

**SEMANTICALLY INTEGRATED  
E-LEARNING INTEROPERABILITY AGENT**

**DICKY EKKLESIA**

**DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES**

**UNIVERSITI TEKNOLOGI PETRONAS**

**FEBRUARY 2008**

**STATUS OF THESIS**

Title of thesis Semantically Integrated E-Learning Interoperability Agent

I DICKY EKKLESIA

hereby allow my thesis to be placed at the Information Resource Center (IRC) of Universiti Teknologi PETRONAS (UTP) with the following conditions:

1. The thesis becomes the property of UTP.
2. The IRC of UTP may make copies of the thesis for academic purposes only.
3. This thesis is classified as

Confidential

Non-confidential

If this thesis is confidential, please state the reason:

\_\_\_\_\_

The contents of the thesis will remain confidential for \_\_\_\_ - \_\_\_\_ years.

Remarks on disclosure:

\_\_\_\_\_

Endorsed by



DICKY EKKLESIA  
Bukit Pamulang Indah V  
Blok A1 No. 3 (Jl. Cicakrawa)  
Pamulang Timur, Ciputat, 15417  
Tangerang, Banten, Indonesia



AZWEEN ABDULLAH  
Universiti Teknologi PETRONAS  
Bandar Seri Iskandar, Tronoh, 31750  
Perak, Malaysia

Date: 03 March '08

Date: 3/3/08

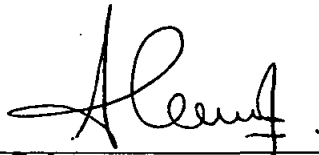
UNIVERSITI TEKNOLOGI PETRONAS

Approval by Supervisor (s)

The undersigned certify that they have read, and recommend to The Postgraduate Studies Programme for acceptance, a thesis entitled "Semantically Integrated E-Learning Interoperability Agent" submitted by (Dicky Ekklesia) for the fulfillment of the requirements for the degree of Master of Science in Information Technology.

\_\_\_\_\_  
Date

Signature :



\_\_\_\_\_  
Dr Azween Bin Abdullah  
Senior Lecturer  
Information Technology/Information Systems  
Universiti Teknologi PETRONAS  
31750 Tronoh  
Perak Darul Ridzuan

Main Supervisor :

Date :

3/3/08.

Signature :

Co-Supervisor :

Date :

TITLE PAGE

UNIVERSITI TEKNOLOGI PETRONAS

Semantically Integrated E-Learning Interoperability Agent

By

Dicky Ekklesia

A THESIS

SUBMITTED TO THE POSTGRADUATE STUDIES PROGRAMME

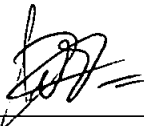
AS A REQUIREMENT FOR THE  
DEGREE OF MASTER OF SCIENCE  
INFORMATION TECHNOLOGY

BANDAR SERI ISKANDAR,  
PERAK

FEBRUARY, 2008

## DECLARATION

I hereby declare that the thesis is based on my original work except for quotations and citations which have been duly acknowledge. I also declare that it has not been previously or concurrently submitted for any other degree at UTP or other institutions.

Signature :  \_\_\_\_\_

Name : DICKY EKKLESIA \_\_\_\_\_

Date : 03 March '08 \_\_\_\_\_

## ABSTRACT

Educational collaboration through e-learning is one of the fields that have been worked on since the emergence of e-learning in educational system. The e-learning standards (e.g. learning object metadata standard) and e-learning system architectures or frameworks, which support interoperation of correlated e-learning systems, are the proposed technologies to support the collaboration. However, these technologies have not been successful in creating boundless educational collaboration through e-learning. In particular, these technologies offer solutions with their own requirements or limitations and endeavor challenging efforts in applying the technologies into their e-learning system. Thus, the simpler the technology enhances possibility in forging the collaboration.

This thesis explores a suite of techniques for creating an interoperability tool model in e-learning domain that can be applied on diverse e-learning platforms. The proposed model is called the e-learning Interoperability Agent or eIA. The scope of eIA focuses on two aspects of e-learning: Learning Objects (LOs) and the users of e-learning itself. Learning objects that are accessible over the Web are valuable assets for sharing knowledge in teaching, training, problem solving and decision support. Meanwhile, there is still tacit knowledge that is not documented through LOs but embedded in form of users' expertise and experiences. Therefore, the establishment of educational collaboration can be formed by the users of e-learning with a common interest in a specific problem domain.

The eIA is a loosely coupled model designed as an extension of various e-learning systems platforms. The eIA utilizes XML (eXtensible Markup Language) technology, which has been accepted as the knowledge representation syntax, to bridge the heterogeneous platforms. At the end, the use of eIA as facilitator to mediate intercommunication between e-learning systems is to engage the creation of semantically Federated e-learning Community (FeC). Eventually, maturity of the FeC is driven by users' willingness to grow the community, by means of increasing the e-learning systems that use eIA and adding new functionalities into eIA.

## ABSTRAK

Kerjasama pendidikan melalui e-pembelajaran adalah salah satu bidang yang telah dimulakan sejak kemunculan e-pembelajaran dalam sistem pendidikan. E-pembelajaran berpiawai (e.g. belajar objek metadata piawai) dan e-pembelajaran seni bena sistem atau rangka, yang menyokong interoperation berhubung kait dengan sistem e-pembelajaran, adalah teknologi-teknologi yang dicadangkan untuk menyokong kerjasama pendidikan. Walaubagaimanapun, teknologi-teknologi ini tidak berhasil dalam mencipta kerjasama pendidikan yang tiada had atau batasan melalui e-pembelajaran. Khususnya, teknologi-teknologi ini menawarkan penyelesaian dengan syarat-syarat dan juga batasan mereka yang tersendiri dan menyerapkan usaha yang mencabar dalam mengaplikasikan teknologi ini ke dalam sistem e-pembelajaran. Oleh itu, teknologi yang lebih mudah dapat menambah kemungkinan dalam menempa kerjasama pendidikan.

Tesis ini menunjukkan kajian tentang teknik-teknik yang sesuai untuk menghasilkan model alat interopera dalam e-pembelajaran dimana ia boleh diaplikasikan pada pelbagai platform e-pembelajaran. Model cadangan adalah dipanggil e-pembelajaran Interoperability Agent atau eIA. Skop eIA menumpukan pada dua aspek e pembelajaran: Belajar Objek-objek (LOs) dan pengguna-pengguna e-pembelajaran itu sendiri. Belajar objek-objek yang diakses melalui Web adalah aset yang bernilai untuk berkongsi pengetahuan dalam pendidikan, membuat latihan, menyelesaikan masalah dan membuat keputusan. Sementara itu, di sana masih tersirat ilmu pengetahuan yang wujud yang tidak didokumentasi terus melalui LOs tetapi diserapkan melalui kepakaran dan pengalaman pengguna-pengguna. Oleh itu, penubuhan kerjasama pendidikan boleh dibentuk oleh pengguna-pengguna e-pembelajaran dengan satu minat yang sama dalam satu masalah yang khusus.

eIA adalah satu model bebas yang direka sebagai satu perluasan bercorak dari berbagai jenis platform e-pembelajaran. eIA menggunakan XML (eXtensible Markup Language) teknologi, yang telah diterima sebagai sintaksis perwakilan pengetahuan, untuk menghubungkan platform-platform heterogen. Pada akhirnya, penggunaan eIA sebagai medium untuk menyelesaikan perhubungan antara sistem-sistem e-

pembelajaran adalah untuk melibatkan penciptaan secara semantik Federated e-learning Community (FeC). Akhirnya, kematangan FeC didorong oleh kesediaan pengguna-pengguna untuk memajukan masyarakat, dengan cara menambahkan sistem-sistem e-pembelajaran yang menggunakan eIA.



## ACKNOWLEDGEMENTS

First of all, I cannot thank enough to my family for always supporting me in my personal and academic endeavors. Their relentless encouragement has enabled me to persevere even in times of duress.

I would like to thank my supervisor Dr. Azween Abdullah for his supervision and guidance throughout the whole work with this thesis. I greatly appreciate his encouragement to go ahead with my work especially in time of lack of my self confidence.

I would also like to thank Dr. Etienne Schneider for his helpful comments, comprehension, and reviewing my research papers even with very short notices. I also want to thank to Dr. Ahmad Kamil Mahmood for his support and leading as head of Computer and Information Sciences Department, UTP.

My sincere thanks to all the administrative staffs: Mr. Azful, Kak Norma, Kak Haslina, Mr. Fadil Ariff and the rest of postgraduate office staffs for their assistance during my time in UTP. Last but not least, I would also like to address my appreciation to my former colleagues, faculty, and staffs of Informatics Engineering Department, Institute of Technology Bandung. Thank you for all the experiences and knowledge. Without them, I would not have the opportunity to pursue my degree further.

## TABLE OF CONTENTS

STATUS OF THESIS.....	i
TITLE PAGE.....	iii
DECLARATION.....	iv
ABSTRACT.....	v
ABSTRAK.....	vi
ACKNOWLEDGEMENTS.....	viii
TABLE OF CONTENTS.....	ix
LIST OF TABLES.....	xii
LIST OF FIGURES.....	xiii
CHAPTER ONE : INTRODUCTION.....	1
1.1 Introduction.....	1
1.2 The Needs of Educational Collaboration.....	2
1.3 e-learning Interoperability Agent.....	4
1.4 Objectives and Methodology.....	6
1.5 Thesis Organization.....	7
CHAPTER TWO : LITERATURE REVIEW.....	8
2.1 Related Work.....	8
2.1.1 E-learning Interoperability.....	10
2.2 Web Services.....	11
2.2.1 Web Services Description Language (WSDL).....	13
2.2.2 Simple Object Access Protocol (SOAP).....	15
2.2.3 Universal Description, Discovery, and Integration (UDDI).....	16
2.3 Semantics.....	16
2.3.1 Information Integration.....	16
2.3.2 Information Retrieval and Information Extraction.....	18
2.3.3 Semantic Integration.....	19
2.4 Standards in E-Learning.....	20
2.4.1 Metadata.....	22
2.4.2 Learner Profile.....	24
2.5 Mapping of Data Elements.....	26
2.6 eXtensible Markup Language.....	27

2.6.1 eXtensible Stylesheet Language Transformation (XSLT).....	28
2.7 Summary .....	31
CHAPTER THREE : E-LEARNING INTEROPERABILITY AGENT.....	32
3.1 Overview of eIA .....	32
3.1.1 Users of eIA .....	32
3.2 eIA Communication Layer .....	35
3.3 eIA Data Layer.....	39
3.4 eIA Query GUI Module.....	40
3.5 eIA Flow State Diagram .....	42
3.6 Generic Interoperability Agent .....	44
3.7 Summary .....	46
CHAPTER FOUR : EIA SEMANTIC INTEGRATION .....	47
4.1 eIA's Metadata.....	47
4.1.1 eIA Learning Object Metadata.....	47
4.1.2 eIA Learner Information Metadata .....	50
4.2 eIA's Basic Services .....	51
4.3 eIA's Extended Services.....	54
4.4 Summary .....	56
CHAPTER FIVE : EVALUATION .....	57
5.1 Prototyping.....	57
5.1.1 Test Bed Platforms.....	58
5.1.1.1 Moodle .....	59
5.1.1.2 ATutor .....	63
5.1.2 eIA Services Implementation.....	66
5.1.2.1 eIA Administration.....	66
5.1.2.2 eIA Dynamic Function Invocation .....	66
5.1.2.3 eIA Learning Objects Services.....	68
5.1.2.4 eIA User Profiles Services .....	70
5.2 Federated e-learning Community .....	73
5.2.1 Message Broadcasting .....	75
5.3 Summary .....	77
CHAPTER SIX : CONCLUSION AND RECOMMENDATIONS.....	78
6.1 Conclusion .....	78
6.2 Recommendations for Further Works.....	79

REFERENCES .....	81
PUBLICATIONS.....	86
APPENDIX A : EIA METADATA SCHEMA .....	87
A.1 eIA Learning Object Metadata.....	87
A.2 eIA Learner Information Metadata .....	94
APPENDIX B : EIA CODE INFORMATION .....	98
B.1 eIA Pilot Prototype Structure .....	98
B.1.1 eIA Web Services Core (eIA_ws_core).....	99
B.1.2 eIA Data Mediation Module (eIA_xslt_lib).....	101
B.1.3 eIA Query GUI Module (eIA_ui_frontEnd and eIA_ui_core).....	102
B.2 eIA Code Statistics .....	103
B.2.1 Mods Package .....	103
B.2.2 Blocks Package .....	104

## LIST OF TABLES

Table 2-1: Techniques used for Schema Integration [Sheth et al., 2005].....	17
Table 2-2: Various Semantic Integration Architectures Characteristics.....	19
Table 2-3: XSLT Transformation Code Example .....	30
Table 3-1: eIA Service Directory.....	36
Table 3-2: Generic Interoperability Agent Mapping .....	45
Table 5-1: Moodle LO Data Mapping .....	62
Table 5-2: Moodle User Profiles Data Mapping.....	62
Table 5-3: ATutor LO Data Mapping.....	65
Table 5-4: ATutor User Profiles Data Mapping .....	65
Table A-1: eIA Learning Object Metadata .....	87
Table A-2: eIA Learner Information Metadata.....	94
Table B-1: A Skeleton Code for the eIA Server Class .....	99
Table B-2: A Skeleton Code for the eIA Client Class.....	100
Table B-3: A Skeleton Code for the eIA Service Schema.....	101
Table B-4: A Skeleton Code for the eIA Service Mapper.....	102
Table B-5: A Skeleton Code for the eIA UI Front End.....	102
Table B-6: A Skeleton Code for the eIA UI Core .....	103

## LIST OF FIGURES

Figure 1-1: Overview of VLE's Interoperability.....	5
Figure 2-1: An Integrated View of Service Interoperability Dimensions.....	8
Figure 2-2: Overview of Web Services .....	13
Figure 2-3: Typical WSDL Document Structure.....	14
Figure 2-4: SOAP Message Format.....	15
Figure 2-5: LOM Metadata Schema .....	24
Figure 2-6: Overview of XSLT.....	28
Figure 2-7: Minimal but Complete XSLT Stylesheet.....	29
Figure 3-1: e-learning Interoperability Agent Structure .....	32
Figure 3-2: eIA Administration .....	33
Figure 3-3: Learning Objects Use Case .....	34
Figure 3-4: User Profiles Use Case.....	35
Figure 3-5: eIA Client Authorization.....	37
Figure 3-6: eIA Web Services.....	38
Figure 3-7: eIA Service Mapping .....	40
Figure 3-8: eIA Data Mediation Module .....	40
Figure 3-9: eIA Query GUI Module.....	41
Figure 3-10: eIA Menu Structure.....	41
Figure 3-11: Overall eIA Flow State Diagram .....	43
Figure 4-1: eIA LOs Rate Filtering.....	48
Figure 4-2: eIA Learning Object Metadata Schema .....	49
Figure 4-3: eIA Learner Information Metadata Schema.....	50
Figure 4-4: eIA WSDL Schema.....	52
Figure 4-5: eIA Network Tree Sample .....	53
Figure 4-6: eIA's Broadcast Algorithm.....	54
Figure 4-7: Sketch of XSLT – WSDL Library Template .....	55
Figure 5-1: eIA Implementation Overview.....	58
Figure 5-2: eIA's Client/Server Architecture .....	59
Figure 5-3: Moodle and e-learning Interoperability Agent.....	60
Figure 5-4: Extracted Moodle Database .....	61
Figure 5-5: ATutor and e-learning Interoperability Agent .....	63

Figure 5-6: Extracted ATutor Database.....	64
Figure 5-7: eIA Service Directory .....	66
Figure 5-8: eIA Dynamic Functions Invocation List.....	67
Figure 5-9: eIA Dynamic Functions Invocation Result.....	68
Figure 5-10: Moodle Publish LO Page .....	69
Figure 5-11: ATutor Publish LO Page.....	69
Figure 5-12: Search for LO(s) Form.....	70
Figure 5-13: Search for LO(s) Result .....	70
Figure 5-14: Moodle User Profile Page .....	71
Figure 5-15: ATutor User Profile Page.....	72
Figure 5-16: Search for User(s) Form.....	72
Figure 5-17: Search for User(s) Result .....	73
Figure 5-18: Example of Federated e-learning Community .....	74
Figure 5-19: Extended Federated e-learning Community .....	74
Figure 5-20: FeC Messages Broadcasting Flow Example.....	76
Figure B-1: eIA Prototyping Diagram in Moodle and ATutor.....	98

## CHAPTER ONE : INTRODUCTION

In this chapter, an introduction to the conducted research is presented. First, an overview of e-learning as the research area is given. Thereafter, an overview of the problem and the proposed solution is given. Furthermore, an outline of the remaining chapter of this thesis is included.

### 1.1 Introduction

Technology is a change agent and the new technology generates a full new paradigm. The advent of Internet and evolvement of web technologies have fundamentally changed the way education is done. Most of the educational institutions have used e-learning or Virtual Learning Environment (VLE) as media to complement the traditional teaching-learning process. Briefly, e-learning is the delivery of a learning, training or education program by electronic means [Stockley, 2003]. A general agreement seems to exist regarding roles played by people in a learning environment as well as regarding the core functionality of modern e-learning platforms. The main players in e-learning systems are the teachers, the learners, and the administrators. Teachers create formal contents, which are used in formal teaching-learning process and stored under VLE's database. These contents are utilized by the learners as one of the sources to achieve knowledge and gain their learning goals. Meanwhile administrators control the VLE so that the system is available all the time for the users.

Contents consumed by learners and created by teachers are commonly handled, stored, and exchanged in units of Learning Objects (LOs). The IEEE Learning Technology Standards Committee (LTSC) gives the following definition:

*“A Learning Object is any entity, digital or non-digital, which can be used, re-used or referenced during technology supported learning. Examples of technology-supported learning include computer-based training systems, distance learning systems, and collaborative learning environments. Examples of learning objects include multimedia content, instructional content, learning objectives, instructional software and software tools, and persons,*



*organizations, or events referenced during technology supported learning.”*  
[IEEE LTSC, 2002]

Contents may also be created by the learners based on their interest, competencies, and knowledge. These kinds of contents are categorized as informal learning contents. Informal learning contents cover any learning that takes place beyond the classroom and formal curriculum, including learning for hobbies, curiosity, personal development, community involvement, and everyday survival. These kinds of contents may not be well documented in form of learning objects, but can be recorded as information about the learner in form of learner profile so that their profile may be shared among other learners and teachers. Learners can gain particular knowledge they need to know from other learners by listing based on relevant keywords of the particular knowledge and enquiring the related learners who possibly can discuss and share about the matters.

Ideally, either LOs or learners' profile can be exchanged between different VLE systems. To achieve this exchangeability, technical awareness of the e-learning system has to be acknowledged. In a general point of view, a fine-grained database-oriented e-learning environment could be conceived as a three layer system [Arapi et al., 2005]. At the lowest level, there is a (usually) relational database management system. At the middle level, there is the specific database of the system along with a number of database transactions used to store and retrieve data including software components for the creation of dynamic HTML pages in case of web-based solutions. At the upper most level there are various user-centered applications providing functions for browsing, authoring, user communication, etc. However, the exchangeability is one of the ways to fulfill the needs of educational collaboration.

## **1.2 The Needs of Educational Collaboration**

Various developments have led to the design of much different kinds of e-learning platforms and tools. While more and more institutions use VLE systems, the need for educational collaboration between them has emerged. To comply with this need, a number of research works have been initiated in the last few years. These research

works try to solve problems which hold back the need, such as: diversity of VLE systems architectures developed by different vendors and various formats or forms of learning objects created by numerous unique authors (teachers). They can be distinguished into:

- E-learning System Architectures

These works propose e-learning architectures or frameworks that support for collaboration. If the proposed architectures used, then collaboration between e-learning systems with the same or related architecture can be forged. Luo et al. [Luo et al., 2006] and Arapi et al. [Arapi et al., 2003] works are some examples of the collaboration architectures.

- E-learning Standards

The goal of the standards is to provide standardized data structures and communication protocols for e-learning objects and cross-system workflows. IEEE LTSC and IMS/GLC (Instructional Management Systems Global Learning Consortium, Inc.) are some examples of organizations working on the standards.

- Collaboration and Interoperability Tools

This category groups tools that enable collaboration or interoperability between users but not necessarily part of the e-learning system itself. ACollab [ATRC, 2004], coMentor [HUD, 2000], and Virtual Whiteboard [Eclipse, 2007] are some examples of the collaboration tools.

However, this work focuses in creating e-learning interoperability tool as an extension of pre-existent e-learning systems, namely e-learning Interoperability Agent (eIA). Interoperability generally refers to “the ability of two or more systems or components to exchange information and to use the information that has been exchanged” [IEEE, 1990]. Meanwhile, ISO/IEC 2382 Information Technology Vocabulary defines interoperability as “the capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units”. Euzenat defines several possible levels of interoperability [Euzenat, 2001]:

- Encoding – being able to segment the representation in characters;
- Lexical – being able to segment the representation in words (or symbols);

- Syntactic – being able to structure the representation in structured sentences (or formulas or assertions);
- Semantic – being able to construct the propositional meaning of the representation;
- Semiotic – being able to construct the pragmatic meaning of the representation (or its meaning in context).

This layered presentation is arguable in general. It is not as strict as it seems. It makes sense because each level can not be achieved if the previous ones have not been completed. Nevertheless, this work focuses on semantic level of the interoperability. eIA works based on semantic integration.

Semantics (the study of meaning) is usually defined to investigate the relation of signs to their corresponding objects [Aroyo et al., 2006]. In relation to this research, semantic integration thus denotes the study of how to bridge differences between e-learning systems on two levels:

1. Access level: where system and organizational boundaries have to be crossed by creating a standardized interface that sharing of system-internal services in a loosely-coupled way.
2. Meaning level: where format agreements about transported data have to be made in order to permit their correct interpretation. This means that semantics should be understood, verified against an agreed standard, and used to endorse and validate reliable information exchange.

### **1.3 e-learning Interoperability Agent**

As an e-learning interoperability tool, eIA has characteristics as follows: work with any content and any e-learning system, and should make the less possible modification to pre-existent e-learning system. Figure 1-1 shows an overview of eIA and its characteristics compared to other interoperability tool. Without eIA, VLEs must make their own tool, i.e.: IT-A, IT-B, and IT-C, to intercommunicate with other VLEs conform the same or correlative standard.

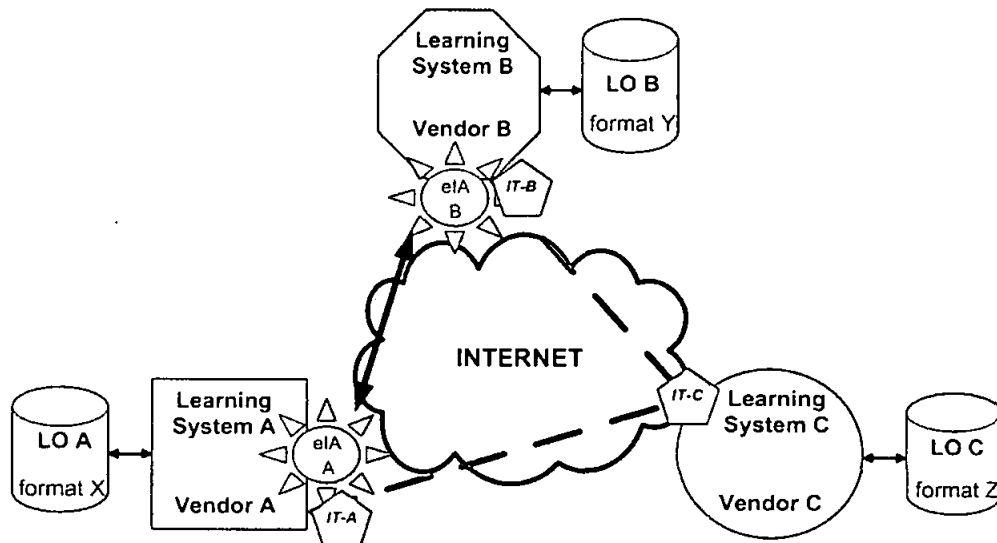


Figure 1-1: Overview of VLE's Interoperability

The core goal of eIA is to enrich academicians' experiences by extending VLE interoperability functional capabilities through a single, common integration approach across multiple VLE systems. Main objectives of eIA's functionalities focus on two things:

1. Learning Objects

Learning objects can be searchable either by categories or by keywords using a search engine. Learning objects do not have to be instructional materials for formal courses as elaborated in section 1.1. They can be small granules of information content that are useful to some learners. Typically this is where the students learn with and from one another, collaboratively. These learning objects (formal and informal) are stored inside every VLE's database and usually not accessible freely (by accessing through the search engine's given URLs) but need to give the authorized particular VLE account. eIA bridges this problem through eIA enterprise authorization so that all VLEs federated by eIA are accessible to share, search for, locate and retrieve appropriate learning objects.

