A Model-Driven Architecture based Evolution Method and Its Application in An Electronic Learning System

PhD Thesis

Yingchun Tian

Software Technology Research Laboratory

De Montfort University

October, 2012

To my husband, Delin Jing and
my mum, Ning Zhang
for their love and support

Declaration

I declare that the work described in this thesis was originally carried out by me during the period of registration for the degree of Doctor of Philosophy at De Montfort University, U.K., from October 2008 to December 2011. It is submitted for the degree of Doctor of Philosophy at De Montfort University. Apart from the degree that this thesis is currently applying for, no other academic degree or award was applied for by me based on this work.

Acknowledgements

For many years I had been dreaming about receiving a PhD, I would like to thank many people who helped me in achieving this dream in different ways since I undertook the work of this thesis.

My deepest gratitude goes to my supervisor, Professor Hongji Yang, for his guidance, support and encouragement throughout my PhD career. He always provided me with many invaluable comments and suggestions for the improvement of the thesis. I am grateful for his leading role fostering my academic, professional and personal growth.

Many thanks go to Professor Hussein Zedan and Doctor Wenyan Wu, for examining my PhD thesis and providing many helpful suggestions. My research career will benefit tremendously from the research methodologies to which Professor Zedan and Doctor Wu introduced me.

I would like to thank colleagues in Software Technology Research Laboratory at De Montfort University, for their support and feedback, and for providing such a stimulating working atmosphere, Professor Hussein Zedan, Doctor Feng Chen, Doctor Amelia Platt, Doctor Antonio Cau, and many other colleagues.

In addition, I would like to thank the Graduate School Office at De Montfort University for the outstanding management.

Finally, I wish to express thanks to my husband, Delin Jing, my mother, Ning Zhang, and my parents in law for their love, encouragements, patience, and support over the past years. This thesis is dedicated to them.

Abstract

Software products have been racing against aging problem for most of their lifecycles, and evolution is the most effective and efficient solution to this problem. Model-Driven Architecture (MDA) is a new technique for software product for evolving development and reengineering methods. The main steps for MDA are to establish models in different levels and phases, therefore to solve the challenges of requirement and technology change. However, there is only a standard established by Object Management Group (OMG) but without a formal method and approach. Presently, MDA is widely researched in both industrial and research areas, however, there is still without a smooth approach to realise it especially in electronic learning (e-learning) system due to the following reasons: (1) models' transformations are hard to realise because of lack of tools, (2) most of existing mature research results are working for business and government services but not education area, and (3) most of existing model-driven researches are based on Model-Driven Development (MDD) but not MDA because of OMG standard's preciseness.

Hence, it is worth to investigate an MDA-based method and approach to improve the existing software development approach for e-learning system. Due to the features of MDA actuality, a MDA-based evolution method and approach is proposed in this thesis. The fundamental theories of this research are OMG's MDA standard and education pedagogical knowledge. Unified Modelling Language (UML) and Unified Modelling Language Profile are hired to represent the information of software system from different aspects. This study can be divided into three main parts: MDA-based evolution method and approach research, Platform-Independent Model (PIM) to Platform-Specific Model (PSM) transformation development, and MDA-based

electronic learning system evolution. Top-down approach is explored to develop models for e-learning system. A transformation approach is developed to generate Computation Independent Model (CIM), Platform-Independent Model (PIM), and Platform-Specific Model (PSM); while a set of transformation rules are defined following MDA standard to support PSM' s generation. In addition, proposed method is applied in an e-learning system as a case study with the prototype rules support. In the end, conclusions are drawn based on analysis and further research directions are discussed as well. The kernel contributions are the proposed transformation rules and its application in electronic learning system.

Table of Contents

Table of Contents

List of Figures

List of Tables

List of Acronyms

ADL	*Advanced Distributed Learning*
AO	*Application Ontologies*
CBD	*Component-based Development*
CIM	*Computation Independent Model*
CORBA	*Common Object Request Broker Architecture*
CWM	*Common Warehouse Metamodel*
DSL	*Domain-Specific Language*
EARS	*ElectroAcoustic Resource Site*
EF	*Eclipse Foundation*
EJB	*Enterprise JavaBeans*
EMF	*Eclipse Modelling Framework*
GMF	*Graphical Modelling Framework*
J2EE	*Java 2 Platform Enterprise Edition*
LTSA	*Learning Technology Systems Architecture*
MDA	*Model-Driven Architecture*
MDA-SDLC	*MDA-based System Development Lifecycle*
MDE	*Model Driven Engineering*
MOF	*Meta Object Facility*
MVC	*Model-View-Controller*
NLP	*Natural Language Processing*
OMG	*Object Management Group*
OO	*Object-Oriented*
OOP	*Object-Oriented Programming*
PHP	*Hypertext Preprocessor*

PIM	*Platform-Independent Model*
PSM	*Platform-Specific Model*
QVT	*Query/View/Transformation*
RO	*Reference Ontologies*
SCORM	*Sharable Content Object Reference Model*
SECC	*Software Engineering Creative Computing*
SOA	*Service-Oriented Architecture*
SOAP	*Simple Object Access Protocol*
SPEM	*Software & Systems Process Engineering Metamodel*
UML	*Unified Modelling Language*
W3C	*World Wide Web Consortium*
WSDL	*Web Service Description Language*
XMI	*XML Metadata Interchange*
XML	*Extensible Markup Language*

Chapter 1
Introduction

Objectives

- To observe the need for the Mode-Driven Architecture based software evolution method
- To explain the research objectives and select the research method
- To discuss research questions and develop research propositions
- To highlight original contributions and define the measure of success
- To outline the organisation of the thesis

1.1 Proposed Research and Overview of Problem

The term “software evolution” is currently a popular and well-accepted one within both industry and research area, which implies that people have already noticed that software are facing a rapidly aging problem caused by many changes in requirement, maintenance, development, and so on.

A software evolution process is a combination of several software processes working on corresponding software. These software processes are interrelated in some ways. In software evolution process, there is a framework supporting evolution in software. Meanwhile, workflow is another presentation for software evolution.

Electronic learning, abbreviated as e-learning, is generally means all forms of electronic supported learning and teaching. It aims to effect the construction of knowledge with reference to individual experience, practice and knowledge of learner [100]. E-learning can be considered as an information and communication system, whether networked learning or not, serving as specific media to implement various learning processes [100]. In last decade, there are thousands e-learning industries have been established in worldwide. Obliviously, e-learning is taking an important role in current and future educational area.

E-learning system is mainly based on internet and multimedia technologies. Consulting, content, technologies, services and support are identified as the five key sectors of the e-learning industry [67]. Meanwhile, there are five parts are generally considered as the key elements of the e-learning system, which are content, display, navigation, evaluation, and communication [5, 42, 58, 65, 87, 97]. There are various standards proposed for e-learning system, such as the Sharable Content Object Reference Model (SCORM) which is a model for defining, packaging, and managing learning objects, and LTSA which is a framework specifying a high level architecture for information technology supported system [42]. However, there is no one standard process for e-learning system development.

Model-Driven Architecture (MDA) is a new technique for software product for evolving development and reengineering methods. The main steps for MDA are to establish models in different levels and phases, therefore to solve the challenges of requirement and technology change. However, there is only a standard established by Object Management Group (OMG) but without a formal method and approach. Presently, MDA is widely researched in both industrial and research areas, however, there is still without a smooth approach to realise it especially in e-learning system due to the following reasons: (1) models' transformations are hard to realise because of lack of

tools, (2) most of existing mature research results are working for business area but not education area, and (3) most of existing model-driven researches are based on Model-Driven Development (MDD) but not MDA because of OMG standard's preciseness.

