the course is ready for future launch.

Upon the learner's request for launch, the LMS will firstly determine the structure of the requested course, and find from database whether it is the first time that the learner accesses the course, or the learner quitted the course normally or abnormally in an earlier time and wants to resume now.

If it is the first time that the learner launches that course, the LMS should determine which SCO is to be launched first. After that, the LMS delivers that SCO to the client browser through HTTP session.

If the learner once quitted the course and wants to resume now, the learning management system should determine (through database) the point where the learner quitted the course last time, and deliver the relevant SCO to the client browser.

It is the responsibility of the SCO to find the API adapter upon launch, through which it could communicate with the LMS. Thereafter the API adapter acts as a broker between the SCO and the LMS.

As per the SCORM specification, the SCO should at least call the methods *LMSInitialize* and *LMSFinish* to be SCORM 1.2 conformant, and depending on the nature of the content, it may call other APIs defined in SCORM model, namely *LMSGetValue*, *LMSSetValue*, *LMSCommit*, *LMSGetLastError*, *LMSGetErrorString*, and *LMSGetDiagnostic*.

Most of the SCOs in the courseware could simply call *LMSInitialize* upon loading and *LMSFinish* upon unloading.

More complex SCOs may call *LMSGetValue* and *LMSSetValue* to operate different SCORM model data elements. Fox example, a SCO containing a test may call *LMSSetValue* to notify LMS the learner's performance on the test, another SCO may call *LMSGetValue* to acquire the learner's name and print a greeting message.

The traverse of a SCO's states during its lifecycle is depicted in Figure 5.

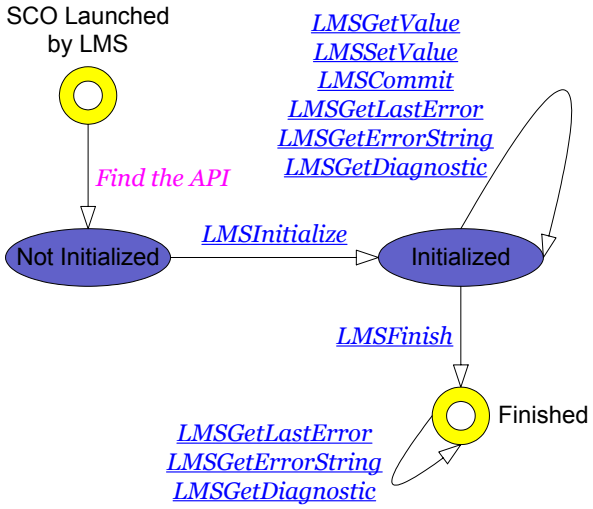


Figure 5. The SCO State Transitions

As can be seen from the figure, all processing relative to the current SCO must be performed prior to calling *LMSFinsih*.

7. Public and Personal LMSs

We further the initiative of customizing LMS in section 4 here by a discussion of differentiation between public and personal learning management systems.

The differentiation of these two kinds of LMSs comes from the consideration that although large and comprehensive LMSs are important for both training and educational use, small and compact LMSs are also indispensable in facilitating the off-line and unconstrained (by sequencing method) use of learning materials in a personal environment.

A comprehensive public LMS should implement the following function regarding SCO launching:

- *Fully Functional API Adapter*

The LMS should supply a stable and robust API adapter that fully implements the required functionality described in [5].

- *Comprehensive Data Elements*

The LMS should implement most data elements

defined in the specification including elements describing student information, learner performance and comment data to enable full support of the SCORM run-time environment data model and utmost trackability of information about the launched SCOs.

- *Persistence Through Database*

Database-enabled persistence mechanism should be provided to maintain the persistence of the complex data elements implemented by the LMS. The database could extensively and efficiently record the learner information, SCO information and learner performance, thereby fulfilling the responsibility of a public LMS.

- *Adaptive Sequencing Engine*

To fully function as defined in the IMS simple sequencing specification and the SCORM 1.3 application profile, the LMS should implement an adaptive sequencing engine that could handle various sequencing requirements arising in Web-based learning environments, hence meeting different technical and instructional needs in different learning contexts.

A personal LMS, however, need not implement as much functionality as a public LMS does.

- *Simple API Adapter*

A personal LMS should supply a simple and compact API adapter that implements the basic functionality during launching of SCOs, for instance, the *LMSInitialize* and *LMSFinish* described in [5].

- *Reduced Data Elements Set*

As proposed in [19], the set of data elements implemented by a personal LMS might be selectively reduced in order to improve the performance of LMS while maintaining its basic functionality.

