Designing an open component for the Web-based learning content model

Xin-hua Zhu
Department of Computer Science
Guangxi Normal University
Guilin 541004, China
Tel: +86-773-5848991
zxh429@263.net

ABSTRACT

On the basis of analyzing the characteristics of content components in the current distance education technology specifications, this paper puts forward an Open Content Object model for the Web-based learning content by extending the Sharable Content Object (SCO) of the Sharable Content Object Reference Model (SCORM) which was established by the Advanced Distributed Learning (ADL) of the Office of the Secretary of Defense (OSD). In this components model, the Open Content Object has the function of requesting services and providing services by the messages passing mechanism which takes the Learning Management System(LMS) as scheduling center of messages, thus the content components can not only be aggregated to compose higher-level units of instruction, but also can form the interoperable associational relationship, thereby it can provide a more flexible and effective approach to the design of the interoperable aggregation and sequence of the Web-based learning content.

Keywords

Distance education technology specification, Content components, Sharable Content Object, Open Content Object

1. Introduction

Nowadays, in order to accelerate the development of the distance education based on WWW and to implement the share and reuse of the learning resources, many international organizations are researching on and establishing the standards and specifications of the distance education. Notably, among these initiatives are the Aviation Industry CBT Committees (AICC,1998),the IMS Global Learning Consortium (IMS,1997), the IEEE Learning Technology Standers Committee (IEEE/LTSC,1997), the W3C (1994), the Advanced Distributed Learning Initiative/ Sharable Content Object Reference Model(ADL /SCORM,1999) and the European Committee for Standardization/Information Society Standardization System (CEN/ISSS,1997). Each established specification of the distance education has a Content Model that describes the components used to build a learning experience from reusable learning resources; the Content Model also defines how these lower-level sharable, reusable learning resources are aggregated to compose higher-level units of instruction(ADL,2004a). Essentially there are two types of the Web-based content components in all Content Models supported by various specifications.

One is pure web page content component that doesn't need the Learning Management System(LMS) to track. This type of content component is a kind of basic learning content, and it can be supported by all the e-learning specifications, such as IMS's "webcontent" (IMS,2003a), ADL's "Assets" (ADL,2004a) and CEN/ISSS's "Information Resource" (CEN/ISSS,2001). Formally, this type of content components can be electronic representations of media, text, images, sound, web pages, assessment objects or other pieces of data that can be delivered to a Web client. All pure web page content components are only encoded in JavaScript and HTML or JavaScript and XML, and they do not need to communicate, using the API and Data Model, back to the Learning Management System (LMS), so LMS can launch these components by using the HTTP protocol directly.

Another type is advanced content components that can be tracked by the LMS. Advanced content components can communicate with an LMS when running, so that it can design different learning contents, approaches and styles for different learners according to their abilities and performances. Additionally, it gives the access to the new instructional technologies such as intelligent instruction and real-time instruction. This kind of content components can only be supported by several specifications, such as ADL/SCORM's "SCO" and AICC/CMI's "Lesson" (AICC,2000). The advanced learning components need to communicate, using the API and Data Model, back to the LMS, so LMS must launch them in a browser window that is a child window or a child frame of the LMS window that exposes the API Adapter as a Document Object Model (DOM) Object(ADL,2004b). The API Adapter must be provided by the LMS.

Being different from pure web page content components, advanced content components need a mechanism that enables them to communicate with an LMS in the run-time environment, but they can only communicate with LMS ,not with each other. Thus, both of the two types of content components are all self-contained in the function and are all made to be independent units, and they can not link with each other in any way; neither references nor messages are allowed between them (ADL,2004a;IMS2003a). The structure of the independent components is closed; they can be used even though breaking away from the context of learning content, so they are convenient to be aggregated, shared and reused. However, just because of the closed structure of the sharable content components, there exist some problems:

1. Content components only have complemental relationships with each other in function, and they can be aggregated to compose higher-level units, but they don't have helpful relationships with each other in running process, namely a content component cannot receive another's support when running.
2. In a typical object-oriented system, there should be a relationship of aggregation and association between objects (Rumbaugh,et al.,1991;Booch ,1993). But In the current specifications, as an instructional object, the content components are forbidden to establish the associational relationship by the messages passing mechanism. For this reason, the object-oriented characters of the learning system is not integrated.
3. The learning sequence cannot be implemented by the association of content components, so at present the simple interoperable learning sequence can only be organized by the structure of content aggregation, while the complex interoperable learning sequence like the flow of branched instruction must be designed in content package outside of content components, and the design should conform to the rule of Sequence Specifications based on XML(IMS,2003b; ADL,2004c). the XML (W3C, 1998) is a markup language that implements data exchange. Its strength is data describing(Birbeck,M.,et al., 2001); using it to control the learning flow is inefficient and complex.