Hence, this thesis is trying to provide solutions to these problems. The research aims to investigate an MDA-based approach and evolution method to improve the existing software development approach for e-learning system, therefore, to address discussed issues.

1.2 Research Objectives and Research Methods

The research described in this thesis has following objectives:

- To develop a Model-Driven Architecture based evolution method
- To create a guideline for electronic learning domain modelling
- To define a transformation method for the Model-Driven Architecture based approach, especially for mapping Platform Independent Model to Platform Specific Models
- To deploy and therefore to validate a Model-Driven Architecture based approach's application in an electronic learning system

The proposed research aims to build a Model-Driven Architecture based evolution method and to apply the approach in an electronic learning system. This section describes the research methods applied in this thesis, which links the knowledge coming from research to the practical outcomes. The research field in this thesis belongs to software engineering. The research method applied in this thesis is the combination of

empirical and constructive research that is of both high practical utility and academic merit. The basic methods used in this thesis are summarised as follows:

- Critical Literature Review: it involves reviewing readily available materials. It has been achieved and reported in Chapter 2. The review concentrated on model-driven engineering, model-driven architecture, software evolution, and electronic learning.

- Methodology: a methodology is proposed in the thesis for model-driven architecture based evolution and implementation for electronic learning system development, which links models and system implementation.

- Positivism: it is applied to increase the confidence on proposed method and approach through empirical observation on transformation rules' implementation and proposed method's application.

- Formal method: With the support of the mathematically proved formula and existing modelling standards, formal methods provide system development with (semi-) automatic solutions.

- Cognitive theory: it mainly relies on domain knowledge and experience in this thesis to support electronic learning modelling.

- Quantitative method: the proposed research reflects qualitative method by discussing "wh-questions" and the discussion of the more specific questions such as "how many" and "how often". Generally speaking, qualitative method provides the precondition of the usage of quantitative method.

- Modelling: the proposed research studies model-driven architecture based development method and defines meta-models to support transformation.

Furthermore, the thesis designs the architecture for electronic learning system representing by models.

- Classification: this thesis studies electronic learning system, therefore, different categories of users and functions are required to be clarified. Additionally, models need to be grouped based on development architecture and design pattern.

1.3 Research Questions and Hypotheses

Research questions are the core part of the structure of the proposed research. The principle research question in this study can be formulated as follows:

How can software systems, especially electronic learning system, be developed maximum automatically by Model-Driven Architecture theory?

The research work described in this thesis aims to address this research question effectively. In order to achieve this aim, a set of manageable and tractable sub-questions are defined that address the problem in details.

RQ1: What is the proper development approach to achieve the requirement of Model-Driven Architecture standard?

- Which models are necessary based on Model-Driven Architecture standard?
- How to present those models to meet standard's requirements?
- How to organise models' structure following Model-Driven Architecture standard?

RQ2: How to realise transformations for Model-Driven Architecture based electronic learning system development approach?

- What is transformation method in the approach?
- Which phases need transformations?
- Which mapping rules should be followed in each transformation step?

RQ3: What is the proper way to modelling electronic learning system?

- What are basic functions required in electronic learning domain modelling?
- Which standards, strategies, or patterns should be involved to support domain modelling for electronic learning?
- How to organise domain models to meet requirements on Model-Driven Architecture based development approach?

RQ4: What is the Model-Driven Architecture based development process for electronic learning system?

- Which phases are necessary in this development process?
- What elements should be presented in each phase?
- How to get each model's elements automatically?
- How to define and represent the relationships between models?

RQ5: What kind of tools is capable to support the proposed Model-Driven Architecture based approach?

In order to explore these research questions, a series of research propositions are developed. The underlying proposition of this thesis is:

Model-Driven Architecture can be used to establish an approach for evolution of electronic learning system in order to automate processes in development and maintenance and, as a result, improves the efficiency of Model-Driven Architecture application in electronic learning domain.

This phase is tested by analysing, developing, and deploying Model-Driven Architecture in software systems. A set of propositions is derived from the kernel one:

RP1: Model-Driven Engineering can be used to lead system development successfully. This proposition can be tested by developing models for different software system. Every system can be modelled by specific rules.

RP2: Model-Driven Architecture is a defective standard can be followed to establish an approach. This proposition is tested by OMG in its standards.

RP3: Electronic learning system can be modelled following MDA standards. This proposition can be tested by analysing e-learning system general requirement.

Furthermore, appropriate concepts, mechanisms, techniques and rules will be proposed to resolve each of the sub-questions. To achieve that end, this research will take advantage of advantage of advances in modelling techniques, transformation methodologies, object oriented and e-learning. Subsequently, all of them will be merged as an integrated evolution method that leads the development of electronic learning system in a progressive, precise and disciplined manner.

1.4 Original Contributions

A Model-Driven Architecture based evolution method is proposed in the context of software engineering and model driven development. It is an application of automatic models and transformations to the task of electronic learning domain. In this research, the kernel contributions are a set of precise transformation rules and its application in electronic learning system. The following are explanations of all of the original contributions:

- C1: A Model-Driven Architecture based evolution method and approach is proposed, aiming to automate the modelling and coding processes in electronic learning system, and thereby to improve the efficiency of e-learning system development and maintenance. Ontology technique is applied into PIM generation process.

- C2: A PIM to PSM transformation method is designed as a set of mapping rules for e-learning system to enrich the Model-Driven Architecture transformation theory in electronic education domain.

- C3: E-learning system is modelled as domain framework. LTSA standard and pedagogical strategy is applied into modelling to support e-learning system.

- C4: Model-View-Controller structure is applied into e-learning domain modelling and proposed MDA-based evolution approach.

- C5: Software Engineering Creative Computing is proposed as a new concept with examples and application.

- C6: An application in an e-learning system is deployed concretely to validate the usability, productivity, interoperability, and efficiency on the proposed Model-Driven Architecture based approach.

- C7: A set of tools is summarised and analysed to support the proposed Model-Driven Architecture based evolution method for e-learning system.

1.5 Success Criteria

The overall measure of success of the proposed Model-Driven Architecture based evolution method is how well it supports a successful electronic learning system

development. The following measures are given to judge the success of the research described in this thesis:

- How to develop electronic learning system following Model-Driven Architecture standard?
- How many standards or theories are supporting electronic learning domain modelling?
- How many phases are automatic or semi-automatic in proposed development process?
- How to realise transformations for models in different levels?
- How many rules are supporting Platform-Specific Model generation?
- How many electronic learning systems are suitable to be developed based on proposed method?

1.6 Organisation of Thesis

The rest of this thesis is organised as follows:

In Chapter 2, an overview of research background and basic concepts is summarised. The areas include electronic learning system, Meta Object Facility (MOF) and Query/View/Transformation (QVT).

In Chapter 3, the related work is discussed. It includes the modelling languages, Model Driven Architecture (MDA), transformation notations and examples, Model Driven Engineering (MDE) and Software Engineering Creative Computing (SECC). Projects related to this research are explored and summarised critically.

In Chapter 4, it is e-learning modelling specification including domain modelling and Model-View-Controller (MVC) modelling. Learning Technology Systems Architecture (LTSA) is employed to support e-learning domain modelling as a kernel standard. Concretely, there are three models proposed and designed with pedagogical strategy background. In addition, MVC is applied to be the main structure in the e-learning modelling.

In Chapter 5, an approach is proposed to MDA-based evolution including evolution method and process. The method is specified individually in CIM, PIM, PSM, and code development. The proposed process is divided into two phases: vocabularies extraction and PIM generation.

