International Jl. on E-Learning (2008) **7***(2),* 219-243

Towards an Object-Oriented Model for the Design and Development of Learning Objects

CHRYSOSTOMOS CHRYSOSTOMOU AND GEORGE PAPADOPOULOS
University of Cyprus, Cyprus
chrisc@ucy.ac.cy
george@cs.ucy.ac.cy

This work introduces the concept of an Object-Oriented Learning Object (OOLO) that is developed in a manner similar to the one that software objects are developed through Object-Oriented Software Engineering (OO SWE) techniques. In order to make the application of the OOLO feasible and efficient, an OOLO model needs to be developed based on Object-Oriented (OO) concepts. A sample OOLO model of inheritance is used to demonstrate the application of such Learning Objects (LOs) to a learning scenario. The benefits of this model are quantified in terms of savings on the number of new LOs that need to be developed and savings on metadata entry. Additionally, benefits extend to the quality of new LOs due to the automatic application of standards and the increased compatibility among LOs (due to inheritance) and the increased extendibility and functionality of LOs due to the OO characteristic of method and data encapsulation and polymorphism, which also make OOLOs more adaptable to the varied teaching styles. The study goes on to examine existing LO design and development models as well relevant tools and assesses the ability of these models and tools to implement the OOLO concept. The study concludes by summing up the benefits that can be realized by the development of OOLOs and by outlining the work that needs to be done for achieving the application of OO techniques to LOs.

After e-learning had become popular through the use of electronic courses delivered through the Web, the researchers in the area were concerned with the inflexibility of those courses, as they were very time consuming, costly to build, and they were not reusable. The researchers have shown that the most

efficient way to deal with this problem was to break the courses down into smaller self-contained modules that could be archived and reused whenever they were needed. In this way instead of creating everything from scratch every time a new course was needed, the e-learning developer could just select a number of existing course modules, add some new ones if necessary, and assemble them into a new course. Probably influenced from their counter-parts in Object-Oriented Software Engineering (OO SWE) – the software objects – these course modules were called Learning Objects (LOs). Although LOs form an attempt to offer to e-learning the benefits that software objects offered to SWE (i.e., reusability, extensibility, efficiency, etc.), it is argued that LOs are not truly Object-Oriented (OO) as they lack important characteristics of OO theory such as inheritance, abstraction, polymorphism, and so forth (Sosteric & Hesemeier, 2002), and consequently they cannot offer the same benefits. A number of authors have supported the idea of applying to Los, design techniques that have so far been applied to SWE. Poulton (2005) for example stated: "Applying software engineering principles in the development of learning objects will bring the benefits of proven software design methods for reusability into the field of education content." Douglas (2001) argues: "Object-oriented software engineering is proposed as a useful basis for new thinking in instructional design methodology." Retalis (2003) commenting on Rehak and Brooks mentioned: "The authors suggest that we need better descriptive models such as CLEO [Customized Learning Experiences Online] or educational modeling languages such as EML. Not only that. We also need better design models for the authoring/aggregation of learning content."

This article follows an introductory work on recent e-learning developments and trends (Chrysostomou & Papadopoulos, 2005), and aims to lay down the root towards an OO model for LO design and development. In one section there is an analysis of the concept of an LO, followed by a section on an analysis of how OO concepts may link to LOs. In the next section there is a reference to the technical and cultural difficulties of using LOs and an outline of ways in which these difficulties may be overcome by applying OO techniques. In another section a learning scenario that requires the application of LOs is described and then the processes of applying both traditional LOs and OOLOs to that scenario are explained. A sample LO inheritance model is used to demonstrate the application of the OOLOs to the scenario. The benefits of applying OOLOs are quantified in terms of effort savings in the creation of new LOs, mostly offered by the inheritance of LO structures and metadata. In the next section, the requirements for an OOLO design and development model and a relevant tool are outlined, followed by a comparative evaluation of existing LO models and tools against those features. The article concludes by summarizing the main benefits that can be offered by applying OO techniques to LOs, and the work that needs to follow in order to implement an appropriate OOLO model.

THE DEFINITION OF A LEARNING OBJECT (LO)

The Institute of Electrical and Electronics Engineers (IEEE) Learning Technology Standards Committee (LTSC) has defined a Learning Object as "…an entity, digital or nondigital, that may be used for learning, education or training" (IEEE LTSC, n.d.). This definition, although it gives a good understanding of the idea of an LO, it leaves quite a few details undefined allowing disagreements on a number of matters regarding LOs. Such matters include mainly the granularity (the size of an LO) and its structure (the components that make up an LO).

A collective attempt to better define different aspects regarding LOs, such as their granularity and structure, has been done by a number of standards organizations (IEEE, Instructional Management Systems [IMS], Aircraft Industry Computer-Based Training Cosortium [AICC]) that have joined forces with the initiative of the Advanced Distributed Learning consortium (ADL) and have created a library of such standards called the Sharable Content Object Reference Model, widely known as SCORM. In one of the SCORM sections (or books), the SCORM Content Aggregation Model (CAM), issues of LO granularity and structure are dealt with. More specifically the SCORM Content Model defines the components that make up SCORM conformant learning resources and the way in which these components are aggregated. There are five aggregation levels within the SCORM content model: asset, Sharable Content Object (SCO), activity, content organisation, and content aggregation. An asset is the most basic form of a learning resource (text file, image, video, audio, html file, animation, etc.). An SCO is a collection of one or more assets. An SCO is the lowest level of granularity of a learning resource that can be tracked by a Learning Management System. An SCO is the SCORM equivalent of an LO. Assets and SCOs make up the learning resources. Learning resources link to structured units of instruction called *activities*. Activities (shown in Figure 1 as "items") may be nested within other activities (only the lower level activities link to resources). A set of activities makes up the *content organisation*, and the whole pack of activities and resources is called the *content aggregation*.

SCORM also defines the way in which learning content should be packaged through the Content Packaging specification. This specification aims in providing a standardised way to exchange learning content between the authoring tools, learning content repositories and learning content management systems. A Content Package (Figure 2) contains two major components, the manifest (an XML document describing the content structure and associated resources of the package including metadata) and the physical files (the learning content).

