A SCORM conformant Sequencing Engine based on Principles of Service Oriented Architecture

Mariana Karmazi¹, Christos Nikolaou¹

¹Transformation Services Laboratory, University of Crete, Greece

Key words: e-learning, learning management systems, SCORM, Web services

Abstract:
The Sharable Content Object Reference Model (SCORM) standard is the most widely adopted e-learning standard in learning management systems. However existing e-learning platforms fail to conform to the SCORM Sequencing and Navigation (SN) specification. This paper presents a SCORM conformant sequencing engine based on the Web services paradigm and principles of the Service-Oriented Architecture (SOA). The Sharable Content Object's (SCO), introduced in SCORM, functionality is wrapped by Web Services using the Web Service Resource Framework. Additionally, the sequencing behaviours are represented by Web Services and composed based on the BPEL standard. The usage of Web services provide us with a plug-and-play sequencing engine that can be easily integrated in existing e-Learning platforms.

1 Introduction
The evolution of Internet has changed society in unparallel ways and its impacts are prevailed in education. Any Internet user has access to a vast collection of information and learning content at any time and any place. Nowadays, academic institutions and industrial training organizations make use of content delivery platforms (e-Learning platforms) to facilitate Web-based education and training by providing local repositories to organize and store learning content as well as a set of synchronous and asynchronous tools to facilitate creation, delivery and presentation of them. However, most of the existing e-Learning platforms have been isolated because their learning content and learning functions are platform dependent and cannot be used outside the system thus prohibiting the communication and collaboration between actors from different systems. Consequently, content developers are obligated to design learning contents according to guidelines imposed by the learning platform they are using or the restrictions imposed from local repository or the service of the learning platform.

Many standards bodies and consortiums have worked on the formation of e-Learning standards with the aim to improve interoperability between learning systems as well as accessibility and reusability of learning content. Most of the work done is focused on the interoperability of the platforms by standardizing the communication models between them and on defining a standard way of describing learning content and ways to aggregate them in packages to conform to a standard form, facilitating their exchange between platforms. The set of the most distinguished e-Learning standards includes the Learning Object Metadata (IEEE LOM, [12]), which provides a schema for metadata annotation, several IMS [8] specifications, concerning the packaging of content in interoperable means (IMS Content Packaging, [9]) and the sequence of learning content in a package (IMS Simple Sequencing, [10]), and finally the Sharable Content Object Reference Model (SCORM, [15]) introduced by the ADL [1], which includes and extends standards and specifications from other standard bodies (e.g., IEEE LOM to tag learning content, IMS Content Packaging to package content and strategies, AICC API [2] specification for the communication of learning content and the
learning environment, IMS Simple Sequencing to create learning paths in the experience of content). The SCORM set of specifications is considered the most important standard in the e-Learning world.

Existing e-Learning platforms already support the content packaging conforming to either IMS Content Packaging or the SCORM extension of it and they have also integrated the specifications and guidelines for the communication of learning content with the environment. But, little work has been done towards the adoption of a sequencing engine as described in the SCORM Sequencing and Navigation specification (SCORM SN), which is based on the IMS Simple Sequencing specification. The SCORM SN defines a method for representing the intended behaviour of the sequencing and navigation in learning material based on data models and provides a description of the basic functionality that a sequencing engine of an e-Learning platform should exhibit.

This paper focuses on designing and developing a SCORM conformant sequencing engine, according to the SCORM SN and based on the Web services paradigm and principles of the Service Oriented Architecture (SOA). We propose a representation of the Sharable Content Object (SCO), a concept introduced by SCORM, as a WS-Resource according to the WS-Resource Framework [23]. Moreover, the sequencing behaviours of the Overall Sequencing Process, which are described in the IMS SS specification, are exposed as well-defined Web services and orchestrated, according to the Business Process Execution Language (BPEL) [14], in a business process. The resulting business process is the sequencing engine we propose. The usage of Web services and BPEL process provide us with a plug-and-play sequencing engine that can be easily integrated in existing e-Learning platforms as all it requires is a simple call to the BPEL process. Furthermore, the Web Services wrappers of SCOs will give us the opportunity to reference and deliver learning materials residing in remote repositories eliminating the need to package and import them to local repositories. Moreover, the use of Web Services to implement functionalities of e-Learning platforms will lead to more flexible systems as new services will easily replace the existing ones. The use of XML standards either for the sequencing behaviours (WSDL, SOAP) or for the learning objects wrappers will enable e-Learning systems to withstand technology evolution and changes without costly redesign or reconfiguration. Finally, the use of the WS-Resource Framework to represent wrappers of the learning objects give us the possibility to combine all the various models introduced by the standards (sequencing definition model, tracking model, communication model) into one representation, this of the WS-Resource [22]. WS-Resource is XML-based so any changes in any of the above three models could be painlessly integrated to the proposed work.

