University of Crete

Department of Computer Science

A SCORM conformant Sequencing Engine based on WS-Resource Framework and Principles of Service Oriented Architecture

By

Mariana Karmazi

Master's Thesis

Heraklion, April 2009

University of Crete

Computer Science Department

A SCORM conformant Sequencing Engine based on WS-Resource Framework and Principles of Service Oriented Architecture

By

MARIANA KARMAZI

A thesis submitted in partial fulfillment of the

requirements for the degree of

Master of Science

Author: _____

Mariana Karmazi, Department of Computer Science

Board of enquiry:

Supervisor: _____

Christos Nikolaou, Professor

Member: _____

Dimitrios Plexousakis, Professor

Member: _____

Georgios Gewrgakopoulos, Associate Professor

Approved by: _____

Panos Trahanias, Professor,
Chairman of the Graduate Studies Committee

Heraklion, April 2009

A SCORM conformant Sequencing Engine based on WS-Resource Framework and Principles of Service Oriented Architecture

MARIANA KARMAZI

Master's Thesis

University of Crete

Computer Science Department

Abstract

Many standards have been defined aiming at supporting interoperability of systems and learning objects as well as accessibility, adaptability and reusability of them. The SCORM set of specifications is the most widely adopted. However, existing e-Learning platforms fail to conform to one specification included in SCORM. This is the SCORM Sequencing and Navigation Specification, which is based on the IMS Simple Sequencing specification, and provides a method for representing the intended behavior of the sequencing and navigation in learning material base on data models and provides a description of basic behaviors that an e-Learning platform should exhibit. A SCORM conformant sequencing engine will be implemented based on the Web Services paradigm and principles of the Service-Oriented Architecture (SOA). The basic concept of SCORM, the Sharable Content Object (SCO) and its functionality will be wrapped by Web Services using the Web Service Resource Framework. Additionally, the sequencing behaviors will be also represented by Web Services and their composition will finally result in the implementation of the sequencing engine. The composition of Web Services that represent the sequencing behaviors will be implemented according to the Business Process Execution Language (BPEL), which is a SOA standard for orchestrating services. The usage of Web services and BPEL process will provide us with a plug-and-play sequencing engine that can be easily integrated in existing e-Learning platforms. The only requirement is a simple invocation 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 them and importing to local repositories. Moreover, the use of Web Services for implementing functionalities of e-Learning platforms will lead to more flexible systems as new services will easily replace the existing ones.

Μηχανή Ακολουθίας Εκπαιδευτικών Αντικειμένων βασισμένη στην προδιαγραφή WS-Resource και σε αρχές που διέπουν τη Service-Oriented Αρχιτεκτονική

ΜΑΡΙΑΝΑ ΚΑΡΜΑΖΗ

Μεταπτυχιακή Εργασία

Πανεπιστήμιο Κρήτης

Τμήμα Επιστήμης Υπολογιστών

Περίληψη

Αρκετά πρότυπα και προδιαγραφές έχουν οριστεί με σκοπό να προωθήσουν την επιτυχή επικοινωνία διαφόρων εκπαιδευτικών συστημάτων και την προσβασιμότητα στα εκπαιδευτικά αντικείμενα, καθώς επίσης στην προσαρμοστική ικανότητα και την επαναχρησιμοποίηση αυτών. Υπάρχουσες εκπαιδευτικές πλατφόρμες προσπαθούν να υιοθετήσουν τις υπάρχουσες οδηγίες. Ελάχιστες προσπάθειες έχουν γίνει σχετικά με την υλοποίηση και την υποστήριξη του SCORM Sequencing and Navigation specification (SCORM SN), το οποίο βασίζεται στο IMS Simple Sequencing specification, το οποίο ορίζει αφαιρετικά μοντέλα δεδομένων για την αναπαράσταση συγκεκριμένης στρατηγικής και ακολουθίας παρουσίασης του εκπαιδευτικού περιεχομένου και περιγράφει τις λειτουργίες που πρέπει να υποστηρίζει μια μηχανη. Αυτή η μεταπτυχιακή εργασία προτείνει την υλοποίηση μιας μηχανής εκτέλεσης των οδηγιών και υποστήριξης των μοντέλων δεδομένων που ορίζει το SCORM SN specification, βασισμένη σε αρχές του Service Oriented Architecture. Τα εκπαιδευτικά αντικείμενα, μια βασική έννοια που διέπει όλα τα εκπαιδευτικά συστήματα, αναπαριστόνται με τη χρήση των υπηρεσιών ιστού (Web services) και συγκεκριμένα με το πρότυπο Web Services Resource Framework. Οι λειτουργίες της μηχανής υλοποιούνται με Web services, η σύνθεση των οποίων παρέχει τη λειτουργικότητα μιας sequencing μηχανής βάση της προδιαγραφής του SCORM SN. Η σύνθεση των υπηρεσιών υλοποιείται με την χρήση του προτύπου WS-BPEL. Ένας από τους στόχους αυτής της εργασίας είναι να επιδείξει πώς έννοιες που παρουσιάζονται στο Service Oriented Architecture μπορούν να εφαρμοστούν σε περιοχές του e-Learning και να επιδείξει αυτή τη δυνατότητα με την υλοποίηση μιας μηχανής που μπορεί εύκολα να εφαρμοστεί στα υπάρχοντα συστήματα.

Table of Contents

List of Figures

List of Tables

List of Listings

Acknowledgements

I would like to thank my supervisor Christos Nikolaou for his assistance and support through this thesis work. I would also like to thank both Prof. D. Plexousakis and G.Georgakoupoulos for agreeing to be members of my board of enquiry and for reading my work.

Special thanks to my friend and colleague, Vangelis Papathanasiou, for his invaluable support in the last months of this work. I would like also to thanks my friends that stayed by my side in the difficult times as well as the members of the Transformation Services Laboratory.

Finally, I would like to thank my family and Constantinos Ampatzidis for the belief they have had in me and their encouragement during these years.

Definitions and Abbreviations Used

AICC	Aviation Industry Computer Based Training Commitee
ADL	Advanced Distributed Learning
CAM	Content Aggregation Model
CBT	Computer Based Training
CMI	Computer Managed Instruction
DOM	Document Object Model
EPR	Endpoint Reference (WS-Addressing specification)
IEEE	The Institute of Electrical and Electronic Engineers
IMS	IMS Global Consortium, Inc
LMS	Learning Management System
LOM	Learning Object Metadata
LTSC	Learning Technology Standard Committee
PIF	Package Interchange File
RTE	Run-time Environment
SCO	Sharable Content Object
SCORM	Sharable Content Object Reference Model
SDM	Sequencing Definition Model
SN	Sequencing and Navigation
SS	Simple Sequencing
TM	Tracking Model
WSRF	Web Service Resource Framework
WSRF-BF	WS-BaseFaults
WSRF-RL	WS-ResourceLifetime
WSRF-RP	WS-ResourceProperties
WSRF-SG	WS-ServiceGroup
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformation

Chapter 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 at 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 learning resources. 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. Content developers are obligated to design learning contents according to guidelines proposed by the learning platform they are using or the restrictions opposed 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 contents. Accessibility refers to the ability of finding relevant learning resources anytime the latter are needed. With this in mind, the concept of Learning Object has been introduced to refer to any content used for education and appropriate standards have been proposed to facilitate their use in e-learning environments. According to Learning Object, any learning resource should be annotated with a specific set of metadata to provide bibliographic information about it in order to ease Internet users finding the appropriate content to meet their needs. A standard-based metadata labeling is also mandatory in order to enhance reusability of content elements and to eliminate restrictions imposed by learning platforms. Most of the work done by the standard bodies is focused on the interoperation of the platforms by standardizing the communication models between them. Moreover, they have tried to define a standard way of describing learning content as well as to aggregate them in packages, which conform to a standard form, facilitating their exchange between platforms.

The most widely adopted standards are those of the Learning Object Metadata (IEEE LOM, [22]), providing a schema for metadata annotation, several IMS [14]specifications concerning the packaging of content in interoperable means (IMS Content Packaging, [15]) and the sequence of learning content in a package (IMS Simple Sequencing, [17]) and the Sharable Content Object Reference Model (SCORM, [36-X]) introduced by the ADL [1]which includes the most important works from other standard bodies.

The SCORM reference model provides a technical framework and guidelines for the creation of packages for importing and exporting content from learning management systems, the structure of these packages, the metadata for describing the content, the run time environment that learning systems must support in order to track learner experience with learning content and the usage of a sequencing engine that sequences learning items of a package in a consistent way. SCORM is a harmonized set of specifications and standards because it includes specifications and standards formed by many distinct e-Learning specifications and standards bodies such as the IEEE Learning

Object Metadata (LOM) standard to tag the learning content, IMS Content Packaging specification to package content and strategies, the AICC API, [2] specification for the communication of learning content and the learning environment and the IMS Simple Sequencing to create learning paths in the experience of content.

Currently, many e-learning systems exist (e.g. Moodle, Claroline, Blackboard and Atutor) that each one has its own features and tools. The integration of the aforementioned standards is slow. Most of them already support the content packaging conforming to other IMS Content Packaging or the SCORM extension of it and also they have integrate the specifications and guidelines for the communication of learning content with environment. But, little work has been done for the adoption of a sequencing engine as described in the SCORM Sequencing and Navigation specification (SCORM SN), which follows the IMS Simple Sequencing specification. The SCORM SN specification is based on the IMS Simple Sequencing specification which defines a method for representing the intended behavior of the sequencing and navigation in learning material based on data models and provides a description of basic behaviors that an e-Learning platform should exhibit. Furthermore, none of the widely used e-learning systems have been designed and implemented with Web Services architecture.

For these reasons, this thesis focuses on designing and developing a SCORM conformant sequencing engine based on the Web services paradigm and principles of the Service Oriented Architecture (SOA). The thesis proposes a representation of the Sharable Content Object (SCO), a concept introduced by SCORM, as a WS-Resource according to the WS-Resource Framework. Moreover, the sequencing behaviors are exposed as Web Services and orchestrated in a business process to exhibit the functionality of the Overall Sequencing Process of the IMS SS specification. The resulting business process is the sequencing engine we propose. The composition of Web Services that represent the sequencing behaviors are composed according to the Business Process Execution Language (BPEL), which is a SOA protocol for orchestrating services. 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 them and importing 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 be easily replace existing ones.

1.1 Thesis Objectives

The first objective of this thesis is to develop a WS-BPEL sequencing engine conforming to the SCORM Sequencing and Navigation and the IMS Simple Sequencing specifications. These specifications define the behaviors that a sequencing engine should exhibit and the necessary concepts utilized by such an engine. The most important concepts are the Activity Tree for organizing Learning Objects and the tracking information needed as well as the overall functionality.

The main objective of this work is to apply Service Oriented Architecture in e-learning systems. Web services are utilized for exposing the sequencing engine's functionalities and they serve as wrappers of the Learning Objects.

1.2 Thesis Contribution

This thesis presents a solution for integrating Service Oriented Computing in e-learning platforms. It provides a sequencing engine solution that exposes the functionality of the Overall Sequencing Process as described in the SCORM and IMS specifications. In this approach,

1. Learning objects are modeled as Web services using the WS-Resource Framework (WSRF). Each Learning object is considered as a WS-Resource.
2. The Overall Sequencing Process is modeled as a business process described in the BPEL language. The resulted business process provides the functionality of a SCORM-conformant sequencing engine.
3. Each behavior described in the specification is represented as a well-defined Web service. The resulting set of Web services is used in the business process to succeed implementing the engine.

The solution outlined in this thesis offers significant advantages; Service-Oriented Architecture (SOA) provides a way to address problems related to the integration of heterogeneous applications in a distributed environment. The sequencing engine is implemented based on existing standards such as WSDL, BPEL and WSRF.

Existing e-learning systems slowly adopt the current e-learning standards and the efforts are mainly focus on the communication of the running Learning Objects with the system to pass information regarding the learner status and on the content packaging specification as described in SCORM to transfer learning resources from system to system. Most of them have not yet implement the sequencing process as described in SCORM SN. The Web service based solution for implementing a sequencing engine can be easily integrated into existing learning platforms as well as into newly developed as the only requirement is a simple invocation to the business process. The plug-and-play engine will provide all the missing functionality and conformance to the SCORM standard and moreover will help those learning management systems that have not been designed and implemented according to standards.

The use of XML standards either for the sequencing behaviors (WDSL, SOAP) or for the Learning Objects Wrappers will enable e-Learning systems to withstand technology evolution and changes without costly redesign or reconfiguration. Furthermore, an easy integration of newly web services that provides new and better functionality is feasible. The use of the WS-Resource Framework as wrappers of Learning Objects has achieved in combining the various models (sequencing definition model, tracking model, communication model) presented by the standards into one representation, this of the WS-Resource. WS-Resource is XML based so any changes in any of the three models could be painlessly integrated to the proposed work.

1.3 Thesis Organization

This thesis begins with describing the basic concepts and standards prevailing in electronic learning. In Chapter 2, Learning Object Technology is introduced to inform readers about the core concept in e-learning issues, the Learning Objects (LOs), and explains why there is a need for them. It moves on describing standards and specifications in the e-learning area proposed by various standards committees with emphasis on Sharable Content Object Reference Model (SCORM) and IMS Simple Sequencing (IMS SS) specification. Some of these standards are used in the proposed thesis architecture. Reviews are provided for existing Learning Management Systems(LMSs) and show how these systems adhere or not to the e-learning standards. Moreover, reviews are given for related

work and research work on applying SOA techniques in LMSs by providing an abstraction of the core functionalities of systems and implementing them as Web services. The last section in this chapter gives an overview over the data models proposed by SCORM SN and IMS SS specifications in order to provide the basic knowledge over the data models proposed by these specifications as the thesis proposed engine is based on them.

Chapter 3 introduces readers to the Web service technologies by describing prevailing standards in the Web services area such as WSDL, SOAP, WS-Addressing, Web Service Resource Framework and BPEL, standards that this thesis is based on. It describes Service Oriented Architecture and Web services stack.

Chapter 4 describes the first phase of this thesis work. It explains how Learning Objects can be wrapped with Web services through the use of the Web Services Resource Framework. It gives an overview over the sequencing engine's implementation part that deals with the construction of stateful resources as WS-Resources and describes how data models presented in Chapter 2 correspond to resource property elements of the stateful resource.

In chapter 5, the general architecture is given explaining how the engine can be applied in existing learning management systems to add them the functionality of a sequencing engine. The behavior of the sequencing engine is implemented as a WS-BPEL process. A set of Web services has been implemented for the process and each one of them is described to provide information of how Simple Sequencing Behaviors correspond to these services.

The last chapter discusses the work done in this master thesis and proposes future work.

Chapter 2:
Learning Objects & e-learning Standards

The introduction begins by establishing some understanding about what Learning Objects are by stating the various definitions exist in the literature and providing information about the basic characteristics that Learning Objects should exhibit. In an ideal world, Learning Objects should be designed so that they could be adapted to fit different educational models, subject disciplines and levels of study. To fully realize the possibility of personalized and reusable educational content requires the establishment of standards for the design and description of learning objects. So current related e-learning standards and specifications are briefly described. The final section of this chapter introduces readers with the basic concepts of the SCORM Sequencing and Navigation specification to provide the necessary information needed for the work in this master.

2.1 e-learning

With the advent of Internet and World Wide Web (WWW) technologies, distance education has enabled a new era of education. The term e-Learning is used for describing the computer-enhanced learning process and it mainly refers to the delivery of educational content through any electronic media, such as Internet, intranets, extranets, satellite broadcast, audio and video tapes interactive TV, CD-ROMs, interactive CDs and computer-based training. Nowadays, tools like web-based teaching materials and hypermedia in general (web pages, discussion boards, simulations and games) are commonly used. Universities, educational institutions and organizations are adopting appropriate e-Learning platforms to support their learning activities and the range of use extends from simple sharing of text material to more complex forms of socializing[31]. Two communication technologies are used widely in the e-Learning platforms, synchronous and asynchronous. The first includes tools such as chats and video conferences well the latter is characterized by the use of tools like blogs, wikis or discussion boards.

2.2 Learning Object Definitions

The concept of Learning Object (LO) is central in the field of e-learning science. Many definitions have been provided so far in order to define the characteristics, attributes and the context of use of content used in e-learning. This section of the thesis will provide an overall idea of what Learning Object Technology is and will review several accepted definitions of Learning Object and the benefits that Learning Object Technology offers to e-learning.

The Learning Object Technology has been developed in order to meet the various needs expressed by individuals. The main problem faced so far was the use of very large in size learning content which parts could not be reused for the construction of other learning content. So, instructional designers used to provide these complete online courses ignoring the issues of reusability and interoperability. The above approach was proven to be inflexible, time consuming and expensive. The need for learning object to be readily available at any time and to address the need of each student has led to the development of the Learning Object Technology as the first step towards achieving personalized content.

The term Learning Object was first introduced by Hodgins in 1994 [12]. The first approach towards standardization of the term 'learning object' in order to achieve and ensure interoperability and

reusability of learning objects created by different instructional designers, organizations, universities and corporations was made from the Learning Technology Standards Committee (LTSC) of the Institute of Electrical and Electronics Engineers (IEEE) formed in 1996 to develop and promote instructional technology standards (LTSC, 2000a). This definition was published by IEEE and defines Learning Object as:

"Any entity, digital or non-digital, which can be used, re-used or referenced during technology supported learning. Examples of technology supported learning include computer-based training systems, interactive learning environments, intelligent computer-aided instruction systems, distance learning systems, and collaborative learning environments. (LOM, 2000)".

The problem with the above definition is that it is extraordinarily broad as it fails to exclude anything. So, many other definitions were provided by different groups outside the LTSC to achieve refining and narrowing of the above IEEE definition. One of the narrowest definitions comes from the NFS-funded Educational Objects Economy which takes a more technical approach and considers as Learning Objects only the Java applets [9]. In [20], a three-part definition is applied, including, the learning objective, a unit of instruction that teaches the objective and a unit of assessment that measures the objective. The fundamental idea behind Learning Objects is that the instructional designers can build small (relative to size of an entire course) instructional components that can be reused as many times in different learning contexts. Based on this assumption, learning object is "*any digital resource that can be reused to support learning*" [42]. According to [26], *a learning object is a digital learning resource that facilitates a single learning objective and which may be reused in a different context*. Through their work, they have introduced other possible forms of Learning Objects such as audio files, video clips, Flash animations, etc. Another approach [34] considers a Learning Object "*as a digitized entity which can be used, reused, or referenced during technology supported learning' as it focuses on two attributes: that Learning Objects must be digital and must be part of a learning event*".

An attempt to evaluate some of the existing definitions in order to specify a new definition that incorporated the main characteristics that a LO should exhibit: reusability and interoperability, is made in [32] and a learning object is defined as "*an independent and self-standing unit of learning content that is predisposed to reuse in multiple instructional contexts*". e-learning vendors tend to give a more specific definition to Learning Object that suits the implementation of those [11].

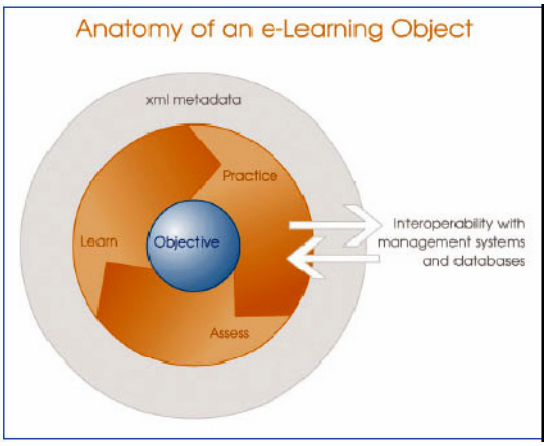


Figure 1: Macromedia's Conceptual Design of Learning Objects

Macromedia defines Learning Object as a unit of instructionally sound content centered on a Learning Objective or outcome intended to teach a focused concept [10]. As displayed in Figure 1, Macromedia presents the idea of learning object as consisting of five parts with the learning objective being central to the learning object:

Despite the inability to find and agree upon a unanimous definition and an appropriate building metaphor, there is an agreement upon the benefits of designing and developing material to be reused as learning objects. These benefits are summarized by [23] in the following categories:

- *Flexibility*, if material is designed to be used in multiple contexts;
- *Easy of updates, searches and content management* with the use of metadata for filtering and selecting only the relevant content for a given purpose;
- *Customization,* just-in-time customization of modular learning objects to personalize content according to the needs of an individual or an organization;
- *Interoperability* with other learning systems and contexts is retained when Learning Objects are designed based on the recommended specifications and standards;
- *Facilitation of competency-based learning* with the use of Learning Object's metadata tags that allow the matching between learning object metadata and individual competency tag;
- *Increased value of content* every time it is reused. This is reflected not only in costs saved by avoiding new design and development time, but also in the possibility of selling content objects or providing them to partners in more than one context.

All different definition of Learning Object are basically "Learning Objects" are all Learning Technology Standards Committee "Learning Objects" in the strictest sense [42]. Despite the benefits listed by Longmire, Learning Object Technology will only be used widely in educational practice if a wide variety of Learning Objects become readily available and if educators finally feel comfortable about using Learning Objects and become adept at searching and creating Learning Objects [25].

2.3 Standards and specifications

Various specifications and standards have been around for many years for online learning and computer-based training, but the adoption of these has been slow. In this section, these specifications of IMS, ADL and AICC and well-known standards such as IEEE PAPI and IEEE LOM are analyzed.

Numerous organizations and consortia exist that work in the area of e-learning standards. For instance, organizations such as the Dublin Core Metadata Initiative, the IEEE, the IMS Global Consortium, the Alliance of Remote Instructional Authoring and Distribution Networks for Europe (ARIADNE), the Aviation Industry CBT committee (AICC), the Advanced Distributed Learning Initiative (ADL) are dedicated or have committees and working groups active in the establishment of learning standards.

The aim of the above working groups is to achieve interoperability between different systems by accomplishing the free sharing of educational content and learner information with the use and implementation of learning standards that standardize the communication models between many systems. Other specifications aim to the reusability of learning content which results in minimizing the effort required to prepare high quality learning materials. Learning contents can be developed once by experts and used many times by others regardless of their learning contexts or environments. Most of the specifications and standards are developed to facilitate the description, packaging, sequencing and delivery of educational content, learning activities and learner

information[5]. The goals of all courseware interface specifications are to promote interoperability, accessibility, durability and reusability.

The four organizations that are leading e-Learning developers towards standards are the Aviation Industry Computer-Based Training Committee (AICC), the Instructional Management System Global Learning Consortium (IMS), the Institute of Electrical and Electronic Engineers (IEEE) and the Advanced Distributed Learning (ADL) initiative.

2.3.1 AICC

The first organization that formulates standards and specifications for e-learning is the Aviation Industry CBT (Computer-Based Training) Committee (AICC) and focuses mainly in the aviation industry as denoted by the name. This committee was shaped from a group of technology-based training professionals in 1988 with the purpose to develop guidelines for aviation industry concerning the development, delivery and evaluation of CBT and related training technologies. They try to promote interoperability between CBT courses and computer managed instruction (CMI) record keeping systems and present a standard syntax and application programming interface (API) for communication between courseware and CMI. The API implementation is a Javascript approach. This was the first run-time interoperability specification for LMS (Learning Management Systems). Another specification promoted by the AICC which is the most interesting one is HACP (HTTP based AICC/CMI Protocol) for CMI that is widely used by LMS and other systems to call content and assessments. AICC HACP protocol is an HTTP-based communication protocol, which enables a web-based course or test to report progress data back to the LMS. Whereas, the work of AICC was not widely adopted outside the government and commercial aviation circles, they set the stage for standards to follow. The importance of this specification is outline by being the base for other learning standards and specifications such as runtime environment data model and API used in SCORM (see Section 2.3.4).

2.3.2 IMS

The IMS Global Consortium [14], also known as IMS/GLC, first started in 1997 and it is a consortium of vendors and implementers, which came into existence as a project within the National Learning Infrastructure Initiative of EDUCAUSE. It is a non-profit organization supported by over 120 organizations that strive to enable the growth and impact of learning technology in the education and corporate learning sectors worldwide. The IMS/GLC has approved and published over 20 standards. Unlike AICC efforts, the IMS standards and specifications are the most widely adopted ones in the e-learning world.

The IMS/GLC mainly focuses on supporting and promoting interoperability of learning resources by investing on the development of XML-based specifications. Moreover, they strive for distributed learning including both online and off-line settings taking place synchronously (real-time) or asynchronously. Some of the subjects that they have been concerned about is metadata, content packaging, common cartridge, enterprise services, question & test, sequencing, and many more. IMS has promoted the core data formatting and packaging specifications that are used in both the AICC and ADL specifications, which in turn have expanded the IMS specifications. This thesis is based on the concepts of the IMS Content Packaging, the IMS Simple Sequencing (SS) specification as they have been the base of the ADL SCORM specification (see Section 2.3.4). Another important specification of IMS is the IMS Digital Repositories Interoperability (DRI).

2.3.2.1 IMS Content Packaging

IMS Content Packaging is a specification for sending learning resources from one virtual learning environment (VLE) or content management system or digital repository to another in a form of an IMS package. It facilitates the easier delivery, reuse and sharing of materials while retaining information describing the media in the IMS package, and its structure. This specification describes an XML file divided into three parts: the metadata (use of the IEEE LOM elements to describe the package), a table of contents or else *organizations* (lessons or other entry points into the resource used by the course and how they are organized) and the resources (list of references to various files, metadata describing resources and URLs -references to external files- utilized by the course). The resulting XML file is called manifest. The following picture illustrates the components of an IMS Content Package.

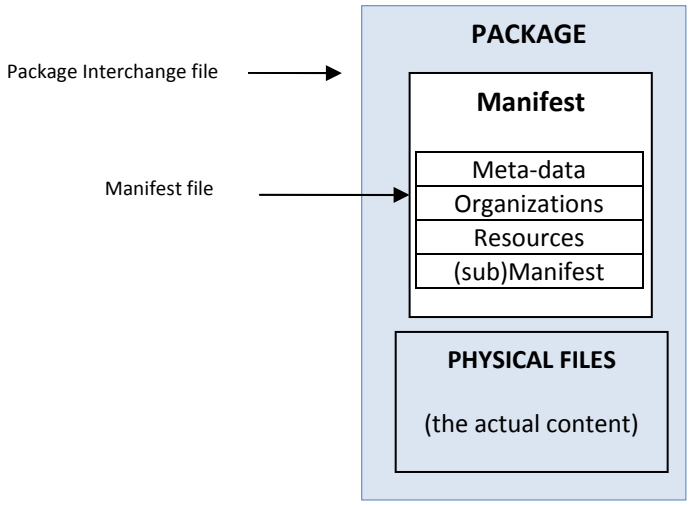


Figure 2: IMS Content Package Description

One of the key benefits of this specification is that all materials or learning resources are bundled together in order to protect mostly the HTML-based materials from broken links or image icons. Additionally, the IMS package can be disaggregated, so resources inside the package can be reused as individual learning objects. Another key benefit is that content packages contain a metadata section through enabling filtering and searching of its content when stored in a repository of learning objects.

2.3.2.2 IMS Simple Sequencing

The *IMS Simple Sequencing (SS)* specification released to the public in March 2003 and defines "a method for representing the intended behavior of an authored learning experience such that any learning technology system (LTS) can sequence discrete learning activities in a consistent way" [17]. It specifies the relative order in which electronic learning activities are to be presented to a learner and the conditions under which a resource is selected, delivered or skipped during presentation. Its primary goal is to present a model and mechanism to describe learning content that captures common structures of learning patterns, including conditional delivery and branching.

It provides the means to represent information needed to sequence learning activities in a variety of

ways according to two data models, the Tracking Model[1] and the Activity State Model[2]. It also defines the required behaviors and functionality that conforming learning management systems must exhibit.

Another useful data model is the *Sequencing Definition Model* (SDM) introduced by the IMS Simple Sequencing and it is an information model used to describe the desired sequencing behaviors.

According to [41-X], the IMS SS specification can be used in various areas of e-learning such as:

- Creating multiple paths through one set of learning activities.
- Creating patterns of formative[3] assessment.
- Create summative[4] assessments.
- Creating Decision Trees.
- Adding Context Sensitive Network Services to Content.
- Creating Step through tutorials.