One of the most important parts of an LO is its metadata. Metadata is the information that accompany the content, according to SCORM, it is part of

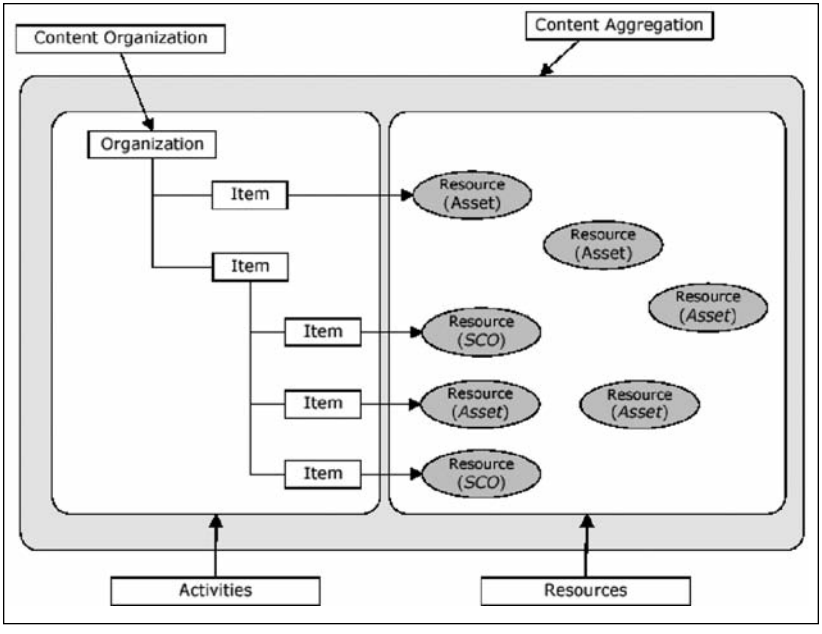

Figure 1. The different aggregation levels of content components and the relations between them as represented in the SCORM CAM (figure from ADL SCORM)

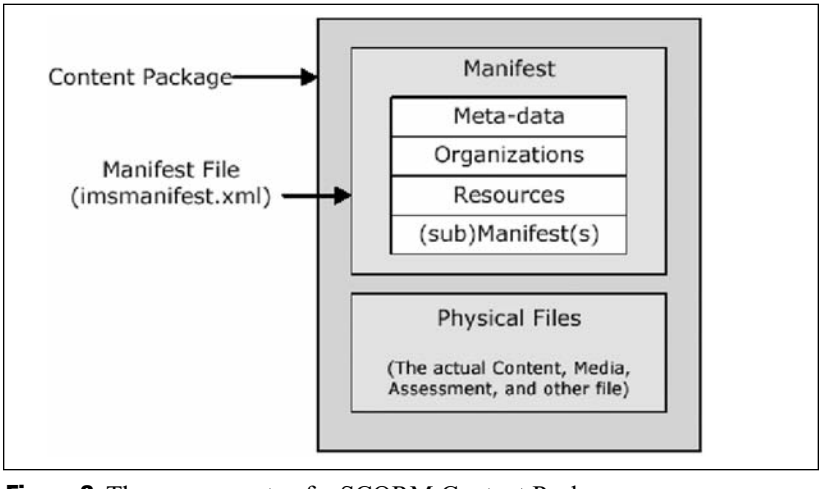

Figure 2. The components of a SCORM Content Package (figure from ADL SCORM)

the manifest file (Figure 2), and it enables interaction with an LO by supplying information about it (such as subject, date, length, learning objective, description, type, language, etc.). The specification of LO metadata has been standardised by the IEEE Learning Technology Standards Committee (LTSC) in 2002 through the Learning Object Metadata (LOM) standard (IEEE standard 1484.12.1; IEEE LTSC LOM, n.d.). The LOM standard (also part of SCORM) defines the data that could be used to fully describe each LO and groups it into nine categories, which group data into a hierarchy of data elements. Data elements may be *aggregate data elements* or *simple data elements*. Aggregate data elements are basically subcategories that include more data elements. Simple data elements are the lowest level data elements (the leaf nodes of the hierarchy). Each simple data element is described by a unique identifier, a name, an explanation, its size, order (whether the order is significant or not), its value space, its datatype, and an example. Figure 3 illustrates part of the LOM Base Schema. Data element 1 defines the high level category "General." Data element 1.1 is an aggregate data element. Data elements 1.1.1 and 1.1.2 are simple data elements for which value space, datatype, and an example are defined. The complete set

Nr	Name	Explanation	Size	Order	Value space	Datatype	Example
1	General	This category groups the general information that describes this learning object as a whole	1	Unspecified	-	-	-
1.1	Identifier	A globally unique label that identifies this learning object	Smallest permitted maximum: 10 items	Unspecified	-	-	-
1.1.1	Catalog	The name or designator of the identification or cataloging scheme for this entry. A namespace	1	Unspecified	Repertoire of SO/IEC 10646-1:2000	CharacterString (smallest permitted maximum: 1000 char)	"ISBN," "ARIADNE," "URI"
1.1.2	Entry	The value of the identifier within the identification scheme that designates or identifies this learning object. A namespace specific string.	1	Unspecified	Repertoire of SO/IEC 10646-1:2000	CharacterString (smallest permitted maximum: 1000 char)	"2-7342-0318," "LEA0875," "http://www.iee.org/documents/1234"

Figure 3. Part of the LOM v1.0 base schema

of data elements can be found in the draft LOM standard available online from the IEEE (IEEE LTSC LOM, n.d.).

Summing up the analysis regarding the nature of LOs we can try and simplify the definition of an LO: An LO is a set of learning resources (raw learning content such as text files, images, video files, audio files, etc.), accompanied by a set of metadata that satisfies a single learning objective and can be used and reused in e-learning.

OBJECT-ORIENTATION AND LEARNING OBJECTS

LOs are self-contained chunks of learning content that can be reused in a variety of learning contexts. This probably reminds you of the OO paradigm for software development, where self-contained programs (software objects) can be reused in a variety of software applications. An LO can ideally be connected to other LOs to form larger learning content, similarly to how a software object can be linked to other software objects to form larger applications. This raises the question whether it would be possible to apply to LOs techniques that have so far been successfully applied to OO SWE. A number of studies have been concerned with this matter in the past. Following is a brief summary of the prevailing ideas that have surfaced through those studies and an analysis of how the OO paradigm can be applied to LOs.