In Chapter 6, there is proposed transformation method and specification on PIM to PSMs. Scope and normative are presented in the beginning. First, Transformation method is explained based on QVT standard. Meanwhile, transformation architecture is designed to present the method in levels. Second, there is specification for proposed source model, metamodel of source model, target model, and metamodel of target model. Then, transformation rules are designed with formula expression which is classified into transformation model (Mt) and metamodel for transformation model (MMt). Last, there is an implementation for mapping rules.

Chapter 7 is a case study supported by a project named ElectroAcoustic Resource Site II (EARSII). Background and issues of EARSII are introduced in the beginning. Then, it is a development approach in which proposed MDA-based method is applied. Additionally, there are toolkit explanation and prototype screen shots.

Chapter 8 summarises the thesis, draws conclusions and discusses the future work. The research questions are revisited and answered in order to evaluate the proposed approach.

Appendix A is the codes generated from the proposed approach application to create prototype of EARS Ⅱ.

Appendix B lists all the related publications written by the author during the PhD study.

Chapter 2
Background and Basic Concepts

Objectives

- To introduce electronic learning system as the background of application domain
- To introduce Meta Object Facility and Query/View/Transformation
- To explain basic concepts and terms related to this research

2.1 Electronic Learning System

2.1.1 Electronic Learning

Electronic learning, abbreviated as e-learning, is a relatively new concept appeared in this decade. Based on involved technology, there is a learning structure showing as

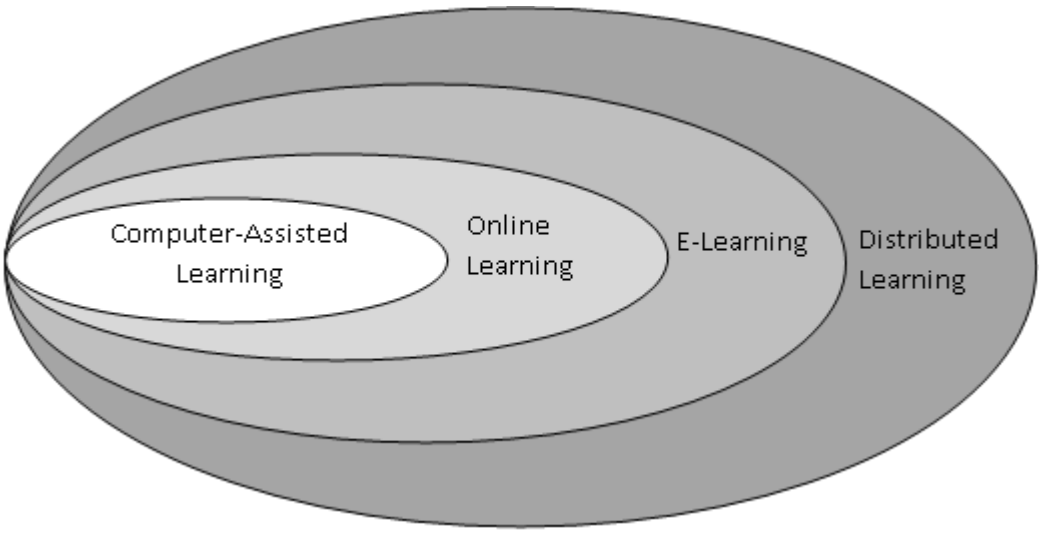


Figure 2-1. Technology-Based Learning Structure

Figure 2-1 which is concluded based on general classification for e-learning.

2.1.2 E-Learning Standards and Specifications

With e-learning's rapid development, there are many rules are proposed by different groups. There are few known as generally recognised standards lists as below with its specifications,

- **Advanced Distributed Learning**

The Advanced Distributed Learning (ADL) Initiative is a strategy sponsored by the government, industry, and academic leaders to facilitate instructional content development and delivery using current and emerging technologies. Specifically, its Sharable Content Object Reference Model (SCORM) project focuses on next-generation open architecture for online learning, including standards for run-time communication, course structure, and content meta-data.

- **Aviation Industry Computer-Based Training Committee**

The Aviation Industry Computer-Based Training Committee (AICC) is an international association of technology-based training professionals that creates guidelines for the development, delivery, and evaluation of training technologies. The AICC pioneered the most widely accepted interoperability standards for computer-based and web-based training. Its relevant publications for standards are the AICC Guidelines and Recommendations (AGRs).

- **IEEE Learning Technology Standards Committee**

The Institute of Electrical and Electronics Engineers (IEEE) Computer Society Standards Activity Board has chartered the Learning Technology Standards Committee

(LTSC) to develop technical standards, recommended practices, and guides for computer implementations of education and training components and systems—specifically, the software components, tools, technologies, and design methods that facilitate their development, deployment, maintenance, and interoperation. Many of the standards developed by LTSC will be advanced as international standards by the International Organisation for Standardization/International Electro technical Commission Joint Technical Committee Subcommittee on Information Technology for Learning, Education, and Training (ISO/IEC JTC1/SC36).

- **IMS Global Learning Consortium**

The IMS Global Learning Consortium, Inc. (IMS) is a nonprofits corporation that began with a focus on higher education. Today, they've expanded their specifications and projects to address a wide range of learning contexts, including school, university, corporate, and government training. Available specifications include: Learning Resource Meta-data, Enterprise, Content Packaging, and Question & Test Interoperability.

2.1.3 E-Learning 2.0

Over the past couple of years, e-learning is one of the strongest uses of web 2.0 technologies. There is a list of new tools in web 2.0,

- Content Creation: blog, E-portfolios (such as ELGG), and video (i.e. YouTube).
- Collaborative Writing: wikis (i.e., PB Wiki, Media Wiki, etc), collaborative bookmarking (i.e., del.icio.us, Furl, etc), and online office applications (i.e., Writely, Gliffy, iRows, etc).

- Aggregator: for example, “Aggregate This”, “MetaxuCafe”, “Postgennomic”, “Edu_RSS”, “Intute”, etc.

The above new tools lead to new communication manners. Therefore, a new e-learning method is required to fit learner’s changes. With web2.0’s great changing in e-learning area, e-learning 2.0 is proposed as a new concept under this circumstance by Stephen Downes [19].

2.1.4 Web-Based E-Learning Development

There are many architectures existing to support web-based development for various requirement. In this section, related web-based architectures are reviewed and discussed including three-tier architecture and Model-View-Controller pattern which are considered suitable for development on e-learning system.

2.1.4.1 Three-Tier Architecture on Web Application

The figure 2-2 shows the three-tier architecture’s application in web system. There are three layers.

- **Presentation tier:** it is generally a user interface (UI). It aims to translate information (i.e., tasks, results, etc) to user’s understanding.

- **Business logic tier:** it is to coordinate application, process commands, make logic decisions and evaluations, and perform calculations. It also manipulates data between “presentation layer” and “data access layer”.

- **Data access tier:** it stores and retrieves information from database or file system. It provides data to business logic tier for processing.

