

A HIERARCHICAL MODELING TOOL FOR INSTRUCTIONAL DESIGN

A THESIS SUBMITTED TO  
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES  
OF  
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MEHMET SERHAT AZGUR

IN PARTIAL FULLFILMENT OF THE REQUIREMENTS  
FOR  
THE DEGREE OF MASTER OF SCIENCE  
IN  
COMPUTER ENGINEERING

JANUARY 2010

Approval of the thesis:

**A HIERARCHICAL MODELING TOOL FOR INSTRUCTIONAL DESIGN**

Submitted by **MEHMET SERHAT AZGUR** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen  
Dean, Graduate School of **Natural and Applied Sciences**

\_\_\_\_\_

Prof. Dr. Müslim Bozyiğit  
Head of Department, Computer Engineering

\_\_\_\_\_

Assoc. Prof. Dr. Ali Hikmet Doğru  
Supervisor, Computer Engineering Dept., METU

\_\_\_\_\_

**Examining Committee Members:**

Prof. Dr. Müslim Bozyiğit  
Computer Engineering Dept., METU

\_\_\_\_\_

Assoc. Prof. Dr. Ali Hikmet Doğru  
Computer Engineering Dept., METU

\_\_\_\_\_

Dr. Attila Özgüt  
Computer Engineering Dept., METU

\_\_\_\_\_

Dr. Faruk Ağa Yarman  
General Manager, HAVELSAN

\_\_\_\_\_

Dr. Mete Ahmet Çakmakcı  
Secretary General, Türkiye Teknoloji Geliştirme Vakfı

\_\_\_\_\_

**Date:**

21.01.2010

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

**Name, Last name :** Mehmet Serhat Azgur

**Signature :**

# **ABSTRACT**

## **A HIERARCHICAL MODELING TOOL FOR INSTRUCTIONAL DESIGN**

Azgur, Serhat Mehmet

M.Sc., Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Ali Dođru

January 2010, 94 pages

A component-oriented tool for hierarchical modeling of instructional designs is developed. The motivation is to show that hierarchical representation of instructional designs is easier, better and more effective for modeling. Additionally a modeling language is developed to provide an effective, flexible and easy to use integration model in which all teaching components are discovered, defined and connected. In order to fulfill the above purposes an abstract notation is developed that is sufficiently general and adapting top-down hierarchic approach to represent Units of Learning (UoL), Operational Knowledge Units (OKU), Learning Objects (LO), and Learning Components (LC) with respect to the common structures found in different instructional models. COSEML, a top-down hierarchic, and component oriented modeling language has been used as a reference and the core concept in developing the Educational Component Oriented Modeling Language (ECOML). The high-level architecture of ECOML provides the means for designing instructional structures. It describes how LOs, UoLs, OKUs and LCs are sequenced in a certain context or knowledge domain. The resulting model can be reused in different contexts and across different educational platforms.

Keywords: Component Oriented Educational Modeling Language, COSEML, Hierarchical Framework, Top-down Graphical Instructional Design, Component Oriented Software Engineering.

# ÖZ

## EĞİTİMSEL TASARIM İÇİN HİYERARŞİK MODELLEME DİLİ

Azgur, Mehmet Serhat

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Doç.Dr. Ali Doğru

Ocak 2010, 94 sayfa

Eğitimsel uygulamaların biçimlendirileceği bileşen-yönelimli bir iskelet mimari geliştirilmiştir. Hedef; her bir öğretim bileşeninin bulunduğu etkili, esnek ve kullanımı kolay bir bütünleme modelinin bulunması, tanımlanması ve ilişkilendirilmesidir. Değişik eğitim modellerinde bulunan ortak yapılar esas alınarak Öğrenme Birimleri (ÖB), Operasyonel Bilgi Birimleri (OBB), Öğrenme Nesneleri (ÖN) ve Öğrenim Bileşen'lerini yeteri kadar genelleyerek gösterebilecek yukarıdan-aşağıya hiyerarşik yaklaşımı esas alacak bir soyut notasyon geliştirilmiştir.

Eğitimsel Bileşen Yönelimli Modelleme Dili (EBYMD) nin geliştirilmesinde, yukarıdan-aşağıya hiyerarşik ve bileşen yönelimli yazılım mühendisliği modelleme dili olan COSEML örnek kaynak ve ana fikir olarak kullanıldı. EBYMD'nin üst düzey mimarisi eğitimsel yapıların tasarımında gerekli araçları sunmaktadır. ÖB,OBB, ÖN ve öğrenim bileşenlerinin, belirli bir ortamda veya ilgi alanı bilgisinde nasıl sıralanacağını tarif eder. Geliştirilen model değişik amaçlarla ve değişik eğitim platformalarında tekrar kullanılabilir.

Keywords: Bileşen Yönelimli Eğitimsel Modelleme Dili, COSEML, Hierarchical Framework, Yukarıdan-aşağıya grafiksel eğitim tasarımı, Bileşen yönelimli yazılım mühendisliği.

To My Father & Mother

## ACKNOWLEDGMENTS

First and foremost I offer my sincerest gratitude to my supervisor, Asoc.Prof.Dr.Ali Doğru, who has supported me throughout my thesis with his advice and knowledge whilst allowing me the room to work in my own way. Without him this thesis would not have been completed or written. One simply could not wish for a better and friendlier supervisor.

The author would also like to thank to Prof.Dr Müslim Bozyiğit, Dr.Attila Özgüt, Dr.Faruk A. Yarman and Dr.Mete A. Çakmakcı for their understanding and support.

In my daily work I have been lucky to be accompanied with a friendly and industrious group of fellow students. Necibe Nur Koç, Deniz Peker and Taner Mansur, who helped me in the Java programming efforts for the realization and implementation of the idea.

## TABLE OF CONTENTS

ABSTRACT.....	IV
ÖZ .....	V
ACKNOWLEDGMENTS .....	VII
TABLE OF CONTENTS .....	VIII
LIST OF TABLES.....	X
LIST OF FIGURES.....	XI
CHAPTER	
1. INTRODUCTION .....	1
1.1. Motivation .....	3
1.2. Organization of the Thesis .....	4
2. BACKGROUND .....	6
2.1. Standards and Definitions of Relevant Terms .....	8
2.1.1. Definitions of Relevant Terms .....	19
2.2.2. Educational Standards .....	19
2.2. Providing a Structure.....	17
2.2.1. Component Based Software Engineering.....	19
2.2.2. COSE Approach and Education .....	20
2.2.3. Instructional design and the ECOML approach .....	23
2.3. Taxonomies .....	24
3. RELATED WORK .....	27
3.1. What is IMS-LD?.....	28
3.2. Authoring Tools .....	29



3.3. Comparison of two authoring tools. ....	32
3.3.1. ReLoad LD Editor .....	33
3.3.2. COSMOS ASK-LTD Editor .....	37
4. A MODELING TOOL FOR INSTRUCTIONAL DESIGNERS .....	44
4.1. Overview of Requirements .....	45
4.2. Software Modeling .....	48
4.3. Defining the Language – Essentials.....	50
4.4. Components - Graphical Modeling Elements.....	60
4.5. An Example Model.....	65
4.6. Comparison of ECOT with Others .....	70
5. CONCLUSION AND RECOMMENDATIONS .....	80
5.1. Quality Concerns .....	81
5.2. Conclusions.....	84
5.3. Future Work and Recommendations .....	85
REFERENCES.....	88

## **LIST OF TABLES**

### **TABLES**

Table 1. Authoring Tools as compiled by IMS-UNFOLD .....	30
Table 2. ECOML Graphical Symbols .....	62
Table 3. Comparison of ECOT with Other Authoring Tools .....	72

## LIST OF FIGURES

### FIGURES

Figure 1. Learning Object and its metadata. ....	13
Figure 2. Unit of Learning vs. Learning Design .....	15
Figure 3. Learning Design Systems (Schneider, 2009)14.....	31
Figure 4. Welcome Page of the ReLoad LD Editor .....	34
Figure 5. ReLoad can be used as either Level A or B or C editor.....	35
Figure 6. Weak points of ReLoad LD editor .....	36
Figure 7. Main menu and the first dialogue page.....	38
Figure 8. It is possible to import IMS complaint LDs. ....	39
Figure 9. Menu that shows selection of possible scenarios .....	40
Figure 10. Main menu and first dialogue page of ASK-LDT .....	41
Figure 11. “Content Packager” dialogue page .....	42
Figure 12. “Educational Scenario” dialogue page .....	42
Figure 14. ECOML’s Approach to Educational Modeling.....	47
Figure 15. ECOML Authoring Process .....	48
Figure 16. Structural Properties of ECOML .....	51
Figure 17. Fields of a COSEML (ECOML) component. ....	55
Figure 18. Class hierarchy of CosemlFrame .....	56
Figure 19. Class hierarchy of CosemlToolBar .....	56
Figure 20. Class hierarchy of CosemlSplitPane.....	56
Figure 21. Class hierarchy of CosemlTree and CosemlTreeNode .....	57
Figure 22. Class hierarchy of CosemlDrawPanel .....	57
Figure 23. Class hierarchy of dialog classes .....	58
Figure 24. Tree data structure .....	59
Figure 25. ECOML’s Graphical Symbols .....	61
Figure 26. ECOML’s Main Menu .....	66
Figure 27. Learning Design Properties in Main Diagram.....	67
Figure 28. Learning Design in Main Diagram .....	68
Figure 29. Hierarchical Representation of Learning Components.....	69

Figure 30. Lowest Level with Physical Components .....	70
Figure 31. An instructional model generated by MOT+ .....	71
Figure 32. An instructional model generated by LAMS .....	73
Figure 33. Instructional models generated by Dialog+ .....	74
Figure 34. An instructional model generated by Collage.....	75
Figure 35. An instructional model generated by Moodle.....	76
Figure 36. An instructional model generated by ASK-LDT <sup>47</sup> . ....	77
Figure 37. Main menu of CopperAuthor.....	78

# CHAPTER 1

## INTRODUCTION

Information and Communications Technologies (ICT) industry has advanced so rapidly that one can find software engineering in almost every aspect of modern life. Field of Education is not an exception. And, effects of ICT can easily be seen in almost all phases of Education as efficient and effective ways of improving the educational competency. Instructional design (ID) and delivery of pre-designed courses with computer-based methods with the help of computers are getting common every day.

In order to improve the quality of different fields of education, efficient use of software tools is gaining wide acceptance among scholars as well. Supporting instructional design with visual tools is one of them. As Botturi (2004) says, visual models are being developed for supporting and enhancing the instructional design process in recent years. The objective of these development efforts is to represent the instructional design as a sequence of steps or as a set of elements that characterize the educational process. Özçınar (2009) has recently made a broader definition of the instructional design as the systematic development of instructional specifications, using learning and instructional theory derived from behavioral, cognitive and constructivist theories, in order to ensure the quality of instruction. It is the entire process of the analysis of learning needs and goals and the development of a delivery system to meet those needs, including development of instructional materials and activities, together with the testing and evaluating of all instruction and learner activities. On the other hand, Paquette (2004) defines the term instructional engineering (IE) as a method that supports the planning, analysis, design and the delivery of a learning system, integrating the concepts, the processes and the principles of ID, *software engineering*, and cognitive science. It has much in common with educational

modeling languages, sharing *software engineering* and cognitive science approaches. The main difference is that IE is a methodology mostly concerned with the processes and principles that will produce specifications of a learning system.

By the same token, teaching/learning processes has become open to wider audiences with the advancements in ICT and Internet technology. Which in turn, made instructional process more complex, sophisticated and more difficult to design, implement and administer. Words like e-learning, blended learning, learning management system or course management system are being used commonly and they already took their place in the instructional and/or pedagogical dictionaries as technical terms.

Yuen and Ma's research (2008) indicated that the successful pedagogical use of technology depends on teachers' attitudes and acceptance towards technology. As Breen et al. (2001) and Marriott et al. (2004) claim that the actual formal use of information technology in undergraduate and graduate studies still remains inconsistent and varies significantly from individual courses to individual institutions (as cited in Yuen A. H.K. & Ma W.W.K, 2008). Of course, use of technology does not involve only the delivery of the course but it starts right from the beginning, with the design of the course.

Blended learning technology certainly opens possibilities for new ways of engagement and invites innovative pedagogies. But, not all teachers are necessarily motivated to use it. Inclusion of new software tools that are helpful and easy to use will definitely encourage the rather computer illiterate and hesitant teachers. Since, it will not be possible to resist the upcoming of new ICT technologies, academicians are opt to follow the trend and make use of new facilities or conveniences that are being developed for them. This researcher believes that afore mentioned engagement process should start with the design of the course and continue with the delivery process. If software tools with user-friendly graphical interfaces are provided for the educators they will certainly get motivated for other innovative blended learning facilities.

The Princeton University defines “education”<sup>1</sup> as “the gradual *process* of acquiring knowledge”, and “the *activities* of educating or instructing”. This definition of education suits very well to the concepts of component-oriented software engineering and process management. The activities and the knowledge acquirement process can easily be defined in terms of components and represented within graphical models. As component modeling and related technologies improve process modeling, modeling of complex systems becomes easier, quicker and more efficient.

### **1.1. Motivation**

The main encouraging idea was to show that hierarchical representation of instructional designs is easier, better and more effective for modeling. Another motivational idea was to develop a modeling language to provide an effective, flexible and easy to use integration model in which all teaching components are discovered, defined and connected. And, in order to fulfill the above purposes an abstract notation was need to be developed that is sufficiently general and adapting top-down hierarchic approaches to represent Units of Learning (UoL), Operational Knowledge Units (OKU), Learning Objects (LO), and Learning Components (LC) with respect to the common structures found in different instructional models.

Karampiperis and Sampson (2005) say that the need for modeling educational systems that support a diverse set of pedagogical requirements has been an important issue in educational environments. But, Caeiro et al. (2004) and Knight et al. (2005) point out to the fact that lack of standardization and considerable abundance of pedagogical approaches have made didactically feasible designs difficult to become widespread.

After the introduction of IEEE Learning Technology Systems Architecture (LTSA) standards, ADL Sharable Content Object Reference Model (SCORM), which refines the IEEE LTSA reference architecture (Karampiperis and Sampson, 2005), and IMS-LD specifications, significant R&D effort started being devoted to modeling educational systems that are tailored to specific pedagogical

---

<sup>1</sup> <http://wordnetweb.princeton.edu/perl/webwn?s=education> last accessed October 04, 2009.

approaches. Various articles in the literature and world-wide presentations in the seminars and conferences indicate the lack of educational (authoring) tools that provide easier representation of units of learning for non-expert authors, including instructional design authors and teachers/instructors who want to share knowledge, skills, perspectives and views with others, in order to develop new knowledge, or reuse existing knowledge resources (Burgos & Griffiths, 2005; Dodero, Tattersall, Burgos, & Koper, 2006; Griffiths & Blat, 2005; Hernandez-Leo, Harrer, Dodero, Asension-Perez, & Burgos, 2006).

Knowing that Component Oriented Software Engineering (COSE) approach, as suggested by Dogru and Tanik (2003) as part of the Abstract Design Paradigm (Tanik and Chan, referenced by Dogru, 1999), can decompose the system structure hierarchically within the specified environments. And, its applicability to the educational processes that are similar in nature and availability of experts in METU motivated the researcher to:

- Develop a graphical educational language that is easier to use in modeling educational processes.
- Support COSE with another case study.
- Provide yet another case to support the justification of COSEML.

A graphical tool with significant set of requirements and easy-to-use features that provide basic scaffolding in the form of wizards and templates may be an ideal authoring environment in support of instructional designers, as well as non-expert authors for a convenient way of modeling their knowledge.

## **1.2. Organization of the Thesis**

The first chapter is an introductory chapter, which gives some preliminary thoughts an insight about the purpose, and motivation of this researcher towards the application of COSE approach to the field of Education. It also connects the link between instructional modeling and the hierarchical approach.

Beyond the first chapter, thesis is organized as follows: In Chapter Two, necessary background on graphical software tools in the field of education, software components, component-based modeling and top-down hierarchical component architectures are given. Chapter Three describes the COSE approach,



defines the ECOML modeling language and its relation with COSEML, and compares ECOML with other authoring tools that are available in the market. Also an example model that is created with ECOML is presented. In Chapter Four, the design overview of the developed educational graphical modeling editor is given. Finally, Chapter Five makes the conclusion, and presents the recommendations for future work.

## CHAPTER 2

### BACKGROUND

The advent of technologies and fast grow of Internet have changed the idea of what a course is. Computer-based teaching, learning and course offerings are swiftly becoming popular. So-called “online learning”, “e-learning” and/or “blended learning” courses are being discussed among scholars and increasingly offered in many schools. This trend in course offerings and delivery brings the changes in the design of the courses, as well.

As Cantoni and Di Blas (2002) say the process of designing courses has grown a more structured and interdisciplinary process, which is becoming a highly complex task even for a professor let alone a simple instructor to cope with. In some respects, as Cantoni and Di Blas (2002) say teaching is developing from craftsmanship to a large-scale production process. The researcher believes that COSE approach may well be the solution and certainly will help the instructional designers at this respect. Representing the process in a component-oriented and top-down hierarchical way will help the instructional designers for better structured and easy-to-understand designs. That is to say, object oriented, component-based and top-down hierarchically approached methodologies are better for developing educational software applications within the instructional and pedagogical paradigms. Then software tools that make use of COSE approach will be the effective guidelines for easier and more efficient ways of instructional modeling efforts. Of course, as Koper and Olivier (2004) said the underlying assumption is that *learning process is a process of consuming content and teaching is envisioned as the art of selecting and offering content in a structured, sequenced way*, and of tracking the learner’s progress and assessing the acquired knowledge. Although, building better tools, which in turn create better educational models does not mean better teaching and/or learning. It is

only the first step for instructional designers to understand and manipulate the design process easier and better. Koper and Olivier (2004) also assume that every educational practice can be represented in a design description. As a consequence, once the process is represented with components that represent content then teaching process can easily be modeled.

There are hundreds of different pedagogical models described in the literature and, there are many so-called lesson plans that are shared on the Internet. Since teaching environments are not equally developed and modernized among different primary, secondary and high schools, as well as higher education institutions through out the world, students are being educated with different teaching methods in different discipline plans. Even though, this poses important problems for modeling software efforts, new methods of teaching/learning designs with new models, and best practices continue to be developed and formulated.

Christiaans and Venselaar (2005) claim that a lot of individuals who possess domain specific knowledge of their chosen field are motivated to develop the modeling of their knowledge into instructional designs, but complexity of modeling tools and lack of sufficient design experience discourage them to translate their educational knowledge into effective instructional modeling for interoperability and share of knowledge between various platforms. As it is also declared by Knight et al. (2005) specifying reusable chunks of learning content and defining an abstract way of designing different units (e.g. courses, lessons etc.) are two of the most current research issues. Hence, the affluence of research studies on educational modeling, complexities of pedagogical methods and the continuing development of software tools created chaos in this field.

Consequently, significant R&D effort has been started and being devoted to set up standards and specifications for modeling educational systems. The main ones that are widely accepted and related to the educational issues are IEEE-1484 Standards, IMS-LD specifications, ADL-SCORM and AICC-AGR. These standards are sometimes complete each other (IEEE-LOM and IMS-LD) and sometimes compete (SCORM and IMS-LD-IM) for similar purposes. Each approach comes with its own definition of educational content and process. The main purpose is to generate content that can be created once and used in many

different systems and situations without modification. And, eventually attain the possibility of creating meta-models of educational systems with the help of new software modeling approaches.

## **2.1. Standards and Definitions of Relevant Terms**

### **2.1.1. Definitions of Relevant Terms**

The terms that are referenced through out this document are briefly explained as follows:

**Advanced Distributed Learning (ADL)** is the product of the ADL Initiative, established in 1997 to standardize and modernize training and education management and delivery. The vision of the ADL Initiative is to provide access to the highest-quality learning and performance aiding that can be tailored to individual needs and delivered cost-effectively, at the right time and in the right place. The ADL Initiative developed SCORM and the ADL Registry.

**SCORM** is explained in detail at the “2.1 Standards and Definitions of Relevant Educational Terms” section of Chapter 2.

**ADL Registry** contains all of the registered entries that contain metadata about the content in a repository. ADL Registry provides centrally searchable information, in the form of metadata records (not actual content). The metadata describes many different kinds of objects to enable their discovery and reuse regardless of their location or origin. ADL uses structured and collaborative methods to convene multi-national groups from industry, academia, and government who develop the learning standards, tools, and content. ADL Co-Laboratory’s web site can be reached at <http://www.academiccolab.org/>.

**Creative Commons License:** Creative Commons is a nonprofit corporation that provides free licenses and other legal tools to mark creative work with the freedom the creator wants it to carry, so others can share, remix, use commercially, or any combination thereof. For more information please have a look at <http://creativecommons.org/about/> (last accessed January 17, 2010).

**GNU:** GNU General Public License (GNU GPL), popular free software license, and the only license written with the express purpose of promoting and

preserving software freedom. Related licenses include the GNU Lesser General Public License (GNU LGPL), the GNU Affero General Public License (GNU AGPL) and the GNU Free Document License (GNU FDL). For more information please have a look at <http://www.fsf.org/licensing/licenses/lgpl.html> (last accessed January 17, 2010).

**Institute of Electrical and Electronics Engineers** (IEEE) is an international non-profit, professional organization for the advancement of technology related to electricity. Brief explanation of the standards that are published by the IEEE Learning Technology Standard Committee can be found in at the “4.3 Standards and Definitions of Relevant Educational Terms” section of Chapter 2. Relevant information is also available at <http://ltsc.ieee.org/p1484>.

**Instructional Design** is the name of the discipline, as it is used in the American literature (Paquette, 2004). It is defined as “the process of analyzing students' needs and learning goals, designing and developing instructional materials” by the University of Texas<sup>2</sup>.

**IMS:** The Instructional Management Systems (IMS) Global Learning Consortium<sup>3</sup> is a non-profit international organization composed of commercial and academic institutions, which creates specifications for distributed learning. These include IMS Metadata, IMS Content Packaging, IMS Simple Sequencing, and IMS QTI (Question and Test Interoperability, that defines a standard format for the representation of assessment content and results)<sup>4</sup>.