## 2. User Profiles

The nature of interoperability between systems implies that there is a distributed process of sharing and exchanging user profiles. This process includes on the one hand providing profile information and on the other hand consuming user profiles. Search engines provide facilities to look for name and information correlated with the particular name, but only for name which already produces something that is accessible or searchable through the Internet. eIA facilitates users to find other users from various VLE, not only searchable users as mentioned before but also other users who are knowledgeable in particular subject domain so that they can exchange ideas, do discussion, etc.

To summarize, the vision is creation of federated e-learning community through eIA regardless of where they come from as long as it is educational related. E-learning users with similar interest in a specific subject or domain will be able to create and maintain their particular learning community to share their knowledge captured either in the form of learning objects or sharing of experiences. However, this work has not covered security issues related to messages passing between VLEs through eIAs.

## 1.4 Objectives and Methodology

As discussed in section 1.2 and 1.3, the main objectives of this work were as follows:

1. To design an e-learning Interoperability Agent model as collaboration framework for learning resources.
2. To validate the model and provide implementation guidelines by developing prototype of the model into pre-existent e-learning systems.

To achieve these objectives, the methodology used in this work included:

- Conduct a literature review on existing research works, state of the art, and theories related to this work.
- Identify and describe all the actors that interact with eIA, followed by building the use-case model.

- Build eIA's semantic schema based on IEEE LOM (Learning Object Metadata) and IMS LIP (Learner Information Package).
- Analyze and design the eIA model.
- Implement the eIA model into Moodle platform.
- Evaluate and refine the eIA model based on Moodle's implementation result.
- Deploy the eIA model based on Moodle's implementation result into ATutor platform.
- Evaluate the eIA model based on the use-case and discuss further implementation of eIA to form Federated e-learning Community (FeC).

### 1.5 Thesis Organization

This thesis is divided into six chapters. This chapter introduces the background that comprises the reasons and ideas of conducting this research. It describes the approach conducted to solve the problem.

Chapter two elaborates a comprehensive background review and discussion of related research works. This chapter also provides a general overview of enabling technologies used to address the proposed model and implementation.

From the issues highlighted in chapter one and two, chapter three presents the e-learning Interoperability Agent model. The structure of eIA and algorithm needed is extensively explained.

Semantics integration employed by eIA and argument related is tackled in chapter four.

Chapter five discusses on implementation issues gained from prototyping experiences conducted during the research. A review of federated e-learning community that can be formed through eIA is also yielded.

Finally, chapter six draws the conclusions of the research and recommendation for future research.

## CHAPTER TWO : LITERATURE REVIEW

This chapter elaborates some selected works that relate to this work and presents the state of the art and methodologies used in designing eIA model: web services, semantics, standards in e-learning, mapping of data elements, and XML – XSLT.

### 2.1 Related Work

There have been several research works examining interoperability either in e-learning or other realms. Figure 2-1 describes a set of interoperability dimensions that need to be considered when integrating interoperability among heterogeneous services [Athanasopoulos et al., 2006]. As introduced in section 1.2, this work concerns interoperability on semantic level, where the problem of common understanding between service providers and service requestors is addressed, in a specific business domain: e-learning.

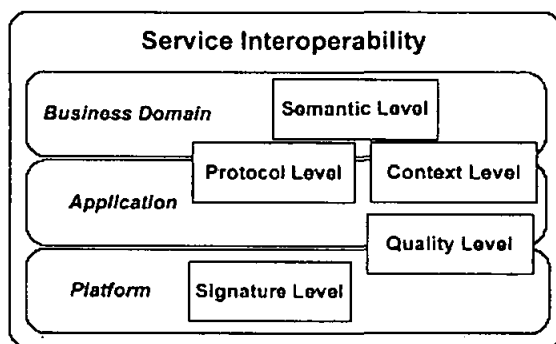


Figure 2-1: An Integrated View of Service Interoperability Dimensions [Athanasopoulos et al., 2006]

The idea and methodology of this work in addressing the interoperability are refined from some previous research works. The first one is the work from Liu et al. [Liu et al., 2003]. Their work proposes a functional model and service architecture for building standard-driven distributed and interoperable learning systems. The functional model provides a visual representation of the components that make up an e-learning environment and the objects that must be moved among these components. E-learning environment is divided into Learning Content Management System

(LCMS) and Learning Management System (LMS) to make each system's functionality more focused and clear. LCMS is a multi-user environment where learning developers can create, store, reuse, manage, and deliver digital learning content from a central object repository. Whereas LMS manages the processes surrounding learning (i.e. interchanges of user profile and user registration environment with other systems, the location of the course from LCMS and gets the learner action from LCMS). To integrate these LCMS and LMS successfully, they propose the service architecture in an e-learning environment. This architecture defines how different e-learning systems exchange messages through the interaction of web service agents in each system. Their work gives some concise ideas about e-learning environment and the emergence of web services to enable interoperability between e-learning systems.

From the emergence of web services to e-learning systems, Pankratius et al. present a distributed, service-oriented architecture for e-learning systems based on web services, and describe the extensions to support software agents for the distributed retrieval of educational content [Pankratius et al., 2004]. Their work addresses the problem of content retrieval in such a distributed environment by using intelligent software agents, which take both preferences of particular users, as well as data into consideration, which are stored inside services with assumption that the entire LCMS functionality including the learning content are implemented as web services. The learning content here is presented as learning objects (LOs), which basically represent reusable units of study, exercise, or practice and can be "consumed" in a single session. The agent platform is an environment, in which software agents can be executed to retrieve LOs, and which is wrapped by a web service. Agents are intended to assist learner with the focused search for LOs, according to the specifications they made. The search parameters of an agent, the start of a search, or the access to the list of retrieved LOs, for example, can be controlled by invoking appropriate web service operations which extract metadata from LOs.

Technically, the agent platform hosts several software agents, each of them having independent "intelligence", and which may move autonomously in form of "mobile code" to other agent platforms. Each software agent contains a list of UDDI (Universal Description, Discovery, and Integration) registries to find educational



content, and a list of other known agent platforms, which may belong to other LCMS. Agents can update their lists by communicating with other agents using a predefined communication protocol. By using a user-friendly interface, operations of the agent platform can be invoked to call an agent with search keywords as parameters. Afterwards, an agent uses a UDDI registry to locate a LO and call the get metadata operation which has to be implemented in every web-service-LO. Finally, the agent compares the metadata and the search keywords for possible matches and presents the search results to the user. Their work gives ideas to extend e-learning systems with software agents in order to retrieve learning contents (LOs).

More detail on LOs is discussed by Lee [Lee, 2005]. Lee's work models distributed and sharable learning resources by two types of LOs: Atomic Learning Object (ALO) and Composite Learning Object (CLO). Both types of LOs are uniformly published as web-services in a constraints-based web service broker. XML (eXtensible Markup Language)-based languages for modeling these two types of LOs are defined. The languages also serve as interchange formats for transferring and sharing LOs. The result of the work is an e-learning service infrastructure that facilitates the authoring, registration, discovery, storing, and invocation of LOs. The infrastructure is built on top of an extended web-service framework, which leverages the constraints-based web service broker and distributed LO repositories to make LOs searchable, sharable, and reusable. Ideas taken from Lee's work are the development of XML-based languages for modeling LOs and constraints-based web service broker. Each of the state of the art technologies carried out here is elaborated into more details in the next sections.

### **2.1.1 E-learning Interoperability**

Simon et al. work addresses challenges of making LCMS interoperable and present an architecture which has been implemented in the context of the Universal project [Simon et al., 2003]. The project has realized an exchange environment for learning resources, called EducaNext, which builds on a web-based tool called the Universal Brokerage Platform (UBP). The UBP provides services for covering critical issues such as the announcement, offering, distribution, and exchange of learning resources

via dispersed LCMS [Guth et al., 2001]. In Universal, the UBP acts as an educational broker. Like an electronic market place the platform provides facilities for the purpose of exchanging learning resources among individuals and organizations.

In comparison with this work, Simon et al. work proposes a centralized interoperability framework and focuses on exchanging of learning resources. Meanwhile, this work proposes a distributed interoperability framework and not only focuses on learning resources but also user profiles and the possibility to extend the services.

## 2.2 Web Services

Web services technology could be viewed as a platform for distributed computing over the web. The World Wide Web Consortium (W3C) defines web services as software applications identified by URI (Uniform Resource Identifier), whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols [W3C, 2002]. These web services are self-contained, modular applications that can be published, located, and invoked across the Internet. After a web service is deployed, other applications (and other web services) can discover and invoke the deployed service. The goal of web services includes universal operability, widespread adoption and ubiquitous accessibility of deployed services. It acts as an abstraction layer separating the platform and programming language specific details of how the application code is actually invoked.

The web services technology stack is built on a core set of XML based open standards: WSDL (Web Service Description Language), SOAP (Simple Object Access Protocol) as the messages, and UDDI (Universal Description, Discovery, and Integration). Through WSDL, web services providers publish their web services and the technical description of the web services. A WSDL document can be posted on the Internet, and its access point (URL – Uniform Resource Locator) together with textual description and the category information can be registered with a web-services

registry such as the UDDI. The content of the registry can then be navigated or searched manually or programmatically to discover and obtain the access information of suitable web services. The web-services allow the runtime binding of a service request to a remote web-service, and provide the SOAP that allows the activation of a remote service by exchanging SOAP messages. Figure 2-2 summarizes the web services overview. Meanwhile, a detailed explanation of each technology stack is provided consecutively in the next sub sections.

From the features of web services, it is feasible that web services technology can be adapted and extended in order to achieve interoperability across diverse e-learning platforms. Liu et al. outline three main reasons of web services feasibility for implementing e-learning systems interoperability [Liu et al., 2003]:

1. The standardization of e-learning, e.g. LOM and IMS content packaging, all have XML binding which is appropriate with web-services technology stack.
2. Web services architecture is platform and language independent. It can promote interoperability and extensibility among various e-learning applications, platforms and frameworks that have existed in the present e-learning market.
3. Web services provide a unified programming model for the development and usage of private Intranet as well as public Internet services. As a result, the choice of network technology can be made entirely transparent to the developer and user of the service.

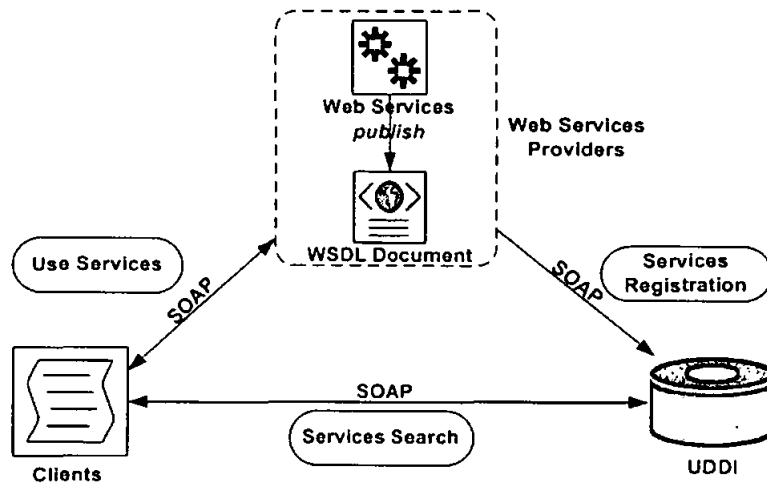


Figure 2-2: Overview of Web Services

### 2.2.1 Web Services Description Language (WSDL)

WSDL is an industry-agreed specification language defining how to describe web services in a common XML grammar. WSDL describes four critical pieces of data needed to describe a web service:

1. Interface information describing all publicly available functions;
2. Data type information for all the message requests and responses;
3. Binding information about the transport protocol to be used;
4. End-point addresses information for location the specified service.

The major elements of WSDL used to describe these four critical pieces of data are as follows:

- **Types** – a container for data type definitions that are made using XML Schema (XSD – XML Schema Definition) or another similar system for data types;
- **Message** – an abstract, typed definition of the data being communicated; it consists of the data types defined in “types” elements;
- **Operation** – an abstract description of an action supported by the service; input or output parameters are defined using the messages defined in “message” elements. There are four types of operations which can be defined in a WSDL document: one-way (the endpoint receives a message), request/response (the

endpoint receives a message, and sends a correlated message), notification (the endpoint sends a message), and solicit/response (the endpoint sends a message, and receives a correlated message);

- Port type – an abstract set of operations supported by one or more end-points;
- Binding – a concrete protocol and data format specification for a particular port type;
- Port – a single endpoint defined as a combination of a binding and a network address;
- Service – a collection of related end-points.

Figure 2-3 shows typical WSDL document structure.

```
<wSDL:definitions xmlns:wSDL="http://schemas.xmlsoap.org/wSDL"
  targetNamespace="your namespace here"
  xmlns:tns="your namespace here"
  xmlns:soapbind="http://schemas.xmlsoap.org/wSDL/soap">
  <wSDL:types>
    <xs:schema targetNamespace="your namespace here (could be another)"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      <!-- Define types and possibly elements here -->
    </xs:schema>
  </wSDL:types>
  <wSDL:message name="some operation input">
    <!-- part(s) here -->
  </wSDL:message>
  <wSDL:message name="some operation output">
    <!-- part(s) here -->
  </wSDL:message>
  <wSDL:portType name="your type name">
    <!-- define operations here in terms of their messages -->
  </wSDL:portType>
  <wSDL:binding name="your binding name" type="tns:port type name above">
    <!-- define style and transport in general and use per operation -->
  </wSDL:binding>
  <wSDL:service>
    <!-- define a port using the above binding and a URL -->
  </wSDL:service>
</wSDL:definitions>
```

Figure 2-3: Typical WSDL Document Structure

The definition to implementation mapping is independent from the language, platform, object model, or messaging system, as long as both the sender and the receiver agree on the service description itself. Conceptually, WSDL represents a contract between the service requester and service provider. With the help of WSDL, a user can locate a web service and write a client code to invoke any of its publicly available operations.

### 2.2.2 Simple Object Access Protocol (SOAP)

SOAP represents a cornerstone of the web service architecture, enabling diverse applications to easily exchange services and data. SOAP is fundamentally a stateless, one-way message exchange paradigm, but applications can create more complex interaction patterns, e.g. request/response. SOAP is silent on the semantics of any application-specific data it conveys. The messages can be carried by a variety of transport protocols, e.g. HTTP, SMTP, FTP, or proprietary transport protocols. A SOAP message consists of four parts as follows:

1. Envelope: defines a framework for describing the content of a message and the way to process it.
2. Header: contains header information. This part is optional.
3. Body: contains a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses.
4. Fault: provides information about errors that occur while processing the message. This part is optional.

Figure 2-4 shows the format of SOAP message with attachments. SOAP message can contain zero or more attachments. The attachment allows the SOAP message to contain not only the XML data but also non-XML data such as binary file.

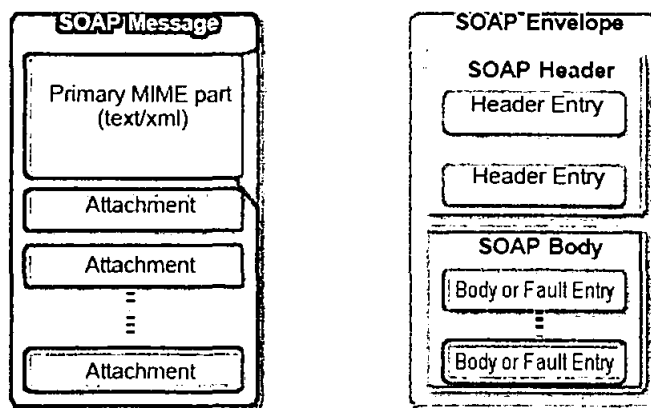


Figure 2-4: SOAP Message Format

### 2.2.3 Universal Description, Discovery, and Integration (UDDI)

UDDI is a platform-independent, XML-based registry, sponsored by Organization for the Advancement of Structured Information Standards (OASIS), enabling service providers to publish their service listings and discover each others and define how the services or software applications interact over the Internet. Mainly, UDDI is used for business purposes. There are three components which must be provided by a UDDI business registration:

- White Pages – address, contact, and known identifiers;
- Yellow Pages – industrial categorizations based on standard taxonomies;
- Green Pages – technical information about serviced exposed by the business.

UDDI is designed to be interrogated by SOAP messages and to provide access to WSDL documents.

## 2.3 Semantics

As introduced briefly in section 1.2, semantics refer to the aspects of meaning that are expressed in a language, code, or any other form of representation. Semantics have been a part of several scientific disciplines, both in the realm of computer science and outside of it. Application or research areas such as Information Integration, Information Retrieval (IR), Information Extraction (IE), Computational Linguistics (CL), Knowledge Representation (KR), Artificial Intelligence (AI), and Data(base) Management (DB) have all addressed issues pertaining to semantics in their own ways [Sheth et al., 2005]. This work is related to information integration and retrieval or extraction areas whereby the agents (eIA) communicate on a semantic basis. This section presents the background and overview of the semantic technique used by eIA.

### 2.3.1 Information Integration

There is, now more than ever, a growing need for several information systems to interoperate in a seamless manner. This sort of interoperation requires that the syntactic, structural and semantic heterogeneities [Hammer and McLeod, 1993]

[Kashyap and Sheth, 1996] between such information systems are resolved. Resolving such heterogeneities has been the focus of a lot of works in schema integration in the past. Table 2-1 presents some techniques used for schema integration with their related types of semantics, which are defined by Sheth et al. Sheth et al. organize different views of semantics into three forms: implicit, formal, and powerful (soft). Implicit semantics appear in unstructured text that has loosely defined and less formal structure, such as IR, IE and CL areas. Formal semantics appear when the data representation takes a more rigid form, well defined syntactic structures, such as KR, AI, and DB areas. Lastly, powerful semantics imply the combination of implicit and formal semantics (simple syntactic structures to represent the meaning of complex ones).

Table 2-1: Techniques used for Schema Integration [Sheth et al., 2005]

	Type of Information Used	Description	Types of Semantics
Linguistic Techniques	Name Similarity	Using canonical name representations, synonymy, hypernymy, string edit distance, pronunciation and N-gram like techniques to match schemas attribute and relation names.	<i>Implicit semantics</i> are exploited by string edit distance, pronunciation and N-gram like techniques. <i>Formal semantics</i> are exploited by synonymy, etc.
	Description Similarity	Processing natural language descriptions associated with attributes and relations.	<i>Implicit semantics</i> are exploited by the NLP techniques deployed.
	Word Frequencies of Key terms	Using relative frequencies of keywords and word combinations at the instance level.	<i>Implicit semantics</i>
Constraint Based Techniques	Type Similarity	Using information about data types of attributes as an indicator of a match between schemas.	<i>Formal semantics</i>
	Key Properties	Using foreign keys, part-of	<i>Formal semantics</i>



	Type of Information Used	Description	Types of Semantics
		relationships and other constraints.	
	Graph Matching	Treating the structure of schemas as graphs algorithms to determine match degree between graphs are used to match schemas.	Combination of <i>Implicit</i> and <i>Formal semantics</i>
	Value Patterns and Ranges	Using ranges of attributes and patterns in the value of attributes as an indicator of similarity between the corresponding schemas.	<i>Implicit semantics</i>

Dealing with data used in VLE systems, which is designed in structured manner, requires the use of techniques listed under constraint based techniques. This work uses a semantic schema integration technique which defines the standardized eIA data field naming convention. The technique used is similar to type similarity technique. Instead of using data types of attributes as an indicator of a match between schemas, it uses data field naming convention as the indicator.

### 2.3.2 Information Retrieval and Information Extraction

Given a request for information by user, information retrieval applications have the task of answering user's "query" by either searching for information in documents, searching for documents themselves, searching for metadata which describe documents, or searching within databases, whether relational stand-alone databases or hyper-textually networked databases such as the world wide web. There are various flavors of such applications: search engines like Google and question answering systems.

Not only retrieving information, one type of more specific fields in information retrieval is information extraction, whose goal is to automatically extract

structured information, i.e. categorized and contextually and semantically well-defined from a certain domain, from unstructured machine readable documents. The significance of information extraction is determined by the growing amount of information available in unstructured form, for instance on the Internet. This knowledge can be made more accessible by means of transformation into relational form, or by marking-up with XML tags.

As introduced in section 1.3, eIA provides two basic functionalities related to learning objects and user profiles. Both functionalities enable users to retrieve and extract either LOs or user profiles data from various educational institutions through their e-learning systems which “connected” between each others with eIA.

### 2.3.3 Semantic Integration

There are some steps that need to be done in order to achieve semantic integration: mappings generation, verifying correctness of mappings, and use mappings to translate among ontologies. There is also a variety of architectures for the semantic integration identified by Uschold and Gruninger [Uschold and Gruninger, 2005]. The architecture differences depend on the following dimension of variation: origins of the semantic mappings, and the nature and degree of the agreements that exists among the anticipated community of interacting agents. Table 2-2 summarizes characteristics of some semantic integration architectures.

**Table 2-2: Various Semantic Integration Architectures Characteristics**  
[Uschold and Gruninger, 2005]

Questions Architecture	Who generates the mappings?	When define agent to agent mapping?	Topology	Degree of Agreement
Global Ontology	No mappings	No mappings	Point-to-point	Agree on everything
Manual Mapping	Agent designers	Before agents interact	Point-to-point	No a priori agreement
Interlingua Ontologies	Agent designers	Auto-generated at agent	Mediated	Agree on interlingua

Questions	Who generates the mappings?	When define agent to agent mapping?	Topology	Degree of Agreement
Architecture		interaction time		ontologies
Community Ontology Mappings	Ontology designers	Auto-generated at agent interaction time	Mediated	Agree on alignment mappings
Ontology Negotiation	Agents themselves	Auto-generated at agent interaction time	Point-to-point	No a priori agreement

eIA's semantic integration built based on the interlingua ontologies architecture. In the interlingua ontologies architecture, each agent designer generates a mapping from their agent's ontology to a standard interchange ontology, or interlingua [Ciocoiu et al., 2001]. This is done before the agents interact. The agent to agent semantic mappings are generated dynamically at agent-interaction time by executing pre-specified mappings to and from the interlingua. In this case, the interlingua ontology mediates the mapping between the agent ontologies. The agents that want to participate in this architecture must agree a priori to use the interlingua ontology. This is a partially automated version of ontology negotiation.

## 2.4 Standards in E-Learning

In achieving the semantic integration, this research also needs to review some of the available e-learning standards developed by some established institutions as base to tailor eIA's standardized data structures. E-learning standards can be grouped into five general categories [Collier and Robson, 2002]:

### 1. Metadata

Learning content and catalog offerings must be labeled in a consistent way to support the indexing, storage, discovery or search, and retrieval of learning objects by multiple tools across multiple repositories. Data used for this purpose is referred to as learning object metadata.

## 2. Content Packaging

Content packaging specifications and standards allow courses to be transported from one learning system to another. This is crucial since learning content can potentially be created by one tool, modified by another tool, stored in a repository maintained by one vendor, and used in a delivery environment produced by a different vendor. Content packages include both learning objects and information about how they are to be put together to form larger learning units. They can also specify the rules for delivering content to a learner.

## 3. Learner Profile

Learner profile standards allow different system components to share information about learners across multiple system components. Learner profile information can include personal data, learning plans, learning history, accessibility requirements, certifications and degrees, assessments of knowledge (skill or competencies), and the status of participation in current learning.

## 4. Learner Registration

Learner registration information allows learning delivery and administration components to know what offerings should be made available to a learner, and provides information about learning participants to the delivery environment.

## 5. Content Communication

When content is launched, there is the need to communicate learner data and previous activity information to the content. As a learner interacts with content, he generates some type of activity results, score or course grade. Sharing the launch, status of learning activities and result across multiple components of a learning environment requires standardization.

eIA currently focuses on two categories: metadata and learner profile. Both categories are taken into account in designing the eIA semantic integration schema to fulfill the basic functionalities (learning objects and user profiles).