The IMS SS relies heavily on the concept of "learning activities" such as content or test questions. Learning activities corresponds to learning resources in an IMS Content Package as described above. Activities are associated with other activities into a hierarchy, resulting in an *Activity Tree* as shown in the following figure**.**

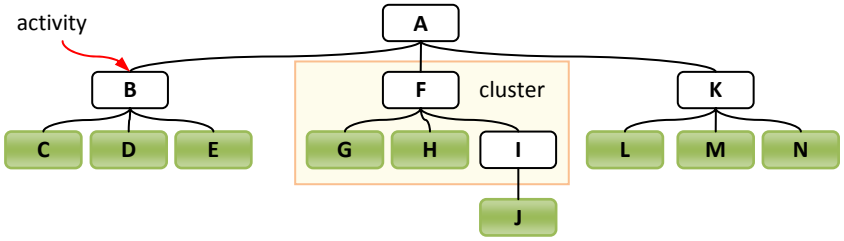


Figure 3: IMS Simple Sequencing Activity Tree

2.3.2.3 IMS Digital Repositories Interoperability Specification

The last specification to overview is the IMS Digital Repository Interoperability (IMS DRI) which final form was released in 30th January 2003 [16]. Its main purpose is to provide recommendations for digital repositories in order to support the interoperation of the most common repository functions and provide a common interface between different types of repositories. Digital repositories are defined as "any collection of resources that are accessible via a network without a prior knowledge of the structure of the collection. Repositories may hold actual assets or metadata that describe assets. The assets and their metadata do not need to be held in the same repository".

[1] Tracking Model is used to record information about the learner's interactions with activities and the learner's record for objectives to control the selection and the sequencing of other activities.

[2] Activity State Model is used to record information about the status of learner's interactions with an activity and set of global attributes for activities.

[3] Formative Assessment is part of the instructional process. It informs both learners and instructors about learner's understanding at a point when timely adjustments can be made

[4] Summative Assessment refers to the assessment of the learning and summarizes the development of learners at a particular time.

The working group mainly focuses on two areas concerning this topic, the digital repositories of learning objects and their metadata and the integration of e-learning with existing and emerging online digital services. This specification aims to address the problem of searching resources across multiple repositories possibly supporting different representations of metadata and heterogeneous access methods. No new schemas are introduced but the existing ones such as IMS Content Packaging and IMS Meta-data are utilized.

The IMS DRI provides a functional architecture of a digital repository as shown in the following figure:

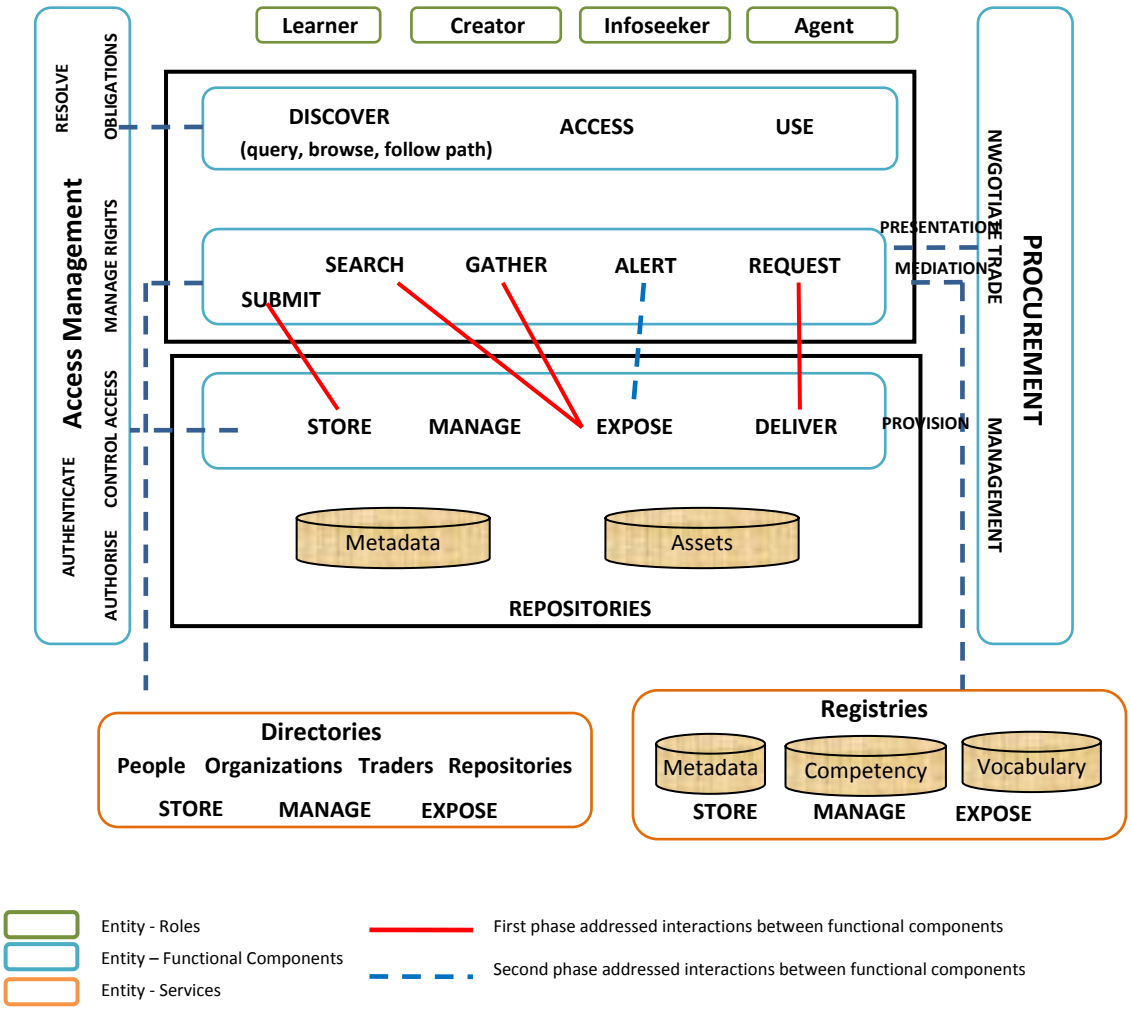


Figure 4: IMS DRI functional architecture

It also provides guidelines for the most common functions exhibited by the repositories and identifies the XQuery, Z39.50 and SOAP as the key technologies for the implementation of these functions:

- *Search* the metadata associated with content exposed by repositories;
- *Gather* metadata exposed by repositories in order to be used in subsequent searches or creating a new metadata repository to be used for Search/Expose Alert functions;

- *Alert/Expose* when a specified metadata record is modified in one or more repositories;
- *Submit/Store,* a resource and its associated content from one network-accessible repository to another location that can be a repository, a learning management system or any other network location;
- *Request/Deliver*. Request access to a resource located via Search or Alert functions and provides access to this resource.

2.3.3. IEEE LTSC

The Institute of Electrical and Electronic Engineers Learning Technology Standards Committee is chartered by the IEEE Computer Society Standards Activity Board and its purpose is to develop internationally accredited technical standards, recommended practices and guides for learning technology. It coordinates with other organizations both formally and informally, that produce specifications and standards for learning technology. The LTSC coordinates formally and informally with other organizations that produce specifications and standards for similar purposes [24]. Standards development is done in working groups via a combination of face-to-face meetings, teleconferences, and exchanges on discussion groups. The LTSC is governed by a Sponsor Executive Committee (SEC) consisting of working group chairs and elected officers.

2.3.3.1 Learning Object Metadata

The most widely acknowledged IEEE LTSC e-learning standard is the IEEE Learning Object Metadata (LOM). The IEEE LTSC LOM is a multipart standard. The part IEEE 1484.12.1-2002 was approved by the IEEE-Standards Association and defines the structure of a metadata instance for a learning object, which definition fits in with the broad IEEE definition of a learning object. According to this part of the standard, a metadata instance for a learning object "describes relevant characteristics of the learning object to which it applies". The standard defines the LOM data model which specifies which aspects of a learning object should be described and what vocabularies may be used for these descriptions. It also defines how this data model can be amended by additions or constraints. Other parts of the Standard are being drafted to define bindings of the LOM data model, i.e. define how LOM records should be represented in XML and RDF (*IEEE 1484.12.3* and *IEEE 1484.12.4* respectively).

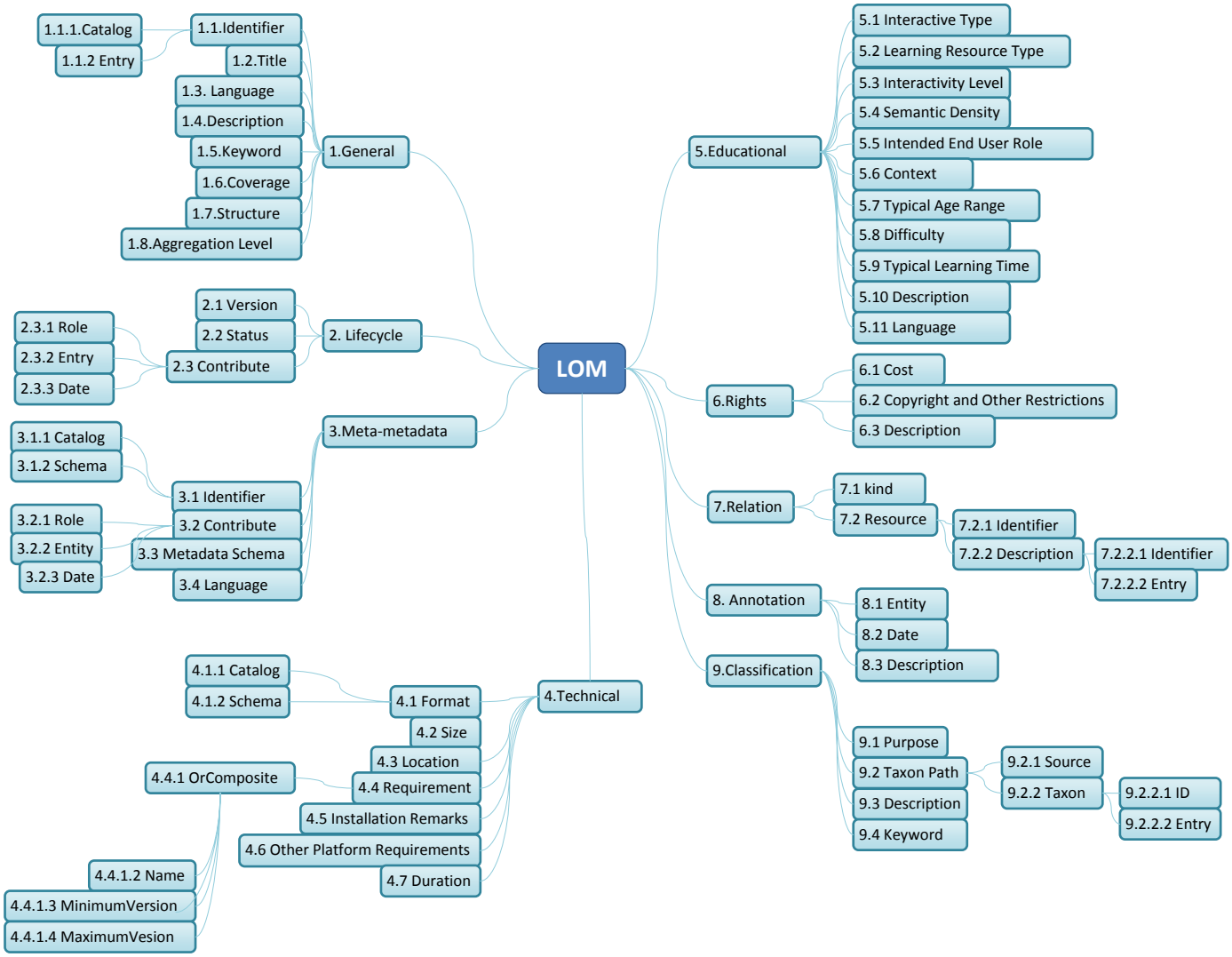


Figure 5: IEEE Learning Object Metadata

As shown in the diagram in Figure 5, the LOM comprises of a hierarchy of elements starting with nine categories in the first level. Each category contains sub-elements which may be simple elements holding data or may themselves contain sub-elements. The value space and data type for each of the simple type elements are also defined in the data model. The value space defines the restrictions on the data that can be entered for the element and the data type allowed for elements are either string of characters or LangString[5] items or Vocabulary[6] items or DateTime and Duration[7] items.

According to the LOMv1.0 standard, the nine categories are:

[5] LangString items contains Language and String parts, allowing the same information to be recorded in multiple languages.

[6] Vocabulary items are the items that their entries are constrained and their values have to be chosen from a controlled list of terms. It supports the Source-Value pairs, with the Source containing the name of the list of terms being used and the Value containing the chosen term.

[7] DateTime and Duration Items contain one part that allows the date or duration to be given in machine readable format, and a second that allows a description of the date or duration.

a) *General*, groups the general information that describes the learning object as a whole;

b) *Lifecycle*, groups the features related to the history and current state of this learning object as a whole;

c) *Meta-Metadata*, groups information about the metadata instance itself (rather than the learning object that the metadata instance describes);

d) *Technical*, groups the technical requirements and technical characteristics of the learning object;

e) *Educational*, groups the educational and pedagogic characteristics of the learning object;

f) *Rights*, groups the intellectual property rights and conditions of use for the learning object;

g) *Relation*, groups features that define the relationship between the learning object and other related learning objects;

h) *Annotation*, provides comments on the educational use of the learning object and provides information on when and by whom the comments were created;

i) *Classification*, describes this learning object in relation to a particular classification system.

When utilizing/implementing the LOM standard, it isn't obligatory to support or use all the elements in the data model nor needs the LOM data model to limit the information which may be provided. Elements from the LOM data model may be excluded from the metadata instance while elements from other metadata schemas can be included. The vocabularies in the LOM may be supplemented with values appropriate to the community.

The IEEE LTSC Learning Object Metadata standard enables data and service providers to provide well structured descriptions of their learning resources that facilitate the discovery, location, evaluation and acquisition of them by students, teachers or automated software processes. Furthermore, LOM can be used with other resource specifications or other resource description schemas to "tag" learning resources to generate the final resource description. From this description, it is permitted to reduce the number of LOM elements being used or add new elements according to a schema specification. So, a resource description that suits to the specialized needs of individuals is constructed. Finally, a reduction in the cost of providing services between resource discovery systems is achievable through the sharing of learning resource's description.

2.3.3.2 IEEE Public and Private Information Specification

Another interesting specification from IEEE is the Public and Private Information Specification (PAPI) which is a standard format for the representation and communication of student profiles. The goal is to allow the creation of student records which can be communicated between educational systems over the lifetime of a learner. It is based on view of intelligent systems where the performance information is the most important part. It also considers interpersonal relationships [13].

The profile has four areas, the Personal information for private uses like the learner's name and address, the Preference Information for public use like the learning style of the learner physical limitations or disabilities, the Performance information to be used by systems and consists of the

observable behavior of the learner and may include grade, reports and logs and the Portfolio information which is a history of previous academic achievements.

The PAPI also incorporates the Dublin Core metadata element set. In general the information in the profile may be inferred, directly input by the learner, or constructed by cooperation of the learner and the system. PAPI tries to address the privacy and security issues related to the stored data.

2.3.4 ADL

The Advanced Distributed Learning Initiative [1] was founded by the US Department of Defense (DoD) and the White House Office of Science and Technology Policy in 1997 to develop a strategy and standards for using information technologies to modernize education and training. Its vision is to "provide access to the highest quality education, training and performance aiding, tailored to individual needs, delivered cost effectively, anytime and anywhere". ADL's strategy is to cooperate with industry, government and academy to promote common, open international specifications and standards that will enable reuse and interoperability of learning content. The ADL high-level requirements for Web-based learning content and systems are: interoperability, accessibility, reusability, durability, maintainability and adaptability.

It succeeded to create probably the most used standard for e-learning: the Sharable Content Object Reference Model – SCORM.

2.3.4.1 Sharable Content Object Reference Model

SCORM 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, IMS, ARIADNE and others) to meet the above ADL's requirements. It fosters the notion of reusable learning objects as "instructional objects". It standardizes communication between management systems and content and the way of aggregating and packaging content as re-usable, self-contained objects that contain metadata

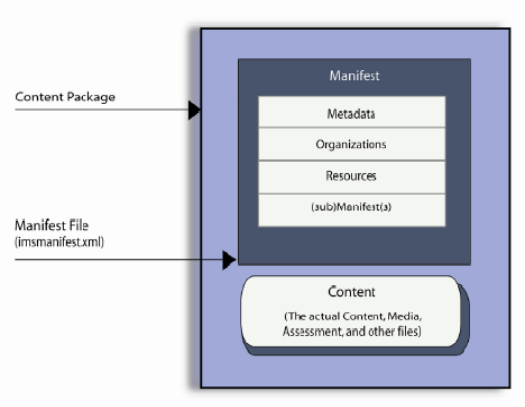


Figure 6: SCORM Conceptual Content Package [36]

The final version of SCORM, SCORM 2004 3rd Edition, appears in October 2006. It is consists of three main technical books, the *Content Aggregation Model*, the *Run-Time Environment* and the *Sequencing and Navigation*. The *Content Aggregation Model* is based on the *IMS Content Packaging* specification and describes packaging of content and learning object, labeling, exchange and discovery of content packages. A Content Package groups one or more learning resources and a content organization and may represent a course, a lesson, a module or just a collection of related

content objects. As described in Section 2.3.2.1, the content organization is described in the imsmanifest.xml file.

The *Run-Time Environment* describes run-time API and data model used for communication between content objects and learning management systems. Its purpose is to guarantee a communication between a specialized form of learning object, the SCO, and LMS from different sources. It also describes the SCO, the standard communication mechanism and the data formats used in this communication. Figure 7 depicts the architecture of a typical Run-Time Environment. The LMS is responsible for providing implementation of the SCORM API while the SCO is responsible for initializing the communication session between itself and the LMS. The SCO uses ECMAScript, commonly known as Javascipt, to communicate with the API implementation provided by the LMS. Finally, SCORM RTE specification provides a common set of defined data fields or data elements that allow SCOs to communicate with different LMS (see Appendix B).

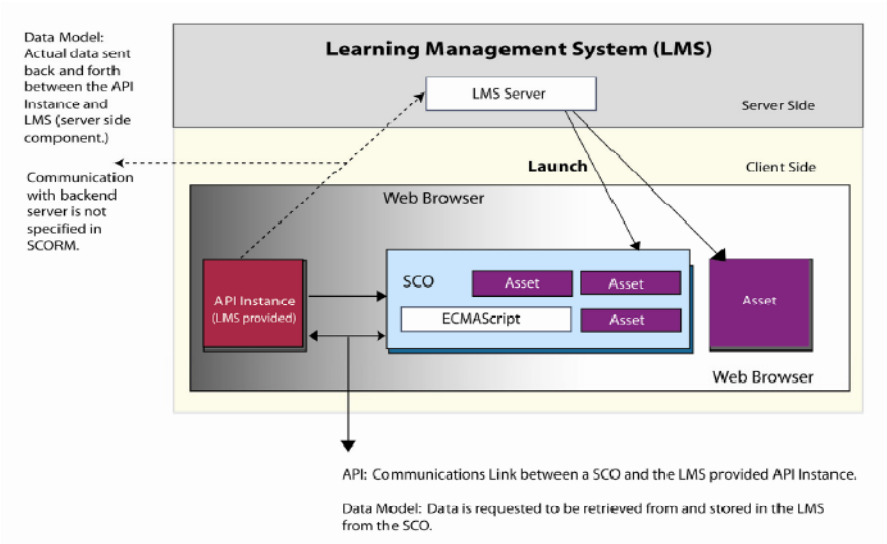


Figure 7: SCORM Conceptual Run-Time Environment [36]

The *Sequencing and Navigation* describes how sequencing between learning activities is defined and interpreted. It defines the behaviors and functionality that an LMS should exhibit and implement to process sequencing information. It utilizes the term 'Activity Tree' to describe the branching and flow of learning activities based on an authored, prescribed, sequencing strategy and the learner's interactions with content objects at run-time. It also deals with navigation tools.

SCORM extends the IEEE LTSC Learning Object Metadata (LOM), the IMS Learning Resource XML Binding and IMS Simple Sequencing Definition Model, the Dublin Core Metadata Initiative vocabularies and the AICC Computer Managed Instruction (CMI) data model.

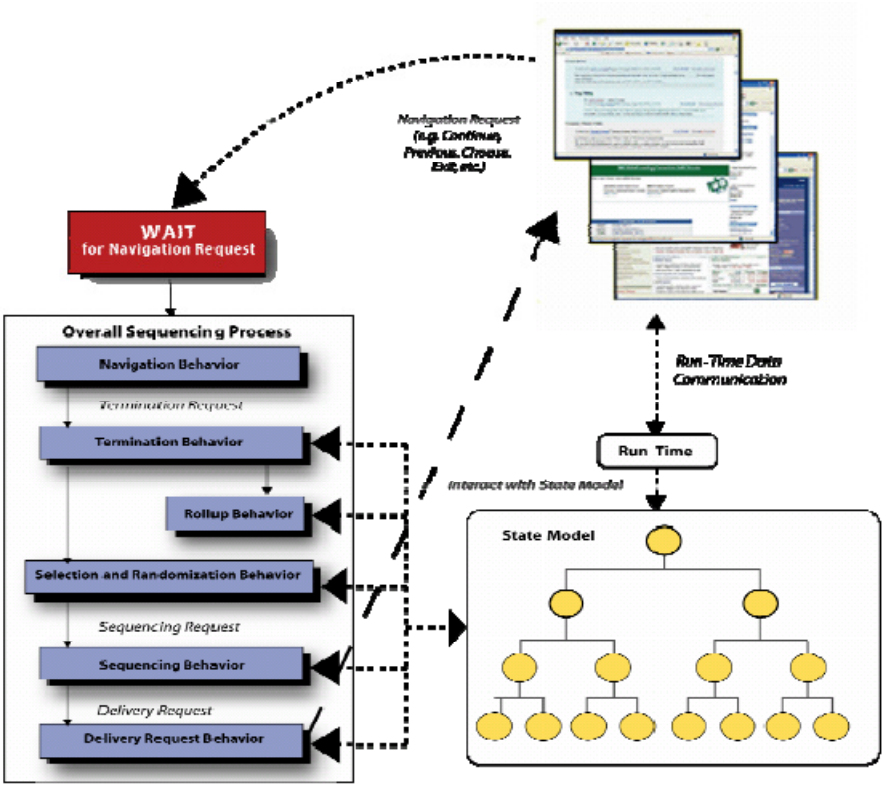


Figure 8: Conceptual Model of a Sequencing Engine [36]

2.4 ADL SCORM 2004 3rd Edition, Sequencing and Navigation

This thesis is based mainly on the SCORM Sequencing and Navigation specification and it provides implementations for the basic data elements and functionalities that are introduced in this specification. SCORM SN encompasses the IMS Simple Sequencing specification (IMS SS). The SCORM SN describes how IMS SS is applied and extended in a SCORM environment. The main purpose of the IMS SS is to enable conformant learning management systems or other run-time components to sequence learning activities in a consistent way, thus promoting interoperability between these systems. The IMS SS standard is labeled as "simple" because it defines a limited number of widely used sequencing behaviors. The SCORM SN furthers extends the IMS SS Sequencing Behavior and describes the functionality that LMSs should implement as well as a navigation data model that enables communication between SCOs and LMSs with the aim of sequencing learning content during run-time. This communication enables LMSs to track learner progress while content is presented to the latter and allows SCOs to indicate navigation requests. So, the final flow of learning activities of a content package is based on the results of the learner's interactions with the content and the authored sequencing strategy.

In the following subsections, general SCORM SN concepts will be introduced, including the basic terminology used such as activity trees, clusters, learning activities, the sequencing information that can be applied to learning activities to define an intended strategy (sequencing definition model), the sequencing behaviors that an LMS should implement and the navigation data model.

2.4.1 Sequencing and Navigation Concepts and Terminology

2.4.1.1 Learning Activity and Learning Experience

A fundamental concept of the IMS SS specification is that of the learning activities. The IMS SS describes learning activity as an instructional event or events embedded in a content resource or as an aggregation of activities that eventually resolve to discrete content resources with their contained instructional events. It is conceptually something the learner does while progressing through instruction.

For example, in the following figure, "*Python Data Structures*" is an activity which is composed of three

sub activities, "*Lists*", "*Tuples*" and "*Dictionaries*". During experiencing with "Python Data Structures", learner will also experience the three sub-activities. There is no limit on the permissible number of levels of sub-activities. Additionally, leaf activities are the activities that do not contain a level of sub-activities and so they are considered to have a content object associated with them. When an attempt begins on a leaf activity, the associated content will be launched by the LMS and both a learner attempt and a learner session will begin for the content object. The series of content objects that the learner experiences, is called a "*learning experience*".

2.4.1.2 Activity Tree and Clusters

As we have pointed out earlier, IMS Simple Sequencing Specification describes paths through a predefined set of learning activities. The layered structure of this collection is called *Activity Tree* and it is typically defined at design time.

Each activity in the Activity Tree may have an associated set of sequencing behaviors, which are defined in the *Sequencing Definition Model*. The sequencing behaviors describe how the activity or how the children of the activity are used to create the desired learning experience.

The sequencing behavior process traverses the activity tree, applying the sequencing rules, to determine the activities to deliver to the learner. Content resources from the identified activities are delivered to the learner to create the desired learning experience.

However, the SCORM SN specification does not constrain authoring tools and LMSs neither on a specific structure of the Activity Tree or on the time of its instantiation. Moreover, the SN does not necessitate the Activity Tree to be a static structure, but LMSs are free to dynamically change the structure of the Activity Tree and the sequencing information applied to the activities of the tree as long as the Sequencing Definition Model (refer to Section 2.4.1.3.3) and the Sequencing Behaviors are adhered to. We must consider an Activity Tree as a general term that represents an instance of hierarchical learning activities and the corresponding sequencing information for the interoperable application of specified sequencing behaviors

It is important to point out that Activity Tree co-exist with Content Organization which was introduced in the previous section. They both represent two different "*views*" on the same objects. Activity trees are derived from the content packages.

A *cluster* is the basic block in the Activity Tree and it is a specialized form of an activity and includes a single parent activity and its immediate children but not the descendent of the activity. A leaf activity is not considered a cluster. The parent activity of a cluster contains the information of the sequencing strategy for the cluster. As we mentioned above, leaf activities of the cluster associates with content objects and they are identified for delivery according to the defined sequencing strategy.

2.4.1.3 Data Model Elements introduced by IMS SS and SCORM SN

Three information models are defined in the IMS Simple sequencing specification, "*Tracking Model*", "*Activity State Model*" and "*Sequencing Definition Model*". Tracking Model is used to record learner's interactions with activities to control the selection and sequencing of other activities (see Section 2.4.1.3.1). Activity State Model is used to record the state or status of learner's interactions with an activity and a set of global attributes for activities (see Section 2.4.1.3.2). Sequencing and Definition Model is used to describe the desired sequencing behavior (see Section 2.4.1.3.3).

2.4.1.3.1 Tracking Model

Tracking model is a set of data model elements that describe the tracking information dynamically maintained for each activity in the Activity Tree, while a learner interacts with launched content object associated with delivered activities. This data model is directly managed from LMS because some SCORM Run-Time Environment Data Model elements correspond directly to elements in the Tracking Model.

Initially, all the tracking model elements are initiated with their default values and during learning experience, these elements are updated to reflect learner interactions with the currently launched content object. Changes in the value of tracking information model elements for an activity may affect the value of the activity's parent tracking status information. The process of evaluating the tracking status for an activity based on the change of tracking status of one of its children is referred to as *rollup*. The *rollup* behavior is initiated by the LMS when tracking status information needs to be updated. This set should be initialized for each activity in the Activity Tree for each learner.

The tracking model defines the following sets of tracking information:

- ***Objective Progress Information***

An activity in the Activity Tree may be associated with local learning objectives in order to determine the learner's state on the activity. These objectives are considered as satisfied or not and a measure may be recorded. For each learning objective of an activity, one set of Objective Progress Information exists per attempt.

Objective Progress Information consists of the following set of elements, the default values of which are used by the LMS until explicit ones are set by the LMS's sequencing implementation

Table 1: Objective Progress Information Elements

Data Element	Description	Default Value
Objective Progress Status	Indicates whether the objective currently has a valid satisfaction value.	Boolean: false
Objective Satisfied Status	Indicates whether the objective is satisfied or not	Boolean: false
Objective Measure Status	Indicates if the objective has a measure value	Boolean: false
Objective Normalized Status	Holds the measure (e.g. standardized score) for the	Real decimal:

	objective	-1.0 … 1.0

➢ *Activity Progress Information*

The *Activity Progress Information* contains the tracking status information of an activity, which has been denoted as tracked, that extends all attempts on that activity.