**Learning Activity (LA)** is defined by the IMS Learning Design specifications as the structured lesson plan, which is organized according to specific pedagogical models. An **activity** can be formally defined as a triple containing the *content* that is delivered by an educational system the *actors* participating in the activity (such as the learner or a group of learners, the tutor etc.) and their corresponding *interactions*. (Karampiperis P, Sampson D, 2005).

---

<sup>2</sup> <http://www.utexas.edu/academic/diia/assessment/iar/glossary.php>, last accessed December 06, 2009

<sup>3</sup> <http://www.imsglobal.org/>, last accessed June 02, 2009.

<sup>4</sup> [http://www.unfold-project.net/general\\_resources\\_folder/AboutLD/introld](http://www.unfold-project.net/general_resources_folder/AboutLD/introld) last accessed June 02, 2009.

**Learning Component (LC)** is any digital or non-digital educational building block, which can be used, re-used or referenced during any platform independent teaching/learning process. A Learning Component can be assembled, disassembled and combined with similar learning objects (and/or learning units) to form a module that can be used or referenced in a pedagogical model. They may incorporate meta-data, which represent educationally meaningful construction such as a learning objective.

Learning Components are meant for run-time connecting modules which may include any digital or non-digital text document, file, book, questionnaire, database, quiz, etc as well as collaborative tasks such as discussion, voting, small group debate, etc.

**Learning Design (LD)** is an application of a pedagogical model for a specific learning objective, target group, as a specific context and content, so that pedagogical models can be shared and reused across instructional contexts and subject domains (Koper & Olivier, 2004). It is the description of the specific ordered learning & support activities that have to be performed (or are performed) by users to attain a specific learning goal (Koper, 2005).

**Learning Object (LO):** any entity, digital or non-digital, which can be used, re-used or referenced during technology supported learning. (David Wiley, 2000, as cited in Cliff Gibson and Stephen Harlow, 2004).

Learning Object is an aggregation of one or more digital assets, incorporating meta-data, which represent an educationally meaningful stand-alone unit. (Daiziel (2002) as cited in Cliff Gibson and Stephen Harlow, 2004).

**Learning Object Metadata (LOM):** is a data model, usually encoded in XML, used to describe a learning object and similar digital resources used to support learning. The purpose of learning object metadata is to support the reusability of learning objects, to aid discoverability, and to facilitate their interoperability, usually in the context of online learning management systems (LMS). The IEEE 1484.12.1 – 2002 Standard for Learning Object Metadata is an internationally recognized open standard (published by the IEEE Standards Association) for the description of ‘learning objects’. Relevant attributes of learning objects to be described include: type of object; author; owner; terms of

distribution; format; and pedagogical attributes, such as teaching or interaction style.<sup>5</sup>

**Operational Knowledge Unit (OKU)** is defined by Tanik et al. (2009) as an innovative method of decomposing courses into electronically manageable “semantic” units. These semantic units correspond to specific learning competencies. The difference is that an OKU is a “process” not just a module in the classical sense. Tanik et al. [2009] argue that in an OKU students need to perform a certain process (activity) to internalize the knowledge presented to them. To be able to demonstrate mastery of a specific knowledge competency, the learner goes through the same process to generate an answer. Hence, OKUs are learning components and/or units of learning, as well as semantically and electronically managed processes. Knowledge becomes the operational piece of the process in every OKU.

With the above factors in mind, OKUs are designed in such a way that they play the role of intelligent interfaces to create knowledge. OKUs pull information together in intelligent ways, including the ability to reason and take inferences from the data and embed logic in the educational model.

**UML:** The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development.<sup>6</sup>

**ZIP file:** A file that has been compressed, or reduced in size, to save storage space and allow faster transferring across a network over the Internet. To read the information, the file must be uncompressed into its original form.<sup>7</sup>

#### 2.1.2. Educational Standards

The researcher believes that any research & development effort should be built on available standards and specifications wherever possible. Obviously, this

---

<sup>5</sup> [http://wiki.cetis.ac.uk/What\\_is\\_IEEE\\_LOM/IMS\\_LRM](http://wiki.cetis.ac.uk/What_is_IEEE_LOM/IMS_LRM), last accessed January 17, 2010

<sup>6</sup> [http://en.wikipedia.org/wiki/Unified\\_Modeling\\_Language](http://en.wikipedia.org/wiki/Unified_Modeling_Language) last accessed January 17, 2010

<sup>7</sup> <http://dl.austincc.edu/students/DLGlossary.html>, last accessed December 06, 2009.

includes the educational standards and specifications that are declared by IMS<sup>8</sup>, IEEE LTSC<sup>9</sup>, AICC<sup>10</sup>, ADL<sup>11</sup>, and ISO/IEC JTC1/SC36<sup>12</sup>. All of these standards specify architectures for information technology-supported learning, education and training systems that describe the high-level system design and the components of those systems. However, as indicated by Knight et al. (2005) no standard has yet been established for the graphical representation of learning designs. But, Knight (2005) also indicates that there are many possible methods of graphical representations.

Each standards organization has its own educational terminology. Brief description of them includes:

**Sharable Content Object (SCO) and Sharable Content Object Reference Model (SCORM).** SCORM is a collection of standards and specifications for web-based learning and it defines communications between client side content and a host system called the run-time environment (commonly a function of a learning management system). SCORM also defines how content may be packaged into a transferable ZIP file. SCORM defines a specific way of constructing Learning Management Systems and training content so that they work well with other SCORM conformant systems. Therefore, as Griffiths (2005) says it is a publishing system for educational resources in www and/or CD/DVD.

On the other hand, a **SCO**<sup>13</sup> is defined as the most granular piece of training in a SCORM world. Some would call it a module, a chapter or a page... the point is that it considerably varies. It is the smallest piece of content that is both reusable and independent.

Griffiths (2005) claims that SCOs and SCORM has been extensively criticized in the educational world with shortcomings such as activities and

---

<sup>8</sup> <http://www.imsproject.org>, last accessed September 20, 2009.

<sup>9</sup> <http://www.ltsc.ieee.org>, last accessed September 20, 2009.

<sup>10</sup> <http://www.aicc.org/pages/aicc2.htm#CMI>, last accessed September 20, 2009.

<sup>11</sup> <http://www.adlnet.org>, last accessed September 20, 2009.

<sup>12</sup> [http://www.iso.org/iso/standards\\_development/technical\\_committees/list\\_of\\_iso\\_technical\\_committees/iso\\_technical\\_committee.htm?commid=45392](http://www.iso.org/iso/standards_development/technical_committees/list_of_iso_technical_committees/iso_technical_committee.htm?commid=45392), last accessed September 20, 2009.

<sup>13</sup> <http://www.scorm.com/scorm-explained/one-minute-scorm-overview>, last at September 20, 2009.



collaborations not being supported. That's why, SCORM V2.0 is getting prepared (Rustici, 2008) but it is not published, yet.

**Learning Object (LO) and IEEE Standards 1484.** The IEEE-1484 Learning Technology Standards is an internationally recognized open standards and it is published by the IEEE Standards Association. As Agostinho et al. (2003) also admits that even though there are alternative terms, which are being used interchangeably with learning objects such as instructional objects, educational teaching objects, knowledge objects, intelligent objects, digital learning items and data objects, "Learning Object" entity has been commonly accepted and widely used as an educational term in the literature. A **Learning Object** is defined in IEEE standards as any entity, digital or non-digital that can be used, reused, or referenced during technology supported learning (Duval, 2002). Dalziel (as cited in Gibson & Harlow, 2004) defines the same, as 'Learning Object' is an aggregation of one or more digital assets incorporating meta-data, which represent an educationally meaningful stand-alone unit. Koper (2001) says that there are several ways of viewing learning objects. The most common, but often implicit, idea is depicted in Figure 1, as given by Koper (2001).

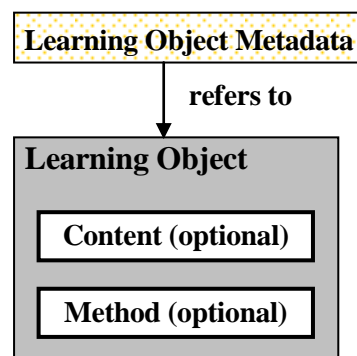


Figure 1. Learning Object and its metadata.

In principle learning objects have content (attributes and other learning objects) and descriptions of the behavior of the learning object (operations).

Clearly, the idea of a learning object model conforms to the principles of objects in the theories of object-oriented design approaches. This implies that principles of encapsulation, abstraction and inheritance may also be present. But, according to Koper (2001) learning objects suffer from a lack of ability to semantically express relations between different types of objects in the context of the use of an educational setting. That is to say, the learning object model fails to provide a model for the structure of the content of different objects. The typing of objects also varies according to different pedagogical stances, so there is a need for another meta-model to describe the relationships. To overcome this issue another group of efforts referred to as Learning Design (LD) has developed.

#### **Unit of Learning (UoL) and IMS Learning Design (LD) Specifications.**

IMS Global Learning Consortium has published the IMS-LD specifications that provide an information model and XML binding which facilitates the conceptualization and formalization of a Learning Design (LD) for the purpose of standardized information exchange and integration with software systems (IMS-LD-IM, 2003). Because of the fact that many of the tools and editors for learning designs are to be developed around this specification, maintaining the compatibility is considered to become an important issue.

A unit of learning (UOL) is defined by Vogten et al. (2006) as a package that consists of meta-data about the course, the learning design of the course and references to physical resources and/or the physical resources themselves (learning objects and learning services) that are used in the course. Similarly, Koper and Manderveld (2004) define UoL as “A Unit of Learning is defined as an artifact that is designed for learners to achieve one or more interrelated learning objectives. A unit of learning can not be broken down into its component parts without losing its semantic and pragmatic meaning”. In another article Koper & Olivier (2004) mention about LD and its relation with UoL as LD being used to specify the learning design of e-learning courses (UoLs). By providing a generic and flexible language, the LD specification supports the use of a wide range of pedagogies. It is based on a pedagogical meta-model (Koper & Manderveld, 2004; Koper & Olivier, 2004).

According to Koper and Olivier (2004), a LD can be defined as an application of a pedagogical model for a specific learning objective, target group, and a specific context or knowledge domain. Koper (2005) assumes that every unit of learning has a learning design that can be described explicitly:

- Learning designs refer to resources (learning objects or services)
- Everyone who is creating or adapting a unit of learning is a 'learning designer' at that moment in time.

And the difference with a unit of learning is explained as:

- Every unit of learning contains a learning design and its connected resources.
- A learning design refers to learning resources, but does not include the resources themselves.

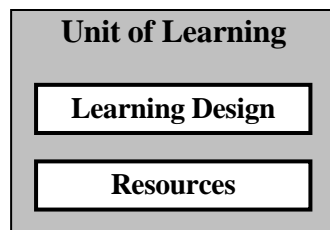


Figure 2. Unit of Learning vs. Learning Design

Eventually, this work led to the development of IMS Learning Design Specifications. IMS LD evolved from Educational Markup Language (EML). IMS LD was formally published in 1993. The latest version is published in 2003. In Europe, several research communities use IMS LD as a basis for various kinds of projects. IMS LD is financially supported by the European Union. But Schneider (2009) claims that there is no end-user ready implementation of an authoring tool or a delivery platform as of April 2008<sup>14</sup>. Since development of reference implementations are continuing, we may expect to see some day a

---

<sup>14</sup> [http://edutechwiki.unige.ch/en/IMS\\_Learning\\_Design](http://edutechwiki.unige.ch/en/IMS_Learning_Design), last accessed December 02, 2009.

friendly end-user solution. Nowadays, there exist some user-friendly tools like the Recourse editor. However, use of these tools still requires understanding the LD standards and as of May 2009, user-friendly players still don't seem to exist<sup>14</sup>.

Knight, Gasevic, and Richards, (2005) say that this process of representing UoLs can be conceptually pictured as pulling learning objects from a repository and using the *learning designs* to integrate learning objects into activities that involve learners. Therefore, *learning designs* can be represented graphically formalized according to an information model. But, the fact of the matter is that, as it is also claimed by Knight et al. (2005), there is no standard has yet been established for the *graphical representations* of learning designs; however, there are many possible methods.

On the other hand, there are still questions regarding the Learning Design yet to be answered. Knight et al. (2005) ask such questions as: How can we employ only specific parts of a learning object, rather than the learning object as a whole in a specific learning design, and how can we reuse the same learning design in different contexts with different learning objects.

**International Standardization Office (ISO)** is an international-standard-setting body composed of representatives from various national standards organizations<sup>15</sup>. ISO standards that are related to Information technology includes:

ISO/IEC 24738:2006 - Information technology -- Icon symbols and functions for multimedia link attributes

ISO/IEC TR 14471:2007 - Information technology -- Software engineering -- Guidelines for the adoption of CASE tools

ISO/IEC 14102:2008 - Information technology -- Guideline for the evaluation and selection of CASE tools

ISO/IEC TR 11580:2007 - Information technology -- Framework for describing user interface objects, actions and attributes.

ISO/IEC 24751-1:2008 - Information technology -- Individualized adaptability and accessibility in e-learning, education and training.

---

<sup>15</sup> <http://www.iso.org/iso/home.htm>, last accessed January 17, 2010

**AICC:** The Aviation Industry CBT (Computer-Based Training) Committee<sup>16</sup> is an international association of technology-based training professionals. The AICC develops guidelines for aviation industry in the development, delivery, and evaluation of computer-based training technologies. The term "AICC Compliant" means that a training product complies with one or more of the 11 AICC Guidelines & Recommendations (AGR's). AGR stands for AICC Guidelines & Recommendations. As the name implies, AGR's are technical recommendations. Each AGR makes a technical recommendation in a specific area such as; **CMI** - Computer-Managed Instruction, **CBT** - Computer-Based Training, **COM** - Communications, **CRS** - Courseware Technology, **EXC** - Executive Committee, **ITL** - Independent Test Lab, **MPD** - Media and Peripheral Devices, **PLT** – Platform, **WOS** - Window & Operating Systems, etc..

## **2.2. Providing a Structure**

What should be the desired software architecture for a graphical educational tool? Besides pedagogical perspectives there are other perspectives that should be taken into consideration when commenting on an educational modeling language. Software architecture is defined as “the structure or structures of a system, which comprises software elements, externally visible properties of those elements, and the relation between them” (Bass et al. 2003, Clemens et al. 2005). Therefore, one needs to pre-establish the system structure as the necessary building blocks that determine the user friendliness, efficiency and effectiveness of the software tool.

Bass et al. (2003) claims that the software architecture as the development product, which gives the highest return on investment with respect to quality, schedule, and cost. This is because an architecture appears early in a product's lifetime. If it is constructed right then it sets the stage smooth for everything to come in the system's life cycle as the development, integration, testing, and modification. If it is wrong then the fabric of the system is wrong, and it cannot be fixed by weaving in a few new threads or pulling out a few existing ones, which often cause the entire fabric to decompose and perish. Bass et al. (2003)

---

<sup>16</sup> [http://www.aicc.org/pages/aicc\\_faq.htm](http://www.aicc.org/pages/aicc_faq.htm), last accessed January 17, 2010

also mentions about the fact that analyzing architectures is inexpensive, compared to other development activities. As a summary, architectures give a high return on investment, because of the fact that: decisions made for the architecture have substantial important consequences, and checking and fixing an architecture is relatively inexpensive. Since life cycle of a good product starts with a good architectural design, one can say that successful product can influence how other products are built and cause better products to be developed.

On the other hand, Wilson et al. (2004) mentions the benefits of a reference architecture for educational organizations and orders them as such:

- A reference architecture provides better return on technology investment. New learning components can be developed or acquired when needed, which means that only those parts of an educational model that really needs to be changed are replaced, retaining the other components that comprise the model. Hence, purchasing and implementation costs are reduced.
- A reference architecture enables faster deployment of technology. As learning technology components are independent it will be easier to deploy new components as long as the new components are compatible with the existing interfaces. If the latter is not the case it may still be simpler to alter or to replace other learning technology components to supply the requirements of the e-Learning solution.
- A reference architecture provides a modular and flexible technology base. The rationale for the reference architecture is to enable the development of flexible educational models, in which individual learning components can be added or replaced more easily than in a solution without it.

Therefore, setting up the reference architecture is a must concern in software development life cycle. In order to ensure a better reference architecture and enforce the provision of a better and globally acceptable quality of an end product, there is an enormous activity of standardization is going on. As mentioned earlier, regarding the field of Education, Americans and Europeans

carry out such efforts. Australians and other communities of the world tend to support either Americans (IEEE, SCORM and AICC) or Europeans (ISO and IMS) or both when and where applicable.

But the fact of the matter is that, although there exist several standards like AICC-AGR-006 (computer managed instruction) or AICC-AGR-009 (icon standards), ISO/IEC 24738:2006 (icon symbols and functions for multimedia link attributes), IMS-LD or IEEE 1484, they do not directly apply to the graphical modeling of instructional processes. Therefore, as it is claimed by Knight (2005) inexistence of standards related to the graphical representations of educational models, one can easily say that there are many variations of graphical models of instructional processes in today's market. Karampiperis and Sampson (2007) paraphrase this fact and say that despite the wide adoption of the IMS LD specification still a common language for graphically representing learning flows is missing. Karampiperis et al. (2007) goes forward and offers BPMN (Business Modeling Notation standard) and adoption of BPEL (Business Process Execution Language) workflows into IMS LD Level-A authoring tools.

At the same time, developments in software engineering have brought new approaches to the design and development of information systems. Moreover, as complexities of educational systems increasing, resultant development times are getting longer and longer. Consequently, modern and better model development techniques are replacing traditional approaches.

#### 2.2.1. Component Based Software Engineering

When we consider the fact that current software systems are large-scale and very complex, as Dogru (1999) also admits, because of their high development costs and unmanageable software quality, development of such systems is not easy in an environment where technologies and requirements also change frequently. Therefore, the focus should be on composing and assembling components that are likely to have been developed separately, and even independently. Hence, components can be added, deleted, modified and/or changed easily without affecting the original model. This approach intends to accelerate software development and reduce development costs by using

prefabricated software components. Szyperski (1998, cited in Dogru, 1999) defines the components as units of implemented software building blocks.

Selecting various components and assembling them together rather than programming an overall system from scratch, develops component-based software systems. Therefore, development life cycle of component-based software systems is different from that of the traditional software systems. As Dogru (1999) says the idea of building systems completely with components and prepare models accordingly solve most of the software engineering problems.

Component Based Software Engineering specifies that component identification (selection), customization and integration are the main activities in the life cycle of component-based systems. This activity comprises two main parts. First, evaluation of each candidate component based on the functional and quality requirements and second, customization of that candidate component, which should be restructured before being integrated into the new component-based software system. Integration as being the last phase of modeling effort means that key decisions can be taken on how to provide communication and coordination among various components of the target software system. When we apply this principle to the instructional modeling and think about it from the educational perspective; pedagogical approach will be leading the designer to make such decisions for the integration phase.

### 2.2.2. COSE Approach and Education

One of the most promising system modeling approaches today is the component-based software development approach. Any component-oriented approach assumes that functions have been implemented in components. Therefore, the system development problem reduces to locating and connection of pre-defined components. Utilization of the components is the main concern when developing the respective model of the system. Despite reported difficulties in their utilization, components are meant for run-time connecting modules. Also when more semantics will be included as part of the component interface specifications in the future, this will definitely aid in locating and integration of components.



Learning systems can also be individually assembled by using the components to provide the functionality that instructional designers really need. ECOML enables instructional designers to graphically model various teaching/learning resources (=components) and incorporate them into their instructional content, speeding up the preparation and design process of the learning system and significantly reducing development cost and time.

Therefore, main concern should be to locate and retrieve the needed component. Software organizations organize the independently developed components in a component repository. The question of “How are the components represented in and retrieved from a repository?” is resolved by the formal specification that describes functional properties of components. This formal specification requires the logical description of the semantics of the software component. Hence, component retrieval is based on the semantic match, and it is based on using formal specification used in the component representation. It is believed that software components, which reside in the repository, should meet the requirements below to be reused easily by the designers, which are also stated by Kara (2001):

- The components should be classified and stored in different component libraries according to their semantics and domains.
- The components should be useable in many contexts.
- The components should be efficient to meet the system performance requirements.
- There should be enough components in a component library such that the designers can find all components needed.
- The components should be independently upgradeable.
- The components should be customizable on the appearance and behavior of them.
- The components should be interconnectable to generate larger components or complete applications.
- The components should know the time, namely, design-time or run-time. Because, at design-time, the component should expose its

property editor and event editor according to some signals. However, at run-time, it should expose its behaviors according to same signals.

The libraries, subroutines, objects, and components are the examples of reusable software. Component Based Software Engineering (CBSE) and Component Oriented Software Engineering (COSE) seem to be two similar approaches for system's modeling that are based on reusable software components. But, because of the fact that educational components are not like software components, which can be added, deleted or changed according to the functional needs of the system, the educational modeling efforts incline more to the COSE approach rather than CBSE.

From the educational point of view the components can be units of learning, operational knowledge units, learning objects, and learning components (which includes all similar educational definitions), as we have defined earlier. There are already established repositories and web services, which are hosting such educational components (e.g. government web sites like <http://internettv.meb.gov.tr>, <http://egitim.gov.tr>, <http://www.free.ed.gov/>, Learning Resource Exchange<sup>17</sup> portal (provided by European Schoolnet<sup>18</sup> together with 20 Ministries of Education from Europe, and it offers more than 130 000 learning objects and assets in more than 20 languages), The Learning Object Repository Network<sup>19</sup> (LORN, Australia's national learning object repository network for the vocational education and training (VET) sector, which contains more than 3000 learning objects) or digital libraries like <http://www.computer.org/portal/web/csdl>, etc. etc.. It is for sure that affluence of such services will need respective UDDI's being prospered in the future.

Since, COSE approach is based on structural decomposition of the system, it enables the analysis and design phases of instructional processes in a higher level of abstraction with graphical representations of various educational components. Changes in the requirements of the system can be easily handled by

---

<sup>17</sup> <http://lreforschools.eun.org/>, last accessed January 05, 2010.

<sup>18</sup> <http://www.europeanschoolnet.org/>, last accessed January 05, 2010.