### 2.4.1 Metadata

Every LO is composed by two parts: the content and the label (or metadata). The metadata describes the content of the LO so that the LO can be found or searched and specified how it can be used. There are several initiatives in creating metadata standards to classify and characterize an LO. This work refers to IEEE LOM standards [IEEE LTSC, 2002] which drawn from earlier work by the IMS [IMS LOM, 2002], ARIADNE [ARIADNE, 2001], and Dublin Core Groups [DCMI, 1999]. Figure 2-5 shows the element and structure of the LOM conceptual data schema. The LOM element is the root of the XML document describing a learning object. There are nine categories of LOM elements, each possibly containing other sub-elements:

1. General: describes general information of the LO, i.e. title, catalogue entry, language, description, keywords, coverage, structure, and aggregation level.
2. Life Cycle: describes features regarding the history and the current state of both the LO under consideration and those ones affecting it during its evolution, i.e. version number, contributor of the LO, and current status of the LO.
3. Meta-Metadata: groups information about the metadata instance rather than about the LO it describes, i.e. record identifier, catalog entry, contributor of the record, format of the record, and language of the record.
4. Technical: specifies the technical requirements and characteristics of the LO, i.e. format of the LO, size of objects in bytes, location of object, system required for delivery, installation remarks, special requirement, and duration of media resource.
5. Educational: describes the educational properties, i.e. type of interactivity, type of resource, level of interactivity, semantic density, intended for use by-in, age or experience of intended user, level of difficulty, typical time required to complete, description (how you can use the resource), and language of intended user, concerning the study of the LO.

6. Rights: states the intellectual property rights, i.e. costs (payment required), copyright (subject), and description (statement of copyright and restrictions).
7. Relation: describes the features of those resources related to the LO in some way, i.e. type of relationship and information of related resource.
8. Annotation: specifies comments about the educational usage of the LO, i.e. person who gave comment, date of comment, and description (the comment itself).
9. Classification: represents characteristics of the LO by means of a series of classification entries, i.e. person who classified, locations in library, description in context, and keywords in context.

General, life cycle, meta-metadata, technical, educational, and rights elements can occur at most once within the LOM element. Relation, annotation, and classification elements can occur zero or more times within the LOM element.

LOM specification is not to be followed strictly. There is no ultimate or perfect metadata description of any LO. Each LO will have multiple overlapping partial descriptions, created by different communities depending on their needs or for different uses of the LO. IEEE LOM specification is only one of the ways to integrate the differences. The previous discussion shows the LOM specification is particularly rich and complex to satisfy the requirements of different types of learners. In addition, the specification covers a broad audience in that it can support “academic” users, as well as learners operating in companies or government offices. As a consequence, the specification might be too cumbersome and complicated in real contexts.

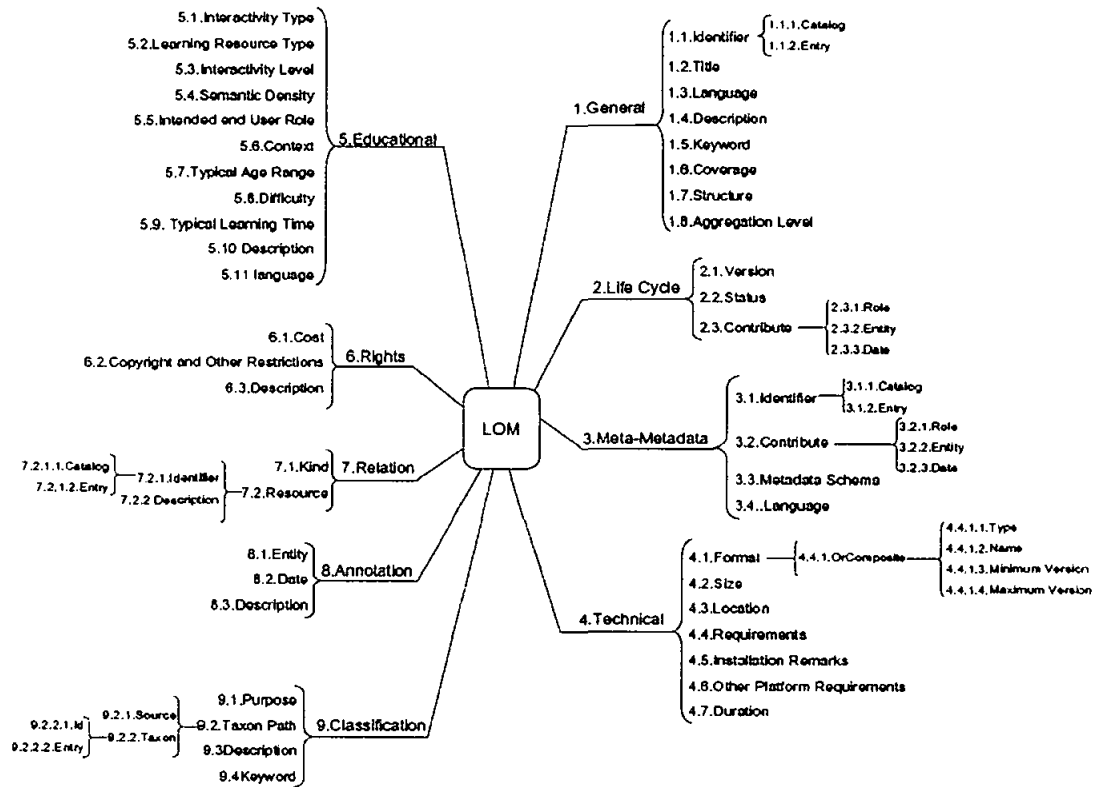


Figure 2-5: LOM Metadata Schema

This work tailors the data structure to describe LOM by taking the essential elements of IEEE LOM. The structure and the reason behind of the chosen elements are presented in Chapter Four.

### 2.4.2 Learner Profile

There are some available standards related to this category: Personal and Private Information (PAPI) by IEEE [IEEE PAPI, 2002] and Learner Information Packages (LIP) by IMS [IMS LIP, 2003]. In PAPI, the most important information in a learner profile is the learner performance. As a consequence, this standard appears particularly suited for intelligent tutoring systems. Meanwhile, LIP stores information to describe complete profile of the learner. Hence, LIP is more suitable to be used in this work.

There are eleven cores of IMS LIP classes:

1. Identification: contains all of the data for a specific individual or organization. This includes data such as: names, addresses, contact information, demographics, and agent.
2. Accessibility: consists of the cognitive, technical, and physical preferences for the learner, their language capabilities, disability, and eligibilities.
3. Goal: consists of the description of the personal objectives and aspirations. These descriptions may also include information for monitoring the progress in achieving those goals. A goal can be defined in terms of sub-goals.
4. QCL: consists of the qualifications, certifications, and licenses awarded to the learner, i.e. the formally recognized products of their learning and work history. This includes information on the awarding body and may also include electronic copies of the actual documents.
5. Activity: consists of the education or training, work and service (military, community, voluntary, etc.) record and products (excluding formal awards). This information may include the descriptions of the courses undertaken and the records of the corresponding evaluation.
6. Transcript: comprises the summary record of the academic performance of an individual with respect to a particular institution. The transcript is normally supplied by the body responsible for evaluating the performance of the individuals.
7. Competency: consists of the descriptions of the skills the learner has acquired. These skills may be associated with some formal or informal training or work history (described in the 'activity') and formal awards (described in 'QCL'). The corresponding level of competency may also be defined.
8. Interest: consists of descriptions of hobbies and other recreational activities. These interests may have formal awards (as described in the associated



'QCL'). Electronic versions of the products of these interests may also be contained.

9. **Affiliation:** is used to store the descriptions of the affiliations associated with the learner, e.g. professional affiliations. A learner's membership of the relevant class, cohorts, groups, etc. undertaken when being educated, trained, etc.
10. **Security Key:** is used to store the descriptions of the passwords, certificates, PINs and authentication keys. These keys are used for transactions with the learner.
11. **Relationship:** is the container for the definition of the relations between the other core data structures, e.g. qcl's and the awarding organization. This enables the construction of complex relationship between the core data structures.

This work tailors the user profiles data structure to describe user profiles by taking the essential elements. There also some new elements added to enrich the profiles. User profiles data structure is presented in Chapter Four.

## **2.5 Mapping of Data Elements**

The semantic integration schema needs to be mapped into e-learning database system. Different e-learning system platforms have different database architecture. Thus, they have different metadata profiles as well as associated value spaces and data types.

Najjar et al. define actions in metadata mapping process as follows [Najjar et al., 2003]:

1. **Mapping of data elements:** mapping of profile elements into their equivalent elements in the standard. There are two types of this mapping:
  - a. **Mapping of independent data elements:**

- 1-to-1 Mapping: Data elements of a profile have the same characteristics as its interrelated data element in the standard. Therefore, each data element is mapped directly into exactly one corresponding data element in the standard.
  - 1-to-N Mapping: One data element of the profile maps into more than one data element in the standard schema.
  - N-to-1 Mapping: Some data elements of the profile map into one data element in the standard schema.
- b. Mapping sets of dependent data elements: Data elements have a dependency relationship with other elements.
2. Mapping values of data elements:
- a. Mapping vocabulary values from a profile value space into values of the standard: 1-to-1, 1-toN, N-to-1, and N-to-Null. One addition type of this mapping is N-to-Null which is one vocabulary value or more don't have any equivalent value among values of the standard.
  - b. Mapping values from one data type into another data type identified by the standard. For example, a group of personal contact information data, such name (type: string), telephone numbers (type: string), and address (type: string) is mapped into one single data (type: vCard).

XML – XSLT is the chosen technology to do the mapping process between the integration schema and particular structure.

## 2.6 eXtensible Markup Language

The XML has been accepted as a universal format for data interchange and publication on the Internet [Abiteboul et al., 2000]. The use of XML in e-learning standards and web services are just some of the examples. XML represents powerful way to overcome semantic barriers to information exchange. One of the case studies is applying into applications in which the data of database needs to be viewed in XML format so that the data being viewed takes richer semantics and allow more flexibility in syntax. Liu and Vincent define two types of these applications [Liu and Vincent,

2003]. In the first, initial data is defined in XML format and is then mapped and stored in a database to achieve better access efficiency. In the other way, the data in database needs to be wrapped in XML format for interchange or publication purpose. In both situations, the user who views the data through XML glass [Sahuguet and Azavant, 1999] will see only XML data, not the database. In the context of this work, one issue of utmost importance is to provide techniques and tools for converting XML data used by eIA to data used by particular e-learning systems. This work uses XSLT (eXtensible Stylesheet Language Transformation) as the chosen technique and tool.

### 2.6.1 eXtensible Stylesheet Language Transformation (XSLT)

XSLT is a W3C standard for transforming XML documents into either other XML documents or regular text documents. As XSLT is invented with the idea to be used for transforming XML documents, it is a suitable solution in terms of time needed to develop transformation, regardless how different their schemas are. Moreover, XSLT is independent of any programming language and can be executed by a program written in almost any up-to-date programming language, e.g. Xalan Processor for Java and Sablotron Processor for PHP. Figure 2-6 illustrates the main principle of document transformation using XSLT. To transform a document, one must provide the XSLT stylesheet modules to be used by the XSLT processor to process the transformation.

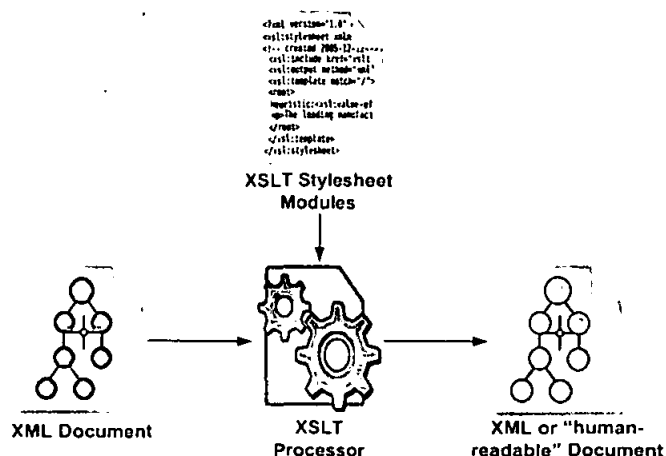


Figure 2-6: Overview of XSLT

An XSLT stylesheet module is an XML document that describes a collection of template rules. Figure 2-7 depicts an example of minimal but complete XSLT stylesheet module. The rules guide the processor in matching certain elements of the input XML document and transform them into the desired output document. XSLT rules are categorized into four types, each of which corresponds to one of the basic editing operations [Ono et al., 2002]:

1. Insert

An insert operation is converted to one of three types of insert rules, depending on the location of the insertion: insert-before, insert-after, and insert-child.

2. Remove

In contrast to the other types of rules, a remove rule cannot be created solely by reference to the result tree. Remove rules are created from a sequence of node-to-node mappings, which ends with a mapping with a null result node.

3. Modify

Modify rules deal with either text content or node attributes. There are four types of modify rules: one is for changing text content, and the other three are for changing attribute values of a source node (modify, remove and add attribute).