The rest of the paper is organized as follows: Section 2 describes the fundamental aspects of SCORM, SOA and Web services. The proposed architecture including the representation of learning objects as Web services with the use of WS-Resource Framework and the architecture of the proposed sequencing engine are described in Section 3. Section 4 describes related work on applying SOC in learning management systems and Section 5 concludes this paper.

2 Fundamentals

This section describes the basic aspects of the SCORM 2004 standard and Web services.

2.1 SCORM 2004 3rd Edition Standard

The Sharable Content Object Reference Model (SCORM) introduced by the Advanced Distributed Learning Initiative (ADL) is a technical framework that provides a harmonized set of guidelines, specifications and standards built on the proven work of prominent organizations (AICC, IEEE LTSC [13], IMS, ARIADNE [3]). The SCORM 2004 3rd Edition
appeared in October 2006 and it consists of three technical books; the Content Aggregation Model, the Run-Time Environment and the Sequencing and Navigation.

The Content Aggregation Model (SCORM CAM) is based on the IMS Content Packaging specification and describes packaging of content, labelling, discovery and exchange of content packages. A Content Package, which is a zip package, groups one or more learning resources in order to ease the transportation of content from system to system and may represent a course, a lesson, a module or just a collection of related content objects. An XML file, named imsmanifest.xml, residing in the content package, describes the resources that are contained in the package, the various hierarchical organizations of them and it contains the IEEE LOM metadata (and extensions) for the resources and the IMS SS sequencing metadata that are used for defining the sequencing rules to be applied to each of the resources and the content package as a whole.

The Run-Time Environment (SCORM RTE) describes a run-time API and a data model (Communication Data Model) to be used for communication between content objects and learning management systems (LMS). Its purpose is to guarantee and standardize the communication between a specialized form of learning object, the Sharable Content Object (SCO), and LMS from different sources. It also describes the SCO, the standard communication mechanism and the data elements used in this communication.

The Sequencing and Navigation (SCORM SN) is based on the IMS SS specification and defines a method for representing the intended behaviour of an authored learning experience such that any SCORM compliant LMS will sequence discrete learning content in a consistent way [15]. It defines the behaviours and functionality that an LMS should exhibit and implement to process sequencing information (Overall Sequencing Process). It utilizes the term ‘Activity Tree’ to describe the branching and flow of learning resources (paths of learning experience) based on an authored, prescribed, sequencing strategy and the learner’s interactions with content objects at run-time. The intended sequencing strategy is defined by the content package’s author at design time based on the Sequencing Definition Model. The Sequencing Definition Model is a data model used to define sequencing rules in the imsmanifest.xml in order to be applied to the nodes of the Activity Tree. The Activity Tree is derived from the content package as described in SCORM CAM and its nodes keep information regarding the Tracking Model, while leave nodes are the ones that correspond to the actual resources to be delivered to the learner. The Tracking Model is one of the data models introduced by SCORM and is used to record learner’s interactions with activities of the Activity Tree at run time in order to further control the selection and sequencing of content. Finally, SCORM SN utilizes the IMS SS’s Activity State Model which keeps state of each activity in the Activity Tree and the global state of the Activity Tree during a sequencing session. This model as well as the Tracking Model and the Communication Model are dynamic run-time data models. Finally, it deals with navigation tools.

2.2 Service-Oriented Architecture and Web services

Service-oriented Architecture (SOA) presents an approach for building distributed systems that deliver application functionality as services to either end-user applications or other services. It provides a promising way to address problems related to the integration of heterogeneous applications in a distributed environment. The most common technology for implementing Service-Oriented Architecture is the Web services technology.