<sup>19</sup> <http://lorn.flexiblelearning.net.au/>, last accessed January 05, 2010.

changing graphical symbols and relational definitions. Unlike other modeling approaches, the pedagogical model, itself, is not affected much from the changes.

Regarding the above mentioned issues of component oriented software engineering, this researcher thinks that educational models can be created in such a way that any teaching/learning scenario can be designed and implemented with components as being OKUs, UoLs, LOs, and LCs. This approach definitely reduces the course development efforts and improves maintainability of the pedagogical model. Hence, this researcher thinks that selecting various learning components and assembling them together with the help of a modeling tool can easily develop component-based educational systems. That's why preparation of future teaching/learning designs will be different than traditional ones. Advancements in Internet technologies and UDDI related web services will bring new design tools, which will make use of the new technologies for easier, better and more efficient course designs. Certainly, ECOML is one them.

### 2.2.3. Instructional design and the ECOML approach

Dogru (1999) described the software process in COSE approach as:

- structural decomposition starts in a top-down manner with dividing the process into logical modules,
- this process continues until reaching the existing components,
- then those components are integrated in a bottom-up fashion in order to build the system.

Because of relational dependencies between learning components of any educational process, similar hierarchical approach is also suitable for instructional modeling efforts. One can find such building blocks in any teaching/training and/or learning effort. Based on the experience level of the instructional designer, and the applied pedagogical method, instructional process can be divided into logical modules. Then related components are identified and they are integrated in a bottom-up fashion for the complete system.

Finally, almost all pedagogical approaches mention about internalization of information in order to construct the "knowledge". Stephan Morris (1994) also admits that construction of knowledge is hierarchical, "belief", and thus

"knowledge", is interpreted as properties of individuals' preferences (that's why, in principle, they are observable choices). Every piece of information is dependent on previous piece that is already internalized. This description of knowledge construction makes instructional designs perfectly suitable for hierarchical modeling tools. COSEML is a good example of such a modeling tool. Hence, ECOML, which is an offspring of COSEML, places itself into that category of CASE tools, as well.

### **2.3. Taxonomies**

As Yongwu et al. (2009) put forward many learning design languages have been developed recently. Yongwu (2009) categorizes the learning designs by saying Falconer and Littlejohn (2008) distinguish two categories of learning designs in a recent study:

- those meant for learning (the executable design) and,
- those meant for teaching (the inspirational design).

The audience of the former learning designs is a machine, and the audience of the latter learning designs is the teacher. The first version of ECOML is meant for inspirational design, as is the case for authoring editors of executable designs. Future versions of ECOML will consider the executable designs, as well.

That's why, the taxonomy, aims at providing the teacher with a classification that is arranged in a hierarchical structure and covering a range of options (at a greater or lesser level of detail). Depending on the particular pedagogical approach, teacher typically organizes course material by parent-child relationships, such as processes of weekly teachable materials or problem/case oriented materials and individual OKUs, UoLs and LOs as parents then related sub-materials (again can be composed of further OKUs, UoLs and/or LOs) can be defined as siblings. Of course, in such a structural relationship, the subtype by definition should have the same constraints as the super type similar to inheritance, or mostly, the relationship can represent composition. Obviously, the lowest level of the hierarchy contains only LOs or LCs. That is to say, individual OKUs and UoLs ought to be designed in accordance with the pedagogical approach used in the methodological representation of the parents, i.e. time wise,

process wise or case/problem wise, etc. The teacher can then use this as a guide when creating a Learning Design.

On the other hand, Instructional Design (ID) is concerned with the processes to produce good specifications of learning experiences (“Smith, P.L., and T.J. Ragan, *Instructional Design* (2<sup>nd</sup> edition) Wiley & Sons Publishers, 1999” as cited in Caeiro Rodriguez, M.; Anido Rifon, L.; Llamas Nistal, M. (2004)). Caeiro et al. (2004) say that instructional design of a course is a top-down activity that starts with the teacher gaining understanding of the learners and their goals, the resources and services available, the pedagogical approach, etc. Which means, as constructors of courses, teachers design their believes, knowledge and experience in hierarchical manner and relay them to the learners.

Caeiro et al. (2004) continue to describe the process by giving the example of a course; the design continues with organizing the activities that have to be carried out by learners and academic staff. In the next step, it is necessary to define the control flow between the different activities, and for each activity to describe the actors involved, the environment (with its resources and services, and the properties, conditions and events associated) the way in which the actors are going to interact, communicate, etc.

The effort of the teacher to identify the primitives and to place them in a hierarchical structure may be a powerful course development activity. Providing teachers with graphical software tools to accomplish the design and representation of such educational structures is definitely an improvement, which ICT brings for them. Also, as Blackwell (1997) and Lewalter (2003) say visual tools allow a synthetic representation of complex objects and reduce the cognitive load of teachers and help them organize their work both mentally and physically.

Development of visual instructional design languages is a fairly recent research trend in the field of educational technology. Koper (2001) says that it all started with a research project aimed at building a semantic notation for complete units of study to be used in e-learning, in the Open Univesity of the Netherlands in 1998.

The notion of units of study was called as an “Educational Modeling Language” at that time. The idea evolved by the time and several educational modeling languages developed.

Sampson et al. (2005) categorized visual instructional design languages into two; form-based and graphical-based. Form-based Learning Design authoring tools are tools that provide form-based interfaces for the definition of the learning scenarios. The main advantage of these tools is that they provide direct control of the Learning Design information model elements. However, as Sampson et al. (2005) said they are rather difficult to be used by less experienced designers and they require pre-processing of the structure of the desired scenario in order for the designer to be able to express it directly in XML notation. Examples of such tools include the reload project<sup>20</sup>. On the other hand, graphical-based Learning Design authoring tools are tools that provide drag-and-drop interfaces for the definition of the learning scenarios. Their main advantage is that they support the definition of a pedagogical scenario without requiring pre-existing knowledge on the details of the IMS Learning Design information model. Examples of such tools include ASK-LTD.

One of the objectives of this study is to introduce yet another one of new professional tools for instructional designers. Therefore, one can say that the main concern is the development of a meta-model of a pedagogical approach. If the tool is graphical-based with ergonomic graphical user interface (GUI) then it will serve the purpose better. It is also desired that the tool should be general enough to cover most of the pedagogical approaches, if not all.

---

<sup>20</sup> [www.reload.ac.uk/](http://www.reload.ac.uk/), last accessed November 14, 2009.

## CHAPTER 3

### RELATED WORK

A pedagogically independent tool that is valid for different instructional designs may well be a starting point for instructional designers. To focus on the modeling activity, without explicit knowledge about the underlying design theories and pedagogies, is certainly a big advantage for efficient instructional preparations. Besides that, inspection and commenting on easy-to-understand graphical models will be more practical and effective.

Significant R&D effort has been kept devoted towards modeling educational systems tailored to specific pedagogical approaches. The proposed modeling languages attempted to provide a formal way of representing the educational process in commonly agreed manners. But, as Caeiro et al. (2004) say that there are abundance of learning theories and approaches for instructional designs in the learning domain. This thesis study tries to list all educational modeling languages, which are discussed within the related literature that this researcher could access. The current version of ECOML, as being a *graphical editor* (or, according to the naming conventions stated in the IMS's *unfold\_d9\_sustainability\_plan*<sup>21</sup>, it can also be named as a *graphical authoring system*), represents top down hierarchical approaches as a better modeling method for educational instruction designs. Since, the researcher believes that hierarchical approach is more appropriate for instructional modeling, he considers only the domain of hierarchical designs and tries to comment on them within the scope of this thesis.

---

<sup>21</sup> [http://dspace.ou.nl/bitstream/1820/639/1/Unfold\\_d9\\_sustainability\\_plan\\_14feb06.pdf](http://dspace.ou.nl/bitstream/1820/639/1/Unfold_d9_sustainability_plan_14feb06.pdf), last accessed July 02, 2009.

### 3.1. What is IMS-LD?

IMS-LD as being the latest, most comprehensive, and most controversial pedagogical standard, and an educational modeling language, which describes technology supported pedagogical scenarios based on instructional design models. EduWiki<sup>22</sup> claims that IMS-LD currently represents the most popular formal language to describe learning designs.

According to the IMS Learning Design (LD) specifications, “The core concept of the LD is that, regardless of the pedagogical approach, a person gets a role in the teaching-learning process, typically a learner or a staff role. In this role he or she works toward certain outcomes by performing more or less structured learning and/or support activities within an environment”. The particular characteristics of the roles, which a person takes on, the activities to be carried out, and the particular characteristics of the environment define a specific learning scenario. This learning scenario can be represented in IMS Learning Design, where it is called a Unit of Learning (UoL). Koper (2001) also defined the smallest unit as the ‘unit of study’, which is providing learning events for learners, satisfying one or more interrelated learning objectives. A unit of study could be delivered through what is called:

- online learning (completely through the web).
- blended learning (mix of online and face-to-face)
- hybrid learning (mix of different media: paper, web, e-books, etc.).

The UoL can then be run on any Learning Design compliant system. IMS claims that Learning Design does not offer a particular pedagogic model or models, but can rather be used to define a practically unlimited range of scenarios and pedagogic models. Because of this it is often referred to as a *pedagogic meta-model*. Learning Design was developed in the context of e-learning, but there is no reason why Units of Learning cannot be used in mixed face-to-face and online learning contexts, or in entirely face-to-face learning (IMS, 2003).

As can be easily deduced, above requirements of IMS-LD are not easy to comply. As Caeiro et al. (2004) also agrees, because of the fact that IMS-LD aims

---

<sup>22</sup> [http://edutechwiki.unige.ch/en/IMS\\_Learning\\_Design](http://edutechwiki.unige.ch/en/IMS_Learning_Design), last accessed December 09, 2009.



to be pedagogically neutral, which is extremely difficult to attain considering the fact that there are abundance of them.

Neumann and Oberhuemer (2009) also admit the same difficulty by saying that what the language IMS LD offers is hard to understand, and that it takes considerable effort to apply. As Barrett-Baxendale says (2007, as cited in Neumann et al. 2009) without sufficient technical support, instructors cannot easily create Units of Learning as it is described in the IMS-LD.

### **3.2. Authoring Tools**

IMS classifies educational software tools with respect to their functional properties:

- Engine, (e.g. Coppercore ).
- Editors, (e.g. Reload editor, ASK LTD, Copperauthor, MOT+, Cosmos).
- Players, (e.g. Reload LD Player, Coppercore Player).

Although there are other software tools that do not comply with IMS specifications, like LAMS<sup>23</sup> (which is both an editor and a player)<sup>24</sup>, the current version of ECOML can be classified as an editor in that respect.

According to IMS-LD, engines, editors and players can be further grouped into three categories:

- Level A; contains the core elements of the meta-language. Described as series of time ordered learning activities to be performed by learners and teachers, using learning objects and/or services.
- Level B; enables the use of generic properties and conditions (i.e. properties (storing information about a person or group), and conditions (placing constraints upon flow)).
- Level C; provides the ability to use notifications, enables activities to be set dynamically (i.e. notifications, triggered events - e.g. student asks a question, the teacher needs to be notified that a response is needed).

Therefore, current version of ECOML is a “Level A” editor.

---

<sup>23</sup> <http://www.lamsinternational.com/>, last accessed December 08, 2009.

<sup>24</sup> <http://moodle.org/mod/forum/discuss.php?d=24384%20%28June,2005%29>, last accessed December 08, 2009, logged in as a “guest”.

It was only in 2005 that the first IMS Learning Design editors and players became available, and were first presented at the UNFOLD meeting in February 2005<sup>21</sup>. Since then IMS Global Consortium makes surveys about *educational technology products* and repeats the survey every year. The latest survey lists 837 of such products<sup>25</sup>. And 266 out of 837 educational technology products are software tools<sup>26</sup>. The survey lists the major products; their descriptions, functions, advantages and disadvantages, then ask the participants to give feedback out of a 7-point Likert scale (Very dissatisfied, moderately dissatisfied, slightly dissatisfied, neutral, slightly satisfied, moderately satisfied, very satisfied). 130 out of 266 has received “slightly satisfied” to “very satisfied” ratings from the participants. And, only 13 out of 130 listed products are graphical editors.

Table 1. Authoring Tools as compiled by IMS-UNFOLD

Name	Producer	Purpose	Ownership
<b>Higher Level Editors</b>			
MOT+	University of Quebec	Graphical editor	Open source
eLive	ELive GmbH Germany	Specialized editor	(Not released)
EduPlone Sequencer	EduPlone	Specialized editor	Open source
EduCreator Editor	Chronotech	General purpose editor	(Not released)
LAMS	LAMS foundation	Learning activity management system	Open source
Dialog+ toolkit	Dialog+	Learning Activity editor	Open source
Collage	University of Valladolid	Learning design authoring	Open source
<b>Lower Level Editors</b>			
aLFanet	AL.Fanet project	Tree editor	(Not released)
ASK-LDT Editor	Informatics & Telematics Institute Greece	Graphical editor	Freeware
CopperAuthor Editor	Open University of the Netherlands	Tree based editor	Open source
Komposer	GTK Press	Tree based editor	Under development
COSMOS editor	Yongwu Miao, University of Duisburg	Tree based editor	Open source
RELOAD Learning Design Editor	Reload project (JISC)	Tree based editor	Open source

<sup>25</sup> <http://www.imsglobal.org/productdirectory/directory.cfm>, last accessed July 02, 2009.

<sup>26</sup> <http://www.imsglobal.org/productdirectory/products.cfm>, last accessed July 02, 2009.

IMS Consortium classified the editors that have been considered in the unfold\_d9\_sustainability\_plan, which includes all significant development initiatives about which the UNFOLD project is aware of.<sup>27</sup> This classification is shown in Table 1.

Botturi (2004) also mentions visual aids in instructional designs. Botturi evaluated two such modeling languages that were available at that time. These were E<sup>2</sup>ML (Educational Environment Modeling Language) and CADMOS-D, which represent the educational activity and models that support the representation of the learning goals, the instructional activities and/or the learning materials. Later on, Schneider (2009) also evaluated educational modeling tools and said that he didn't find any stable, user-friendly and easy to install LD runtime environment as of May 2009<sup>14</sup>. He continued with claiming that the only popular learning design system environment seems to be LAMS<sup>28</sup> (but it is not based on IMS LD).

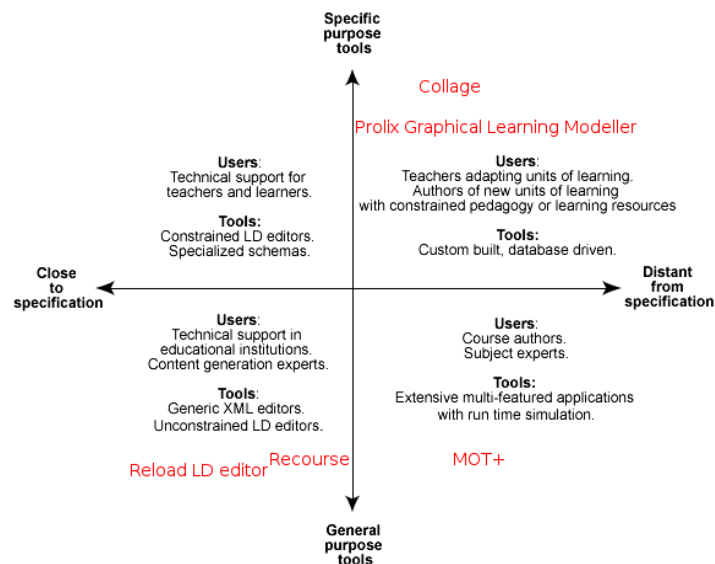


Figure 3. Learning Design Systems (Schneider, 2009)<sup>14</sup>

<sup>27</sup>[http://dspace.ou.nl/bitstream/1820/639/1/Unfold\\_d9\\_sustainability\\_plan\\_14feb06.pdf](http://dspace.ou.nl/bitstream/1820/639/1/Unfold_d9_sustainability_plan_14feb06.pdf), last accessed July 02, 2009.

<sup>28</sup> <http://www.lamsfoundation.org/>, last accessed December 06, 2009.

On the other hand, Griffith et al. (2005, as cited in Schneider, 2009) defined two dimensions with respect to LD tool designs. The two dimensions are:

- Close to specification - distant from specification,
- General purpose tools - specific purpose tools.

Later on Oberhuemer (2008, as cited in Schneider, 2009)), using Griffith's approach, refined and developed the categorization of LD tools. Schneider (2009) reproduced Oberhuemer's categorization of LD tools with some currently available LD authoring tools. Oberhuemer's approach and Schneider's inclusion of sample LD tools is depicted in Figure 3.

Schneider (2009) criticizes the classification because it implies typical teachers are not supposed to learn a general purpose LD tool. Useful tools for course authors seem to be distant from the specification, in the sense that "real" users should not understand IMS LD and that their designs are just compiled into IMS LD. Then he claims that if such tools could import (and not just export IMS LD), IMS LD would turn into a sort of "assembly" language of pedagogical designs. But this raises the question of how designs could be described with a common high-level language in order to promote exchange, i.e. why we should still need "mid-level" languages? And, he is searching an answer for "would one want to be able to author with the same tool several types of activities within a larger course-level design?" That is to say, does the educational world need to distinguish between general-purpose tools and specific tools (e.g. a Computer-Supported Collaborative Tool (CSCL) like Collage<sup>29</sup>)?

Sodhi, Miao, Brouns and Koper (2007), as being the forefront supporters of IMS-LD specifications, also admit that there is a deep conceptual gap between the needs of the non-expert authors and the support that is afforded to them in Today's IMS-LD authoring tools. Therefore, this researcher has tried to select, to the best of his current knowledge, the two editors that are user-friendly.

### **3.3. Comparison of two authoring tools.**

This researcher selected the following two authoring tools for evaluation:

---

<sup>29</sup> <http://edutechwiki.unige.ch/en/Collage>, last accessed December 09, 2009.

- Reload editor. Selected based on the information given in Figure 2. Reload represents the tools closest to the “general purpose” and it is also the one that is closest to the IMLS-LD specifications. It is a reference implementation of IMS-LD<sup>30</sup>.
- The other selected editor is the ASK-LDT (ASK Learning Design Toolkit). Because it is a hierarchically represented graphical-based LD authoring tool. ASK-LDT’s starting point is the Business Process Execution Language (BPEL), which is an XML based language that represents a business process (Karampiperis and Sampson, 2007) and based on the graphical design elements and standards of Business Process Modeling Notation (BPMN). BPEL was developed to enable business user to model business processes in an understandable graphical representations (White S.A., 2005). White (2004) also says that BPMN is based on a flowcharting technique tailored for creating graphical models of business process operations, which means in essence it supports hierarchical representations.

Both tools are IMS-LD specifications conformant and general purpose.

### 3.3.1. ReLoad LD Editor

ReLoad (Reusable eLearning Object Authoring and Delivery) is a software tool to create IMS Learning Design compliant Units of Learning. It is free, open source and cross-platform. It was developed as part of the European-funded TENCompetence project<sup>31</sup>. There are two separate products, which can work integrated; Reload LD editor and the Reload LD player. ReLoad LD editor is the modeling tool that can export a zip file (called as Learning Design Package) with all the files related with the model already depicted in. The ReLoad LD Player is the player tool that can accept the exported zip file to execute the modeled Learning Design. The editor’s latest version was released in January 2009. According to IMS-LD specifications ReLoad editor is a Level A editor (i.e. a

---

<sup>30</sup> <http://lemill.org/trac/wiki/ReportLearningDesignAnalysis>, last accessed December 23, 2009.

<sup>31</sup> <http://www.tencompetence.org/ldauthor/>, last accessed December 06, 2009.

series of activities (assessment, discussion, and simulation), performed by one or more players (learners, teachers etc.) in an environment consisting of learning objects or services).

Version 2.1.3 of ReLoad Learning Design Editor is downloaded from its main web site (<http://www.reload.ac.uk>). This editor is implemented with Eclipse, needs Java installed on your computer and supposed to run on all platforms. Unfortunately, in order to use it effectively users should be familiar with the IMS-LD language, i.e. understand its logic and the purpose of its most important elements.

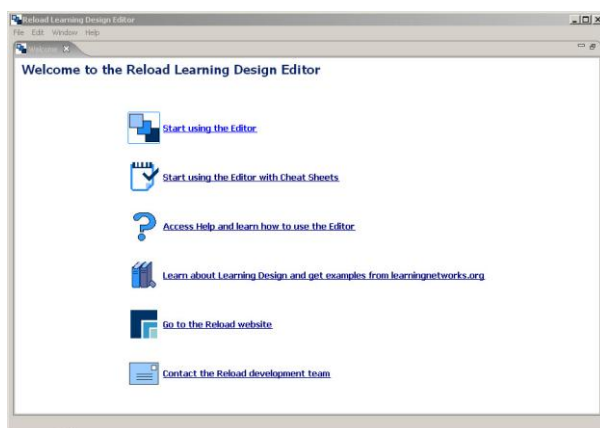


Figure 4. Welcome Page of the ReLoad LD Editor

Although, it is claimed that ReLoad editor is a tree-based editor by IMS (see Table 1), this researcher was not able to figure out how it hierarchically represents the instructional models (see Figure 5). Internal representation and data structures may be implemented with tree-based methods, but the constructed and displayed instructional model is not a hierarchical one, in regard to common software terminology. Therefore, this researcher claims that Reload Editor is a form-based editor (see Figure 5). It is not possible to represent learning units and/or learning objects with graphical symbols and show them in a hierarchical tree-based model. Another important aspect is that, because of its compliance

with IMS-LD specifications, users must have enough knowledge about the IMS-LD language. That's why welcome page of the editor contains help entries for inexperienced users to learn more about Learning Design and other related IMS-LD concepts (see Figure 4).

Although IMS claims that the ReLoad LD Editor is a Level A editor, usage of the editor permits to designate the model in Level A, B or C (see Figure 5). A ready made sample, which was obtained from the web site of ReLoad editor was imported to execute at the ReLoad LD Player. Surprisingly, the attempt was not successful, because of the reason that the Player gave errors during parsing the designated zip file. Execution of the finished models is beyond the scope of this thesis study. Therefore, further work on why the attempt failed on the sample file, which was recommended by the official ReLoad web site, was not researched and finalized for any conclusion.