It consists of the following set of elements, the default values of which are used by the LMS until explicit ones are set by the LMS's sequencing implementation.

Table 2: Objective Progress Information

Data Element	Description	Default Value
Activity Progress Status	Indicates whether the activity progress information is meaningful or not for the activity.	Boolean: false
Activity Absolute Duration	Contains the cumulative duration of all attempts on the activity. (the time from the initial start of the activity to the end of the activity)	Decimal
Activity Experienced Duration	Contains the cumulative *experienced* duration of all attempts on the activity. (the time from the initial start of the activity till the end not including any time elapsed while the activity is suspended)	Decimal
Activity Attempt Count	Indicates the number of attempts on the activity	Non-Negative Integer: 0

➢ *Attempt Progress Information*

The *Attempt Progress Information* is a set of elements that contains information that has been gathered for each attempt on an activity that has been denoted as "tracked".

It consists of the following set of elements, which LMS uses their default values until explicit ones are set by the LMS's sequencing implementation.

Table 3: Attempt Progress Information Elements

Data Element	Description	Default Value
Attempt Progress Status	Indicates whether the attempt progress is meaningful or not for the activity attempt. The value of this element is True when some piece of tracking information is recorded for the current attempt on the activity.	Boolean: false
Attempt Completion Amount	Describes the degree of completion of the current attempt on the activity. The value is normalized between 0…1 (inclusive) where 1 means that the activity attempt is complete and any lesser value means the attempt is not complete.	Real Number: 0..1(inclusive)
Attempt Completion Status	Indicates whether the activity attempt is completed or not	Boolean: false
Attempt Absolute Duration	Denotes the duration of the current attempt on the activity. (i.e time from the start of the attempt to the end	Duration

	of the attempt)	
Attempt Experienced Duration	The total experienced duration of current attempt on the activity.	Duration

2.4.1.3.2 Activity State Model

Activity State Model describes the state of the most recent attempt at an activity and the overall state of sequencing. Activity State Information describes the status for an activity. The elements described in the Activity State Information are shown in the following Table:

Table 4: Activity State Information Elements

Data Element	Description	Default Value
Activity is Active	Indicates that an attempt is currently in progress for an activity The activity has been delivered to the learner and has not been terminated yet Is an ancestor of the *Current Activity*(see Table 5)	Boolean: false
Activity is Suspended	Indicates the activity is suspended	Boolean: false
Available children	An ordered list of the activity's available children	*All children*

Global State Information describes the overall state of the sequencing and this information is initiated once for the Activity Tree. The elements are described in the following table:

Table 5: Global State Information Elements

Data Element	Description	Default Value
Current Activity	Indicates the current activity in the Activity Tree. Current Activity will be one from the following: If an activity is experienced by a learner, the current activity is the activity delivered by the most recently completed *Content Delivery Environment Process.* If an activity is not experienced by a learner, the current activity is the last activity identified to terminate by the most recently *Termination Request Process* or the undeliverable target activity of the lass successful *Choice Sequencing Request Process.*	Boolean: false
Suspended Activity	Indicates the activity is suspended	Boolean: false

A cluster is considered suspended if any of its children has the "Activity is Suspended" equal to true. The suspended state of an activity describes how the next attempt on that activity is initiated. The next attempt on the activity will resume the previous attempt and use the previous tracking model, no new tracking information will be initialized

2.4.1.3.3 Sequencing and Definition Model (SDM)

The SCORM SN defines a set of elements that can be used to describe and affect various sequencing behaviors. These elements may also be used by content developers to define intended sequencing behaviors. This information model is derived from IMS SS and SCORM enhances it by defining

additional specific elements in order to extend behaviors and restrictions beyond those defined in IMS SS.

This information model is applied to learning activities within the context of the Activity Tree. Each element of this set has a default value that is to be assumed by any sequencing implementation in the absence of an explicitly defined value. The effects of these elements only apply during the application of SCORM Sequencing Behaviors.

In order LMS to be conformant with the SCORM standard, they must support the behaviors that results from the values associated with all of the defined SDM elements, including both explicitly declared and default values.

The Sequencing Definition Model elements are organized in the following categories:

- ***Sequencing Control Modes(SDM 1)***

The elements in this category are used to constrain a desired learning experience. The Control Modes can be applied, with any combination or not, to any activity in the Activity Tree, however, some of them will have no effect if applied to leaf activities. These elements are used to determine whether or not a navigation request will translate into a valid sequencing request, to affect how the activities will be considered for delivery and to affect how tracking status information is managed. A default value exists for each Control Model that will be used, if the element is not explicitly defined for an activity.

Table 6: Sequencing Control Modes

Data Element	Description	Default Value
Sequencing Control Choice	Determines those activities in a cluster that a learner is free to choose.	Boolean: true
Sequencing Control Choice Exit	Indicates the activity can be terminated if the learner chooses an activity that is not a descendant of this.	Boolean: true
Sequencing Control Flow	If true, the sequencing engine is allowed to choose any child from the cluster depending on the sequencing request “Continue” or “Previous”	Boolean: false
Sequencing Control Forward Only	If true, the sequencing engine must traverse the children of a cluster only in a forward order. This means that a “Previous” navigation request is not allowed.	Boolean: false
Use Current Attempt Objective Information	Determines if the *Objective Progress Information* (Tracking Model) will be used in the evaluation of rules and rollup.	Boolean: true
Use Current Attempt Progress Information	Determines if the *Attempt Progress Information* (Tracking Model) will be used in the evaluation of rules and rollup.	Boolean: true

- ***Sequencing Rules***

The elements in this category employ a rule-based model. Zero or more Sequencing Rules can be applied to activities in the context of the Activity Tree and they are evaluated during sequencing behaviors with the use of tracking information. Sequencing Rules consists of a set of conditions and a set of actions. Their form is:

If [condition-set] then [action]

With each rule, there is a behavior associated with that and is performed when the condition set is evaluated to *True*. The *condition-set* parameter will check a learner’s progress and performance for

current activity. The action parameter provides the intended action for the learning management system during the sequencing process. The actions are categorized into three groups based on

- the timing they will be evaluated,
- the pre-condition actions which are to control sequencing decisions when traversing the Activity Tree to determine if an activity will be delivered,
- the post-condition actions, which control sequencing flow by issuing sequence requests and applies when an attempt on an activity terminates and
- the exit actions which are applied after a descendant activity's attempt terminate.

The possible values of the conditions and the actions of sequencing rules can be found in the Appendix A.

➢ ***Limit Conditions***

There are optional elements that define constraints on the access to an activity. Limit Conditions are conditionals based on an activity tracking status information. When a limit condition is met or exceeded, the activity becomes unavailable for delivery. Whenever is needed, the default data is used if the data is not instantiated for the activity.

SCORM SN in contrast with IMS SS only requires the support for *Limit Condition Attempt Limit* and does not require the evaluation of any time-based limit conditions.

Limit Conditions are separated into the following two categories, "*Attempt Limits*" and "*Attempt Absolute Duration*". The first set provides the means to content developers to limit the number of the attempts permitted on an activity and the second set provides the means to content developers to limit the duration that can be spent in a single attempt of a learning activity. For a description of these sets' elements refer to Appendix A.

➢ ***Auxiliary***

SCORM does not require implementing this data element as there is no guarantee of interoperability.

➢ ***Rollup Rules***

There is no direct way to apply learner progress information to a cluster activity because clusters do not associate with content objects. IMS SS provides the Rollup Rules to describe the relationship of learning status between children activity and their parent. The term "Rollup" refers to situation in which a change in the tracking information of a learning activity may affect the tracking status of its parent activity.

The rollup behaviors are initiated by LMS when tracking status information needs to be updated. In each activity, zero or more Rollup Rules can be applied. The structure of a Rollup Rule is:

```
if [condition_set] True for [child activity set] then [action]
```

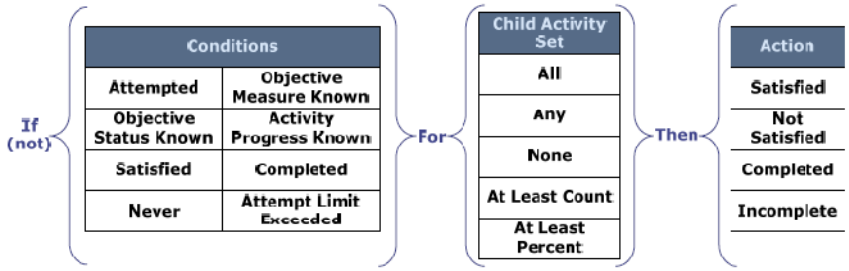


Figure 9: Rollup Rules Structure

Each Rollup Rule consists of a set of child activities to consider, a set of conditions evaluated against the tracking information of the included child activities and a corresponding action that sets the cluster's tracking status information. Rollup Rules have no effect when applied to leaf activities.

There are two possible ways to combine the conditions, either with a single '*and'* combination (all conditions must be evaluated to True) or a single *'any'* combination (only one condition must me true).Each condition references an item in the tracking model for the activity. The conditions that reference objectives refer to the objective that has the attribute value *Objective Contributes to Rollup* value of True. There is only one action that may result if the rollup conditions evaluate to True and this action may result in setting of the *Attempt Completion Status* or *Objective Satisfied Status* for the activity. The rollup condition and actions are described in Appendix A.

By default, all children of a cluster contribute to their parent's rollup, if they are tracked. IMS SS restricts the possible children by defining the Rollup Controls. The following table provides information for these elements:

Table 7: Rollup Rules

Data Element	Description	Default Value
Rollup Objective Satisfied	It indicates whether the activity contributes to the evaluation of its parent's *Satisfied and Not Satisfied* Rollup Rules. If this element is defined as False then the LMS will not consider the activity's tracking status information in any of its parent's Rollup Roles that have a Satisfied or Not Satisfied *Rollup Action*, even if the activity has tracking status information recorded.	Boolean: true
Rollup Objective Measure Weight	It serves as a weighting factor applied to the *Objective Normalized Measure* for the objective, which has the *Objective Contributes to Rollup* equal to True, associated with the activity during rollup for the parent activity.	Real type: 1.0
Rollup Progress Completion	It indicates whether the activity contributes to the evaluation of its parent's *Complete and Incomplete* Rollup Roles. If this element is defined as false, the LMS will not consider the activity's tracking status information in any of its parent's Rollup Rules that have a Completed or Incomplete Rollup Action, even if the activity has tracking status information recorded.	Boolean: true

- ***Objective***

The term *learning objective* refers to a set of locally or globally scoped data items, each with satisfaction status and satisfaction measure. Learning objectives are separated from activities in IMS SS but the latter makes no assumption as to how to interpret objectives. Activities in an Activity Tree may associate with zero or more local learning objectives or they may reference globally shared objectives. Furthermore, multiple activities may reference the same global learning objective, thus sharing the data values.

Each learning objective associated with the activity will have a set of tracking status information that allows learner progress toward the learning objective to be tracked, thus enabling conditional sequencing decisions.

Table 8: Objective Elements

Data Element	Description	Default Value
ObjectiveID	A unique identifier of an objective which can also be used as a link to the corresponding *Objective Progress Information*.	ID

Objective Satisfied by Measure	Indicates whether the element *Objective Minimum Satisfied Normalized Measure* is to be used in order to determine if the objective of the activity has been satisfied or not.	Boolean: false
Objective Minimum Satisfied Normalized Measure	A real type element ([-1.0 … 1.0] (inclusive)) that indicates the minimum satisfaction measure for the objective. If the *Objective Measure Status* for the objective is true and the *Objective Normalized Measure* for the objective is equal to or exceeds this value, the *Objective Progress Status* is set to True and the *Objective Satisfied Status* is set to True. If the *Objective Measure Status* for the objective is true and the *Objective Normalized Measure* for the objective is less than this value, the *Objective Progress Status* is set to True and the *Objective Satisfied Status* is set to False.	Real type: [-1.0 … 1.0]
Objective Contributes to Rollup	A Boolean type element (default value is false) that indicates whether the *Objective Satisfied Status* and *Objective Normalized Measure* for the objective will be used during Rollup.	Boolean: false

By default, an activity can access *Objective Progress Information* for those objectives defined for the activity and these objectives are considered as "local". This means that two activities can assign the same ObjectiveIDs to their associated objectives.

In order to relate an activity's local objective to a globally shared objective, an *Objective Map* must be defined. In this way, multiple activities can share the same *Objective Progress Information*. It is allowed for an activity to have an unlimited number of *Objective Maps* by defining a set of *Objective Maps* to describe how local objective information is mapped to shared global objectives. The *Objective Map* data are evaluated whenever local objective information is processed. For any given local objective, a 'read' map with at most one global objective may be defined and also for any global objective, for any activity, a 'write' map with at most one local objective can be defined.

The following elements can be used to describe an Objective Map

- *Activity Objective ID:* A unique identifier for a local objective associated with the activity.
- *Target Objective ID:* An unique identifier of a global shared objective targeted for mapping
- *Read Objective Satisfied Status:* A Boolean type element (default value is true) that indicates that the *Objective Progress Status* and *Objective Satisfied Status* values for the identified local objective (*Activity Objective ID*), should be retrieved (True or False) from the identified shared global objective (*Target Objective ID*), when the progress for the local objective is undefined (*Objective Progress Status* for the identified local objective is False).
- *Write Objective Satisfied Status:* A Boolean type element (default value is false) that indicates that the *Objective Progress Status* and *Objective Satisfied Status* values, for the identified local objective (*Activity Objective ID*), should be transferred (True or False) to the identified shared global objective (*Target Objective ID*), upon termination of an attempt on the activity.
- *Read Objective Normalized Measure* A Boolean type element (default value is true) that indicates that the *Objective Measure Status* and *Objective Normalized Measure* values for the identified local objective (*Activity Objective ID*), should be retrieved (True or False) from the identified shared global objective (*Target ObjectiveID*), when the measure for the

local object is undefined (*Objective Measure Status* for the identified local objective is False).

- *Write Objective Normalized Measure:* A Boolean type element (default value is false) that indicates that the *Objective Measure Status* and *Objective Normalized Measure* values, for the identified local objective (*Activity Objective ID*), should be transferred (True or False) to the identified shared global objective (*Target Objective ID*), upon termination of an attempt on the activity.

➢ ***Randomization and Selection Controls***

Selection Controls are an optional set of elements (attributes) that might be defined for any activity in an Activity Tree and are used to define sequencing information that indicates when to select certain activities and limit the number of activities to be chosen. Default values will be used if no selection of control element is defined for an activity. These elements have no effect when applied to leaf activities.

Randomization Controls are an optional set of elements(attributes) that can be applied in any cluster in the Activity Tree and describe the way that children of an activity can be reordered during the sequencing process (while performing various sequencing behaviors). These elements have no effect when applied to a leaf activity.

For a description of these sets' elements refer to Appendix A.

➢ ***Delivery Controls***

Delivery Controls is an optional set of elements (attributes) that describes actions and controls used when an activity is delivered and shall be used to by LMSs to aid in the management of the activity's tracking status information. For a full description of these elements refer to Appendix A.

2.5 Theoretical Studies and Frameworks

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 [40]. In [56] 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 [40] 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 [6] 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 [7] 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 [37] 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 [38], 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 [8] an approach towards the personalization of learning content based on distributed environments based on semantic web services is presented.

2.6 Existing LMSs and SCORM support

A considerable great number of open source and commercial Learning Management Systems exist in the web that can be used to facilitate e-Learning in organization, industries and in academia. Some of the most popular ones are described in the following table:

Table 9: Learning Management Systems and SCORM integration

LMS	Type	SCORM	Web Services support
ATutor 1.6.2 http://www.atutor.ca/	Open-source	SCORM 1.2 SCORM 2004 Content Packaging	no
Claroline 1.8.11 http://www.claroline.net	Open-source	SCORM 2004 Content Packaging SCORM 2004 Run-Time API	no
Moodle 1.9.3 http://moodle.org/	Open-source	Fully Support of SCORM 1.2 SCORM 2004 Content Packaging SCORM 2004 Run-Time (partially)	Web service module for integrating Web Services
DoceboLMS 3.6 http://www.docebo.org/doceboCms/	Open-source	SCORM 1.2 (fully). SCORM 2004 Content Package. SCORM 2004 Run-Time API. SCORM 2004 Sequencing and Navigation still in beta form.	no
Blackboard http://www.blackboard.com	Commercial	SCORM 1.2 (fully) SCORM 2004 Content Packaging SCORM 2004 Run-Time API SCORM 2004 Sequencing and Navigation, not evidence that supports this.	no
Desire2Learn http://www.desire2learn.com/	Proprietary System	SCORM 1.2 SCORM 2004 Content Packaging SCORM 2004 Sequencing is not fully supported	Web Service interface for retrieving data from the LMS.

The integration of SCORM 2004 is not fully accomplished yet. Most of the above platforms support the importing and exporting of learning content packaged in SCORM Packages as described in the SCORM Content Aggregation specification and they provide implementations for the SCORM 2004 Run-Time API for the communication of the Learning Objects with the LMS. Only Desire2Learn and DoceboLMS have tried to implement a sequencing engine based on the SCORM Sequencing and Navigation specification but their work is still in beta form and their engine is not fully functional. Furthermore, none of the platforms are designed based to Service-Oriented Architecture. Moodle, which base functionality is extended by modules implemented by its large community, provides a support for integration of Web Services by utilizing the Web Service Module. In addition,

the base functionality of Moodle does not support importing or exporting of SCORM packages but this functionality is implemented in the AICC module. All of them supports SCORM 1.2 specification, but the latter does not contain the Sequencing and Navigation specification.

2.7 Summary

This chapter has introduced readers to the Learning Object Technology and current e-learning standards and highlights the basic concepts and definition models defined in the IMS Simple Sequencing specification and adopted by the SCORM Sequencing and Navigation Model. The knowledge of these concepts is essential for better understanding the SCORM standard which this work is based on. The next chapter will discuss the Web Services technologies.

Chapter 3
Web Service Technologies

This chapter introduces readers with the concepts and of the Service Oriented Architecture (SOA) and Web services. As the web base approach is based on several standards, they are briefly described in the following sections. Great emphasis is placed on the description of concepts defined in the Web Service Resource Framework and the WS-BPEL standard as both are the building blocks of this thesis.

3.1 Service-Oriented Architecture

Service-oriented architecture 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 perceived value of SOA is that it provides a powerful framework for matching needs and capabilities to address needs expressed by entities (people and organizations). Capability is defined as an ability to perform a specific set of functions resulting in a real world effect. A cornerstone of SOA is that capabilities can be used without needing to know all the details.

Service-oriented architecture can be implemented using various technologies like Web Services, Service Component Architecture (SCA), Enterprise JavaBeans (EJB), CORBA and so on. It is possible for a service developed to have different kinds of interfaces. For example, a service can have a web service interface and a Java-based SCA-service interface. However, Web services technology is the most common new technology for implementing Service oriented Architecture. Despite some current limitations, SOA with Web services is an ideal combination of architecture and technology for consistently delivering robust, reusable services that support present business needs and that can without difficulty be adapted to satisfy changing business requirements [27]. SOA based on web services aims at simplifying integration by providing universal connectivity to existing systems and data.

3.2 Web Services

Web services have de facto become the most significant technological-by-product primarily because Web services provide a distributed computing approach for integrating extremely heterogeneous applications over the Internet. 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 artifacts. A Web Service supports direct interaction with other software agents using XML-based messages exchanged via internet-based protocols*". Many developers extend this definition to require that the services be defined and described using the Web Services Description Language (WSDL) [46] standard and the Simple Object Access Protocol (SOAP) [45] be used as a communication protocol. The general idea behind Web service is outlined in the following definition, which describes web services as "any process that can be integrated into external systems through valid XML documents over Internet protocols" [35]. Examples of web services include online reservation, ticket purchase, stock trading, and auction.

The heart of the Web Services is the standards [39]. Three key XML-based standards have been defined to support web service deployment, the Simple Object Access Protocol (SOAP) as a messaging envelope format, the Web Services Description Language (WSDL) as service description format and the Universal Description Discovery and Integration (UDDI) [44] for defining metadata that is used for service discovery. These are standards proposed by Organization for the Advancement of Structured Information Standards OASIS and W3C.

The Web Service Reference Model comprises of three types of participants exhibiting specific operations in order to cooperate together, the *service provider* that implements the service and publish the service description in a service registry, the *service client* (or *requester*) that represents a human or a software agent and that uses the service and the *service registry,* a searchable registry that provides service descriptions. The figure 10 shows these collaborations in the web service model.

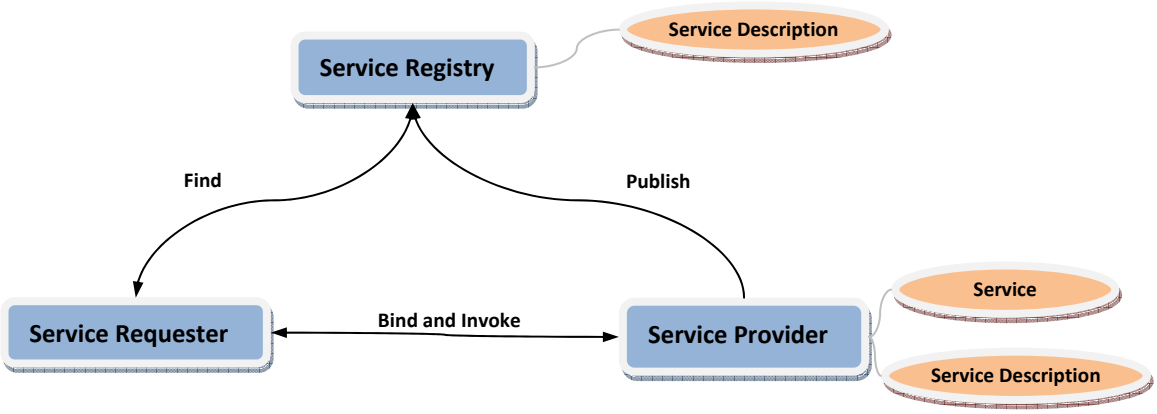


Figure 10: Web Services Reference Model

A complete Web Service Protocol Stack is illustrated in the following figure 11:

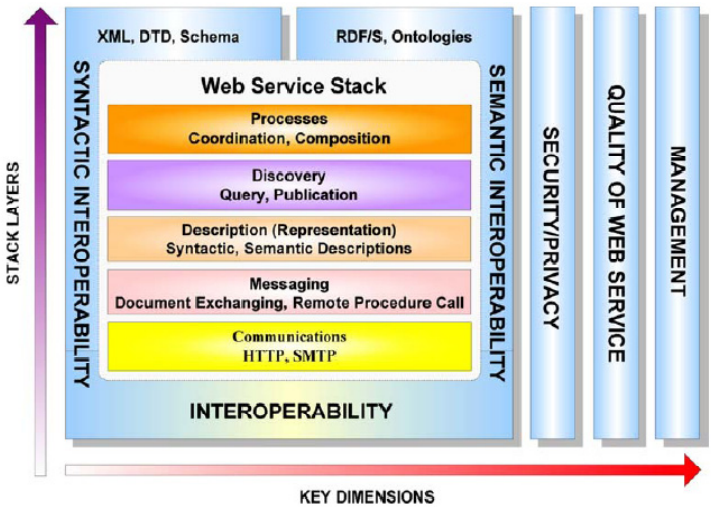


Figure 11: Web Services Protocol Stack [33]

The five key layers are: communications, messaging, descriptions, discovery and processes, which are shown along the vertical direction in figure. Each layer provides certain functionality to support interoperation between Web Services and service clients or among Web services. The Web Service protocol stack provides capabilities for loosely integrating software services and in its full form is

shown in the above figure. Briefly, web services interact by passing XML data, with types specified using XML Schema. SOAP can be used as a communication protocol and the I/O signatures for web services are given by the WSDL. All of those can be defined before binding web services to each other. The messaging, representation and discovery layers achieve simple interactions with Web Service, where service requestors invoke a single operation. The process layer supports more complex interactions between Web Services. It relies on the basic interaction functionalities provided by the technologies at lower layers in the Web Service stack. For example, it needs the discovery and representation layers for querying and locating Web Services. Then, the selected services can used to construct a process, which consists of a sequence of coordinated Web services. The process layer enables organization to outsource existing Web services, reduce complexity because complicated services are constructed from simple ones, and reduce business risks and increasing organization's revenue by outsourcing their services.

The Web services Interoperability Organization has an important role as standards integrator to help Web services advance in a structured and coherent manner. WS-I standards and guidelines are seen as an enabler for Web service interoperability. Web services Interoperability Organization (WS-I) is an open industry consortium of about 150 companies, representing diverse industries such as automotive, consumer packaged goods, finance, government, insurance, media, telecommunications, travel and other computer industries. The objectives of the WS-I is to promote web services interoperability across platforms, operating systems, and programming languages with the use of generic protocols for interoperable exchange of messages between services, to encourage web services adoption and to accelerate deployment by providing guidance, best practices and other resources for developing interoperable web services. It supports the relationships between standard bodies who own specifications and fosters communication and cooperation with industry consortia and other organizations. Other standards bodies also active in Web services include Internet Engineering Task Force (IETF), Java Community Process (JCP), and Object Management Group (OMG). Web Service development based on these standards is supported by different implementation platforms such as .Net and J2EE [19].

3.2.1 SOAP

The Simple Object Access Protocol (SOAP) is a communication protocol between applications and depends on the XML language for the format of messages.

A typical SOAP message written in XML consists of the following parts: an *Envelope* element to identify that the message is a SOAP message, a *Header* element that contains header information, a Body element contains call and response information and a *Fault* element containing error and status information.

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
   <soap:Header>
      ...
      ...
   </soap:Header>
   <soap:Body>
      ...
      ...
      <soap:Fault>
        ...
        ...
      </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

Listing 1: SOAP Message Elements

The SOAP protocol is language and platform independent. It allows for application communication running on different operating systems, with different technologies and programming languages. But it is considered to be slower than any other middleware technologies such as CORBA.

The SOAP body of a message element depends on the style of Web service. In an RPC-literal style, the Web service is invoked as a remote procedure call; in this case, the SOAP body element contains the name of the method and the parameters as XML payload. In a document-literal style, only the XML payload is added to the body element and it is not wrapped with the method name.

3.2.2 WSDL

The Web Service Description Language (WSDL) is an XML language for describing interfaces of Web services. When a new service is developed, a WSDL file is written for it and is published in a public service repository. Clients can find this WSDL file to use the service. In a WSDL file, there is information regarding where the service is located, the operations it supports and the messages format for its operations.

WSDL describes four critical pieces of data: the data type information for all message requests and responses, interface information describing all public operations, binding protocol about the transport protocol to be used and address information to locate the service.

Listing 2 shows a basic WSDL file and its components:

```
<definitions>
        <type> Definitions of types</type>
<message>…</message>
<portType>….</portType>
<binding>…..</binding>
<service>….</service>
</definitions>
```

Listing 2: WSDL Message Elements

- ***<type>:*** is used to describe data types to be used in messages or operations. Inside "type" element, an XML schema is used to define data types.
- ***<message>:*** is used to describe data elements of an operation. It can be used as a request or response message of an operation. Its child elements are zero or more part elements used to describe the body of the message. The parts can be compared with the parameters passed into a function and the message serves as a wrapper.
- ***<portType>:*** is used to define one or more operations of the Web service. For each operation, we declare the input and output messages. PortType defines the
- ***<binding>:*** is used in conjunction with a portType to define the transport protocol and the style of web service. For each operation declared in the portType, a soap action must be defined and for its input and output message should be specified if they will be encoded or not. It describes how the Web service will be implemented on the wire.
- ***<service>:*** defines the address for invoking the web service

3.2.3 WS-Addressing

In this thesis, this specification [47] is fully utilized as we will see in Chapter 4. This specification standardizes the endpoint reference (EPR) that is used to identify/reference a Web service deployed

at a given network endpoint. It defines a protocol-neutral mechanism to address WS messages. A WS-Addressing endpoint reference is an XML structure encapsulating useful information for addressing a Web service and it can contain additional metadata and parameters, called *reference properties*, to route the message to the destination. This specification also specifies message headers to encapsulate in SOAP messages. These information headers include addressing for source and destination endpoints as well as a message identity.

