Zhu, X. (2007). Extending the SCORM Specification for references to the Open Content Object. *Educational Technology & Society*, 10 (1), 248-264.

Extending the SCORM Specification for references to the Open Content Object

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

ABSTRACT

The Open Content Object (OCO), which was put forward in the author's previous paper, is an object that has the function of requesting services and providing services through the messages-passing mechanism, which relies on the Learning Management System (LMS) as the scheduling center of messages. The OCOs can offer a particular and flexible approach to designing the interoperable web-based content aggregation through its containers and organizing the learning sequence through the associational relationships between OCOs. This paper discusses how to extend the Sharable Content Object Reference Model (SCORM) Specification for applying the Open Content Object. The extensions to the SCORM Specification include extending SCORM's Content Aggregation Model and Run-Time Environment in light of the OCO's open characteristic. The paper also presents the XML binding and implementation of extending SCORM's Content Aggregation Model and some simple examples of using this extended reference model. The Open Content Object can be applied to the web-based learning contents through the extended SCORM specification presented by this paper, and it is hoped that the Open Content Object will offer some references for the next generation model of ADL/SCORM's content object.

Keywords

SCORM Specification, Content components, Content components reference model, Sharable Content Object, Open Content Object.

1. Introduction

Recently, in order to accelerate the development of the distance education based on the World Wide Web, many international organizations are researching and establishing the specifications for distance education. Notable among these initiatives are the AICC (1998), the IMS (1997), the IEEE/LTSC (1997), the W3C (1994), the CedMA (1991), the ADL (1999), the ARIADNE (1996), the CEN/ISSS (1997), the ISO/IEC JTC1 SC36 (1999) and the CETIS (2001). The interoperability and reusability are two common goals pursued by all the communities that are engaged in the technological specifications related to distance education. There are three levels of interoperability and reusability in all technological specifications related to distance education.

The first level is based on metadata, and the IEEE 1484.12.1 Learning Object Metadata (LOM) Standard (IEEE/LTSC, 2002) is the most typical specification. In this level, learning resources described by metadata can be systematically searched and retrieved for use and reuse. The information that describes a learning resource by metadata elements includes the name of learning resource, author, owner, terms of distribution, and format. The metadata specifications enable only the sharing and exchange of single learning resources across any technology-supported learning systems, but prevent aggregating several learning resources into an interoperable collection.

The second level is based on content aggregation, of which the most typical specification is IMS Content Packaging Specification (IMS, 2003b). Through the implementation of Content Packaging Specification, designers and implementers of instruction can aggregate learning resources to be an interoperable collection for the purpose of delivering a desired learning experience. In order to easily aggregate, share, and reuse learning resources, at present, IMS Content Packaging Specification requires that the structure of the independent learning resources be closed in a content package, so that learning resources can be used even though they break away from the context of learning content. However, because of the close structure of the sharable resources, the resources in a content package have only the relationships of functional complementarity with each other , namely they can be aggregated to compose higher-level units, but they do not provide service for each other during operation. Therefore, the object-oriented characters of learning systems are not integrated (Rumbaugh et al., 1991).

The third level is based on learning sequencing, of which the most typical specifications are IMS Simple Sequencing (SS), (IMS, 2003c), and ADL/SCORM Sequencing & Navigation (SN) (ADL, 2004d). At this level, designers and implementers of instruction can represent the intended behavior of an authored learning experience so that any Learning Management System (LMS) will sequence discrete learning activities in a consistent way. This level of interoperability and reusability is the most advanced and leads to new instructional technologies such as intelligent instruction and real-time guidance (Sleeman, 1982). This level is also the most complex and requires more flexible approaches to designing learning sequences in a content package. At present, for designing advanced learning sequences such as the branched learning sequence, both the IMS SS Specification and the SCORM SN lie on ADL/SCORM advancing Sharable Content Object (SCO)(ADL,2000), which is a new learning object that can be tracked by an LMS. The SCO can communicate with an LMS at run-time, so that different learning contents, approaches, and styles can be designed for different learners according to their abilities and performances. Because of the requirement that the structure of learning objects must be closed in a content package, SCOs can only communicate with LMS, but not with each other, so, at present, the interoperable learning sequence must be all designed in a content package that is outside of content components, and the design should conform to the rule of sequence specifications based on XML (IMS, 2002; IMS, 2003d; ADL, 2004e). The XML (W3C, 1998) is a markup language that implements data exchange. Its strength is data description (Birbeck, et al., 2001; Decker & Melnik, 2000); but using it to control the learning flow is inefficient and complex.

To reduce the above side effects, which are caused by the closed structure of content components, the author puts forward a new content-components model in a previous paper, which breaks through the closed structure of content components on the basis of the SCORM's SCO (Sharable Content Object) (Xinhua, 2005). Its basic idea is to extend the component-unit of the learning content to be an Open Content Object (OCO) that has the function of requesting services and providing services by the message-passing mechanism which relies on the Learning Management System (LMS) as the scheduling center of messages. The open characteristic of OCO is that the structure and function of an OCO can be extended by sending messages to LMS for requesting other OCOs to provide services. This open-components model allows content components to form the interoperable associational relationship by describing the message mapping about the requested OCO object's appointment in a content package, and makes the object-oriented characters of the learning systems more comprehensive and integrated. Therefore, this components model enables the learning sequence to be partly implemented within the OCO object by sending messages in Java scripts or Java applets, while the message mapping that appoints associational relationship between OCOs is described as an interoperable standardized manner in content packages outside of the OCO objects. So this approach to designing learning sequence is more flexible and effective than both the IMS SS and the SCORM SN specifications.

The Open Content Object model is only a conceptual one in the author's previous paper As for the applications of the Open Content Object, a correlative reference model, composed of a content aggregation model and a run-time environment, must be designed. This paper will present how to accomplish the extensions to SCORM Specification (ADL, 2004b) for the Open Content Object in light of the open characteristic of the Open Content Object.