Robson (1999) viewed learning resources (content, assessments, references, activities, etc.) as objects in an OO model that have methods (such as rendering and assessment) and properties (such as content and relationships to other resources). He also identified the drawbacks of existing web-authoring techniques for the creation of online courses as being: time, cost, inflexibility, and interdependency of content with design.

Downes (2001) identified the same inefficiencies as Robson and proposed the use of SWE techniques for the development of learning objects. The aim is to avoid creating all content from scratch every time it is needed and allow for content to be applied to larger audiences. The author concentrates on the method that OO design uses to construct *object prototypes*, referring basically to the idea of constructing a class (prototype) acting as a template from which objects may be created when needed with hardly any effort. He also pays great attention to the concept of *inheritance* through which new learning objects may be created from existing similar ones by extending their properties (adding more functionality to them).

Friesen (2003) identified the parallels between software objects and LOs and goes on to discuss another aspect of Object Oriented Programming (OOP) that could be reflected on learning objects, that of the "*black box*," according to which the implementation of an object should be hidden from its users. Interactions with the users should only be available through the objects interfaces (methods) that will control how a user can customize the object.

Polsani (2003) identified reusability as the major functional requirement of LOs and suggests that LOs should be created with a high level of abstraction, as this will provide independence from usage scenarios and the ability of the LO to join other LOs in a variety of contexts. The article concludes by suggesting as immediate necessities the commonly accepted, accurate, and functionally effective definition of an LO and the reengineering of the design and development process of LOs in a "multidisciplinary and cooperative model of development to create knowledge that is appropriate for the emergent network society."

Permanand and Brooks (2003) suggested the notion of the "Object-Oriented Learning Object" (OOLO). According to the authors, there should be an LO super class from which all LOs should derive. The LOs should:

- have *properties to metadata* instances;
- have *properties to other objects* (e.g., version object, context object, combination object, etc.);
- have *methods* (e.g., query methods, version control methods, insertion, deletion, etc.);
- support *aggregation* relationships to allow hierarchies of LOs to be created out of simple LOs;
- *support inheritance* for producing new LOs out of existing ones; and
- not have "*using*" *relationships* because they reduce reusability due to coupling between LOs.

Permanand and Brooks (2003) concluded by expressing their beliefs on the future of the OOLO: "We believe that the object-oriented approach can go a long way towards achieving the vision currently being promoted for learning objects. Learning objects with object-oriented features provide a solid foundation for the effective reuse of learning resources on the Web."

Morris (2005) also supported the idea of LOs being developed as classes in an OO environment. The author stresses the importance of inheritance, which enables new LO classes to be developed by extending existing ones and polymorphism that enables new LO classes to change the way in which they implement the methods of the classes they inherit from.

Having studied the work of the authors that have been involved with the idea of applying OO techniques to the design and development of LOs we can draw some parallels between the OO paradigm of SWE and the application of this paradigm to the development of Los:

- in OO SWE, software is created as *classes* that include attributes (data) and methods (operations that can be performed on the data). Those classes serve as templates for creating Software Objects. In a similar manner LO classes can be created to include attributes (content, metadata, etc.) and methods (query, insertion, deletion, etc.). These classes

can serve as templates from which LOs can be created. Each new LO will include all the attributes and methods that are included in the relevant class;

- new LO classes should have the ability to be created from existing ones by applying rules of *inheritance* to avoid duplicated effort when the required LO is similar to an existing LO;
- *aggregation* could also be applied when the desired class can be created by a number of instances of other classes (for example a "Test" class can be created by aggregating a number of instances of the "Exercise" class);
- LO classes should also demonstrate *polymorphic* behavior. A child class should be able to override and implement the methods of its parent class in a way that will better serve its purposes. This could enable LO classes to easily adapt to different environments and also better serve the varied teaching styles of each individual instructor; and
- finally LO classes should be created with a high level of *abstraction* in order to hide their implementation details and make their creation and usage feasible and simple to users that may not have extended technical knowledge.

An LO that demonstrates the characteristics could be considered as an OOLO. In order for this OOLO to really take advantage of its OO characteristics however, it must form part of a larger hierarchy of OOLOs. The idea is to create an LO superclass from which a hierarchy of LOs can be created. A basic hierarchy of LO classes should exist that includes LO templates for most of the common LO types. From those templates new LOs can be instantiated and new templates can be created by applying rules of inheritance.

THE TECHNICAL AND CULTURAL DIFFICULTIES OF USING LOS

In the development of LOs a number of standards need to be applied (SCORM) and a large number of metadata (LOM) needs to be entered. Applying all the SCORM and LOM requirements is quite time consuming and requires a great deal of knowledge on behalf of the developer. Modern LO authoring tools aim in minimizing the knowledge and effort required to build LOs, by automating processes such as XML document generation. In the current LO research agenda, subjects such as automatic metadata generation and the development of LO design techniques and tools exist that are expected to minimize time, effort, and cost of developing LOs by further automating processes and by more efficiently reusing existing LOs. Applying techniques (i.e., OO) that have already been applied elsewhere (i.e., SWE) with proven positive outcomes is expected to help in further overcoming current problems in LO development.

In addition to the technical problems that the usage of LOs involves there

are cultural problems as well, that make the application of LOs unappealing to a number of members of the academic community. Such problems mainly involve the individual preferences of instructors that have a tendency to avoid using learning content created by others, as it may no suit their preferred teaching style.

So, exactly what are the problems that currently exist in LO development? What follows is a representative, however not exhaustive, list of problems that have been identified through a literature review and personal study of LOs:

- creating LOs requires knowledge of LO theory, standards (e.g., SCORM, LOM) and sometimes even markup languages (XML), although some authoring tools make this process easier;
- following the existing model for LOs (i.e., SCORM Content Aggregation Model), each LO has to be created from scratch or existing LOs have to be modified to produce the desired content;
- for each new LO, a large number of LOM has to be entered manually (automatic metadata generation is still in its infancy);
- similar LOs are often created from scratch causing unnecessary waste of time, money, effort, and possible risk of incompatibilities;
- each new LO is untested leading to unreliability of new LOs;
- LOs are often bound to specific context and cannot be easily reused;
- developers often find the internal structure of an LO confusing especially when they are not very familiar with the related technologies (e.g., XML); and
- pre-written learning content usually follows a specific form that does not suit each individual's teaching style making them unwilling to use it.