To improve the above problems that are caused by the content components independent of learning content, this paper puts forward a new content components model on the basis of the SCORM's SCO (Sharable Content Object). The model breaks the closing of content components. Its basic idea is to extend the component-units of the learning content to be Open Content Objects(OCOs) that have the function of requesting services and providing services by the message passing mechanism. This open components model allows OCOs to form the interoperable associational relationship ,and makes the object-oriented characters of the learning systems more comprehensive, thereby it can provide a more flexible and effective approach to the design of the interoperable aggregation and sequence of Web-based learning content.

2. Open Content Object

Open Content Object (OCO) is an open content component that has the function of requesting services and providing services. The open characteristic of the OCO is evolved from the characteristic that ADL/SCORM's SCO can communicate with an LMS in the run-time environment. As the same as the program objects in C++ and Java programming, an OCO object can request other objects to provide services by sending messages to the LMS, thus the associational relationship between OCO objects is formed. OCO object may request other OCO objects to provide services when it offers services to others, and the learning experience can be completed in the process of continual object requesting and object serving.

To enable that OCO object can be moved independently between different LMSs, but not be moved together with the requested OCO object, the representation of the associational relationship between OCO object and its requested OCO object must be picked up from the OCO object's internal design, thus replacing a message that sends service request to the LMS. The message mapping about the requested OCO object's appointment is put off, and it is not expressed as an interoperable standardized manner in the content packages outside of OCO object until OCO is aggregated, ultimately the service request and response in a learning experience will be dynamically accomplished via the LMS . The figure 1 illustrates how an OCO object and its requested OCO are dynamically linked by the messages passing mechanism which takes the LMS as scheduling center of messages.

When issuing, an OCO must bind related meta-data to describe the listing of request services and the interface of service functions. Depending on the meta-data, the OCO requesting services and the OCO providing services in a content package can be designed by different developers, and an OCO object's services request is allowed to be responded by different OCO objects in different content packages. Thereby OCO object can be independently moved with a package from one LMS environment to another, and the 0associational relationship that is formed between OCOs by the messages passing mechanism is interoperable in different LMSs (shown in the figure 2).

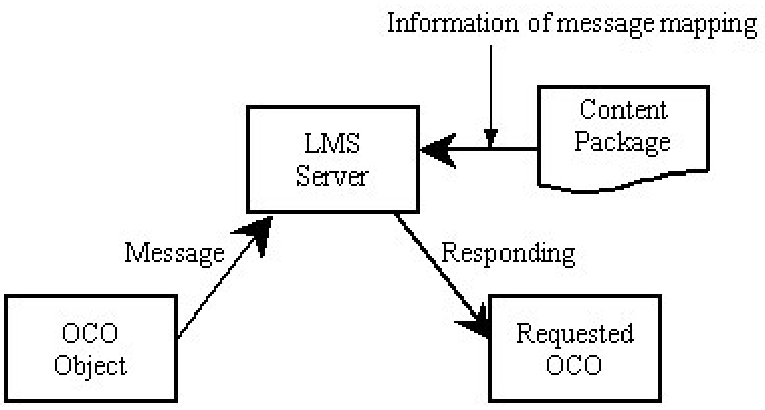


Figure 1. The process of dynamic link between an OCO object and its requested OCO