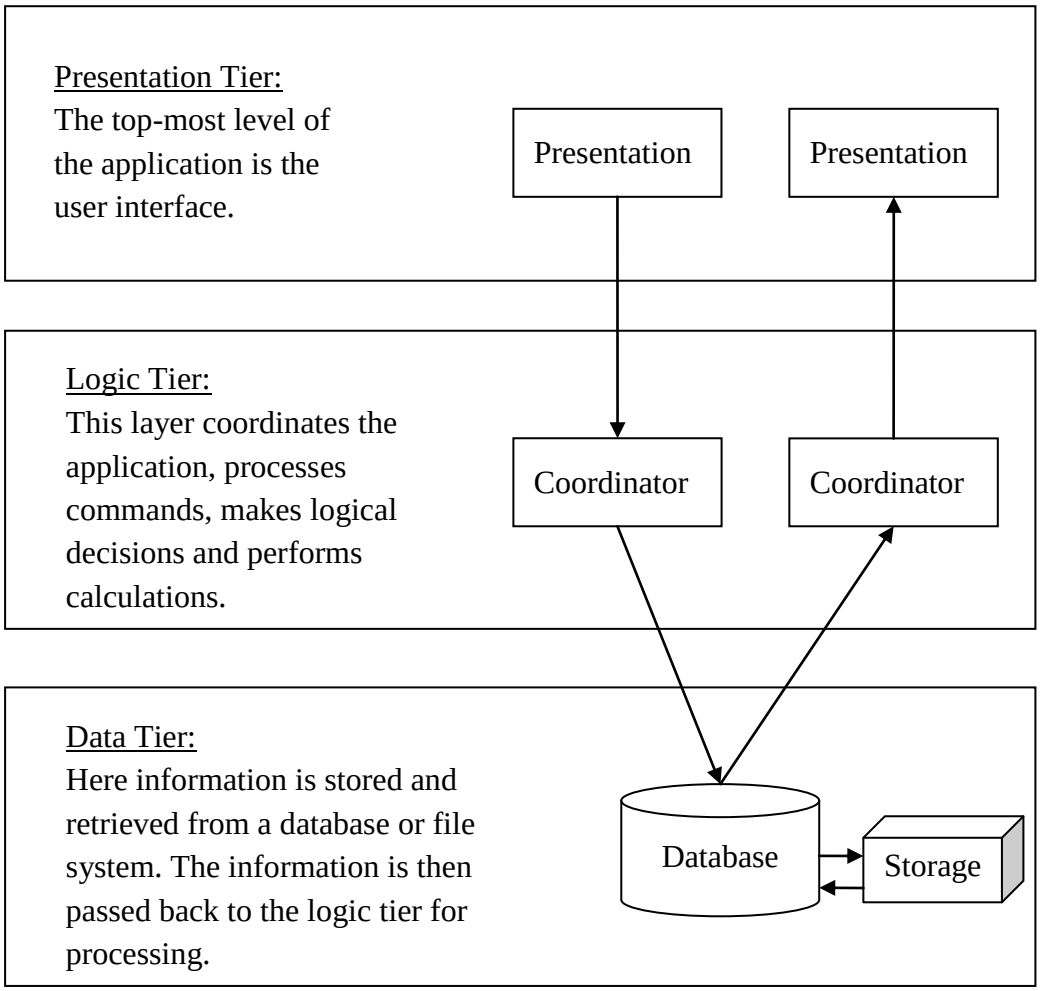


Figure 2-2. Overview of Three-Tier Application

2.1.4.2 Model-View-Controller Architecture

Model-view-controller, abbreviated to MVC, is software architecture in software engineering. It is also considered as an architectural pattern. Initially, MVC is presented as a software design pattern for Smalltalk in 1974. It aims to realise a dynamic system design process to simplify further maintenance, and to improve the reusability. The MVC pattern contains three modules: Model, View, and Controller. Figure 2-3 shows the MVC basic structure.

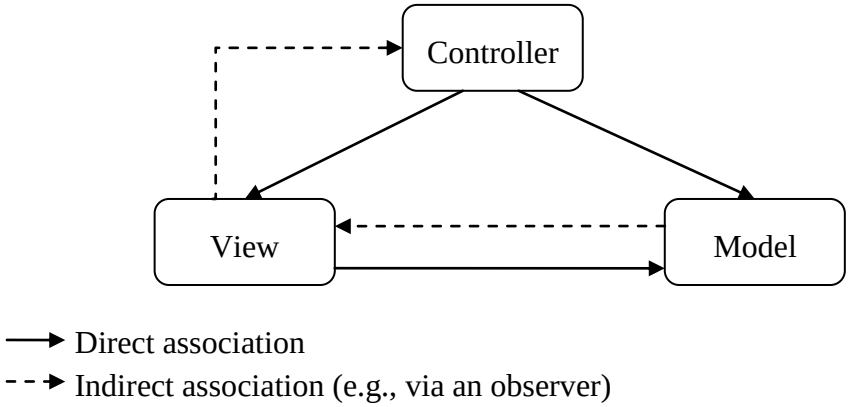


Figure 2-3. MVC Concept

- **Model:** it manages the behaviour and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller). In event-driven systems, the model notifies observers (usually views) when the information changes so that they can react.
- **View:** it renders the model into a form suitable for interaction, typically a user interface element. Multiple views can exist for a single model for different purposes. A viewport typically has a one to one correspondence with a display surface and knows how to render to it.
- **Controller**: it receives user input and initiates a response by making calls on model objects. A controller accepts input from the user and instructs the model and a viewport to perform actions based on that input.

In short, the *model* is the application object, the *view* is the screen presentation, and the *controller* defines the way the user interface reacts to user input [104].

The MVC pattern provides visualised software architecture by simplify the complexity. Fundamental components are divided from each other and acquired their own functions.

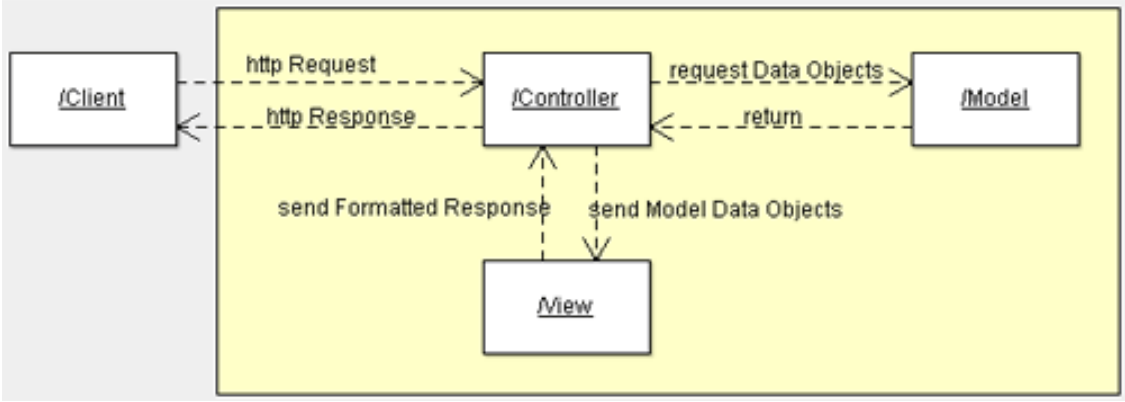


Figure 2-4. MVC Collaboration Diagram

The above figure contains the MVC Collaboration Diagram, where the links and dependencies between figures can be observed.

Furthermore, there is a short php example with a simple structure to present a clear impression on MVC's implementation which is putting each MVC module in one folder:

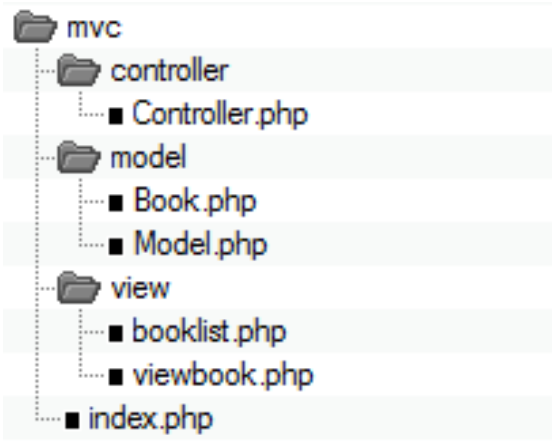


Figure 2-5. A MVC-Based PHP Project Example

2.2 Meta Object Facility and Query/View/Transformation

2.2.1 Meta Object Facility

The Meta Object Facility (MOF) is an adopted OMG standard. Its latest version is MOF 2.0. It provides a metadata management framework, and a set of metadata services to enable the development and interoperability of model and metadata driven systems [77].