4. Copy

A copy operation creates two XSLT rules: one for replicating a subtree, and the other for inserting the replicated subtree.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
</xsl:stylesheet>
```

Figure 2-7: Minimal but Complete XSLT Stylesheet

There are four fundamental elements needed in applying the rules:

1. <xsl:output>

This element defines the format of the output document. The allowed methods are xml, html, text, and name.

## 2. &lt;xsl:template&gt;

An XSL stylesheet consists of a set of templates. These templates contain rules that are applied when a specific matching node is found. Template element structure is quite simple. At the most basic level they simply replace the element they match with whatever is in template. Refer to example given in Table 2-3, when <xsl:template match="users/user"> is matched, the XML source document node <users><user> turns into XML result document node <name>.

## 3. &lt;xsl:apply-templates&gt;

This element instructs the XSL processor to apply matching templates to the current element, or the current element's child nodes.

## 4. &lt;xsl:value-of&gt;

This element extracts the value of a selected node.

There are a few other common and useful elements, such as <xsl:if> and <xsl:choose> which allows testing of conditions before applying rules, <xsl:for-each> which allows looping through elements, and <xsl:sort> which allows sorting of output. Table 2-3 shows a simple example of XSLT transformation.

Table 2-3: XSLT Transformation Code Example

XML Source Document	XSLT Stylesheet Module	Result Document
<pre>&lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;users&gt;   &lt;user username="user1"&gt;   &lt;firstname&gt; one&lt;/firstname&gt;   &lt;lastname&gt; lastone&lt;/lastname&gt; &lt;/user&gt;   &lt;user username="user2"&gt;   &lt;firstname&gt;</pre>	<pre>&lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/ 1999/XSL/Transform" xmlns:fo="http://www.w3.org/ 1999/XSL/Format"&gt; &lt;xsl:output method="xml" indent="yes"/&gt; &lt;xsl:template match="/"&gt;   &lt;data&gt;     &lt;xsl:apply-templates/&gt;   &lt;/data&gt;</pre>	<pre>&lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;data xmlns:fo="http://www.w3.org/ 1999/XSL/Format"&gt;   &lt;name username="user1"&gt; one &lt;/name&gt;   &lt;name username="user2"&gt; two &lt;/name&gt; &lt;/data&gt;</pre>

XML Source Document	XSLT Stylesheet Module	Result Document
<pre>two&lt;/firstname&gt; &lt;lastname&gt; lasttwo&lt;/lastname&gt; &lt;/user&gt; &lt;/users&gt;</pre>	<pre>&lt;/xsl:template&gt; &lt;xsl:template match="users/user"&gt;   &lt;name username="{@username}"&gt;     &lt;xsl:value-of select="firstname"/&gt;   &lt;/name&gt; &lt;/xsl:template&gt; &lt;/xsl:stylesheet&gt;</pre>	

## 2.7 Summary

The purpose of this chapter is to provide the reader with sufficient background information to understand the foundations and concepts elaborated in the rest of this thesis. Section 2.1 discusses some selected works related to this work. The major difference between this work and theirs is that this work designs an interoperability tool model as extension to pre-existent system, not proposing a new e-learning system. The rest of the sections discuss the approach used in designing the model: web services, semantics, standards in e-learning, mapping of data elements and XML-XSLT.

## CHAPTER THREE : E-LEARNING INTEROPERABILITY AGENT

This chapter carries out a detail description of the e-learning interoperability agent model and its architecture. Firstly, an overall structure and users of eIA are described. Then consecutively, each layer of the structure is elaborated.

### 3.1 Overview of eIA

Figure 3-1 gives an architectural diagram of the e-learning Interoperability Agent (eIA) and shows how eIA attaches and interacts with VLE system. The eIA is deployed as an extension or plug-in into the pre-existent VLE system. eIA is built up by three layers: web services, data mediation module, and query GUI module. Layered structure is used to provide different abstraction views of the structure and hide complexities of specific implementation.

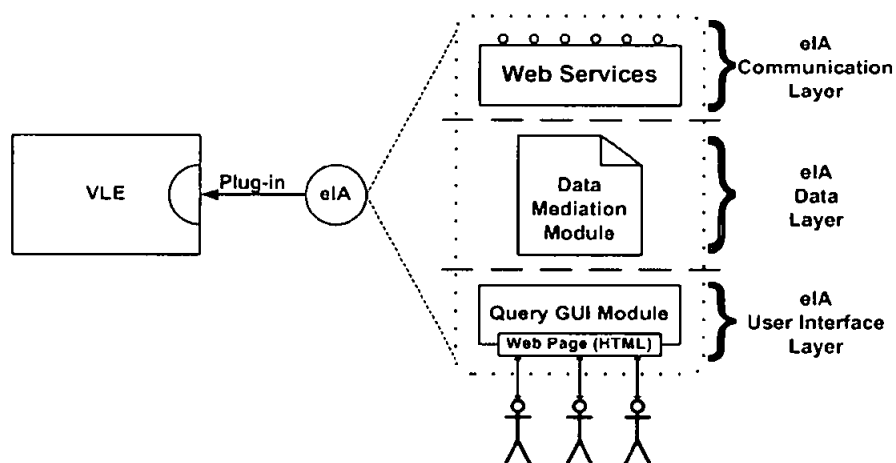


Figure 3-1: e-learning Interoperability Agent Structure

#### 3.1.1 Users of eIA

In a typical e-learning environment, there are several groups of people or users involved: authors or teachers and learners or students, who are the main players, and administrators. eIA also categorizes users into three roles: administrator, teacher, and

student. Both teacher and student also can be combined into learner roles. As discussed in section 1.1, the only difference between teacher and student in the aspect of learning object is that teacher responsible to create formal content and student may create informal content. There are three use cases taken into consideration regarding roles and basic functionalities of eIA:

#### 1. eIA Administration

One ultimate goal of eIA is as one alternative to establish educational collaboration. There is one vital process that needs to be done after implementing eIA in the e-learning system. The process is creating partnership with other educational institution, called as the “registration” process. Administrator needs to register their eIA first with other institution before being able to use the services offer by the eIA community and participate in the community. The registration process itself depends on each institution policy. The process is completed after the corresponding administrators update their service directory (section 3.2).

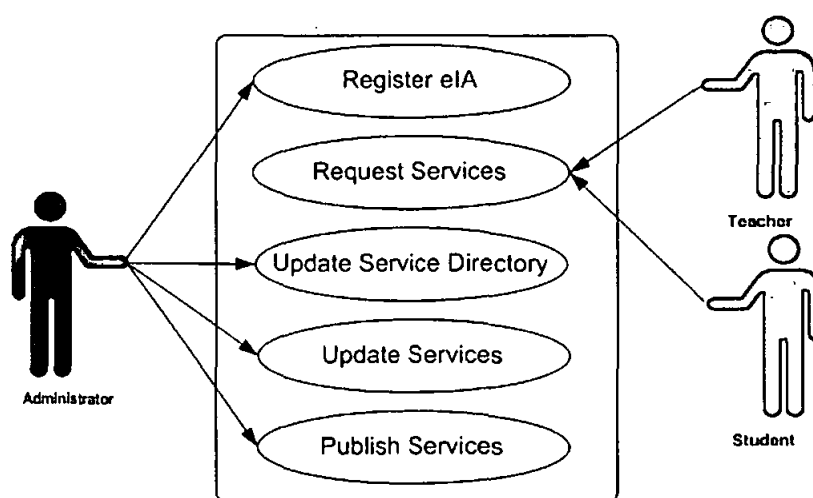


Figure 3-2: eIA Administration

After the registration, the users, teachers and students, can start using eIA. Users are also able to request new service registration if they found other institutions which have not registered with their institution. Administrators have the responsibility to deal with this request. Beside that, administrators



also have the responsibility to update any changes of services provided (by means of WSDL). Figure 3-2 depicts the eIA administration use case diagram.

## 2. Learning Objects Services

Both teachers and students can search, view, and retrieve LO(s). Teachers are responsible to publish formal LO(s) and students may publish informal LO(s). There is also one optional service: rate LO(s). This service availability depends on capability of each VLE system. If the system provides it, then users can utilize it. Figure 3-3 depicts the learning objects use case diagram.

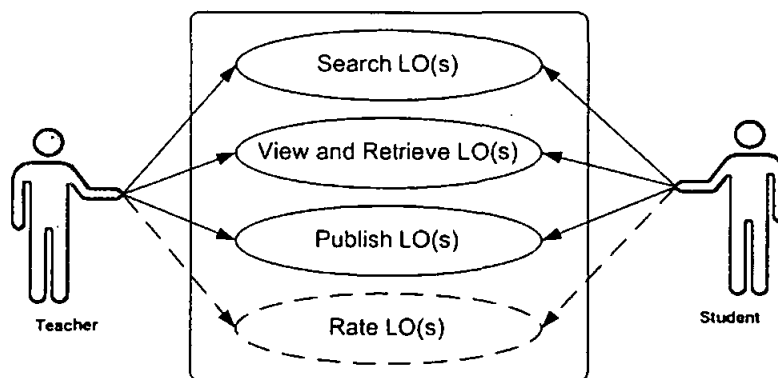


Figure 3-3: Learning Objects Use Case

## 3. User Profiles Services

Both teachers and students can search for user(s), view their profiles, and initiate inter-institutional communication through messages. There are two types of messages: private and public messages. Private messages are messages that only can be seen between two corresponding users. Meanwhile, public messages are messages that can be seen by users who view their profiles. Public messages have more or less similarity with forum-like or comment hosted by each user. All messages which appear on their profile are subjected to the host approval. Figure 3-4 depicts the user profiles use case diagram.

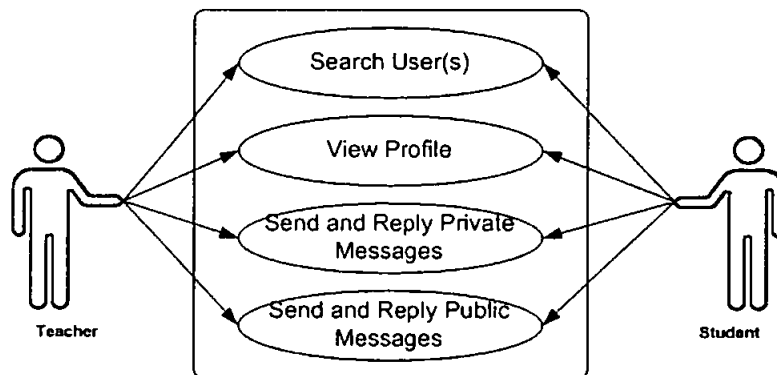


Figure 3-4: User Profiles Use Case

### 3.2 eIA Communication Layer

The first eIA layer is communication layer, which is performed by web services component. Web services enable flexible connectivity of applications or resources by representing every application or resource as service with a standardized interface enabling them to exchange structured information (messages, documentation, learning objects) and mediate the message exchange [Ma et al., 2005]. This flexibility enables new and existing applications to be easily and quickly combined to address interoperability, in this case e-learning interoperability through eIA. eIA adapts web services technology into three sub components:

1. eIA Service Gateway

eIA Service Gateway (eSG) is a run-time component that provides configurable mapping based on WSDL document. The interface, binding, and service endpoint of a learning service defined in WSDL are mapped into the gateway. Through the gateway, eIA communicate with other eIA by exchanging SOAP messages.

2. eIA Service Directory

eIA Service Directory (eSD) works like UDDI registry. It is a database for storing information of registered distributed eIA services. Table 3-1 shows the conceptual schema of eSD. Similar to UDDI, eIA can use the information provided in an eSD registry to perform three types of searches:

- A white pages search returns basic information such as identifiers about educational institutions and its provided services.
- A yellow pages topical search retrieves information according to categorizations, such as locations of the institutions.
- A green pages service search retrieves technical information about eIA services, as well as information describing how to execute these services.

Table 3-1: eIA Service Directory

	name	[mandatory] eIA ID or name of one particular VLE.
White Pages	description	[mandatory] Description of the particular educational institution which hosts the VLE.
	overviewURL	[mandatory] Overview URL (web page) of the educational institution.
	contacts	[optional] Contacts information regarding eIA registration.
Yellow Pages	location	[optional] Geographical location of the educational institution.
Green Pages	accessPoint	[mandatory] Access point or gateway of its eIA services server.
	verifiedIP	[mandatory] Verified IP of VLE which has been recognized in registration and can be used to access eIA server (the host of this eSD).
	authorizationCode	[mandatory] Authorization code to access the particular eIA services server.

Besides as UDDI, eSD also works as the security gateway database among eIAs (Figure 3-5). eSD stores authorization code between registered eIAs. The code is used to authenticate connection request, so that eIA must provide it to initiate intercommunicate with other eIA. If the code is valid, the communication session will be created.

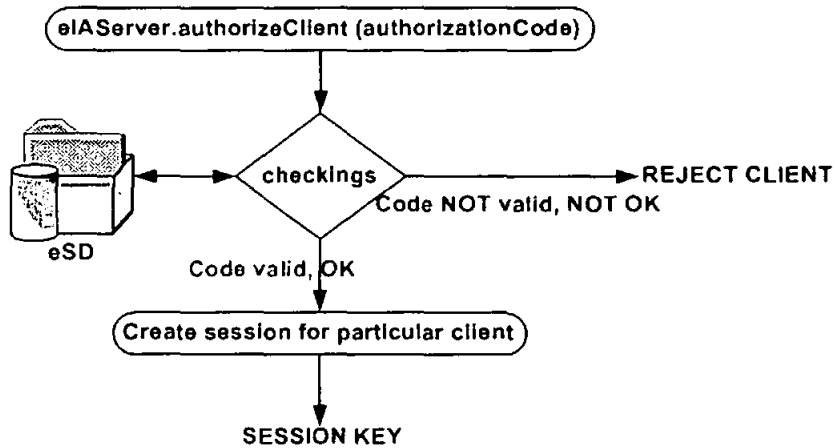


Figure 3-5: eIAClient Authorization

### 3. eIA Service Core

eIA Service Core (eSC) is the core library of web services component. It processes the messages passed up by eSD. The processed messages invoke the appropriate service functions. Then, service functions communicate with the data mediation module to get the desired result from the particular VLE system. Finally, the result given by data mediation module is wrapped up and passed up to eSG to be sent back to the client.

Figure 3-6 draws how eIA web services sub components interact with each other. There are six types of interaction:

#### 1. publish

"Publish" interaction is a relation between eSC and WSDL. All of the provided services exposed at WSDL so that eIA community is informed and can utilize the services.

#### 2. reside

"Reside" interaction is a relation between WSDL and eSG. WSDL document resides inside the eSG and the community accesses it through eSG.

#### 3. send and receive

"Send and receive" interaction is a relation between particular eIA, especially eSG, and the community. eIA communicates with the community by exchanging SOAP messages.

## 4. pass messages

“Pass messages” interaction is a relation between eSG and eSC. The used messages’ format here follows the eIA standardized data schema. The schema itself is presented in Chapter Four.

## 5. communicate

“Communicate” interaction is a relation between web services component, especially eSC, and data mediation module. This interaction is to process the requested service and retrieve the feedback from VLE system.

## 6. manipulate

“Manipulate” interaction is a relation between eSC and eSD. This interaction is needed to retrieve list of registered eIA services and their authorization code as explained in eSD sub component before.

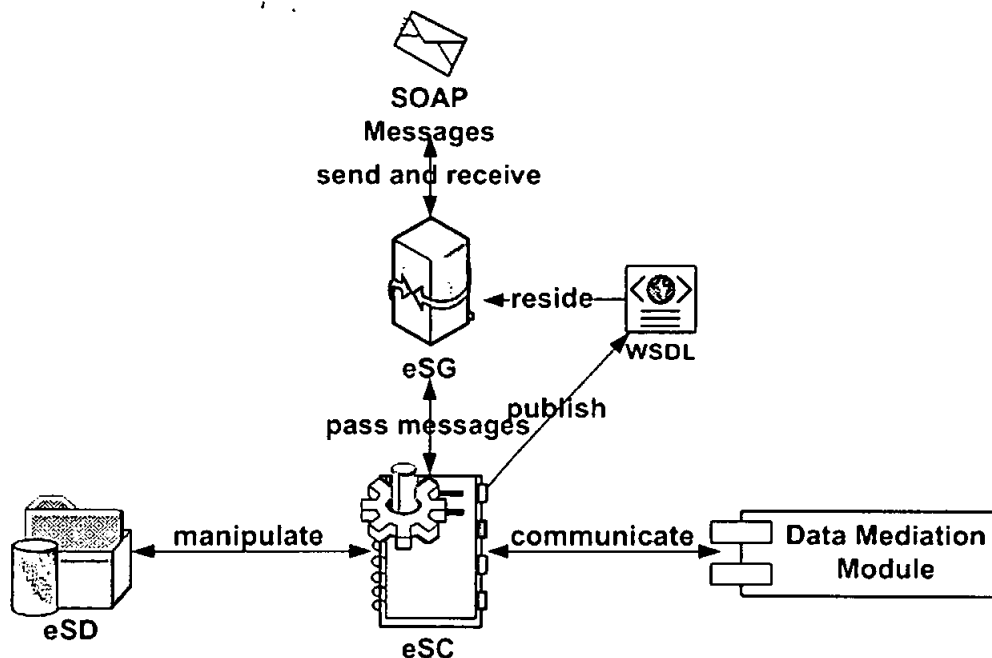


Figure 3-6: eIA Web Services

### 3.3 eIA Data Layer

To create a loosely-coupled connection between eIA and pre-existent e-learning systems, data mediation module is proposed as an interface to bridge database architecture differences. The interface allows eIA to access or query the database of VLE system transparently, ignoring the underlying database structure model. There are three sub components of this module:

1. eIA Service Functions

eIA Service Functions (eSF) is a mediator between the web services layer, especially eSC, and this layer. eSF consists of implementation of service functions published in WSDL. eSF passes up the data to the XSLT module.

2. eIA XSLT Module

This module has a role of a bridge between eIA standardized data and VLE database structure. eIA standardized data is the semantic integration to overcome interoperability barrier between various VLE systems, which is semantic heterogeneity. The semantic heterogeneity itself is in form of different architectures and databases, which may ascribe disparate meanings to the same terms or use distinct terms to convey the same meaning. Chapter Four elaborates more on this semantics matter.

This work use XML as the wrapper for interchange purpose and specifically XSLT as the wrapper technology. XSLT enable the proposed semantic integration to be deployed into pre-existent VLE systems as converter either from eIA standardized data into particular VLE database schema or vice versa. Thus, one needs to adjust the XSLT stylesheet module according to the related VLE database schema.

3. eIA Database Functions

eIA Database Functions (eDF) query the particular VLE database, retrieve the results, and pass up the results back to XSLT module to be transformed into eIA standardized data. The eDF itself is invoked dynamically by XSLT

module. Figure 3-7 shows the mapping process from eIA service to specific database function.

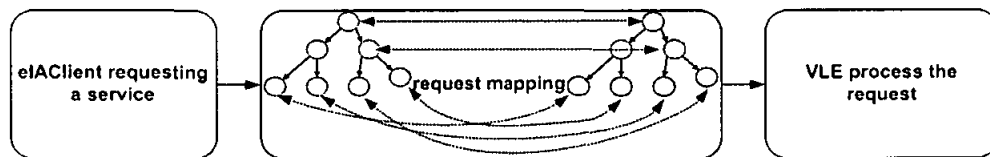


Figure 3-7: eIA Service Mapping

Figure 3-8 shows the interaction of each data mediation module sub component. The flow starts at eSC by passing up the request data and also ends at eSC by passing up the response data.

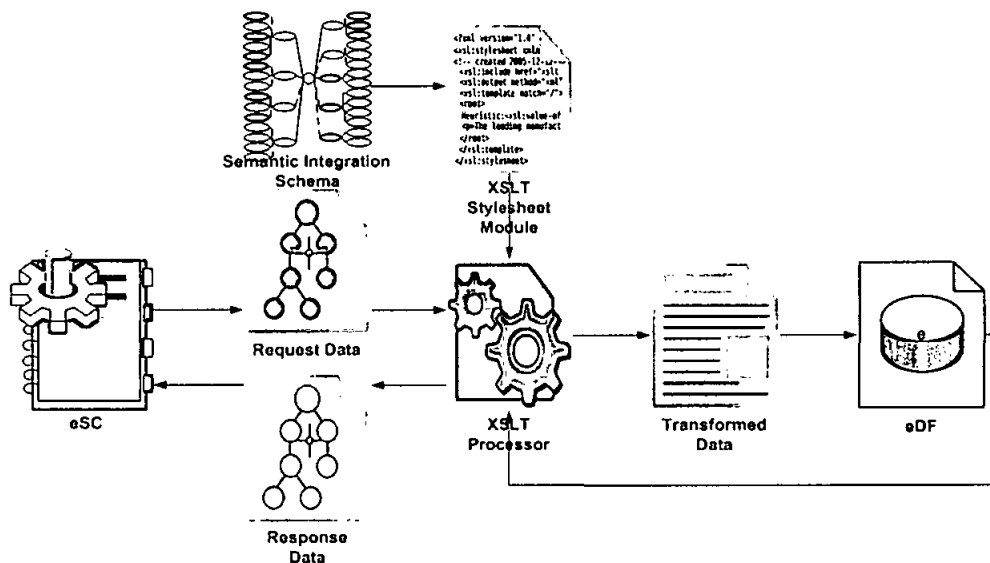


Figure 3-8: eIA Data Mediation Module

### 3.4 eIA Query GUI Module

Query GUI module works as an interface between users of VLE system and eIA. The module is embedded into user interface of the pre-existent VLE system and controlled by the VLE system. This module generates HTML pages or forms dynamically for interaction between users and eIA. UI processor processes the user requests by

cooperating with either eSC or eDF and gives the feedback through the interface. Figure 3-9 depicts the flow of the module.

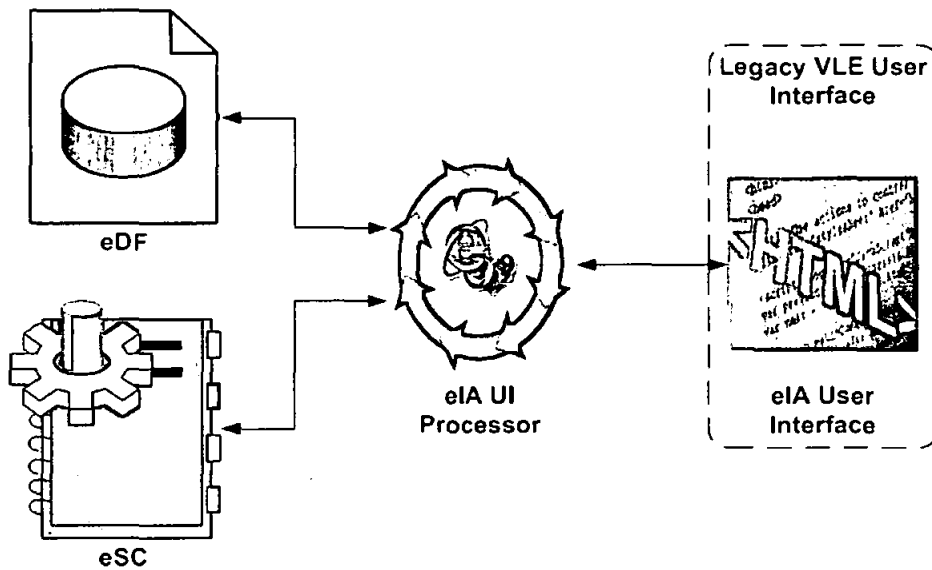


Figure 3-9: eIA Query GUI Module

Figure 3-10 shows the menu structure of eIA. There are two menus for basic functions invocation and one menu for dynamic functions invocation. Basic functions invocation serves learning object and user profiles services. Meanwhile dynamic functions invocation serves other extended services if provided by each of the registered eIA server, e.g. forums.

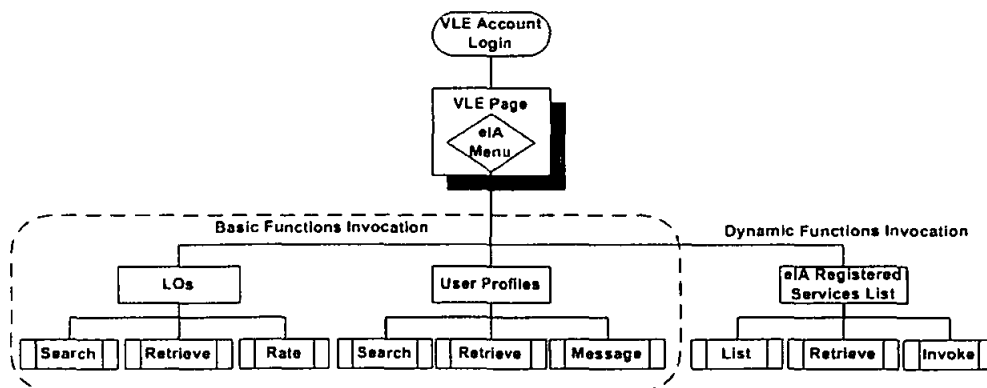


Figure 3-10: eIA Menu Structure



### 3.5 eIA Flow State Diagram

To summarize the whole eIA components described in previous sections, Figure 3-11 depicts a big picture of the basic functionalities use of eIA and the interaction between each of the component.

The client side flows are as follows:

1. eIA user interface is shown (0).
2. Users choose one of the available menus: list of eIA services (1.1), learning objects (1.2), and user profiles (1.3). If users choose list of eIA services, then the interface shows available services of chosen eIA (1.1 → 1.1.1).
3. The chosen service is sent to the web services layer to be invoked (1.1.1 or 1.2 or 1.3 → 2.1).