Summing up, the LO development process still requires extensive knowledge, effort, time, cost, and often causes reusability, interoperability, and reliability problems. The usage of LO is often problematic as well, due to the dependency on specific learning styles, that usually come in conflict with instructors' individual preferences. For once again, if we compare these problems with problems that the SWE area faced in the past we can see the analogy. It should be possible to solve these problems of LO development by applying the lessons we have learned from applying object-orientation to software development.

Following is an analysis of ways in which OO techniques can help in minimizing the problems of the LO development and usage:

- By following a hierarchical model of inheritance in developing LOs, a number of abstract LOs (LO templates) will exist through which new LOs will be created. In this way:

- new LOs can be developed faster, easier, and with less cost by extending existing LOs;
- new LOs will inherit the properties of their predecessors including metadata (minimizing metadata input);
- new LOs will be based on existing-tested-ones (more reliable LOs);
- existing LOs will already follow the appropriate standards and consequently the developer will not need to deal with ensuring the application of standards;
- all new LOs will be created following a common (standardized) structure and they will consequently display increased interoperability;
- abstract LOs will not be bound to any specific context or design and hence be highly reusable;
- the developer does not have to know the internal structures of the LOs (less knowledge needed to develop LOs);
- the hierarchy can be extended in a way that best suits specific domains or organizations (i.e., new more specialized abstract LOs can be created to create more specialized LOs);
- LO can be better maintained (e.g., easier modification of LOs by modifying the LO class they inherit from); and
- LOs can easily be aggregated to form larger learning contents, which can also be reused when necessary.

• By creating LOs in an abstract, polymorphic, and context independent way:

- LOs will have the ability to be used in a variety of contexts;
- instructors will be able to use the LOs in a variety of teaching styles or apply to them their own preferred style; and
- LOs will become more appealing to instructors and consequently to learners.

In the previous paragraphs we have argued in favor of the OOLO approach and outlined a number of benefits that this approach may have to offer against traditional LO development. It is probably not easy and maybe not even feasible to test and quantify with accuracy these benefits without actually implementing, using, and assessing the OO model for LOs. However, in support of these arguments, it would be useful to examine how the OOLO approach would apply to a real-life learning scenario. In the section that follows, a traditional learning scenario in higher education is described and then the work that needs to be done to transform this class based course into an e-learning course by making use of LOs is analyzed. This analysis is done first for the case where traditional (non OO) LOs are used and then for the case that an OOLO model is applied.

A COMPARISON OF THE APPLICATION OF LOS AND OOLOS TO A LEARNING SCENARIO

The Learning Scenario

A lecturer is teaching an Information Systems (IS) class based course. The course plan in brief includes the following:

- two hours of lecture every week (for the lectures power point presentations are used);
- one hour of seminar work every week, during which students are involved in conversations around issues relevant to the week's subject, answer questions, work with case studies, solve exercises, or carry out other tasks that are relevant to each week's subject. Materials for the seminars are given to students in printed MS Word format;
- towards the middle of the semester the lecturer carries out a revision session by supplying the students with a printed document (MS Word) that includes a set of revision questions;
- following the revision is a midterm exam;
- after the midterm exam classes continue as normal, followed by another review session just before the final exam; and
- during the term the lecturer also hands out an assignment that the students should solve and return by a due date.

The lecturer has been assigned the task of adapting the above course for web delivery and consequently decides to break the course down into small, self-contained, reusable LOs. An LO authoring tool is available for use. The main tasks that need to be performed in brief include:

- the disassembling of all material into self standing single learning objective units;
- the development of LOs by creating appropriate XML packages;
- the entry of appropriate metadata for each LO; and
- the aggregation of LOs to form larger learning units.

Following (Figure 4) is the detailed course outline for the course.

Following is an analysis of the number and types of LOs that will be needed:

- from the breakdown of the course outline in Figure 4, there is an average of approximately eight individual learning objectives in each week's lecture. Each of the learning objectives would require a short power-point presentation for assisting with demonstrating the main points (approximately 100 single objective presentations will be needed); many of the objectives also require graphical explanations such as pictures, diagrams, and so forth, that will form separate resources (assets);

Week	Subject	Description
1	What is IS?	Why IS? What is an IS? The business perspective of IS: Organization, Management and Technology. The socio-technical approach to IS. The role of IS.
2	Types of IS	Major types of IS in organizations: Transaction Processing, Knowledge Work and Office Systems, MIS, DSS, ESS. The functional perspective of IS: Sales and Marketing, Manufacturing and Production, Finance and Accounting, Human Resources. Enterprise applications.
3	Organisations and IS	What is an organization? Organization features. The role of IS in organizations: IT infrastructure, Effect of IS on organizations. Managers and decision-making: the role of managers in organizations, decision making. Information Systems and Business Strategy.
4	Hardware and Software	Hardware: The computer system, computer processing, storage, input and output. Categories of computer systems. Software: types of software, OS, Programming languages, application and productivity software. Managing hardware and software assets.
5	Data management	Managing data resources. The file environment: terms and concepts, problems with the file environment. The database approach: Database Management Systems (DBMS), Types of databases. Basic SQL. Designing Databases. **Hand out assignment.**
6		***REVISION***
7		**MIDTERM EXAM**
8	Development of IS	Linking IS to the Business Plan, Organizational information requirements. Organizational Change. Business Process Re-engineering. Total Quality Management. Systems development process. Alternative approaches to systems development.
9	Management of IS	Business value of IS. Management of change for IS success. Management of IS implementation – Critical Success Factors (CSF).
10	Security of IS	Systems vulnerability. Concerns for system builders and users. System quality problems. Creating a control environment. Systems security. Ensuring system quality.
11	Knowledge Management	Knowledge management. Information and Knowledge work systems. Artificial Intelligence. Expert systems. Neural Networks. Decision Support Systems.
12	Networking	Telecommunication systems. Communication Networks. E-commerce & E-Business. The Internet. The WWW. Support technology for e-commerce and e-business. Management issues and decisions.
13		***REVISION***
14		**FINAL EXAM**

Figure 4. The course outline for an introductory information systems course

- in addition there must be at least one discussion subject per week (this could be in the form of a case study). A set of questions or other kind of tasks must also exist. We could assume that three to five tasks must exist for each week's seminar averaging to approximately 50 tasks;
- a set of tasks will be needed for the revision sessions (at least 10 per session);
- a number of tasks will also be needed to form the midterm and final exams (approximately 10); and
- additional assessment tasks are required for the assignment.