- *Lightweight Persistence Mechanism*

Since a personal LMS cares little about the management of the learner information, the detailed SCO information and the evaluation of learner performance, it is recommended that the basic disk file-based persistence mechanism be implemented, making the LMS small, efficient and easy to deploy.

- *Basic Sequencing Engine*

The personal LMS may only implement the basic *choice* sequencing method, neglecting the instructional designer's sequencing requirements and allowing for the learner's freely browsing the learning material, most probably off-line.

Although SCORM claims to be pedagogically neutral, the learning scenario of launching SCOs in a Web-based run-time environment has been deemed individual-centric[26][27][28], even in a pubic LMS environment. As stated in [26], "SCORM is essentially about a single-learner, self-paced and self-directed." The purpose of the differentiation of personal LMSs from public ones, however, is to adapt the LMSs in different learning environments, thereby avoiding a one-size-fits-all solution in this rapidly changing E-Learning application market.

We hope that the in the near future, SCORM will evolve into an E-Learning standard that is not only technically mature, but also pedagogically sound. And it is the responsibility of LMS vendors to tune their products, not only to fit training use, but also to meet the progressive need for collaborative learning.

8. Conclusion and Prospect

In this paper, we present our own practice of the SCORM 1.2 model in an LMS-centered distributed E-Learning architecture. In the architecture, a comprehensive LMS is tailored for college use, and the process of making an HTML-based courseware SCORM conformant is detailed for demonstration. Finally, a discussion focusing on the differentiation of public and personal LMSs is given, from pedagogical and technical perspectives.

At this time, a campus-wide E-Learning application conforming to the proposed architecture, with Xindice[29] as its XML metadata repository and WebLogic as the application server is being developed in our university. A public LMS with extended data element set based on Oracle 8i database has been implemented to fit the need of college education.

The learning management system may be improved in the following aspects:

- *Introduction of Adaptive Sequencing Engine*

As the main contribution of SCORM 1.3 application profile to the SCORM model, sequencing mechanism has been deemed not only a technical improvement of LMSs, but also a pedagogical requirement of instructional designers and content developers. For simplicity, there are only *flow* and *choice* sequencing methods available in current implementation.

We hope that in the near future, an adaptive sequencing engine complying with the simple sequencing model in SCORM 1.3 application profile could be designed and developed, hence providing instructional designers and content developers with a variety of choices of sequencing their contents depending on their technical and pedagogical needs.

- *Implementation of All SCORM Model Data Elements*

Although laborious for implementers, the SCORM version 1.3 will make all SCORM run-time environment data model elements mandatory to increase interoperability. The concept of optional data model elements from an LMS perspective has been removed. As practitioners of a comprehensive public LMS, we consider it desirable to implement as many SCORM model data elements as possible to benefit both the content developers

and the users.

- *Integration of Personal LMSs*

With the growing number of conformant LMSs and contents, a single user is likely to freely browse conformant contents off-line instead of logging on an LMS, registering for a course and browsing material in the sequence predefined by instructional designers. To fit these needs, a personal LMS implementing the basic APIs and data model of run-time environment should be provided.

9. References

[1] IEEE Learning Technology Standard Committee Working Group 12, "Final 1484.12.1 LOM draft standard", http://ltsc.ieee.org/doc/wg12/LOM_1484_12_1_v1_Final_Draft.pdf, September 2002.

[2] Dublin Core Metadata Initiative, "Dublin Core Metadata Element Set, Version 1.1", http://dublincore.org/usage/terms/dc/current-elements/, October 2002.

[3] Advanced Distributed Learning, "The SCORM Overview, Version 1.2", http://www.adlnet.org/ADLDOCS/Documents/SCORM_1.2_Overview.pdf, October 2001.

[4] Advanced Distributed Learning, "The SCORM Content Aggregation Model, Version 1.2", http://www.adlnet.org/ADLDOCS/Documents/SCORM_1.2_CAM.pdf, October 2001.

[5] Advanced Distributed Learning, "The SCORM Run-Time Environment, Version 1.2", http://www.adlnet.org/ADLDOCS/Document/SCORM_1.2_RunTimeEnv.pdf, October 2001.

[6] Aviation Industry CBT Committee, "AICC CMI Guidelines for Interoperability Revision 3.5 Release 2", http://www.aicc.org/docs/tech/cmi001v3-5.pdf, April 2001.

[7] IMS Global Learning Consortium, "IMS Learning Resource Meta-data Information Model Version 1.2.1 Final Specification", http://www.imsglobal.org/metadata/imsmdv1p2p1/imsmd_infov1p2p1.html, September 2001.