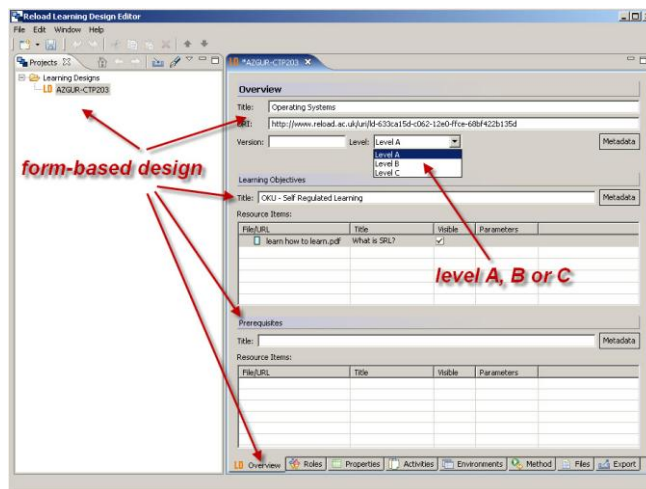


Figure 5. ReLoad can be used as either Level A or B or C editor

For every Learning Design (i.e. instructional model) the ReLoad editor keeps a separate xml file and distinguishes them by assigning different project names on the monitor. That is to say, although ReLoad Editor shows every Learning Design on the same screen, it treats them as independent and separate

entities. Which exhibits another weak point of the ReLoad editor when compared to ECOML. It is not possible to define a certain project within another one, in ReLoad Editor (see Figure 6). For instance, any LD can be designed as an independent OKU part in any instructional model then this independent OKU can be assigned as a Learning Component in another model, even it can be repeated as separate LCs in the same model. This recurrence of independent Learning Components is not possible in ReLoad Editor while such a structure is perfectly representable in ECOML.

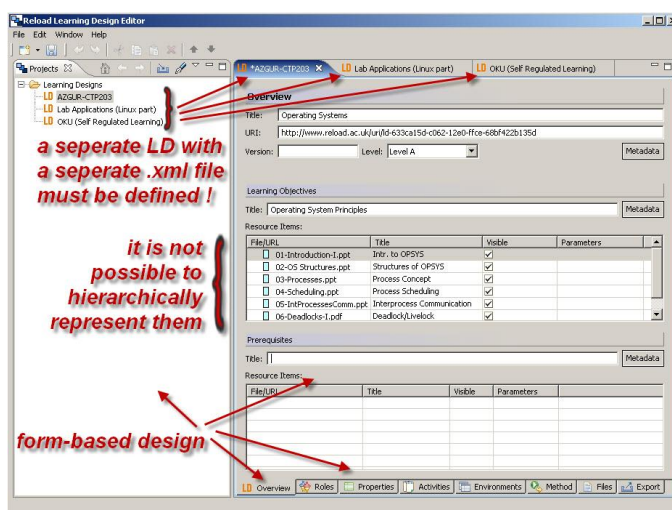


Figure 6. Weak points of ReLoad LD editor

ReLoad is also criticized with its role assignment capability<sup>32</sup>. According to IMS-LD role assignments and corresponding activity assignments are possible. IMS-LD defines two roles teacher (staff) and learner (student). However, if an instructional designer had two main roles for students: Student A and Student B, s/he therefore knew that s/he would have to add two new Learner roles. If the roles where not assigned in the Roles interface section, it would not be possible to assign the different activities to the different role-parts in the method section.

<sup>32</sup> <http://lemill.org/trac/wiki/ReportLearningDesignAnalysis>, last accessed on December 23, 2009.



This might be considered as an interface design problem in the Reload editor but it could also come from the way roles are handled in IMS LD.

There are some other criticized points of the ReLoad editor but, they are usually related to its connection with the LD players, which is out of the scope of this thesis study. As a result, evaluation of ReLoad LD editor indicated that it offers an additional product that can be integrated for playing the instructional models. Its form-based construction is easy to use once the user is familiar with the IMS-LD terminology. Therefore, it may create some challenges for inexperienced designers and classroom teachers.

Additionally, ReLoad's inability to represent instructional models hierarchically with graphical symbols makes ECOML a better modeling tool for instructional designers.

### 3.3.2. COSMOS ASK-LTD Editor

Sampson et al. (2005) defines the ASK (Advanced eServices for the Knowledge Society Research Unit)<sup>33</sup> Learning Designer Toolkit (ASK-LDT) as an authoring tool based on the use of IMS LD Level B specification that provides the environment for a pedagogical designer to define complex learning scenarios.

ASK-LDT was presented with real-time examples during the Unfold project meeting in Barcelona, Spain, during 20-22 April 2005<sup>34</sup>. This researcher was not able to find out any installable version of the original ASK-LDT software tool, although it is claimed by Sampson et al. (2006) ASK-LDT is funded by the European Union Commission, Information Technology Society Programme, FP6 Project (ICLASS contract IST-507922). All the links given in the above mentioned project meeting (and others elsewhere) were found to be broken, including the one ~15x10<sup>6</sup> Euro-worth iClass project's link<sup>35</sup>. ASK\_LDT is supposed to be developed in the context of the iClass project<sup>34</sup>. Project leader is Greek's Informatics and Telematics Institute's CERTH (the Center of Research

---

<sup>33</sup> <http://www.ask4research.info/new/index.php>, last accessed December 23, 2009.

<sup>34</sup> <http://www.unfold-project.net/project/events/cops/bcna2/overview>, last accessed December, 12 2009.

<sup>35</sup> (broken link): <http://www.iclass.info/>. iClass: Intelligent distributed Cognitive-based open Learning System for Schools. Project was funded by European Commission, Information Technology Society Programme, FP6, during the period 2004-2008.

and Technology Hellas)<sup>36</sup>. Original web page of the ASK-LDT is the CErTH's [http://www.ask4research.info/products\\_toc.php](http://www.ask4research.info/products_toc.php). But, neither iClass nor ASK-LDT's home page has got any live www link on the respective web pages where it is introduced. Only documents related to its development and pictures of its menu dialogue pages exit in the literature. Even Schneider (2009) couldn't figure out whether this language/system is still alive or being further developed.

The evaluated copy is downloaded from the European Union's Cosmos Portal<sup>37</sup>. Version 1 of the installer carries the last update tag as 19/05/2008. According to the information that is given in the COSMOS web site<sup>38</sup> the COSMOS Learning Activities Authoring Tool is based on *ASK Learning Design Toolkit* and enables the definition of Learning Activities implementing the COSMOS Generic Technology-Enhanced Educational Scenario Templates. Learning Activities created with COSMOS ASK-LDT are stored in zip format and can be viewed with the Reload Learning Activity Viewer. Therefore COSMOS ASK-LDT itself is not a Level B tool.

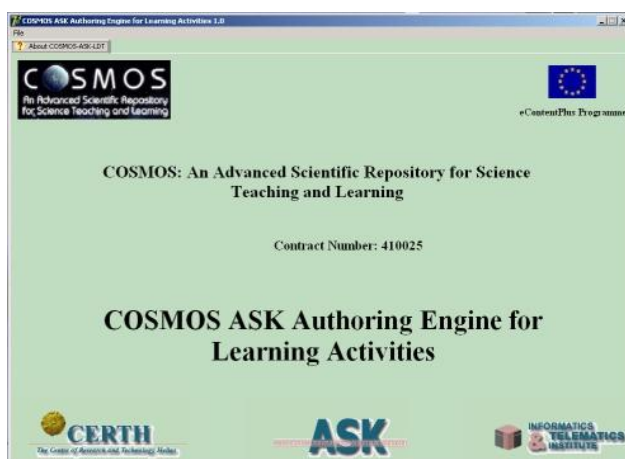


Figure 7. Main menu and the first dialogue page

<sup>36</sup> <http://www.ask4research.info/project.php>, last accessed December 12, 2009.

<sup>37</sup> <http://www.cosmosportal.eu/cosmos/toolbox/cosmostools>, last accessed December 12, 2009.

<sup>38</sup> <http://www.cosmosportal.eu/cosmos/node/3498>, last accessed December 12, 2009.

Figure 7 indicates the opening page, which shows the main menu and the first dialogue page of the COSMOS ASK-LTD Editor. As it can be easily verifiable on the first dialogue page, COSMOS ASK-LDT is a product of Informatics and Telematics Institute's CERTH program.

Authoring with COSMOS ASK-LDT involves five steps:

- Create New Educational Scenario: during this step the user can create a new educational scenario based on six pre-defined Educational Scenario Templates, namely the “Guided Research Model”, “Inquiry-based Teaching”, “Project-based Learning”, “The 5E Instructional Model”, “The Learning Cycle (Supporting Conceptual Change)” and the “ICT supported Culture Awareness Learning (ICCAL)”. Preparation of a model based on any other pedagogical approach is simply not possible.
- Assign Resources to Learning Activities of the Educational Scenario: Once the educational scenario is selected then the user can assign resources (html pages, images, etc.) to learning activities of the educational scenario or change the existing ones.

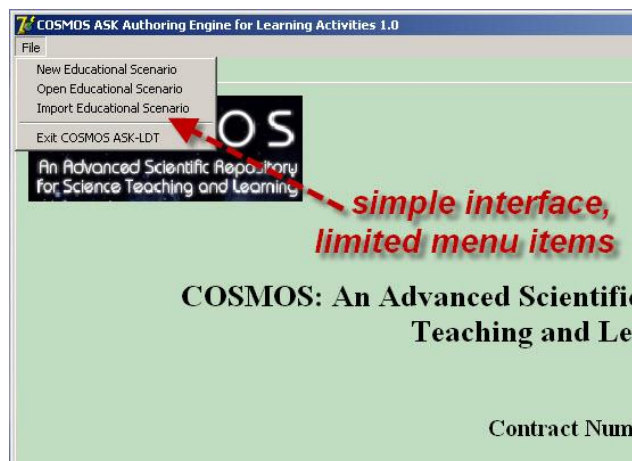


Figure 8. It is possible to import IMS compliant LDs.

- Open Educational Scenario (Optional). The user can open the previously created educational scenario that is already stored in the local repository of the COSMOS ASK-LDT. Then re-work or continuation of the earlier work can be carried out easily.
- Content packaging. The user can save an educational model as a Content Package (zip format), which conforms the IMS LD specifications. This ability makes the COSMOS ASK-LDT to be playable in any IMS-LD compliant player. Since players are not within the interest area of this study, validity of the exported files are not verified with respect to different IMS-LD players. Besides that, unfortunately, “Create Content Package” part of the editor was not possible to activate. The button remained dimmed no matter what type of information is entered. Therefore, even it is considered as a valid extension of the editor, it was not possible to comment about the file that is supposed to be created by the editor.

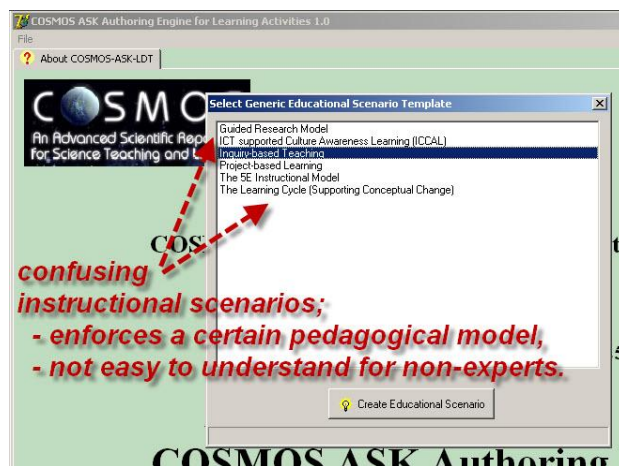


Figure 9. Menu that shows selection of possible scenarios

- COSMOS ASK-LDT official help document also describes the product as “The COSMOS ASK-LDT is used in order to design educational scenarios based on pre-defined educational scenarios

templates and generate packages conformant to the IMS Learning Design specification”.<sup>39</sup> Therefore, COSMOS ASK-LDT is not immune to pedagogical methods and enforces the user to a selected few pedagogical approaches. It can be easily said that it is not for non-expert educational designers (See Figure 9).

- The resulting UoL makes use of IMS Learning Design Level B, but the tool does not set out to provide a general purpose Level B authoring environment (See figure 10).

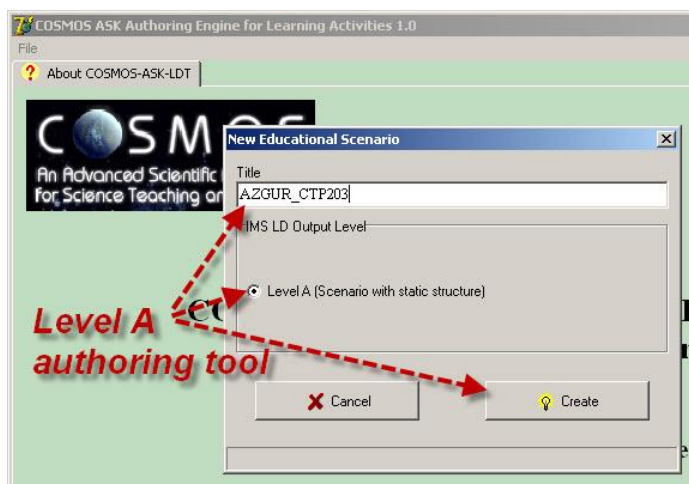


Figure 10. Main menu and first dialogue page of ASK-LDT

- COSMOS ASK-LDT offers a relatively easy path into Learning Design authoring (although it is intended as a tool for learning designers rather than classroom teachers). Therefore, there is no “help” provided inside the tool. The menu items are not self-explanatory (e.g. names of the pre-defined educational scenarios are not easy to understand for non-experts, nor any help button provided

<sup>39</sup> [http://www.cosmosportal.eu/cosmos/files/help/Learning\\_Activities\\_Authoring.pdf](http://www.cosmosportal.eu/cosmos/files/help/Learning_Activities_Authoring.pdf), last accessed December, 12 2009.

for run-time explanations). Consequently, ASK-LDT and its off spring COSMOS ASK-LDT is not for non-experts (see figure 11).

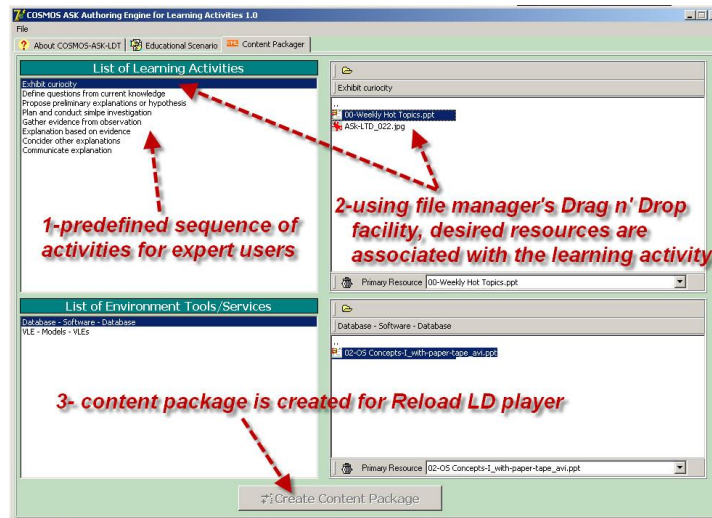


Figure 11. “Content Packager” dialogue page

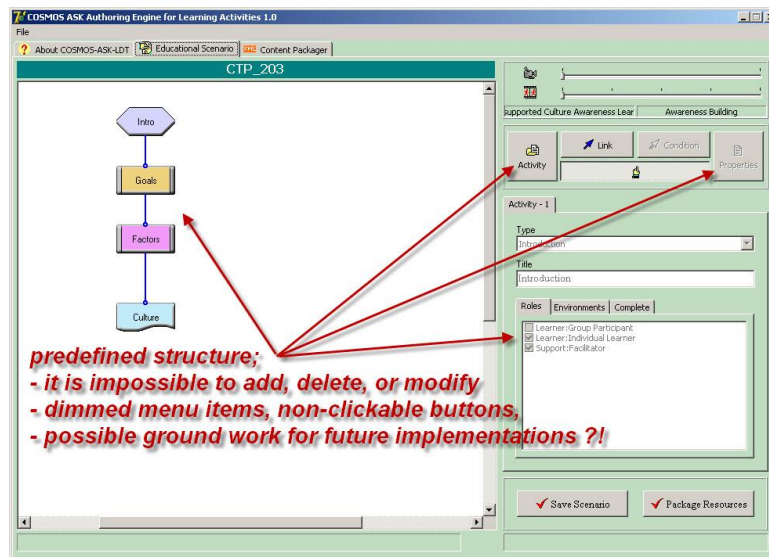


Figure 12. “Educational Scenario” dialogue page

- COSMOS ASK-LDT offers also an “Educational Scenario” dialogue page, although it is not possible to edit the *given* scenario. Inspection of the items on the dialogue page indicated that the graphical symbols are old-fashioned, far from today’s 3D and/or animated graphical icons. Secondly, although ASK-LDT claims it is a hierarchical modeling tool, it does not have the ability to show the overall graphical representation of the model (i.e. activities with learning components, see Figure 12).

Evaluation of COSMOS ASK-LDT indicated that it seems relatively easy to develop Learning Design models, since it offers pre-formatted educational scenarios. But, it is intended as a tool for learning designers rather than classroom teachers. Because of unavailability of help menu items, it turns into a rather difficult and user-unfriendly tool when used by non-experts. Although, it claims its main function is to prepare IMS-LD conformant “Content Packages” so that the design can be played later with another IMS-LD conformant player, because of non-functional button it was not possible to create any file.

Therefore, the impression of this researcher is that even though its much publicized and published literature, the inspected COSMOS ASK-LD version is not serving as a general-purpose instructional design modeling language neither it is a hierarchical modeling tool in the sense that ECOML offers to the instructional designers.

## **CHAPTER 4**

### **A MODELING TOOL FOR INSTRUCTIONAL DESIGNERS**

The educational system is primarily focused on the question of "What should people learn?" and from this the answer to "What should instruction attempt to teach?" is driven. Then, the next question of "How do we teach?" that most educational systems address may partially be derived from the answers of the first two questions. Although, in general, answers to the earlier questions seem to be independent from the pedagogical method, in reality they are directly related to the applied pedagogical philosophy and method that is used in that educational system. While this philosophy attempts to form a comprehensive view of what would be useful for an individual to learn, it also dictates how this information will be relayed to the learner, in practice.

Therefore, the core of the philosophy is the model that humans interact within the system (and further this system often interacts with others) to exchange information under certain methodological restrictions. Consequently, humans have devised various mental and physical tools to study and influence the system that they can make use of while interaction. Hence, in order for humans to learn about the system, they are required to learn the tools that will empower them to manipulate that system. Eventually, the success of the tool and its user will depend upon the efficiency and the effectiveness of the tool and what it offers to its users.

Choosing the right modeling tool can be quite a challenge. Already, there are many instructional modeling products to choose from on the market today. How can one choose? This chapter elaborates the topic and tries to list the helpful decisive factors, which are found in the literature. Meanwhile, ECOML's answers to those factors are also demonstrated.



#### 4.1. Overview of Requirements

Educational system is not an exception and the tools that are created to model it are rather new and still developing. Koper (2001) suggests the following requirements for an educational modeling language, which describes a unit of study:

1. The notational system must describe units of study in a formal way, so that automatic processing is possible (formalization). *The current version of ECOML satisfies the requirement by creating an IMS-LD complaint imsmanifest.xml file for possible export to an IMS-LD player.*

2. The notational system must be able to describe units of study that are based on different theories and models of learning and instruction (pedagogical flexibility). *The current version ECOML does not impose any pedagogical method to start with in its modeling structure for an instructional designer. Any pedagogical approach should be possible to be modeled with in ECOML.*

3. The notational system must explicitly express the semantic meaning of the different learning objects within the context of a unit of study. It must provide for a semantic structure of the content or functionality of the typed learning objects within a unit of study, alongside a reference possibility (explicitly typed learning objects). *With its easy-to-understand graphical icons and GUI, ECOML may be the best tool that satisfies this requirement.*

4. The notational system must be able to fully describe a unit of study, including all the typed learning objects, the relationship between the objects and the activities and the workflow of all students and staff members with the learning objects (completeness). And regardless of whether these aspects are represented digital or non-digital. *ECOML permits the abstraction of learning units, learning objects and/or learning components in LDs and/or OKUs. Therefore, graphical representation of any instructional component (digital or non-digital) is possible in the models that are created with ECOML.*

5. The notational system must describe the units of study so that repeated execution is possible (reproducibility). *Even recursive applications are possible in ECOML's hierarchical approach, which in turn, makes repeated executions possible.*

6. The notational system must be able to describe personalization aspects within units of study, so that the content and activities within units of study can be adapted based on the preferences, prior knowledge, educational needs and situational circumstances of users. In addition, control must be able to be given, as desired, to the student, a staff member, the computer or the designer (personalization). *ECOML provides the facility to personalize LDs and/or OKUs within its IMS-LD complaint structural properties. But, current version of ECOML cannot execute the pre-designed instructional models. This feature (i.e. LD player) is being implemented as a future add-on.*

7. The notation of content components, where possible, must be medium neutral, so that it can be used in different publication formats, like the web, paper, e-books, mobile, etc. (medium neutrality). *Current version of ECOML is written in Java using the Eclipse software development environment. Therefore, the notational representation of content components should be possible in all complaint mediums.*

8. When possible, a ‘wall’ should be placed between the standards that are used for notating units of study and the technique used to interpret the notation of the units of study. Through this, investments in educational development will become resistant to technical changes and conversion problems (interoperability and sustainability). *Current version of ECOML is not coded to satisfy this requirement.*

9. The notational system must fit in with available standards and specifications (compatibility). *Up to the knowledge of this thesis study, there are no published standards for graphical symbols, which are indicated several times with up-to-date references through out this study. Nevertheless, IMS-LD and SCORM based standards/specifications, concerning the authoring tools (i.e. Level-A editors), are tried to be satisfied.*

10. The notational system must make it possible to identify, isolate, decontextualize and exchange useful learning objects, and to re-use these in other contexts (reusability). *ECOML abstracts learning objects with graphical symbols. Although, it is the responsibility of the instructional designer to validate and*

*maintain the availability of learning objects, there should not be any problem in the reproduction of the model in other contexts.*

11. The notational system must make it possible to produce, mutate, preserve, distribute and archive units of study and all of its containing learning objects (life cycle). *Current version of ECOML saves the instructional model with all its structural properties within its own database, and it also exports the IMS-LD complaint xml file on demand for possible reproduction.*

This comprehensive list embraces almost all necessary requirements that can be desired to have in an educational modeling language. Obviously, whole list is not easy to comply with. Nevertheless, ECOML's COSE approach satisfies almost all of Koper's above stated requirements. How current version of ECOML approaches to the educational modeling is also presented in Figure 14.