Applying LOs to the Learning Scenario

From the described analysis we can roughly estimate that approximately 200 individual learning assets will be needed to deliver and assess the course. Sharable Content Objects (SCOs or LOs) will be formed by individual assets or groups of assets. It can be estimated that more than 100 LOs will need to be created to satisfy the entire course's learning objectives and assessments. The LOs will be grouped into Activity Items to create larger learning and assessment units such as lectures, seminar sessions, revision sessions, assignment, and exams. Finally, Activity Items will be grouped into content organizations to form the complete course. Based on SCORM requirements each one of these items will require a set of metadata to be attached to it and packaged into an appropriate XML document. Following a traditional SCORM approach, the process would require the following tasks:

- breaking the content down into appropriate assets (resources such as single objective power-point presentations, graphics, questions, revision notes, case studies, tasks, etc.);
- attaching appropriate metadata to each asset (SCORM asset metadata);
- grouping assets into Sharable Content Objects (SCOs) and attaching appropriate metadata to them;
- grouping assets and SCOs into aggregated LOs (Activity Items) and attaching appropriate metadata to them (SCORM Activity Metadata);
- grouping Activity Items into a Content Organization attaching appropriate metadata to it (SCORM Content Organization metadata); and
- creating appropriate XML packages for these learning resources.

The most time-consuming and repetitive task involved in the process is probably the creation and embedding of the appropriate metadata within the XML documents for the different resources. According to the LOM standard there are 58 pieces of lower level metadata (simple data elements) grouped into nine top level categories and then further into other subcategories. Not all metadata are always necessary in describing an LO. The SCORM Content Aggregation Model specifies, which metadata is mandatory for the dif-

ferent types of resources. Specifically it specifies 8 out of the 58 lower level metadata as mandatory for assets and 11 for SCOs, activities, and content organizations.

Having approximately 200 assets produces a requirement for approximately 1600 pieces of metadata (if minimum metadata is used) and around 12000 pieces of metadata (if all possible metadata is used). Additional metadata will need to be attached to each one of a minimum of 100 SCOs (LOs). This will produce a requirement for at least an additional 1100 pieces of metadata (if minimum metadata is used) and 6000 pieces of metadata (if all possible metadata is used). Activity Items and Content Organizations will also need metadata, but for simplicity purposes we will not consider these as the number of activity items and content organizations in the course will not be large compared to the large number of assets and SCOs. In order to create all the necessary resources the lecturer has to import each asset into the LO authoring tool and manually enter the metadata for each one. Depending on the automation of the authoring tool the lecturer may also need to create a large number of XML packages for the LOs that make up the learning content.

Judging by the complexity of the XML packages described in SCORM CAM, the large amount of metadata prescribed in the LOM model and the large amount of assets and LOs needed to create learning resources, the task of creating each resource and attaching the appropriate metadata to it is highly repetitive, time consuming, tiring, and consequently error prone.

In addition to the complexity of creating the traditional LOs, the usage of them may also be quite problematic. It is expected that a number of functions will need to be applied on an LO (i.e., create, edit, delete, insert, remove, etc.). A traditional LO includes only content and metadata. The ways that it is used depends entirely on the developer/user and the runtime environment using the LO. An OOLO including the methods providing the necessary functionality to the LO, will make it more self-contained, abstract to the user, consistent and reusable, as we have very well learned from our experiences with the software objects.

Applying OOLOs to the Learning Scenario

In the previously described, scenario we have described the process followed to create an e-learning course by using a traditional LO approach. In this section, we assume the existence of an OO model for the design and development of LOs. This model should incorporate concepts similar to the ones used in OO SWE. Most important concepts being those of data and method encapsulation and inheritance. The Java language for example provides a hierarchical library of classes (known as the Application Programming Interface – API). At the top of the Java API hierarchy there is a class called "Object," this class includes functionality (data and methods) that is common to all Java classes. All Java classes inherit the functionality of the

Object class. Further down the hierarchy there are more classes that extend the "Object" class and provide additional functionality. In the case of LOs, there should be a top level "LearningObject" class that includes the functionality that is common to all LOs. Functionality in the case of LOs should include the appropriate metadata as well as functions that an LO may perform (create, edit, delete, insert, remove, etc.). More specialized classes of LOs should extend the "LearningObject" class by supplying additional functionality, suitable for the type of LO they describe. The idea is that the user may choose a suitable LO from this hierarchy and by using one of the LO's methods will add the appropriate content to it. The packaging and all common (for that type of object) metadata will already be included and the LO will be ready for use. If none of the LOs in the hierarchy is suitable, then the user should be able to select the one that is closest to their required object and extend it to add the missing functionality or metadata. As Poulton (2005) argued: "…the most reused feature of a learning object is its structure and design as opposed to its content…."

From the teaching and learning scenario described earlier we can extract a number of possible LO types including:

- presentation
- graphical representation
- question
- task
- case Study
- case study exercise
- assignment task
- revision question
- exam question

More LO types may exist, such as animation, essay question, multiple choice question, and so forth. Each one of these LO types could form an LO class from which LO "objects" may be created. Metadata and functionality will be inherited from the parent classes and the developer will only have to input a limited amount of metadata for these subclasses of LOs.

The process to be followed in creating LOs for the course described earlier will now be easier and faster since the XML packages for each type of object will exist or be created by extending existing ones and most of the metadata and functionality will be inherited from other LO classes instead of having to repeatedly input it for each LO. For example an LO class called "Question" may exist. The metadata that are common to all questions will be attached to that class. Whenever the developer needs to create a question LO they can use the existing class so that they will avoid inputting all common metadata and creating LO packages. A subclass may also exist or be

created (e.g., "CaseStudyQuestion") that will incorporate even more metadata that are common to all questions that relate to case studies. Figure 5 demonstrates part of the proposed hierarchy which shows an example of the LO classes that may exist that will enable the easy and fast creation of LOs that would satisfy the earlier described scenario.