The W3C’s Web Services Architecture Working Group jointly agreed on the following working definition of a Web service. “A Web service is a software application identified by a URI, whose interfaces and bindings are capable of been defined, described, and discovered as XML artefacts. A Web service supports direct interaction with other software agents using XML-based messages exchanged via internet-based protocols” [26]. The heart of the Web
services is the standards [18]. Three key XML-based standards have been defined to support web service deployment, the Simple Object Access Protocol (SOAP) [24] as a messaging envelope format, the Web Services Description Language (WSDL) [27] as service description format and the Universal Description Discovery and Integration (UDDI) [25] for defining metadata that is used for service discovery. Moreover, Web services can be combined to construct business processes. A business process can be defined as a set of interrelated tasks linked to an activity that spans functional businesses [19] and it specifies the potential execution order of operations from a collection of Web services, the data shared between these Web services, which partners are involved and how they are involved in the business process, joint exception handling for collections of web services and other issues. The language that is used to define and execute complex business processes with Web services is the Business Process Execution Language for Web services (WSBPEL or BPEL) [14]. BPEL is an XML-based workflow definition language developed by OASIS to specify business processes as a set of interactions between Web services. BPEL provides the artefacts (grammar) to describe business processes based on XML including its control and message format. It depends on the WSDL to describe the external services that are needed by the process.

Another interesting set of specifications is the Web Services Resource Framework (WSRF) published also by OASIS. This specification is vital for this work because important design issues are based on it. This specification consists of a set of proposed Web services specifications for modelling, accessing and interacting with stateful resources using Web services. It defines all the conventions for managing state in the Web services context, enabling the various applications to discover, inspect and interact with stateful resources in standard and interoperable ways. From a client point of view, these conventions regard the message exchanges used to interact with state. According to the WSRF, the state is kept in a separate logical entity called resource and particularly, stateful resource. Each stateful resource has a set of zero or more properties, a well-defined lifecycle and it is assigned with a unique key that is used by one or more Web services whenever a stateful interaction is needed. A stateful resource can also be comprised of other stateful resources. WSRF does not include resource’s identifier in the WSDL description but instead it uses the WS-Resource endpoint reference that implicitly includes in all messages addressed through it. This endpoint reference comprises of the resource’s unique identifier and the URL of the Web service that manage this resource and adheres to the Web Service Addressing specification (WS-Addressing)[21].

3 Proposed Architecture

The proposed architecture has been designed with the objective to provide an engine that encapsulates the Overall Sequencing Process according to the Sequencing and Navigation (SN) specification in the SCORM 2004 3rd Edition and incorporating this functionality into existing LMSs. Web services technologies have been utilized to provide the functionality of the engine in terms of Web services. The processes and sub-processes, described in the specification and used in the sequencing engine, have been grouped into sets. Web services have been designed and implemented for each of these sets. Finally, these Web services have been orchestrated according to the BPEL standard to provide the sequencing engine as a process. A detailed description of this work can be found in [11].

The main architecture comprises of two important parts; the first part deals with representation and the instantiation of Web services as wrappers of learning objects and the second part deals with the definition of the BPEL process that exhibits the behaviour of the Overall Sequencing Process as described in the SCORM SN. A generalized architecture of a recommended e-Learning system, including the BPEL engine as external component, a LMS, remote servers and repositories, and the client is illustrated in Figure 1.
The BPEL Sequencing engine acts as a remote standalone component of a LMS and can interact with the local or remote repositories and remote file systems to query for SCORM conformant Content Packages. One of engine’s responsibilities is to locate the requested content package and create the Activity Tree needed for the sequencing purposes. The Activity Tree is created at run-time and is consisted of WS-Resources wrappers of learning content existing in the package. A SCORM package contains the learning resources (physical files) intended to deliver to learners alongside with a manifest file, imsmanifest.xml, and IMS and SCORM schema files. The manifest file describes the hierarchical organization of resources and metadata. These metadata describe the resources and are based on the IEEE LOM standard. Also, the Sequencing Definition Model (SDM) is attached to the items in this manifest file to define an instructional strategy for sequencing purposes.