4. Web services layer sends the request to eIA server (2.1 → 3.1).
5. Web services layer receives the response from eIA server (3.3 → 2.2).
6. Web services layer passes up the response to the interface (2.2 → 0).

The server side flows are as follows:

1. Web services layer of eIA server receives the request (3.1).
2. Web services layer invokes the appropriate service function (3.1 → 4).
3. Data mediation module processes the request (4).
4. If the request needs to be passed up to the community, then eIA server forwards the request to other eIA server (4 → 3.2 → 3.1). "Forward request" process is explained in Chapter Five.
5. Data mediation module passes up the results to web services layer (4 → 3.3).
6. Web services layer sends the response to the client (3.3 → 2.2).

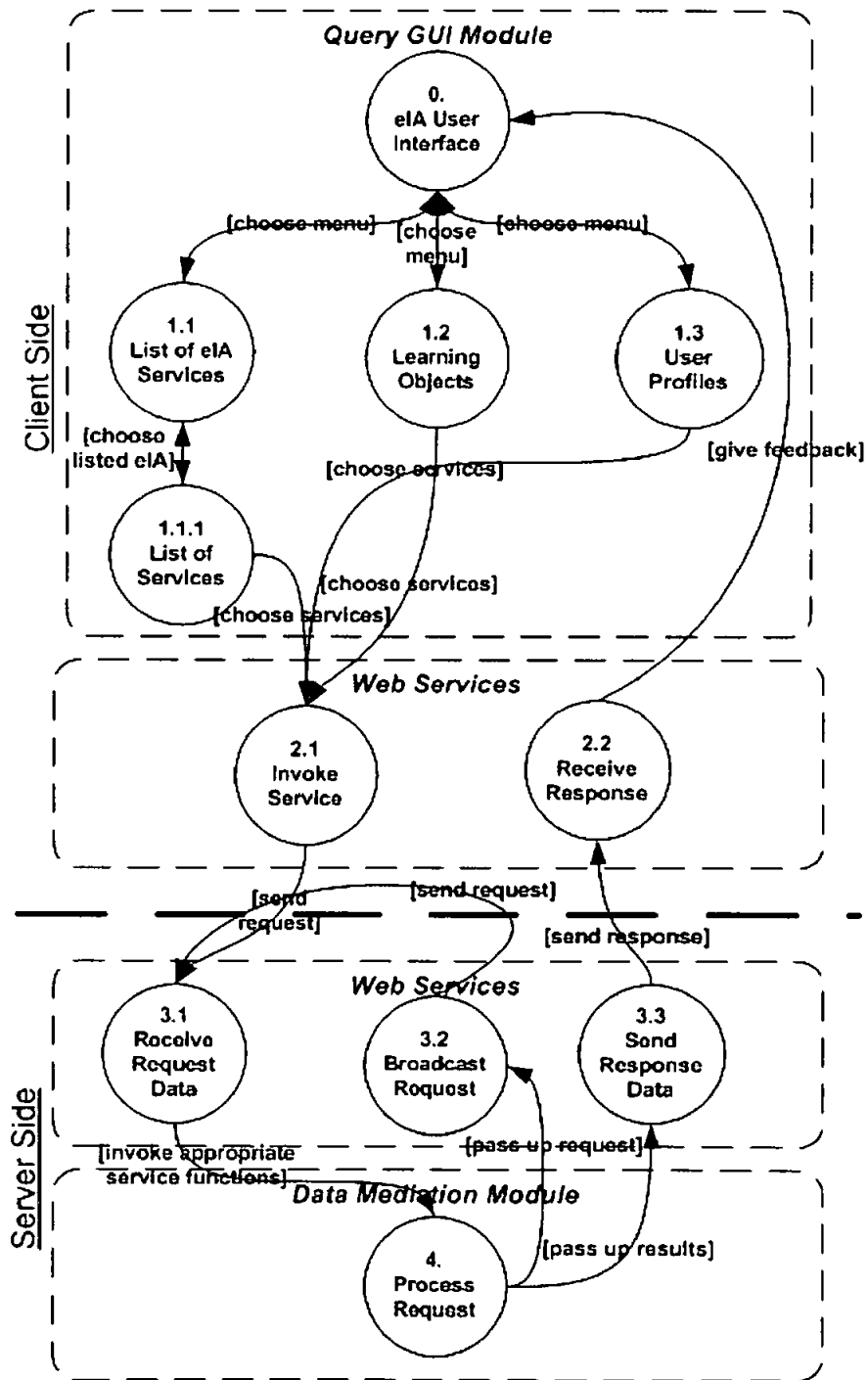


Figure 3-11: Overall eIA Flow State Diagram

### 3.6 Generic Interoperability Agent

This thesis presents an interoperability agent in a very specific domain: e-learning. Through the breakdown of eIA model in this chapter, it is feasible to generalize the model and implement it in other domains. There are some important things needed to be pondered in generalizing the model and implementing it in other domains:

- a. Why does the particular domain need the IA (Interoperability Agent)?
- b. What are the basic services of IA to fulfill the point (a) needs?
- c. What are the data involved in those services described in point (b)?
- d. Is there any data standardization used in point (c) available in the market?
- e. How to mediate the data architecture differences between parties in the particular domain?
- f. How to mediate intercommunication between parties in the particular domain?

As an example of this discussion, one wants to implement the model in a health care domain, such as hospital or clinic, namely Healthcare Interoperability Agent (HIA). There is a need in this domain to get a comprehensive individual's record for one's health, i.e. previous diagnoses, including allergies, and genetic dispositions, or medications used. The purpose is to avoid health professionals to act blindly without any background data of the patient. Hence, they have to repeat the tests and other practitioners often do not know an individual's previously identified conditions. In worse case, wrong medication could lead the patient to death.

Thus, HIA's basic service will be to retrieve individual's health or medical record. This record can be gathered from several hospitals or clinics which have treated the individual before. The integration data involved in retrieving the record will be IC (identity card) number or maybe passport number. Next step is to decide the data standardization for describing a health record. One of the organizations regarding this matter is Health Level Seven, Inc. (HL7) [HL7, 2007]. HL7 has produced standards, guidelines, and methodologies to enable the exchange and interoperability of electronic health records. Such guidelines or data standards are a set of rules that allows information to be shared and processed in a uniform and consistent manner. These data standards are meant to allow healthcare organizations

to easily share clinical information. The final step is to consider the communication interchange between the hospitals or clinics. Table 3-2 summarizes the mapping process from eIA to HIA.

Table 3-2: Generic Interoperability Agent Mapping

	eIA	Generic Interoperability Agent (IA)	HIA
<b>Communication</b>	eIA Communication Layer: Web service.	IA Communication Layer: Web service is a software system designed to support interoperable machine to machine interaction over a network, such as the Internet. The requested service is executed on a remote system hosting the service. Thus, web service is applicable to mediate the communication between different systems.	HIA Communication Layer: Web service is applicable to mediate the communication.
<b>Data</b>	eIA Data Layer: 1. Basic services: Learning Objects and User Profiles. 2. Standardized data based on IEEE LOM and IMS LIP. 3. Data mapping technology: XSLT.	IA Data Layer: 1. Basic services needed in the particular domain. 2. Standardized data involved in processing the basic services. 3. Data mapping technology: XML primary purpose is to facilitate the sharing of structured data across different systems. Thus, XML technologies, specifically XSLT, can be used as the mapping platform.	HIA Data Layer: 1. Basic services: to retrieve an individual health or medical record. 2. Standardized data based on HL7 for example. 3. Data mapping technology: XSLT is applicable to map the data schema.
<b>Interface</b>	eIA Query GUI Module: Web based interface (HTML).	The interface layer purpose is as media of the agent to interact with users. Thus, the agent's user interface depends on the applications used in the particular domain. Generally, the agent's interface is integrated easier on web-based applications.	HIA User Interface depends on the applications used by the particular hospitals or clinics.

### 3.7 Summary

This chapter has presented the whole concept of e-learning Interoperability Agent except for the semantic integration which will be explained in Chapter Four. With the eIA architecture, it is feasible that the implementation of eIA causes only minor changes in the pre-existent e-learning systems. eIA's user interface layer can be smoothly integrated within the existing e-learning web-based interface. Meanwhile, the data and communication layer are integrated as extension and interact using the existing e-learning architecture. However, some database changes needed i.e. to store eSD and other data needed by eIA.

## CHAPTER FOUR : EIA SEMANTIC INTEGRATION

In the context of this work, interlingua ontology is the proposed semantic integration schema. The schema relies on metadata publishing to allow ontology to be linked or mapped. Metadata publishing is the process of making metadata data elements available to external users, both people and machines, in this case eIA. This work approaches metadata publishing by two steps:

1. Analyze requirements of eIA basic functionalities (learning objects and user profiles) and produce eIA metadata;
2. Analyze needed standard functions and formalize the WSDL document.

### 4.1 eIA's Metadata

eIA's metadata consists of two data classes to fulfill each of the basic functionalities. The first class is eIA Learning Object Metadata (eLOM) for the learning objects functionality. The second class is eIA Learner Information Metadata (eLIM) for the user profiles functionality.

#### 4.1.1 eIA Learning Object Metadata

By scrutinizing IEEE LOM, it is found that the importance of each element of the IEEE LOM varies. Some elements, for example title of the general category and location of the technical category, are indispensable for nearly all learning objects, while duration of the technical category may not make any sense to an image, and relation category is not useful for most raw media, too. Hence, it is not effective and flexible to make all the LOM elements mandatory to all users indiscriminatingly. Therefore, eIA limits elements of learning object metadata only to elements considered necessary. Some works related to LOM, such as from Najjar et al. [Najjar et al. 2003], Xiang et al. [Xiang et al., 2003], and Chan et al. [Chan et al., 2004], are also taken into account in delivering the eLOM.

Figure 4-2 draws the schema of eLOM. There are mandatory categories and optional categories. Mandatory categories, general and technical, are drawn by solid lines. Optional categories are drawn by dash lines. Inside each of the categories, there is also solid and dash lines. It means that not all elements of each category needed. Meanwhile, if all the elements are drawn with solid lines, it means that all the elements are needed to form the particular category. There is one new element, rating of annotation category, which is not defined in IEEE LOM.

“Rating” is an optional element to provide users with inter-rater capability, depends on each of VLE system availability. This element tries to assist users in selecting LOs by filtering the LOs based on a rate. The rate is an evaluation value of a learning object on a five points scale. This work simplifies the rate as one value of overall evaluation of a learning object, compares with Learning Object Review Instruments (LORI) [BELFER, 2003] which measures nine separates qualities of learning objects. Figure 4-1 depicts the flowchart of eIA rate-filtering. A full description of eLOM is given in Appendix A.

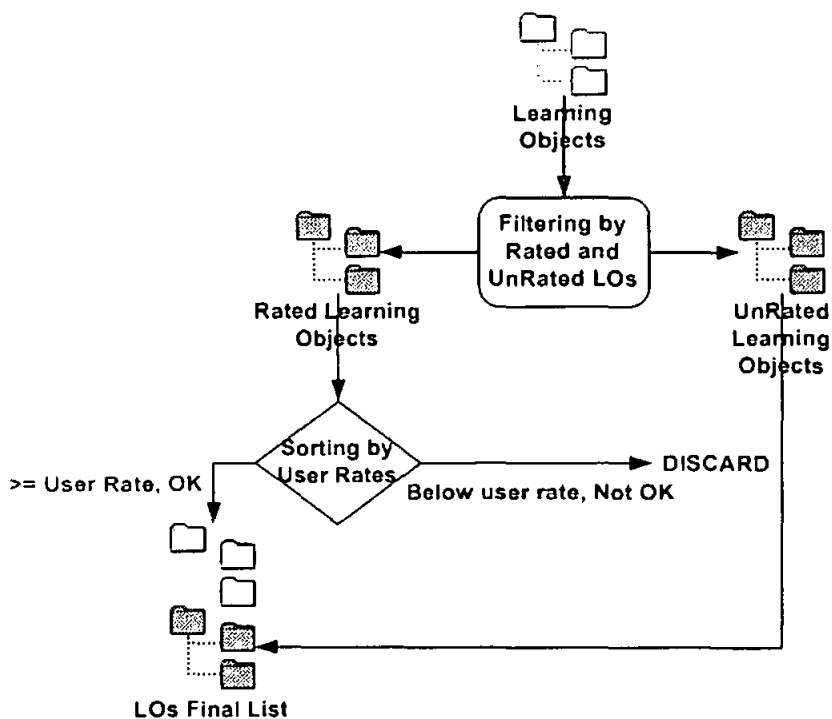


Figure 4-1: eIA LOs Rate Filtering

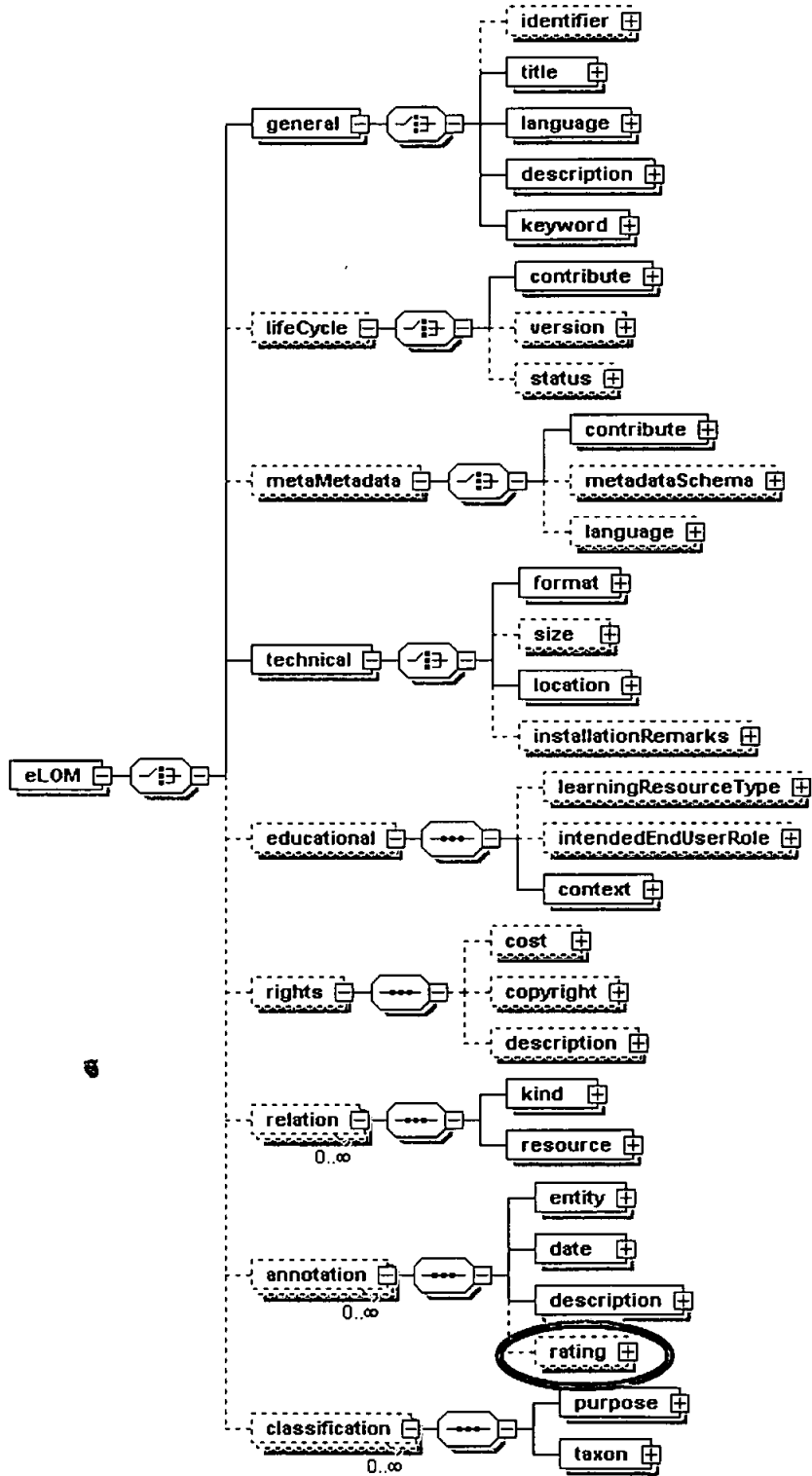


Figure 4-2: eIA Learning Object Metadata Schema



#### 4.1.2 eIA Learner Information Metadata

The eLIM schema is designed based on core classes defined in IMS LIP. Related work from Madhour et al. [Madhour et al., 2006] also taken into account as references. Figure 4-3 shows the schema of eLIM. The full description of the schema is elaborated in Appendix A. eLIM groups the categories into profile and pedantic groups. Profile group is mandatory and pedantic group is optional. Profile group contains data for relatively static information about learners and their preferences, similar to those found in a curriculum vita, and comprises largely of the fields found in the LIP. It consists of two mandatory categories (identification and accessibility) and two optional categories (interest and relationship). Meanwhile pedantic group consists of two optional categories, competency and knowledgeRepository.

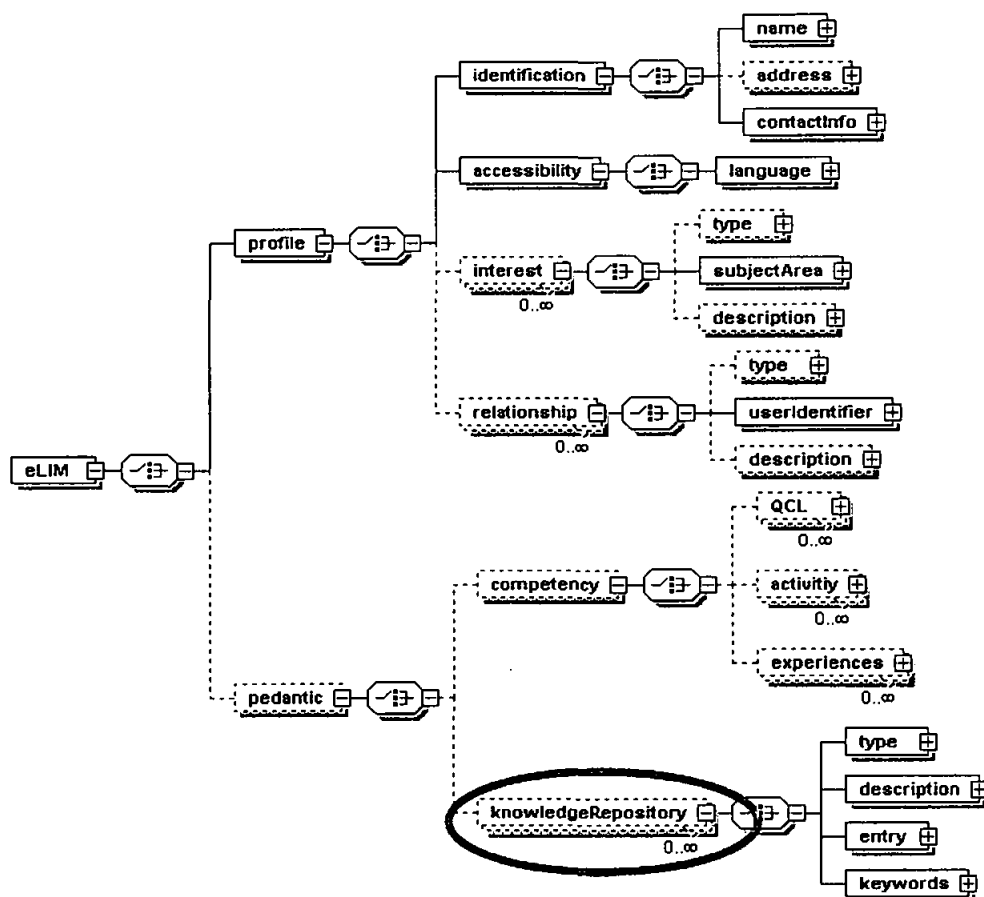


Figure 4-3: eIA Learner Information Metadata Schema

The conception of the learner's knowledge model goes beyond what is described in LIP competency, to include not only competencies in terms of demonstrable knowledge but also descriptions of what the learner knows and can build upon. Therefore, there are two new add-on things: experiences element of competency category and knowledgeRepository category. Experiences element tackles information related to learner's experiences or knowledge that are not described in formal term (QCL element). knowledgeRepository category enable learners to share their own knowledge, in form of informal learning objects and also forum-like or discussion-like messages (section 3.1.1).

#### **4.2 eIA's Basic Services**

eIA basic services are standardized services that should be provided by each of eIA implementation. There are two types of these services: explicit and implicit services. Explicit services are services that published in WSDL documents. In other words, users of eIA can retrieve list of available services by extracting them from WSDL documents and utilize them. Explicit services themselves are categorized into communication and utilization services.

Communication services are services initializing interaction with other eIAs. The services are login service and logout service. Login service is a service to authorize eIA client interaction requests. As explained in section 3.2, login service returns session key if the authorization is valid. After an eIA client get the session key, eIA client can start the interaction with eIA server with the given session key. Logout service is requested by eIA client when eIA client wants to close the interaction with the server. Thus, the session key is no longer valid. New session key must be obtained if the client wants to re-open the interaction.

Utilization services are services related to two basic functions of eIA: learning objects and user profiles (UP). Search and retrieve LOs are two mandatory LO services. Meanwhile rate LOs is optional as described in section 4.1.1. eIA gives users to provide some parameters for filtering the LOs: title, language, keywords, date of contribution, format, intended end user role, context, cost, and rating. Return values

expected contains information as defined in eLOM (section 4.1.1). Each of the eIA servers is given freedom to use their own algorithm when searching for learning objects into their database.

Meanwhile UP services provide functionalities to search for users, retrieve the profiles, and contact the users. eIA gives users to provide some parameters for filtering the users: country, language, and subject areas. Return values expected contains information as defined in eLIM (section 4.1.2). Figure 4-4 summarizes all the explicit services. The specific algorithm and implementation of all the services are left to each of the VLE systems.

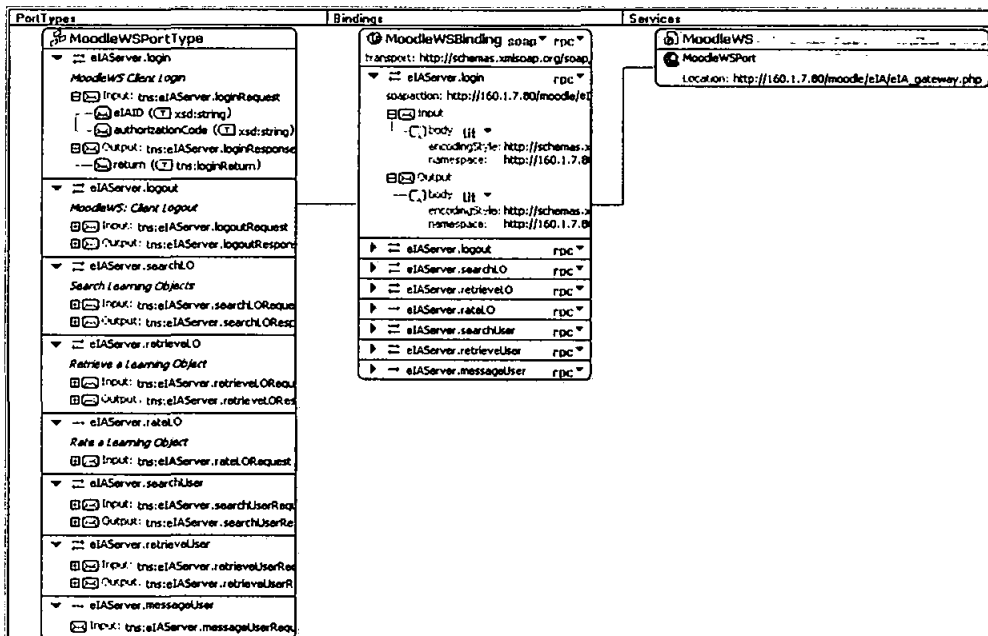


Figure 4-4: eIA WSDL Schema

eIA has one implicit service: broadcast service. The role of this service is related with the automatic interfacing of network eIA services. This service broadcast request messages from eIA client to other eIA server if the request is still in the constraint. There are two types of constraint: depth and time. Depth constraint limits the broadcast service in matter of depth. Figure 4-5 shows a sample of eIA network. The arrows in the sample means the registered eIA, e.g. A is registered with B, C, and D; B is registered with A and E. For example, if users from A specify the depth

constraint as one, then eIA only broadcasts the request to B, C, and D. If users from A specify the depth constraint as two, then eIA only broadcasts the request to B, C, D, E, F, G, and H.

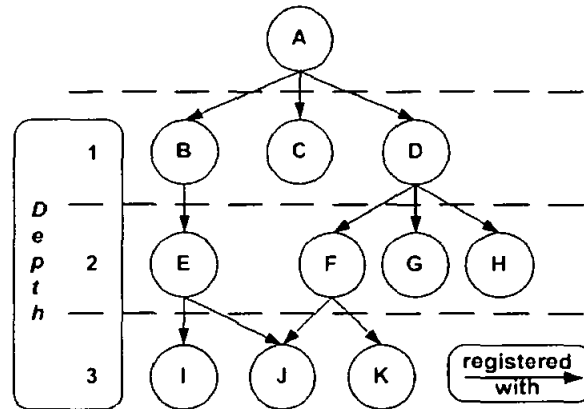


Figure 4-5: eIA Network Tree Sample

Meanwhile a time constraint means limits the maximum physical time (in seconds) that eIA spends to execute the requested service. For example, when the request message from A reaches F, eIA-F checks for the current execution time. The current execution time is counted since the message sent from A until it reaches F. If the current execution time is still below the time constraint, then F will broadcast the message to J and K, else the message broadcasting stops at F.

Besides those two constraints, there is also a condition where the same request can be broadcasted to the same eIA twice or more. For example, J can receive the same request broadcasted by E and F. Thus, if J found the same request, then J only processes one of the messages and discards the others. Figure 4-6 summarizes the broadcast algorithm.

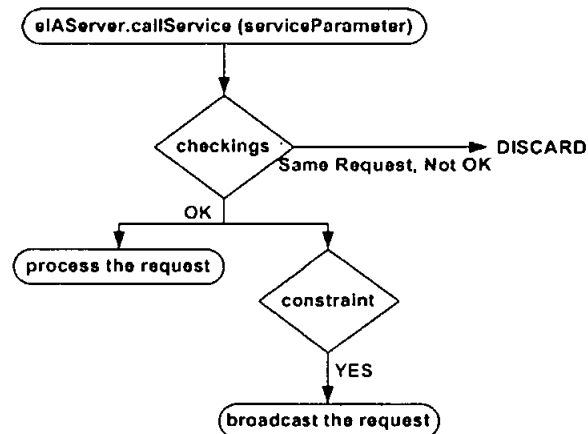


Figure 4-6: eIA's Broadcast Algorithm

### 4.3 eIA's Extended Services

Beside basic services, eIA comes with capability to extend other services, such as sharing of forums. These extended services are published at WSDL. There are some conventions which need to be followed if one wants to publish extended services:

1. There are two types of messages for the particular service: request message and response message. A service may either use both of the messages or just one of the messages. The naming convention of the messages follows this rule: `servicenameRequest` for request messages and `servicenameResponse` for response messages. For example, there is a service called `searchForums`, then the messages name should be `searchForumRequest` and `searchForumResponse`.
2. The service must provide the documentation of the service, i.e. `<documentation>` element. The documentation tells the usage of the service.

Based on the naming conventions, eIA with its XSLT library gives users functionality to retrieve these extended services lists and invoke them. There are two important processes involved: translating the WSDL into users' form (to invoke the services) and translating the server response into users' result page. This library template was created and can be extended to allow for rapid customization of usages. Figure 4-7 depicts the sketch of XSLT-WSDL library template.