To quantify the amount of work that will be saved using the OOLO approach, the LOM standard has been studied with the purpose of identifying the number of metadata that would be common between similar LOs. For example all "Question" objects will have the same format, location, platform requirements, interactivity type, learning resource type, intended end user role, language, purpose, and so forth. Furthermore, each "CaseStudyQuestion" (for the same case study) will have the same "relation" metadata (kind, catalog, entry, description) as all such questions will relate to the same LO (the specific case study). It has been estimated that 33 to 42 out of the 58 lower level metadata in LOM could be common between similar LOs due to their nature.

Based on this study and the metadata requirements as defined in SCORM, we can estimate the amount of metadata input that can be avoided by using the OOLO model. Table 1 summarizes the effort savings that could occur when using the OOLO model.

The tasks required to create the LOs for the web-based delivery of the course described earlier would now include:

- breaking the content down into appropriate assets;
- most of the LO (i.e., the XML packages) already exist for the different types of LOs (e.g., Presentation, CaseStudyQuestion, RevisionQuestion). These existing LOs can be used by attaching the appropriate assets to them and adding any missing or specialized metadata;

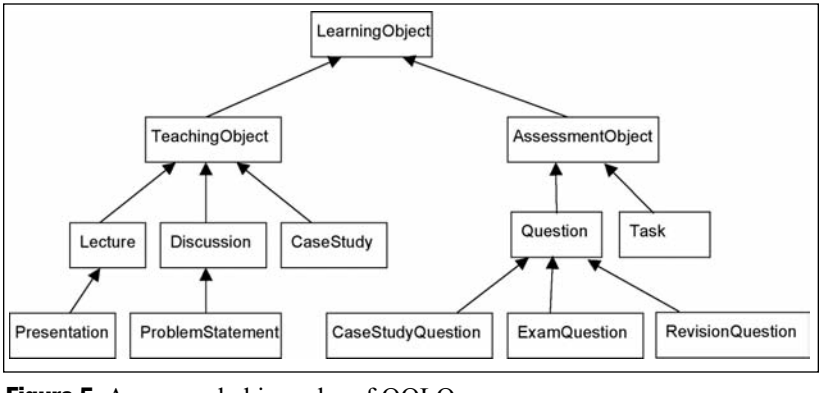

Figure 5. An example hierarchy of OOLOs

Table 1
Effort Savings That Can be Offered by Using an OOLO Model in Creating Learning Resources (all numbers are approximations)

	Traditional LO approach		OOLO approach	
	Minimum metadata used	Maximum metadata used	Minimum metadata used	Maximum metadata used
Number of assets to be used	200		200	
Number of LOs to be created	At least 100		At least 100	
Number of asset metadata to be entered	1600	12000	0	6000
Number of LO (SCO) metadata to be entered	1100	6000	0	3000
LO packages to be created from scratch	At least 100		0	

- if a specialized kind of LO does not exist in the hierarchy of predefined LOs then it can be created by extending an existing LO (e.g., an AssignmentTask LO can be created by extending the Task LO); and
- Aggregate LOs may be created by creating a new LO (e.g., Seminar) that is made up of a number of existing LOs (e.g., CaseStudy, CaseStudyQuestion, Task etc.). A number of commonly used aggregated LO may also be predefined so that effort and time will be further minimized. This will enable the fast and easy creation of SCORM Activity Items and Content Organizations.

From the example we can easily realize that the existence of a well structured set of hierarchically related OOLOs (i.e., LO templates with appropriate attributes and functionality already attached to them), can minimize the effort, time and cost required to develop LOs. To develop such an efficient hierarchy it is necessary to:

- define the best way to structure and represent OOLOs so that they demonstrate OO functionality (encapsulation of operations and data, inheritance, polymorphism);
- analyze, structure, and represent the LO Metadata into an appropriate hierarchy that will make the metadata more efficient for OOLO; and
- devise an extensible hierarchy of predefined OOLOs and OOLO aggre-

gations that will minimize as much as possible the need for the creation of entirely new LOs.

LO DESIGN MODELS AND TOOLS

LO Design Models

In the previous sections it has been demonstrated that a number of benefits may be realized by applying an OO approach to LO design and development. However, for the OOLO concept to be put in use, an OOLO model has to be developed. Such a model should aim in enabling the design and development of LOs in a way that will be more efficient than traditional LO models. The literature review and the author's primary research carried out so far, has revealed a number of features that have been found to be necessary for such a model. These features include:

- the specific and detailed definition of the general structure of the OOLO, which should include properties and methods;
- a hierarchy of predefined OOLOs that can be used or be extended to create the required learning resources for any learning scenario. This will be similar to the Application Programming Interfaces (APIs) offered by some programming languages (e.g., the Java API);
- a set of schema definitions for defining the elements making up the OOLO hierarchy;
- the mark-up documents that will be used for creating the LOs including extension mechanism that will support inheritance;
- support of relevant standards (i.e., SCORM, LOM);
- an appropriate notation for designing the OOLO;
- appropriate notations for representing the hierarchies of the LOs. For this purpose an existing notation scheme (such as Unified Modeling Language [UML]) may be used (i.e., extended to enable the design of OOLO), or a new scheme may be developed if an existing one cannot be adapted;
- notations for representing the LO metadata. If possible metadata should also be represented in an OO hierarchical way; and
- the model should reflect an OO approach that will provide flexibility, efficiency, and reusability to all the elements of the new LO model.

This list represents the required features of an OOLO model as they have been formulated through the literature review and research work done so far. It is expected that additional requirements may surface as further work is carried out, however this list of requirements will form a starting point for evaluating existing LO design models, and identifying the gaps between the current state of the art and the required OOLO model. To identify these gaps,

an analysis of existing LO models was carried out. Models studied include complete specifications that are the result of the work of organizations that are involved with the development of learning technologies, but also ideas for models that have been suggested by individuals or groups of people working in the area of e-learning.

The models studied include:

- the IMS Learning Design (IMS);
- a schema for Learning Object based on Object Oriented Model of Object Inheritance (Daniel & Honggang, 2003);
- application of the UML in modeling SCORM-conformant contents (Hu, 2005);
- an Instructional Design Model for Constructivist Learning (Sun & Williams, 2003);
- the CLEO model (Rehak & Blackmon, 2001); and
- a practical example of LO creation using Standard Generalized Markup Language (SGML)/Extensible Markup Language ([XML], Bartz, 2002).