2. Extending the SCORM Specification for the Open Content Object

Nowadays the definition and application of the specifications in distance education are implemented through Extensible Markup Language (XML) (W3C, 1998). Every definition of specification's format is described in a name space of XML schema. The XML has the inherent extensibility that is future-oriented compatibility (Martin et al, 2000; Walmsley, 2002). So, on the one hand, the specifications are allowed to refer to each other, e.g. the IMS Learning Resource Metadata Information Model (IMS, 2001a) refers to the LEEE/LTSC's Learning Object Metadata Standard, and the ADL/SCORM Specification refers to the IMS Content Packaging Specification (IMS, 2003b) and AICC Computer Managed Instruction Data Model (AICC, 2000a); on the other hand, every specification can be extended by software development groups according to their demands (IEEE/LTSC, 2002; IMS, 2003a), e.g. Wenchih et al(2004) propounded that "Enhancing SCORM metadata for assessment authoring in e-Learning", and Changtao & Wolfgang (2004) propounded that "Integrating XQuery-enabled SCORM XML Metadata Repositories into an RDF-based E-Learning P2P Network".

The open characteristic of the OCO is evolved from the characteristic that ADL/SCORM's SCO can communicate with a Learning Management System (LMS), so the Reference Model for the Open Content Object also consists of

an Aggregation Model based on SCORM Content Aggregation Model and a Run-Time Environment based on SCORM Run-Time Environment, and it extends the relevant SCORM's sub-specifications in light of the OCO's open characteristic. The extensions to the SCORM Specification mainly include the following points:

1. In order to describe the requesting services from others and the providing services for others in an OCO , two sub-elements ,<requirement> and <service>, are added to the <general> element of SCORM Metadata Specification.
2. In order to unify the naming of message and method that are used within all the OCOs in a content package, two sub-elements ,<message> and <method>, are added to the <resource> element of SCORM Content Packaging Specification.
3. In order to describe the message mappings between the OCOs in a learning experience, a <messagemapping> sub-element is added to the <item> sub-element of the <organization> element of SCORM Content Packaging Specification.
4. The OCO Launch Scheme is extended to be one that allows LMS to launch more than one OCO for every learner at a time.
5. In order to enable an OCO to request LMS to launch the service object for it, a function Launch (message_responder) is added to the SCORM Application Program Interface (API).
6. In order to enable an OCO to send the messages of the service request to the LMS, three mandatory elements, <message_name>, <message_responder> and <message_responsemethod>, are added to the SCORM Data Model.

3. Extending the SCORM Aggregation Model for the Open Content Object

As the SCORM specification, the Aggregation Model for the Open Content Object also consists of a Metadata Model and a Content Packaging Model. It is aimed to provide a common method of packaging, issuing and exchanging for OCO-based learning contents. The Aggregation Model for the Open Content Object can be established on the Metadata and Content Packaging of the SCORM Content Aggregation Model (CAM) Version 1.3 (ADL, 2004a), and it extends properly the relevant SCORM specifications in light of the specific description information that is needed when OCOs are aggregated.

3.1 Extensions to the SCORM Metadata Information Model

"Information about information" is called metadata (IMS 2001a; W3C, 2001a). "Describing the components with metadata facilitates the search and discovery of the components across systems. An LMS could use the metadata to give the learner information about the content organization (i.e., course, lesson, module, etc) (ADL, 2004a). The SCORM Metadata Information Model refers to the IEEE 1484.12.1-2002 Learning Object Metadata standard (IEEE/LTSC, 2002) and the IEEE 1484.12.3 Draft Standard for Extensible Markup Language Binding for Learning Object Metadata Data Model (IEEE/LTSC, 2004). The SCORM Metadata Information Model is made up of elements of ten kinds, which are <lom>, <general>, <lifeCycle>, <metaMetadata>, <technical>, <educational>, <rights>, <relation>, <annotation> and <classification>. The element of each kind describes the characteristics of certain aspect of about content components, among them the <general> element describes the resource as a whole. An OCO may be a receiver of services as well as a provider of services, so in the OCO resource metadata, two kinds of information, the services requested to others and the services provided to others, should be described respectively. Therefore, two sub-elements should be added to the <general> element in the extended SCORM metadata Specification; their formats are as follows:

< requirement > element

- Description: This element describes the characteristics and functions of a service requested by an OCO, and the name of the corresponding message.
- Multiplicity: This element should occur 0 or more times within a <general> element.
- Sub-elements:
    - <message>: This sub-element describes the name of the message sending to LMS when this service is requested within an OCO ,and it must occur 1 and only 1 time in a < requirement > element.
    - <langstring>: This sub-element describes the characteristics and the functions of this requested service.

< service > element

- Description: This element describes the functions of a service provided by an OCO and the method name of performing these functions.
- Multiplicity: This element should occur 0 or more time within a <general> element.
- Sub- elements:
    - <method>: This sub-element describes the method name of performing this service provided by an OCO ,and it must occur 1 and only 1 time in a < service > element.
    - <langstring>: This sub-element describes the functions of this service.

The <langstring> sub-element can occur 0 or more times within the < requirement> and < service > element. However, each langstring is required to contain a different xml:lang attribute (ISO, 2002). Generally, a typical OCO resource describes the requesting list and service interface in its perfromace through several <requirement> elements and <service> elements in its metadata when it is issued, so that the OCO resources can be correctly reused in different learning systems.

3.2 Extensions to the SCORM Content Packaging Model

The purpose of the content package is to provide a standardized way to exchange learning content between different systems or tools. The content package also provides a place for describing the structure (or organization) and the intended behavior of a collection of learning content (IMS, 2003b; ADL, 2004a). SCORM Content Packaging Model is a set of specific use examples, or application profiles, of the IMS Content Packaging Specification (ADL, 2004a).

A content package manifest conforming to SCORM Specification consists of Metadata about the package, an optional Organization section that defines content structure and behavior, and a list of references to the resources in the package.

The SCORM Content Packaging Model supports only the aggregation relationship between content components. While in an OCO's content package, the OCOs can not only form the aggregation relationship with each other, but also set up the association between themselves by message mapping. In order to realize the establishment of the association between OCOs, it is necessary to make correspondent extension to the Organization elements and Resource elements in the SCORM's content package manifest.

3.2.1 Extensions to the <resource> element

In a content package, the different OCOs may be designed by different developers, so the message names for requesting service and method names for responding the requests, which are used within the OCOs, may not be unified between different OCOs. The different messages or methods within different OCOs may have the same name, so all messages and methods must be renamed and unified in a content package. Therefore, two sub-elements, <message> and <method>, should be added to the <resource> element (IETF, 1998). Their formats are as follows:

<message> element

- Description: This element uses an identifier in the content package to rename a message within an OCO.
- Multiplicity: This element may occur 0 or more times in a <resource> element.
- Attributes:
    - Identifier: A label for this message, which is globally unique within the content package.
    - Messagenameref: A reference to the name of this message within an OCO.