```

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:fo="
http://www.w3.org/1999/XSL/Format" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="
http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" >
  <xsl:key use="@name" name="datatype" match="//xsd:schema//xsd:complexType"/>
  <xsl:key use="@binding" name="accesspoint" match="//wsdl:service//wsdl:port"/>
  <xsl:template match="/">
    <!-- Page and Table Header here -->
    <xsl:for-each select="wsdl:definitions/wsdl:message">
      <xsl:choose>
        <xsl:when test="contains(@name,'Request')">
          <xsl:if test="not(contains(@name,'login')) and not(contains(@name,'logout')) and
not(contains(@name,'getLO'))"><xsl:call-template name="messageFilter"/></xsl:if>
        </xsl:when>
      </xsl:choose>
    </xsl:for-each>
    <!-- Page and Table Footer here -->
  </xsl:template>
  <xsl:template name="messageFilter">
    <form action="eIA_ui_services_core.php" method="post">
      <input type="hidden" name="accessPoint">
        <xsl:attribute name="value">
          <xsl:value-of select="
concat(substring-before(key('accesspoint','{ns:eIAWSBinding}/soap:address/@location,'eIA_gateway.php'),wsdl.
php)"/>
        </xsl:attribute>
      </input>
      <input type="hidden" name="action">
        <xsl:attribute name="value"><xsl:value-of select="@name"/></xsl:attribute>
      </input>
      <xsl:value-of select="@name"/><br/><small><i><xsl:value-of select="wsdl:documentation"/></i></small>
      <!-- Sub Table Header here -->
      <xsl:for-each select="wsdl:part">
        <xsl:choose>
          <xsl:when test="contains(@type,'xsd:')">
            <xsl:value-of select="@name"/> ( <xsl:value-of select="substring-after(@type,'xsd:')"/> )
            <input type="text">
              <xsl:attribute name="name"><xsl:value-of select="@name"/></xsl:attribute>
            </input>
            <input type="hidden" name="dataarray" value="no"/>
            <input type="hidden" name="parameters[]">
              <xsl:attribute name="value"><xsl:value-of select="@name"/></xsl:attribute>
            </input>
          </xsl:when>
          <xsl:when test="contains(@type,'{ns:}')">
            <xsl:value-of select="substring-after(@type,'{ns:}')"/>
            <xsl:for-each select="key('datatype',substring-after(@type,'{ns:}'))/xsd:all/xsd:element">
              <xsl:choose>
                <xsl:when test="contains(@name,'requestID')">
                  <input type="hidden" name="requestID" value="eIA_N/A"/>
                  <input type="hidden" name="parameters[]">
                    <xsl:attribute name="value"><xsl:value-of select="@name"/></xsl:attribute>
                  </input>
                </xsl:when>
                <xsl:otherwise>
                  <xsl:value-of select="@name"/> ( <xsl:value-of select="substring-after(@type,'xsd:')
"/> )
                  <input type="text">
                    <xsl:attribute name="name"><xsl:value-of select="@name"/></xsl:attribute>
                  </input>
                  <input type="hidden" name="dataarray" value="yes"/>
                  <input type="hidden" name="parameters[]">
                    <xsl:attribute name="value"><xsl:value-of select="@name"/></xsl:attribute>
                  </input>
                </xsl:otherwise>
              </xsl:choose>
            </xsl:for-each>
          </xsl:when>
          <xsl:otherwise>
            <xsl:value-of select="@name"/> ( <xsl:value-of select="@type"/> )
            <input type="text">
              <xsl:attribute name="name"><xsl:value-of select="@name"/></xsl:attribute>
            </input>
            <input type="hidden" name="dataarray" value="no"/>
            <input type="hidden" name="parameters[]">
              <xsl:attribute name="value"><xsl:value-of select="@name"/></xsl:attribute>
            </input>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:for-each>
    </form>
  </xsl:template>
</xsl:stylesheet>

```

Figure 4-7: Sketch of XSLT – WSDL Library Template

#### **4.4 Summary**

This chapter has elaborated the semantic integration used by eIA. Not all of the elements defined in eLOM and eLIM are mandatory, but one eIA must implement all the mandatory elements (eLOM: general and technical; eLIM: identification and accessibility) to fulfill its desired functionality. In the matter of eIA's basic services, the VLE is given privilege to apply the search algorithm based on existing functionalities in their own system. In this way, they can protect private data that they considered should not be exposed outside their own system.

## CHAPTER FIVE : EVALUATION

The evaluation of the eIA model was done through prototyping the model into two widely used open source e-learning systems as the test-bed platforms. The initial eIA prototype provided much insight into what was required to help VLE owners extend their systems with eIA. The prototype, however, was developed with only the primary functionalities implemented. Lastly, this chapter also discusses the creation of Federated e-learning Community (FeC) through e-learning and eIA.

### 5.1 Prototyping

As introduced in section 1.1, a fine-grained database-oriented e-learning environment could be conceived as three layered system: Database System (DS) at the lowest layer, E-learning Core (EC) at the middle layer, and User Interface (UI) as the upper most layer. eIA with its three components (web services, data mediation module, and query GUI module) interacts with all of VLE system layers. Data mediation module and query GUI module are two components that tightly attach to the VLE system. Tightly attach here means that both components are interacting directly with the VLE system, meanwhile web services component is used to interact with other eIAs and interacts with VLE system through data mediation module. All of the eIA components are deployed inside EC. Through EC, eIA with its query GUI module accesses the UI layer and able to do interaction with VLE users. Meanwhile with its data mediation module, eIA can access the DS. Figure 5-1 shows the layers of common e-learning system architecture and how eIA attaches to the system.



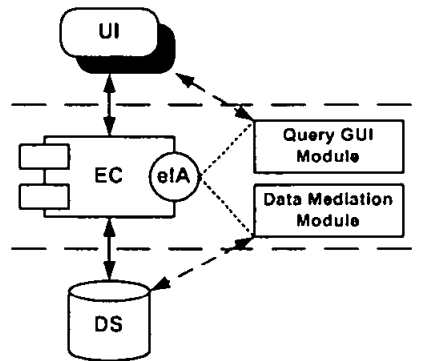


Figure 5-1: eIA Implementation Overview

### 5.1.1 Test Bed Platforms

This work evaluates the agent by prototyping it into two different e-learning system platforms. When deciding the e-learning test bed platforms, there are some requirements and considerations taken into account:

- The e-learning systems should be open source systems.
- The use of Moodle (Modular Object-Oriented Learning Environment), one of the open source systems, as e-learning platform at Universiti Teknologi PETRONAS is a considerable choice of platform to test the agent in real system.
- Because of research time limitation, both of the chosen platforms have more or less the same platform environment so it can shorten the time needed for implementation.

Thus, Moodle is the first chosen test bed platforms. The second chosen test bed platform which also fulfills the requirements and considerations is ATutor. Both Moodle and ATutor have been evaluated and chosen as top two open source e-learning systems out of 36 candidates [van den Berg, 2005].

Figure 5-2 shows the eIA's client/server architecture in Moodle and ATutor. Moodle and ATutor are fully web-based, written in PHP and MySQL, and run on Apache web server. eIA implements the SOAP based on NuSOAP [Ayala, 2002] library.

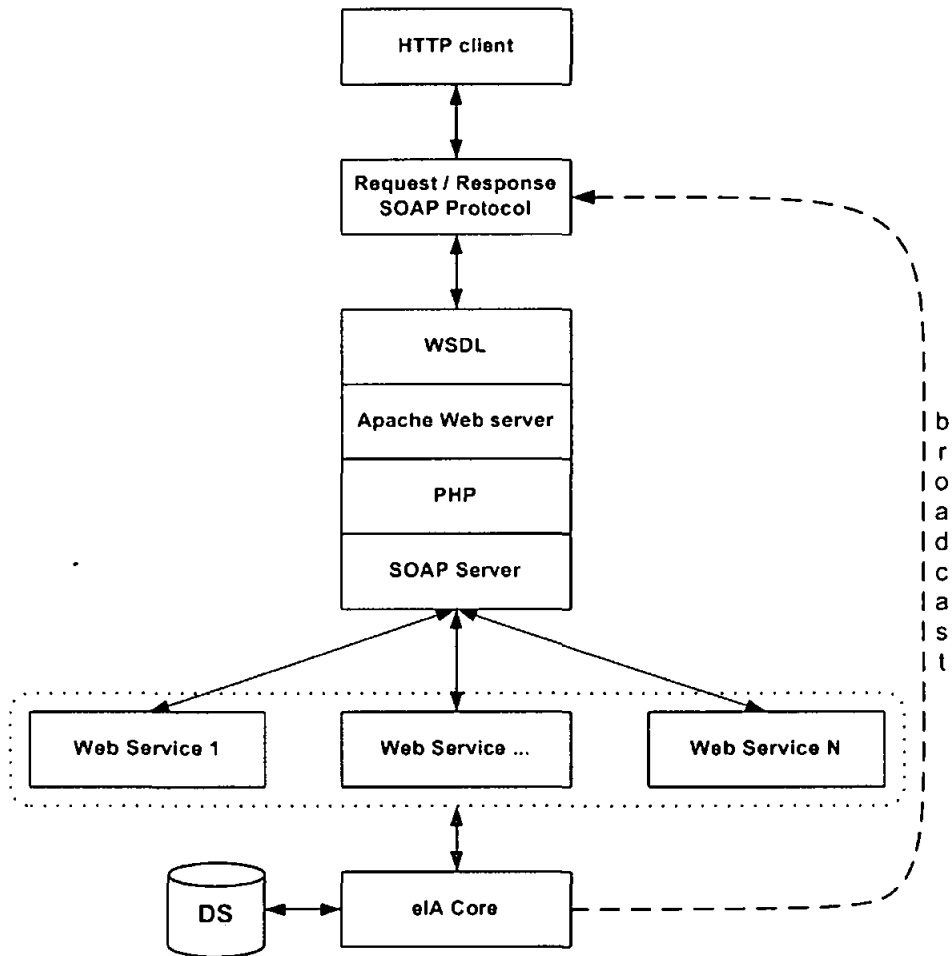


Figure 5-2: eIA's Client/Server Architecture

#### 5.1.1.1 Moodle

Moodle is an open source VLE system created by Martin Dougiamas, a former WebCT administrator at Curtin University, Australia. It has a significant user base with 32171 registered sites with 13,510,225 users in 1,340,683 courses and more than 75 supported languages in 175 countries (as of July 1, 2007) [Moodle, 2007]. In Malaysia itself, there are 157 sites registered (as of July 1, 2007). This statistics means that eIA has a good prospect to be widely used by each of Moodle sites. Thus, it also means it has a better chance to create the FeC.

Moodle has been developed with a modular approach, which should make it easy for administrators to configure a customized version, and for developers to add new extension modules. The Moodle structure is such, that new modules can be added by adding directories to certain parts of the directory tree. Guidelines and rules for developing new modules are published in the Moodle website [Moodle, 2007]. However, eIA's user interface resides as one of the Moodle blocks (see Figure 5-3).

The screenshot displays a Moodle course page with the following components:

- Upcoming Events:**
  - New MPPUTP P&P Medium: Friday, 25 January (11:45 PM) - Sunday, 2 March (12:00 AM)
  - Collection of Foundation to Undergraduate Otter Letter: Friday, 1 February (09:50 AM) - Tomorrow (09:50 AM)
- Contact Us:** System Administrator, Email: elearn@utp.edu.my
- eLearning Interoperability Agent:**
  - List of Registered Service(s)
  - Search for LO(s)
  - Search for User(s)
  - My Profile
- Calendar:** February 2008. A calendar grid showing the current date as the 17th.
- My courses:**
  - Latest News from Rector Jan 2008
  - Object-Oriented Programming
  - MPPUTP
  - All courses...
- Messages:** No messages waiting. Messages...
- Activities:** Forums
- Footer:** Moodle logo and "You are logged in as Tutor OOP (Logout)"

Figure 5-3: Moodle and e-learning Interoperability Agent

The eIA's implementation remarks in Moodle are as follows:

1. eIA User Interface Layer (Query GUI Module)

As mentioned before, eIA's user interface resides as one of the Moodle blocks. Blocks resides either on the left or right of the web page with the centre column containing the course content. Blocks may be added, hidden, deleted, and moved up, down and left/right when editing is turned on.

2. eIA Data Layer (Data Mediation Module)

Moodle currently does not come with ready-to-use Metadata profiles. Thus, eIA needs to extract the required data from Moodle database. The extracted data to fulfill the basic LOs and user profiles (UP) functionalities are extracted from several tables as drawn in Figure 5-4.

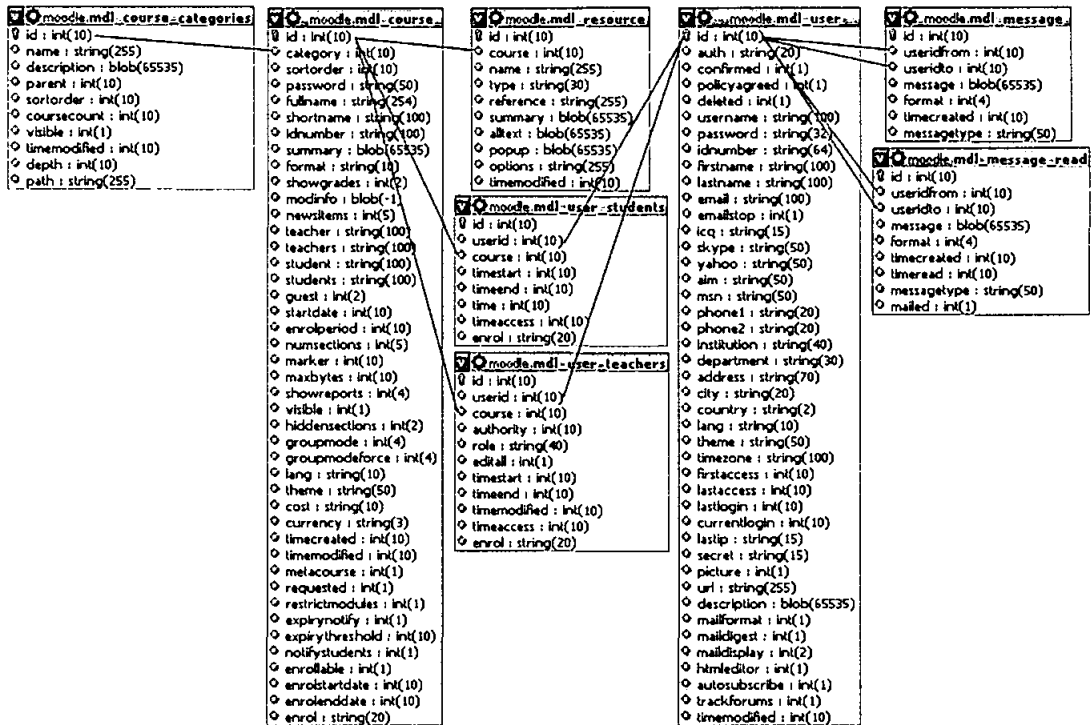


Figure 5-4: Extracted Moodle Database

Table 5-1 affirms the mapping of extracted Moodle’s database into learning object metadata to fulfill the LO functionality. There are two tables entangled: mdl\_resource and mdl\_course. Each of the learning objects stored in the database (mdl\_resource) is owned by one particular course (mdl\_course). Thus in filtering based on the keyword(s), the description of a learning object can also be extended with the owner course description.

Table 5-1: Moodle LO Data Mapping

Category	Element	Extracted Moodle's Database
General	title	mdl_resource.name
	language	mdl_course.lang
	description	mdl_resource.summary
	keyword	mdl_resource.summary, mdl_course.summary
Life Cycle	contribute (date)	mdl_resource.timemodified
Technical	format	mdl_resource.type, mdl_resource.reference
	location	mdl_resource.id, mdl_resource.course
Rights	cost	mdl_course.cost

Table 5-2 affirms the mapping of extracted Moodle's database into user profile metadata to fulfill the UP functionality. There are four tables entangled: mdl\_user, mdl\_user\_students, mdl\_user\_teachers, and mdl\_course. Each of the users usually enrolls with one or more course. Thus in filtering based on subjectarea (interest or competency), the metadata of the users can also be extended with information of the enrolled course ( $mdl\_user \cap mdl\_user\_students \cap mdl\_course$  or  $mdl\_user \cap mdl\_user\_teachers \cap mdl\_course$ ).

Table 5-2: Moodle User Profiles Data Mapping

Category	Element	Extracted Moodle's Database
Identification	name	mdl_user.firstname, mdl_user.lastname
	address(country)	mdl_user.country
	contactinfo	mdl_user.email, mdl_user.url
Accessibility	language	mdl_user.lang
Interest or Competency	subjectarea	mdl_user.department, mdl_user.description, $mdl\_user \cap mdl\_user\_students \cap mdl\_course$ or $mdl\_user \cap mdl\_user\_teachers \cap mdl\_course$ (course taken)

Beside those four tables, there are two more tables used by eIA related to UP functionality: `mdl_message` and `mdl_message_read`. These tables are used for the message service of the UP functionality.

### 5.1.1.2 ATutor

ATutor (Accessible Tutor) is an open source web-based LCMS created by the Adaptive Technology Resource Centre University of Toronto, Canada. It is used internationally and has been translated into over fifteen languages with support for over seventy additional language modules currently under development (as of July 1, 2007) [ATutor, 2007].

ATutor introduces the concept of modules, providing developers with a framework to implement additional functionality in a coherent and loosely coupled way. The framework defines methods for assigning privileges, backing-up, and restoring content, deleting course specific content, and adding side menu blocks, student tools, course management and administrative tools, as well as public tools and other types of added functionality. The intent is to allow for the development and distribution of modules independent of the ongoing development and release of ATutor. Hence, eIA is feasible to be deployed inside ATutor as a new modules. Figure 5-5 shows how eIA resides as a modules inside ATutor's user interface.

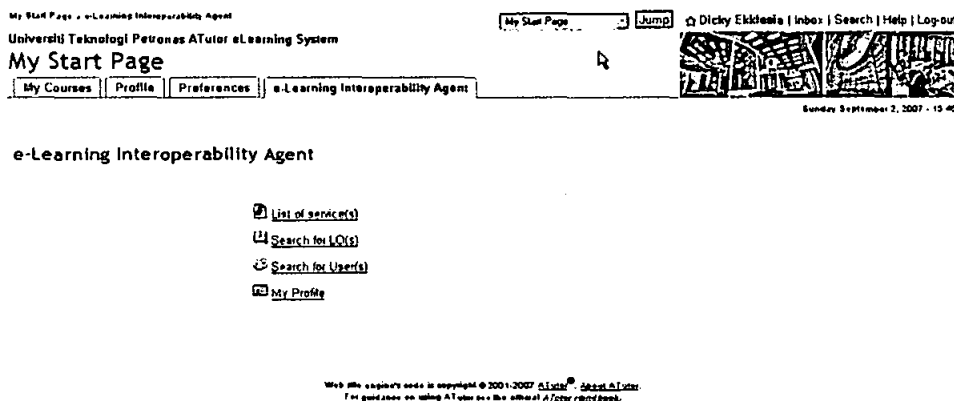


Figure 5-5: ATutor and e-learning Interoperability Agent

The eIA's implementation remarks in ATutor are as follows:

1. eIA User Interface Layer (Query GUI Module)

As mentioned before, eIA is deployed as one of the ATutor modules. The Modules resides as one of the tab menu in the e-learning page.

2. eIA Data Layer (Data Mediation Module)

ATutor currently does not come with ready-to-use Metadata profiles. Thus, eIA needs to extract the required data from ATutor database. The extracted data to fulfill the basic LOs and user profiles functionalities is extracted from several tables as drawn in Figure 5-6.

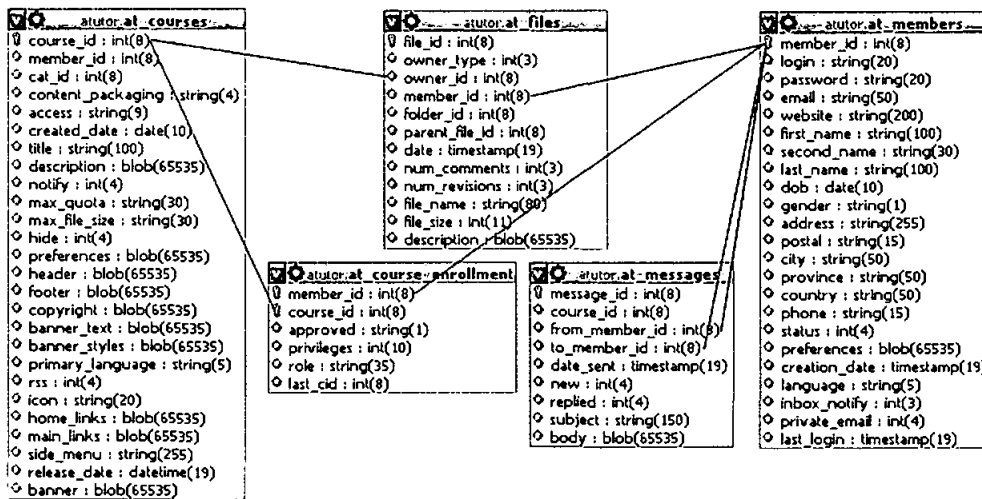


Figure 5-6: Extracted ATutor Database

Table 5-3 affirms the mapping of extracted ATutor's database into learning object metadata to fulfill the LO functionality. There are two tables entangled: at\_files and at\_courses. Each of the learning objects stored in the database (at\_files) is owned by one particular course (at\_courses). Thus in filtering based on the keyword(s), the description of a learning object can also be extended with the owner course description.

Table 5-3: ATutor LO Data Mapping

Category	Element	Extracted ATutor's Database
General	title	at_files.title
	language	at_courses.primary_language
	description	at_files.description
	keyword	at_files.description, at_courses.description
Life Cycle	contribute (date)	at_files.date
Technical	format	at_files.title
	size	at_files.file_size
	location	at_files.file_id

Table 5-4 affirms the mapping of extracted ATutor's database into user profile metadata to fulfill the UP functionality. There are three tables entangled: at\_members, at\_course\_enrollment, and at\_courses. Each of the users usually enrolls with one or more course. ATutor database does not provide description of the users. Thus in filtering based on subjectarea (interest or competency), the metadata of the users can be filled-in with information of the enrolled course ( $at\_members \cap at\_course\_enrollment \cap at\_courses$ ).

Table 5-4: ATutor User Profiles Data Mapping

Category	Element	Extracted ATutor's Database
Identification	name	at_members.first_name, at_members.second_name, at_members.last_name
	address(country)	at_members.country
	contactinfo	at_members.email, at_members.website
Accessibility	language	at_members.language
Interest or Competency	subjectarea	$at\_members \cap at\_course\_enrollment \cap at\_courses$

Beside those three tables, there is one more table used by eIA related to UP functionality: at\_message. This table is used for the message service of the UP functionality.



## 5.1.2 eIA Services Implementation

As elaborated in section 3.1.1, there are three use cases taken into consideration regarding roles and basic functionalities of eIA: eIA administration, dynamic functions invocation, learning objects functionality, and user profiles functionality. This section presents implementation results of the services introduced in the use cases.

### 5.1.2.1 eIA Administration

The eIA pilot prototype does not provide any specific interface for the administrator to update any changes of the registered services. In this prototype, phpMyAdmin [phpMyAdmin, 2006] was used as the updating tool of the eIA service directory (refer to Figure 5-7).

Server: localhost Database: moodle Table: mdl\_eia\_service\_directory  
"Registered eIA Database"

Browse
  Structure
  SQL
  Search
  Insert
  Export
  Import
  Operations

Empty
  Drop

Field	Type	Collation	Attributes	Null	Default	Extra
<input type="checkbox"/> id	int(10)		UNSIGNED	No		auto_increment
<input type="checkbox"/> name	varchar(64)	utf8_unicode_ci		No		
<input type="checkbox"/> description	varchar(256)	utf8_unicode_ci		Yes	NULL	
<input type="checkbox"/> overviewURL	varchar(128)	utf8_unicode_ci		Yes	NULL	
<input type="checkbox"/> accessPoint	varchar(128)	utf8_unicode_ci		No		
<input type="checkbox"/> authorizationCode	varchar(8)	utf8_unicode_ci		No		
<input type="checkbox"/> verifiedIP	varchar(15)	utf8_unicode_ci		No	0.0.0.0	

Check All /  Uncheck All With selected:

Figure 5-7: eIA Service Directory

### 5.1.2.2 eIA Dynamic Function Invocation

As elaborated in section 4.3, eIA server administrator may extend eIA functionalities with new services beside the required learning objects and user profiles services. The users may invoke the functions by browsing to the available list as shown in Figure 5-8. In the figure, eIA service directory contains two registered eIA server, namely:

Localhost ATutor and Office Moodle. By choosing one of the registered eIA servers, user is taken to another page which shows the list of particular eIA services. This list is produced based on the transformation results of WSDL document published by the eIA server (refer to section 4.3).

Home » eIA » List of Registered Service(s)

### List of Registered Service(s)

Name	Description	Overview Link
Localhost ATutor	ATutor at Localhost	http://164.04.9/atutor
<b>Office Moodle</b>	Moodle at Office	http://160.1.7.80/moodle/

Home » eIA » List of service(s) » Retrieve Service(s)

### Available Service(s) on Office Moodle

Service Name	Service Parameter
eIAServer.searchUPRequest <small>Some Docs for searchUP parameters</small>	language ( string ) <input type="text"/> country ( string ) <input type="text"/> area ( string ) <input type="text"/>
eIAServer.searchLORequest <small>Some Docs for searchLO parameters</small>	title ( string ) <input type="text"/> keywords ( string ) <input type="text"/> language ( string ) <input type="text"/> format ( string ) <input type="text"/>

Figure 5-8 shows the dynamic functions invocation list. It includes a table of registered services and a detailed view of available services on Office Moodle with input fields for parameters.

Figure 5-8: eIA Dynamic Functions Invocation List

Thereafter, user can choose one of the available service(s) listed and enter the required service parameter(s). eIA client sends the chosen service with its service parameter(s). In the end, eIA client receives the service response and transforms the response based on the XSLT library provided by the eIA server as shown in Figure 5-9.

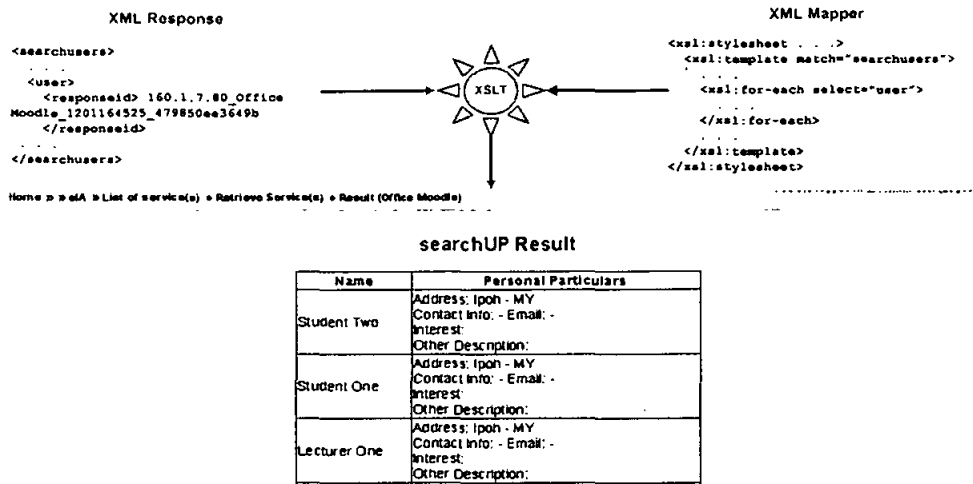


Figure 5-9: eIA Dynamic Functions Invocation Result

The skeleton code of the XML mapper is as follow:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">

<xsl:template match="eIAServiceResults">

  <!-- Page or Table Header -->
  <!-- iterate through the results ->
  <xsl:for-each select="eIAObjectResults">

    <!-- Print each of the Results -->

    <xsl:value-of select="objectSpecificField"/>

  </xsl:for-each>

</xsl:template>
</xsl:stylesheet>
```

### 5.1.2.3 eIA Learning Objects Services

The eIA pilot prototype does not provide any specific interface for the users to upload or update their published learning objects. Users can upload or update their published learning objects through the pre-existent e-learning system features as shown in Figure 5-10 and Figure 5-11. Their published LOs are searchable by eIA based on implementation of the data mediation module in the particular system. Thus, copyrighted or private LOs might be hindered from being searched by eIA.

Adding a new Resource to week 2

Link to a file or web site

Name:

Summary:

Path:

Location:

Window:

Parameters:

Visible to students:

Figure 5-10: Moodle Publish LO Page

Universiti Teknologi PETRONAS ATutor eLearning System  
**Software Engineering**

[Home](#) | [Forums](#) | [Glossary](#) | [File Storage](#) | [Manage](#)

Tuesday February 5, 2008 - 23:47

---

**File Storage**

File Storage Hide

[New File](#)      [Create Folder](#)

\*Upload File

Or [Create a New File](#)

Description

**Content Navigation**

[Home](#)

Welcome To ATutor

**Related Topics**

None Found

**Users Online**

Dicky Ekhsale

Guests are not listed

Figure 5-11: ATutor Publish LO Page

eIA provides user with search for LO(s) form as shown in Figure 5-12. Then, eIA search the LO(s) based on user's LO(s) parameters. Figure 5-13 shows the search LO(s) results and gives user possibility to retrieve the LO(s).

Home > eIA > Search for LO(s)

### Search for LO(s)

Attribute Value

Title

Keywords

Language : UNSPECIFIED

Format :  audio  application  document  
 image  video  UNSPECIFIED

Rating :  1  2  3  4  5

<vsdl:operation name="eIAServer.searchLO">

Figure 5-12: Search for LO(s) Form

Home > eIA > Search for LO(s) Result

### Search for LO(s) Result

Name	Description
[localhost ATutor] SE_Course_Outline.doc	Software Engineering Course Outline
[localhost ATutor] Trees ppt	This file is about tree. Tree in Programming of course.
[Office Moodle] Introduction to problem solving	Lecture on Introduction to problem solving <ul style="list-style-type: none"> <li>• Introduction to structured programming</li> <li>• Problem solving phases                             <ul style="list-style-type: none"> <li>• Algorithm and pseudocode</li> </ul> </li> <li>• Software development methods</li> <li>• Software development concept</li> </ul>
[Office Moodle] Basic Program Structure for Language and Application	Lecture on Basic Program Structure for Language and Application <ul style="list-style-type: none"> <li>• Introduction</li> <li>• Algorithm</li> <li>• Control Structures (Sequence, Selection, Iteration (Loop))</li> <li>• Introducing C Programming</li> </ul>
[Office Moodle] Static Data Structure: Array	Lecture on Static Data Structure: Array <ul style="list-style-type: none"> <li>• Concept and uses of Arrays</li> <li>• One-Dimensional Array</li> <li>• Array Declaration and Initialisation</li> <li>• To be able to pass arrays and array elements to functions</li> <li>• Two-Dimensional Array</li> </ul>

- Click on LO Name to get detail of Selected LO -

Figure 5-13: Search for LO(s) Result

#### 5.1.2.4 eIA User Profiles Services

The eIA pilot prototype does not provide any specific interface for the users to update their profile. Users can update their profile through the pre-existent e-learning system features as shown in Figure 5-14 and Figure 5-15. Their profiles are searchable by eIA based on implementation of the data mediation module in the particular system.

Home » Tutor OOP » Edit profile

### Tutor OOP

[Profile](#)   [Edit profile](#)   [Forum posts](#)

First name:   
 Surname:   
 Email address:   
 Email display:  Allow only other course members to see my email address  
 Email activated:  This email address is enabled  
 Email format:   
 Email digest type:   
 Forum auto-subscribe:  Yes: when I post, subscribe me to that forum  
 Forum tracking:  No: don't keep track of posts I have seen  
 When adding text:  Use HTML editor (some browsers only)  
 City/town:   
 Country:   
 Timezone:   
 Preferred language:   
 Description: 

Trabuchet 1 (8 pt)


The following items are optional:

Current picture:   
 New picture:    
Max size: 50MB  
 Web page:   
 ICQ number:   
 Skype ID:   
 AIM ID:   
 Yahoo ID:   
 MSN ID:   
 ID number:  (for the lecturer only)  
 Phone 1:  (for the lecturer only)  
 Phone 2:  (for the lecturer only)  
 Address:  (for the lecturer only)

Figure 5-14: Moodle User Profile Page

My Courses | Profile | Preferences | e-Learning Interoperability Agent

Profile | Change Password | Change Email | Picture

Profile  Profile

**Required Information**

Login Name  
dicky

Email Address  
dicky@dicky.com  Keep email hidden from others.