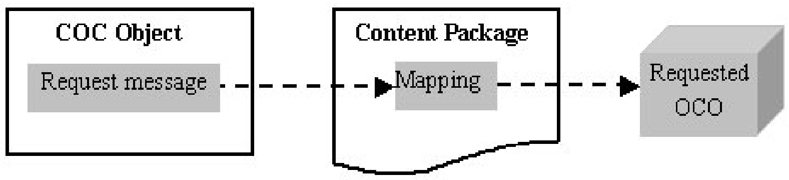
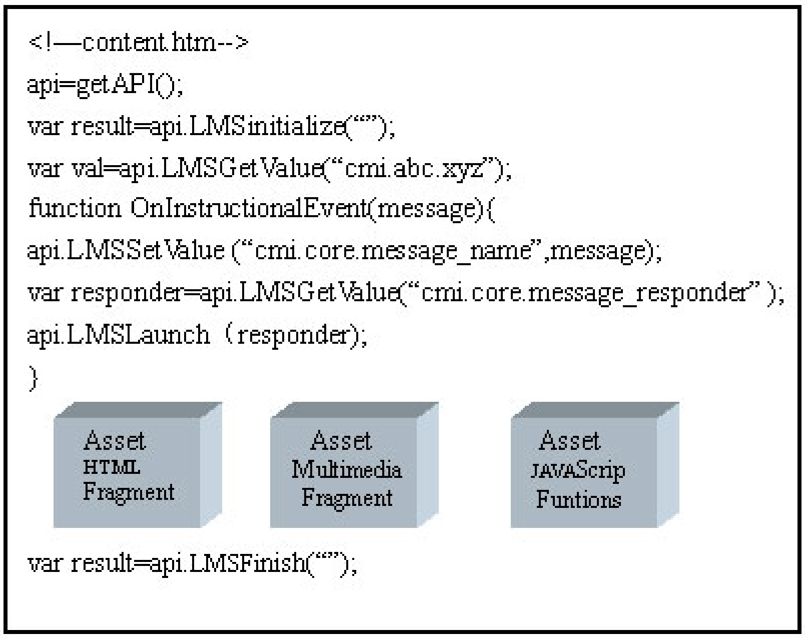


Figure 2. The interoperable associational relationship between OCOs

Being based on the Run-Time Environment in ADL/SCORM specification, the Run-Time Environment of OCO can be formed by extending SCORM's API adapter methods and communication data model, and OCO can be formed by using some properly extended API calls that send service request messages to the LMS in SCORM's SCO object. In the structure of OCO based on SCO shown in the figure 3, when instructional events that trigger the service request happened, OCO sends a message about service request to the LMS through extended API calls LMSSetValue ("cmi.core.message.name","messagename"), and meanwhile it requests the LMS to launch the service object which is dynamically appointed by the LMS through extended API calls LMSLaunch("responder").



```
<!—content.htm-->
api=getAPI();
var result=api.LMSinitialize("");
var val=api.LMSGetValue("cmi.abc.xyz");
function OnInstructionalEvent(message){
api.LMSSetValue ("cmi.core.message_name",message);
var responder=api.LMSGetValue("cmi.core.message_responder" );
api.LMSLaunch (responder);
}
```

```
var result=api.LMSFinish("");
```

Figure 3. OCO

3. Instances of the Application of OCO in the Design of Content Aggregation and Learning Sequence

In the present specifications, to absolutely insure the reusability of content components, external learning resources should not be referenced within components in direct or indirect manner and all components in content packages are equal and independent, and there is no container that contains other components. For these reasons, the object-oriented characters of the learning system are not integrated ,so the aggregation of learning content only can be designed outside of the content components according to the content packaging specification (IMS, 2003a; ADL,2004a), and the learning sequence must be designed outside of the content components by using the rules of sequence specifications (IMS,2003b; ADL,2004c). This causes the design of the content aggregation and the learning sequence to be complicated and ineffective. After designing an open component OCO for the Web-based learning content model, the learning contents can be aggregated through OCO containers and the learning sequences can be organized through the associational relationship between OCOs. This particular approach to design content aggregation and learning sequence has not been offered by any present specification.

In the following text, there are application instances of the use of OCO in the content aggregation and the learning sequence design.

3.1. An Instance of Aggregating Learning Contents through OCO Container

Suppose there are three knowledge points in the same section of a course and their learning content components respectively are OCO1, OCO2 and OCO3. Now those learning content components are required to be aggregated to compose a bigger sectional learning unit. This can be accomplished by designing another container OCO4 that contains these three components. It is shown in the figure 4.

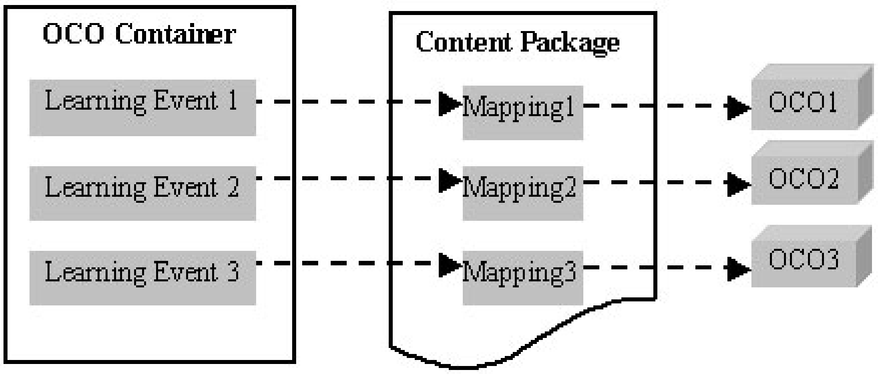


Figure 4. The structure of OCOs' container

The detail design steps are:

(1) Three learning events, which can be triggered by the action of some links, menu items or command buttons, are designed in the OCO4 container to control the navigational sequence of the three contained components.
(2) In the program processing Learning events within the OCO4 container, indirectly linked the three contained components by sending messages to the LMS.
(3) In the manifest file of content packages, only OCO4 will directly be the item of the aggregation structure, while OCO1, OCO2, OCO3 will only be three responders of OCO4's three messages, but not be the items of the aggregation structure. Therefore, the design of aggregation structure will be simplified.

3.2. An Instance of Organizing Branched Instruction Through the Associational Relationship between OCOs

After the learning process of an instructional unit is finished, different subsequent instructional units will be provided to the learners according to their grades. This is the so-called branched instruction, which is a basic and common instructional manner in Computer-Based Instruction (Kearsley, 1983). In the content design that conforms to IMS and ADL specifications, the interoperable flow of branched instruction can only be organized in content package according to the rules of sequence specifications, and the implementing process is very complex. After OCO is imported, the interoperable flow of branched instruction can be organized through the associational relationship between OCOs, and the implementing process is simple and perspicuous. For example, as for the branched instruction shown in the figure 5, it can be organized through the associational relationship shown between OCOs shown in the figure 6.

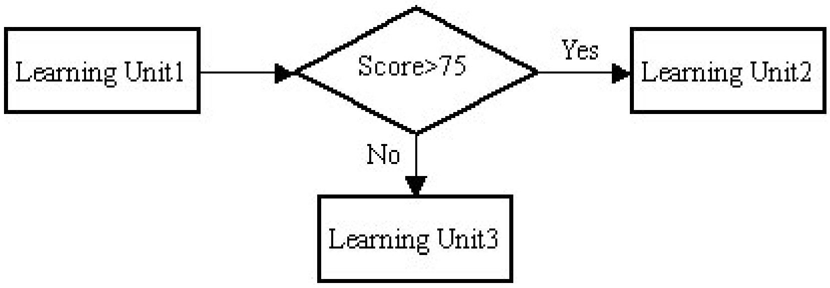


Figure 5. A flow of branched instruction

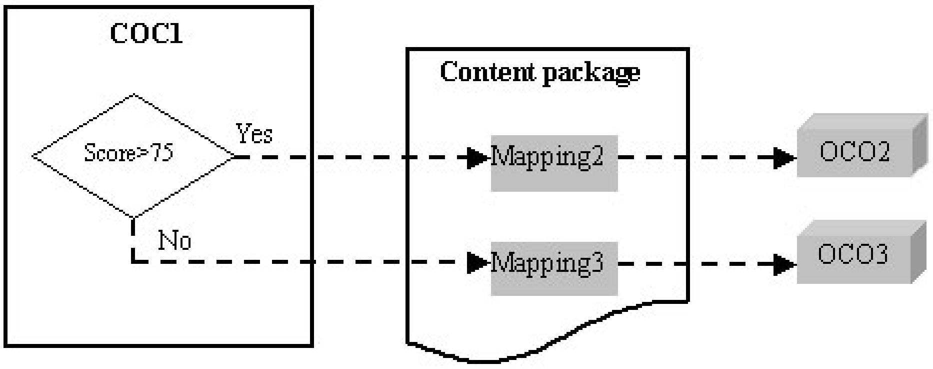


Figure 6. The OCOs' association relationships which are used for organizing branched instruction

The detail design steps are:

(1) The preceding learning unit and two succeeding learning units are respectively designed to be OCO1, OCO2 and OCO3.