<method> element

- Description: This element renames a method within an OCO using an identifier in the content package.
- Multiplicity: This element may occur 0 or more times in a <resource> element.
- Attributes:

- Identifier: A label for this method, which is globally unique within the content package.
- Methodnameref: A reference to the name of this method within an OCO.

Compared with the SCORM's SCO, the OCO is a newly increased learning resource. In order to enable an LMS to correctly identify, launch and track an OCO, the type of the <resource> element referring to an OCO must be indicated to be "OCO". For this purpose, the SCORM's resource type attribute must be extended, and a new resource type of "OCO" is required to be added.

3.2.2 Extensions to the <item> sub-element of the <organization> element

In the IMS content package and the SCORM content package, an <organization> element is responsible to describe the content structure and sequence rules in a learning experience, so the association between the two OCOs should also be described in an <item> sub-element of the <organization> element. Therefore, a sub-element <messagemapping> should be added to the <item> element so as to describe the message mappings and to represent the association between the two OCOs in a learning experience. Its format is as follows:

<messagemapping> element

- Description: This element describes the message mappings between the OCO resource referred to in an <item> element and other OCOs.
- Multiplicity: This element may occur 0 or 1 time in an <item> element.
- Sub-element :<mapping>.

<mapping> Sub-element

- Description: This element describes a message mapping between the OCO resources.
- Multiplicity: This element may occur 1 or more times in a <messagemapping> element.
- Attributes:
  - Message: A reference to the identifier of a requesting message sent by the OCO resource referred to in this <item> element.
  - Responder: A reference to the identifier of the OCO resource that responds to the requesting message.
  - Responsemethod: A reference to the identifier of the method that really responds to this requesting message within the responding OCO.

3.3 The XML binding and implementation of the extended SCORM Aggregation Model

The Aggregation Model for the Open Content Object is an extension to SCORM's Metadata Specification and Content Packaging Specification. The method to implement these extensions (IMS, 2003a; IEEE/LTSC, 2004; W3C, 2001b; Walmsley, 2002) is as follows:

1. The extensions to the SCORM Metadata Specification and to the Content Packaging Specification are respectively defined in two different name spaces. Therefore, two XML Schema files should be built up to define the corresponding elements.
2. In an OCO's content package, when the extended element is used, a prefix representing its name space must be appended.

The implementation of the extensions to the SCORM Content Packaging Specification and its use will be introduced as an instance in the following text. The implementation of the extensions to the SCORM Metadata Specification is analogous with the instance.

3.3.1The implementation of the extensions to the SCORM Content Packaging Specification

The extended elements of SCORM Content Packaging Specification are defined in a XML schema file OCOCP.xsd (ADL, 2004e ;Walmsley, 2002; W3C, 2001b; Martin et al., 2000), which contains a hypothetical schema name

space: “http://www.OCORM.org/xsd/ococp_v1p0” . The content of OCOCP.xsd schema file is listed and shown in the Appendix A.

3.3.2 How to use the extended elements in an OCO’s content package

In an OCO’s content package, all the elements that are used respectively adhere to three different name spaces (IMS, 2001b; IMS, 2003a; W3C, 2001b; Walmsley, 2002):

1. The name space of IMS Content Packaging Specification.
2. The name space of the elements extended by SCORM Specification to the IMS Content Packaging Specification.
3. The name space of the elements extended by OCO Aggregation Model to the IMS Content Packaging Specification.

In an OCO’s content package, only the name space of IMS Content Packaging Specification is defined as the default name space. Therefore, when the elements extended by the OCO Aggregation Model are used, a prefix “ococp” representing the name space of the OCO Aggregation Model must be appended. An instance of the OCO Content Aggregation Model is given in Appendix B.

4. The Run-Time Environment for Open Content Object

The SCORM Run-Time Environment (RTE) provides an interoperable method for its Sharable Content Object(SCO) to run between multiple Learning Management Systems (LMSs), including a common way to start SCO learning resources, a common mechanism for SCO learning resources to communicate with an LMS and a predefined language or vocabulary forming the basis of the communication (ADL, 2004c). The Run-Time Environment of Open Content Object (OCO) is based on the SCORM Run-Time Environment; and it extends properly the SCORM RTE in light of the specific message passing mechanisms which is needed by the OCOs to implement their associational relationship. The structure of the OCO Run-Time Environment is shown in Figure 1.

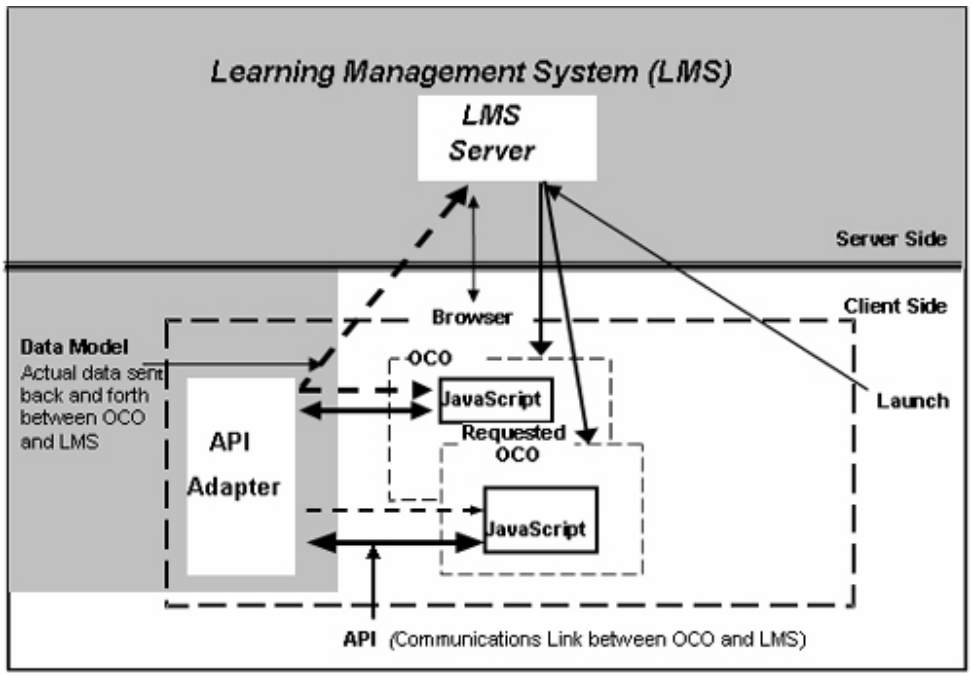

Figure 1. The OCO Run-Time Environment based on SCORM’s

4.1 Extensions to the SCORM Launch Scheme

The SCORM Launch Scheme requires that an LMS only launch one SCO at a time and that only one SCO is active at a time. The SCORM Launch Scheme also requires that only LMSs may launch SCOs, SCOs may not launch other SCOs (ADL, 2004c).