\* First Name

Second Name

\* Last Name

**Personal Information (Optional)**

Date of birth  
Year:  Month:  Day:

Sex  
 Male  Female  Not specified

Street Address

Postal/Zip Code

City

Province/State

Country

Telephone Number

Web Site

Figure 5-15: ATutor User Profile Page

eIA provides user with search for User (s) form as shown in Figure 5-16. Then, eIA search the User(s) based on user's parameters. Figure 5-17 shows the search user(s) results and gives user possibility to initiate collaboration with other user(s) by sending message(s).

Home » eIA » Search for User(s)

**Search for User(s)**

Attribute	Value
Country	<input type="text" value="UNSPECIFIED"/>
Language	<input type="text" value="UNSPECIFIED"/>
Subject Areas	<input type="text"/>

`<vcard:operation name="eIAServer.searchUP">`

Figure 5-16: Search for User(s) Form

Home > eIA > Search for User(s) Result

Search for User(s) Result <waid:operation name="eIAServer.sendMessage">

Name	Contact	
[Localhost A Tutor] Dicky Ekdesia	Email: dicky@dicky.com	<input checked="" type="checkbox"/>
[Localhost A Tutor] Student One	Email: student1@student.com	<input checked="" type="checkbox"/>
[Office Moodle] Student Two	Email:	<input checked="" type="checkbox"/>
[Office Moodle] Student One	Email:	<input checked="" type="checkbox"/>
[Office Moodle] Lecturer One	Email:	<input checked="" type="checkbox"/>
[Office Moodle] s1stud1 s1stud1	Email:	<input checked="" type="checkbox"/>
[Office Moodle] m80Lecturer One	Email: m80lec1@m80.edu	<input checked="" type="checkbox"/>
[Office Moodle] m80Lecturer Two	Email: m80lec2@m80.edu	<input checked="" type="checkbox"/>
[Office Moodle] m80Student One	Email: m80stu1@m80.edu	<input checked="" type="checkbox"/>
[Office Moodle] m80Student Two	Email: m80stu2@m80.edu	<input checked="" type="checkbox"/>

- Click on User Name to view detail of the user -

Figure 5-17: Search for User(s) Result

## 5.2 Federated e-learning Community

Through the implementation of eIA inside every VLE system, this work envisions a new type of semantically Federated e-learning Community (FeC) that can be established on, but not limited to, sharable learning objects and users who want to share their knowledge and discuss their interest with other users who have the same interests. The eIA provides a dynamic and collaborative e-learning tool to engage the creation of the community. Each community is dynamically formed by a community of VLE users who have common interest in a specific subject of learning, and who can provide and/or use the knowledge for solving problems in that domain.

Figure 5-18 depicts an example of one single community. There are two types of connections: direct connections and indirect connections. Direct connections, drawn by solid lines, mean that both eIAs have registered to each other, e.g.: eIA-A and eIA-B, eIA-B and eIA-C, and eIA-C and eIA-D. Indirect connections, drawn by dashed lines, mean that both eIAs have not registered to each other. But, both eIAs can still connect between each others through other eIA, which has direct connections with the particular eIA, as the mediator, e.g. eIA-A and eIA-C through eIA-B, eIA-B and eIA-D through eIA-C, eIA-A and eIA-D through eIA-B and eIA-C.



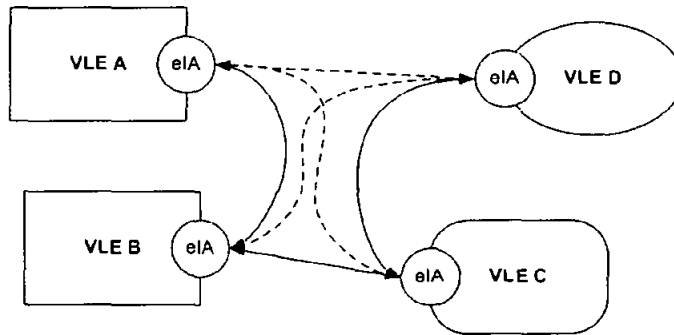


Figure 5-18: Example of Federated e-learning Community

eIA is able to provide a larger set of services by forming a larger or extended FeC from groups of simple FeC. Figure 5-19 illustrates an example of extended FeC. In the example, one single simple FeC consists of VLEs which have direct connections between each others. If one of the VLE members has a direct connection with other VLE from other FeC, e.g. eIA C – eIA D and eIA F – eIA I, then a larger FeC can be formed. It means that eIA with its broadcasting functionality able to intercommunicate with all eIAs within the extended FeC, e.g. eIA-A intercommunicates with eIA-H by passing the message through eIA-C, eIA-D, eIA-F, and eIA-I. eIA-A can have a direct connection with eIA-H if only eIA-A and eIA-H have registered to each other.

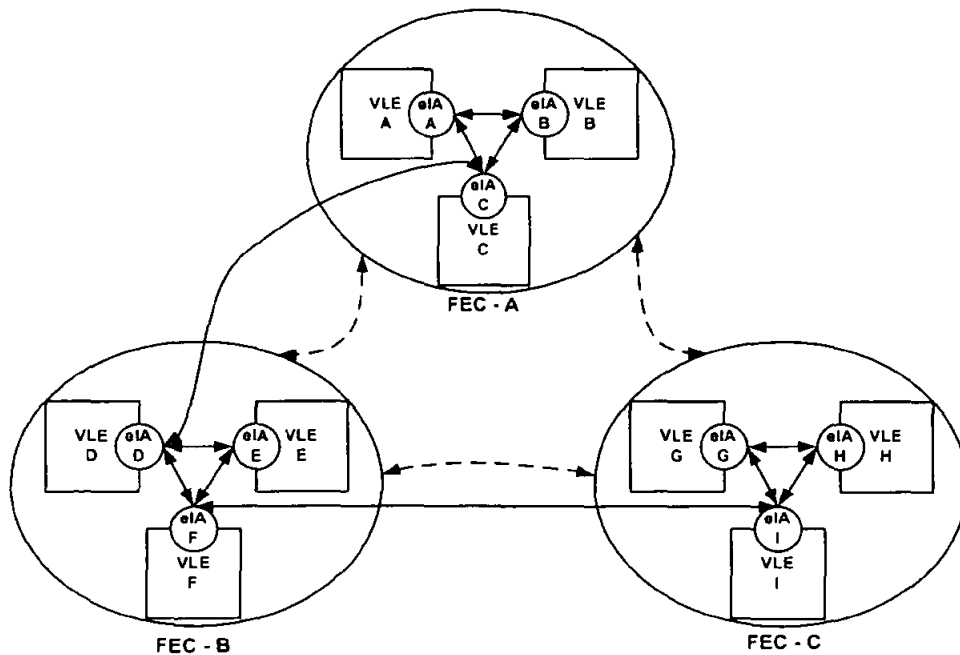


Figure 5-19: Extended Federated e-learning Community

### 5.2.1 Message Broadcasting

Currently, message broadcasting only applies to learning objects and user profiles functionalities. Section 4.2 has elaborated the broadcast service algorithm. Figure 5-20 gives an example of message broadcasting service flow. However, eIA have to abide some rules in broadcasting the message.

The rules are as follows:

1. In message broadcasting, eIAs are distinguished into three groups:
  - eIA Client: original service requestor, e.g. eIA-A.
  - eIA Server: eIA which receives the request and sends the response back to eIA Client or eIA intermediary, e.g. eIA-B, eIA-E, eIA-G, and eIA-H.
  - eIA Intermediary: eIA which acts like eIA server and also has role to route back the response received from the relayed eIA server, e.g. eIA-C, eIA-D, eIA-F, and eIA-I.

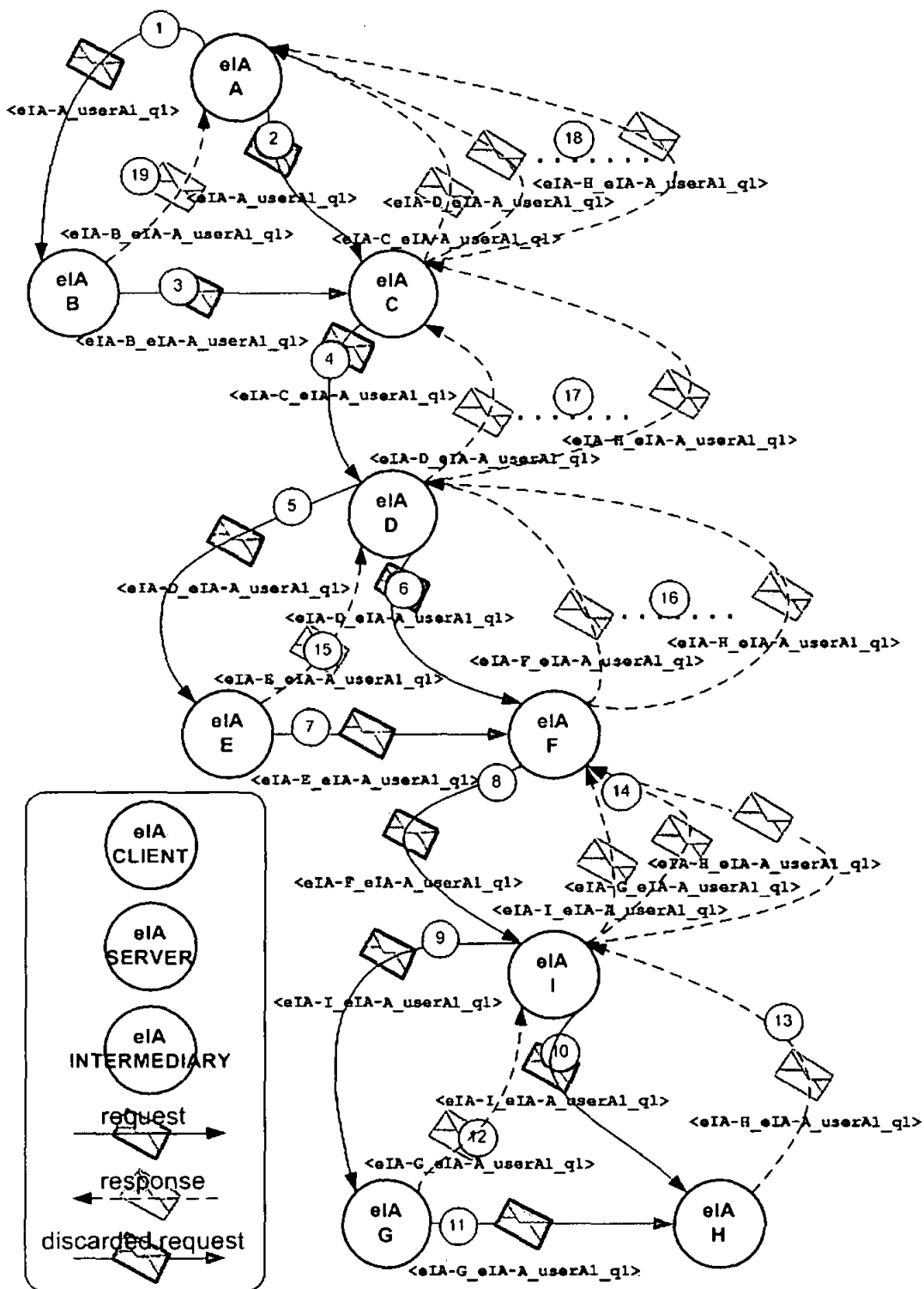


Figure 5-20: FeC Messages Broadcasting Flow Example

2. The message ID naming rules is as follows:
  - <eIAClientID\_userID\_queryID>: request message ID from eIA client. Message flows from eIA client to either eIA server or eIA intermediary, e.g. message number 1 and 2.
  - <eIAIntermediaryID\_eIAClientID\_userID\_queryID>: request message ID from eIA intermediary. Message flows from eIA intermediary to either another eIA server or eIA intermediary, e.g. message number 4, 5, 6, 8, 9, and 10.
  - <eIAServerID\_eIAClientID\_userID\_queryID>: response message ID from eIA server. Message flows from eIA server to either directly back to eIA client or through eIA intermediary. Example of directly back to eIA client is messages number 18 and 19. Example of back through eIA intermediary is message number 12, 13, 14, 15, 16, and 17.
  
3. There are also discarded relay messages, e.g. message number 3, 7, and 11. eIA-C, eIA-F, and eIA-H discards the messages received because they have received the same request messages from either eIA client or other eIA intermediary, e.g. eIA-A, eIA-D, and eIA-I.

### 5.3 Summary

eIA prototyping example into Moodle and ATutor can be taken as guidelines to implement the model into other platforms. One important effort in integrating eIA to other platforms is the availability of XSLT template modules. The modules must be provided depending on the platform structure, i.e. database structure and availability of pre-existent functions. From the implementation of eIA inside every VLEs, Federated e-learning Community (FeC) can be engaged. The limitation of the FeC itself is only the web services accessibility.

## CHAPTER SIX : CONCLUSION AND RECOMMENDATIONS

This final chapter is organized into two sections. The first section provides a conclusion of the research results. The second section provides some recommendations for further work.

### 6.1 Conclusion

The thesis evaluated the proposed e-learning Interoperability Agent model by prototyping the model into Moodle and ATutor. Throughout the work and from the prototyping results, some advantages/benefits and disadvantages/drawbacks of the model can be addressed.

The advantages are as follows:

- **Interoperability**  
Service-oriented approach with its web service and standards sets of protocols (i.e. HTTP, XML, and SOAP) enables the interoperability in the various web-based environments and others environments which have the Internet accessibility. The model and algorithm within can be implemented in any programming environment platforms.
- **Extensibility**  
The framework provides extensible environment where administrator can customize or grow the services comfortably without interrupting the pre-existent VLE core architecture. New educational web services can be added and published by using the WSDL to enable other eIAs in utilizing the services. Other eIAs can also adapt and provide the same services in their own systems.
- **Flexibility**  
Agents are always flexible as they can move in a network (FeC) to find the requested information (LOs or users).

- Scalability

Through its flexibility, the use of VLE is extended, not only to be used within the VLE's institutions, but also connected to other institutions. The connectivity itself depends on the accessibility of web services. It means that the connectivity can be expanded as long as one can access the web services.

The limitations are as follows:

- One has to do some efforts in implementing the model into their system, especially if the system does not provide ready-to-use metadata functionality. One of the important efforts in here is exploring the data representation used by the system and how to utilize the data to be used by eIA. Different platforms need their own techniques in implementing the model.
- eIA depends on the Internet as the communication infrastructures. Thus, e-learning systems without the Internet accessibility can not utilize the model.

## 6.2 Recommendations for Further Works

There are number of challenges need to be pondered:

- The current implementation of the model is only tested on two e-learning system platforms: Moodle and ATutor, with only the basic necessary functionality implemented. It means that the reliability and maturity of the model is not evaluated yet in the real e-learning environment. Thus, a further implementation on various existing e-learning systems can carry out deeper analysis on the model and also how effective the usefulness of FeC.
- This work has not covered and analyzed security issues of the data or messages exchange between eIAs. If a client sends SOAP requests to a server, can eIA ensure that the communication remains confidential? There are some solutions can be applied to tackle these issues. Two of the possible solutions are SSL (Secure Sockets Layer) and W3C XML Encryption Standard. SSL is a proven encryption technology, widely deployed, and a viable option for encrypting messages. Meanwhile W3C standard provides a framework for

encrypting and decrypting entire XML documents or just portions of an XML document, and it is likely to receive widespread industry support. Thus, further research needed to address these issues.

- The metadata, either learning object metadata and learner information metadata, is always an issue. There are always changes in the metadata schema following the current needs. Thus, the semantic integration schema used in this work might be changing and also might be added by new schema for extended functionalities.

All in all, this work is a starting point to reach two goals. First goal is the creation of boundless educational collaboration through e-learning systems and eIA. Second goal is the generalization of eIA model so that other domains can also use the model.

## REFERENCES

- ABITEBOUL, S., BUNEMAN, P., and SUCIU, D. (2000). Data on the Web – From Relations to Semistructured Data and XML. *Morgan-Kaufman Publisher*.
- ARAPI, P., MOUMOUTZIS, N., MARAGOUDAKIS, Y., and CHRISTODOULAKIS, C. (2005). Extending Existing E-Learning Platforms to Support Automatic Conversion of Learning Content to SCORM. *In the proceedings of the Joint KNOSOS-CHIRON Workshop “e-Learning Solutions – On the Way to Ubiquitous Applications”*. Sandanski, Bulgaria.
- ARIADNE. (2001). ARIADNE Foundation for the European Knowledge Pool. Available online at: <http://www.ariadne-eu.org/> (accessed July 2007).
- AROYO, L., DOLOG, P., HOUBEN, G.J. KRAVCIK, M., NAEVE, A., NILSSON, M., and WILD, F. (2006). Interoperability in Personalized Adaptive Learning. *Educational Technology and Society*, 9(2): p.4-18.
- ATHANASOPOULOS, G., TSALGATIDOU, A., PANTAZOGLU, M. (2006). Interoperability among Heterogeneous Services. *In Proceedings of the IEEE International Conference on Services Computing*, p.174-181.
- ATRC. (2004). ACollab: Accessible Collaboration Environment. The Adaptive Technology Resource Centre (ATRC), University of Toronto. Available online at: <http://www.atutor.ca/acollab> (accessed July 2007).
- ATUTOR. (2007). Accessible Tutor Learning Content Management System. Available online at: <http://www.atutor.ca/> (accessed July 2007).
- AYALA, D.. (2002). NuSOAP – Web Services Toolkit for PHP. Available online at: <http://dietrich.ganx4.com/nusoap> (accessed July 2007).



- BELFER, K., NESBIT, J.C., ARCHAMBAULT, A., and VARGO, J. (2003). Learning Object Review Instrument (LORI) Version 1.5. Available online at: <http://www.elera.net/eLera/Home/Articles/LORI%201.5.pdf>.
- CHAN, T., SHARPLES, M., VAVOULA, G., and LONSDALE, P. (2004). Educational Metadata for Mobile Learning. In *Proceeding of the 2<sup>nd</sup> IEEE International Workshop on Wireless and Mobile Technologies in Education (WMTE)*.
- CIOCOIU, M., GRUNINGER, M., and NAU, D. (2001). Ontologies for integrating engineering applications. *Journal of Computing and Information Science in Engineering*, 1: p.45-60.
- COLLIER, G., and ROBSON, R. (2002). E-Learning Interoperability Standards. *Sun Microsystems White Paper*.
- DCMI. (1999). Dublin Core Metadata Element Set, Version 1.1: Reference Description. Dublin Core Metadata Initiative. Available online at: <http://dublincore.org/documents/1999/07/02/dces/>.
- ECLIPSE. (2007). Virtual Whiteboard. Eclipse Solutions. Available online at: <http://www.virtual-whiteboard.co.uk/> (accessed July 2007).
- EUZENAT, J. (2001). Towards a principled approach to semantic interoperability. *Proceeding IJCAI workshop on Ontologies and Information Sharing*, p.19-25. Seattle (WA US).
- GUTH, S., NEUMANN, G., and SIMON, B. (2001). UNIVERSAL – Design Spaces for Learning Media. In *Proceedings of the 34<sup>th</sup> Hawaii International Conference on System Sciences*. USA.
- HAMMER, J., and McLEOD, D. (1993). An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database

- Systems. *International Journal of Intelligent and Cooperative Information Systems*, p.51-83.
- HL7. (2007). Health Level Seven. Available online at: <http://www.hl7.org/>. (accessed July 2007).
- HUD. (2000). coMentor. Learning & Teaching Support Unit, University of Huddersfield. Available online at: <http://comentor.hud.ac.uk/> (accessed July 2007).
- IEEE. (1990). IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York.
- IEEE LTSC. (2002). Draft Standard for Learning Object Metadata. IEEE 1484.
- IEEE PAPI. (2002). Personal and Private Information Learner Specification.
- IMS LIP. (2003). Learner Information Package Specification v1.0. Available online at: <http://www.imsglobal.org/profiles>.
- IMS LOM. (2002). IMS Learning Resource Meta-data Specification Version 1.3. Available online at <http://www.imsglobal.org/metadata>.
- KASHYAP, V., and SHETH, A. (1996). Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies. *Cooperative Information Systems*.
- LEE, G. (2005). A Web-Service-Based E-Learning Service Infrastructure for Achieving Dynamic and Collaborative E-Learning. *Doctoral Dissertation*. Department of Computer and Information Science and Engineering, University of Florida.
- LIU, X., EL SADDIK, A., and GEORGANAS, N.D. (2003). An Implementable Architecture of An E-Learning System. *IEEE CCECE*, Canada: p.717-720.

- LIU, J., and VINCENT, M. (2003). Query Translation from XSLT to SQL. *In proceedings the 7<sup>th</sup> IDEAS*, p.87-96.
- LUO, J., LI, W., CAO, J., and GE, L. (2006). Integrating Heterogeneous E-learning Systems. *In Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services*, p.9.
- MADHOUR, H., FORTE, M.W., and FERNANDES, E. (2006). Semantic learning model and extended student model: towards an AHAM-based adaptive system. *In Proceedings of the 5<sup>th</sup> IASTED International Conference on Web-Based Education*, p.349-354. Puerto Vallarta, Mexico.
- MOODLE. (2007). Modular Object-Oriented Dynamic Learner Environment. Available online at: <http://moodle.org> (accessed July 2007).
- NAJJAR, J., DUVAL, E., TERNIER, S., and NEVEN, F. (2003). Towards Interoperable Learning Object Repositories: The ARIADNE Experience. *In Proceedings IADIS International Conference on WWW/Internet*, vol.1: p. 219-226.
- ONO, K., KOYANAGI, T., ABE, M., and HORI, M. (2002). XSLT Stylesheet Generation by Example with WYSIWYG Editing. *In proceedings of the 2002 Symposium on Applications and the Internet*, IEEE.
- PANKRATIUS, V., SANDEL O., and STUCKY, W. (2004). Retrieving Content with Agents in Web Service E-Learning Systems. *IFIP Conference on Artificial Intelligence on Applications and Innovations*, France.
- PHPMYADMIN. (2006). phpMyAdmin 2.8.1. Available online at: <http://www.phpmyadmin.net/> (accessed July 2007)

- SAHUGUET, A., and AZAVANT, F. (1999). Looking at the Web through XML Glasses. *CoopIS*, p.148-159. Edinburgh, Scotland.
- SHETH, A., RAMAKRISHNAN, C., and THOMAS, C. (2005). Semantics for the Semantic Web: The Implicit, the Formal and the Powerful. *International Journal on Semantic Web and Information Systems*, 1(1): p.1-18.
- SIMON, B., RETALIS, S., and BRANTNER, S. (2003). Building Interoperability among Learning Content Management Systems. In *Proceedings of The Twelfth International World Wide Web Conference*. Budapest, Hungary.
- STOCKLEY, D. (2003). E-learning Definition and Explanation. Available online at: <http://derekstockley.com.au/learning-definition.html> (accessed July 2007).
- USCHOLD, M., and GRUNINGER, M. (2005). Architectures for Semantic Integration. In *Proceedings Dagstuhl Seminar*, 04391.
- VAN DEN BERG, KARIN. (2005). Finding Open Options: An Open Source software evaluation model with a case study on Course Management Systems. *Master Thesis*, Information Management and Science Program, Tilburg University.
- W3C. (2002). Web services architecture requirements. W3C Web Services Architecture Working Draft. Available online at: <http://www.w3.org/TR/2002/> (accessed July 2007).
- XIANG, X., SHEN, Z., GUO, L., SHI, Y. (2003). Introduction of the Core Elements Set in Localized LOM Model. *Learning Technology*, 5, IEEE Computer Society.

## PUBLICATIONS

1. Dicky Ekklesia and Azween Abdullah: Semantically Federated e-Learning Community through eIA. In *Proceedings of Regional Conference on Computational Science and Technologies (RCCST'07)*. Kota Kinabalu, Malaysia, November 29 -30, 2007.
2. Dicky Ekklesia and Azween Abdullah: Semantically Integrated Interoperability Agent. In *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education (E-Learn'07)*. Quebec City, Canada, October 15-19, 2007, AACE (Association for the Advancement of Computing in Education).
3. Dicky Ekklesia and Azween Abdullah: E-Learning Interoperability Agent Based on Semantic Integration. In *Proceedings of Conference on Information Technology Research & Application (CITRA'07)*. Subang, Malaysia, April 4-5, 2007.