MOF has contributed significantly to some of the core principles of Model Driven Architecture [77]. This technology provides a model repository that can be used to specify and manipulate models. Therefore, MOF is a technology to encourage consistency in manipulating models in all phases of the use of MDA.

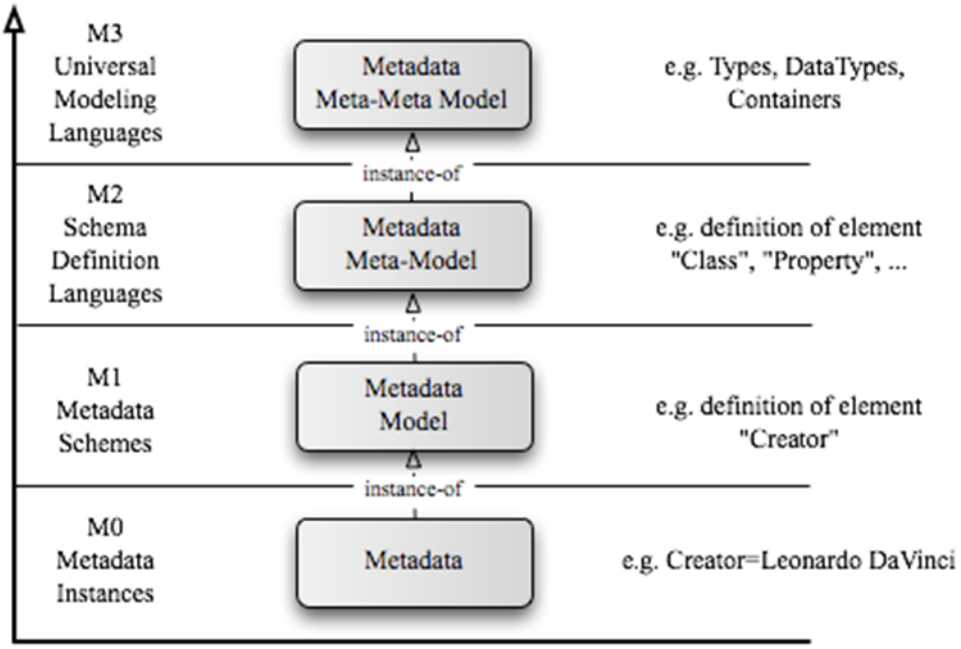


Figure 2-6. MOF Meta-Levels Hierarchy [32]

Figure 2-6 [32] shows a typical four-layered modelling architecture based on meta-model and MOF. Specification of MOF defines the most abstract layer M3 which

provides an abstract language and a framework for specifying, constructing, and managing meta-models. Meta-model reside on layer M2 which provides meta-data to construct model. Layer M1 is for the models which represent software system and real life.

Related OMG standards are UML, MOF, CWM, SPEM, XMI, and various UML profiles. Those technologies use MOF and MOF derived technologies for metadata-driven interchange and metadata manipulation [77].

The MOF Model is used to model itself as well as other models and other metamodels [77]. XML Meta-data Interchange (XMI) [76, 82] is used as a standard common model exchange format which enables the developers to achieve the same understanding and interpretation when exchanging models via different technologies and tools. XMI is an XML standard for exchanging UML models. XML schema conversion rules indicate how UML model can be converted to XML document.

2.2.2 Query/View/Transformation

Query/View/Transformation (QVT) is a standard for model transformation, specifically in MDA, proposed and defined by the Object Management Group. Here, the QVT is MOF 2.0 QVT. It can be considered as the kernel in MDA although there are no specific rules defined for model transformation. In MDA guide [62], a MOF QVT transformation model is used as a example for transformation specification. QVT defines standards to guide the design of transformation rules.

QVT provides a formal method for model transformation research in MDA. There are three features in QVT standard: strong expression ability, well-defined semantics, and fully automated.

The QVT specification has a hybrid declarative/imperative nature, with the declarative part being split into a two-level architecture, which forms the framework for the execution semantics of the imperative part [74].

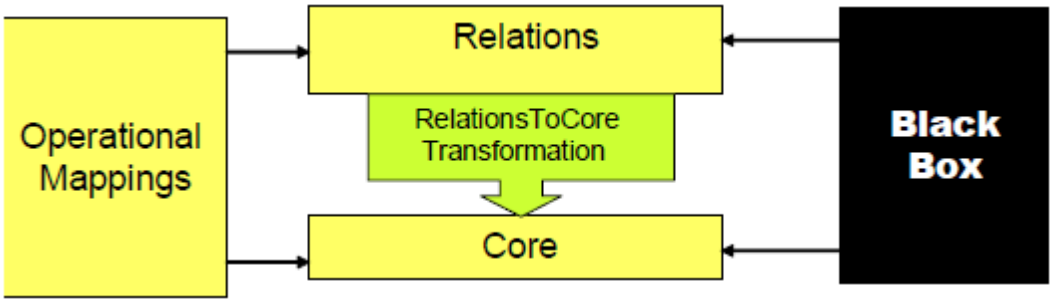


Figure 2-7. Relationships Between QVT Metamodels [74]

As figure 2-7 [74] shows, the declarative parts of this specification are structured into a two-layer architecture. The layers are [74]:

- A user-friendly Relations metamodel and language that supports complex object pattern matching and object template creation. Traces between model elements involved in a transformation are created implicitly.

- A core metamodel and language defined using minimal extensions to EMOF and OCL. All trace classes are explicitly defined as MOF models, and trace instance creation and deletion is defined in the same way as the creation and deletion of any other object.

In addition to the declarative Relations and Core Languages that embody the same semantics at two different levels of abstraction, there are two mechanisms for invoking imperative implementations of transformations from Relations or Core: one standard language, Operational Mappings, as well as non-standard Black-box MOF Operation implementations [74]. Each relation defines a class that will be instantiated to trace

between model elements being transformed, and it has a one-to-one mapping to an Operation signature that the Operational Mapping or Black-box implements [74].

2.3 Basic Concepts and Related Terms

The proposed research will be related to the following terms from model-driven architecture and model-driven engineering [4, 41] in the domain of software engineering.

Model: A model of a system is a description or specification of that system and its environment for some certain purpose. A model is often presented as a combination of drawings and text. The text may be in a modelling language or in a natural language.

Platform: A platform is a set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns, which any application supported by that platform can use without concern for the details of how the functionality provided by the platform is implemented.

Computation Independent Model (CIM): A computation independent model is a view of a system from the computation independent viewpoint. A CIM does not show details of the structure of systems. A CIM is sometimes called a domain model and a vocabulary that is familiar to the practitioners of the domain in question is used in its specification.

Platform Independent Model (PIM): A platform independent model is a view of a system from the platform independent viewpoint. A PIM exhibits a specified degree of platform independence so as to be suitable for use with a number of different platforms of similar type.

Platform Specific Model (PSM): A platform specific model is a view of a system from the platform specific viewpoint. A PSM combines the specifications in the PIM with the details that specify how that system uses a particular type of platform.

Model transformation: Model transformation is the process of converting a set of models to another set of models or to themselves for the same system. It can be divided into two broad categories: model translation and model rephrasing. Model transformation is transforming a model into another language described model. Model rephrasing is changing models under a same language.