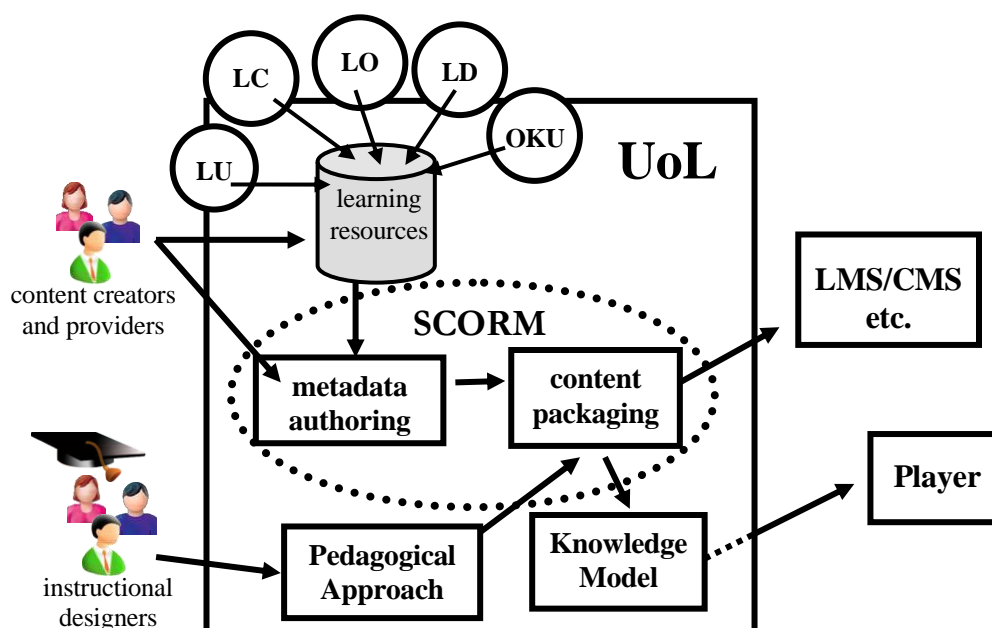


Figure 14. ECOML's Approach to Educational Modeling

Recently Waters and Gibbons (2004) pointed out that almost all creative and technological fields have developed one or more notation systems and design

languages. Unfortunately, no standards have been established for graphical representation of educational learning designs and models, yet (Knight et. al, 2005). ECOML has used COSEML as the reference model when developing the graphical modeling language.

#### 4.2. Software Modeling

As Richards (2005) quotes what Nobel Prize winning physicist Charles Townes elaborated the question of “What is the purpose or meaning of life? Or of our universe?”. And concluded as “These are the questions which should concern us all.... If the universe has a purpose, then its structure, and how it works, must reflect this purpose”.

How can human beings be able to study the structure of a system and comments on how the system works? Most probably, modeling efforts stem from this fact. Models describe the structure and behavior of a real-life system whose requirements are being analyzed in a certain notational system. A model is an abstraction of the desired real-life facts with small, simple and discrete components. This abstraction makes the study and investigation easier when compared to the actual behavior of the real-life system. Booch et al. (2000) emphasize the importance of software modeling by saying a model is a simplification of reality and they are important for visualizing and controlling the system's architecture. Models provide a better understanding of the processes, which expose opportunities for simplification and reuse. ECOML models the educational process as depicted in Figure 15.

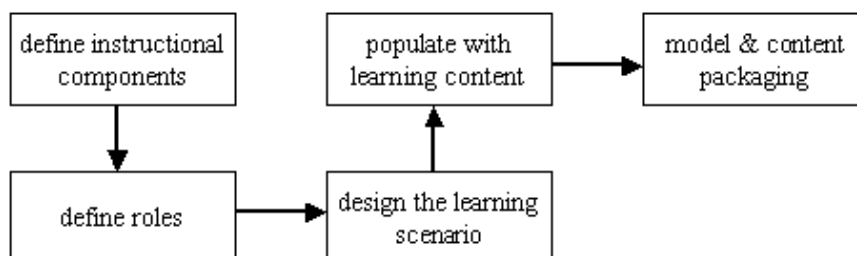


Figure 15. ECOML Authoring Process

As is the case for every type of modeling efforts, software modeling also needs rules, according which the model is designed and described. In educational modeling efforts, rules should define the software in such a way that the outputted model will be able to successfully describe how people (teacher, learner) in roles carries out activities (teaching, learning) with resources (LUs, LOs, LCs, etc.).

Knight et al. (2005) emphasized the ground rules about what learning design must poses. From which we can deduct the structural components.

A Learning Design can choreograph;

- the order in which the content will be presented,
- how it will be sequenced,
- how it will be assigned to learners in a lesson,
- how it will be assessed.

And Knight et al. (2005) continued describing the educational model with saying “Conceptually this can be pictured as pulling learning objects from a repository and using the Learning Designs to integrate learning objects into activities that involve learners. Hence, Learning Designs can be represented graphically or formalized according to an information model. No standard has yet been established for the graphical representation of learning designs; however, there are many possible methods”.

Since this thesis study considers the COSE approach, which is based on structural decomposition of the system, and claims that educational systems can be represented best in hierarchical models. Then the hierarchical decomposition of the system should be the starting point. In any hierarchical component-oriented approach the main rule is the ownership and dependency relations. Every component must have an *owner* (the component that is responsible for its creation and destruction). Next is the *dependency*, which means any bit of data that is used to evaluate a dependent component must have a precedent. Last rule is the identification of the *identity* that means all components have unique identity (meaning; everybody that is accessing an identical object can expect the same uniform behavior). Component properties of ECOML are described in Figure 16.

### **4.3. Defining the Language – Essentials**

ECOML provides facilities to represent all instructional components for modeling purposes. Modeling can start either in top-down or bottom-up fashion. Usually, instructional modeling starts with top-down decomposition of the system, but construction of the model continues in bottom-up fashion. Modeling starts with a top-down decomposition of the system, and simply, abstract building blocks of the system are found in this phase. This top-down activity continues with searching the graphically represented learning components among the ones offered by the modeling language. When they are found, component composition is carried out bottom-up to reach the desired representation of the system.

Development of a visual notation system for supporting and enhancing the design process is a time consuming effort that needs different skills and experience. Visual notation that defines graphical elements should be clear enough to indicate the intended meaning. Therefore, learning goals should also be considered when defining some of the graphical elements, while others can easily address the definition of learning activities, services or learning materials. Obviously, the visual notation system should provide users with an easy to use GUI and comprehensive design environment for creating educational interactive content. If the model is going to be played with the help of a LD player then creation of the interactive content gains more importance rather than simply representing the instructional model with graphical symbols.

On the other hand, when working in the design phase, instructional designers who can be instructors, course authors, administrators and/or mentors and coaches, basically deal with the problem-solving activities about possible content of teaching/learning activity and how it will be delivered. This problem solving activity encompasses the identification of people their roles and activities, which will be carried out together with the resources and the learning services that are involved in the instructional process. Therefore, a graphical modeling tool should be designed in such a way that it should help to reduce the cognitive load, which refers to the load on working memory during problem solving, thinking and reasoning of the instructional designer. And such a tool should provide enough

number of enhanced communication facilities between the designer and the modeling tool via smart and easy to use interfaces.

Secondly, It should be a design tool for developing instructional materials using all available learning elements. That is to say, all possible learning elements (i.e. UoL, OKU, LD, LU, LO, LC and others) should be represented in a meaningful way and relations should be clearly defined with respect to the delivery of the course content.

On top of all, the outcome must be pedagogically sound, as well. Different pedagogical methods must be demonstrable in a comprehensible model. In order to provide a universal acceptance, this flexibility of the pedagogical immunity should be the most essential part of the modeling language.

In order to satisfy these concerns, and come up with a better approach, ECOML has further updated the IMS-LD defined properties (literature-wise criticized items were taken into consideration). ECOML's structural properties are depicted in Figure 15.

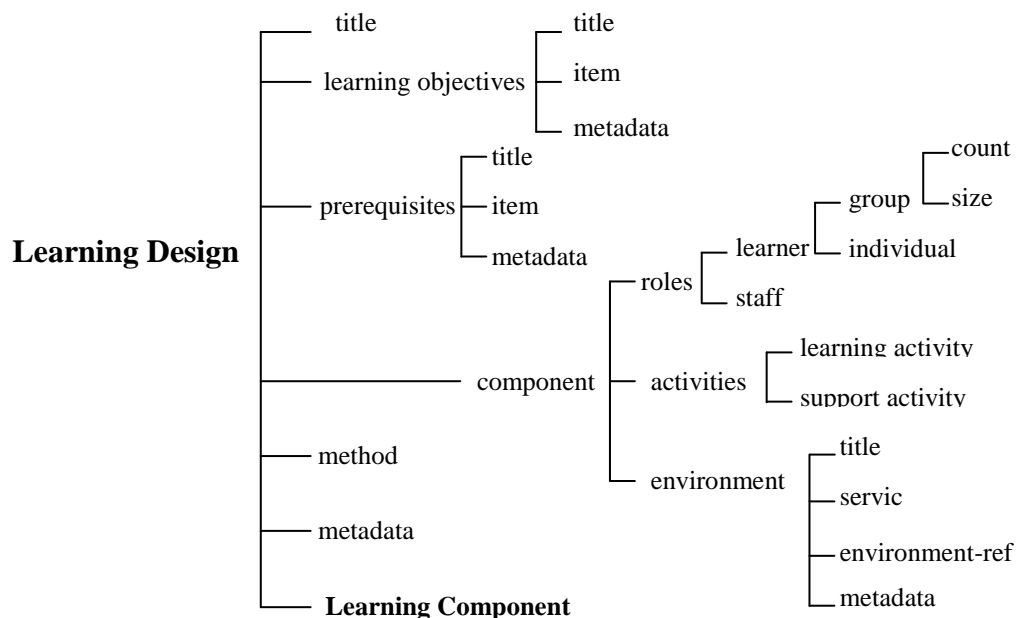


Figure 16. Structural Properties of ECOML

There are two different component types in ECOML. These symbols are defined in Section 4.4. Mainly, the first group is the package abstraction group that indicates further related elements in an encapsulation. An instructional package that is named as Learning Design or Learning Component can contain further packages, and component abstractions. Both of them are the fundamental structural elements that are used in the definition of part-whole relations. System decomposition is made using packages and decomposition is detailed using other component abstractions including physical elements that abstract physical components of the instruction. Data, activity and control abstractions are designed as form-based. The related information is kept upon if and when the user submits relevant data. If user wants to play the instructional model in the future, s/he needs to enter the activity and control related data, otherwise the instructional model stays as a graphically represented educational unit of learning and can be saved for future references for the same purpose.

Physical elements, their communications with other components and interfaces are also represented with symbols. There are connectors to represent communications among components, both in abstract and physical levels. There are two types of connectors and their main purpose is to show the parent-sibling relation in the hierarchical tree. In abstraction, a connector may represent many message or event connections between two components. It represents at least one, but possibly more than one message (or event) link. A message link may represent a function call (local or remote) originating in one component and terminating at the interface of another. Events are similar to messages but semantically they stand for calls initiated by external causes in contrast to calls made under program control. In ECOML, one type of connector indicates the relation between abstract elements, i.e. Learning Design, OKU and Learning Components that may contain further LDs, and OKUs. This type of connector can be described as an event connection and indicates external activation of the abstract element during model playing with Level-B or Level-C authoring tools. The other type simply indicates the parent-sibling relation between physical elements and the immediate root structure. This type is solely a message connection. Its activation is done automatically given the satisfaction of the



constraint that is imposed upon the abstract element during model playing with Level-B or Level-C authoring tools. Table 2 shows the graphical representation of such elements of ECOML.

ECOML is based on COSEML's structural architecture and using the same software elements as its basic fundamental modules. COSEML's main Java code is also used as the basic infrastructure and groundwork when developing the ECOML.

Mili et al. (1995) discusses the software reuse in software engineering as the (only) realistic approach to bring about the gains of productivity and quality that the software industry needs. Software reuse includes reusing both the products of previous software projects and the processes deployed to produce them. According to Mili et al. (1995) software reuse can be vertical or horizontal. Vertical reuse can exist if there is a single application, and in this application others can share many software modules developed by one. Horizontal reuse can exist if there are many application areas. Vertical reuse offers more benefits, but requires developers to make domain analysis in order to design systems.

The success of COSEML makes itself reusable in both vertically and horizontally. ECOML proves that COSEML can be horizontally used in another area of interest like Education. And, additionally as a hierarchical modeling language COSEML's software modules can also be reused in similar projects where main philosophy is the hierarchically represented models. Therefore, ECOML reuses the COSEML in both vertically and hierarchically.

Software reuse can be measured by looking at the ratio of reused code to total code in terms of Lines Of Code (LOC). However, Mili et al. (1995) says that it is more convenient for programmers and project managers, to measure the reuse in terms of modules. And, Mili et al. suggest the following steps for a better approach;

- existing program modules should be rewritten for reuse,
- the purpose, capability, constraints, interfaces, required resources, objects, and interfaces between objects of existing program modules should be well-defined,

- before writing of a new module, existing modules should be searched. If it is not found then that module should be written by obeying the first and second steps.

In fact, this is partly the theory that lies beneath the COSE approach (i.e. re-usage of components where and whenever applicable).

If advantages of the software reuse can be briefly defined;

- it yields substantial productivity benefits. For example, Mili et al. claims that software reuse in the construction phase increases the productivity by 20% or more by using previously developed software modules,
- it reduces the costs of construction and re-testing of the system,
- it increases overall quality and reliability. Obviously when quality software modules are reused, overall product quality will improve, as well. Of course, reliability of the modules and their certification with a certain level of confidence is an important aspect. But, such concerns are beyond the coverage of this thesis study.

As it is stated earlier, COSEML is a proven software code and a modeling tool. And, ECOML reused COSEML's software modules as its starting point. COSEML is composed of 6 packages and 6653 lines of code, altogether makes up three different modules, namely GUI, UTILITY and GRAPHICAL OBJECTS modules. ECOML has reused ~80% of the GUI module (e.g. ToolBar, DiagramTree, MainDiagram, etc.), ~25% of Graphical Objects module (e.g. AkNode, AkShape, AKRows and AKModelTree) and 100% of Utility Classes module. Reused code is roughly equal to 65% of COSEML's total software code. Approximately 100 lines of code were deleted because of their unrelated functions. And, ECOML has introduced 2800 newly developed lines of code. As a result, one can say that ~35% of COSML's software coding has been modified and/or inline code is introduced.

A software component is defined by Kara (2001) as the independent and replaceable part of a system that has a function in a well-defined architecture. As it is also described by Kara (2001) any component in COSEML (and so in

ECOML) contains four fields as in Figure 17. *Description* field gives the information about the component to semantically search for the component (e.g. the statement of “This component is a general purpose button” is a description). *Properties* field is used to change the characteristics of the component (e.g. “Color” is a property of the button component). *Methods* field gives the services provided by the component (e.g. “GetButtonState” is a method of the button component). *Events* field is used to set the actions to be performed when an event occurs in the component or from the environment (e.g. “OnButtonClick” is an event handler of the button component).

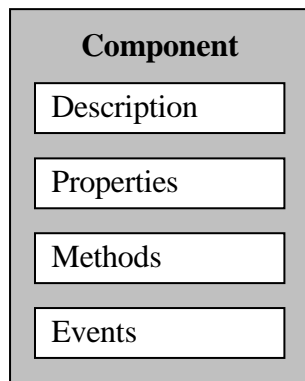


Figure 17. Fields of a COSEML (ECOML) component.

The following software components and definitions are borrowed from COSEML whose characteristics are explained by Kara (2001) in his thesis study. Some of the components are coded new, and, they are described at the proper place inside this thesis study. Some of the components (like Graphical User Interface (GUI); tool bar, main frame, split pane, and Graphical Editor (such as the abstract and component level trees) are reused in ECOML and their structural properties are as follows: Please note that Figures 18 thru 24 are taken from Kara’s (2001) thesis study.

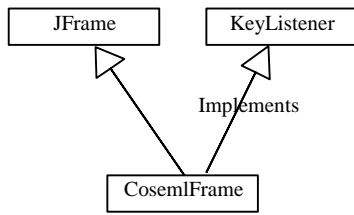


Figure 18. Class hierarchy of CosemlFrame

`CosemlFrame` is the main frame class in which the user interface fields, like the main menu, main tool bar, split pane, and status bar, are created. Its class hierarchy is represented in Figure 18. It implements the `KeyListener` class to get the key clicks from keyboard.

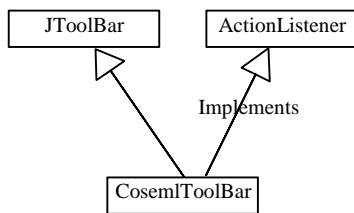


Figure 19. Class hierarchy of CosemlToolBar

`CosemlToolBar` class, which is depicted in Figure 19, creates the main tool bar. It implements the `ActionListener` class to know the pressed button. This class gets the figures of the buttons from gif files in Images directory.

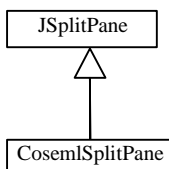


Figure 20. Class hierarchy of CosemlSplitPane

The split pane of the main frame contains the tree view of containers and modeling page. The user can change the size of these elements by moving the splitter between those. The class hierarchy is represented in Figure 20.

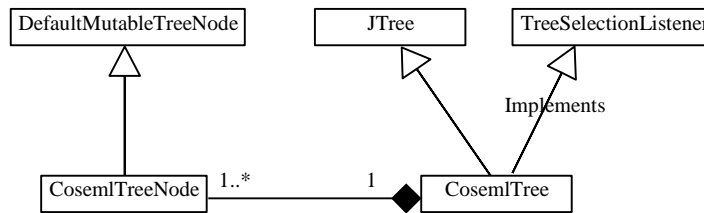


Figure 21. Class hierarchy of CosemlTree and CosemlTreeNode

Figure 21 shows the tree view of the containers is in a scrolling pane. It enables to user to select the nodes, CosemlTreeNode, of the tree by implementing TreeSelectionListener class.

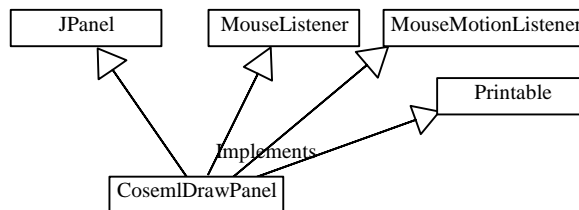


Figure 22. Class hierarchy of CosemlDrawPanel

The modeling page, i.e. draw panel, is also in a scrolling pane. It is derived from JPanel class as depicted in Figure 22. It implements the MouseListener, and MouseMotionListener classes to get the mouse events. Since the tool can print the model, it also implements the Printable class.

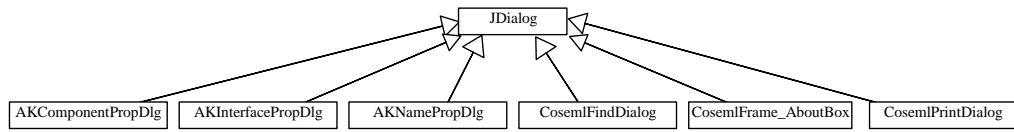


Figure 23. Class hierarchy of dialog classes

The tool contains many dialog windows, some of them are derived from JDialog class, and some of them are created from built-in message windows. The dialog classes derived from JDialog are in Figure 23.

Tree algorithms are designed to graphically balance the locations of the nodes such that every parent is centered above its children. There are recursive routines to accomplish this goal. Kara (2001) mentions about three passes, which are carried out in COSEML. Nodes are left justified and they are organized as one linked-list per level at the end of the first pass,. Every level starts on the left of the screen, and there are no gaps between the elements of the row.

Figure 24 shows the quadruply-linked-list structure used in the algorithm. On the left, there is a linked-list (rows), holding pointers to the first and the last elements of the rows. This list has an element for every row. On the right, there is the tree that consists of nodes where each node contains the node information and the pointers to parent, child, left and right nodes. The Child pointer points to the leftmost child only. The same structure is used in ECOML with the exception that if the child is defined as a LO or LU then it is not possible to assign another child. It is perfectly possible to assign another child to its left or right, though. Otherwise as long as the node is assigned as a parent then it is possible to branch downwards with further siblings.