## APPENDIX A : EIA METADATA SCHEMA

### A.1 eIA Learning Object Metadata

Table A-1: eIA Learning Object Metadata

Level 1	Level 2	Level 3	Level 4	Size	Description	Data Type	Example / Vocabulary		
1	General			M	1	This category groups the general information that describes the learning object as a whole.	-	-	
	Identifier			O	>=1	A globally unique label that identifies the learning object.	-	-	
		Catalog			M	1	The name or designator of the identification or cataloguing scheme for the entry. A namespace scheme.	String	"ISBN", "ARIADNE", "URI"
		Entry			M	1	The value of the identifier within the identification or cataloguing scheme that designates or identifies the learning object. A namespace specific string.	String	"2-7342-0318", "LEAO875", "http://elearning.utp.edu.my/view.php?1234"
	Title				M	1	Name / title given to the learning object.	(Langtype, String)	("en", "Using eIA")
	Language				M	>=1	The primary human language or languages used within the learning object to	String (Language Tag)	"en", "en-GB", "es_ES", "de", "fr_CA", "it"

Level 1	Level 2	Level 3	Level 4	Size	Description	Data Type	Example / Vocabulary	
					communicate to the intended user.			
	Description	M		1	A textual description of the content of the learning object.	(Langtype, String)	("en", "How to use eIA")	
	Keyword	M		>=1	A keyword or phrase describing the topic of the learning object.	(Langtype, String)	("en", "Learning Management System")	
2	Life Cycle	O		1	This category describes the history and current state of the learning object and those entities that have affected the learning object during its evolution.	-	-	
	Contribute	M		>=1	Those entities (i.e., people, organizations) that have contributed to the state of the learning object during its lifecycle.	-	-	
			Role	M	1	Kind of contribution	String	"author", "publisher", "unknown"
			Entity	M	1	The identification of and information about entities (i.e. people, organizations) contributing to the learning object. The entities shall be ordered as most relevant first.	String (vCard converted to String, data type can be varies)	"BEGIN:VCARD FN:Dicky Ekklesia NICKNAME:de END:VCARD"
			Date	M	1	The date of the contribution.	Date	"2002-10-28"
	Version	O		1	The edition of the learning object.	(Langtype, String)	("en", "1.1 alpha")	

Level 1	Level 2	Level 3	Level 4	Size	Description	Data Type	Example / Vocabulary
	Status O			1	The completion status or condition of the learning object.	String	"draft", "final", "revised", "unavailable"
3	Meta-metadata O			1	Data about the metadata.	-	-
	Contribute M			>=1	Those entities (i.e., people, organizations) that have contributed to the metadata of the learning object.	-	-
		Role M		1	Kind of contribution	String	"author", "publisher", "unknown"
		Entity M		1	The identification of and information about entities (i.e. people, organizations) contributing to the metadata of the learning object. The entities shall be ordered as most relevant first.	String (vCard converted to String)	"BEGIN:VCARD FN:Dickie Ekklesia NICKNAME:de END:VCARD"
		Date M		1	The date of the metadata contribution	Date	"2002-10-28"
	Metadata Schema O			1	Note / version of the metadata schema.	String	"elAv1.0"
	Language O			>=1	The language of the metadata, called as Langtype and used as data type for other elements.	String	"en", "en-GB", "es_ES", "de", "fr_CA", "it"
4	Technical M			1	This category describes the technical requirements and characteristics of the learning object.	-	-



Level 1	Level 2	Level 3	Level 4	Size	Description	Data Type	Example / Vocabulary
	Format	M		1	Technical datatype(s) of (all the components of) the learning object. This data element shall be used to identify the software needed to access the learning object.	String	"video/mpeg", "document/pdf", "document/doc", "presentation/ppt", "text/html", "text/txt", "worksheet/xls", "any"
	Size	O		1	The size of the digital learning object in bytes (can be automatically calculated). The size is represented as a decimal value. The unit in bytes.	Decimal	"2,000"
	Location	M		1	A string that is used to access the learning object. It may be a location (e.g. URL), or a method that resolves to location (e.g. URI).	String	"http://localhost"
	Installation Remarks	O		1	Description how to install or use the learning object.	(Langtype, String)	("en", "Microsoft Word"), ("en", "unzip the zip file and launch the index.html in your browser")
5 Educational		M		1	This category describes the key educational or pedagogic characteristics of the learning object.	-	-
	Learning Resource Type	O		>=1	Specific kind of learning object. The most dominant kind shall be first.	String	"exercise", "simulation", "questionnaire", "diagram", "video"
	Intended End User Role	M		>=1	Principal user(s) for which the learning object was designed, most dominant first.	String	"teacher", "author", "learner", "manager", "unspecified"

Level 1	Level 2	Level 3	Level 4	Size	Description	Data Type	Example / Vocabulary
	Context	M		>=1	The principle environment within which the learning and use of the learning object is intended to take place. The most relevant first.	String	"K12", "high school", "higher education", "training", "informal", "unspecified"
6 Rights				1	This category describes the intellectual property rights and conditions of use for the learning object.	-	-
	Cost	O		1	Whether use of the learning object requires payment.	String	"yes", "no"
	Copyright and Other Restriction	O		1	Whether copyright or other restrictions apply to the use of the learning object.	String	"yes", "no"
	Description	O		1	Comments on the conditions of use of the learning objects.	{Langtype, String}	("en", "Use of this learning object is only permitted after written permit from its author")
7 Relation				>=1	This category defines the relationship between the learning object and other learning object if any.	-	-
	Kind	M		1	Nature of the relationship between the learning object and the target learning object, identified by resource.	String	"is part of", "has part", "is version of", "has version", "is format of", "has format of", "references", "is referenced by", "is based on", "is based for", "requires", "is required by"
	Resource	M		1	The target learning object that this relationship references.	-	-
		Identifier	M	>=1	A globally unique label that identifies the target learning	-	-

Level 1	Level 2	Level 3	Level 4	Size	Description	Data Type	Example / Vocabulary	
					object.			
			Catalog	M	1	The name or designator of the identification or cataloguing scheme for the entry. A namespace scheme.	String	"ISBN", "ARIADNE", "URI"
			Entry	M	1	The value of the identifier within the identification or cataloguing scheme that designates or identifies the target learning object. A namespace specific string.	String	"2-7342-0318", "LEAO875", "http://localhost/view.php?1234"
		Description	M		1	Description of the target learning object.	(Langtype, String)	("en", "Using eIA in French")
8	Annotation			O	>=1	This category provides comments on the educational use of the learning object, and information on when and by whom the comments were created.	-	-
		Entity	M		1	Entity (i.e. people, organization) that created this annotation.	String (vCard?)	"BEGIN:VCARD FN:Dicky Ekklesia NICKNAME:de END:VCARD"
		Date	M		1	Date that this annotation was created.	Date	"2002-10-28"
		Description	M		1	The content of this annotation.	String	"I really enjoy using eIA"
		Rating	M		1	Quality rating of the learning object. Specified in decimal value: 1 - 5 (1: not recommended, 2: so so but not recommended, 3: so so, 4:	Decimal	"5"

Level 1	Level 2	Level 3	Level 4	Size	Description (good, 5: recommended)	Data Type	Example / Vocabulary
9	Classification			>=1	This category describes where this learning object falls within a particular classification system.	-	-
	Purpose	M		1	The purpose of classifying the learning object.	String	"discipline", "idea", "prerequisite", "educational objective", "accessibility", "restrictions", "educational level", "skill level", "security level", "competency"
	Taxon Path	M		1	A taxonomic path in specific classification system. Each succeeding level is a refinement in the definition of the preceding level.		
		Source	O	1	The name of the classification system. This data element may be use any recognized "official" taxonomy or any user-defined taxonomy.	String	("en", "ACM"), ("en", "ARIADNE")
		Taxon	M	1	A particular term within taxonomy. A taxon is a node that has a defined label or term. A taxon may also have an alphanumeric designation or identifier for standardized reference. Either or both the label and the entry may be	-	-

Level 1	Level 2	Level 3	Level 4	Size	Description	Data Type	Example / Vocabulary
					used to designate a particular taxon.		
			Identifier	O 1	The identifier of the taxon, such as a number or letter combination provided by the source of the taxonomy.	String	"320", "BF180", "4.3.2"
			Entry	M 1	The textual label of the taxon.	(Langtype, String)	("en", "Computer and Information Sciences")
			Description	M 1	Description of the learning object relative to the classification.	(Langtype, String)	
			Keyword	M >=1	Keywords and Phrases descriptive of the learning object relative to the classification.	(Langtype, String)	

**A.2 eIA Learner Information Metadata**

**Table A-2: eIA Learner Information Metadata**

Level 1	Level 2	Level 3	Level 4	Size	Description	Data Type	Example/Vocabulary
1 Profile	M			1	Static information about the learner.	-	-
	Identification	M		1	Personal information such as name, address, contact info, and gender.	-	-
		Name	M	M 1	The detailed/full name of the learner.	String	

Level 1	Level 2	Level 3	Level 4	Size	Description	Data Type	Example/Vocabulary
		Address O		1	The detailed address of the individual.	-	-
			PO Box	1	Post Office Box Number	String	
			Street O	1	The necessary street part of the address, including the street number, etc.	String	
			City O	1	City part of the address.	String - as per ISO standard.	
			Country O	1	Country part of the address.	String - as per ISO standard.	
			StatePr O	1	The State / Province part of the address.	String - as per ISO standard.	
			PostCode O		The postcode part of the address.	String - as per ISO standard.	
		Contactinfo M		1	The detail contact information of the individual.	-	-
			Telephone O	>=1	The telephone number, including country and area code	String	
			Mobile O	>=1	The mobile number, including country code and area code.	String	
			Email M	>=1	Email address.	String	
			Web O	>=1	Web address defined as the URL.	String	
		Gender O		1	The gender of the learner.	String	"Male", "Female", "Unknown"
	Accessibility M			1	Learner accessibility issues for language, disability, preferences, and eligibility.	-	-
		Language M		>=1	The language reading, writing and speech capabilities of the learner.	-	-
			Langtype M	1	The language used for the language proficiency.	String - as per ISO standard.	

Level 1	Level 2	Level 3	Level 4	Size	Description	Data Type	Example/Vocabulary
			Proficiency M	>=1	The language proficiency of the learner, i.e. oral, reading, writing.	String	"oral", "reading", "writing"
	Interest O			>=1	Interest of the learner.	-	-
		Type O		1	Type of interest.	String	
		Subject Area M		>=1	Related Area.	String	
		Description O		1	The description of the interest.	String	
	Relationship O			>=1	The relationship to be established between the other learners.	-	-
		Type O		1	The type of relationship.	String	
		User Identifier M		1	Unique identifier of the related learner.	String	
		Description O		1	Description of the relationship.	String	
2 Pedantic M				1			
	Competency M			1	Acquired learning competencies, at least one of the user competency element must be filled in.	-	-
		QCL O		>=1	Received qualifications, certifications and licenses (for activities that have been completed).	-	-
			Title O	1	The title of the qualification, certification or license.	String	
			Organization O	1	The organization responsible for the awarding of the qualification, certification or license.	String	
			Level M	1	The level/grade of the QCL.	String	
			Date O	1	Recorded dates appropriate to the qualification, certification or license.	Date	
			Description O	1	The description of the qualification, license or certification or license.	String	

Level 1	Level 2	Level 3	Level 4	Size	Description	Data Type	Example/Vocabulary
		Subject Area	M	>=1	Subject area related to the QCL.	String	
		Activity	O	>=1	A learning activity that learner performs.	-	-
		Name	M	1	A title for the activity for quick reference.	String	
		Status	O	1	The progress status of an activity (in percentage).	0 - 100%	
		Date	O	1	Start timeline or and End timeline.	Date	
		Objectives	O	1	The objective of the specific activity.	String	
		Subject Area	M	>=1	The subject of the specific activity	String	
		Description	M		The description of the activity itself.	String	
		Experiences	O	>=1	The experience related to the learner.	-	-
		Type	M	1	The type of education, training, vocational, service, etc.	String	
		Date	O	1	Recorded dates appropriate to the activity.	Date	
		Outcome	M	1	The outcome of the experience.	String	
		References	O	>=1	A reference for the learner by someone associated with this experience, including referee name, date, and the testimonial itself.	String	
		Subject Area	M	>=1	Subject area related to the experiences.	String	
	Knowledge Repository		O	>=1			
		Type	M	1	Type of the knowledge.	String	"document"
		Description	M	1	Description of the knowledge.	String	
		Entry	M	1	The value of the knowledge, e.g. URL to retrieve it, etc.	String	



## APPENDIX B : EIA CODE INFORMATION

This appendix provides further information on eIA prototype implementation as elaborate in section 5.1. At several points in this appendix, snippets of pseudo code are included. It has to be remarked that these pieces of code are highly simplified specifications of the real code. This appendix intends to give more description on the prototype implementation without going in-depth programming details of the software code.

### B.1 eIA Pilot Prototype Structure

Figure B-1 draws the diagram of eIA pilot structure in Moodle and ATutor. eIA\_gateway, WSDL, and eIA\_ws\_core (with its service directory) modules are the eIA communication layer. eIA\_xslt\_lib and eIA\_data\_core modules are the eIA data manipulation layer. eIA\_data\_core module interacts with the Moodle's and ATutor's database system (DS). eIA\_ui\_core and eIA\_ui\_frontEnd modules are the eIA user interface layer. eIA\_ui\_frontEnd delivers the interface through the web page of Moodle and ATutor systems.

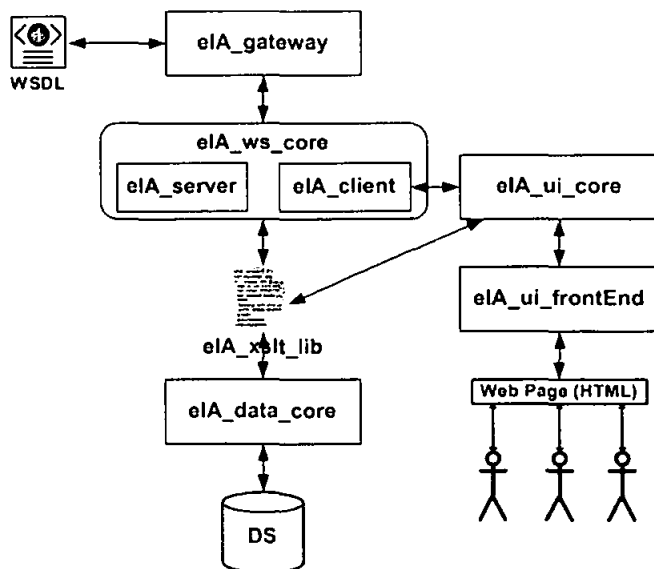


Figure B-1: eIA Prototyping Diagram in Moodle and ATutor

### B.1.1 eIA Web Services Core (eIA\_ws\_core)

eIA\_ws\_core is the most important components whereas eIA interacts with other eIA, eIA interacts with VLE system through eIA\_data\_core, and eIA interacts with users through eIA\_ui\_core. There are two classes of eIA\_ws\_core: eIA\_server and eIA\_client. Table B-1 illustrates a skeleton for eIA server class. Meanwhile Table B-2 illustrates a skeleton for eIA client class.

Table B-1: A Skeleton Code for the eIA Server Class

```
// #require_once
// include all the needed configurations and libraries

/**
 * eIAServer Class
 *
 * @package eIA
 * @subpackage WSLibrary
 * @version $Id: eIA_server.php,v 1.0 2007/12/01 dekklesia
 * @access public
 */
class eIAServer {
    // class attributes
    var $server; // soap server object

    /**
     * eIAServer::eIAServer()
     */
    function eIAServer ($HTTP_RAW_DATA)
    {
        // create the server instance
        // check for any error
        // invoke the requested service
    }

    /**
     * eIAServer::__destruct()
     */
    function __destruct ()
    {
        // destroy client's session
    }

    /**
     * eIAServer::init()
     *
     * initializes a connection to a client by generating a random session key
     * to be used for communications with this specific client.
     *
     * @param mixed $client
     * @return mixed
     */
    function init ($client)
    {
        // authenticate client
        // check for client's session
        // if no session or session expired, then create session for client
        // return session data to client
    }
}
```

```

)

/**
 * eIAServer::eIAService($inputArray)
 *
 * eIA specific service implementation
 *
 * @param mixed $inputArray // input data to be processed by the server
 * @return
 */
function eIAServiceName ($inputArray)
{
    // prepare input data
    // convert input data through data library
    // execute the converted data
    // convert results to eIA standardized data
    // call broadcasts if needed
    // combine local results + broadcast results
    // return results
}

/**
 * eIAServer::error()
 *
 * sends an error response back to the client
 *
 * @param string $msg
 * @return mixed A Soap Fault
 */
function error ($msg)
{
    return new soap_fault ('Client', '', $msg);
}

/**
 * eIAServer::broadcast()
 *
 * @param mixed $inputArray
 * @param mixed $service
 * @return mixed
 */
function broadcast ($inputArray, $service)
{
    // check for any redundant request
    // if not redundant then check for constraint(s)
    if (constraint is not violated) {
        // update request ID in input array
        // get eIA data from eIA service directory
        // broadcast the request to all the registered eIA
        // return the broadcast results
    }
}
) // end of eIAServer class

```

Table B-2: A Skeleton Code for the eIA Client Class

```

// #require_once
// include all the needed configurations and libraries

/**
 * eIAClient Class
 *
 * @package eIA
 * @subpackage WSLibrary
 * @version $Id: eIA_client.php,v 1.0 2007/12/01 dekklesia
 */
class eIAClient {

```

```

// class attributes
var $sessionKey; // session key from one particular server to communicate
var $client; // soap client object
var $dataArray; // input data to be sent to server
var $result; // results received from server

/**
 * eIAClient::eIAClient()
 *
 * @param string $accessPoint
 */
function eIAClient ($accessPoint) {
    // create the client instance - initializes a connection to server
    // check for any error
}

/**
 * eIAClient::invoke_service()
 *
 * invoke a server service
 * Pre-condition: dataArray has been set-up before calling this function.
 *
 * @param string $request
 */
function invoke_service ($request){
    // invoke the server service ($request)
}
} // end of eIAClient class
?>

```

### B.1.2 eIA Data Mediation Module (eIA\_xslt\_lib)

There are two core XSLT library used by eIA in processing the services: service schema and service mapper. Service schema, as illustrated in Table B-3, is used to describe VLE specific database function with its needed parameter and response schema of particular VLE database structure. Meanwhile service mapper, as illustrated in Table B-4, is used to map eIA client request data with eIA server schema described in service schema.

**Table B-3: A Skeleton Code for the eIA Service Schema**

```

<?xml version="1.0" encoding="UTF-8"?>
<eIAServiceResults>

    <action>VLEspecificFunction</action>

    <requestparameter>parameter required for the function</requestparameter>

    <requestdata>
        List all of data mapping to request parameter
    </requestdata>

    <responseparameter>
        Response object schema
    </responseparameter>

</eIAServiceResults>

```

**Table B-4: A Skeleton Code for the eIA Service Mapper**

```

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:output method="text"/>
  <xsl:template match="eIAServiceResults">

    <xsl:apply-templates select="requestdata" />

    $result=<xsl:value-of select="action"/>(<xsl:value-of
select="requestparameter"/>);

  </xsl:template>

  <xsl:template match="requestdata">
    Map eIA request data to VLE specific parameter
  </xsl:template>
</xsl:stylesheet>

```

### B.1.3 eIA Query GUI Module (eIA\_ui\_frontEnd and eIA\_ui\_core)

eIA Query GUI Module is built up by eIA\_ui\_frontEnd and eIA\_ui\_core. eIA\_ui\_frontEnd renders the front end page for the user interaction and passes up user's request to eIA\_ui\_core. eIA\_ui\_core processes the request by cooperating either with eIA\_ws\_core or eIA\_data\_core.

**Table B-5: A Skeleton Code for the eIA UI Front End**

```

/**
 * eIA_ui_frontEnd.php
 *
 * @package eIA
 * @subpackage UILibrary
 * @version $Id: eIA_ui_frontEnd.php,v 1.0 2007/12/01 dekklesia
 */

// #require_once
// include all the needed configurations and libraries

// #require_param
// check and retrieve for all needed parameters

// print page header

if (file_exists ($eIA_CFG->blocksdir.'/lang/menus.php')) {

  include $eIA_CFG->blocksdir.'/lang/menus.php';
  //print content header
  // include handler of the specific requested page
  include './UILibrary/'.$handler[$mid];

} else
  // error message

// print page footer

```

Table B-6: A Skeleton Code for the eIA UI Core

```

/**
 * eIA_ui_core.php
 *
 * @package eIA
 * @subpackage UILibrary
 * @version $Id: eIA_ui_core.php,v 1.0 2007/12/01 dekklesia
 */

// #require_once
// include all the needed configurations and libraries

// #require_param
// check and retrieve for all needed parameters

// process user's request:
// - get eIA data from eIA service directory
// - prepare data to be sent to server
// - create eIAClient object and invoke the server
// - receive response from server
// - process the results
// - print page header
// - print results
// - print page footer

```

## B.2 eIA Code Statistics

The following statistics were collected from the 1/12/2007 snapshot of the Moodle's eIA code.

### B.2.1 Mods Package

#### ./moodle/eIA (eIA Core)

Filename	Size (bytes)	Source Code Lines		
		File	Code	Comment
eIA_client.php	3299	127	78	41
eIA_gateway.php	664	30	15	9
eIA_config.php	483	18	10	5
eIA_mod_config.php	171	12	4	5
eIA_wsCore.php	22181	617	400	168
eIA_download.php	1265	44	30	7
wsdl.php	8791	228	216	9
<b>TOTAL</b>	<b>36854</b>	<b>1076</b>	<b>753</b>	<b>244</b>

**./moodle/eIA/WSLibrary (Web Services Library)**

<i>Filename</i>	<i>Size (bytes)</i>	<i>Source Code Lines</i>		
		<i>File</i>	<i>Code</i>	<i>Comment</i>
nusoap.php	232057	6609	4461	1802
searchUPResult.xsl	1137	26	26	0
searchLOResult.xsl	742	25	25	0
<b>TOTAL</b>	<b>233936</b>	<b>6660</b>	<b>4512</b>	<b>1802</b>

**./moodle/eIA/DataLibrary (Data Library)**

<i>Filename</i>	<i>Size (bytes)</i>	<i>Source Code Lines</i>		
		<i>File</i>	<i>Code</i>	<i>Comment</i>
up_schema.xml	1102	42	42	0
lo_schema.xml	815	31	31	0
up_mapper.xsl	580	12	12	0
lo_mapper.xsl	631	14	14	0
eIA_xml_parser.php	5564	186	117	60
eIA_xslt_parser.php	2746	123	57	56
eIA_dataLibrary.php	7058	228	144	75
<b>TOTAL</b>	<b>18496</b>	<b>636</b>	<b>417</b>	<b>191</b>

**B.2.2 Blocks Package****./moodle/blocks/eIA (eIA's User Interface Core)**

<i>Filename</i>	<i>Size (bytes)</i>	<i>Source Code Lines</i>		
		<i>File</i>	<i>Code</i>	<i>Comment</i>
eIA_blocks_config.php	232	13	4	6
eIA_ui_frontEnd.php	966	33	21	7
block_eIA.php	1454	57	32	22
eIA_ui_core.php	10924	273	248	20
<b>TOTAL</b>	<b>13576</b>	<b>376</b>	<b>305</b>	<b>55</b>

**./moodle/blocks/eIA/UILibrary (User Interface Library)**

<i>Filename</i>	<i>Size (bytes)</i>	<i>Source Code Lines</i>		
		<i>File</i>	<i>Code</i>	<i>Comment</i>
eIA_ui_lo_searchForm.php	3290	62	50	12
eIA_ui_up_searchForm.php	1282	36	26	7
eIA_ui_mp_mainPage.php	698	24	15	7
eIA_ui_services_list.php	916	25	17	7
eIA_ui_services_retrieve.php	1658	48	30	8
eIA_ui_services_WSDLRetriever.xsl	5508	140	140	0
eIA_ui_services_core.php	3552	99	76	10
eIA_ui_message_send.php	1472	50	39	8
eIA_ui_message_process.php	1881	55	39	8
eIA_ui_message_list.php	3195	83	71	10
eIA_ui_message_view.php	2125	55	45	8
<b>TOTAL</b>	<b>25577</b>	<b>677</b>	<b>548</b>	<b>85</b>

**./moodle/blocks/eIA/lang (Language Package)**

<i>Filename</i>	<i>Size (bytes)</i>	<i>Source Code Lines</i>		
		<i>File</i>	<i>Code</i>	<i>Comment</i>
countries.php	7519	246	242	1
languages.php	5254	198	189	8
mimetypes.php	1146	32	18	6
menus.php	537	24	14	7
<b>TOTAL</b>	<b>14456</b>	<b>500</b>	<b>463</b>	<b>22</b>