3.3 Business Process Execution Language

One of the main concepts introduced by SOA is the ability to connect single services into more complex ones. Web services have been characterized as self-contained modular business process applications that are based on standard technologies such as WSDL, UDDI and SOAP and help us to connect different components even across organizational boundaries in a platform and language independent manner. 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. 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 how they are involved in the business process, joint exception handling for collections of web services and other issues. It allows us to specify long-running transactions between Web services, increasing consistency and reliability for Web services applications.

The process-oriented approach to SOA requires a language for relatively simple description of how Web services should be composed into business processes. Web services can be composed in two ways: *orchestration* and *choreography*. In orchestration, a central process takes control over the involved Web services and coordinates the execution of different operations on the Web services involved in the operation. The involved Web services do not know that are involved in a web service composition and that they are part of a higher business process. Only the central coordinator knows this so the orchestration is centralized with the explicitly definitions of operations and the order of invocation of web services (see figure 12). Choreography on the other hand does not rely on a central coordinator. Each web service involved in the choreography knows exactly when to execute its operations and whom to interact with. It is a collaborative effort focused on exchange of messages. All the participants of the choreography need to be aware of the business process, operations to execute, messages to exchange and the timing of message exchanges (see figure 12).

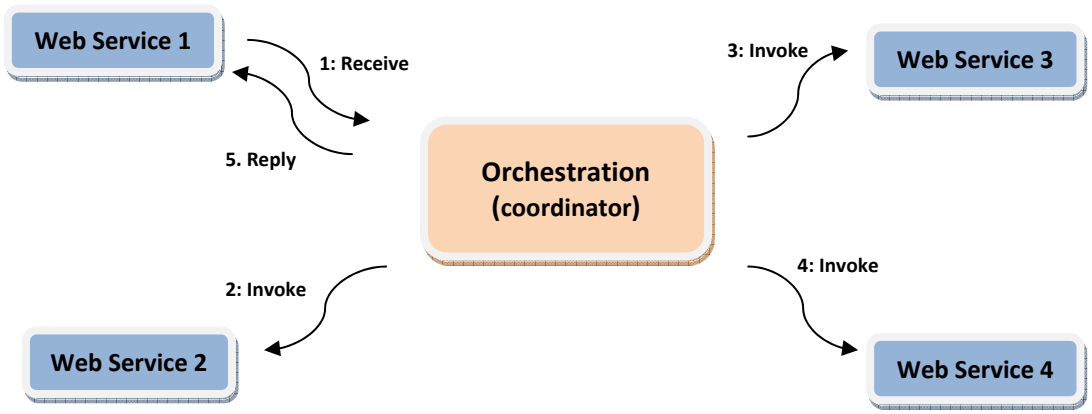


Figure 12: Orchestration of Web services

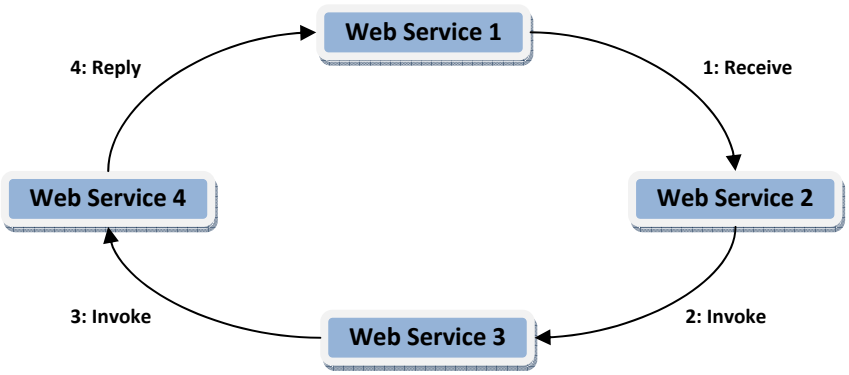


Figure 13: Choreography of Web services

The Business Process Execution Language for Web services (WSBPEL or BPEL) [28] is used to define and execute complex business processes with Web services. It is an XML-based workflow definition language developed by the OASIS to specify business processes as a set of interactions between web services. It combines IBM's Web Service Flow Language (WSFL) and Microsoft's XLANG, which is the orchestration language used by the Microsoft's Biztalk server. It combines the features of a block structured process language (XLANG) with those of a graph-based process language (WSFL). BPEL provides the constructs to build business processes and these business processes are fully executable and portable between BPEL-conformant environments.

BPEL provides the artifacts (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.

BPEL is an orchestration engine for describing exchanges of information internally and externally. BPEL deals explicitly with the functional aspects of business processes: control flow (branch, loop, parallel), asynchronous conversations and correlation, long running nested units of work, faults and compensation. BPEL directly addresses these business process challenges: coordinating asynchronous communication between services, correlating message exchanges between parties, implementing parallel processes of activities, manipulating data between partner interactions, supporting long-running business transactions and activities and providing consistent exception handling.

A major design goal of BPEL is to separate the public aspects of business process behavior from the internal ones. This separation helps BPEL to conceal internal decisions from their business partners. Moreover, internal changes of the process implementation no longer affect the public business protocol. Therefore, BPEL has both abstract and executable processes to support the separation.

Abstract process is a partially specified process that is not intended to be executed as it describes only the behavior by hiding certain information such as modeling artifacts. It serves a descriptive role and it is used to guide executable processes. An abstract process also refers to the business protocol as it specifies the public aspects of in the business interactions. Specifically, an abstract process only deals with the message exchanges between business partners. It is isolated from the execution of a process flow. An *executable process* contains the logic and the state of the process. It specifies the sequence of the Web service interactions conducted in the business of each business partner.

3.3.1 BPEL and WS-* Stack

BPEL is layered on top of and extends the WSDL service model. The WSDL defines the specific operations allowed while BPEL defines how WSDL operations are orchestrated to satisfy a business process. BPEL also specifies extensions to WSDL in support of long-running asynchronous business processes. Furthermore, BPEL processes follow the WSDL service model so they are considered and implemented as services. They have WSDL definitions and are implemented as Web services (WSDL, SOAP, UDDI).

BPEL supports the web services technology stack including SOAP, WSDL, UDDI, WS-Reliable Messaging, WS-Addressing, WS-Coordination and WS-Transaction. It uses WS-Addressing for the dynamic assignment of Web services endpoints and expects processes to leverage WS-Transaction for long running business transactions. BPEL also recommends the use of WS-Security framework to address authenticity and other security concerns.

3.3.2 BPEL Elements

A process described with BPEL can be either synchronous or asynchronous. A synchronous process blocks the client until it finishes and returns a result, while an asynchronous one does not block the client and uses callback to return the result to client. The building blocks of BPEL are the descriptions of the parties participating in the process, the data that flow through the process and the activities performed during the execution of the process. BPEL processes can be executed via their own Web service interface, or through internal triggers defined inside the process. An external trigger is a message received on a port exposed by the process, internal triggers are time driven and defined inside the process.

A BPEL process specifies the exact order in which participating web services should be invoked and it is described in an XML file with extension .bpel. A BPEL process always starts with the “process” element. Inside “process” there must be at least one activity either a primitive or a structured one. Activities represent the steps in a BPEL process and describe the interaction of the process with its partners. Primitive activities refer to specific actions that the business process must execute with respect to the constituent Web services.

Primitive activities represent basic constructs and are used for common tasks such as:

- *invoke*: process invokes, asynchronously or synchronously, other Web services that have been defined as partners.
- *receive*: this activity plays an important role in the lifecycle of a business process. It is usually used to initiate the process and its main task is to block and wait for an incoming message.
- *reply*: is used to send a response to clients
- *assign*: is used to manipulate variables by updating their values
- *throw*: is used to generate and signal faults from inside the process
- *wait*: specifies a period of time to wait or a period of time to pass. It offers the possibility to build in a certain time to sleep or wait until a specified deadline has passed.
- *exit*: immediately terminate the running process
- *empty*: no-op instruction. Sometimes it’s necessary for the process not to do anything, like suppressing faults.
- *compensate:* invoke compensation on all completed child scopes in default order
- *compensateScope:* invoke compensation on one completed child scope
- *extensionActivity:* wrapper for language extension

Structure activities are used to combine primitive activities to define complex algorithms inside a process. With structured activities control patterns, data flow, fault handling and coordination of messages can be achieved:

- *flow*: is used to combine primitive activities to be executed in parallel
- *pick*: associates activities with events and waits until an event is triggered. The activities corresponding to the event are executed. The event that occurs first is processed if multiple events are triggered. Pick can be used like the receive activity in order to initialize a process.
- *sequence*: allow us to structure primitive activities that will be executed in a sequential order. The order of execution is determined by their place inside the structure activity.
- *while*: is used to define iterative walk through certain activities inside process.
- *if-elseif-else:* conditional branching inside process
- *repeatUntil*: just like while. It executes the containing activity while a condition is true
- *scope*: is used to split a process into parts. A scope can have its own variables, fault handlers, partnerlinks, etc. It provides the context for its enclosed activity.
- *compensationHandler*: it defines a set of activities to be executed when a problem is occurred in the process. It is used to undo certain steps that have already been completed. Another use of the compensationHandler is to define alternative steps that need to be done when a certain event happens.
- *faultHandler*: is attached to a scope in order to define fault handling activities ("catch").
- *eventHandler*: specifies the steps to take when certain events happen. The two types of eventHandler is the "*onMessage*" and the "*onAlarm*".

For its client, a BPEL process looks like any other Web service. When we define a new BPEL process, we actually define a new Web service that is a composition of existing ones. The interface of the new BPEL composite Web service uses a set of *portTypes* through which it provides operations like any other Web service. To invoke a business process described in BPEL, we have to invoke the resulting composite Web service.

Another important BPEL element is the "*partnerLink*", which represents the interaction of a BPEL process with the involved parties, either Web services or clients. At least one client partner link is mandatory as there has to be a client that invokes the BPEL process. Furthermore, client partner links can be used to support asynchronous interactions. A BPEL process can further provide different functionalities to different clients and this distinction is feasible with the use of partner links. For each partner link one(synchronous) or two(asynchronous) roles are specified and these roles are associated with portTypes.

3.3.3 Building a BPEL process

In an orchestration scenario, the messages from the ports are processed by the BPEL instances running at a partner site and are subject to that partner's business rules. The next step is to define the variables which can be XML types or elements and WSDL message types. Variables are used as input and output containers for service invocation and to store the state of the process. Following the variable declaration, fault handlers are specified in order to define sets of activities that need to be executed when an error occurs while process is executed. Finally, a set of activities, both primitive and structured ones, are defined that makes up the main body of the process execution. An abstract view of a BPEL process is depicted in the following figure:

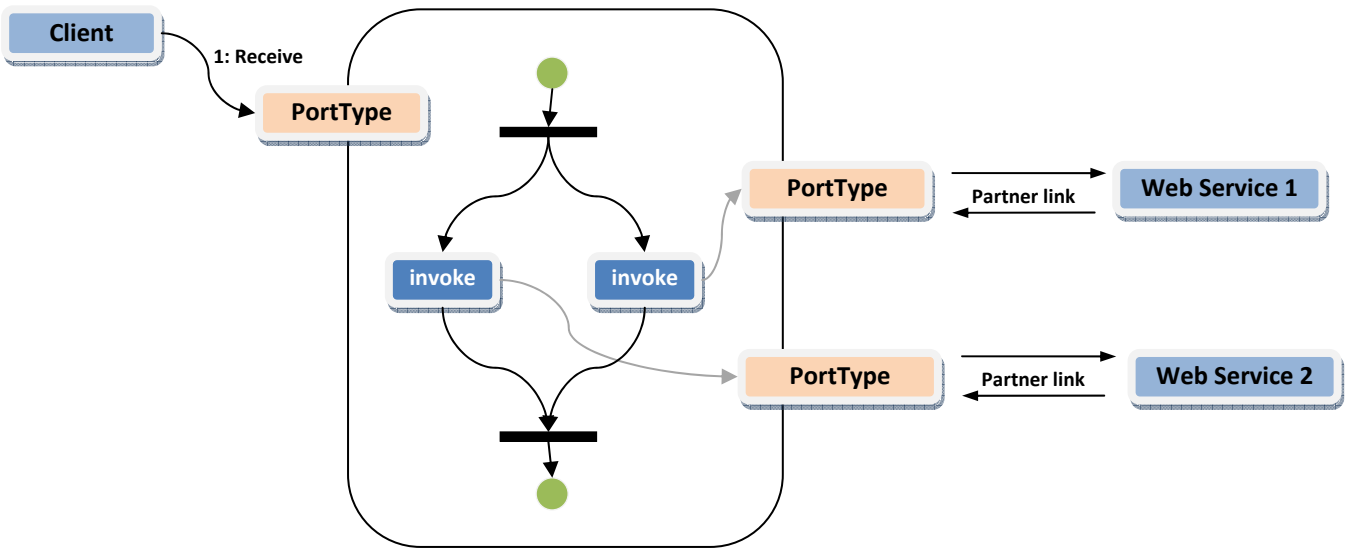


Figure 14: Abstract View of BPEL partner links

3.3.4 BPEL Engines

To execute BPEL executable processes an orchestration engine is needed, which provides the run-time environment for the execution. BPEL is strongly related to web services and to the modern software platforms that support Web service development, particularly to Java 2 Enterprise Edition (J2EE) and Microsoft .NET. BPEL orchestration servers exist both for J2EE and .NET. BPEL engines execute, coordinate and monitor running process instances defined by a .bpel file. Existing engines are ActiveBPEL engine from Active Endpoints, ODE engine from Apache Software Foundation, Oracle BPEL Process Manager, Italio, Twister from Agila BPEL.

3.4 The Web Services Resource Framework

This section is dedicated to the Web Resource Framework (WSRF) specification published by the OASIS. This specification is vital to this thesis because important design issues are based on it. This specification consists of a set of proposed Web services specifications for modeling, 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.

3.4.1 WS Resource Framework Family of Specifications

The WSRF consists of the following technical specifications:

- *WS-ResourceLifetime* (WSRF-RL) defines the means for destroying and managing the lifecycle of a WS-Resource [51].
- *WS-ResourceProperties* (WSRF-RP) defines the means for defining the properties of a WS-Resource and their projection in XML documents and how these properties can be queried, modified or deleted. The documents are only a view of the state of a WS-Resource and it is typically not equivalent to the state [52].
- *WS-ServiceGroup (WSRF-SG)* defines the ways to aggregate or group together Web services and WS-Resources for domain specific purposes and specifies the management functionalities for these collections [53].

- *WS-BaseFaults* (WSRF-BF) defines a standardized a common XML fault type for WSRF and the rules for its usage by Web services [49].
- *WS-Notification* and *WS-RenewableReferences*

The above specifications are based on XML schemas and WSDL interfaces.

3.4.2 WS-Resource Framework Concepts

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. Files in a filesystem, rows in a relational database, accounts in a bank, jobs in a job submission system or even objects such as Enterprise Java Beans (EJB) can all be modelled as stateful resources.

The key concept in WSRF is the *WS-Resource* [50]. The WS-Resource is the construct used to model stateful resources. It is an entity that maintains the state between calls made to it and that can be accessed through a Web services interface. A WS-Resource is defined as the composition of a resource, which can be used in the execution of Web service message exchanges, and a Web service. WS-Resource can be addressed and accessed with the use of WS-Addressing *endpoint reference* in terms of an implied resource pattern. The resource is not attached to any URL but it provides the URL of the Web service that manages it.

WSRF does not define how WS-Resources can be implemented or represented. Furthermore, 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).

The resource property elements reflect different parts of the resource state and can be repeated multiple times. The cardinality of a resource and its resource property elements is one to many. Each resource property is an XML element and the whole resource can be serialized in XML format to be embedded in a SOAP message before sending across the network. The set of these properties constitutes the *resource properties document* which is an XML document defined using XML Schema and it is a projection of the actual state of the WS-Resource.

WSRF specification describes the relationship between Web services and stateful resources in terms of an implied resource pattern. The *implied resource pattern* is a convention of XML, WSDL and WS-Addressing that allows the state of a resource, in terms of a resource properties document, to be defined and associated with the description of a Web service interface. The term *"implied*" is used because the identity of the resource associated with the message exchange is not a part of a request message but it is implicitly contained in the message using the *reference properties* feature of WS-Addressing. The resource's identity is implied by the context of the message and its association with an endpoint reference. The term *"pattern"* is used to indicate that the association of a Web service with a stateful resource is described with these conventions. This WS-Resource qualified endpoint reference contains two important components: the *wsa:Address* that refers to the network transport-specific address of the Web service, which is the one that appears in the service's WSDL port and the *wsa:ReferenceProperties* that contains the XML serialization of the resource's identifier and must appear as SOAP header elements in message exchanges.

The *WS-ResourceProperties* specification outlines the declaration of a WS-Resource's properties definition as a part of the Web service interface. This means that service requestors can determine the WS-Resource's type by retrieving the WSDL portType definition. This resource property document is attached to the WSDL's portType that defines the Web service interface. This association defined the WS-Resource. Each instance of a particular WS-Resource must implement a logical resource properties document of the type associated with the WSDL portType. At the type level, a WSDL 1.1 portType, defining the interface to a Web service, can be associated with at most one 'kind' or 'type' of a WS-Resource. One 'type' of WS-Resource can be associated with many

WSDL 1.1 portTypes (one-to-many relationship). At the instance level, a stateful resource can be associated with one or more Web services. This allows different Web services interfaces to categorize and subset messages that act upon the stateful resource and also allows multiple network endpoints to process messages for the WS-Resource.

3.4.3 WSRF Generic Operations

The WSRF specification provides a standard set of *Message Exchange Patterns* (MEPs) to manipulate the resource property elements that are exposed to the resource property document such as retrieving or modifying their values and even completely removing them from the resource property document. This message exchanges are defined in the *WS-ResourceProperties* specification.

Any WSDL portType that has an associated WS-Resource property document declared by the *wsrp:ResourceProperties* attribute, should include these message exchanges as WSDL operations. The only mandatory operation is *GetResourceProperty*, all other operations are optional.

- *GetResourceProperty*: retrieves a single resource property element value using a simple Web Service request/response message exchange. The WS-Resource is identified from the endpoint reference and the resource property name is identified by the qualified name (QName) of its GED and these identifiers are included in the request message. The response message to such request will contain zero or more values, depending on the cardinality specified in the resource properties document schema.
- *GetMultipleResourceProperties*: retrieves the values of multiple resource property elements with a single request/response message.
- *GetResourcePropertyDocument*: retrieves the value of the whole resource property document. The difference with the *GetMultipleResourceProperties* is that the former retrieve the values of all resource property elements while the latter retrieves the values of those resource properties defined in the request message. However, the *GetMultipleResourceProperties* can be used to retrieve the whole document.
- *PutResourcePropertyDocument*: allows the client to completely replace the values of all properties of a resource properties document.
- *SetResourceProperties*: The WSRF specification supports dynamic insertion and deletion or resource properties at runtime. It defines the standard Message Exchange Pattern (MEP) to manipulate the values of multiple resource property elements in the form of the *SetResourceProperties* operation. This operation has:
    - Insert (*InsertResourceProperties*): to insert a new resource property in the resource property document.
    - Update (*UpdateResourceProperties*): to update the value of an existing resource property.
    - Delete (*DeleteResourceProperties*): to delete an existing resource property from the resource property document.
- *QueryResourceProperties*: allows the execution of an arbitrary query expression such as an XPath 1.0 expression. This operation is the *QueryResourceProperties*.

Finally, WS-ResourceProperties optionally uses WS-Notification (WS-BaseNotification, WSTopics) to allow clients to request notification of inserts, updates and deletions made to the values of one or more resource properties.

3.5 Summary

This chapter provides readers with the basic concepts governing Service Oriented Architecture and the world of Web service technologies. A small introduction has been given to specific Web services standards such as WSDL, SOAP and BPEL describing the basic elements combined in those. These standards will be the basis for this work proposed sequencing engine implementation. In the following section the first phase of the design will be introduced, concerning the representation of Learning Objects with Web services.

Chapter 4:
Learning Objects as Web Services

As described in the overview of the SCORM Sequencing and Navigation specification, for each content organization, which resides in a content package, an Activity Tree is constructed where each node keeps information regarding the *Tracking Model*. In addition, another model is introduced in the SCORM Run-Rime Environment specification, the *Communication Model*. The tracking model and the communication model keep records for the learner's interactions with the content of the package. Furthermore, the IMS SS defines the *Sequencing Definition Model* for defining sequencing rules which apply to the nodes of the Activity Tree in order to define different learning strategies in the Activity Tree. These rules restrict the selection and delivery order of learning content. Another data model presented by IMS SS is the '*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.

In this thesis, these four definition models are combined together in one representation, the stateful resource, as described in the WS-Resource Framework. To recall, the stateful resource keeps the state of a Web Service.

In the following sections, the WS-Resource approach to the architecture of the system is introduced. The resource property elements of the stateful resource are described as well as the system components which have the responsibility to instantiate and manage those stateful resources.

4.1 Implementing Web service wrappers of Learning Objects

The first phase of this thesis is to define Learning Objects characteristics and functionality with Web services. Web services are stateless services while Learning Objects are stateful entities. In order to represent the latter, the WS-Resource Framework is utilized. As described in Chapter 3, this framework provides the means to keep state between Web service's interactions and this state is represented in an XML Document. Each Learning Object which exists in a SCORM-conformant Content Package will be represented with the WS-Resource Framework constructs at run-time. This representation will be described latter in the following sections.

The components of the system that deal with creating the WS-Resource representation and manipulation of learning content are depicted in figure 15. 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.

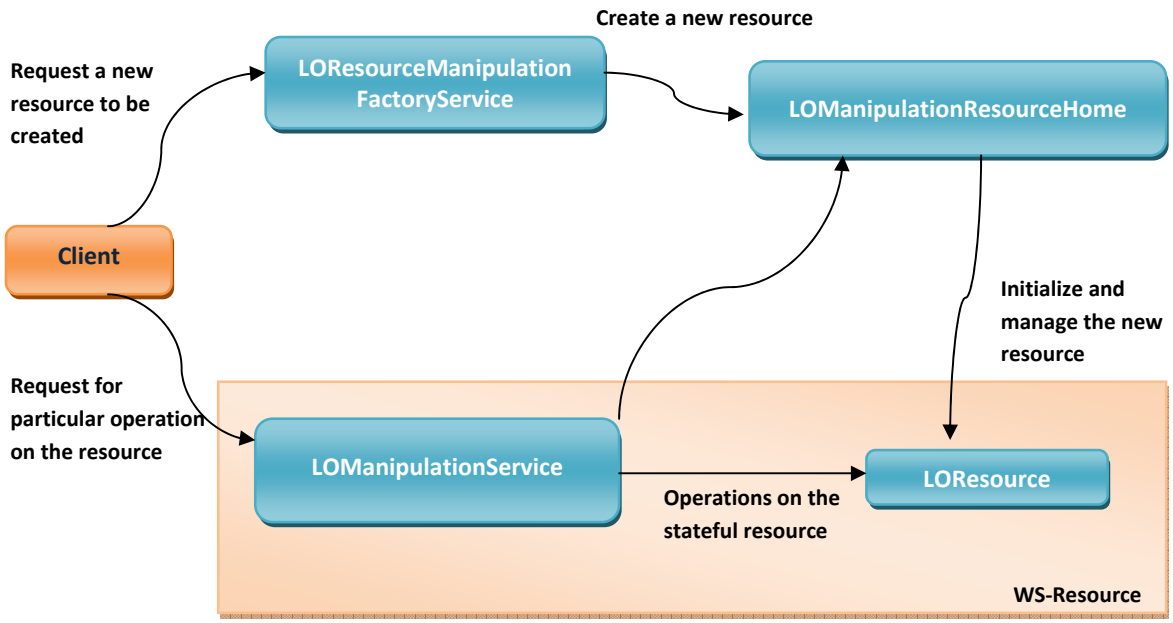


Figure 15: Components dealing with WS-Resources

The following sections describe the components of the WS-Resource design and implementation phase of this thesis. 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 edition encapsulates the final specifications of the WS-Resource framework.

4.1.1 LOManipulationFactoryService

The LOManipulationFactoryService is the factory Web service of the WSRF implementation and part of its WSDL file is depicted in listing 3 . Its responsibility is first to create a resource based on the URL of a package, a learner identifier and a learner name, secondly to assign a unique identifier to this resource and last to return an endpoint reference of the WS-Resource to the client. The learner id and learner name are mandatory because the WS-Resource is created dynamically at run-time and the prior are used for the identification of the WS-Resource that corresponds to the user. This means that WS-Resources corresponding to learning contents of a SCORM package do not exist prior the invocation but are created when the client asks for this package.

```
<message name="CreateResourceRequest">
     <part name="request" element="tns:createResource"/>
</message>
<message name="CreateResourceResponse">
     <part name="response" element="tns:createResourceResponse"/>
</message>
 <portType name="FactoryLOManipulationPortType">
     <operation name="createResource">
         <input message="tns:CreateResourceRequest"/>
         <output message="tns:CreateResourceResponse"/>
     </operation>
 </portType>
```

Listing 3: Part of LOManipulationFactory's service WSDL

Learner name and identifier are used in BPEL correlation sets for identifying the process instance of the sequencing engine assigned to the learner (see Chapter 5). This web service contains only one operation for the creation of the WS-Resource.

The following listing illustrates a SOAP request, with WS-Addressing headers, to the *LOManipulationFactoryService* to create a new WS-Resource given the URL package, learner name, and learner id.

```
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
 <soapenv:Header>
  <wsa:MessageID>uuid:738bf4b0-fb8f-11dd-ac0f-8160bf5057e8</wsa:MessageID>
  <wsa:To>http://localhost:8080/wsrf/services/LOManipulationFactoryService</wsa:To>

<wsa:Action>http://wsrf.master.karmazi/lomanipulation/FactoryLOManipulationPortType/cr
eateResourceRequest</wsa:Action>
  <From xmlns="http://www.w3.org/2005/08/addressing">
   <Address>http://www.w3.org/2005/08/addressing/anonymous</Address>
  </From>
 </soapenv:Header>
 <soapenv:Body>
  <createResource xmlns="http://wsrf.master.karmazi/lomanipulation">
   <lourl
xmlns="">jar:file:/C:/ADL/SCORM_2004_3rd_Ed_Sample_RTE_V1.0.2/Sample_RTE/apache-
tomcat/webapps/adl//packageJarUploads/4EFD48CA807642C0108390B5753D5AEB/LMSTestPackage_
RU-08b.zip!/</lourl>
   <learner_id xmlns="">admin</learner_id>
   <learner_name xmlns="">Admin, Joe</learner_name>
  </createResource>
 </soapenv:Body>
</soapenv:Envelope>
```

Listing 4: SOAP message request to the LOManipulationFactoryService

LOManipulationFactoryService ‘s SOAP response message contains the endpoint reference of the newly created WS-Resource. This endpoint reference includes a *<wsa:Address>* element to return the Web service part of the WS-Resource and a *<wsa:ReferenceParameters>* element to deliver the stateful resource identifier that will be used latter by the *LOManipulationResourceHome* to identify the resource. An example of such a SOAP response message is listed below:

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wsa="http://www.w3.org/2005/08/addressing">

<soapenv:Header>
<wsa:MessageID>
        uuid:739cbd90-fb8f-11dd-9a91-a929162d497b
</wsa:MessageID>
<wsa:To>
        http://www.w3.org/2005/08/addressing/anonymous
</wsa:To>
<wsa:Action>http://wsrf.master.karmazi/lomanipulation/FactoryLOManipulationPortType/cr
eateResourceResponse</wsa:Action>
<From xmlns="http://www.w3.org/2005/08/addressing">
        <Address>
                http://localhost:8080/wsrf/services/LOManipulationFactoryService
        </Address>
        </From>
<wsa:RelatesTo RelationshipType="http://www.w3.org/2005/08/addressing/reply">
        uuid:739a4c90-fb8f-11dd-ac0f-8160bf5057e8
        </wsa:RelatesTo>
</soapenv:Header>

 <soapenv:Body>
        <createResourceResponse xmlns="http://wsrf.master.karmazi/lomanipulation">
          <wsa:EndpointReference>
            <wsa:Address>
                http://192.168.1.3:8080/wsrf/services/LOManipulationService
            </wsa:Address>
            <wsa:ReferenceParameters>
                <ns1:LOKey xmlns:ns1="http://lomanipulation.wsrf.master.karmazi/lo">
                        13063581
                </ns1:LOKey>
            </wsa:ReferenceParameters>
          </wsa:EndpointReference>
        </createResourceResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Stateful Resource Identifier

Listing 5: LOmanipulationFactoryService's SOAP response message

4.1.2 LOManipulationResourceHome

LOManipulationResourceHome is a typical Java class and it is in charge for managing the resources. For example, the stateless service will use the resource home to retrieve the stateful resource. The resource home of this thesis 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 it to find a resource with a given key.