The BPEL Sequencing engine waits for navigation requests that result in starting a new sequencing session (create Activity Tree and possibly deliver the first learning content) or identifying the “next” learning content to deliver or nothing (exception occurs or sequencing strategy is violated, so there is nothing to deliver) or ending successfully the sequencing session of the learner. For each learner and a SCORM package request, a new BPEL process instance is created and initialized. If the process identifies an activity, which its content is to be delivered to the client, the endpoint reference of this activity is returned. This endpoint reference could be used by the LMS’s Run-Time Service to query the WS-Resource and for storing and retrieving communication data between client and LMS. These communication data is based on the Communication Data Model introduced in SCORM Run-Time Environment.

The Web Services wrappers of SCOs give us the opportunity to reference and deliver learning materials residing in remote repositories eliminating the need to package them and importing to local repositories. Moreover, the use of Web Services to implement functionalities of e-learning platforms lead to more flexible systems as new services will easily replace the existing ones.

### 3.1 Learning Objects as Web Services

We propose a representation of the Sharable Content Object (SCO) as a WS-Resource according to the WS-Resource Framework (WSRF) in order to define it’s characteristics and functionality with Web service. The WSRF provides the means to keep state between Web service’s interactions and this state is represented in an XML document. Each learning resource, which exists in a SCORM-conformant content package, is represented with the WS-Resource Framework constructs at run time. The components of the system that deal with creating the WS-Resource representation and manipulation of learning content are depicted in Figure 2.
The WS-Resource Factory Pattern is used to create instances of WS-Resources through a factory service. Whenever a client wants to create an instance of WS-Resource, it will invoke the factory service, which responsibility is to create and initialize a new resource. The new resource will be assigned with a unique key and the factory service will return an endpoint reference to the newly created stateful resource. This endpoint reference could be further used by the client to invoke the instance’s service operations to the resource.

The **LOManipulationFactoryService** is the factory Web service of the WSRF implementation and its responsibility is first to create a resource based on the URL of a package, a learner’s identifier and name and second to assign a unique identifier to this resource and to return the endpoint reference of the created WS-Resource to the client. Learner name and identifier are needed in the BPEL correlation sets for identifying the process instance of the sequencing engine assigned to the learner.

The **LOManipulationResourceHome** is a Java class and it is in charge for managing the resources. The resource home is managing multiple resources and it must keep track of several resources at the same time. The factory service uses the resource home to create new resources and the instance service will use this service to find a resource with a given key.

The **LOManipulationService** is the stateless front end. The client interacts only with this Web service to query the stateful resource. This Web service and the **LOResource** constitute our WS-Resource implementation of learning objects contained in a SCORM conformant zip package. The operations provided by this service includes the functions as described in SCORM Run-Time Environment API, the WSRF generic operations, such as get- and set-methods, and a group of operations that can be used to modify certain property elements in the resource property document.

Finally, the **LOResource** is a Java class that implements the stateful resource of this work approach. This class implements the interfaces Resource, ResourceProperties and ResourceIdentifier defined in the WS-Core 4.2.0 library of the Globus toolkit implementation of the WS-Resource Framework. Each private member of this class representing the resource properties are instances of the SimpleResourceProperty defined in the WS-Core 4.2.0 library, as well. The “set” methods are implemented in order the resource to preserve the most recent changes. “Get” methods are implemented only for the methods that correspond to the API implementation of the SCORM RTE.