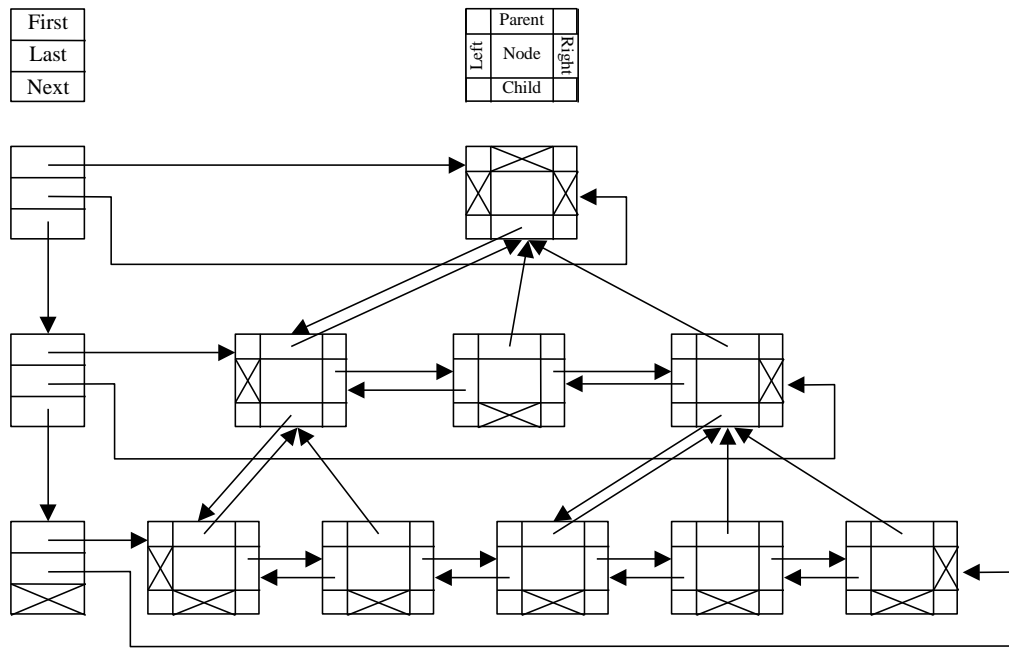


Figure 24. Tree data structure

A bottom-up second pass makes sure that every parent is centered above its children. A third top-down pass adjusts one parent at a time, to assign the final horizontal position. If a parent has to be shifted right, the amount of shift has to be propagated to all the related sub-tree. The shift value for a parent is utilized at the lower level, adding to the required shifts for the nodes at this lower level. So this pass actually propagates a shift from top to bottom, with an added value to this shift at every level per sub-tree.

The main modules to find tree nodes are LastChild, Siblings, MidSiblings, RightCousin, and HasRightSibling. LastChild module returns the last child node of a given parent node by a search starting from left most child and moving to the right. The tree traversal is done in depth-first pre-order (prefix) traversal. Other modules traverse the tree similarly and return an appropriate value (depending on which module is called) provided that the child exists, otherwise either a “Null” or “False” is returned (e.g. HasRightSibling).

There are six different modules used for inserting new nodes. These are InsertLeft, InsertRight, InsertChild, StartFamily, StartRow, and EnlargeFamily.

InsertLeft algorithm inserts a given node to the left of another given node. InsertRight algorithm inserts a given node to the right of another given node. While inserting, the algorithm updates the first pointer of the current tree row if inserted node is the first. InsertChild algorithm inserts a given node as a child of another given node. While inserting, if the row of parent node is the last row, the algorithm starts a new row by calling StartRow algorithm. If the row is not the last and the parent has no child then it starts the family of this parent by calling StartFamily algorithm. If the parent has children then the algorithm enlarges the family of that parent by calling EnlargeFamily algorithm.

After the insertion of nodes, all nodes in the tree are centered automatically according to their children. This is done by calling another module that is called as the PrepareTree algorithm.

As stated earlier, ECOML has reused ~80% of the GUI (e.g. ToolBar, DiagramTree, MainDiagram, etc.), ~25% of Graphical Objects (e.g. AkNode, AkShape, AKRows and AKModelTree) and 100% of Utility Classes modules. Reused code is roughly equal to 65% of COSEML's total software code. And, ECOML has introduced 2800 newly developed lines of code. As a result, ~35% of COSML's software coding has been modified.

#### **4.4. Components - Graphical Modeling Elements**

As Component Orientation is a new Software Engineering paradigm for systems development, where development by integration is suggested. ECOML is similarly designed to be a modeling language that is flexible enough to incorporate all kinds of educational components to be represented in the modeling phase. In order to accomplish various representations for object oriented and component related modeling, commonly used graphical symbols are adopted where applicable.

ECOML is based on a process model for the development of educational models utilizing available components. As a graphical language it supports the following components; OKUs, LCs and LOs. When OKUs, and/or LCs are used as parent modules in a hierarchical design, their sub-components can be defined as other primitives that are used during the delivery of the course, training or any



other similar teaching/learning process. Instructional model starts with the parent node that is designated as the Unit of Learning as it conforms the IMS-LD specifications. A UoL may have OKUs and/or LCs as other abstract sublevels.

The hierarchy is a key concept in design cognition, and it is not supported effectively in all educational modeling languages. To address this concept better, ECOML utilizes multiple hierarchical diagrams, as well as singled out designs each of which shows the abstract decompositions and educational component compositions together. ECOML's Main Diagram does not necessarily have to represent the complete model. ECOML permits the user to have more than one package with more than one sub packages (like OKUs and LCs). Each of them can be represented in different linked-list structures at separate diagrams. Meanwhile, split pane is designed to show the overall structure of the learning design. That is to say, different levels and/or groups of components can be selected in different diagrams, as well as selecting different kinds of links. The structural breakdown will remain as the background view, and other views can be turned off and on by selecting different levels from the split pane.

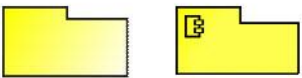

	 <p style="text-align: center;">...</p>
<p style="text-align: center;"><b>Main Packages</b></p>	<p style="text-align: center;"><b>Physical Components (LOs, LUs and/or LCs)</b></p>

Figure 25. ECOML's Graphical Symbols

ECOML can addresses both abstract and physical components in every level of the hierarchical model. In every-level, elements represent the abstractions for package, data, function, and control. Figure 24 depicts the symbols used in

ECOML. The symbols that are used to abstract the physical components (i.e. graphical symbols of LOs and/or LUs) have created, copied or edited from the worldwide available icons that claim either Creative Commons License or restricted GNU Lesser General Public License, which permit the usage of graphical symbols for educational purposes.

The “Learning Design” (root) abstraction indicates the instructional package. Therefore, it is represented by the UML’s package symbol. In the lowest level only graphical symbols that abstract physical components can be used. These are the components that indicate the resources, which are being used during the delivery of the instructional process.

Besides the learning unit or learning object abstractions, also association links are required to connect the symbols, which are depicted in Table 2. There are two types of links, which are either the default link between two symbols that is called as the *message link* and represent the mandatory control flow (represent method relation) between the components. The other type of link is the event connector that indicates the *association relation*. Association relation represents the relation between higher-level abstractions of OKUs and/or LCs, which can be an independent part of the overall instructional design. Both links show ownership relations.

Table 2. ECOML Graphical Symbols




Symbol	Explanation
	<b>Learning Design:</b> Package is a container that wraps system-level entities and functions etc. at a decomposition node. Can contain further LCs, OKUs, LUs and LOs. Can own event connector ports
	<b>OKU or Learning Component:</b> Can contain further Learning Components, OKUs, LOs, LUs. Can own event connector ports.
	<b>LO, LU, LC:</b> represents a help information

Table 2. Continued






















Symbol	Explanation
	<b>LO, LU, LC:</b> represents a read-only document (e.g. pdf).
	<b>LO, LU, LC:</b> represents a time dependent activity.
	<b>LO, LU, LC:</b> represents an application. Can be any real-life application (e.g. lab activities, outdoor activities, etc.) with the teacher.
	<b>LO, LU, LC:</b> represents book(s).
	<b>LO, LU, LC:</b> represents certain chapter(s) or reading material from an information resource (e.g. book(s)).
	<b>LO, LU, LC:</b> represents a mail document
	<b>LO, LU, LC:</b> represents a multimedia file
	<b>LO, LU, LC:</b> represents a sound file
	<b>LO, LU, LC:</b> represents an important document
	<b>LO, LU, LC:</b> represents a HTML document
	<b>LO, LU, LC:</b> represents a Wiki document
	<b>LO, LU, LC:</b> represents a chat session.
	<b>LO, LU, LC:</b> represents a forum session

Table 2. Continued

Symbol	Explanation
	LO, LU, LC: represents an in-class discussion
	LO, LU, LC: represents an assessment item (quiz, exam, etc.)
	LO, LU, LC: represents out of classroom assignments (homework, etc.)
	LC: represents a face-to-face lecture or a teaching/learning session.
	LO, LU, LC: represents a presentation or a guest speaker
	LO, LU, LC: represents a video conference session
	<b>Message Connector:</b> A method relation. Represents control flows across the system modules of the lowest level.
	<b>Event Connector:</b> An association relation. Represents control flows across the system modules of the highest level.

Yet another important factor, which must be taken into consideration, is the “constraints”. Because of the fact that every phase of human life involves constraints and one cannot think of a real-life process without constraints. Bartak (1999) says a constraint is simply a logical relation among several unknowns (or variables), each taking a value in a given domain. The constraint thus restricts the possible values that a variable can take; it represents partial information about the variables of interest. The important feature of constraints is their declarative manner, i.e., they specify what relationship must hold without specifying a computational procedure to enforce that relationship. First of all, any legal model

should satisfy such constraints. Secondly, to eliminate unwanted interpretations, users may enforce additional constraints. So, every modeling language must come up with abstractions of constraints, as well. These constraints stem from the modeled domain, the implementation domain (e.g., no multiple inheritance), and even from the modeling process domain (like UML's sequence diagrams, which is required for each external event).

ECOML as an instructional modeling language bears both internal external constraints. Internal constraints include syntactic and semantic constraints, which are mainly inherited from COSEML and define the meaning of the language primitives as well as the right way to use them. That's why, ECOML is bounded by the same internal constraints of COSEML. External constraints are the ones that must be taken into consideration when abstracting real-life instructional processes into educational models. Mainly, such constraints become important when an external player (like ReLoad or CopperAuthor) plays the instructional model. Such as a sequence of an activity may depend on the learner's answers to a certain assessment process, quiz, exam or homework. External constraints are defined as conditions by IMS-LD specifications. Conditions in unit of learning are used to show or hide parts of pages using classes, environments, activities-structures and support-activities. They all have a basic structure, consisting of an <if> statement checking a condition, a <then> part which describes what to do when the condition is true and an <else> part which describes what to do when the condition is false. ECOML describes such activities within its "Assessment" group of graphical symbols and description of the constraint resides in the prerequisites component structure (see Figure 16).

#### **4.5. An Example Model**

In this section, a sample model is prepared and explained. A typical course material that is delivered in a blended learning environment, is modeled by ECOML. Usages of various menu items are also explained with the help of screen shots.

An undergraduate course, which is offered in the Department of Computer and Instructional Technology Teacher Education of Bilkent University, was

modeled as an UoL to reflect the intended blended learning methodology with OKUs, LCs and other learning materials (LOs and LUs). It is an introductory course that teaches the fundamental concepts of Operating Systems to the sophomore students in their 3<sup>rd</sup> semester. The course consists of three semi-independent parts. One part is designed as an independent OKU; concepts that are referenced in OKU make up the main topics of the operating systems course. This part can be taken out from the learning design without affecting the main structure of the course. Primary learning objective of the OKU part is to teach self-regulated learning concepts to the students. Second semi-independent part contains the computer-lab practices where students learn and develop Unix shell scripts using lab facilities. This part is also an independent part of the learning design. Although, some of its concepts are relevant to the main course, it can be easily taken out without affecting the delivery of the main course. And the third part is the conventional lectures, which consist of formal face-to-face lectures and discussion sections about the concepts of operating systems.

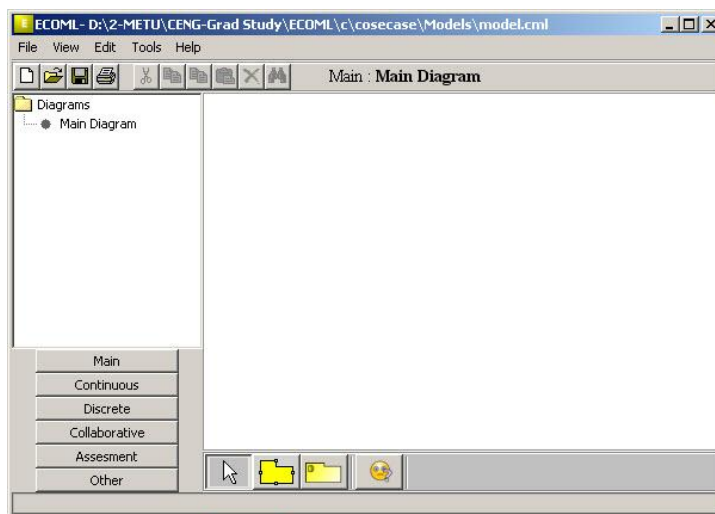


Figure 26. ECOML's Main Menu

The modeling language provides enough primitives to represent both physical and logical entities as well as implementation units for instructional

designs. Figure 26 depicts the first dialogue page of ECOML. This page consists of three parts. “Main” is the area that is provided to draw the hierarchical tree representation of the Learning Design. In the bottom of this part user can find the necessary graphical symbols (abstractions) of educational components. Left part of the page is reserved for the split pane and the graphical symbol groups. Split pane contains the hierarchical structure of the Unit of Learning. User can select different Learning Designs that represent different levels of the hierarchical structure, and the corresponding part of the LD (or part of the tree) is shown on the right hand part of the page, which is designated as the “Main” area.

The upper part indicates the general controlling functions like, File, View, Tools and Help. Also a toolbar which provides convenient functions to the user, like “new file”, “save” “print”, “copy” “paste”, “delete”, “”search”, etc, is also provided in this section. Selectable functions that are applicable to the current session become visible, so they are usable by the designer.

Figures 27 through 30 provide the detailed decomposition of the example course with some of the graphical symbols that are listed in Table 2.

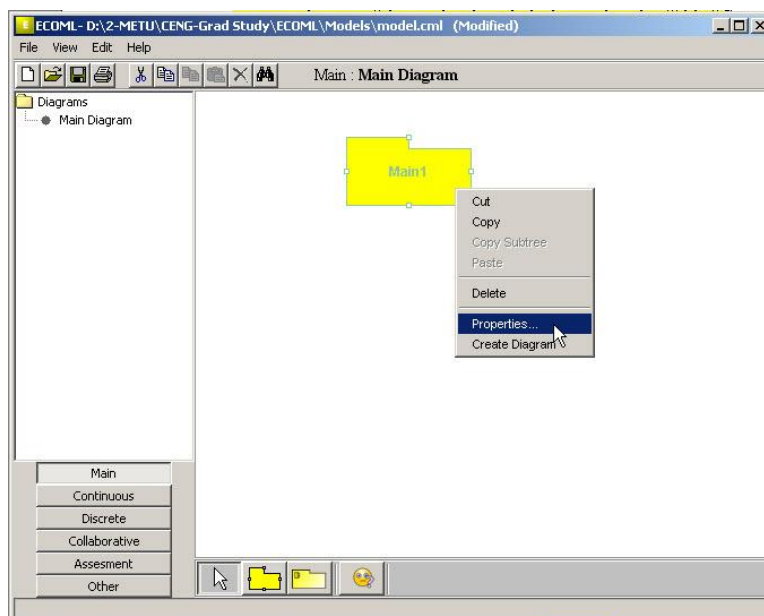


Figure 27. Learning Design Properties in Main Diagram

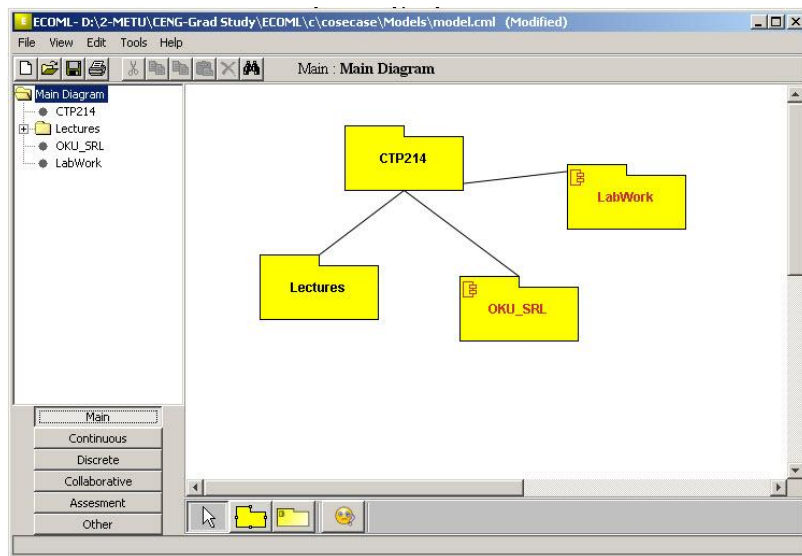


Figure 28. Learning Design in Main Diagram

Graphical representation of various physical resources is grouped under six different categories as shown in Figure 27. Every instructional model starts with the abstract symbol, which indicates that this is a learning design (LD). Hence, the starting symbol is called a LD or Main. The “Main” group contains another abstraction that is called as OKU (which is an LC). These represent independent or semi-independent parts of a course. Each visual notation inherits a “learning objective” of the course, but each part can also be taken out completely without affecting the delivery of the rest of the course. Each LD or OKU and/or LC can be used in other contexts and/or platforms for similar purposes (satisfying the inherited learning objective).

The Main Diagram (Unit of Learning) starts with the Main 1 symbol. Right-click with the mouse on top of Main 1 symbol permits the user to enter the design properties of the instructional model (see Figure 27). This pop-up window also lets the designer to use “copy”, “paste”, “delete”, and etc. facilities for easier modeling.



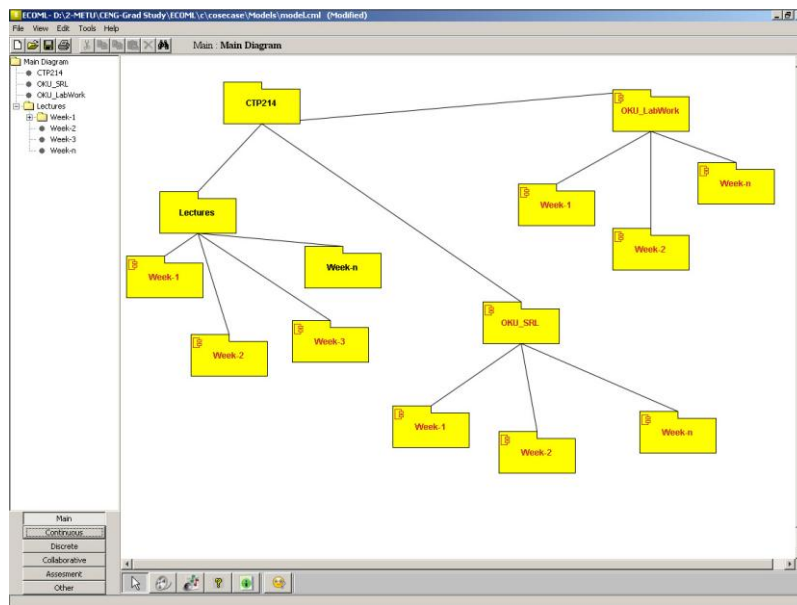


Figure 29. Hierarchical Representation of Learning Components

The other five categories of graphical symbols represent abstractions of various physical elements, which are being used during the delivery of the course. This second group of visual notations is defined as:

- “*Continuous*” Group: contains abstract that indicate educational resources, which are continuous in nature, like multimedia files or wikis, etc.
- “*Discrete*” Group: is for static resources like various text, or pdf files.
- “*Collaborative*” Group: contains interactive resources that describe situations in which particular forms of interaction among people are expected to occur. Such resources can be used either in face-to-face educational environment (like classrooms) or in computer-supported environments (like, chat, forum, etc.).
- “*Assessment*” Group: contains abstractions, which enables the designer to conditional activities and assessment facilities, like quiz, exam, etc.. Such conditional abstracts can be used to realize the learning scenarios when UoL is tried to be played in Level-B or Level-C authoring tools.

- “Others” Group: This group is used for abstractions that are not be categorized in any one of the above explained four groups. This group will also contain new graphical symbols that may appear in the future, which were not thought of at the time of first implementation of ECOML.

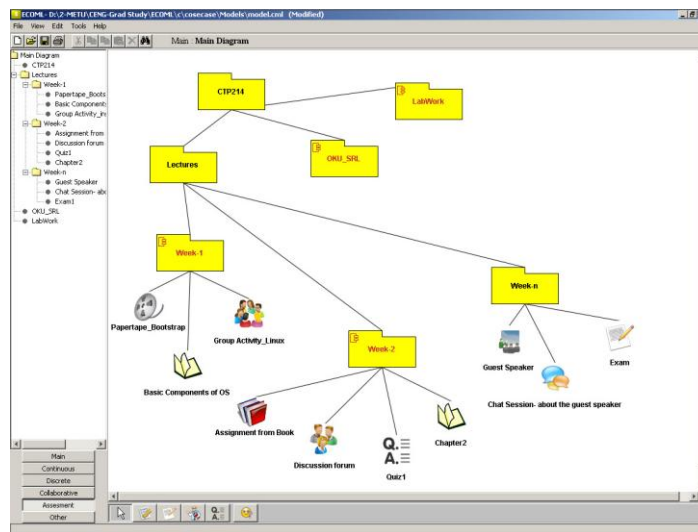


Figure 30. Lowest Level with Physical Components

Finally, the output of ECOML can be either a file (an XML file) that contains information about; resources, activities, environments, properties, conditions and/or notifications of the Learning Design specification or a visual graphically represented instructional model. This model (the XML file) can be played with IMS-LD compliant Level-B or Level-C authoring editors.

#### 4.6. Comparison of ECOT with Others

ECOT (Educational Component Oriented modeling Tool) is the authoring tool, which models educational processes according to the guidelines that are prescribed by the ECOML. Table 3 compares ECOT with some of the well-known authoring tools that are available in the market today. Compared authoring tools are selected from the ones, which are proposed by the IMS (see Table 1).

The tools that are under development or not released are not included into the comparison (see Table 1). Compared quality concerns (i.e. columns of Table 3) represent the quality concerns that are mentioned in Section 5.1. The relevant explanation of the columns of table with respect to its rows (i.e. chosen editors) is given below.

MOT+ is developed by the University of Quebec. It is defined as an object-oriented modeling tool, but rather than an authoring tool, it is a specialized concept map editor (see Figure 31, which is given by Paquette et al. (2006)). MOT+ represents instructional models in concept map style with rather (un)user-friendly graphical symbols. Therefore it does not produce any hierarchically represented instructional models. It is also claimed that MOT's representation technique is wide-ranging. It applies to all cognitive fields and makes it possible to build various types of models such as class or component hierarchies, sequential, parallel or iterative procedures, verification theories and structures, processes and methods.<sup>40</sup> But, its usage is awkward and needs user training.