LOManipulationResourceHome (extends ResourceHomeImpl)
+ create(url: String, learner_id: String, learner_name: String)

Figure 16: The LOManipulationResourceHome class

4.1.3 LOManipulationService

LOManipulationService is the stateless front end. The client only interacts with this Web service to query the stateful resource. This Web service and the class "*LOResource*" (see Section 4.1.4), constitute our WS-Resource implementation of Learning Objects contained in a SCORM conformant zip package. The portType part of the associated WSDL file is shown in listing 6.

```
<portType name="LOManipulationPortType" wsdlpp:extends="wsrf-rpw:GetResourceProperty
                   wsrf-rpw:GetMultipleResourceProperties
                   wsrf-rpw:QueryResourceProperties        1
                   wsrf-rpw:SetResourceProperties
                   wsrf-rlw:ImmediateResourceTermination"
                   wsrf-rp:ResourceProperties="tns:LOResourcePropertiesSet">  2

                                                  3
   <operation name="getLearningObjectRP">
            <input  name="GetLearningObjectRPRequest"  message="tns:GetLearningObjectRPRequest" />
            <output name="GetLearningObjectRPResponse" message="tns:GetLearningObjectRPResponse" />
   </operation>

   <operation name="setActivityStateInformation">
            <input  name="ActivityStateInformationRequestMessage"
                    message="tns:ActivityStateInformationRequestMessage" />
            <output name="EmptyResponse" message="tns:EmptyResponse" />
   </operation>

   <operation name="setActivityProgressInformation">
            <input  name="ActivityProgressnformationRequestMessage"
                    message="tns:ActivityProgressInformationRequestMessage" />
            <output name="EmptyResponse" message="tns:EmptyResponse" />
   </operation>

   <operation name="setGlobalObjectiveProgressInformation">
            <input  name="GlobalObjectiveMessage"  message="tns:GlobalObjectiveMessage" />
            <output name="EmptyResponse" message="tns:EmptyResponse" />
   </operation>

   <operation name="newAttemptOnActivity">
            <input  name="NewAttemptRequest"  message="tns:newAttemptRequest" />
            <output name="NewAttemptResponse" message="tns:newAttemptResponse" />
   </operation>

   <operation name="setCurrentAttemptProgressInformation">
            <input  name="SetCurrentAttemptInfoProgressRequest"
                    message="tns:SetCurrentAttemptInfoProgressRequest" />
            <output name="EmptyResponse" message="tns:EmptyResponse" />

   </operation>
        <operation name="setObjectiveProgressInformation">
            <input  name="setObjectiveProgressInformationRequest"
                    message="tns:setObjectiveProgressInformationRequest" />
            <output name="EmptyResponse" message="tns:EmptyResponse" />
   </operation>

   <operation name="setGlobalObjectiveProgressInformationForSatisfiedStatus">
            <input  name="setGlobalSatisfiedStatusRequest"
                    message="tns:setGlobalSatisfiedStatusRequest" />
            <output name="EmptyResponse" message="tns:EmptyResponse" />
   </operation>

   <operation name="setGlobalObjectiveProgressInformationForNormalizedMeasure">
            <input  name="SetGlobalMeasureStatusRequest"
                    message="tns:SetGlobalMeasureStatusRequest" />
            <output name="EmptyResponse" message="tns:EmptyResponse" />
   </operation>
```

```
        <operation name="initialize">                                                    4
            <input name="initializeRequest" message="tns:initializeRequest" />
            <output name="initializeResponse" message="tns:initializeResponse" />
        </operation>


        <operation name="terminate">
            <input name="terminateRequest" message="tns:terminateRequest" />
            <output name="terminateResponse" message="tns:terminateResponse" />
        </operation>

        <operation name="getValue">
            <input name="getValueRequest" message="tns:getValueRequest" />
            <output name="getValueResponse" message="tns:getValueResponse" />
        </operation>

        <operation name="setValue">
            <input name="setValueRequest" message="tns:setValueRequest" />
            <output name="setValueResponse" message="tns:setValueResponse" />
        </operation>

        <operation name="commit">
            <input name="commitRequest" message="tns:commitRequest" />
            <output name="commitResponse" message="tns:commitResponse" />
        </operation>

        <operation name="getLastError">
            <input name="lastErrorRequest" message="tns:lastErrorRequest" />
            <output name="lastErrorResponse" message="tns:lastErrorResponse" />
        </operation>

        <operation name="getErrorString">
            <input name="errorStringRequest" message="tns:errorStringRequest" />
            <output name="errorStringResponse" message="tns:errorStringResponse" />
        </operation>

        <operation name="getDiagnostic">
            <input name="diagnosticRequest" message="tns:diagnosticRequest" />
            <output name="diagnosticResponse" message="tns:diagnosticResponse" />
        </operation>

         <operation name="getCMIException">
            <input name="cmiExceptionRequest" message="tns:cmiExceptionRequest" />
            <output name="cmiExceptionResponse" message="tns:cmiExceptionResponse" />
        </operation>
    </portType>
```

Listing 6: PortType of LOManipulationService service WSDL

The operations provided by the LOManipulationService can be separated into three groups. The first group contains the WSRF generic operations including "*GetResourceProperty*", for retrieving a single property element, "*GetMultipleResourceProperties*", for retrieving multiple resource properties, "*SetResourceProperties*", for setting new property elements in the resource document and "*ImmediateResourceProperty*" as described in the WS-Resource Lifetime, for the destruction of resources. The above operations are implicitly provided by the service as they are associated with the service's portType through the use of "*wsdlpp:extends*", (Number 1 in the above listing) . The second group of arguments contains the operations from "*getLearningObjectRP*" to "*setGlobalObjectiveProgressInformationForNormalized-Measure*" and are used to modify certain property elements in the resource property document. The third group contains the rest operations that are listed under the comment 4 in the Listing 6 which implement the API of the SCORM Run-time Environment.

Finally, the resource property set for the learning objects is attached to the portType with the use of "*wsrf-rp: ResourceProperties*" as described in the WS-Resource Framework.

4.1.4 LOResource

LOResource is a Java class that implements the stateful resource of this thesis WS-Resource approach.

LOResource (implements Resource, ResourceProperties, ResourceIdentifier)
+ initialize(String url, String learner_name, String learner_id)
+ setGlobalObjectiveProgressInformation(globals: String[])
+ setActivityProgress(ap: ActivityProgressInformationType)
+ setActivityState(as: ActivityStateInformationType)
+ setObjectiveProgressInformation(opit: ObjectiveProgressInformationType)
+ newAttemptOnActivity(orgEPR: EndpointReferenceType)
+ setCurrentAttemptProgressInformation(AttemptProgressInformationType atpit)
+ initializateCMI(param: String)
+ terminateCMI(param: String)
+ getValueFromCMI(param: String)
+ setValueToCMI(param1: String, param2: String)
+ getLastError(void)
+ getErrorString(void)
+ getDiagnostic(void)
+ getCMIException(void)

Figure 17: The LOResource class

This class implements the interfaces Resource, ResourceProperies and ResourceIdentifies 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.

4.2 The WS-Resource Implementation

So far, we have discussed the WS-Resource implementation of the Learning Objects from the architecture point of view by describing the various components of the system that are necessary for creating and

manipulating the WS-Resources. In this section, we are presenting the stateful resource as a Learning Object by describing the components that constitute the resource property document.

4.2.1 LO-Resource Properties Document

As discussed in Chapter 3, 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. The following listing contains the definition of the LO-Resource resource property document.

```
<element name="LOResourcePropertiesSet">
   <complexType>
      <sequence>
         <element ref="myTypes:mylearningObject" minOccurs="0" maxOccurs="1"/>
         <element ref="myTypes:node" minOccurs="1" maxOccurs="1"/>
         <element ref="myTypes:resource" minOccurs="0" 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>
```

Listing 7: LO-Resource Properties Document Definition

The resource properties elements of the LO-Resource are:

- *myLearningObject*: contains information regarding the learner name, the learner id, packageURL for which this LO-Resource has been instantiated.
- *node*: is defined as an abstract type and contains either the information of <*item*> or <*organization*> as found in the *imsmanifest.xml*. These two elements are defined in the "*imscp_v1p1.xsd*" schema file published by the IMS.
- *resource*: contains the <*resource*> XML element as found in the *imsmanifest.xml* as defined in the "*imscp_v1p1.xsd*" schema file published by the IMS.
- *manifestBase*: is a string representation and contains the value of the <*manifest*> *"xml:base*" attribute combined with the <resources> "xml:base" attribute.
- *imsss:sequencing*: contains the XML sequencing element as found in the imsmanifest.xml for the associated <*item*>. The XML definition of this element is contained in the "*imsss_v1p0*.xsd" published by the IMS.
- *scormNavExt*: contains the XML SCORM navigation extension as found in the imsmanifest.xml for the associated <*item*>. The XML Schema definition is contained in the *adlnav_v1p3.xsd* file published by the ADL.
- *anyExtension*: contains any XML extensions for latter uses. This element is of type AnyContentType and for the purposes of this master stores the ADL SCORM extensions (*adlcp_v1p3.xsd*)
- *ObjectiveProgressSet*: contains the XML element of a set of *Objective Progress Information* as defined in the Tracking Model according to the attempt. The type of this element is described in Section 2.4.1.3.1.

- *ActivityProgressSet*: contains the XML element of the *Activity Progress Set* of the Tracking Model as described in Section 2.4.1.3.1.
- *AttemptProgressSet*: contains the XML element of the *Attempt Progress Set* of the Tracking Model as described in Section 2.4.1.3.1.
- *ActivityStateInformation*: contains the *Activity State Information* as defined in the Tracking Model and its complex type is described in Section 2.4.1.3.2
- *CMIGlobalState*: contains the XML element of the *Communication model (CMI)* defined for the purposes of the master.

This XML representation of the LO-Resource is attached to the portType of the LOManipulationService as described in Section 4.1.3.

4.2.2 Sequencing Data Models as Resource Property Elements

An XML Schema type is defined for each set of information in the Tracking Model, Activity State Model and Communication Data Model. The following listings contain the definition of these types which are included in the property document of the LO-Resource.

4.2.2.1 Objective Progress Information Set

As mentioned in Section 2.4.1.3.1, an activity may be associated with zero or more learning objectives and Tracking Model is used to keep records of the learner's interactions relating to these objectives (*Objective Progress Information*). For this reason, a set is defined containing all the Objective Progress Information for each objective of an activity per learner attempt. The XML representation of such a set includes the objectiveID that the progress information is associated with, the identifier of a shared objective, if any, the '*Objective Progress Status*', '*Objective Satisfied Status*', '*Objective Measure Status*', and '*Objective Normalized Measure*' as described in the Tracking Model and information for the global objective, if there is an Objective Map defined for an activity.

```
<xs:complexType name="ObjectiveProgressInformationType">
   <xs:attribute name="ObjectiveID" type="xs:string" use="requires" />
   <xs:attribute name="sharedObjective" type="xs:boolean" use="requires" />
   <xs:sequence>
       <xs:element name="ObjectiveProgressStatus" type="xs:boolean" default="false" />
       <xs:element name="ObjectiveSatisfiedStatus" type="xs:boolean" default="false" />
       <xs:element name="ObjectiveMeasureStatus" type="xs:boolean" default="false" />
       <xs:element name="ObjectiveNormalizedMeasure" type="measureType" default="0.0" />
       <xs:element name="ReadSatisfiedStatusFrom" type="xs:string" minOccurs="0"
              maxOccurs="1" nillable="true" />
       <xs:element name="ReadNormalizedMeasureFrom" type="xs:string" minOccurs="0"
              maxOccurs="1" nillable="true" />
        <xs:element name="WriteSatisfiedStatusTo" type="xs:string" minOccurs="0"
              maxOccurs="unbounded"  nillable="true" />
       <xs:element name="WriteNormalizedMeasureTo" type="xs:string" minOccurs="0"
              maxOccurs="unbounded" nillable="true" />
   </xs:sequence>
</xs:complexType>
```

Listing 8: XML Schema type definition for Objective Progress Information of an objective

```
<xs:element name="ObjectiveProgressInformation" type="ObjectiveProgressInformationType" />

<xs:complexType name="ObjectiveProgressSet">
    <xs:attribute name="noAttempt" type="xs:int" use="requires" />
    <xs:sequence>
       <xs:element ref="ObjectiveProgressInformation" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
```

Listing 9: Set of Objective Progress Information for an activity

The above listing describes the set of Objective Progress Information for each objective recorded by the activity. For each attempt on the activity, a different set is recorded. For this reason, each objective set contains the attempt number.

4.2.2.2 Activity Progress Status

The *Activity Progress Status* describes a learner's progress on an activity. This information set should be instantiated for each activity in the Activity Tree. The elements and data types are described in Section 2.4.1.3.1. The XML Schema type definition of this set is the following:

```
<xs:complexType name="ActivityProgressInformationType">
   <xs:sequence>
       <xs:element name="ActivityProgressStatus" type="xs:boolean" default="false" />
       <xs:element name="ActivityAbsoluteDuration" type="xs:duration" default="0.0" />
       <xs:element name="ActivityExperiencedDuration" type="xs:duration" default="0.0" />
       <xs:element name="ActivityAttemptCount" type="xs:nonNegativeInteger" default="0" />
   </xs:sequence>
</xs:complexType>
```

Listing 10: XML Schema type definition of Activity Progress Information of an activity

4.2.2.3 Attempt Progress Status

The *Attempt Progress Information* describes a learner's progress for a unique attempt on an activity. This information should be instantiated for each new attempt on the activity. The complex type defined for this set is the following:

```
<xs:complexType name="AttemptProgressInformationType">
   <xs:sequence>
       <xs:element name="AttemptProgressStatus" type="xs:boolean" default="false" />
       <xs:element name="AttemptCompletionAmount" type="imsss:percentType" default="0.0" />
       <xs:element name="AttemptCompletionStatus" type="xs:boolean" default="false" />
       <xs:element name="AttemptAbsoluteDuration" type="xs:duration" default="0.0" />
       <xs:element name="AttemptExperiencedDuration" type="xs:duration" default="0.0" />
   </xs:sequence>
</xs:complexType>
```

Listing 11: XML Schema type definition of Attempt Progress Information of an activity

4.2.2.4 Activity State Model

Another model defined in the IMS SS is the *Activity State Model*. This model records the state or status of the learner's interaction with activities. These attributes are used to control the overall sequencing process. The model has been described in Section 2.4.1.3.2.

4.2.2.4.1 Activity State Information

Activity State Information describes a learner's state or status for an activity. This information should be instantiated for each activity in the Activity Tree. The complex type defined in this thesis is the following:

```
<xs:complexType name="ActivityStateInformationType">
   <xs:sequence>
       <xs:element name="ActivityIsActive" type="xs:boolean" default="false" />
       <xs:element name="ActivityIsSuspended" type="xs:boolean" default="false" />
       <xs:element name="AvailableChildrenIndices" type="indices" minOccurs="0" >
        </xs:element>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="indices">
    <xs:sequence>
       <xs:element name="i" type="xs:int" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

Listing 12: XML Schema type definition of Activity State Information of an activity

4.2.2.4.2 Global State Information

Global State Information describes the state of the Activity Tree as a whole and needs to be instantiated only once. This data model does not apply to each activity so this will not be defined as a resource property of a LO-Resource. Global State Information will be utilized and maintained by the BPEL process. In BPEL, global variables can be used to keep the state of the process. *Current Activity* and *Suspended Activity* elements of the Global State Information are defined as variables in the BPEL process with the type of an endpoint reference because these two elements indicate activities in the Activity Tree. In our implementation activities are identified by unique endpoint references.

4.2.3 SCORM Run-time Environment Data Model

SCORM Run-time Environment Data model contains a set of data model elements that can be tracked by the SCO with an LMS during the run-time communication of the SCO. For the purposes of this thesis, only a set of elements defined in this data model are implemented. The communication data model is based on an IEEE standard: IEEE 1484.11.1-2004. The IEEE 1484.11.3-2005 standard specifies an XML schema for this data model. Appendix B presents a graphical overview of the top level element that serve as a container and is referenced with "*ref='cmi:cocd'*".

The following XSD listing contains the complex type definition of the CMI model.

```
<xs:complexType name="CMIGlobalStateType">
      <xs:sequence>
          <xs:element name="initializedState" type="xs:boolean" minOccurs="1" />
          <xs:element name="terminatedState" type="xs:boolean" minOccurs="1" />
          <xs:element name="error" type="CMIException" minOccurs="1" />
          <xs:element ref="cmi:cocd" minOccurs="0" maxOccurs="1" nillable="true" />
      </xs:sequence>
</xs:complexType>
```

Listing 13: XML Schema type definition of the Communication Data Model

The XML representation of the Communication Data Model contains two boolean values "*initializedState*" and "*terminatedState*" indicating if the communication between the SCO and LMS has started and if this communication has been terminated by the SCO. It also contains an element indicating

if an exception has occurred. The "*cmi:cocd*" element is the container element of all CMI elements defined in the IEEE 1484.11.3-2005 schema.

The error type is the following:

```
<xs:complexType name="CMIException">
   <xs:sequence>
        <xs:element name="errorCode" type="xs:int" minOccurs="1" />
        <xs:element name="errorDescription" type="xs:string" minOccurs="1" />
        <xs:element name="diagnosticErrorDescription" type="xs:string" minOccurs="0"
                    nillable="true" />
   </xs:sequence>
</xs:complexType>
```

Listing 14: XML Schema definition for error in the CMI

4.4 Summary

This chapter has described the first phase of the thesis's system implementation and particularly the part of the system that deals with the representation of Learning Objects with the use of Web services. The WS-Resource framework is utilized for this representation. A resource property document is defined for containing parts of the *imsmanifest.xml* file found in a SCORM conformant Content Package and XML elements corresponding to Tracking Model, Activity State Model and SCORM Communication Data Model. For these three data models described in SCORM Sequencing and Navigation Model and in the IMS Simple Sequencing specification, XML schema types and elements have been defined in order to represent resource property elements. Finally, the WS-Resource manipulation part of the system is described and its main components where introduced.

Chapter 5:

The BPEL Sequencing Engine based on the SCORM 2004 SN Specification

This chapter is dedicated to the design and implementation of a Sequencing Engine as a BPEL process that conforms to the Sequencing and Navigation specification of the SCORM 2004 3rd Edition. This specification provides the guidelines for sequencing learning objects to create learning paths for learning experience. It also defines the behaviors that a sequencing engine should exhibit and defines the grammar and elements to prescribe learning paths. The sequencing behaviors described in the specification are represented by a set of Web Services which further are orchestrated into a BPEL process to deliver a plug and play sequencing engine.

The chapter introduces readers to the main purpose and functionality that the BPEL sequencing engine exhibits and shows how this process can fit into existing learning management systems. It provides insights towards a Web service based e-Learning system. The following sections describe the functionality of the BPEL process and the services that have been implemented for it. For each service, the corresponding IMS processes and subprocesses are described to provide readers information regarding the existing concepts defined in specifications and the implementation details. The last section shows how all the Web services designed and implemented for the purposes of this thesis are orchestrated into a process based on the BPEL standard.

5.1 An Overview of the Architecture

The proposed architecture has been designed with the objective to provide an engine that encapsulates the Overall Sequencing Process as described in the Sequencing and Navigation Specification in the SCORM 2004 3rd Edition and incorporating this functionality into existing learning management systems. For this purposes, Web services technologies have been utilized to provide the functionality of the engine in terms of Web services. The processes and subprocesses described in the specifications and used in the sequencing engine have been grouped into sets and Web services have been designed and implemented for each of these sets. Finally, these Web services have been orchestrated according to BPEL standard to provide the Sequencing Engine as a process. The implemented engine has then been incorporated into the SCORM Run-Time Environment implemented which is provided freely by ADL. This incorporation was realized for testing purposes and to provide real time data to the BPEL-based engine such as used id and navigation control data, to exhibit the BPEL-based engine functionality

This architecture serves two purposes; to provide a Web service representation of Learning Objects and to define a BPEL process for the sequencing implementation.

The main architecture comprises of two important parts, the part that deals with representation and the instantiation of Web services as wrappers of Learning Objects, discussed in Chapter 4, and the BPEL process that exhibits the behavior of the Overall Sequencing Process as described in the SCORM SN

book. A generalized architecture of the recommended system, including the BPEL engine as external component, a LMS, remote servers and repositories, and the client is depicted in the following figure.

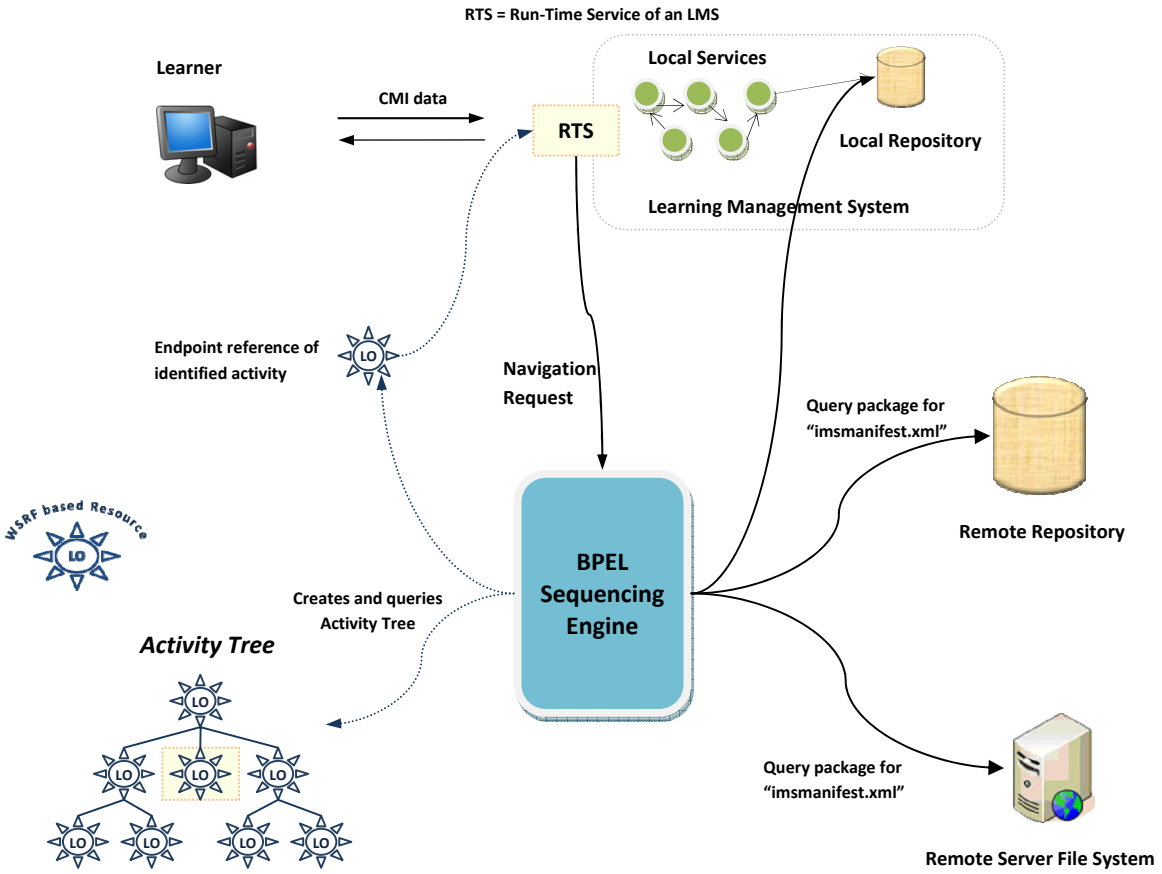


Figure 18: Generalized Proposed Architecture

The engine acts as a remote standalone component service of existing learning management systems 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 (see Chapter 2). Activity tree is created at run-time and is consisted of WS-Resources wrappers of learning content existing in the package (see Chapter 4). It is worthy to recall that 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 is attached to the items in this manifest file to define an instructional strategy for sequencing purposes. The LOM metadata and the XML infoset of the Sequencing Definition Model is attached as resource property elements in the WS-Resources.

The BPEL Sequencing engine waits for navigation requests that result in starting a new sequencing session (create Activity Tree and possible 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 book and it's representation has been described in Chapter 4.

In the next sections, the Web services that compose the sequencing engine and the process itself are discussed.

5.2 Components of the BPEL sequencing engine

The BPEL process representation of the Overall Sequencing Process orchestrates Web services that correspond to the sequencing behaviors as described in the IMS SS specification.

The Overall Sequencing Process, as defined in the IMS SS, provides the overarching control process for the LMS's sequencing implementation and describes how the various sequencing behaviors 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. The exit point of the process is either the identification of the "next" activity to deliver, which could be nothing or an exception. SCORM does not define behavior for the cases where no activity is identified for delivery or if some processes produced an exception. The Overall Sequencing Process exhibits the behavior of a sequencing loop when a sequencing session starts until it ends.

Based on SCORM SN, the Overall Sequencing Process ends in the following situations:

1) A "*Suspend All*" navigation request is issued and the attempt on the Current Activity is terminated abnormally. The activity is not considered as complete as well as the attempts. The BPEL process is responsible to record the current state of the Activity Tree. The attempts may be resumed at some time in the future, meaning that there is no new attempt.
2) An "*Abandon All*" navigation request is issued resulting in terminating attempts on the Current Activity and all of its ancestors. Attempt on the current activity will not be resumed at some time in the future and there is no rollback of any tracking information.

The set of web services invoked by the BPEL process is analyzed in the following sub-sections. Each service implements a set of processes and subprocesses as described in the IMS Simple Sequencing specification. Each of the latter is described in the following sections. An abstract representation of the BPEL process and the service that orchestrates are depicted in Figure 19. This figure depicts the sequence of the web services and how they fit with concepts and implementation details described in Chapter 4.

The services are highly depend on the endpoint references of three specific activities, the root of the Activity Tree, the *Current Activity,* where the most sequencing decisions are made from and the *Suspended Activity* (see section 2.4.1.3.2).

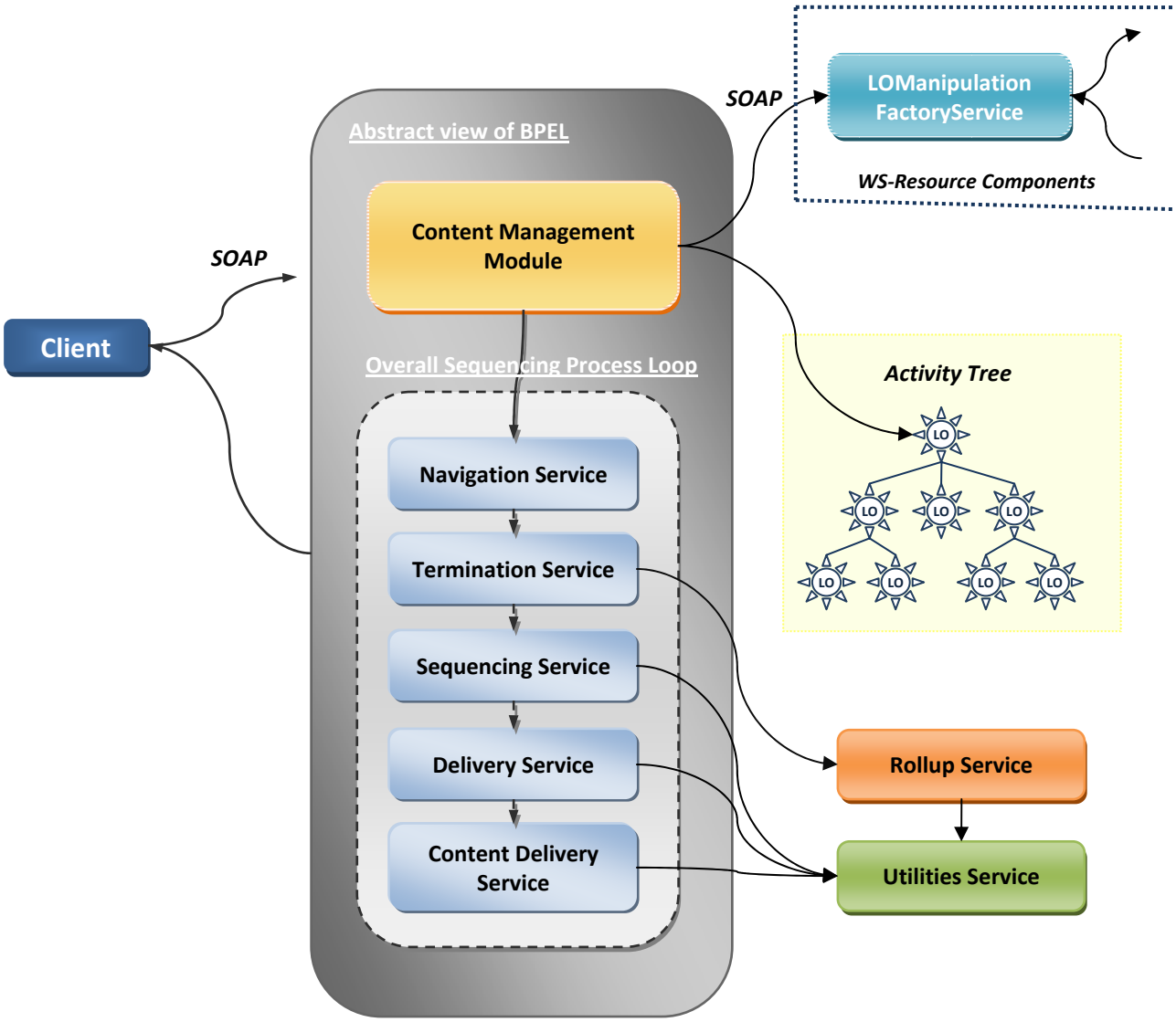


Figure 19: Abstract View of BPEL process and Architecture Component

5.2.1 NavigationProcessService

The *NavigationProcessService* implements the *Navigation Behavior* which is the entry point in the Overall Sequencing Process. This behavior tests for the validity of the issued navigation request and it is responsible for calling either the *Termination Behavior* or *Sequencing Behavior* 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 issues by UI controls and indicate system's or user's intention to navigate through content in some manner. This navigation requests are vocabulary specific. The following Table borrowed from the SCORM SN specification summarizes the possible navigation request's values and the actions to be taken by the Navigation Behavior:

Table 10: SCORM Navigation Requests [36]

Navigation Request	Action
Start	If *Current Activity* is undefined, issue a *Start* sequencing request
Resume All	If *Current Activity* is undefined and the *Suspended Activity* is defined, issue a *Resume*

	All sequencing request
Continue	If *Activity is Active* for the *Current Activity* is true. Issue an *Exit* termination request and a *Continue* sequencing request
Previous	If *Activity is Active* for the *Current Activity* is true. Issue an *Exit* termination request and a *Previous* sequencing request
Forward	Not specified in SCORM 2004 3rd edition
Backward	Not specified in SCORM 2004 3rd edition
Choice	If *Activity is Active* for the *Current Activity* is true. Issue an *Exit* termination request and a *Choice* sequencing request. The request is accompanied by the identification of the target activity.
Exit	Issue an *Exit* termination request and an *Exit* sequencing request
Exit All	Issue an *Exit All* termination request and an *Exit* sequencing request
Suspend All	Issue a Suspend All termination request and an *Exit* sequencing request
Abandon	Issue an *Abandon* termination request and an *Exit* sequencing request.
Abandon All	Issue an *Abandon All* termination request and an *Exit* sequencing request

Navigation Service is implemented as a document-style web service.