As discussed in the previous section, the resource property document is a set of resource properties elements, XML elements, that keeps the state and it is described as an XML document defined with an XML Schema. Figure 3 contains the definition of the **LOResource** resource property document in our case. The elements of this property set correspond to parts of the imsmanifest.xml file of the content package as described in the SCORM standard; the resource (content) to be delivered to learner (if any) which is associated with the WS-Resource instance; the sequencing metadata needed for interpreting the sequencing rules by the sequencing engine; any SCORM navigation extensions needed to control the navigation events and future SCORM extensions in order to facilitate any future changes in the standard; elements like package’s URL, learner’s identifier and name that are needed by the sequencing
engine. The resource property document also contains a set of elements corresponding to the following data models; the Tracking Model (objective progress, activity progress, attempt progress), the Activity State Model and the Communication Data Model. For these three data models described in SCORM Sequencing and Navigation Model, in the SCORM RTE and in the IMS Simple Sequencing specification, XML schema types and elements have been defined in order to represent resource property elements.

```xml
<element name="LO-ResourcePropertiesSet">
  <complexType>
    <sequence>
      <element ref="myTypes:mylearningObject" minOccurs="0" maxOccurs="1"/>
      <element ref="myTypes:node" minOccurs="1" maxOccurs="1"/>
      <element name="manifestBase" type="string" minOccurs="0" maxOccurs="1"/>
      <element ref="imsss:sequencing" minOccurs="0" maxOccurs="1"/>
      <element ref="myTypes:scormNavExt" minOccurs="0" maxOccurs="1"/>
      <element ref="myTypes:anyExtension" minOccurs="0" maxOccurs="1"/>
      <element ref="myTypes:ObjectiveProgressSet" minOccurs="0" maxOccurs="unbounded"/>
      <element ref="myTypes:ActivityProgressInformation" minOccurs="0" maxOccurs="1"/>
      <element ref="myTypes:AttemptProgressInformation" minOccurs="0" maxOccurs="unbounded"/>
      <element ref="myTypes:ActivityStateInformation" minOccurs="0" maxOccurs="1"/>
      <element ref="myTypes:CMIGlobalState" minOccurs="0" maxOccurs="1"/>
    </sequence>
  </complexType>
</element>
```

Figure 3. LO-Resource Properties Document Definition

For the implementation, the WS-Core library is used from the Globus Toolkit. WS-Core is a java implementation of the WS-Resource Framework, the Web Service Notification (WSN) specifications, WS-Security technology and ServiceGroup implementation. The 4.2.0 encapsulates the final specifications of the WS-Resource Framework.

### 3.2 Sequencing Behaviours in BPEL

As mentioned in Section 2, SCORM SN describes the basic behaviours that a LMS should implement in order to support sequencing of learning material residing in a content package, according to the sequencing rules and learner’s progress. These behaviours constitute the Overall Sequencing Process of a sequencing engine according to the IMS SS. The Overall Sequencing Process provides the overarching control process for the LMS’s sequencing implementation and describes how the various sequencing behaviours are applied within the context of a sequencing session. The entry point of the Overall Sequencing Process is a navigation request issued by the LMS and the exit point of this process is either the identification of the “next” activity to deliver, which could be nothing, or an exception. The Overall Sequencing Process exhibits the behaviour of a sequencing loop when a sequencing session starts until it ends.

In this work, this process is defined as part of a business process using the BPEL. The Web services that are composed into this process represent the sequencing behaviours as described in SN. This BPEL process is published as well as a Web service and can be used by any e-learning platform that lacks support of the SCORM SN. Each implemented Web service represents a set of processes and sub-processes as described in the IMS SS and SCORM SN. An abstract representation of the BPEL process and the services that orchestrates are depicted in Figure 4. Figure 4 also depicts the sequence of the Web services and how they fit with concepts and implementation details described in Section 2 and 3.1. The services highly depend on the endpoint references of three specific elements; the root of the Activity Tree, the Current Activity, where most sequencing decisions are made from, and the Suspended Activity (an activity that indicates that the previous sequencing session has been suspended and the current session should begin from this activity). The set of Web services invoked by the BPEL process are described below.

The NavigationService implements the Navigation Behaviour which is the entry point in the Overall Sequencing Process loop. This behaviour tests for the validity of the issued navigation request and it is responsible for calling either the Termination Behaviour or Sequencing Behaviour or both. This means that for a navigation request a corresponding
termination request or sequencing request or both is issued. Navigation requests corresponds to navigation events which are external events issued by UI controls and indicate system’s or user’s intention to navigate through content in some manner. These navigation requests are vocabulary specific.

The **TerminationService** implements the **Termination Behaviour** as described in the IMS SS specification. This behaviour is responsible to end the attempt on the current activity prior to processing any sequencing request and to ensure the state of the Activity Tree is in the most current valid state. This service is invoked whenever a termination request is issued by the **NavigationService**.