(2) When the preceding learning object OCO1 is being designed, the following branched controlling sentences will be added at the end of OCO1's interior:

```
if (score>75) then
  send message1 to the LMS
else
    send message2 to the LMS
```

(3) In the content package, the responder of message1 will be mapped to be OCO2, and the responder of message2 will be mapped to be OCO3 by an extended element MessageMapping that is required for OCOs' packaging in a content package:

```
<MessageMapping>
      <mapping message=”MESSAGE1” responder=” OCO2” />
      <mapping message=”MESSAGE2” responder=” OCO3” />
</MessageMapping>
```

4. Conclusions

Designing content components to be open ones which have the associational relationship makes the interrelationship between content components more comprehensive, thereby the learning contents can be aggregated through OCO containers and the learning sequences can be organized through the associational relationships between OCOs. This particular approach to design the interoperable aggregation and sequence of Web-based learning content is flexible and effective and has not been offered by any present specification. It has two characteristics:

The design of content aggregation and learning sequence is partly implemented witin the OCO object. The reason is: When content aggregation and learning sequence are designed through OCO containers and associational relationships, the message sending, the navigational control of learning event and the control of learning flow are implemented within an OCO object through the advanced programming language like JavaApplet, JavaScript. The advanced programming language is inherently smart in flow control, so this approach is flexible and effective in the design of learning content aggregation and sequence.
The content aggregation and the learning sequence designed through OCO containers and associational relationships are interoperable. The reason is that the associational relationship between OCOs is formed by OCO sending service request message to the LMS, but not formed by OCOs' referencing directly with each other inside of the OCO, and the corresponding message mapping is described as an interoperable standardized manner in the content packages outside of the OCO objects

The Open Content Object is only a conceptual model yet.As for the application of the Open Content Object , it is prerequisite that a correlative reference model, which will be composed of content aggregation model and run-time environment, should be designed. Fortunately, the OCO Reference Model can be established based on ADL/SCORM. Now, the author is engaged in the related research work.

Acknowledgements

This work has been supported by the Guangxi Science Foundation of China under the contract number 0447034.

References

ADL (1999). *Sharable Content Object Reference Model (SCORM)*, retrieved September 23, 2004, from http://www.adlnet.org.

ADL (2004a). *SCORM Content Aggregation Model (CAM) Version 1.3*, retrieved September 23, 2004, from http://www.adlnet.org.

ADL (2004b). *SCORM Run-Time Environment (RTE) Version 1.3*, retrieved September 23, 2004, from http://www.adlnet.org.

ADL (2004c). *SCORM Sequencing and Navigation (SN) Version 1.3*, retrieved September 13, 2004, from http://www.adlnet.org.

AICC (1988). *Aviation industry CBT committees*, retrieved September 13, 2004, from http://www.aicc.org.

AICC (2000). *CMI001 Guidelines for Interoperability Version 3.4*, retrieved September 13, 2004, from http://www.aicc.org/.

Birbeck, M., Duckett, J., Gudmundsson, O. G., Kobak, P., Lenz, E., Livingstone, S., Marcus, D., Mohr, S., Pinnock, J., Visco., K., Watt, A., Williams, K., Zaev, Z., & Ozu, N. (2001). *Professional XML (Programmer to Programmer) (2nd Ed.)*, Indianapolis: Wrox Press.

Booch , G.(1993). *Object-oriented analysis and design with applications*, Massachusetts: Addison Wesley.

CEN/ISSS (1997). *CEN Information Society Standardization System*, retrieved September 13, 2004, from http://www.cenorm.be/ISSS.

CEN/ISSS (2001). *CEN Workshop Agreement (CWA) 14356*, retrieved September 13, 2004, from http://www.cenorm.be/ISSS.

IEEE/LTSC (1997). *IEEE Learning Technology Standers Committee*, retrieved September 13, 2004, from http://ltsc.ieee.org.

IMS (1997). *IMS Global Learning Consortium, Inc*, retrieved September 13, 2004, from http://www.imsglobal.org.

IMS (2003a). *IMS Content Packaging Specification, Version 1.1.3 Final Specification*, retrieved September 23, 2004, from http://www.imsglobal.org/content/packaging.

IMS (2003b). *IMS Simple Sequencing Information and Behavior Model, Version 1.0 Final Specification*, retrieved September 23, 2004, from http://www.imsglobal.org/simplesequencing.

Kearsley, G. (1983). *Computer-based training: a guide to selection and implementation Reading*, Massachusetts: Addison-Wesley.

Rumbaugh, J. R., Blaha, M. R., Premerlani, W., Eddy, F., & Lorensen, W. (1991). *Object-Oriented Modeling and Design*, New Jersey: Prentice Hall.

W3C (1998). *XML: eXtensible Markup Language Version 1.0*, retrieved September 13, 2004, from http://www.w3.org/XML.