These models have been evaluated against the degree to which they satisfy the desired features of an OOLO model as they were specified earlier. The results of this evaluation are outlined in Table 2. From the evaluation it can be concluded that the work done so far on LO design and development models is very limited. Very few models have been adequately developed to be put into practice and most of them concentrate on the pedagogy level. The LO design and development is currently carried out rather arbitrarily. Some attempts to create OO models for LO design and development have also been done, but apparently they have not been adequately extended and they do not support most of the requirements that are required for modeling OOLO.

LO Tools

When an OOLO model is developed, an appropriate tool will be needed to assist with the creation of OOLOs. Such a tool should include features that will enable the design and development of LOs based on the OOLO model. Since a number of LO tools already exist, it is necessary to evaluate those tools to assess whether it is possible to use any of them for the design and development of OOLOs and if not then identify the features that are required but are absent from existing systems. At this stage this evaluation process cannot exhaustively define all the necessary features, as the OOLO model itself does not yet exist. However based on the initial requirements of an OOLO model that have been previously defined, an initial appreciation can be made on some features that an OOLO tool is required to incorporate. These features are:

- enable the design of LOs based on the notations (for LOs, Metadata, LO relationships) defined by the OOLO model (existing tools cannot be

effectively assessed against this requirement as the specific notations are not yet known);

- enable the creation of LO classes (including properties and methods) representing the different possible types of LOs applying SCORM and LOM standards (probably through the creation of schemas and mark-up documents);
- enable the creation of LO classes by extending existing ones through inheritance relationships; and
- enable the creation of LO aggregations to represent the different LO aggregation levels as described in SCORM;

For the purpose of carrying out an initial evaluation of existing LO tools, a number of well-known commercial LO tools have been examined and their main purpose and functionality has been identified and compared to the above requirements. The tools evaluated include:

- ReLoad Metadata and Content Packaging Editor (http://www.reload.ac.uk/);
- InSite Studio (http://thorax.erc.msstate.edu/insite/default.aspx);
- Quest (http://www.ops.ltd.uk/products/prods/quest.html);
- Learning Activity Management System - LAMS (http://www.lamsinternational.com);
- Websphere (http://www.ibm.com/websphere); and
- Authorware (http://www.macromedia.com/software/authorware/).

The fact that the OOLO model is not yet implemented makes the evaluation of existing tools against it rather arbitrary. However from the initial evaluation of these examples of e-learning tools that make use of LOs, it can be estimated that the majority of the existing tools may not be able to support OOLO design and development, due to the absence of any OO features, such as the creation of LOs from existing ones through inheritance and the encapsulation of methods as well as metadata within LO classes. Nevertheless, one of the systems that have been examined adopts an OO view of LO design and development and it can be anticipated that such a system may be applicable for an OOLO model.

The comparative evaluation of all of the systems mentioned against some of the initial requirements for an OOLO tool is summarized in Table 3. However, further research should be carried out after implementation of the OOLO model, for a more complete specification of the requirements for a tool to support such a model and a more thorough evaluation of the capabilities of existing tools in supporting the model.

Table 2
A Comparative Evaluation of Existing LO Models Towards the Requirements of an OOLO Model

OOLO Model Desired Features	IMS Learning Design Information Model	A schema for LO based on OO Model of Inheritance	Application of the UML in modelling SCORM content	Instructional Design Model for Constructivists Learning	CLEO	A practical example of LO creation using SGML/XML
Applied at the LO level.	No. Pedagogy level.	Yes	Yes	Yes	Yes	Yes
Defines the structure of an OOLO	No	Yes	Yes	No	Yes	No
LOs encapsulate properties and methods	No	Yes	No	No	Yes	No
Offers a predefined hierarchy of LOs	No	No	No	Partly	No	Only a general example
Offers Schema definitions for predefined LOs.	No	No	No	No	No	Example only
Mark-up documents for LO creation that support inheritance.	No	Yes	No	No	Not specified	No
Supports LOM	Yes	No	Yes	Not specified	No	No
Supports SCORM	Yes	No	Yes	Not specified	No	No
Offers notations for representing LOs and LOM	UML	Yes	Extended UML	Yes	Yes (for LOs)	Yes
Supports aggregation.	Yes	Yes	No	Yes	Yes	No
Offers abstraction	No	No	No	No	Yes	No
Adequately developed (usable)	Yes	Yes	No	Yes	Yes	Yes

Table 3
A Comparative Evaluation of Existing LO Tools Towards the Requirements of a Tool to Support OOLOs

	ReLoad	InSite Studio	Quest	LAMS	WebSphere	Authorware
Level at which it can be applied	LO / Pedagogy	Mostly Pedagogy	Pedagogy	Pedagogy	LO	Mostly pedagogy
LO design and development tools included in the package	Content package, metadata and learning design editor.	Instructional design editor, LO editor.	Course authoring system	Learning Design editor	Dynamic course assembly system	Multimedia training authoring system
Implementing a desired notation for graphical representation of LO, Metadata and relationships	Yes `	Yes	Yes	Yes for LOs and relationships	Yes	Yes
Enable the creation of LO "classes" from scratch (by defining LOM as LO properties and LO methods)	No	No	No	No	No	No
Enable the creation LO "classes" by extending existing ones (inheritance)	No	No	No	No	No	No
Enable the creation of LO aggregations.	Yes	Yes	Yes	Yes	Yes	Yes
Supports LOM	Yes	Yes	No	No	Yes	Yes
Supports SCORM	Yes	Yes	No	No	Yes	Yes

CONCLUSIONS

This study has attempted to collect the varied views on the definition of an LO and give a clearer understanding of what an LO is currently considered to be. It has then proposed a view of an LO that incorporates attributes from the OO discipline and explained how SWE OO concepts can be linked to LOs. An explanation was then given on how these OO techniques can minimize the technical and cultural difficulties of using LOs. Through the use of a real life learning scenario it has attempted to prove that the use of such an OOLO as part of a larger OOLO model will overcome a number of the problems that using LOs involve. Such problems include mainly the complexity and time needed for LO design and creation, as well as reliability, reusability, extensibility, maintainability, and portability of LOs. A simple example of an OOLO model has been presented, and the way that this can be applied to the learning scenario has been described. The benefits of applying this model were quantified in terms of the number of LOs to be created from scratch and metadata to add to each learning resource. Finally, a survey into existing LO design models and tools has shown that the specification of a complete OOLO design model will be needed, and appropriate tools must be created or adapted.