The **SequencingService** implements the **Sequencing Behaviour** as described in the IMS SS specification. This behaviour is responsible to determine the next activity by traversing the Activity Tree based on the current activity or to initiate a new learning sequencing session by identifying the first activity to deliver. This behaviour does not alter the state of the Activity Tree. The sequencing request comes from either **NavigationService** or **TerminationService**.

The **DeliveryService** implements the **Delivery Behaviour** as defined in the IMS SS specification. This behaviour is responsible to validate the identified activity from the **SequencingService** and to deliver or not the content.

Finally, the **ContentDeliveryService** implements the **Content Delivery Environment Process** as defined in the IMS SS specification. This service plays the role of a bridge between the SCORM delivery mechanism and the LMS sequencing engine. It manages the state of the Activity Tree pending an assumed delivery of a content object and identifies the learning resource to the SCORM delivery mechanism. The SCORM delivery mechanism defines a common way for LMSs to start an attempt on Web-based content objects. This mechanism defines the procedures and responsibilities for the establishment of communication between the delivered content object and the LMS. The communication protocols are standardized using a common API. This common delivery scheme enables consistent content object delivery behaviour across LMSs without specifying the underlying LMS implementation.

There are two services that are not directly invoked by the BPEL process, the **RollupService** and the **UtilitiesService**. The **RollupService’s** operations are invoked from the **TerminationService**. This service is an implementation of the **Rollup Behaviour** described in the IMS SS specification. The IMS SS Rollup Behaviour defines the process to determine a cluster’s status information based on its children’s status information and applies to all parts of the Tracking Model. The **UtilitiesService’s** operations are called from all other services. The operations implemented by this service correspond to the following processes and sub-

---

**Figure 4. Abstract View of BPEL process and Architecture Component**

The **TerminationService** implements the **Termination Behaviour** as described in the IMS SS specification. This behaviour is responsible to end the attempt on the current activity prior to processing any sequencing request and to ensure the state of the Activity Tree is in the most current valid state. This service is invoked whenever a termination request is issued by the **NavigationService**.

The **SequencingService** implements the **Sequencing Behaviour** as described in the IMS SS specification. This behaviour is responsible to determine the next activity by traversing the Activity Tree based on the current activity or to initiate a new learning sequencing session by identifying the first activity to deliver. This behaviour does not alter the state of the Activity Tree. The sequencing request comes from either **NavigationService** or **TerminationService**.

The **DeliveryService** implements the **Delivery Behaviour** as defined in the IMS SS specification. This behaviour is responsible to validate the identified activity from the **SequencingService** and to deliver or not the content.

Finally, the **ContentDeliveryService** implements the **Content Delivery Environment Process** as defined in the IMS SS specification. This service plays the role of a bridge between the SCORM delivery mechanism and the LMS sequencing engine. It manages the state of the Activity Tree pending an assumed delivery of a content object and identifies the learning resource to the SCORM delivery mechanism. The SCORM delivery mechanism defines a common way for LMSs to start an attempt on Web-based content objects. This mechanism defines the procedures and responsibilities for the establishment of communication between the delivered content object and the LMS. The communication protocols are standardized using a common API. This common delivery scheme enables consistent content object delivery behaviour across LMSs without specifying the underlying LMS implementation.

There are two services that are not directly invoked by the BPEL process, the **RollupService** and the **UtilitiesService**. The **RollupService**’s operations are invoked from the **TerminationService**. This service is an implementation of the **Rollup Behaviour** described in the IMS SS specification. The IMS SS Rollup Behaviour defines the process to determine a cluster’s status information based on its children’s status information and applies to all parts of the Tracking Model. The **UtilitiesService**’s operations are called from all other services. The operations implemented by this service correspond to the following processes and sub-
processes defined in the IMS Simple Sequencing specification: Limit Conditions Check Process (UP.1), Sequencing Rules Check Process (UP.2), Sequencing Rule Check Subprocess (UP.2.1), Terminate Descendent Attempts Process (UP.3), End Attempt Process (UP.4) and Check Activity Process (UP.5).