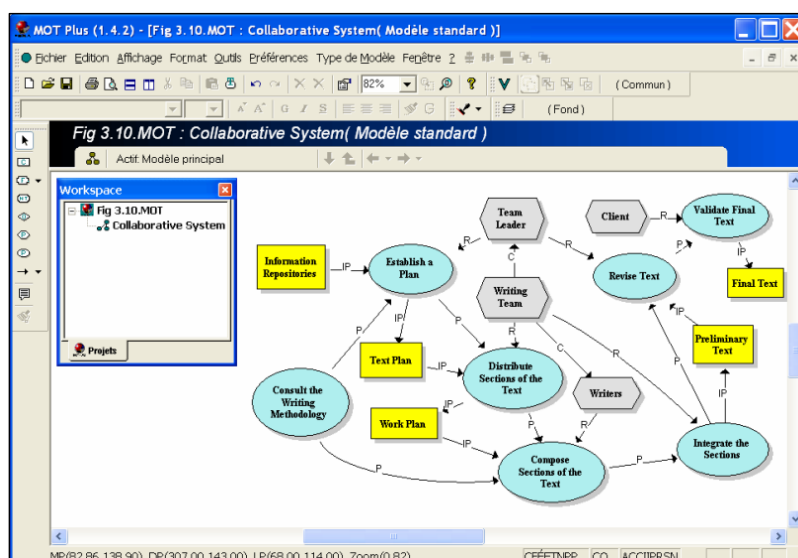


Figure 31. An instructional model generated by MOT+

<sup>40</sup> <http://www.cogigraph.com/Produits/MOTetMOTplus/tabid/995/language/en-US/Default.aspx>, last accessed January 15, 2010

Table 3. Comparison of ECOT with Other Authoring Tools

Name	Producer	Ownership	LD Editor	Conformed Standard	Hierarchical modeling with symbols	Context Sensitivity	Eclectic Benefits	Course Quality Assessment	Pedagogical Immunity
<b>MOT+</b>	University of Quebec	open source	general purpose	IMS-LD	No	No	Yes	Yes, executable model	Yes
<b>LAMS</b>	LAMS foundation	open source	general purpose	as future work (IMS-LD)	No	Yes	Yes	Yes, executable model	No
<b>Dialog+</b>	US & UK Universities	open source	online learning activity creator	SCORM, IMS-LD	No	N/A	Yes	Yes	No
<b>Collage</b>	University of Valladolid	open source	special purpose	IMS-LD	No	No	Yes	No	No
<b>Moodle</b>	Moodle Trust, Martin Dougiamas	open source	not an LD editor	SCORM	No	Yes	Yes	partly executable model	No
<b>ASK-LDT</b>	Informatics & Telematics Institute Greece	freeware (unavailable)	claimed as “general purpose”	SCORM, IMS-LD	claimed so, but unavailable for testing	unavailable for testing	unavailable for testing	unavailable for testing	claimed so, but unavailable for testing
<b>CopperAuthor Editor</b>	Open University of the Netherlands	open source	general purpose	IMS-LD	No	No	No. Difficult to use	Yes, executable model	Yes
<b>COSMOS</b>	University of Duisburg	open source	specific to few pedagogical methods	IMS-LD	Partly (only prespecified templates)	No, only prespecified templates	Likely. Needs the designer to know IMS	Yes, executable model	No
<b>RELOAD</b>	Reload project (JISC)	open source	general purpose	SCORM, IMS-LD	partly (only one LD is displayed)	Yes	No. Difficult to use	Yes, executable model	
<b>ECOT</b>	METU-CENG Turkiye	freeware	general purpose	as future work (SCORM, IMS-LD)	Yes	Yes	Yes	No	Yes

Learning Activity Management System (LAMS) is a learning design system to define and deliver learning activities, i.e. it is learning activity management software. It is described as a visual authoring interface to design and create learning sequences from a list of building blocks of individual or collective activities.<sup>41</sup> That is to say it is an educational modeling tool with authoring of scenarios that is based on learning activities. In LAMS, groups (can be considered as courses) are created and learning sequences are assigned to those groups. Therefore, as it is shown in Figure 32, instructional models are not represented hierarchically but they are represented in sequences of activities, one after the other. LAMS cannot represent courses where more than two independent or semi-independent parts, which should be executed in parallel, exists. Which makes it undesirable from the context sensitivity point of view.

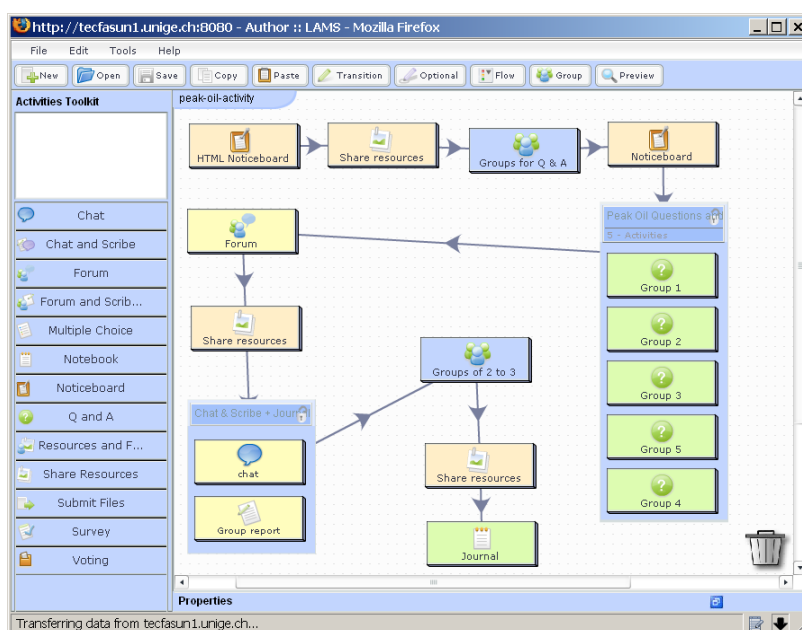


Figure 32. An instructional model generated by LAMS<sup>42</sup>

<sup>41</sup> <http://edutechwiki.unige.ch/en/LAMS>, last accessed January, 15 2010.

<sup>42</sup> <http://edutechwiki.unige.ch/en/LAMS>, last accessed January, 15 2010

As it is also admitted by Hernández-Leo et al. (2006) LAMS is capable of supporting a range of pedagogical approaches, in that designers can select different activities, which match their preferred style. But the fact of the matter is that it does neither have the same pedagogical immunity that ECOML has got nor it is creating hierarchically represented instructional models.

The DialogPLUS Toolkit is an online browser-based application to guide and support teachers as they create, modify, and share *learning activities* and resources. Its primary aim is explained by Davis et al. (2007) as to develop a distributed enabling information infrastructure for the support of learning and teaching in Geography; and innovative approaches to teaching and learning, based on this infrastructure. Therefore, from context sensitivity point of view, Dialog+ not suitable for mixed-mode course or face-to-face lecture designs.

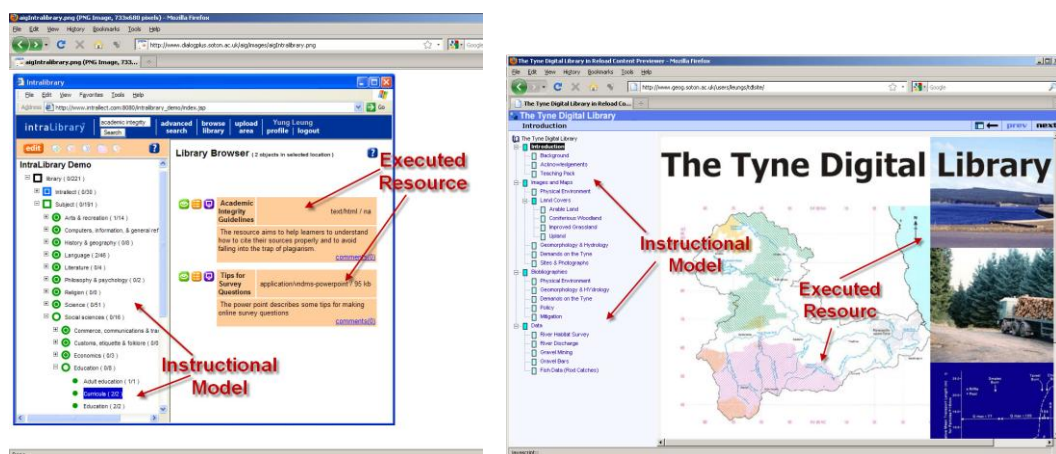


Figure 33. Instructional models generated by Dialog+<sup>43</sup>

Instructional design is made using pre-specified templates.<sup>44</sup> Dialog+ does not provide any visual notation for the design of instructional models. Therefore, it does not provide any hierarchical model with graphical symbols (see Figure 33).

<sup>43</sup> <http://edutechwiki.unige.ch/en/LAMS>, and <http://www.geog.soton.ac.uk/users/leungs/tlslite/> last accessed January, 15 2010

<sup>44</sup> <http://www.nettle.soton.ac.uk/toolkit/userarea/default.aspx> last accessed January 15, 2010

Additionally, although it is claimed that it covers tens of different teaching methods, how pedagogical immunity can be possible with a form-based tool that offers pre-specified methods?

Hernández-Leo (2006) describes the Collage authoring tool as a visual, high-level and specialized Learning Design authoring tool for collaborative learning. He also admits that it is a specialized editor for collaborative pedagogical approaches. Hence, Collage is not a general purpose authoring tool to start with. By the same token, one cannot mention about its context sensitivity as a quality concern. As can be seen in Figure 34, Collage does not represent instructional models in hierarchically either. Collage provides easy-to-use graphical symbols for modeling. Therefore, it satisfies the quality concerns as far as the eclectic benefits are concerned.

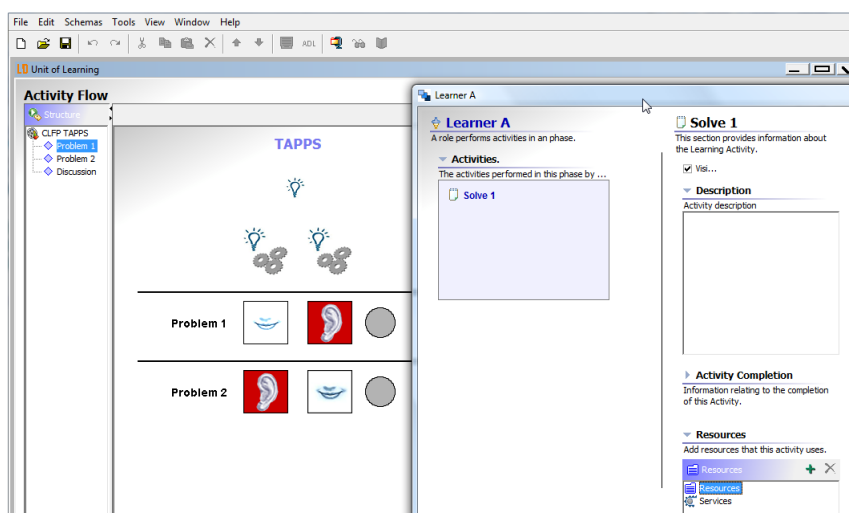


Figure 34. An instructional model generated by Collage.<sup>45</sup>

Moodle is a free web application that educators can use to create online learning sites. It is not an authoring tool in the classical sense. But, it is perfectly possible to produce instructional models with learning activities and resources.

<sup>45</sup> <http://edutechwiki.unige.ch/en/Collage> last accessed January 15, 2010

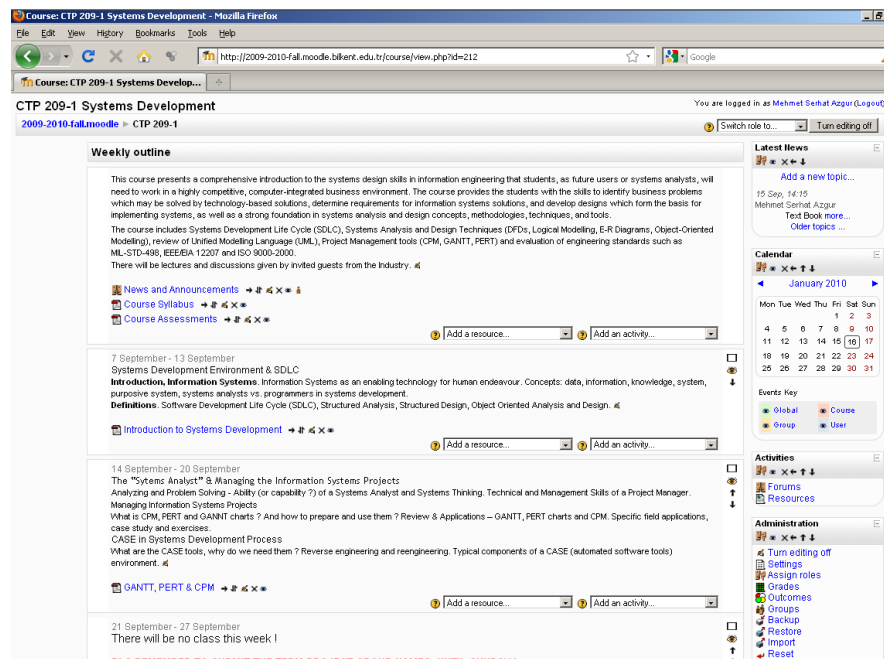


Figure 35. An instructional model generated by Moodle.

As it is shown in Figure 35, a university undergraduate course is modeled in a blended-learning fashion using the Moodle tool. Although Moodle is not IMS-LD compliant, it can deliver content in standard SCORM packages.<sup>46</sup> Unfortunately, it cannot produce hierarchical instructional models and it is not pedagogically independent (models ought to be based on time schedules therefore, problem-based and/or case-based pedagogical designs are not possible to represent).

Sampson (2006) describes the ASK-LDT as “The core design concept of the ASK-LDT is to provide a graphical user interface for the design and sequencing of learning activities, which, on one hand uses a standard low-level notation language for the description of learning scenarios and on the other hand enables pedagogical designers to use their own design notation (high-level notation) for the definition of learning scenarios”. Figure 36 is obtained from an

<sup>46</sup> <http://moodle.org/about/> last accessed January, 15, 2010



article written by Sampson et al. (2006) and published in Ed/ItLib (Education and Information Technology Digital Library)<sup>47</sup>.

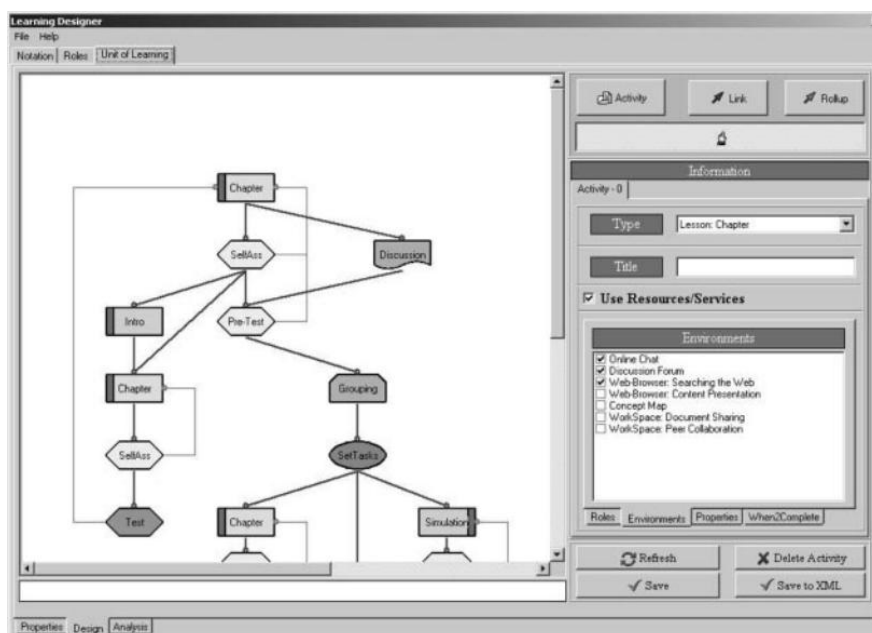


Figure 36. An instructional model generated by ASK-LDT<sup>47</sup>.

As it is declared in Section 3.3.2, neither this researcher nor others<sup>48</sup> was able to find out whether this project is still alive or not. Therefore, it is not possible to make any inferences about the quality of the tool because of the unavailability of the tool in spite of the fact that there are affluent of published documents about it.

CopperAuthor<sup>49</sup> is an open-source form-based editor developed by the Open University of the Netherlands. It is general purpose and supports IMS-LD Level A and B specifications. It can export UoL files, which can be validated and

<sup>47</sup> <http://www.editlib.org/> last accessed January 15, 2006

<sup>48</sup> <http://edutechwiki.unige.ch/en/ASK-LDT> last accessed on January 16, 2010

<sup>49</sup> <http://sourceforge.net/projects/copperauthor/> last accessed January 16, 2010

executed by the CopperCore engine. Last version is V1.6 that is published in February 2006.<sup>50</sup>

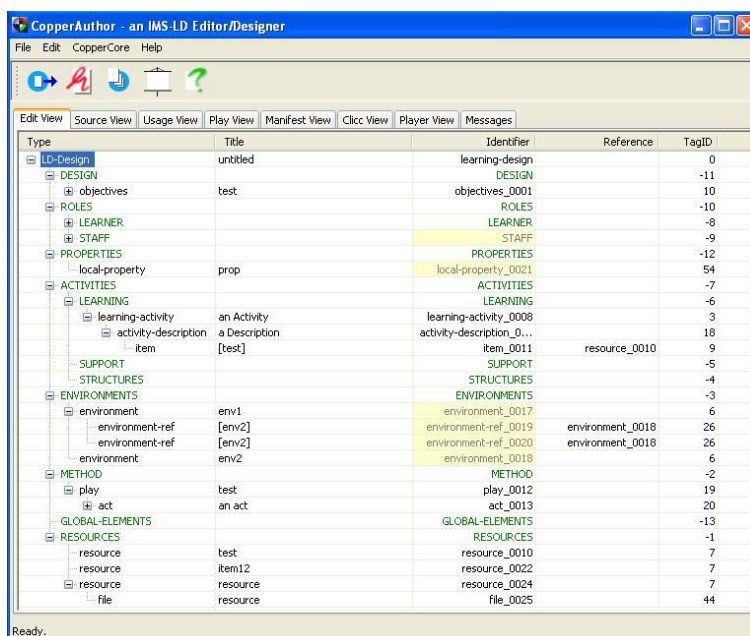


Figure 37. Main menu of CopperAuthor.

CopperAuthor is a form-based editor. Therefore it cannot represent hierarchical models. And, it cannot produce any graphical model of an instructional process. Secondly, Koper R (2009) says that courses that have no online components or automation, courses that are designed and taught by the same person all the time (classroom-teaching model), and simple, non-adaptive courses) are not within the scope. Therefore, its *context sensitivity* is in doubt. CopperAuthor is only for web-based designed courses. Its *eclectic benefits* is also very restricted because of the fact that one has to be in very good command of IMS-LD terminology in order to be able to design any instructional model (see Figure 37). Non-expert designers have great amount of difficulty when developing even for web-based simple courses.

<sup>50</sup> <http://dSPACE.nl/simple-search?query=CopperAuthor&start=0> last accessed January, 17, 2010

COSMOS ASK\_LDT and RELOAD authoring tools are reviewed earlier. Please see Sections 3.3.2 and 3.3.1, respectively.

For specific implementation and design details of ECOT please have a look at Chapter 4. Obviously, there are a few shortcomings of ECOT V1.0 when compared to some other similar tools. For example, the current version of ECOT cannot export imsmanifest files for execution by IMS-LD compliant players. This feature is started being implemented. Version 2.0 will have this feature operational. Another desirable feature of ECOML would be the ability to execute hierarchical models in e-learning environments. Planning and design of this part as another stand-alone educational tool is also continuing at the time when this thesis study has made.

## **CHAPTER 5**

### **CONCLUSION AND RECOMMENDATIONS**

Usually the only documentation that is available after a course is designed and developed, are the actual learning materials. Most of the time reproduction or the reuse of the same instructional design may not be possible. This raises some questions concerning the effort required in redesign or adaptation process when the original developer is not available.

ECOML, as a graphical tool with significant set of requirements and easy-to-use features provides basic scaffolding in the form of visual notations. This is an ideal authoring environment in support of instructional designers, as well as non-expert authors for a convenient way of modeling their knowledge. The resulting model can be easily reused in different contexts and across different educational platforms.

The implemented first version provides a flexible modeling tool in which all teaching components are effectively discovered, defined and connected. Development of an abstract notation that is sufficiently general to represent Units of Learning (UoL), Learning Components (LC), Learning Objects (LO), and Operational Knowledge Units (OKU) with respect to the common structures found in different instructional models was made possible. As it is shown, so is proven with the first version, hierarchical decomposition and modeling of instructional processes is easier, better, much quicker and visually more effective than any other method. Certainly ECOT is a good example, which is capable of producing such instructional models.

On the other hand, COSEML was proposed as the primary modeling language for Component Oriented Software Engineering approach by Dogru (1999). Since then, researchers have been contributing for the improvement of it. However, ECOML is the first modeling language that is an offspring of

COSEML. Since, ECOML produces hierarchically models of educational processes with a “build by integration” approach, it also becomes another case study that proves COSECASE can also be integrated with (educational) components architecture.

### 5.1. Quality Concerns

Evaluating a modeling language from the educational point of view is not an easy task. Because there are as many issues that should be taken into consideration as there are pedagogical approaches, which exist today. Nonetheless, there are some key issues that are discussed in the literature that are considered during the evaluation process. According to Botturi (2004) an instructional design language has to cope with at least four issues, some related to design in general, others specific to educational settings. Those four key issues are “context sensitivity”, “eclectic benefits”, “course quality assessment” and the “importance of time”. Keeping in those four key issues in mind, one can elaborate the conformance of ECOML as such:

- *Context Sensitivity.* Botturi (2004) claims that the actual use and effectiveness of a design language strictly depend on the designer, the type of instruction to be designed, and the overall institutional and educational context. Whether the course is suitable for system-level, a mixed-mode course or a face-to-face lecture series, ECOML provides variety of graphical symbols for different components of the instructional approach. Depending on the pedagogical approach and competencies of the designer, organizational and operational context can be specified and by selecting different graphical symbols of the course components, the corresponding model can easily be achieved. Additionally, dynamic behavior of the educational system in both abstract and component levels is also made possible in ECOML. Furthermore, ECOML permits the modeling of the top-down sequential interactions among instructional components.
- *Eclectic Benefits.* Botturi (2004) says that in order to evaluate the impact of a tool, one should figure out benefits it brings to its users.