In the extended SCORM Specification for OCO, an OCO can request other OCO objects to provide services at run-time, so the OCO Launch Scheme should be extended to be one that allows an LMS to launch more than one OCO for every learner at a time and maintains an independent communication data structure for every launched OCO in the LMS. But among the OCOs that were launched at a time, only one OCO is still in active-state, and others are in interrupted inactive-state, so that the LMS can manage and track those OCOs. Therefore, the multi-OCOs that have been launched in learning system will be executed in turn in an interruption-based serial way, but not in a parallel way.

The OCO run-time environment still keeps that only LMSs can launch OCOs, and OCOs can not launch other OCOs . When an OCO requests another OCO to provide services, it must send a message to the LMS through an API calls, then the LMS launches the requested OCO and interrupts the instructional activity of the current OCO. This Scheme of launching the requested OCO is very important to keep the interoperability and reusability of the associational relationship between OCOs.

4.2 Extension to the SCORM Application Program Interface

The SCORM is based directly on the run-time environment functionality defined in AICC's CMI001 Guidelines for Interoperability document (AICC, 2000b). The functions of the SCORM Application Program Interface (API) Adapter object are threefold (ADL, 2004c):

- Execution State: Two of the API functions, Initialize() and Terminate(), handle execution state.
- State Management: The API has three functions that are used to handle errors. They are: GetLastError(), GetErrorString(errornumber) and GetDiagnostic(parameter).
- Data Transfer: The remaining three API functions are used to transfer data to and from an LMS: GetValue(data model element), SetValue(data model element, value) and Commit().

The API Adapter object of OCO's RTE still keeps the above functions, but it is required to add a function Launch(message_ responder) that enables an OCO to request the LMS to launch a service object. The function's format is as follows:

Launch (Parameter)

- Description: This function allows an OCO to request the LMS to launch a service object.
- *Syntax:* Launch(parameter).
- Parameter: A reference to the identifier of the OCO that responds the current message. The referred identifier can be returned from the API calls of GetValue("cmi.message_responder") .
- Return Value: String representing a Boolean:
    - "true" result indicates that the Launch () was successful.
    - "false" result indicates that the Launch () was unsuccessful.

4.3 Extensions to the SCORM Data Model

SCORM Run-Time Environment Data Model derived directly from the AICC CMI Data Model described in the AICC CMI Guidelines for Interoperability. All data model elements described by SCORM are required to be implemented and their behaviors supported by an LMS. All data model elements are optional for use by SCOs. SCOs are required only to use the API functions Initialize("") and Terminate(""); they are not required to use SetValue() or GetValue(). SCOs may be very, very small and not designed to be tracked in detail. However, if they are to be tracked, they must conform to a common data model for reusability across multiple LMS environments (ADL, 2004c).

The OCO Run-Time Environment Data Model keeps the structure and data elements of the SCORM Data Model, and in order to enable an OCO to get the mapping information of the current service request message form the LMS, it is required to add three mandatory elements, message_name, message_responder and message_ responsemethod, into the SCORM Data Model. Their definitions, usages and data types are as follows:

cmi.message_name

- Definition: Normally, the official name used for the message within the OCO.
- Usage: Used to represent the message official name within the OCO.
- Data Type: CMIString255.
- Supported API calls: GetValue(), SetValue().
- LMS Behavior:
    - This data model element is mandatory and shall be implemented by the LMS as read/write.
    - When an OCO object set this data element through a SetValue() API calls, the OCO needs only to send the name of message to be used within the OCO. The LMS is responsible for translating this message name within the OCO into a message identifier uniformly nominated within the content package. The LMS achieves this translation by querying the message mapping described in the content package.

cmi.message_ responder

- Definition: Unique alpha-numeric code or identifier that refers to an OCO object of the LMS system.
- Usage: Used to represent the identifier of the OCO that responds to the current message.
- Data Type: CMIIdentifier.
- Supported API calls: GetValue().
- LMS Behavior:
    - This data model element is mandatory and shall be implemented by the LMS as read-only.
    - The LMS is responsible for initializing the cmi.message_ responder data model element based on the result that LMS queries the corresponding <messagemapping> element in the content package . The LMS queries this <messagemapping> element according to the current value of the cmi.message_name data model element.

cmi.message_ responsemethod

- Definition: Normally, the official name used for the method within the OCO.
- Usage: Used to represent the name of the method that really responds to the current message in the OCO.
- Data Type: CMIString255.
- Supported API calls: GetValue().
- LMS Behavior:
    - This data model element is mandatory and shall be implemented by the LMS as read-only.
    - The LMS is responsible for initializing the cmi.message_ responsemethod data model element based on the result that LMS queries the corresponding <messagemapping> element and <method> element in the content package according to the current value of the cmi.message_name data element.
    - GetValue():The LMS shall return a method name within the OCO, and this method is the responder of the current message.

4.4 The state transitions of the OCO requesting service and the OCO responding service

An OCO shall send a message to the LMS when requesting another OCO to provide services at run-time. After receiving the message, the LMS shall query the message mapping information in the content package to locate and launch the OCO that responds to the request, and meanwhile, the current OCO will be interrupt by the LMS, and after the responding OCO has ended，it will be resumed. Figure 2 and Figure 3 respectively shows the transitions of the state between the OCO requesting service and the OCO responding service.