2.4 Summary

In this chapter, the background and basic concepts of model-driven architecture based software engineering are introduced:

- A brief overview of electronic learning system is presented. The concept "electronic learning" is introduced with its background, features, and relationship with web technologies. Standards and specifications of e-learning are introduced then. In addition, there is also a discussion of "e-learning 2.0" which is e-learning's evolution under web 2.0 impact. A series of web 2.0 tools are listed. Furthermore, its development methods based on web service are discussed including three-tier architecture on web application and Model-View-Controller (MVC) architecture.

- MOF and QVT are studied. There is a presentation of MOF specification and its model's application. Meanwhile, QVT's standards and its application in transformations are explained.

- Related concepts and terms are summarised to provide a systematised knowledge about the whole research.

Chapter 3
Related Work

Objectives

- To introduce UML and its usages in MDA approach
- To introduce XMI, XML and its role in this research
- To discuss software evolution and related methods
- To introduce PHP and its usages in electronic system
- To explore software engineering creative computing and its application in e-learning system
- To discuss related projects covering MDA and e-learning

3.1 Unified Modelling Language

3.1.1 Unified Modelling Language 2.x

Unified Modelling Language (UML) is a standardised general-purpose modelling language in the field of object-oriented software engineering. It is used to specify, visualise, modify, construct and document the artefacts of an object-oriented software-intensive system under development [2]. It offers a set of elements [79, 80] to create visual models for system's architecture including "Activities", "Actors", "Business processes", "Database schemas", "Logical components", "Programming language statements", and "Reusable software components". Based on these elements, UML includes a set of diagrams to create models to visualise architecture for system. A diagram is a partial graphic representation of a system's model. The model also

contains documentation that drives the model elements and diagrams such as written use cases. Besides, UML models can be exchanged among UML tools.

UML diagrams represent two different views of a system model [33]:

- Static view: it is also called structural view which emphasises the static structure of the system using objects, attributes, operations and relationships. The structural view includes class diagrams and composite structure diagrams.
- Dynamic view: Equally with behavioural view, it emphasises the dynamic behaviour of the system by showing collaborations among objects and changes to the internal states of objects. This view includes sequence diagrams, activity diagrams and state machine diagrams.

UML	
Concepts	**Object oriented:** Object-oriented programming; Object-oriented analysis and design. **Structure:** Actor; Attribute; Artefact; Class; Component; Interface; Object; Package; Profile diagram. **Behaviour:** Activity; Event; Message; Method; State; Use case. **Relationships:** Aggregation; Association; Composition; Dependency; Generalisation (or Inheritance). **Extensibility:** Profile; Stereotype. **Other concepts:** Multiplicity.

Diagrams	**Structural**	Class diagram; Component diagram; Composite structure diagram; Deployment diagram; Object diagram; Package diagram
	Behaviour	Activity diagram; State Machine diagram; Use case diagram.
		Interaction: Communication diagram; Sequence diagram; Interaction overview diagram; Timing diagram.
Derived Languages	Systems Modelling Language (SysML); UML eXchange Format (UXF); XML Metadata Interchange (XMI).	

Table 3-1. UML Structure

Structure diagrams emphasise the things that must be present in the system being modelled. Since structure diagrams represent the structure, they are used extensively in documenting the software architecture of software systems.

- **Class diagram:** describes the structure of a system by showing the system's classes, their attributes, and the relationships among the classes.
- **Component diagram:** describes how a software system is split up into components and shows the dependencies among these components.
- **Composite structure diagram:** describes the internal structure of a class and the collaborations that this structure makes possible.

- **Deployment diagram:** describes the hardware used in system implementations and the execution environments and artefacts deployed on the hardware.

- **Object diagram:** shows a complete or partial view of the structure of an example modelled system at a specific time.

- **Package diagram:** describes how a system is split up into logical groupings by showing the dependencies among these groupings.

- **Profile diagram:** operates at the metamodel level to show stereotypes as classes with the <<stereotype>> stereotype, and profiles as packages with the <<profile>> stereotype. An extension relation indicates what metamodel element a given stereotype is extending.

Behaviour diagrams emphasise what must happen in the system being modelled. Since behaviour diagrams illustrate the behaviour of a system, they are used extensively to describe the functionality of software systems.

- Activity diagram: describes the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

- UML state machine diagram: describes the states and state transitions of the system.

- Use case diagram: describes the functionality provided by a system in terms of actors, their goals represented as use cases, and any dependencies among those use cases.

Interaction diagrams, a subset of behaviour diagrams, emphasise the flow of control and data among the things in the system being modelled:

- Communication diagram: shows the interactions between objects or parts in terms of sequenced messages. They represent a combination of information taken from Class, Sequence, and Use Case Diagrams describing both the static structure and dynamic behaviour of a system.

- Interaction overview diagram: provides an overview in which the nodes represent communication diagrams.

- Sequence diagram: shows how objects communicate with each other in terms of a sequence of messages. Also indicates the lifespan of objects relative to those messages.

- Timing diagrams: a specific type of interaction diagram where the focus is on timing constraints.

3.1.2 Unified Modelling Language and Model-Driven Architecture

Consequently, it is expected that this major revision to UML will play an important role in furthering the goals of MDA [79]. It is noted in the OMG's Executive Overview of MDA:

"[MDA] is built on the solid foundation of well-established OMG standards, including: Unified Modelling Language™ (UML™), the ubiquitous modelling notation used and supported by every major company in the software industry; XML Metadata Interchange (XMI™), the standard for storing and exchanging models using XML; and CORBA™, the most popular open middleware standard" [78].

The following sections explain how UML 2 supports the most prominent concepts in the evolving MDA vision.

- **Family of languages:** UML is a general purpose language that is expected to be customised for a wide variety of domains, platforms and methods. Towards that end, this UML specification refines UML 1.x's Profile mechanism so that it is more robust and flexible, and significantly easier to implement and apply. Consequently, it can be used to customise UML dialects for various domains (e.g., finance, telecommunications, aerospace), platforms (e.g., J2EE, .NET), and methods (e.g., Unified Process, Agile methods). For those whose customisation requirements exceed these common anticipated usages and who want to define their new languages via metamodels, the Infrastructure Library is intended to be reused by MOF 2. Tools that implement MOF 2 will allow users to define entirely new languages via metamodels
- **Specifying a system independently of the platform that supports it:** As was the case with its predecessor, the general purpose UML 2 specification is intended to be used with a wide range of software methods. Consequently, it includes support for software methods that distinguish between analysis or logical models, and design or physical models. Since analysis or logical models are typically independent of implementation and platform specifics, they can be considered "Platform Independent Models" (PIMs), consistent with the evolving MDA terminology. Some of the proposed improvements to UML that will make it easier for modellers to specify Platform Independent Models include the ability to model logical as well as physical Classes and Components, consistent with either a class-based or component-based approach.

- **Specifying platforms:** Although UML 1.x provided extremely limited support for modelling Platform Specific Models (PSMs, the complement of PIMs), this specification offers two significant improvements. First, the revised Profile mechanism allows modellers to more efficiently customise UML for target

platforms, such as J2EE or .NET. (Examples of J2EE/EJB or .NET/COM micro-profiles can be found in the UML Superstructure Specification.) Secondly, the constructs for specifying component architectures, component containers (execution runtime environments), and computational nodes are significantly enhanced, allowing modellers to fully specify target implementation environments.

- **Choosing a particular platform for the system:** This is considered a method or approach requirement, rather than a modelling requirement. Consequently, we will not address it here.

- **Transforming the system specification into one for a particular platform:** This refers to the transformation of a Platform Independent Model into a Platform Specific Model. The UML Superstructure Specification specifies various relationships that can be used to specify the transformation of a PIM to a PSM, including Realisation, Refine, and Trace. However, the specific manner in which these transformations are used will depend upon the profiles used for the PSMs involved, as well as the method or approach applied to guide the transformation process. Consequently, we will not address it further here.