The BPEL process invokes the ContentManagementModule service prior to the Overall Sequencing Process loop. Its responsibility is to create the Activity Tree of the WS-Resources and therefore provides operations for creating and initializing the Activity Tree as well as the resources that corresponds to the activities, creating the TOC based on imsmanifest.xml, writing suspended state of the Activity Tree and initializing an Activity Tree according to previous suspended state (if any).

The BPEL engine used for executing the implemented BPEL process is the Active BPEL engine from Active Endpoints. This engine has been chosen because it is open source, runs on the Jakarta server and support Axis 1.4. Axis 1.4 has been chosen for the implementation of every Web service mainly because the Globus toolkit and specifically the Java WS-Core 4.2.0 library of the WS-Resource Framework depends on the Axis 1.4 for the message invocation. For testing purposes, the proposed sequencing engine has been integrated into the ADL SCORM Run-Time Environment, which is an open source Web application based on Java servlets and JavaServer Pages Technology (JSP). The ADL SCORM RTE serves as the layer that provides the sequencing engine the necessary parameters for execution. These parameters are the learner id and the navigation event. The navigation event is triggered by the environment’s specific UI controls and informs the sequencing engine about the learner’s intention to continue with content, quit or suspend the application. Furthermore, AJAX technology has been used for communicating with the server-side of the ADL’s RTE because the WS-Resource qualified endpoint reference is used to invoke the API methods of the WS-Resource.

4 Related Work

Theoretical studies about the application of web services in e-learning systems exist in the literature. Some researchers propose a SOA-architecture of generic e-learning platforms. They subdivide the main-functionality of an e-learning system into a number of standalone applications which have been implemented as Web services [20]. In [28] a functional architecture is presented to achieve interoperable learning systems that conform to standards such as IEEE LOM and SCORM. They propose a division of a learning system into learning management and content management in order to clarify the functional responsibilities of each component constituting a LMS. A SOA-based model is introduced in [20] for a Web-based framework based on the LTSA architecture that learning objects can be searched, discovered and assembled based on the metadata annotations of the LOM. An interesting study is presented in [5] where a framework is proposed for the adoption of the whole SCORM model in a SOA-based architecture. They have provided insights on how the LMS’ services described in the SCORM RTE can be externalize and provided as web services. The authors argue the fact that while SCORM promotes content interoperability, the web service paradigm can promote LMS interoperability. A similar work is described in [6] where a SOA architecture is presented for implementing the SCORM RTE service as a service, while existing LMS can adopt this service in order to support a SCORM RTE. A more technical work is presented in [16] where the SOAP protocol is used as a communication protocol between a SCORM RTE and an LMS established on a .NET platform.

Other studies are concerned with the searching of learning objects and the adaptability of them in the learners’ need. In [17], a Web-based e-learning architecture is proposed constituting of various distributed servers that communicate in order to exchange learning-related information such as learner data and learning strategies. Furthermore, it enables the automatically modifications of the learning objects sequence to meet the learner’s learning
styles and strategies. Further studies propose a mapping of e-learning standards, such as SCORM and LOM, to RDF and DAML-S to deliver e-Learning systems within the context of the Semantic Web [4]. In [7] an approach towards the personalization of learning content based on distributed environments based on semantic web services is presented.

5 Conclusion and Future Work

In this paper, we presented an approach towards applying Service Oriented Architecture concepts in the design and implementation of e-learning systems based on Web services. The idea behind this work is to connect the world of e-learning systems and the world of Web services. The main purpose is to propose a loosely coupled architecture of e-learning systems by providing external services for their missing functionalities. However, there are some missing features in the implementation and particular those features corresponding to standards from the Web services stack. The sequencing engine and the learning objects should secure their communication, their message exchanges, with the WS-Security. The WS-Policy could be further utilized for passing policies between clients.

References:


**Author(s):**

Mariana Karmazi, PhD student  
University of Crete, Computer Science Department  
Prefabricated Buildings of Knossos, Heraklion, 71409, Greece  
karmazi@tsl.gr

Christos Nikolaou, Professor  
University of Crete, Computer Science Department  
Prefabricated Buildings of Knossos, Heraklion, 71409, Greece  
nikolau@tsl.gr