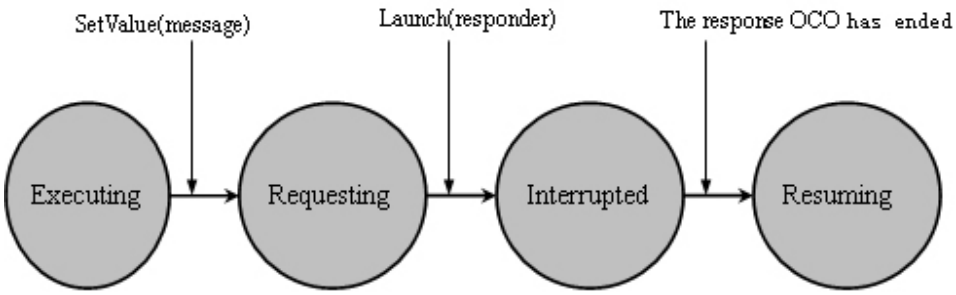

Figure 2. The state transitions of the OCO requesting service

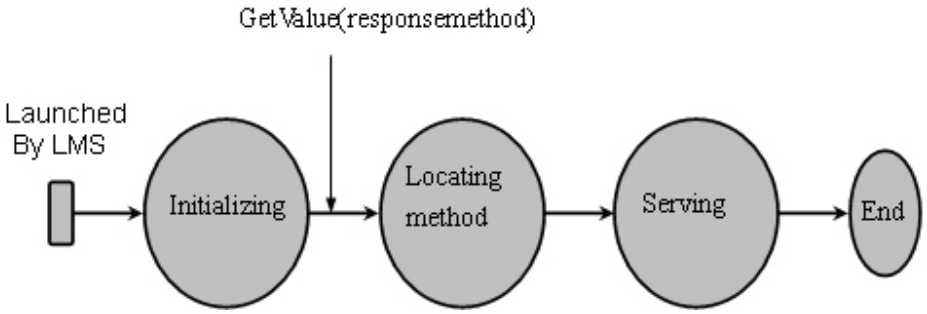

Figure 3. The state transitions of the OCO responding service

4.5 An instance of the implementation of designing the service request and service response within the OCO

In the process of designing an OCO, when an OCO object needs other's service in the handling procedure of an instructional events, at first it sends a message that requests the service to the LMS through a SetValue (message_name) API calls, then gets the responder of the message from the LMS through a GetValue(message_responder) API calls, and finally it requests the LMS to launch the responding object through a Launch(message_responder) API calls. These can be implemented by the following fragment of JavaScript program(Gibbons & Fairweather, 1998).

```
function OnInstructionalEvent(message)
{
  api.SetValue ("cmi.message_name","message");
     //This calls shall set the data model element of cmi.message_name into
       the supplied message name. The LMS shall translate this message name
       within the OCO into the message identifier uniformly nominated within
       the content package by queriying the <message> element in the
       <resource> element of the content package
  var responder=api.GetValue("cmi.message_responder");
  if (responder != NULL)
     api.Launch(responder);
  else
     //If there is no object to provide response within the content package,
       the responding code should be provided by this OCO
}
```

The OCO that responds to a service request, in its internal procedure for initialization, firstly gets the name of the method that responds to this request from the LMS by a GetValue(message_responsemethod) API calls, and then

performs the service by calling the corresponding method. These can be implemented by the following fragment of JavaScript program (Lewis, 1987).

```
function initialization
{
 var responsemethod = api.GetValue("cmi.message_ responsemethod");
   //This calls shall return the name of the method that responds to the
     current service request from the LMS .
 if (responsemethod == NULL) defaultmethod();
    //If there is no service request, the OCO will provide the default
      learning content for the learner directly. The OCOs that don't
      provide any service for other will start operating from this entry
 else if (responsemethod == "methodname1") methodname1();
    //The OCO shall call the corresponding method to perform the requested
       service
 else if (responsemethod == "methodname2") methodname2();
  . . .
 else if (responsemethod == "methodnameN") methodnameN();
}
```

In fact, the OCOs that provide the services for other and the OCOs that don't provide any service for other have the same structure in the initializing procedure. According to different conditions of launch, an OCO may provide the default learning content for the learner directly, or it may call some method to respond to a service request of another OCO.

5. Conclusions

Through the extended SCORM specification presented by this paper, the OCO can be applied on the following two aspects:

1. Designers can use the containers of OCO to aggregate learning contents.
2. Designers can use the associational relationship between OCOs to organize learning sequences.

In the IMS and SCORM Content Packaging specifications, to absolutely ensure the reusability of content components, the content components can not refer to any external learning resources in direct or indirect way and all components in a content package are equal and independent, and there is no container that contains any other components. For these reasons, the object-oriented character of the learning system is not completed and integrated, so the aggregation of learning content can be only designed outside of the content components according to the content packaging specification (IMS, 2003a; ADL, 2004a), and the learning sequence also can be only designed outside of the content components according to the rules of sequence specifications (IMS, 2003c; ADL, 2004d). This causes the design of the content aggregations and learning sequences to be complicated and inefficient. After the Open Component Object (OCO) is designed for the web-based learning content model, the learning contents can be aggregated through the OCO's containers, and the learning sequences can be organized through the associational relationship between OCOs. This particular approach to designing the content aggregations and the learning sequences can't be found in the present specifications. Appendix C and Appendix D give two application instances to show the use of OCO in the design of the content aggregation and learning sequence. But because of the associational relationship between OCOs, the OCO content components model will weaken the independency of learning resources, and the OCO objects are required to map much information when being aggregated within a content package and to bind much information of metadata when being moved with a package from one LMS environment to another.

Using containers to aggregate the resource components also occurs in the MPEG-21 Digital Item Declaration (DID)(ISO/IEC, 2005), which is a digital resources packaging approach established by the ISO/IEC in the multimedia framework (MPEG-21)(ISO/IEC, 2004) .MPEG-21 DID offers a general Data Model that describes a set of abstract terms and concepts for defining Digital Items. Within this model, a Digital Item is the digital representation of a "work" (e.g., a digital music album or an e-book), and a Container is defined to be a grouping of

Containers and/or Items; designers can use the Containers and these groupings of Items and Containers to form logical packages (ISO/IEC, 2005). There is an essential difference between the containers in the MPEG-21 DID and the containers of OCO. The containers in the MPEG-21 DID are formed by statically describing their components of items before being delivered to the users, and they are aimed to decompose a resource component to be several smaller units of items and to reduce the redundancy of resource aggregation. Whereas the OCO containers are formed by dynamically sending messages to the LMS at run-time, and they are aimed to increase the flexibility of the content aggregation and to simplify its structure.