With its middleware independent, UML forms foundation of MDA. Besides, extensions to the UML language will be standardised for specific purposes including many will be designed specifically for use in MDA [62].

3.1.3 UML Profile

OMG has started working on UML profiles. They include the UML profile for CORBA, for use by models specific to the CORBA platform, and the EDOC profile, for use in platform independent models for certain classes of platforms, as well as

profiles for enterprise integration and real-time platforms [62]. A profile in the Unified Modelling Language (UML) provides a generic extension mechanism for customising UML models for particular domains and platforms [6]. Extension mechanisms allow refining standard semantics in strictly additive manner, so that they can't contradict standard semantics [6].

A UML profile is a specification that does one or more of the following [83]:

- Identifies a subset of the UML metamodel.
- Specifies "well-formed rules" beyond those specified by the identified subset of the UML metamodel. "Well-formed rules" is a term used in the normative UML metamodel specification to describe a set of constraints written in UML's Object Constraint Language (OCL) that contributes to the definition of a metamodel element.
- Specifies "standard elements" beyond those specified by the identified subset of the UML metamodel. "Standard element" is a term used in the UML metamodel specification to describe a standard instance of a UML stereotype, tagged value or constraint.
- Specifies semantics, expressed in natural language, beyond those specified by the identified subset of the UML metamodel.
- Specifies common model elements, expressed in terms of the profile.

Profiles are defined using stereotypes, tag definitions, and constraints that are applied to specific model elements, such as Classes, Attributes, Operations, and Activities. A Profile is a collection of such extensions that collectively customise UML for a

particular domain such as aerospace, healthcare, financial, etc, or platform like J2EE, .NET, and so on. There are two examples.

- **SysML** is an Object Management Group (OMG)-standardised profile of Unified Modelling Language that is used for system engineering applications.

- **MARTE** is the OMG standard for modelling real-time and embedded applications with UML2.

Presently, OMG is developing a series UML profile for domains [83].

- UML Profile for CORBA and CORBA Component Model (CCCMP)
- UML Profile for Data Distribution
- UML Profile for Enterprise Application Integration (EAI)
- UML Profile for Enterprise Distributed Object Computing (EDOC)
- UML Profile for Modelling and Analysis of Real-time and Embedded Systems (MARTE)
- UML Profile for Modelling QoS and Fault Tolerance Characteristics and Mechanisms
- UML Profile for Schedulability, Performance and Time
- UML Profile for Software Radio (also referred to as PIM & PSM for Software Radio Components)
- UML Profile for System on a Chip (SoCP)
- UML Profile for Voice
- UML Testing Profile (UTP)

3.2 Overview of Extensible Markup Language (XML), XML Metadata Interchange, and Hypertext Preprocessor

Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

The XML Metadata Interchange Format (XMI) specifies an open information interchange model that is intended to give developers working with object technology the ability to exchange programming data over the Internet in a standardised way, thus bringing consistency and compatibility to applications created in collaborative environments [34]. “XMI Document” and “XMI Schema” are defined as documents and schemas produced by the XMI production [82].

Hypertext Preprocessor, recursive acronym for PHP, is a general-purpose server-side scripting language originally designed for web development to produce dynamic web pages [85]. PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document. It also has evolved to include a command-line interface capability and can be used in standalone graphical applications [85].

3.3 Software Evolution

3.3.1 Software Changes and Evolution

Software is a kind of product which is never complete and continues to evolve. Presently, changing comprises a heavy and important part of the software life cycle.

There are four categories of software were catalogued by Lientz and Swanson [51]. These have since been updated and normalised internationally in the ISO/IEC 14764:2006 [1] :

- Corrective maintenance: Reactive modification of a software product performed after delivery to correct discovered problems;
- Adaptive maintenance: Modification of a software product performed after delivery to keep a software product usable in a changed or changing environment;
- Perfective maintenance: Modification of a software product after delivery to improve performance or maintainability;
- Preventive maintenance: Modification of a software product after delivery to detect and correct latent faults in the software product before becoming effective faults.

All of the preceding take place when there is a known requirement for change. Although these categories were supplemented by many researchers, international standard ISO/IEC 14764:2006 has kept the basic four categories. More recently the description of software maintenance and evolution has been done using ontology [17, 18, 39, 90] which enrich the description of the many evolution activities. Presently, the activities of software change can be classified into three categories: maintenance, reengineering, and evolution [107].

- **Software maintenance** is the partly or fully modification of software based on changes on product requirements or hardware environments. The modification process is efficiently using the existing software. It is a term suitable to describe

the occasionally correction which aims to enable the software to continue to do what it used to do.

- **Software reengineering** is modifying and restructuring existing software based on reverse engineering. New version software is the reengineering prospected result. Its kernel feature is farthest realise software's reusability.
- **Software evolution** is a developed maintenance process implementing and validating the possible major changes without being able a priori to predict how user requirements will evolve [37]. It helps software to meet new business opportunities rapidly.

These three activities can be considered as separated concepts; however, they are tightly connected at the same time. Reengineering is still the basic technique for evolving software systems. Software change can be seen as the basic operation of software evolution [37]. Reengineering is a single change cycle, while evolution will carry on indefinitely – software evolution is repeated software engineering [107]. In traditional software life cycle, all the changing activities are belonging to maintenance phrase.

3.3.2 Laws of Software Evolution

The laws of software evolution [45-47, 53] are proposed by Meir M. Lehman and his colleagues when he was working at Imperial College London. Initially the laws were related to large systems. Therefore, Lehman clarified large systems by a classification scheme distinguishing three types of programs S, P and E [46, 48, 49].

- S-type program: it presents the program which can be formally specified [52] which is showing as Figure 3-1.