```
  <wsdl:message name="NavigationRequest">
    <wsdl:part name="NavigationRequest" element="tns:navigationInput" />
  </wsdl:message>
  <wsdl:message name="NavigationResponse">
    <wsdl:part name="NavigationResponse" element="tns:navigationOutput" />
  </wsdl:message>
  <wsdl:message name="NavigationException">
    <wsdl:part name="NavigationException" element="tns:navigationFault" />
  </wsdl:message>

  <wsdl:portType name="NavigationProcessPortType">
    <wsdl:operation name="startNavigationProcess">
      <wsdl:input name="NavigationRequest"  message="tns:NavigationRequest"/>
      <wsdl:output name="NavigationResponse" message="tns:NavigationResponse"/>
      <wsdl:fault name="NavigationException" message="tns:NavigationException"/>
    </wsdl:operation>
 </wsdl:portType>
```

Listing 15: NavigationService's messages and portType parts

The above listing contains the message parts of the service and the portType. Only one operation is public and corresponds to the *Navigation Request Process NB.2.1* of the IMS Simple Sequencing specification.

The Navigation Service waits for an input that essentially should contain the navigation request and optionally endpoint references to the organization activity, the current activity, the suspended activity. If the navigation request is a "Choice" request then it is mandatory to specify the target activity's endpoint. The input message of the service is the following:

```
<xs:complexType name="NavigationInput">
   <xs:sequence>
       <xs:element name="navigationRequest" type="xs:string" />
       <xs:element name="targetActivityEPR" type="wsa:EndpointReferenceType"  nillable="true"
                minOccurs="0"/>
       <xs:element name="currentActivityEPR" type="wsa:EndpointReferenceType" nillable="true"
                minOccurs="0" />
       <xs:element name="suspendedActivityEPR" type="wsa:EndpointReferenceType" nillable="true"
                minOccurs="0"/>
       <xs:element name="orgEPR" type="wsa:EndpointReferenceType" minOccurs="0" />
   </xs:sequence>
</xs:complexType>
```

Listing 16: NavigationService's input message

The output of the service contains the termination or sequencing or both requests evaluated against the input navigation request and the Boolean type element that it indicates whether the navigation requests was valid or not.

```
<xs:complexType name="NavigationOutput">
   <xs:sequence>
       <xs:element name="valid" type="xs:boolean" />
       <xs:element name="terminationRequest" type="xs:string" nillable="true" minOccurs="0"/>
       <xs:element name="sequencingRequest" type="xs:string" nillable="true" minOccurs="0"/>
       <xs:element name="targetActivityEPR" type="wsa:EndpointReferenceType" nillable="true" minOccurs="0"/>
       <xs:element name="exception" type="xs:string" nillable="true" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
```

Listing 17: NavigationService's output message

Upon any fault, the NavigationService returns an Axis Fault of the following type:

```
<xs:complexType name="NavigationException">
    <xs:sequence>
        <xs:element name="code" type="xs:string" />
        <xs:element name="description" type="xs:string" nillable="true" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
```

Listing 18: NavigationService's exception

5.2.2 TerminationProcessService

The *TerminationProcessService* implements the *Termination Behavior* as described in the IMS Simple Sequencing specification. This behavior 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.

Also, this service provides implementation for the following subprocesses defined in IMS SS:

- *Sequencing Exit Action Rules Subprocess's (TB.2.1)* responsibility is to find at least one ancestor of the Current Activity that contains *exit sequencing rules*. If an ancestor is found then it becomes the Current Activity and the attempts of its child descendents are terminated.
- The *Sequencing Post Condition Rules Subprocess's (TB.2.2)* responsibility is to apply the *post conditions sequencing rules* if they exist any to the Current Activity and to determine a termination and a sequencing request if possible.

Table 11: SCORM Termination Requests [36]

Termination Request	Action
Exit	The current attempt on the Current Activity is terminated normally; the attempt is over
Exit Parent	The current attempt on the Current's Activity parent is terminated normally; the attempt is over
Exit All	The current attempts on the active activities (from the root to the Current Activity inclusive) are terminated normally. The attempts are over
Suspend All	The current attempts on the active activities (from the root to the Current Activity

	inclusive) are suspended. The attempt on the Current Activity may be resumed.
Abandon	The current attempt on the Current Activity is terminated abnormally and the activity is not complete. The attempt may be not resumed. There is no rollback of any tracking data.
Abandon All	The current attempts on the active activities (from the root to the Current Activity, inclusive) are terminated abnormally and the activities are not complete. Attempts on any abandoned activity may not be resumed. There is no rollback of any tracking data.

TerminationProcessService is implemented as a document-style web service. The following listing contains the message and portType parts of the service. Only one operation is provided which corresponds to the *Termination Request Process TB.2.3* defined in the IMS Simple Sequencing specification. Internally, the service implements the "*Sequencing Exit Action Rules Subprocess – TB.2.1*" and the "*Sequencing Post Condition Rules Subprocess – TB.2.2* " both defined also in the IMS SS.

```
<wsdl:message name="TerminationRequest">
  <wsdl:part name="TerminationRequest" element="tns:terminationInput"/>
</wsdl:message>
<wsdl:message name="TerminationResponse">
  <wsdl:part name="TerminationResponse" element="tns:terminationOutput"/>
</wsdl:message>
 <wsdl:message name="TerminationException">
  <wsdl:part name="TerminationException" element="tns:terminationFault" />
</wsdl:message>

<wsdl:portType name="TerminationProcessServicePortType">
  <wsdl:operation name="terminationRequestProcess">
    <wsdl:input name="TerminationRequest" message="tns:TerminationRequest"/>
    <wsdl:output name="TerminationResponse" message="tns:TerminationResponse"/>
     <wsdl:fault name="TerminationException" message="tns:TerminationException"/>
  </wsdl:operation>
</wsdl:portType>
```

Listing 19: Message and PortType parts of the TerminationProcessService WSDL

The TerminationProcessService is invoked whenever a termination request is issued by the NavigationService. The input message of this service must contain the termination request as well as the endpoint reference of the organization activity which is the root of the Activity Tree. Optionally, if there is a Current Activity, its endpoint reference should be passed to the service's input message. The input message of the service is the following:

```
<xsd:complexType name="TerminationProcessInput">
  <xsd:sequence>
      <xsd:element name="terminationRequest" type="xsd:string" maxOccurs="1" minOccurs="1"
                   nillable="false"/>
      <xsd:element name="orgEPR" type="wsa:EndpointReferenceType" nillable="false"
                   minOccurs="1"/>
      <xsd:element name="currentEPR" type="wsa:EndpointReferenceType" minOccurs="0"
                   nillable="true"/>
  </xsd:sequence>
</xsd:complexType>
```

Listing 20:TerminationProcessService's input message

After service execution, a sequencing request may be returned for invoking the SequencingProcessService (see Section 5.2.3), an identification if the termination request is valid or not and optionally the endpoint references of the Current Activity and the Suspended Activity. The service's output is depicted in the following Listing.

```
<xsd:complexType name="TerminationProcessOutput">
  <xsd:sequence>
      <xsd:element name="valid" type="xsd:boolean" nillable="false" maxOccurs="1"
                   minOccurs="0"/>
      <xsd:element name="sequencingRequest" type="xsd:string" nillable="true"/>
      <xsd:element name="exception" type="xsd:string" nillable="true"/>
      <xsd:element name="currActivityEPR" type="wsa:EndpointReferenceType" nillable="true"/>
      <xsd:element name="suspendenActivityEPR" type="wsa:EndpointReferenceType" minOccurs="0"
                        maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>
```

Listing 21: TerminationProcessService's output message

Upon any fault, the TerminationProcessService returns an Axis Fault of the following type:

```
<xsd:complexType name="TerminationException">
  <xsd:complexContent>
    <xsd:extension base = "ss:SequencingException">
        <xsd:sequence>
            <xsd:element name="currActivityEPR" type="wsa:EndpointReferenceType"
                         nillable="true" minOccurs="0" />
        </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Listing 22: TerminationProcessService's exception

The fault contains the endpoint reference of the Current Activity because during the termination process the current activity may change so it must be propagated to the other services.

5.2.3 SequencingProcessService

The *SequencingProcessService* implements the *Sequencing Behavior* as described in the IMS Simple Sequencing specification. This behavior 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 behavior does not alter the state of the Activity Tree. The valid requests for this service are: *start*, *resume All*, *continue*, *previous*, *choice*, *retry* and *exit*. This sequencing request comes from either *NavigationService* or *TerminationProcessService*. The following listing shows the messages and portType parts of the SequencingProcessService WSDL file. The core operation of the SequencingProcessService is the "*startSequencingRequestProcess*" which invokes all the others to fully exhibit the functionality of the IMS SS Sequencing Behavior. The other operations are invoked according to the sequencing request.

Table 12: Sequencing Subprocess depending one the Sequencing request

Sequencing Request	Sequencing Request subprocess
start	`startSequencingRequestOperation`
resume All	`resumeAllSequencingRequestOperation`
continue	`continueSequencingRequestOperation`
previous	`previousSequencingRequestOperation`
choice	`choiceSequencingRequestOperation`
retry	`retrySequencingRequestOperation`
exit	`exitSequencingRequestOperation`

The *SequencingProcessService* encapsulates the following processes and subprocesses defined in the IMS Simple Sequencing Specification as public or private operations:

- *Flow Tree Traversal Subprocess SB.2.1*: determines the next activity in a preorder traversal of the Activity Tree.
- *Flow Activity Traversal Subprocess SB.2.2*: determines if a single activity should be delivered by checking limit conditions and sequencing rules, optionally traversing the Activity Tree to the next activity.
- *Flow Subprocess SB.2.3*: traverses the activity tree in a specified direction to find the next activity allowed for delivery
- *Choice Activity Traversal Subprocess SB.2.4*: determines if a *Choice* sequencing request is permitted for a single activity by checking limit conditions and sequencing rules
- *Start Sequencing Request Process SB.2.5*: processes a *Start* sequencing request
- *Resume All Sequencing Request Process SB.2.6*: processes a *Resume All* sequencing request
- *Continue Sequencing Request Process SB.2.7:* processes a *Continue* sequencing request
- *Previous Sequencing Request Process SB.2.8*: processes a *Previous* sequencing request
- *Choice Sequencing Request Process SB.2.9*: processes a *Choice* sequencing request
- *Choice Flow Subprocess SB.2.9.1*:
- *choice Flow Tree Traversal Subprocess SB.2.9.2*:
- *Retry Sequencing Request Operation SB.2.10*: processes a *Retry* sequencing request
- *Exit Sequencing Request Operation SB.2.11*: processes an *Exit* sequencing request

```
<wsdl:message name="startSequencingRequest">
  <wsdl:part name="startSequencingRequest" element="tns:navigationInput" />
</wsdl:message>
<wsdl:message name="startSequencingResponse">
  <wsdl:part name="startSequencingResponse" element="tns:navigationOutput" />
</wsdl:message>
<wsdl:message name="flowSubprocessRequest">
  <wsdl:part name="flowSubprocessRequest" element="tns:flowInput" />
</wsdl:message>
<wsdl:message name="flowSubprocessResponse">
  <wsdl:part name="flowSubprocessResponse" element="tns:flowOutput" />
</wsdl:message>
 <wsdl:message name="choiceFlowRequest">
  <wsdl:part name="choiceFlowRequest" element="tns:choiceTraversalInput" />
</wsdl:message>
<wsdl:message name="choiceFlowResponse">
  <wsdl:part name="choiceFlowResponse" element="tns:choiceTraversalOutput" />
</wsdl:message>
<wsdl:message name="WSResourceEndpointType">
  <wsdl:part name="WSResourceEndpointType" type="types:WSResourceEndpointType" />
</wsdl:message>
<wsdl:message name="SequencingProcessException">
  <wsdl:part name="SequencingProcessException" element="tns:sequencingFault" />
</wsdl:message>

<wsdl:portType name="SequencingProcessPortType">
  <wsdl:operation name="startSequencingRequestProcess">
    <wsdl:input name="StartSequencingRequest" message="tns:startSequencingRequest"/>
    <wsdl:output name="StartSequencingResponse" message="tns:startSequencingResponse"/>
    <wsdl:fault name="SequencingProcessException" message="tns:SequencingProcessException"/>
  </wsdl:operation>
  <wsdl:operation name="startSequencingRequestOperation">
    <wsdl:input name="StartSequencingOperationRequest" message="tns:startSequencingRequest"/>
    <wsdl:output name="StartSequencingOperationResponse" message="tns:startSequencingResponse"/>
    <wsdl:fault name="SequencingProcessException" message="tns:SequencingProcessException"/>
  </wsdl:operation>
  <wsdl:operation name="resumeAllSequencingRequestOperation">
    <wsdl:input name="ResumeAllSequencingOperationRequest" message="tns:startSequencingRequest"/>
    <wsdl:output name="ResumeAllSequencingOperationResponse"
                 message="tns:startSequencingResponse"/>
    <wsdl:fault name="SequencingProcessException" message="tns:SequencingProcessException"/>
  </wsdl:operation>
   <wsdl:operation name="continueSequencingRequestOperation">
    <wsdl:input name="ContinueSequencingOperationRequest" message="tns:startSequencingRequest"/>
    <wsdl:output name="ContinueSequencingOperationResponse"
                 message="tns:startSequencingResponse"/>
    <wsdl:fault name="SequencingProcessException" message="tns:SequencingProcessException"/>
  </wsdl:operation>
  <wsdl:operation name="previousSequencingRequestOperation">
    <wsdl:input name="PreviousSequencingOperationRequest" message="tns:startSequencingRequest"/>
    <wsdl:output name="PreviousSequencingOperationResponse"
message="tns:startSequencingResponse"/>
    <wsdl:fault name="SequencingProcessException" message="tns:SequencingProcessException"/>
  </wsdl:operation>
  <wsdl:operation name="choiceSequencingRequestOperation">
    <wsdl:input name="ChoiceSequencingOperationRequest" message="tns:startSequencingRequest"/>
    <wsdl:output name="ChoiceSequencingOperationResponse" message="tns:startSequencingResponse"/>
    <wsdl:fault name="SequencingProcessException" message="tns:SequencingProcessException"/>
  </wsdl:operation>
   <wsdl:operation name="retrySequencingRequestOperation">
    <wsdl:input name="RetrySequencingOperationRequest" message="tns:startSequencingRequest"/>
    <wsdl:output name="RetrySequencingOperationResponse" message="tns:startSequencingResponse"/>
    <wsdl:fault name="SequencingProcessException" message="tns:SequencingProcessException"/>
  </wsdl:operation>
   <wsdl:operation name="exitSequencingRequestOperation">
    <wsdl:input name="ExitSequencingOperationRequest" message="tns:startSequencingRequest"/>
    <wsdl:output name="ExitSequencingOperationResponse" message="tns:startSequencingResponse"/>
    <wsdl:fault name="SequencingProcessException" message="tns:SequencingProcessException"/>
```

```
    </wsdl:operation>
     <wsdl:operation name="flowSubprocess">
      <wsdl:input name="flowOperationRequest" message="tns:flowSubprocessRequest"/>
      <wsdl:output name="flowOperationResponse" message="tns:flowSubprocessResponse"/>
      <wsdl:fault name="SequencingProcessException" message="tns:SequencingProcessException"/>
    </wsdl:operation>
     <wsdl:operation name="choiceFlowSubprocess">
      <wsdl:input name="choiceFlowRequest" message="tns:choiceFlowRequest"/>
      <wsdl:output name="WSResourceEndpointType" message="tns:WSResourceEndpointType"/>
    </wsdl:operation>

  </wsdl:portType>
```

Listing 23: Message and PortType parts of the SequencingProcessService WSDL

The service is expecting a sequencing request coming either from the *NavigationService* or the *TerminationProcessService* and endpoint references for the Current Activity, the organization activity which is the root of the Activity Tree and the Suspended Activity, if any. If the sequencing request is "*choice*", then the endpoint reference of the target activity is mandatory. For each valid sequencing request, the proper operation is invoked.

The input message for the SequencingProcessService is the following:

```
<xs:complexType name="sequencingProcessInput">
    <xs:sequence>
      <xs:element name="sequencingRequest" type="xs:string" minOccurs="1" maxOccurs="1"
                 nillable="true" />
      <xs:element name="activityEPR" type="wsa:EndpointReferenceType"  minOccurs="0"
                 nillable="true" />
      <xs:element name="suspActivityEPR" type="wsa:EndpointReferenceType" minOccurs="0"
                  nillable="true" />
      <xs:element name="orgEPR" type="wsa:EndpointReferenceType" nillable="true"
                  minOccurs="0" />
      <xs:element name="targetActivityEPR" type="wsa:EndpointReferenceType" nillable="true"
                  minOccurs="0" />
    </xs:sequence>
</xs:complexType>
```

Listing 24: SequencingProcessService's input message

The output message of the service is the following:

```
<xs:complexType name="sequencingProcessOutput">
   <xs:sequence>
       <xs:element name="validSeqRequest" type="xs:boolean"></xs:element>
       <xs:element name="deliveryEPR" type="wsa:EndpointReferenceType" minOccurs="0" />

       <xs:element name="endSequencingSession" type="xs:boolean" />
       <xs:element name="exception" type="xs:string" minOccurs="0" />
       <xs:element name="currActivityEPR" type="wsa:EndpointReferenceType" minOccurs="0" />
   </xs:sequence>
 </xs:complexType>
```

Listing 25: SequencingProcessService's output message

The output message of the service contains information regarding if the sequencing request was valid or not, an endpoint reference to an activity, if an activity has been identified, otherwise null, whether sequencing session should end("true") or not ("false") and the endpoint reference of the Current Activity.

Upon exception, the service reports an Axis fault with type:

```
<xs:complexType name="SequencingException">
   <xs:sequence>
      <xs:element name="code" type="xs:string" />
      <xs:element name="description" type="xs:string" nillable="true" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
```

Listing 26: SequencingProcessService's exception

According to SCORM SN book, sequencing requests can be grouped into four categories:

1) The group that begins a sequencing session, including "*Start*", "*ResumeAll*" and "*Choice*".
2) The group that causes the traversal of the Activity Tree to the "next" activity, including "*Continue*", "*Previous*" and "*Choice*" (after the sequencing session has began).
3) The group that repeat the Current Activity, including the "*Retry*".
4) The group that ends the sequencing session containing only the "*Exit*".

5.2.3.1 Flow Subprocess [SB.2.3]

One of the most important operations in the Sequencing Process is the "*flowSubprocess*". This operation is used to traverse the Activity Tree in order to identify the next activity to deliver. This process stops only to leaf activities, which are the only nodes in the Activity Tree that contains actual content.

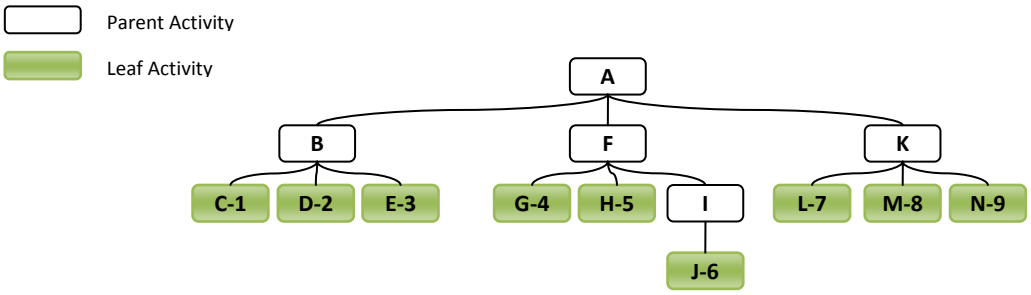


Figure 20: Order of Activity Delivery during a "continue" or "previous" sequencing request

The *flowSubprocess* accepts as input a traversal direction ("*forward*" or "*backward*") indicating how the Activity Tree will be traversed and the endpoint reference of an activity and tries to traverse the Activity Tree from this activity in the specified direction. It returns the activity that the traversal stopped and an indication whether this activity is allowed to be delivered. The algorithm of this sub-process is described below.

The input message of the operation is:

```
<xs:complexType name="flowTraversalInput">
   <xs:sequence>
      <xs:element name="activityEPR" type="wsa:EndpointReferenceType" minOccurs="0" />
      <xs:element name="orgEPR" type="wsa:EndpointReferenceType" />
      <xs:element name="traversalDirection" type="xs:string" minOccurs="0" />
      <xs:element name="previousTraversalDirection" type="xs:string" minOccurs="0" />
      <xs:element name="considerChildrenFlag" type="xs:boolean"></xs:element>
   </xs:sequence>
</xs:complexType>
```

Listing 27: flowSubprocess's input message

This operation utilizes two other sub-processes defined in the IMS Simple Sequencing Specification, the "*Flow Tree Traversal Subprocess*" (SB.2.1) and the "Flow *Activity Sequencing Subprocess*" (SB.2.2)

- *Flow Tree Traversal Subprocess* traverses the tree either in forward or backward direction to identify the next activity (one single step in the tree traversal). Only precondition sequencing rules related to traversing the tree in a "Flow" control mode are applied. On success, it returns the next activity in the tree in the traversal order.
- *Flow Activity Sequencing Subprocess* is used to determine if an activity should be delivered or if the traversal should examine the next activity in the tree. The process is applied in a single activity and checking both limit conditions and precondition sequencing rules.

The algorithm of the *Flow Subprocess* taken from the IMS SS is the following:

1) *Obtain a candidate activity as trying to move away from the activity in the indicated traversal direction.*
2) *Loop*
3) *If the sequencing control flow of the parent is false, ending the flow subprocess with nothing to deliver*
4) *If the candidate activity is Skipped, attempt to move away from this activity in the indicated direction*
5) *If the candidate activity is Disabled, then exit flow sub-process with nothing to deliver*
6) *If the candidate activity violates Limit conditions, exit flow subprocess with nothing to deliver.*
7) *If the candidate activity is a leaf, end flow sub process with this activity*
8) *If the candidate activity is a cluster, enter cluster with in the appropriate direction*
    a. *If traversing forward, the first available children is the next activity*
    b. *If traversing backward and the cluster has a Forward Only as False, the next activity is the last child*
    c. *If traversing backward and the cluster has a Forward Only as True, the next activity is the first child – temporarily (flow forward)*
9) *If nothing is identified for delivery, exit flow sub process with nothing to deliver*
10) *Continue loop step 2*

The output message of the FlowSubprocess is listed below. It contains the endpoint reference of the activity identified for delivery and an indication whether the sequencing session must end or not.

```
<xs:complexType name="flowTraversalOutput">
   <xs:sequence>
      <xs:element name="deliverable" type="xs:boolean" />
      <xs:element name="nextActivityEPR" type="wsa:EndpointReferenceType" minOccurs="0" />
      <xs:element name="endSequencingSession" type="xs:boolean" />
      <xs:element name="exception" type="xs:string" minOccurs="0" />
      <xs:element name="exceptionDescription" type="xs:string" minOccurs="0" />
   </xs:sequence>
</xs:complexType>
```

Listing 28: FlowSubprocess's output message

5.2.4 DeliveryService

The DeliveryService is implementing the *Delivery Behavior* as defined in the IMS Simple Sequencing specification. This behavior is responsible to validate the identified activity from the SequencingProcessService and to deliver or not the content. The messages and portType parts of the WSDL file associated with this service is shown in the following listing:

```
<wsdl:message name="deliveryOperationRequest">
   <wsdl:part  name="deliveryOperationRequest" element="tns:deliveryInput" />
</wsdl:message>

<wsdl:message name="deliveryOperationResponse">
   <wsdl:part name="deliveryOperationResponse" element="tns:deliveryOutput" />
</wsdl:message>

<wsdl:message name="DeliveryException">
   <wsdl:part name="DeliveryException" element="tns:deliveryFault" />
</wsdl:message>

<wsdl:portType name="DeliveryPortType">
   <wsdl:operation name="checkDeliveryOperation">
     <wsdl:input name="DeliveryOperationRequest" message="tns:deliveryOperationRequest"/>
     <wsdl:output name="DeliveryOperationResponse"
message="tns:deliveryOperationResponse"/>
     <wsdl:fault name="DeliveryException" message="tns:DeliveryException"/>
   </wsdl:operation>
</wsdl:portType>
```

Listing 29: Messages and PortType of Delivery Service

The *DeliveryService* expects as input the endpoint references of the organization activity, the Current Activity and the identified for delivery endpoint reference from the *SequencingProcessService*. The service will validate as true the activity passed as input if it is a leaf, it is not disabled (no sequencing rule with 'Disabled' rule action) and no limit conditions are violated.