At present, many research groups and corporations have extended the SCORM Specification when appling it to software systems or software development tools. For example, Wenchih et al (2004) extended SCORM metadata according to an assessment analysis model for e-Learning operations, and incorporated the measured aspects of the following list into the SCORM metadata description at the question cognition level, the item difficulty index, the item discrimination index, the questionnaire style and the question style, and finally implemented an assessment authoring system based on extended assessment metadata to help teachers in authoring examination. Macromedia (2002) also designed a Wrapper Extension for Dreamweaver MX to support and extend the SCORM Run-Time Environment; the Wrapper Extension can automatically inserts all necessary JavaScript code and HTML tag attributes to automatically find the API object provided by the management system. It also calls the Initialize and Finish methods of the object per the ADL specification requirements. The Wrapper Extension extends the Data Model of SCORM Run-Time Environment by adding an optional element that represents the completion status of a lesson. As for, all extensions to SCORM specification are only based on the some or other aspect of the SCORM specification. , and as the author known ,there is no paper like this one extends the SCORM Content Object, SCORM Aggregation Model, the SCORM Launch Scheme and the SCORM Application Program Interface one by one.

In this paper, the messages passing mechanism, which implements the associational relationship between OCOs, is implemented by extending the Application Program Interface (API) and Data Model of the SCORM Run-Time Environment Specification. In fact, the IMS Shareable State Persistence Information Model (IMS, 2004a; IMS, 2004b; IMS, 2004c) can also implement the messages passing mechanism.

The Open Content Object can be applied to the web-based learning contents through the extended SCORM specification presented by this paper, and the Open Content Object may offer some references for the next generation model of ADL/SCORM's content object. In future, the author will focus on how to design a particular Flow Control Object based on the Open Content Object and how to establish a Learning Sequencing Model based on the Flow Control Object.

Acknowledgements

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

References

ADL (1999). *Advanced distributed learning*, retrieved December 13, 2006, from http://www.adlnet.org.

ADL (2000). *Sharable Content Object Reference Model (SCORM) Version 1.0*, retrieved December 13, 2006, from http://www.adlnet.org.

ADL (2004a). *SCORM Content Aggregation Model (CAM) Version 1.3*, retrieved December 13, 2006, from http://www.adlnet.org.

ADL (2004b). *Sharable Content Object Reference Model (SCORM) Version 1.3*, retrieved December 13, 2006, from http://www.adlnet.org.

ADL (2004c). *SCORM Run-Time Environment Version (RTE) 1.3*, retrieved December 13, 2006, from http://www.adlnet.org.

ADL (2004d). *SCORM Sequencing and Navigation(SN) Version 1.3*, retrieved December 13, 2006, from http://www.adlnet.org.

ADL (2004e). *SCORM XML Controlling Document - SCORM CAM Version 1.3 Content Packaging Extensions XML XSD Version 1.0*, retrieved December 13, 2006, from http://www.adlnet.org.

AICC (1988). *Aviation industry CBT committees*, retrieved December 13, 2006, from http://www.aicc.org.

AICC (2000a). *Computer Managed Instruction (CMI) Data Model*, retrieved December 13, 2006, from http://www.aicc.org/.

AICC (2000b). *CMI001 Guidelines for Interoperability Version 3.4*, retrieved December 13, 2006, from http://www.aicc.org/.

ARIADNE (1996). *Alliance of remote instructional authoring & distribution networks or Europe*, retrieved December 13, 2006, from http://ariadne.unil.ch/project/main.content.html.

Birbeck, M., Ozu, N., Duckett, J., Watt, A., Mohr, S., Gudmundsson, O. G., Duckett, J., Watt, A., Mohr, S., Williams, K., & Mani, R. (2001). *Professional XML (Programmer to Programmer) (2nd Ed.)*, Indianapolis: Wrox Press.

CedMA (1991). *Computer education management association*, retrieved December 13, 2006, from http://www.cedma.org.

CEN/ISSS (1997). *CEN Information Society Standardization System*, retrieved December 13, 2006, from http://www.cenorm.be/ISSS.

CETIS (2001). *The Centre for Educational Technology Interoperability Standards*, retrieved December 13, 2006, from http://www.cetis.ac.uk/.

Changtao Q., & Wolfgang N. (2004). Integrating XQuery-enabled SCORM XML Metadata Repositories into an RDF-based E-Learning P2P Network. *Educational Technology & Society, 6* (4), 30-47.

Decker, S. & Melnik, S. (2000). The Semantic Web: The roles XML and RDF. *IEEE Internet Computing, 4* (9), 63-73.

Gibbons, A. S., & Fairweather, P. G. (1998). *Computer-based Instruction: Design and Development*, Englewood-Cliffs, NJ: Educational Technology Publications.

IEEE/LTSC (1997). *IEEE Learning Technology Standers Committee*, retrieved December 13, 2006, from http://ieeeltsc.org/.

IEEE/LTSC (2002). *IEEE 1484.12.1:Learning Object Metadata (LOM) Standard*, retrieved December 13, 2006, from http://ieeeltsc.org/wg12.

IEEE/LTSC (2004). *IEEE 1484.12.3: Draft Standard for Extensible Markup Language (XML) Binding for Learning Object Metadata Data Model*, retrieved December 13, 2006, from http://ieeeltsc.org/wg12.

IETF (1998). *IETF RFC 2396:1998, Universal Resource Identifiers (URI): Generic Syntax*, retrieved December 13, 2006, from http://www.ietf.org/.

IMS (1997). *IMS Global Learning Consortium*, retrieved December 13, 2006, from http://www.imsglobal.org.

IMS (2001a). *IMS Learning Resource Metadata Information Model, Version 1.2.1 Final Specification*, retrieved December 13, 2006, from http://www.imsglobal.org/metadata.

IMS (2001b). *IMS Learning Resource Metadata XML Binding, Version 1.2.1 Final Specification*, retrieved December 13, 2006, from http://www.imsglobal.org/metadata.

IMS (2002). *IMS Simple Sequencing XML Binding Version 1.0 Public Draft Specification*, retrieved December 13, 2006, from http://www.imsglobal.org/simplesequencing.