Summing up the findings of this research work, we can conclude that an OO approach for the design and development of LOs has a lot to offer in terms of efficiency in the creation and reusability of LOs. Such an approach is expected to:

- minimize the time and effort needed to create learning resources by minimizing the amount of LOs to be created from scratch and metadata to be entered;
- minimizing the technical knowledge required to create LOs;
- increase the reliability and interoperability of new LOs as they will be based on existing tested ones;
- ensure the application of the appropriate standards without the developer's involvement;
- increase the reusability of LOs, as abstract LOs will not be bound to any specific context; and
- minimize the resistance of instructors to using existing learning resources due to the ability of OOLOs to easily adapt to different teaching styles, because of the absence of interdependency between content and context.

To make the application of such a model feasible a number of tasks need to be achieved. These include:

- the definition of the best way to structure and represent OOLOs so that

they demonstrate OO functionality (i.e., as a class/template with properties and methods);

- the implementation of OOLOs in a way that they will offer abstraction (hiding of implementation details from the user by providing an interface to enable interactions), enable inheritance and aggregation;
- the definition of an extensible hierarchy of predefined OOLOs;
- the definition or adaptation of appropriate notations to represent all the components of the model;
- the definition or adaptation of an appropriate (probably mark-up) language for implementing the OOLO (e.g., schemas and mark-up documents); and
- the creation or adaptation of a tool that will enable the design and development of the OOLO.

References

Advanced Distributed Learning Committee (ADL) Sharable Content Object Reference Model (SCORM, n.d.). *SCORM overview*. Retrieved January 1, 2008, from http://www.adlnet.gov/scorm/index.aspx

Bartz, J. (2002). Great idea, but how do I do it? A practical example of learning object creation using [Standard Generalized Markup Language] SGML. *Canadian Journal of Learning and Technology, 28*(3). Retrieved January 1, 2008, from http://www.cjlt.ca/content/vol28.3/bartz.html

Chrysostomou, C., & Papadopoulos, G. (2005, May). *An evaluation of e-learning technologies and trends: Establishing an object-oriented approach to learning object design and development.* Paper presented at the First International Conference on E-Business and E-learning (EBEL'05; pp. 343-348), Amman-Jordan. Retrieved January 1, 2008, from http://www.cs.ucy.ac.cy/~george/EBEL05a.pdf

Daniel, B., & Honggang, W. (2003, July). Developing a schema for learning object based on object oriented model of object inheritance. *Proceeding of the 3rd IEEE International Conference on Advanced Learning Technologies*, Athens, Greece. Retrieved January 1, 2008, from http://csdl.computer.org/comp/proceedings/icalt/2003/1967/00/19670439.pdf

Douglas, I. (2001, October). *Instructional design based on reusable learning objects: Applying lessons of object-oriented software engineering to learning systems design.* Paper presented at the 31st ASEE/IEEE Frontiers in Education Conference, Reno, NV. Retrieved January 1, 2008, from http://citeseer.ist.psu.edu/524398.html

Downes, S. (2001). Learning objects: Resources for distance education worldwide. *International Review of Research in Open and Distance Learning, 2*(1). Retrieved January 1, 2008, from http://www.irrodl.org/index.php/irrodl/article/view/32

Friesen, N. (2003). What are educational objects? *Interactive Learning Environments, 9*(3), 219-230. Retrieved January 1, 2008, from http://www.ingentaconnect.com/content/routledg/ilee/2001/00000009/00000003/art00003

Hu, S.C. (2005). *Application of the UML in modelling SCORM-conformant contents.* Kaifeng, Taiwan: Providence University, College of Computing and Informatics. Retrieved January 1, 2008, from http://csdl2.computer.org/comp/proceedings/icalt/2005/2338/00/23380200.pdf

Institute of Electrical and Electronics Engineers (IEEE) Learning Technology Standards Committee (LTSC; n.d.). *Learning technologies standards committee.* Retrieved January 1, 2008, from http://ltsc.ieee.org

Institute of Electrical and Electronics Engineers (IEEE) Learning Technology Standards Committee (LTSC) Learning Object Metadata (LOM; n.d.). *Draft standard for learning object metadata.* Retrieved January 1, 2008, from http://ieeeltsc.org/

Instructional Management Systems (IMS; n.d.). *IMS learning design information model.* Retrieved January 1, 2008, from http://www.imsglobal.org/learningdesign/index.html

Permanand, M., & Brooks, C. (2003). *Engineering a future for web-based learning objects.* Retrieved January 1, 2008, from http://www.cs.usask.ca/~cab938/icwe2003_mohan_brooks.pdf

Morris, E. (2005). Object oriented learning objects. *Australasian Journal of Educational Technology, 21*(1), 40-59. Retrieved January 1, 2008, from http://www.ascilite.org.au/ajet/ajet21/morris.html

Polsani, R. P. (2003). Use and abuse of reusable learning objects. *Journal of Digital Information, 3*(4), Article No. 164. Retrieved January 1, 2008, from http://jodi.tamu.edu/Articles/v03/i04/Polsani/

Poulton, C. (2005). *Applying principles of software engineering design to the development of reusable learning objects.* Retrieved January 1, 2008, from http://www.ecs.soton.ac.uk/~cmp301/comp6009/IRP%20cmp301.pdf

Rehak, D.R., & Blackmon, H. R. (2001). *Speculations: Content models – CLEO.* Retrieved January 1, 2008, from http://141.225.40.64/lsal/expertise/projects/cleo/report20010701/ speculations/contentmodels.html

Retalis, S. (2003) *Commentary on keeping the learning in learning objects.* Retrieved January 1, 2008, from http://www-jime.open.ac.uk/2003/1/reuse-05.html

Robson, R. (1999, June). Object-oriented instructional design and applications to the web. *Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications* (pp. 698-702), Seattle, WA.

Sosteric, M., & Hesemeier, S. (2002). *When is a learning object not an object: A first step towards a theory of learning objects.* Retrieved January 1, 2008, from http://www.irrodl.org/index.php/irrodl/article/view/106/185

Sun, L., & Williams, S. (2003). *An instructional design model for constructivist learning.* Retrieved January 1, 2008, from http://www.ais.reading.ac.uk/papers/con50-An%20Intructional%20design.pdf