[8] IMS Global Learning Consortium, "IMS Content Packaging Information Model Version 1.1.2 Final Specification", http://www.imsglobal.org/content/packaging/cpv1p1p2/imscp_infov1p1p2.html, August 2001.

[9] IMS Global Learning Consortium, "IMS Learning Resource Meta-data XML Binding Version 1.2.1 Final Specification", http://www.imsglobal.org/metadata/imsmdv1p2p1/imsmd_bindv1p2p1.html, September 2001.

[10] IMS Global Learning Consortium, "IMS Content Packaging XML Binding Version 1.1.2 Final Specification", http://www.imsglobal.org/content/packaging/cpv1p1p2/imscp_bindv1p1p2.html, August 2001.

[11] Advanced Distributed Learning, "ADL SCORM Version 1.3 Application Profile Working Draft 1.0", http://www.adlnet.org/adldocs/Other/SCORMV1.3_AppProfile.zip, March 2003.

[12] IMS Global Learning Consortium, "IMS Simple Sequencing Information and Behavior Model Version 1.0 Final Specification", http://www.imsglobal.org/simplesequencing/ssv1p0/imsss_infov1p0.html, March 2003.

[13] Advanced Distributed Learning, "The SCORM Implementation Guide: A Step-by-Step Approach", http://www.adlnet.org/adldocs/Documents/SCORM_IG.pdf, November 2002.

[14] Learning Systems Architecture Laboratory, Carnegie Mellon University, "SCORM Best Practices Guide for Content Developers", http://www.lsal.cmu.edu/lsal/expertise/projects/developersguide/developersguide/guide-v1p0-20030228.pdf, February 2003.

[15] Learning Systems Architecture Laboratory, Carnegie Mellon University, "SCORM Simple Sequencing Templates and Models", http://www.lsal.cmu.edu/lsal/expertise/projects/developersguide/sstemplates/templates-v1p0-20030228.pdf, February 2003.

[16] Nakabayashi K., Kubota Y., Yoshida H., Shinohara T., "Design and implementation of WBT system components and test tools for WBT content standards", In *Proceedings of 1st IEEE International Conference on Advanced Learning Technologies*, August 2001.

[17] Shih T.K., Wen-Chih Chang, Lin N.H., Lin L.H., Hun-Hui Hsu, Ching-Tang Hsieh, "Using SOAP and .NET web service to build SCORM RTE and LMS", in *Proceedings of 17th IEEE International Conference on Advanced Information Networking and Applications*, 2003.

[18] Bohl O., Schellhase J., Sengler R., Winand U., "The sharable content object reference model (SCORM) - a critical review", in *Proceedings of 1st IEEE Conference on Computers in Education*, 2002.

[19] Qu C., W. Nejdl, "Towards Interoperability and Reusability of Learning Resource: a SCORM-conformant Courseware for Computer Science Education", in *Proceedings of 2nd IEEE International Conference on Advanced Learning Technologies*, IEEE Computer Society Press, September 2002.

[20] Qu C., W. Nejdl, "Towards Open Standards: the Evolution of an XML/JSP/WebDAV Based Collaborative Courseware Generating System", in *Proceedings of the 1st International Conference on Web-based Learning*, August 2002.

[21] Canada's Department of National Defense, "SCORM Dynamic Appearance Model White Paper", February 2002.

[22] Edward R. Jones, "Implications of SCORM and Emerging E-learning Standards On Engineering Education", in *Proceedings of the 2002 ASEE Gulf-Southwest Annual Conference*, March 2002.

[23] Advanced Distributed Learning, "SCORM Version 1.2 Conformance Requirements", http://www.adlnet.org/ADLDOCS/Documents/SCORM_1.2_ConformanceReq.pdf, February 2002.

[24] Scott Wilson, "Experts question SCORM's pedagogic value", *CETIS Article*, http://www.cetis.ac.uk/content/20020802112525, August 2002.

[25]] Aviation Industry CBT Committee, "AICC Icon Standards: User Interface Version 1.0", http://www.aicc.org/docs/AGRs/agr009.zip, June 1996.

[26] Wilbert Kraan and Scott Wilson, "Dan Rehak: "SCORM is not for everyone"", *CELTIS Article*, http://www.cetis.ac.uk/content/20021002000737, October 2002.

[27] Juliette, "Is SCORM coming out ahead?", http://www.september15.net/log_september15_archive/000100.html#000100, October 2002.

[28] Edward Welsch, "SCORM: Clarity or Calamity?", *Online Learning Magazine*, http://www.onlinelearningmag.com/onlinelearning/magazine/article_display.jsp?vnu_content_id=1526769.

[29] http://xml.apache.org/xindice/