As being a visual modeling language ECOML brings the easiness to define and construct the courses by using graphical symbols. Components are designated by abstract symbols and different learning units can easily be represented using these abstract symbols. Hence, any designer can grasp the rationale of a course by examining its visual model. Graphical symbols can be cut, and/or copy pasted easily, which makes the construction process easier to manipulate. Graphical symbols are universal, therefore easier to transport cross platforms, which makes ECOML models can be reused in different contexts and across different educational platforms.

- *Course Quality Assessment.* Botturi (2004) says quality of the product as a result of the modeling process is another relevant item in evaluating the modeling language. Botturi (2004) argues what makes a course a good one? Is a course a good course because all learners achieve its objectives, although none of them were able to do any other course in the same term because of work overload? And, continues with other examples. The elements that indicate the quality of a course are many – pedagogical, administrative, institutional, teaching expertise, etc. – and they are often tightly intertwined. Probably, Botturi (2004) considers a course as an end product of an instructional modeling language, which can play the instructional model. Of course the delivered/played course quality depends very heavily on the quality of the model, in turn of the language that creates the model. In this respect, current version of ECOML, as being a Level-A editor that cannot play the instructional models in real-time, is not responsible for the quality of the model that is designed.
- *The Importance of Time.* Botturi (2004) mentions about a book that was published in 1967, authored by McLuhan M., & Fiore Q. *The medium is the message*. New York: Bantam Books. McLuhan & Fiore say that a new medium, as a new communication tool or a new language, does not bring a sudden revolution, rather smoothly presses on our perception and experience and slowly brings forth huge

modifications. The book was published almost twenty years earlier than the widespread use of Internet and claims that the dominant communication media of our time will shape the way humans think, act, and ultimately perceive the world around them. Similarly, Botturi (2004) predicts that a language (e.g. a modeling language) creates – step by step – a new communication environment, where new concepts are used and new expressions are possible, while some old concepts and expressions might become out of date, or even not possible any more. Therefore, Botturi (2004) claims that time is of paramount importance for the integration of a language in a community's practice. That's why, Botturi (2004) thinks that a complete evaluation should observe the evolution of the design practice and of the instruction over a longer period of time. Regarding the Botturi's (2004) lastly stated criteria, not only ECOML but, other educational modeling languages must also wait and see how they change the world of education.

Of course the issues that are presented above do not cover the whole set. The quality of the tool also depends on the choice of technology, different personal and cultural matters of the software developers, expressive power of the modeling language, collaboration of designers and instructors, etc..

Luca Botturi (2004) also mentions about the *Institutional Events* as being another matter of consideration in the evaluation of instructional design languages. The impact of a language could also be observed on the social dimension, as it provides for example the possibility to define different pedagogical patterns, or to create a shared repository of courses. Besides that, the tool can easily be included into the training of novice designers and/or instructors, the sharing of expertise and best practices, the reuse of design, and the communication inside and outside the instructional design team as elements of knowledge management. Obviously, ECOML satisfies all above concerns when used properly with the targeted objective is in mind.

The last key issue is the *Expressive Power* of the modeling language. Botturi (2004) mentions about the extension of the domain of objects that a design

language can describe. Can it answer the concerns like the language is able to equally well represent the instruction to be delivered with different media, or in different settings? Can it comprehend and relay the essence of different pedagogical approaches? ECOML, being a visual modeling language for the design of educational environments, does not depend on the particular pedagogical approach or the type of media or delivery settings. Its top-down representation of instructional scenarios allows designers to represent the instructional process in any method that is desired using the proper graphical symbols as long as the corresponding symbol is in the repository of the ECOML. If a symbol is not in the repository it can be easily added up as an update.

Hoyer and Brooke (2001) say that the quality of a tool is its adequacy to a problem solving activity for its users. Let's further clarify how ECOML helps to solve educational problems with an example; consider the complexity of instructional design process and suppose a new designer in charge of redesigning two courses developed by someone else. S/he has only the course materials for the former, and a complete documentation (including instructional models developed with ECOML) for the latter. Aid of the documentation, along with the measure of effectiveness (e.g. time spent), would give the measure of the impact of ECOML on this particular situation. The tool, with the help of graphical models, would help the designer to overcome the problem of grasping the former instructor's personal/cultural approach and his or her pedagogical method of delivering the course.

## **5.2. Conclusions**

This researcher's objective was to prove that educational processes could be represented better with hierarchically decomposed models and develop a visual educational modeling language where IMS-LD specified UoLs could be hierarchically designed. Each UoL, as described by Koper and Tattersall (2005), *refers to a complete, self-contained unit of education or training, such as a course, a module, a lesson, etc. The creation of a Unit of Learning involves the creation of a learning design and also the bundling of all its associated resources, either as files contained in the unit or as Web references, including assessments,*



*learning materials and learning service configuration information.* As it is demonstrated in Section 4.5, ECOML represents the instructional model of a UoL, which involves the creation of learning designs and all its associated resources, in a top-down hierarchical fashion.

Educational Component Oriented modeling Tool (ECOT) is the first implementation of ECOML. Version 1.0 of ECOT is able to produce hierarchically represented instructional models. As can be seen in Table 3, ECOT is at least as good as the well-known authoring tools that are proposed by IMS. Excluding ASK-LDT, neither of the available authoring tools is able to produce hierarchical models. As for ASK-LDT, in spite of all kinds of efforts, this researcher was not able to find out its installable software program for comparison.

### **5.3. Future Work and Recommendations**

A basic literature survey easily indicates the following current issues in Learning Design (with respect to research, development and implementation phases). As stated in the introduction part, there are several topics that are of major interest at the moment. These are the current research topics and future work for instructional modeling tools. These were analyzed in the editorials of several issues of the IEEE Educational Technology & Society journals (like IEEE Transactions on Education and/or IEEE Transactions on Learning Technology, etc.), and can be summarized as follows:

- a) The use of ontologies and semantic web principles and tools to:
  - create a new, and more precise binding for Learning Design;
  - integrate learning objects and learning designs;
  - represent specific pedagogical approaches (learning design knowledge);
  - build software agents that operate on the learning design knowledge to support in the development of Units of Learning.
- b) The use of learning design patterns:
  - to support learning designers to develop specific learning designs (e.g. collaborative designs, adaptive designs);

- that are automatically detected (pattern recognition) in Learning Design coded Units of Learning;
  - to capture best practices and learning design knowledge (relates to ontologies points c and d).
- c) The development of Learning Design Authoring and Content Management Systems, includes the following issues:
- The development of a (standard) graphical notation for learning designs;
  - How to support the reuse of Learning Design Knowledge and Learning Design Packages;
  - The development of learning design specific tools to support teachers in a specific context;
- d) The question how learning designers should be supported with tools and how teachers (the teacher as a designer) should be supported with tools;
- The integration of learning design and assessment editors in a single authoring environment.
- e) The development of *Learning Design Players*, including the following issues:
- How to integrate the variety of specifications (eg, IMS LD, IMS QTI, SCORM, IMS LIP) and the connections to other systems in an e-learning infrastructure (student administration, portfolio systems, financial systems) into a single, easy to use learning environment.
  - How to instantiate and integrate communication and collaboration services that are called by a Learning Design. Eg, forums, wiki's, chats; are generic service oriented architectures suitable to do the job? At what costs?
  - How to design a usable, powerful and flexible user-interface for a Player environment?

- How to integrate Learning Design into existing Learning Management Systems (like Moodle, Blackboard and LAMS)?
  - How to integrate Learning Design Authoring Systems and Learning Design Players, including the question how to deal with runtime adaptations?
- f) How to use an integrated set of Learning Design tools in an integrated way in a variety of settings (e.g. in universities, training, blended learning).

Considering all of the above mentioned research topics, there are quite a few recommendations that can be done to improve ECOML as an instructional tool. Therefore, as a future work;

- Overview diagrams can be constructed to show the dependencies between activities and the activity flow (a sort of graphical time table). Which will make the ECOML model executable by any IMS-LD compliant Learning Design Player.
- ECOML's integration with OpenCourseWare can be investigated and proper interface modules can be developed.
- ECOT version 1.0+ can be developed to export SCORM compliant zip files, which can be imported and executed by LMSs like Moodle.
- ECOT version 1.0+ can be developed to execute the instructional models that were developed in previous versions.
- ECOT version 1.0+ can be developed to make ECOT able to import all imsmanifest.xml files, which are created by other 3<sup>rd</sup> party developers, and execute them properly as a LD Player.

## REFERENCES

- Agostinho S, Bennett S, Lockyer L and Harper B. Integrating learning objects with learning designs. Crisp G., Thiele D, Scholten I, Barker S, and Baron J.(Eds), Interact, Integrate, Impact: Proceedings of the 20th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education. Adelaide, 7-10 December 2003. Last accessed on December 20, 2009 from <http://www.ascilite.org.au/conferences/adelaide03/docs/pdf/571.pdf>
- Azgur S. M. and Dogru A. H., (2009). A Component Oriented Modeling Language for Top Down Educational Design. Twelfth Transdisciplinary Conference-Workshop on Integrated Design & Process Science November 1-5, 2009 Montgomery, Alabama 8 pp.
- Bartak Roman, "Constraint Programming: In Pursuit of the Holy Grail" in Proceedings of Week of Doctoral Students (WDS99), Part IV, MatFyzPress, Prague, June 1999, pp. 555-564. Last accessed on December 26, 2009 from <http://kti.mff.cuni.cz/~bartak/downloads/WDS99.pdf>
- Bass, L; Clements, P; Kazman, R; Software Architecture in Practice – Second Edition, SEI Series in Software Engineering, Addison Wesley, 2nd Edition, 2003.
- Booch G., Rumbaugh J., Jacobson I., "The Unified Modeling Language User Guide", Addison Wesley, April 2000.
- Botturi Luca, (2004). Visual Languages for Instructional Design: an Evaluation of the Perception of E2ML. *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (EDMEDIA) 2004* (pp. 243-250). Lugano, Switzerland, 2004.
- Botturi Luca, (2005). *A framework for the evaluation of visual languages for instructional design: the case of E<sup>2</sup>ML*. Journal of Interactive Learning

Research ISSN: 1093-023X Dec 22, 2005. Also on The Free Library: retrieved July 09, 2009 from [http://www.thefreelibrary.com/A framework for the evaluation of visual languages for instructional...-a0139682090](http://www.thefreelibrary.com/A+framework+for+the+evaluation+of+visual+languages+for+instructional...-a0139682090)

Burgos Daniel, Griffiths David, (2005). E-learning specifications. An introduction. Last accessed on October 21, 2009 online <http://dspace.ou.nl/handle/1820/547>

Caeiro Rodriguez, M.; Anido Rifon, L.; Llamas Nistal, M., (2004). Towards IMS-LD extensions to actually support heterogeneous learning designs: a pattern-based approach. *IEEE International Conference on Advanced Learning Technologies. Proceedings*. 30 Aug.-1 Sept. 2004 Page(s): 565 – 569. Retrieved September 30, 2009 from <http://ieeexplore.ieee.org/search/wrapper.jsp?arnumber=1357478>

Christiaans H., & Venselaar, K. (2005). *Creativity in Design Engineering and the Role of Knowledge: Modeling the Expert*. International Journal of Technology and Design Education, 2005(15), 217-236.

Clemens P; Bachmann, F; Bass, L; Garlan, D; Ivers, R; Nord, R; Stafford, J; Documenting Software Architecture – Views and Beyond. Addison-Wesley. 2<sup>nd</sup> Edition. 2003.

Davis H., DiBiase D., Fill K., Martin D., Rees P., (2007). The DialogPLUS project consortium. 2007 (January). Last accessed online January, 15, 2010, either <http://www.dialogplus.soton.ac.uk/outcomes.php>, or [http://www.dialogplus.soton.ac.uk/outcomes%5Cdialogplus\\_final\\_report.pdf](http://www.dialogplus.soton.ac.uk/outcomes%5Cdialogplus_final_report.pdf)

Dodero Juan Manuel, Tattersall Colin, Burgos Daniel, Koper Rob., (2006). Transformational techniques for model-driven authoring of learning designs. Unpublished Paper. Last accessed online September, 01, 2009, <http://dspace.ou.nl/handle/1820/783>

- Dogru A. H., TR-99-3, Component-Oriented Software Engineering Modeling Language: COSEML. Middle East Technical University, Department of Computer Engineering. Tech. Rep. METU-CENG-1999-3.
- Dogru A, H., Tanik M. M., A Process Model for Component Oriented Software Engineering, *IEEE Software*, Vol.20, No. 2, March/April 2003: pp.34-41
- Gibson Cliff and Harlow Stephen, (2004). E-learning Standards Overview. Last accessed online September, 01, 2009, <http://www.steo.govt.nz/download/DraftStandardsOverview.pdf>
- Griffiths, David; Blat, Josep, (2005). The Role Of Teachers In Editing And Authoring Units Of Learning Using IMS Learning Design. *International Journal on Advanced Technology for Learning*, Special Session on "Designing Learning Activities: From Content-based to Context-based Learning Services", volume 2, issue 4, October 2005. Last accessed December 05, 2009 online [http://dspace.ou.nl/bitstream/1820/586/1/griffiths\\_atl\\_2005.pdf](http://dspace.ou.nl/bitstream/1820/586/1/griffiths_atl_2005.pdf)
- Hernández-Leo, D., Harrer, A., Dodero, J. M., Asension-Pérez, J. I., & Burgos, D. (2006). Creating by reusing Learning Design solutions. Proceedings of 8th Simposio Internacional de Informática Educativa, León, Spain: IEEE Technical Committee on Learning Technology. Last accessed December 05, 2009 online <http://dspace.ou.nl/handle/1820/788>.
- Hernández-Leo, D, Villasclaras-Fernández, E. D., Asensio-Pérez, J. I, Dimitriadis, Y., Jorrín-Abellán, I. M., Ruiz-Requies, I., Rubia-Avi, B. (2006). COLLAGE: A collaborative Learning Design editor based on patterns. *Educational Technology & Society*, 9 (1), 58-71.
- Hoyer, R.W. & Brooke B.Y. (2001). What is Quality? Quality Progress, July 34(7), 53-62
- IEEE Learning Technology Standards Committee, IEEE Computer Society. Draft Standard for Learning Technology-Learning Technology Systems Architecture (LTSA). IEEE P1484.1/D9, 2001-11-30, 2001.

- Kara A., M. S. Thesis study, “A Graphical Editor for Component Oriented Modeling”. Department of Computer Engineering, METU, April 2001.
- Karampiperis P, Sampson D. (2005). Towards next generation activity-based Web-based educational systems. *Advanced Learning Technologies*, 2005. ICALT 2005. Fifth IEEE International Conference on (2005), pp. 868-872.
- Karampiperis P, Sampson D. (2007). Towards a Common Graphical Language for Learning Flows: Transforming BPEL to IMS Learning Design Level A Representations. *IEEE CS Digital. ICALCT 2007*. Seventh IEEE International Conference on Advanced Learning Technologies, pp 798-800.
- Knight, C., Gasevic, D., Richards, G., (2005), “Ontologies to integrate learning design and learning content”. *Journal of Interactive Media in Education*. 2005/07.
- Koper Rob, (2001). “Modeling units of study from a pedagogical perspective the pedagogical meta-model behind EML”. Last accessed on December 05, 2009, <http://dspace.ou.nl/bitstream/1820/36/1/Pedagogical%20metamodel%20behind%20EMLv2.pdf>
- Koper, Rob, Manderveld, J., (2004). “Educational Modeling Language”. *British Journal Of Education Technology*.
- Koper Rob, Olivier Bill, (2004). Representing the Learning Design of Units of Learning. *Educational Technology & Society*, 7(3), pp 97-111.
- Koper Rob (2005). “Modelling Pedagogy with IMS Learning Design. The use of IMS LD to notate units of learning”. UNFOLD CoP Meeting, Braga, 15 June 2005. Last accessed on January 05, 2010, <http://hdl.handle.net/1820/363>
- Koper Rob (2009). IMS Learning Design State-of-the-Art. Open University of the Netherlands. 6 October 2009. Nice IMS-LD Presentation. Last accessed on December 17, 2010 at <http://dspace.ou.nl/handle/1820/2043>

- Miao, Y., Van der Klink, M., Boon, J., Sloep, P. B., & Koper, R. (2009). Enabling Teachers to Develop Pedagogically Sound and Technically Executable Learning Designs [special issue: Learning Design]. *Distance Education*, 30(2), 259-276. Also accessed online on December 05, 2009 online <http://dspace.ou.nl/handle/1820/1605>
- Mili H., Mili F., Mili A., Boite P., 1995. “*Reusing Software: Issues and Research Directions*,” *IEEE Transactions on Software Engineering*, Vol. 21, Issue 6, pp. 528-562, June 1995
- Morris Stephen, (1994). Revising Knowledge: A Hierarchical Approach. *Proceedings of TARK V*. Last accessed on 30.November.2009, [http://www.tark.org/proceedings/tark\\_mar13\\_94/p160-morris.pdf](http://www.tark.org/proceedings/tark_mar13_94/p160-morris.pdf).
- Neumann, Susanne and Petra Oberhuemer (2009, to appear). User Evaluation of a Graphical Modeling Tool for IMS Learning Design, International Conference on Web-based Learning (ICWL) 2009. To be published in: *Lecture Notes in Computer Science*, Springer. Last accessed December 08, 2009 online <http://www.heyerlevel.de/calimero/tools/proxy.php?id=12856>.
- Ozcinar Zehra, (2009). *The topic of instructional design in research journals: A citation analysis for the years 1980-2008*. *Australian Journal of Educational Technology*, 2009, 25(4), 559-580.
- Paquette Gilbert, (2004). Educational Modeling Languages, from an Instructional Engineering Perspective, in R. McGreal (ed), *Online education using learning objects*, pp 331-346. London : Routledge/Falmer. Last accessed on December 05, 2009, online at <http://www.liceftelug.quebec.ca/Portals/29/docs/pub/ingenierie/Article%20EML-MISAedited.doc>
- Paquette, G., Léonard, M., Lundgren-Cayrol, K., Mihaila, S., Gareau, D. (2006). Learning Design based on Graphical Knowledge-Modelling. *Educational Technology & Society*, 9 (1), 97-112.



Richards Jay W., (2005). What Intelligent Design Is—and Isn't: The more scientifically sophisticated we get, the stronger the argument for intelligent design. May 13, 2005. Last accessed December 27, 2009 online <http://www.discovery.org/scripts/viewDB/index.php?command=view&printerFriendly=true&id=2571>

Sampson D., Karampiperis P., and Panayiotis Zervas. (2005). Developing Web-Based Learning Scenarios Using the IMS Learning Design: The ASK-LDT Environment. Web Information Systems Engineering – WISE 2005 Workshops. WISE 2005 International Workshops, New York, NY, USA November 20-22, 2005. Volume 3807/2005, pp. 104-113. Last accessed on Dec 12, 2009 online at <http://springerlink.com/content/703779838152rjr3> or pdf can be found at <http://springerlink.com/content/703779838152rjr3/fulltext.pdf>

Sampson, D.G. Karampiperis, P. (2006). Towards Next Generation Activity-Based Learning Systems. *International Journal on E-Learning*, 5(1), 129-149. Last accessed online January 15, 2010 at <http://www.editlib.org/f/21766>

Schneider Daniel, K. (2009). He is the coordinator of EduTechWiki, which is about Educational Technology ([http://edutechwiki.unige.ch/en/Main\\_Page](http://edutechwiki.unige.ch/en/Main_Page), last accessed online on December 06, 2006). Also [http://edutechwiki.unige.ch/en/User:Daniel\\_K.\\_Schneider](http://edutechwiki.unige.ch/en/User:Daniel_K._Schneider) and also <http://tecfa.unige.ch/tecfa-people/schneider.html>, all of them last accessed on December 06, 2009.

Sodhi T, Miao Y, Brouns F and Koper R. (2007). *Design Support for non-expert authors in the creation of units of learning - a first exploration*. Last accessed at 09.July.2009, <http://dspace.ou.nl/handle/1820/984> or, <http://en.scientificcommons.org/35787533>.

Tanik Murat M., Tanju Murat N., Dogru Ali H., Azgur Serhat M., “Generating anytime anywhere knowledge units as learning competencies,” Middle East

Technical University, Ankara, Turkiye. Tech. Rep. METU-CENG-2009-01, July 2009.

Vogten, H., Tattersall, C., Koper, R., van Rosmalen, P., Brouns, F., Sloep, P., van Bruggen, J. & Martens, H. (2006). "Designing a Learning Design Engine as a Collection of Finite State Machines". *International Journal on E-Learning*, Vol. 5, Issue 4, pp. 641-661. October 2006. Association for the Advancement of Computing in Education (AACE), Chesapeake, VA, USA.

Wilson, S; Oliver, B; Jeyes, S; Powell, A; *A Technical Framework to Support e-Learning*; JISC paper; vol. 2005. JISC, 2004; downloaded from [www.jisc.org](http://www.jisc.org)]

White, S. A. (2004) *Introduction to BPMN*. Last accessed at December 23, 2009.  
[http://www.bpmn.org/Documents/Introduction\\_to\\_BPMN.pdf](http://www.bpmn.org/Documents/Introduction_to_BPMN.pdf)

White, S. A. (2005) An Example of Using BPMN to Model a BPEL Process. Last accessed at December 23, 2009.  
[http://www.bpmn.org/Documents/Mapping\\_BPMN\\_to\\_BPEL\\_v3.pdf](http://www.bpmn.org/Documents/Mapping_BPMN_to_BPEL_v3.pdf)