IMS (2003a). *IMS Content Packaging Best Practice Guide ,Version 1.1.3 Final Specification*, retrieved December 13, 2006, from http://www.imsglobal.org/content/packaging.

IMS (2003b). *IMS Content Packaging Specification ,Version 1.1.3 Final Specification*, retrieved December 13, 2006, from http://www.imsglobal.org/content/packaging.

IMS (2003c). *IMS Simple Sequencing Information and Behavior Model,Version 1.0 Final Specification*, retrieved December 13, 2006, from http://www.imsglobal.org/simplesequencing.

IMS (2003d). *IMS Simple Sequencing Best Practice and Implementation Guide Version 1.0 Final Specification*, retrieved December 13, 2006, from http://www.imsglobal.org/simplesequencing.

IMS (2004a). *IMS Shareable State Persistence Information Model Version 1.0 Final Specification*, retrieved December 13, 2006, from http://www.imsglobal.org/content/packaging.

IMS (2004b). *IMS Shareable State Persistence XML Binding Version 1.0 Final Specification*, retrieved December 13, 2006, from http://www.imsglobal.org/content/packaging.

IMS (2004c). *IMS Shareable State Persistence Best Practice and Implementation Guide Version 1.0 Final Specification*, retrieved December 13, 2006, from http://www.imsglobal.org/content/packaging.

ISO (2002). *ISO 639-2: Codes for the representation of names of languages*, retrieved December 13, 2006, from http://www.iso.org/iso/en/ISOOnline.frontpage.

ISO/IEC (2004). *ISO/IEC 21000-1: Multimedia framework (MPEG-21)*, retrieved December 13, 2006, from http://www.itscj.ipsj.or.jp/sc29/29w42911.htm#MPEG-21.

ISO/IEC (2005). *ISO/IEC 21000-2: MPEG-21 Digital Item Declaration (DID)*, retrieved December 13, 2006, from http://www.itscj.ipsj.or.jp/sc29/29w42911.htm#MPEG-21.

ISO/IEC JTC1 SC36 (1999). *ISO/IEC JTC1 SC36 Standards for: information technology for Learning, Education, and Training*, retrieved December 13, 2006, from http://jtc1sc36.org/index.html.

Lewis, B. (1987). *Computer Based Training*, New York: Parthenon.

Macromedia (2002). *SCORM Runtime Wrapper Extension for Dreamweaver MX*, retrieved December 13, 2006, from http://www.macromedia.com/resources/elearning/extensions/dw_ud/scorm.html.

Martin, D. Birbeck, M., Kay, M., Livingstone, S., Mohr, S., Pinnock, J., Loesgen, B., Ozu, N., Seabourne, M., & Baliles, D. (2000). *Professional XML*, Indianapolis: Wrox Press.

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

Sleeman, D., & Brown, J. S. (1982). *Intelligent Tutoring Systems*, New York, NY: Academic Press.

W3C (1994). *World Wide Web Consortium*, retrieved December 13, 2006, from http://www.w3.org.
W3C(1998). *XML: eXtensible Markup Language Version 1.0*, retrieved December 13, 2006, from http://www.w3.org/XML.

W3C (2001a). *W3C Metadata Activity*, retrieved December 13, 2006, from http://www.w3.org/Metadata/Activity.html.

W3C (2001b). *XML Schema Part 2: Datatypes*, retrieved December 13, 2006, from http://www.w3.org/XML

Walmsley, P. (2002).*Definitive XML Schema*, Indianapolis: Prentice Hall.

Wenchih, C., Huihuang, H., Smith, T. K., & Chunchia, W. (2004). Enhancing SCORM metadata for assessment authoring in e-Learning. *Journal of Computer Assisted Learning, 20* (4), 305-316.

Xinhua, Z. (2005). Designing an open component for the Web-based learning content model. *Educational Technology & Society, 8* (2), 118-124.

APPENDIX A

OCOCP.xsd File Listing:

```
<xsd:schema targetNamespace=”http://www.OCORM.org/xsd/ococp_v1p0”
    xmlns=”http://www.OCORM.org/xsd/ococp_v1p0”
    xmlns:xsd=http://www.w3c.org/2001/XMLSchema>
  <xsd:attribute name="ocormType">
     <xsd:simpleType>
        <xsd:restriction base="xs:string">
              <xsd:enumeration value="oco" />
             <xsd:enumeration value="sco" />
             <xsd:enumeration value="asset" />
        </xsd:restriction>
     </xsd:simpleType>
  </xs:attribute>
  <xsd:simpleType name=”namerefType”>
     <xsd:restriction base="xsd:string">
        <xsd:maxLength value="2000"/>
     </xsd:restriction>
   </xsd:simpleType>
  <xsd:element name="message" type="messageType" />
  <xsd:element name="method" type=" methodType" />
  <xsd:element name="messagemapping" type=" messagemappingType" />
  <xsd:complexType name="messageType">
    <xsd:attribute name="identifier" type="xsd: ID" use="required" />
    <xsd:attribute name="messagenameref" type="namerefType" use="required" />
  </xsd:complexType>
  <xsd:complexType name="messageType">
    <xsd:attribute name="identifier" type="xsd: ID" use="required" />
    <xsd:attribute name=" methodnameref " type="namerefType" use="required" />
  </xsd:complexType>
  <xsd:complexType name="mapType">
    <xsd:attribute name="message" type="namerefType" use="required" />
    <xsd:attribute name="responder" type="namerefType" use="required" />
    <xsd:attribute name="responsemethod" type="namerefType" use="required" />
   </xsd:complexType>
   <xsd:complexType name="messagemappingType">
    <xsd: complexCntent>
       <xsd:sequence>
           <xsd:element name="map" type="mapType" />
        </xsd:sequence>
       </xsd: complexCntent>
  </xsd:complexType>
</xsd:schema>
```