The input message of the service is:

```
<xsd:complexType name="deliveryProcessInput">
   <xsd:sequence>
     <xsd:element name="activityEPR" type="wsa:EndpointReferenceType" minOccurs="0"
                  nillable="true" />
     <xsd:element name="deliveryEPR" type="wsa:EndpointReferenceType" minOccurs="0"
                  nillable="true" />
     <xsd:element name="orgEPR" type="wsa:EndpointReferenceType" minOccurs="0"
                  nillable="true" />
     </xsd:sequence>
 </xsd:complexType>
```

Listing 30: Delivery's input message

The output message of the service is:

```
<xsd:complexType name="deliveryProcessOutput">
   <xsd:sequence>
      <xsd:element name="deliverable" type="xsd:boolean"/>
      <xsd:element name="exception" type="xsd:string" minOccurs="0" nillable="true" />
   </xsd:sequence>
</xsd:complexType>
```

Listing 31: Delivery's service output message

The service returns the result of the validation. If the result is false, the exception code and description information will be contained in the "*exception*" element.

5.2.5 ContentDeliveryService

The *ContentDeliveryService* is implementing the Content Delivery Environment Process 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 behavior across LMSs without specifying the underlying LMS implementation.

This service implements and provides as operations the following process and subprocesses of the IMS SS specification:

- *Content Delivery Environment Process* DB.2
  - This is the main operation of the service and its description was given above
- *Clear Suspended Activity Subprocess* DB.2.1
  - This operation is invoked to change the state of the activities from suspended to not suspended for every activity belonging in the path from the root of Activity Tree to the Suspended Activity, inclusive.

The message and portType parts of the service's WSDL is shown in the following listing:

```
<wsdl:message name="ContentDeliveryEnvironmentOperationRequest">
   <wsdl:part name="part1" element="tns:contentDeliveryInput"/>
</wsdl:message>
<wsdl:message name="ContentDeliveryEnvironmentProcessOperationResponse">
   <wsdl:part name="part1" element="tns:contentDeliveryOutput"/>
</wsdl:message>
<wsdl:message name="ContentDeliveryException">
   <wsdl:part name="ContentDeliveryException" element="tns:contentdeliveryFault" />
</wsdl:message>
<wsdl:message name="ClearSuspendedActivityOperationRequest">
   <wsdl:part name="part1" element="tns:clearSuspendedActivityInput"/>
</wsdl:message>
<wsdl:message name="ClearSuspendedActivityOperationResponse">
   <wsdl:part name="part1" element="tns:WSResourceEndpointOutput"/>
</wsdl:message>

<wsdl:portType name="ContentDeliveryServicePortType">
   <wsdl:operation name="contentDeliveryEnvironmentOperation">
      <wsdl:input name="ContentDeliveryEnvironmentOperationRequest"
                  message="tns:ContentDeliveryEnvironmentOperationRequest"/>
      <wsdl:output name="ContentDeliveryEnvironmentProcessOperationResponse"
                  message="tns:ContentDeliveryEnvironmentProcessOperationResponse"/>
      <wsdl:fault name="ContentDeliveryException" message="tns:ContentDeliveryException"/>
   </wsdl:operation>
   <wsdl:operation name="clearSuspendedActivityOperation">
      <wsdl:input name="ClearSuspendedActivityOperationRequest"
                  message="tns:ClearSuspendedActivityOperationRequest"/>
      <wsdl:output name="ClearSuspendedActivityOperationResponse"
                  message="tns:ClearSuspendedActivityOperationResponse"/>
   </wsdl:operation>
</wsdl:portType>
```

Listing 32: ContentDeliveryEnvironmentService's messages and portType

The input message of the service is:

```
<xsd:complexType name="contentDeliveryEnvironmentInput">
   <xsd:sequence>
      <xsd:element name="currentActivityEPR" minOccurs="0" type="wsa:EndpointReferenceType"
                   nillable="true" />
      <xsd:element name="deliveryEPR" minOccurs="0" type="wsa:EndpointReferenceType"/>
      <xsd:element name="suspendedActivityEPR" minOccurs="0" type="wsa:EndpointReferenceType"
                   nillable="true" />
      <xsd:element name="orgEPR" type="wsa:EndpointReferenceType"/>
   </xsd:sequence>
</xsd:complexType>
```

Listing 33: ContentDeliveryEnvironmentService's input message

The service expects as input the endpoint references of Current Activity (optional), Suspended Activity (optional), identified Delivery Activity(optional) and the organization activity (mandatory).

The output of the service is:

```
<xsd:complexType name="contentDeliveryEnvironmentOutput">
   <xsd:sequence>
      <xsd:element name="currentActivityEPR" minOccurs="0" type="wsa:EndpointReferenceType"/>
      <xsd:element name="deliveredActivityId" type="xsd:string" minOccurs="0" nillable="true" />
      <xsd:element name="deliveryLocation" type="xsd:string" minOccurs="0" nillable="true" />
      <xsd:element name="numAttempts" type="xsd:nonNegativeInteger" minOccurs="0"
                   nillable="true" />
      <xsd:element name="suspendedActivityEPR" minOccurs="0" type="wsa:EndpointReferenceType"
                   nillable="true" />
       <xsd:element name="exception" minOccurs="0" type="xsd:string" nillable="true"/>
   </xsd:sequence>
</xsd:complexType>
```

Listing 34: ContentDeliveryEnvironmentService's output message

The service returns the endpoint references of the Current Activity and the Suspended Activity, the activity identifier of the delivered activity as defined in the imsmanifest.xml, the url of the delivered activity and a number indicating the attempts on the activity for the learner.

Upon exception, the service returns an Axis fault of type:

```
<xs:complexType name="SequencingException">
   <xs:sequence>
      <xs:element name="code" type="xs:string" />
      <xs:element name="description" type="xs:string" nillable="true" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
```

Listing 35: ContentDeliveryEnvironmentService's exception type

5.2.6 OverallRollupService

The *OverallRollupService* is not directly invoked from the BPEL process but operations defined in this service are called from the *TerminationProcessService*. This service is an implementation of the Rollup Behavior described in the IMS SS specification. The IMS SS Rollup behavior 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:

- Objective Progress Information: information about results of the learner's interaction related to an objective
- Attempt Progress Information: information about the results of the learner's interaction on an attempt at an activity.

The Rollup Process is controlled by parts of the Sequencing Definition Model such as *Rollup Controls*, *Rollup Rule Definitions*, *Objective Description* and *Delivery Controls*. The rollup conditions and actions are described in Appendix A.

The operations described in the WSDL and provided by this service are:

- *Measure Rollup Operation RB 1.1*: sets the cluster's measure to the average weighted measure of the cluster's children. The cluster will always have a defined value if any of its tracked children has a defined measure for its rollup objective.
- *Objective Rollup Using Measure Process RB.1.2.a*: is used to rollup objective information. An activity's primary objective is satisfied if the Objective Normalized Measure is equal or greater than the Objective Minimum Satisfied Normalized Measure. This operation is called only if the Objective Satisfied by Measure attribute is True for the activity's primary objective.
- *Objective Rollup Using Rules Process RB.1.2.b*: rollup rules with Rollup Actions of "satisfied" or "not satisfied" are evaluated for an activity. Values for Objective Progress Status and Objective Satisfied Status are updated.
- *Activity Progress Rollup Operation RB.1.3*: invoked to determine the completion status (Attempt Progress Status and Attempt Completion Status) of the current attempt on the activity from the completion status of the children of the activity. Rollup actions of "completed" and "incomplete" are evaluate for the activity.
- *Rollup Rule Check Subprocess RB.1.4*: evaluates an activity against a set of rollup rules of a specified Rollup Action.
- *Evaluate Rollup Conditions Subprocess RB.1.4.1*: evaluates an activity against a set of Rollup Conditions.
- *Check Child For Rollup Operation RB.1.4.2*: determines if tracking information for an activity should be included during rollup. Tests the activity's Rollup Controls SM.8 against an intended Rollup Action.
- *Overall Rollup Process RB 1.5*: implements the rollup behavior as defined in the IMS SS specification. The result of it depends in terms of different processes, each of which is applied to one part of the tracking model. This operation propagates tracking results upward through the entire Activity Tree. Rollup path is the path from the root of the activity tree to the current activity. Rollup is applied to all activities in this path, starting from the parent of the activity that triggered the rollup and proceeding backward along the path of parents of activities to the root of the activity tree.

```
 <wsdl:message name="OverallRollupRequest">
    <wsdl:part  name="OverallRollupRequest" element="tns:overallInput" />
  </wsdl:message>
  <wsdl:message name="OverallRollupResponse">
    <wsdl:part name="OverallRollupResponse" element="types:emptyElement" />
  </wsdl:message>
  <wsdl:message name="MeasureRollupRequest">
    <wsdl:part name="MeasureRollupRequest" element="tns:rollupInput" />
  </wsdl:message>
  <wsdl:message name="MeasureRollupResponse">
    <wsdl:part name="MeasureRollupResponse" element="types:emptyElement" />
  </wsdl:message>
  <wsdl:message name="ObjectiveRollupMeasureRequest">
    <wsdl:part name="ObjectiveRollupMeasureRequest" element="tns:rollupInput" />
  </wsdl:message>
  <wsdl:message name="ObjectiveRollupMeasureResponse">
    <wsdl:part name="ObjectiveRollupMeasureResponse" element="types:emptyElement" />
  </wsdl:message>
  <wsdl:message name="ObjectiveRollupRulesRequest">
    <wsdl:part name="ObjectiveRollupRulesRequest" element="tns:rollupInput" />
  </wsdl:message>
  <wsdl:message name="ObjectiveRollupRulesResponse">
    <wsdl:part name="ObjectiveRollupRulesResponse" element="types:emptyElement" />
  </wsdl:message>
  <wsdl:message name="ActivityProgressRollupRequest">
    <wsdl:part name="ActivityProgressRollupRequest" element="tns:rollupInput" />
  </wsdl:message>
  <wsdl:message name="ActivityProgressRollupResponse">
    <wsdl:part name="ActivityProgressRollupResponse" element="types:emptyElement" />
  </wsdl:message>
<wsdl:message name="RollupRuleCheckRequest">
    <wsdl:part name="RollupRuleCheckRequest" element="tns:rollupRuleCheckInput" />
  </wsdl:message>
<wsdl:message name="RollupRuleCheckResponse">
    <wsdl:part name="RollupRuleCheckpResponse" element="tns:rollupRuleCheckOutput" />
  </wsdl:message>
  <wsdl:message name="EvaluateRollupRequest">
    <wsdl:part name="EvaluateRollupRequest" element="tns:evaluateRollupInput" />
  </wsdl:message>
  <wsdl:message name="EvaluateRollupResponse">
    <wsdl:part name="EvaluateRollupResponse" element="tns:evaluateRollupOutput" />
  </wsdl:message>
  <wsdl:message name="CheckChildRollupRequest">
    <wsdl:part name="CheckChildRollupRequest" element="tns:checkChildForRollupInput" />
  </wsdl:message>
  <wsdl:message name="CheckChildRollupResponse">
    <wsdl:part name="CheckChildRollupResponse" element="tns:checkChildForRollupOutput" />
  </wsdl:message>
```

```
<wsdl:portType name="OverallRollupPortType">
    <wsdl:operation name="overallRollupProcess">
      <wsdl:input name="OverallRollupRequest" message="tns:OverallRollupRequest"/>
      <wsdl:output name="OverallRollupResponse" message="tns:OverallRollupResponse"/>
    </wsdl:operation>
    <wsdl:operation name="measureRollupOperation">
      <wsdl:input name="MeasureRollupRequest" message="tns:MeasureRollupRequest"/>
      <wsdl:output name="MeasureRollupResponse" message="tns:MeasureRollupResponse"/>
    </wsdl:operation>
    <wsdl:operation name="objectiveRollupUsingMeasureOperation">
      <wsdl:input name="ObjectiveRollupMeasureRequest"
message="tns:ObjectiveRollupMeasureRequest"/>
      <wsdl:output name="ObjectiveRollupMeasureResponse"
message="tns:ObjectiveRollupMeasureResponse"/>
    </wsdl:operation>
    <wsdl:operation name="objectiveRollupUsingRulesOperation">
      <wsdl:input name="ObjectiveRollupRulesRequest"
message="tns:ObjectiveRollupRulesRequest"/>
      <wsdl:output name="ObjectiveRollupRulesResponse"
message="tns:ObjectiveRollupRulesResponse"/>
    </wsdl:operation>
    <wsdl:operation name="activityProgressRollupOperation">
      <wsdl:input name="ActivityProgressRollupRequest"
message="tns:ActivityProgressRollupRequest"/>
      <wsdl:output name="ActivityProgressRollupResponse"
message="tns:ActivityProgressRollupResponse"/>
    </wsdl:operation>
    <wsdl:operation name="rollupRuleCheckOperation">
      <wsdl:input name="RollupRuleCheckRequest" message="tns:RollupRuleCheckRequest"/>
      <wsdl:output name="RollupRuleCheckRollupResponse"
message="tns:RollupRuleCheckResponse"/>
    </wsdl:operation>
    <wsdl:operation name="evaluateRollupConditionsOperation">
      <wsdl:input name="EvaluateRollupRequest" message="tns:EvaluateRollupRequest"/>
      <wsdl:output name="EvaluateRollupResponse" message="tns:EvaluateRollupResponse"/>
    </wsdl:operation>
    <wsdl:operation name="checkChildForRollupOperation">
      <wsdl:input name="CheckChildRollupRequest" message="tns:CheckChildRollupRequest"/>
      <wsdl:output name="CheckChildRollupResponse" message="tns:CheckChildRollupResponse"/>
    </wsdl:operation>
  </wsdl:portType>
```

Listing 36: Message and PortType parts of the OverallRollupService WSDL

5.2.7 UtilitiesProcessService

The *UtilitiesProcessService* is not directly invoked from the BPEL process but operations defined in this service are called from all other services. The operations implemented by this service correspond to the following processes and subprocesses defined in the IMS Simple Sequencing specification:

- *Limit Conditions Check Process UP.1*: this operation applies the limit conditions of the SDM to determine if any of the activity's limit conditions have been violated. It is controlled by the limit conditions (limits on how many times, how long and when activity is allowed) and delivery controls (describe if the activity is tracked)
- *Sequencing Rules Check Process UP.2*: this operation evaluates a set of sequencing rules for an activity, attempting to identify an action that must be applied.
- *Sequencing Rule Check Subprocess UP*.2.1: evaluates a set of conditions against an activity's current tracking information.
- *Terminate Descendent Attempts Process UP*.3: this operation finds the common ancestor of an activity and the Current Activity and for the descendents of the identified common ancestor applies the End Attempt Process.

- *End Attempt Process UP*.4: this operation sets the Attempt Progress Status and Attempt Completion Status as well as the Objective Progress Status and Objective Satisfied Status if the activity is a leaf and it is tracked. If the activity is a cluster, it determines if the activity should be suspended depending of the activity status of its children. Also, if the activity is leaf and is tracked then a mapping between run-time data model and tracking model is take place.
- *Check Activity Process UP*.5: this operation determines if limit conditions and Disabled rules are violated for a given activity.

The messages and portType parts are shown in the following Listing.

```
<wsdl:message name="limitConditionsCheckProcessRequest">
    <wsdl:part name="limitConditionsCheckProcessRequest"
               element ="tns:limitConditionsCheckProcessInput"/>
  </wsdl:message>
  <wsdl:message name="limitConditionsCheckProcessResponse">
    <wsdl:part name="response" type="xsd:boolean" />
  </wsdl:message>

  <wsdl:message name="sequencingRulesCheckProcessRequest">
        <wsdl:part name="sequencingRulesCheckProcessRequest"
                   element="tns:sequencingRulesCheckProcessInput" />
  </wsdl:message>
  <wsdl:message name="sequencingRulesCheckProcessResponse">
        <wsdl:part name="sequencingRulesCheckProcessResponse" type="xsd:string" />
  </wsdl:message>

  <wsdl:message name="terminateDescendentAttemptProcessRequest">
        <wsdl:part name="terminateDescendentAttemptProcessRequest"
element="tns:terminateDescendentAttemptProcessInput" />
  </wsdl:message>
  <wsdl:message name="terminateDescendentAttemptProcessResponse">
        <wsdl:part name="terminateDescendentAttemptProcessResponse"
                    element="utilTypes:emptyElement" />
  </wsdl:message>

  <wsdl:message name="endAttemptProcessRequest">
        <wsdl:part name="endAttemptProcessRequest" element="tns:endAttemptInput" />
  </wsdl:message>
  <wsdl:message name="endAttemptProcessResponse">
        <wsdl:part name="endAttemptProcessResponse" type="xsd:string" />
  </wsdl:message>

  <wsdl:message name="checkActivityProcessRequest">
        <wsdl:part name="checkActivityProcessRequest" element="tns:checkActivityInput" />
  </wsdl:message>
  <wsdl:message name="checkActivityProcessResponse">
        <wsdl:part name="checkActivityProcessResponse" type="xsd:boolean" />
  </wsdl:message>


  <wsdl:message name="commonAncestorRequest">
        <wsdl:part name="commonAncestorRequest" element="tns:commonAncestorInput" />
  </wsdl:message>
  <wsdl:message name="commonAncestorResponse">
        <wsdl:part name="commonAncestorResponse" element="tns:commonAncestorOutput" />
  </wsdl:message>

 <wsdl:portType name="UtilitiesProcessServicePortType">
    <wsdl:operation name="limitConditionsCheckProcess">
      <wsdl:input name="limitConditionsCheckProcessRequest"
                  message="tns:limitConditionsCheckProcessRequest"/>
      <wsdl:output name="limitConditionsCheckProcessResponse"
                   message="tns:limitConditionsCheckProcessResponse"/>
    </wsdl:operation>
```

```
    <wsdl:operation name="sequencingRulesCheckProcess">
        <wsdl:input name="sequencingRulesCheckProcessRequest"
                    message="tns:sequencingRulesCheckProcessRequest" />
        <wsdl:output name="sequencingRulesCheckProcessResponse"
                    message="tns:sequencingRulesCheckProcessResponse" />
    </wsdl:operation>

    <wsdl:operation name="terminateDescendentAttemptProcess">
        <wsdl:input name="terminateDescendentAttemptProcessRequest"
message="tns:terminateDescendentAttemptProcessRequest"></wsdl:input>
        <wsdl:output name="terminateDescendentAttemptProcessResponse"
message="tns:terminateDescendentAttemptProcessResponse"></wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="endAttemptProcess">
        <wsdl:input name="endAttemptProcessRequest" message="tns:endAttemptProcessRequest" />
        <wsdl:output name="endAttemptProcessResponse"
                     message="tns:endAttemptProcessResponse" />
    </wsdl:operation>
    <wsdl:operation name="checkActivityProcess">
        <wsdl:input name="checkActivityProcessRequest"
                   message="tns:checkActivityProcessRequest" />
        <wsdl:output name="checkActivityProcessResponse"
                     message="tns:checkActivityProcessResponse" />
    </wsdl:operation>
    <wsdl:operation name="commonAncestor">
        <wsdl:input name="commonAncestorRequest" message="tns:commonAncestorRequest" />
        <wsdl:output name="commonAncestorResponse" message="tns:commonAncestorResponse" />
    </wsdl:operation>

  </wsdl:portType>
```

Listing 37: Message and PortType parts of the UtilitiesProcessService WSDL

5.2.8 SCORMPackageUtilService

The SCORMPackageUtilService is invoked from the BPEL process prior starting 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 LO-Resources, creating the TOC based on *imsmanifest.xml*, write suspended state of the Activity Tree and initialize an Activity Tree according to previous suspended state. Parts of its WSDL file are shown in the following listing:

```
<wsdl:message name="initRequest">
    <wsdl:part name="initRequest" element="tns:initParameters" />
  </wsdl:message>
  <wsdl:message name="initResponse">
    <wsdl:part name="initResponse" element="tns:SCORMPackageObject" />
  </wsdl:message>

  <wsdl:message name="writeToFileRequest">
    <wsdl:part name="writeRequest" element="tns:writeRequestParams" />
  </wsdl:message>
  <wsdl:message name="writeToFileResponse">
    <wsdl:part name="writeResponse" element="tns:emptyElement" />
  </wsdl:message>

  <wsdl:message name="createActivityTreeRequest">
    <wsdl:part name="createActivityTreeRequest" element="tns:createActivityTreeInput" />
  </wsdl:message>
  <wsdl:message name="createActivityTreeResponse">
    <wsdl:part name="createActivityTreeResponse" element="tns:createActivityTreeOutput" />
  </wsdl:message>


  <wsdl:message name="createTOCRequest">
    <wsdl:part name="createTOCRequest" element="tns:tocparameters" />
  </wsdl:message>
  <wsdl:message name="createTOCResponse">
    <wsdl:part name="createTOCResponse" element="tns:tocLinks" />
  </wsdl:message>


  <wsdl:portType name="SCORMPackageUtilServicePortType">
    <wsdl:operation name="initialization">
      <wsdl:input name="initializationRequest" message="tns:initRequest"/>
      <wsdl:output name="initializationResponse" message="tns:initResponse"/>
    </wsdl:operation>
    <wsdl:operation name="writeActivityTreeToFile">
      <wsdl:input name="writeFileRequest" message="tns:writeToFileRequest"/>
      <wsdl:output name="writeFileResponse" message="tns:writeToFileResponse"/>
    </wsdl:operation>
    <wsdl:operation name="createActivityTree">
      <wsdl:input name="createActivityTreeRequest" message="tns:createActivityTreeRequest"/>
      <wsdl:output name="createActivityTreeResponse" message="tns:createActivityTreeResponse"/>
    </wsdl:operation>
    <wsdl:operation name="createTOCList">
      <wsdl:input name="createTOCRequest" message="tns:createTOCRequest"/>
      <wsdl:output name="createTOCResponse" message="tns:createTOCResponse"/>
    </wsdl:operation>
  </wsdl:portType>
```

Listing 38: Message and PortType parts of the SCORMPackageUtilService WSDL

The functionality of this service is divided into the following operations:

- i*nitialization*: creates an Activity Tree for a student based on the URL of the package. This service remotely "sees into" the package .zip file, locates the "imsmanifest.xml" file and constructs each LO-Resource based on the information found in the xml file.
- *writeActivityTreeToFile*: saves the information regarding Tracking Model and the Run-time CMI model into an XML file when the learner suspends his interaction
- *createActivityTree*: creates an Activity Tree for a given URL which previously was suspended and an suspended XML file was created including all the saved information for Tracking Model and Run-Time CMI model.
- *createTOCList*: creates the Table of Contents for a given URL corresponding to a .zip package

5.3 Implementing Sequencing as BPEL process

The Overall Sequencing Process as defined by the IMS SS specification is just a loop over sequencing behaviors with the aim to identify the next deliverable activity, The pseudo code of this process is depicted in listing 39. The IMS SS assumes that one request is processed, until processing other. Till now, we silently refer to the definition of the sequencing session without explaining how this session is determined. Sequencing session refers to the time passed between the first successful execution of the Overall Sequencing Process until a successful termination on the root activity.

```
1. Loop - Wait for a navigation request
1.1.      Apply the Navigation Request Process to the navigation request
1.2.      If the Navigation Request Process returned navigation request Not Valid Then
1.2.1.        Handle the navigation request error
1.2.2.        Continue Loop - wait for the next navigation request
          End If
1.3.      If there is a termination request Then
1.3.1.        Apply the Termination Request Process to the termination request
1.3.2.        If the Termination Request Process returned termination request Not Valid Then
1.3.2.1.          Handle the termination request error
1.3.2.2.          Continue Loop - wait for the next navigation request
              End If
1.3.3.        If Termination Request Process returned a sequencing request Then
1.3.3.1.          Replace any pending sequencing request by the sequencing request returned by the
                  Termination Request Process
              End If
          End If
1.4.      If there is a sequencing request Then
1.4.1.      Apply the Sequencing Request Process to the sequencing request
1.4.2.      If the Sequencing Request Process (SB.2.12) returned sequencing request Not Valid Then
1.4.2.1.        Handle the sequencing request error
1.4.2.2.        Continue Loop - wait for the next navigation request
          End If
1.4.3.    If the Sequencing Request Process returned a request to end the sequencing session Then
1.4.3.1         Exit Overall Sequencing Process -
          End If
1.4.4.    If the Sequencing Request Process did not identify an activity for delivery Then
1.4.4.1.        Continue Loop - wait for the next navigation request
          End If
1.4.5.    delivery request is for the activity identified by the Sequencing Request Process
          End If
1.5.      If there is a delivery request Then
1.5.1.        Apply the Delivery Request Process to the delivery request
1.5.2.        If the Delivery Request Process returned delivery request Not Valid Then
1.5.2.1.          Handle the delivery request error
1.5.2.2.          Continue Loop - wait for the next navigation request
          End If
1.5.3.    Apply the Content Delivery Environment Process to the delivery request
          End If
2.    End Loop - wait for the next navigation request
```

Listing 39: Overall Sequencing Process pseudocode

The above pseudocode has been implemented with constructs introduced by the BPEL language. The first step is to define the partner links that participate in the process. For each service discussed in the Section 5.2, a <*partnerLink*> and a role are defined. The process makes synchronous invocations to the services so for each partner only one role is declared. The second step is to declare the global variables of the process that holds either the state of it or are used for the invocations. Variables corresponding to input and output messages have been defined depending on the XML Schema types

constructed for these messages for each service. Furthermore, variables of endpoint reference type as defined in the WS-Addressing XML Schema have been declared corresponding to the Global State Information, the *Current Activity* and the *Suspended Activity* because every sequencing service depends on this information for execution (see Chapter 2). Correlation sets are created depending on the learner identifier and the sequencing session identifier because a BPEL engine depends on these sets for finding the appropriate process instance. The next step is to define the sequence of services' invocations. The first service that is invoked, is the SCORMPackageUtilService (see Section 5.2.8) in order to construct the Activity Tree. This service is invoked only if the navigation request is "*Start*", "*Resume All*" and "*Choice*" and the sequencing session has not started already. The other services are called in the same sequence as described in Figure 19, so a <*sequence*> structured activity is defined. This sequence is wrapped in a <*while*> activity in order to accept incoming messages from the client and reply to them. The process is completed when the sequencing session is ended.

The input of the BPEL process is described with the following XML Schema complex type:

```
<complexType name="ProcessInput">
  <sequence>
    <element name="studentID" type="xsd:string" />
    <element name="studentName" type="xsd:string" />
    <element name="packageURL" type="xsd:string" />
    <element name="navigationEvent" type="xsd:string" />
    <element name="sequencingSession" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <element name="organizationEPR" type="wsa:EndpointReferenceType" minOccurs="0" nillable="true" />
    <element name="currentActivityEPR" type="wsa:EndpointReferenceType" nillable="true" minOccurs="0"/>
    <element name="suspendedActivityEPR" type="wsa:EndpointReferenceType" minOccurs="0" nillable="true"/>
    <element name="targetActivityEPR" type="wsa:EndpointReferenceType" minOccurs="0" nillable="true"/>
    <element name="organizationID" type="xsd:string" nillable="true" minOccurs="0"/>
  </sequence>
 </complexType>
```

Listing 40: BPEL message input type

The output of the BPEL process is:

```
<complexType name="ProcessOutput">
  <sequence>
    <element name="outputStr" type="xsd:string"/>
    <element name="exitCourse" type="xsd:boolean" />
    <element name="endSequencingSession" type="xsd:boolean" />
    <element name="currentActivityEPR" type="wsa:EndpointReferenceType" nillable="true" minOccurs="0"/>
    <element name="deliveredActivityId" type="xsd:string" minOccurs="0" />
    <element name="deliveryEPR" type="wsa:EndpointReferenceType" nillable="true" minOccurs="0" />
    <element name="activityLocation" type="xsd:string" nillable="true" minOccurs="0" />
    <element name="numAttempts" type="xsd:int" minOccurs="0" nillable="false" />
    <element name="exception" type="tns:ExceptionType" minOccurs="0" nillable="true" />
  </sequence>
 </complexType>
```

Listing 41: BPEL output message type

The BPEL engine used for the implemented BPEL process is the Active BPEL engine from Active Endpoints. The engine was chosen because it is open source, runs on the Jakarta server and support Axis 1.4 in contrast with the Apache ODE Engine which is considered as the best open source BPEL engine but it only supports Axis 2.0. Axis 1.4 has been chosen for the implementation of every Web service of this master 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 and till now there is no compatibility between the Axis 1.4 with Axis 2.0 or JAXB 2.0.

5.4 Integrating BPEL Sequencing Engine to the SCORM Run-Time Environment

The final phase of this work is the integration of the proposed sequencing engine 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 or quit or suspend the application.