APPENDIX B

An Instance of the OCO Content Packaging:

```
<manifest identifier="Manifest" version="1.1"
   xmlns="http://www.imsproject.org/xsd/imscp_rootv1p1p2"
   xmlns:ococp=http://www.OCORM.org/xsd/ococp_v1p0
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.imsproject.org/xsd/imscp_rootv1p1p2.xsd
                       http://www.OCORM.org/xsd/ococp_v1p0.xsd">
   <metadata/>
   <organizations default="TOC1">
     <organization identifier="TOC1" structure="hierarchical">
      <item identifier="ITEM1" identifierref="OCO1" isvisible="true">
        <ococp:messagemapping>
          <ococp:mapping message=”message1” responder=” RES3” responsemethod=”method1”/>
        <ococp:messagemapping/>
      </item>
      <item identifier="ITEM2" identifierref="OCO2" isvisible="true">
        <ococp:messagemapping>
          <ococp:mapping message=”message2” responder=” RES3” responsemethod=”method2”/>
        <ococp:messagemapping/>
      </item>
    </organization>
```

```
  </organizations>
  <resources>
    <resource identifier="OCO1" ococp:ocormtype="oco" type="webcontent" href="lesson1.htm">
      < ococp:message identifier=”MESSAGE1” messagenameref=” requirement1”/>
    </resource>
    <resource identifier="OCO2" ococp:ocormtype="oco" type="webcontent" href="lesson1.htm">
      < ococp:message identifier=”MESSAGE2” messagenameref=” requirement1”/>
    </resource>
    <resource identifier="OCO3" ococp:ocormtype="oco" type="webcontent" href="help.htm">
      < ococp:method identifier=”method1” methodnameref=”help1”/>
      < ococp:method identifier=”method2” methodnameref=”help2”/>
    </resource>
  </resources>
</manifest>
```

In this example, it is supposed that the OCO3 can offer corresponding helps to the OCO1 and the OCO2 by using different methods. The OCO3 neither directly participates in any aggregation of content structure, nor is present in any sequence rule of SCORM. However, in the instructional experience, the OCO3 can be triggered to run by the corresponding instructional events of the OCO1 and the OCO2.. Thereby, a flexible content aggregation structure distinguished from IMS and SCORM specifications is formed.

APPENDIX C

An Instance of Aggregating Learning Contents through the 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 the three learning content components are required to be aggregated to compose a bigger learning unit. In this instance, we can design a container OCO4 that contains these three components to implement the aggregation, which is shown in Figure 4.

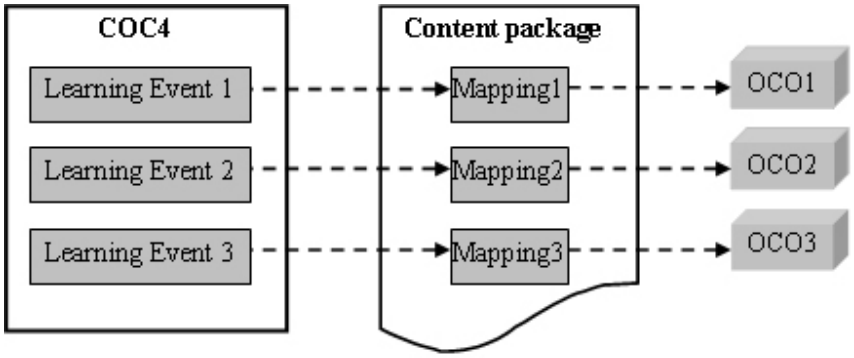

Figure 4. The structure of OCO Container

The detailed design steps are as follows:

(1) Three learning events, which can be triggered by some super-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 handling learning events, the OCO4 container indirectly linked the three contained components by thrice calling the API function SetValue (message_name) to send corresponding messages to the LMS.

(3) In the manifest file of the content packages, only the OCO4 is an Item of in the aggregation structure, and the OCO1, the OCO2 and the OCO3 are only three responders of the OCO4’s three messages, but not the items of the aggregation structure. Therefore, the structure of content aggregation will be simplified.

APPENDIX D

An Instance of Organizing Branched Instruction through the Associational Relationship between OCOs

The branched instruction shown in Figure 5 can be organized through the associational relationship shown between OCOs shown in Figure 6.

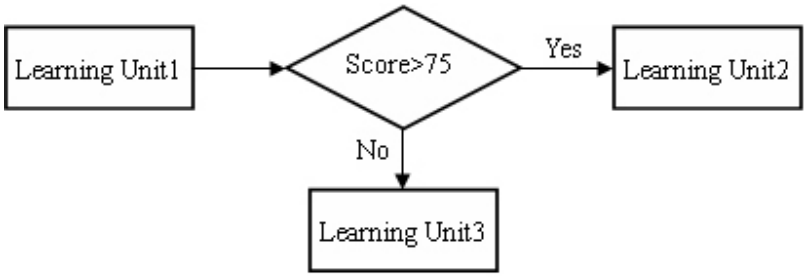

Figure 5. A flow of branched instruction

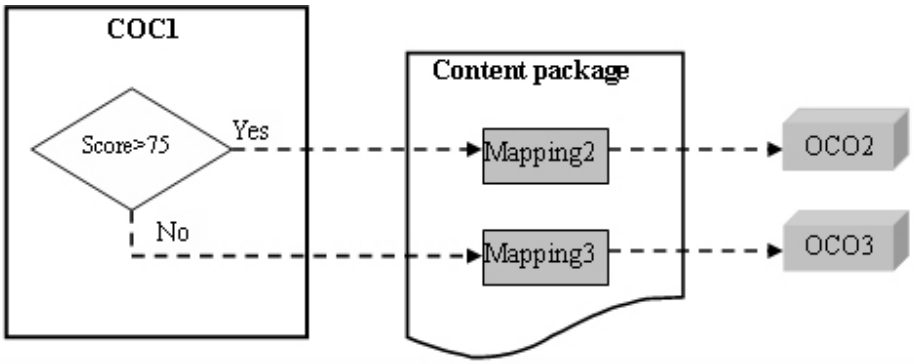

Figure 6. The OCOs' associational relationship applied to organize branched instruction

The detailed design steps are as follows:

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

(2) Within the learning object OCO1 , the following sentences to control branched flow will be added at the end of its program:

```
if (score>75)
   api.SetValue ("cmi.message_name","message1")
 //Send message1 to the LMS

else
   api.SetValue ("cmi.message_name","message2")
 //Send message2 to the LMS
```

(3) In the content package, the responder of message1 will be mapped into the OCO2, and the responder of message2 will be mapped into the OCO3 by an extended element <messagemapping> that is required for the content packaging:

```
<messagemapping>
   <mapping message="message1" responder=" OCO2" />
   <mapping message="message2" responder=" OCO3" />
</messagemapping>
```