In this environment the communication between a content object and the LMS is achieved through Javascript calls to methods which signatures are defined in the SCORM RTE book. The delivered learning content is responsible for initiating the communication with the LMS and to achieve this, the SCORM RTE book defines that the LMS should provide an API implementation which is exposed as a Document Object Model (DOM) object. The content responsibility is to search this API Instance to the window hierarchy until it is found. The ADL Run-Time Environment uses a Java applet that communicates with a Java Servlet residing in the server-side part of the LMS. For integration purposes, we do not used this applet but AJAX technology has been used for communicating with the server-side of the application because the WS-Resource qualified endpoint reference is used to invoke the API methods of the WS-Resource.

5.5 Summary

This chapter has provided the information regarding the implementation details of the BPEL sequencing engine. It has described the service components of the engine by explaining which processes and subprocesses defined in the IMS SS specification are implemented by which service. It has also depicted the data elements that were defined for the input and output messages of these Web services and some parts of their WSDL interfaces. Finally, the process that exhibits the overall sequencing functionality has been defined.

Chapter 6.
Conclusions and Future Work

This master's thesis report 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. For this reason, we have analyzed existing e-Learning systems and found out that most e-Learning platforms that try to be redesigned and re-implemented according to the guidelines of SCORM are still missing the functionality of a sequencing engine as introduced in the IMS Simple Sequencing specification. So this work provides the functionality of such an engine as an external web service to be integrated in existing platforms. Moreover, Web services wrappers were presented for the representation of Learning Objects. This has been the first approach for representing characteristics and functionalities of learning content with Web Services. This approach further refines the problem of the existence of multiple data models introduced by the SCORM standard into one whole representation.

This work utilizes Web services standards like WSDL, BPEL, WS-Addressing, WS-Resource and finally BPEL for the implementation. The services that constitute the engine have been implemented in the Java programming language and used well-established open source libraries and frameworks like Java WS-Core library implementation of the WS-Resource Framework and Axis 1.4 for implementing and deploying the Web services of this work. The ActiveBPEL engine has been used as the runtime BPEL engine for executing the implemented sequencing process. Tomcat server is used as the deployment server of the services, the WSRF container (a servlet of the Globus Toolkit) containing our implementation of the WS-Resources wrappers of Learning Objects and for the ActiveBPEL engine. For testing purposes, the BPEL sequencing engine has been integrated in the ADL SCORM Sample Runtime Environment.

However, there are some missing features in this 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. The use of the WS-Addressing for the implementation makes possible this integration as policies can become parts of the endpoint references. The architecture of this system can be further enhanced with a repository which could be used for the discovery of learning objects based on metadata for learning objects such as the IEEE Learning Object Metadata Standard.

Appendix A

Sequencing Definition Model

A.1 Conditions in Sequencing Rules

Condition	Description
Satisfied	The condition evaluates to True if the *Objective Progress Status* for the objective associated with the activity (indicated by the *Rule Condition Referenced Objective*) is True and the *Objective Satisfied Status* for the objective associated with the activity (indicated by the *Rule Condition Referenced Objective*) is True.
Completed	The Condition evaluates to True if the *Attempt Progress Status* for the activity and the *Attempt Completion Status* for the activity is True.
Attempted	The Condition evaluates to True if the *Activity Progress Status* for the activity is True and *Activity Attempt Count* for the activity is positive (i.e., the activity has been attempted).
always	The condition always evaluates to True
Objective Status Known	The Condition evaluates to True if the *Objective Progress Status* for the objective associated with the activity (indicated by *Rule Condition Referenced Objective*) is True.
Objective Measure Known	The Condition evaluates to True if the *Objective Measure Status* for the objective associated with the activity (indicated by *Rule Condition Referenced Objective*) is True.
Objective Measure Greater Than	The Condition evaluates to True if the *Objective Measure Status* for the objective associated with the activity (indicated by the *Rule Condition Referenced Objective*) is True and the *Objective Normalized Measure* for the objective associated with the activity (indicated by *Rule Condition Referenced Objective*) is greater than the *Rule Condition Measure Threshold*.
Objective Measure Less Than	The Condition evaluates to True if the *Objective Measure Status* for the objective associated with the activity (indicated by *Rule Condition Referenced Objective*) is True and the *Objective Normalized Measure* for the objective associated with the activity (*indicated by Rule Condition Referenced Objective*) is less than the *Rule Condition Measure Threshold*
Activity Progress Known	The Condition evaluates to True if the *Activity Progress Status* for the activity is True and the *Attempt Progress Status* for the activity is True.
Attempt Limit Exceeded	The condition evaluates to True if *Activity Progress Status* for the activity is True and the *Limit Condition Attempt Limit Control* for the activity is True and the *Activity Attempt Count* for the activity is equal or greater than the *Limit Condition Attempt Limit* for the activity

A.2 Pre-condition Rule Actions

Condition	Description

Skip	The activity is not considered a candidate for delivery during a *Flow* sequencing request.
Disabled	The activity may not be the target of any sequencing or delivery request.
Hidden from Choice	The activity may not be the target of a *Choice* sequencing request
Stop Forward Traversal	The activity will prevent activities following it (in a preorder traversal of the tree) from being considered candidates for delivery

A.3 Post-condition actions

Condition	Description
Exit Parent	Process an *Exit Parent* termination request
Exit All	Process an *Exit All* termination request and return an *Exit* sequencing request.
Retry	Return a *Retry* sequencing request
Retry All	Process an *Exit All* termination request and return a *Start* sequencing request
Continue	Return a *Continue* sequencing request
Previous	Return a *Previous* sequencing request

A.4 Exit-condition actions

Condition	Description
Exit	Unconditionally terminate the activity

A.5.1 Limit Conditions: Attempt Limits

Data Element	Description	Default Value
Limit Condition Attempt Control	Indicates that a limit condition on the number of attempts for the activity has been established (True or False) for the activity. If the value is False, there is no constraint on how many times the activity may be attempted	Boolean: false
Limit Condition Attempt Limit	Contains the maximum number of attempts for the activity. A zero value indicates the activity may not be accessed. The value is unreliable unless Limit Condition Attempt Control is True.	Non-negative integer: 0

A.5.2 Limit Conditions: Attempt Absolute Duration

Data Element	Description	Default Value
Limit Condition Attempt Absolute Duration Control	Indicates that a limit condition on the maximum time duration that a learner is permitted to spend on any single attempt on the activity has been established (True or False) for the activity. If the	Boolean: false

	value is False, there is no constraint on how long the learner may spend on the activity.	
Limit Condition Attempt Absolute Duration Limit	The maximum time duration that a learner is permitted to spend on any single attempt on the activity. This limit applies to the time the activity is active – from the time the activity begins until the time the activity ends, including any time the activity was suspended. A zero value indicates the activity may not be accessed. The value is unreliable unless Limit Condition Attempt Absolute Duration Control is True.	Duration - Accuracy 0.1 sec: 0

A.6.1 Rollup Rules Conditions:

Data Element	Description
Satisfied	The Condition evaluates to True if the *Objective Progress Status* for the rolled-up objective associated with the child activity is True and the *Objective Satisfied Status* for the rolled-up objective associated with the child activity is True.
Objective Status Known	The Condition evaluates to True if the *Objective Progress Status* for the rolled-up objective associated with the child activity is True.
Objective Measure Known	The Condition evaluates to True if the *Objective Measure Status* for the rolled-up objective associated with the child activity is True.
Completed	The Condition evaluates to True if the *Attempt Progress Status* for the child activity is True and the *Attempt Completion Status* for the child activity is True.
Activity Progress Known	The Condition evaluates to True if the *Activity Progress Status* for the child activity is True and the *Attempt Progress Status* for the child activity is True.
Attempted	The Condition evaluates to True if the *Activity Progress Status* for the child activity is True and *Activity Attempt Count* for the child activity is positive (i.e., the child activity has been attempted).
Attempt Limit Exceeded	The Condition evaluates to True if *Activity Progress Status* for the child activity is True and the *Limit Condition Attempt Limit Control* for the child activity is True and the *Activity Attempt Count* for the child activity is equal to or greater than the *Limit Condition Attempt Limit* for the child activity.

A.6.2 Rollup Rules Child Activity Set

Data Element	Description
***All* (default value)**	The rollup rule evaluates to True if and only if all of the children have a rollup condition (the result of the Condition Combination) value of True.
Any	The rollup rule condition evaluates to True if any of the children have a rollup condition (the result of the Condition Combination) value of True.
None	The rollup rule condition evaluates to True if none of the children have a rollup condition (the result of the Condition Combination) value of True.
At Least Count	The rollup rule condition evaluates to True if at least the number of children specified by the *Rollup Minimum Count* attribute have a rollup condition (the result of the Condition Combination) value of True.
At Least Percent	The rollup rule condition evaluates to True if at least the percentage of children specified in the *Rollup Minimum Percent* attribute have a rollup condition (the result of the Condition Combination) value of True.
***All* (default value)**	The rollup rule evaluates to True if and only if all of the children have a rollup condition (the result of the Condition Combination) value of True.

A.6.3 Rollup Rules Actions

Data Element	Description
Satisfied (default)	Set the: - *Objective Progress Status* for the rolled-up objective associated with the activity to True. - *Objective Satisfied Status* for the rolled-up objective associated with the activity to True.
Not Satisfied	Set the: - *Objective Progress Status* for the rolled-up objective associated with the activity to True. - *Objective Satisfied Status* for the rolled-up objective associated with the activity is set to False.
Completed	Set the: - *Attempt Progress Status* for the activity to True. - *Attempt Completion Status* for the activity to True.
Incomplete	Set the: - *Attempt Progress Status* for the activity to True. - *Attempt Completion Status* for the activity to False.

A.7.1 Selection Controls

Data Element	Description	Default Value
Selection Timing	This element indicates when a selection should occur. The possible values are: ▪ *Never* (default value): It denotes that the selection shall never be applied to cluster. All of the children of the activity are selected by default ▪ *Once:* It denotes that selection is applied before the first attempt on an cluster ▪ *On Each New Attempt:* It denotes that selection is applied before each new attempt on an activity	*Never*
Selection Count Status	Its value indicates if the *Selection Status* is (True or False) meaningful for the activity.	Boolean: false
Selection Count	Indicates the number of child activities that must be selected from the set of child activities associated with the activity. ▪ If *Selection Count* is larger than the number of child activities, all child activities are selected. ▪ If *Selection Count Status* is false, all child activities are selected.	Non-negative integer: 0

A.7.2 Randomization Controls

Data Element	Description	Default Value
Randomization Timing	This element indicates when the ordering of the children of the activity should occur. The possible value for this	*Never*

	element are: ▪ *Never* (default value): this value denotes that the randomization is never occurred ▪ *Once*: this value denotes that the randomization is applied before the first attempt on the activity ▪ *On Each New Attempt*: this value denotes that the randomization is applied before each new attempt on the activity.	
Randomize Children	Indicates that the order of the child activities is randomized.	Boolean: false

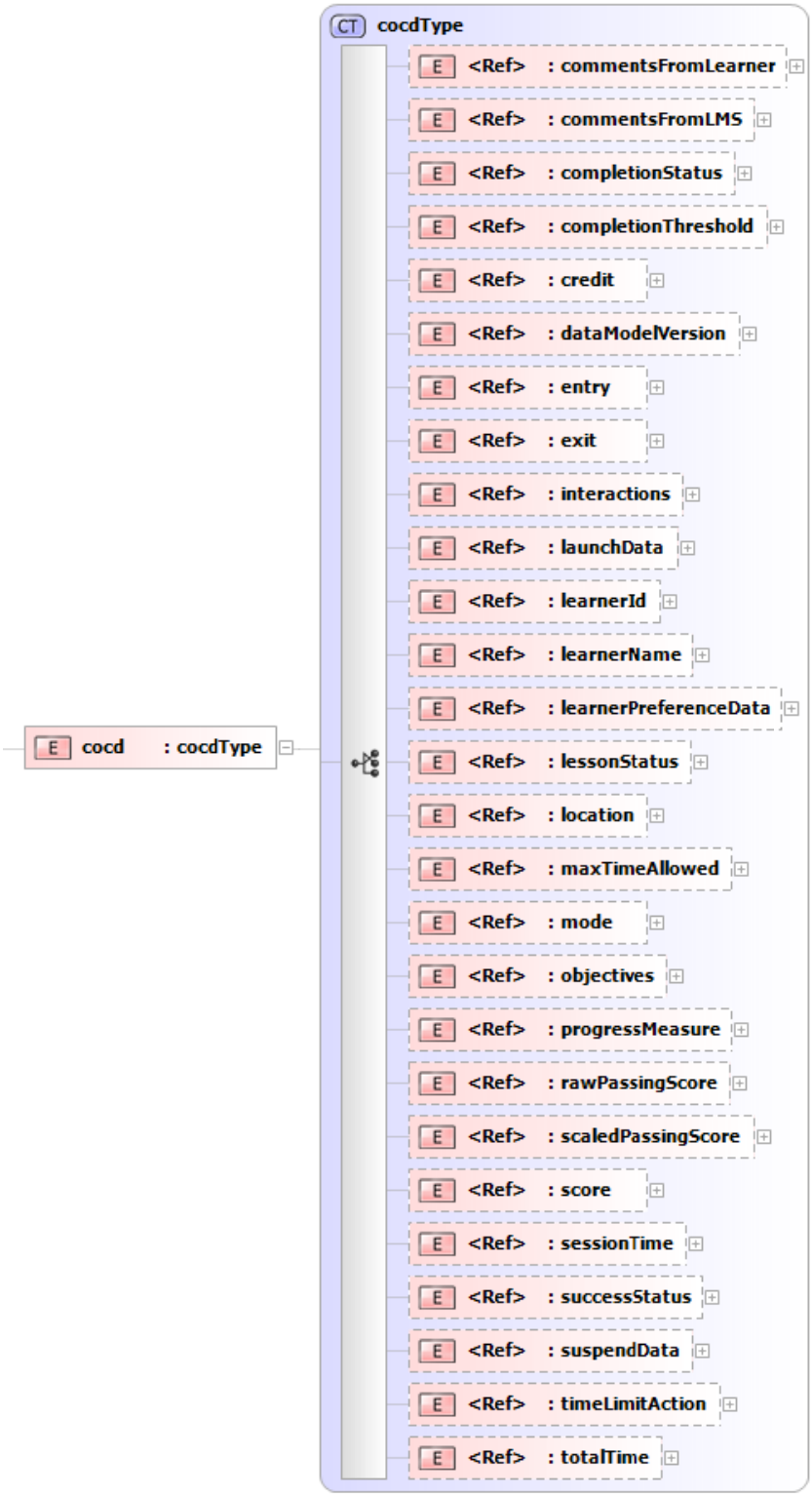
A.8. Delivery Controls

Data Element	Description	Default Value
Tracked	Indicates that the *Objective Progress Information* and *Activity/Attempt Progress Information* for the attempt should be recorded (True or False) and the data will contribute to the rollup for its parent activity, unless other sequencing information prevents it.	Boolean: true
Completion Set by Content	Indicates whether the *Attempt Completion Status* for the activity will be set by the activity's associated content object. This means that the content object associated with the activity is responsible for communicating whether or not the activity is complete.	Boolean: false
Objective Set by Content	Indicates whether the *Objective Satisfied Status* for the activity's associated objective, that has the *Objective Contributes To Rollup* value of True, will be set by the activity's associated content object. This means that the associated content object is responsible for communicating whether or not the activity's rolled-up objective is satisfied.	Boolean: false

Appendix B

SCORM RunTime Environment Data Model

This data model is used for the communication of the Sharable Content Object (SCO) with the LMS. The data set is described in the ieee_1484.11.3-2005.xsd schema and contains the following elements.



Appendix C

Example of a Resource Property Document


```
<ns1:node xmlns:ns1="http://test.wsrf.resource.manipulation">
        <ns1:item identifier="ITEM-C2D1FAF9-FF04-07BD-9A28-CDC8DC6EC0B8"
                identifierref="RES-65C9FBA6-5173-A9A8-C1A0-FE9EDFAB4788" isvisible="true">
        <ns1:title xsi:type="xsd:string">Introduction</ns1:title>
        <ns1:myEPR>
                <wsa:Address>http://192.168.1.103:9083/wsrf/services/LOManipulationService</wsa:Address>
                <wsa:ReferenceParameters>
                        <ns1:LOKey xmlns:ns1="http://lomanipulation.wsrf.master.karmazi/lo">20778986</ns1:LOKey>
                </wsa:ReferenceParameters>
        </ns1:myEPR>
        <ns1:parentEPR>
                <wsa:Address>http://192.168.1.103:9082/wsrf/services/LOManipulationService</wsa:Address>
                <wsa:ReferenceParameters>
                        <ns1:LOKey xmlns:ns1="http://lomanipulation.wsrf.master.karmazi/lo">18734604</ns1:LOKey>
                </wsa:ReferenceParameters>
        </ns1:parentEPR>
    </ns1:item>
</ns1:node>
<ns2:manifestBase xmlns:ns2="http://test.wsrf.resource.manipulation"/>
<ns3:resource href="Introduction/Introduction.html" identifier="RES-65C9FBA6-5173-A9A8-C1A0-FE9EDFAB4788"
                type="webcontent" xmlns:ns3="http://test.wsrf.resource.manipulation">
        <ns3:file href="Introduction/Introduction.html" xsi:type="ns3:fileType"/>
</ns3:resource>
<ns4:scormNavExt xmlns:ns4="http://test.wsrf.resource.manipulation">
    <adlnav:presentation xmlns:adlnav="http://www.adlnet.org/xsd/adlnav_v1p3">
                <adlnav:navigationInterface>
                        <adlnav:hideLMSUI>continue</adlnav:hideLMSUI>
                        <adlnav:hideLMSUI>previous</adlnav:hideLMSUI>
                </adlnav:navigationInterface>
    </adlnav:presentation>
</ns4:scormNavExt>
<imsss:sequencing xmlns:imsss="http://www.imsglobal.org/xsd/imsss">
        <imsss:controlMode choice="true" choiceExit="true" flow="true" forwardOnly="false"
                useCurrentAttemptObjectiveInfo="true" useCurrentAttemptProgressInfo="true"/>
</imsss:sequencing>
<ns5:ActivityStateInformation xmlns:ns5="http://test.wsrf.resource.manipulation">
        <ns5:ActivityIsActive>false</ns5:ActivityIsActive>
        <ns5:ActivityIsSuspended>false</ns5:ActivityIsSuspended>
</ns5:ActivityStateInformation>
<ns6:ActivityProgressInformation xmlns:ns6="http://test.wsrf.resource.manipulation">
        <ns6:ActivityProgressStatus>true</ns6:ActivityProgressStatus>
        <ns6:ActivityAbsoluteDuration>PT0S</ns6:ActivityAbsoluteDuration>
        <ns6:ActivityExperiencedDuration>PT0S</ns6:ActivityExperiencedDuration>
        <ns6:ActivityAttemptCount>1</ns6:ActivityAttemptCount>
</ns6:ActivityProgressInformation>
<ns7:CMIGlobalState xmlns:ns7="http://test.wsrf.resource.manipulation">
        <ns7:initializedState>true</ns7:initializedState>
        <ns7:terminatedState>false</ns7:terminatedState>
        <ns7:error>
                <ns7:errorCode>0</errorCode>
                <ns7:errorDescription/>
                <ns7:diagnosticErrorDescription/>
        </ns7:error>
        <ns8:cocd xmlns:ns8="http://ltsc.ieee.org/xsd/1484_11_3">
                <ns8:completionStatus>completed</ns8:completionStatus>
                <ns8:entry>ab_initio</ns8:entry>
                <ns8:exit>suspend</ns8:exit>
        </ns8:cocd>
</ns7:CMIGlobalState>
```

References

1. ADL (2004). Advanced Distributed Learning Initiative. Retrieved December 20th, 2008 from: http://www .adlnet. gov/
2. AICC(2004). CMI Guidelines for Interoperability Revision Version 4.0. Retrieved from http://www.aicc.org/docs/tech/cmi001v4.pdf
3. ARIADNE (2004). Alliance of Remote Instructional Authoring and Distribution Networks for Europe. Retrieved December 20th. 2008 from:
4. Aroyo L., Pokraev S, & Brussee, R. Preparing SCORM for the Semantic Web, Proceedings "On the move to meaningful Internet systems 2003 : Coopis, DOA, and ODBASE, Catania Sicily, Italy, November 2003". LNCS 2888 p.621- 628
5. Campbell,L. (2002). Introduction to Learning Technology Interoperability Standards and CETIS. Centre for Educational Technology Interoperability Standards, SURF Education Days. The Hague.
6. Chu, C.P., Chang, C.P., Yeh C.W., Yeh Y.F., A Web-service oriented framework for building SCORM compatible learning management systems, *Proceedings of International Conference on Information Technology: Coding and Computing*, (2004), 156 - 161 Vol.1
7. Costagliola G, Ferrucci F, Fuccella V, SCORM Run-Time Environment As a Service, Proceedings of the 6th international conference on Web engineering, Palo Alto, California, USA, p. 103 - 110
8. Dolog, P., Henze, N., Nejdl, W., & Sintek, M. (2004). Personalization in Distributed eLearning Environments. *Proceedings of the International WWW'04 Conference*, New York: ACM Press, 170-179.
9. EOE (2000). Educational Objects Economy website. http://www.natomagroup.com/eoe.html
10. Heins, T. and Himes, F. (2002). Creating Learning Objects With Macromedia, Flash MX. Macromedia White Paper.. April 2002.
11. Heng, S. (2003). Learning Objects – Where Next? In the Proceedings of the 3rd Annual CM316 Conference on Multimedia Systems. 2003.
12. Hodgins, H.W. (1994). The future of learning objects. In D.A. Wiley (Ed.). The Instructional use of Learning Objects. Available on http://reusability.org/read/chapters/hodgins.doc
13. IEEE PAPI, (2002). IEEE Public And Private Information. Available on jtc1sc36.org/doc/36N0186.pdf
14. IMS (2003). IMS Global Consortium: Overview of Specifications
15. IMS CP (2004). IMS Content Packaging Information Model Version 1.1.4. Retrieved December 20th, 2008 from: http://www.imsglobal.org/content/packaging/.
16. IMS DRI (2003). IMS Digital Repositories Interoperabilty Specification Version 1.0. Retrieved from http://www.imsglobal.org/digitalrepositories/
17. IMS SS (2003). IMS Simple Sequencing Specification. Retrieved December 20th, 2008 from: http://www.imsglobal.org/simplesequencing/index.html
18. Kılınç, N. (2006). Learning web modeling language using Moodle, Unpublished Master Thesis, Işık University.
19. Kleijnen, S., Raju. S. (2003), An open web services architecture. Queue, vol.1, no. 6, 38-46
20. L'Allier, J. J. (1998). c: Retrieved December 20th, 2008 from http://www.netg.com/ research/pskillpaper.htm
21. Learning Management System, http://www.learningcircuits.org/NR/exeres/72E3F68C-4047-4379-8454-2B88C9D38FC5.htm
22. LOM (2002). Draft Standard for Learning Object Metadata. Retrieved December 20th, 2008 from http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf
23. Longmire, W. (2000). A Primer on Learning Objects. ASTD Learning Circuits, March 2000. Retrieved December 20th, 2008 from: http://www.astd.org/LC/2000/0300_longmire.htm
24. LTSC (2000). Learning Technology Standards Committee web site at http://ieeeltsc.org/
25. Millar, G. 2002. Learning Objects 101: A Primer for Neophytes. Sidebars. A Publication of the Learning Resources Unit of the British Columbia Institute of Technology.

26. Mohan, P., and Greer, J. (2003). Using Learning Object Technology to Tackle Education Challenges of the Carribean
27. Newcomer, E.,& Lomow, G. (2005). Understanding SOA with Web Services. Addison-Wesley, 2005
28. OASIS Web Services Business Process Execution Language Version 2.0. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.html
29. OASIS Web Services Resource Framework. http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-01.pdf
30. Pahl, C., Barrett, R.. (2004). A web services architecture for learning object discovery and assembly, *Proceedings of the 13th int. World Wide Web conference on Alternate track papers* (May 2004), 446-447
31. Paulsen, M. F. (2002). An Analysis of Online Education and Learning Management Systems in the Nordic Countries.. Online Journal of Distance Learning Administration, V, III.
32. Polsani, P., R. 2003. Use and Abuse of Reusable Learning Objects. Journal of Digital Information, Volume 3, Issue 4, Article No.164. Retrieved December 20th, 2008 from http://jodi.tamu.edu/Articles/v03/i04/Polsani/
33. Qi Y, Xumin Liu and Bouguettaya A., and Medjahed B. . Deploying and managing Web services: issues, solutions and directions. The VLDB Journal, Springer. (pp 537-572).
34. Rehak**,** D. and Mason, R., (2003). Keeping the Learning in Learning Objects: A Sustainable Approach to eLearning. Allison LittleJohn. Kogan Page. London.
35. Robak, S., & Franczyk, B.(2003). Modeling Web Services Variability with Feature Diagrams. In *Web, Web-Services, and Database Systems*, (pp 120-128). SpringerLink.
36. SCORM (2004). Sharable Content Object Reference Model 2004. Available on http://www.adlnet.gov/downloads/DownloadPage.aspx?ID=237
37. Shih, T.K., Chang, W. C., Lin, N.H., Lin, L.H., Hsu, H. H., Hsieh, C. T., Using SOAP and .NET web service to build SCORM RTE and LMS, *Proceedings of Advanced Information Networking and Applications* (Xi'an, China, Mar 2003), 408-413
38. Tamura Y., Yamamuro, T., Distributed and Learner Adaptive e-Learning Environment with Use of Web Services, *Proceedings of the International Conference on SCORM 2004*(Taipei, Taiwan, Jan 2006), 11-15
39. Vaughan-Nichols, S.J. (2002). Web services: beyond the hype. IEEE Comput. **35**(2), 18–21.
40. Vossen, G., Westerkamp, P. (2003). E-learning as a Web service, *Proceedings of Seventh International Database Engineering and Applications Symposium*, (July 2003), 242 – 249.
41. Warwick, B. (2005). What Is…IMS Simple Sequencing. Retrieved at http://www.icodeon.com/pdf/ss2brief.pdf.
42. Wiley, D., A. (2000). Connecting Learning Objects to Instructional Design Theory: A Definition, A Metaphor and A Taxonomy. Retrieved December 20th, 2008 from: www.reusability.org/read/chapters/wiley.doc
43. W3C: Simple Object Access Protocol (SOAP). (2003) . http://www.w3.org/TR/SOAP/
44. W3C: Universal Description, Discovery, and Integration (UDDI). (2003). http://www.uddi.org
45. W3C: Web ServicesArchitecture. (2003). http://www.w3.org/TR/ws-arch /
46. W3C: Web Services Description Language (WSDL).(2003). http://www.w3.org/TR/wsdl
47. *Web Services Addressing 1.0 - Core***,** M. Gudgin, M. Hadley, W3C Working Draft, 31 March 2005, available at http://www.w3.org/TR/2005/WD-ws-addr-core-20050331/
48. *Web Services Base Notification 1.3 (WS-BaseNotification),* S. Graham, D. Hull, B. Murray , OASIS Standard, 2006, available at http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf
49. *Web Services Base Faults 1.2 (WS-BaseFaults)* L. Liu, S. Meder (editors), OASIS Standard, 2006, available http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf
50. *Web Services Resource 1.2 (WS-Resource).* S. Graham, A. Karmarkar, J. Mischkinsky, I. Robinson, I. Sedukhin (editors), OASIS Standard, 2006, available at http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf

51. *Web Services Resource Lifetime 1.2 (WS-ResourceLifetime).* L. Srinivasan, T. Banks (editors), OASIS Public Standard, 2006, available http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-spec-os.pdf
52. *Web Services Resource Properties 1.2 (WS-ResourceProperties).* S. Graham, J. Treadwell (editors). OASIS Standard, 2006, available http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-os.pdf
53. *Web Services Service Group 1.2 (WS-ServiceGroup).* T. Maguire, D. Snelling (editors). OASIS Standard, 2006, available at http://docs.oasis-open.org/wsrf/wsrf-ws_service_group-1.2-spec-os.pdf
54. *Web Services Topics 1.3 (WS-Topics)* W. Vambenepe (editor). OASIS Standard, 2006, available at http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.pdf
55. *XML Schema part 1: Structures* . H. Thompson, D. Beech, M. Maloney, N. Mendelsohn .W3C Recommendation, 2001, available at http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/
56. Xiaofei L., El Saddik, A., Georganas, N.D., 2003. An implementable architecture of an e-learning system, *Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering* (May 2003), 717 - 720 vol.2