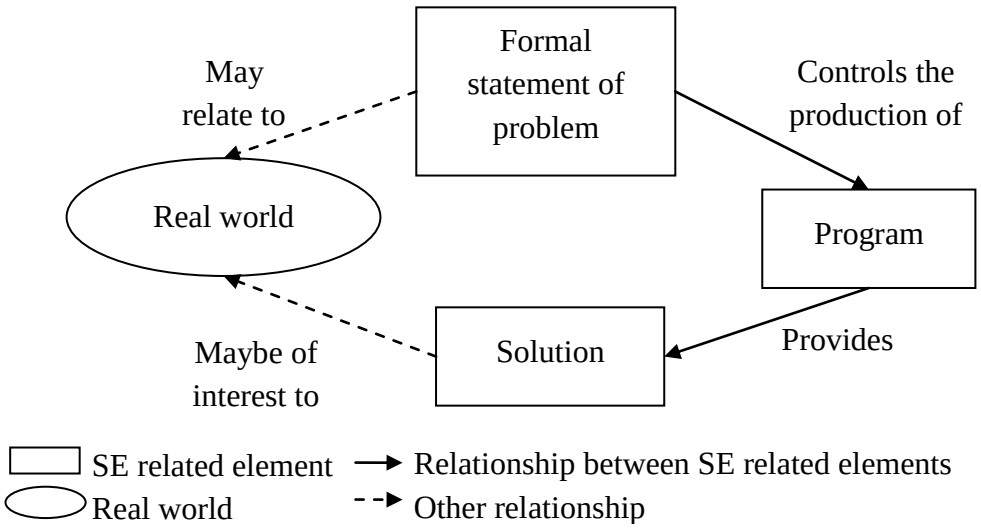


Figure 3-1. S-Type

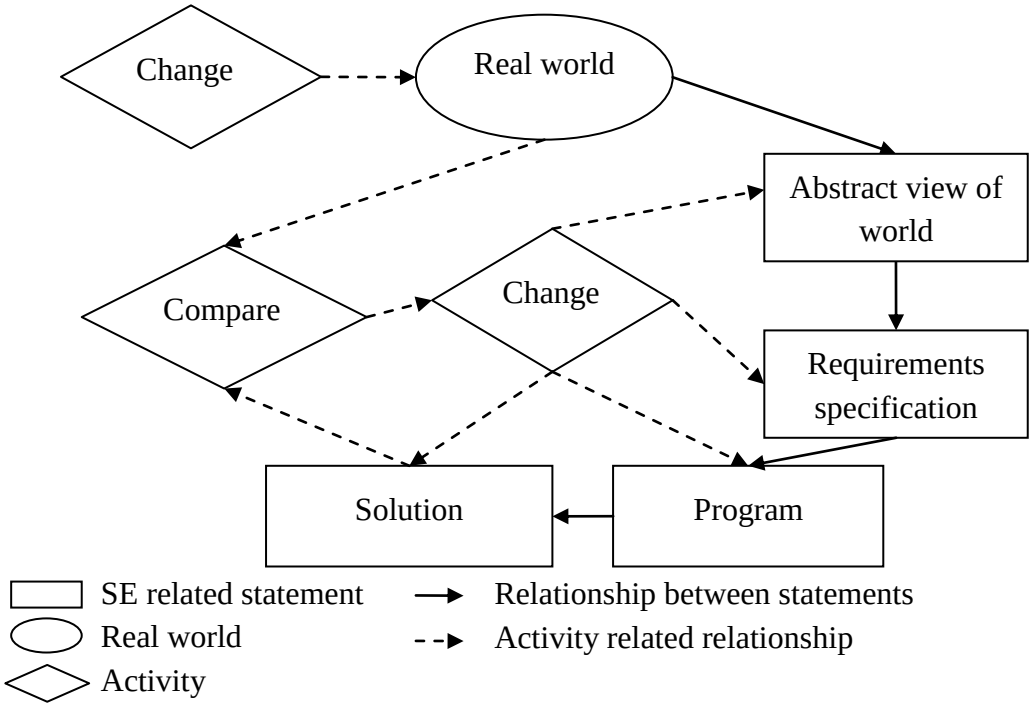


Figure 3-2. P-Type

- P-type program: it stands for the category which is iterative process that cannot be specified [52]. Figure 3-2 is a picture for it with activities, relationships, and statements.

- E-type program: it is embedded in real world, and it is a computer program that solves a problem or implement a computer application in the real world domain [52] showing as Figure 3-3.

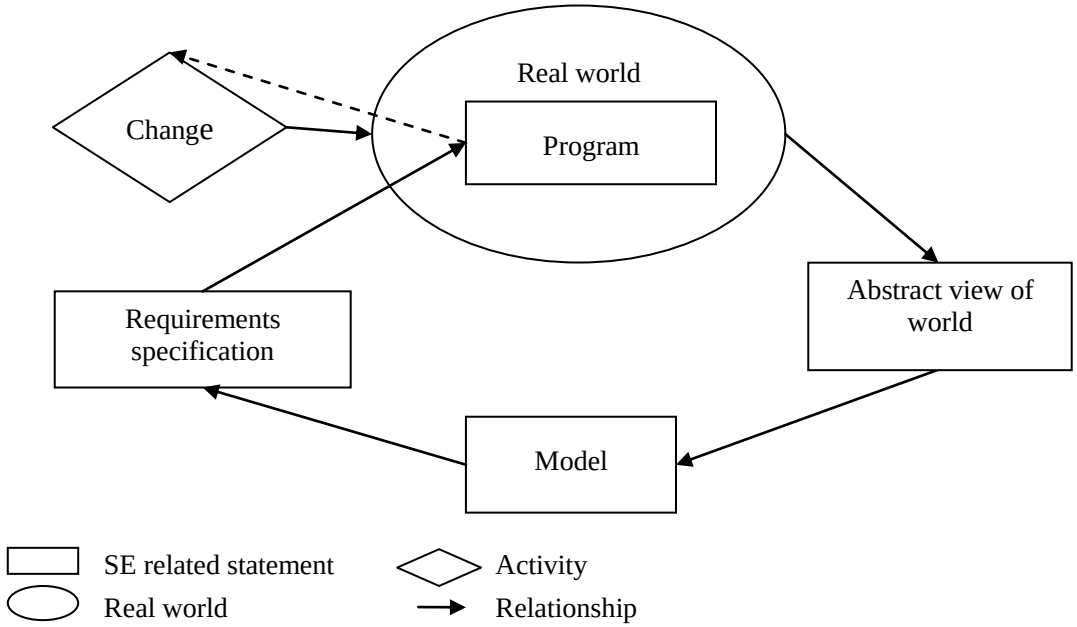


Figure 3-3. E-Type

The laws are suitable to apply to al E-type systems irrespective of their size, functional nature, domain of application, or the management structure and organisation responsible for them [49]. Table 2-1 [45] shows the eight Lehman's laws.

Law	**Description**
I. Continuing Change	An E-type program that is used must be continually adapted else it becomes progressively less satisfactory.
II. Increasing Complexity	As a program is evolved its complexity

	increases unless work is done to maintain or reduce it.
III. Self Regulation	The program evolution process is self regulating with close to normal distribution of measures of product and process attributes.
IV. Conservation of Organisational Stability	The average effective global activity rate on an evolving system is invariant over the product life time.
V. Conservation of Familiarity	During the active life of an evolving program, the content of successive releases is statistically invariant.
VI. Continuing Growth	Functional content of a program must be continually increased to maintain user satisfaction over its lifetime.
VII. Declining Quality	E-type programs will be perceived as of declining quality unless rigorously maintained and adapted to a changing operational environment.
VIII. Feedback System	E-type Programming Processes constitute Multi-loop, Multi-level Feedback systems and

	must be treated as such to be successfully modified or improved.

Table 3-2. Lehman's Laws of Software Evolution [45]

3.4 Model Driven Architecture and Model Driven Engineering

3.4.1 Model Driven Engineering

There is a methodology initiative from the software engineering community named Model Driven Engineering (MDE). The MDE approach to software development suggests that a model of the system should be developed first under related study, which is then transformed into the real entity including code, component, etc [26]. The idea of MDE stems from software engineering, and more specifically, from the recent research in software development [26]. MDE evolved as a paradigm shift from the object-oriented technology, in which the main principle is that everything is an object; into the model engineering paradigm, based on the principle that everything is a model [26]. MDE aims to raise the level of abstraction in program specification and increase automation in program development [11].

MDE is not only containing model but also working with relations between models and systems, metamodels, and model transformations. Similar to the object-oriented technology, MDE can be characterised by two main relations which are representation and conformance. Representation is a model representing software artefact or real-world domain. Conformance is a model conforms to a metamodel. According to Jean-Marie Favre, MDE is a field of system engineering in which the process heavily

relies on the use of models and model engineering. In the context, model engineering is considered the disciplined and rationalised production of models [26].

MDE focuses on creating and exploiting domain models rather than on the computing concepts. Its approach is meant to increase productivity by maximising compatibility between systems via reuse of standardised models, simplifying the process of design via models of recurring design patterns in the application domain, and promoting communication between individuals and teams working on the system via a standardisation of the terminology and the best practices used in the application domain.

As it pertains to software development, MDE refers to a range of development approaches that are based on the use of software modelling as a primary form of expression. Sometimes models are constructed to a certain level of detail, and then code is written by hand in a separate step. Sometimes complete models are built including executable actions. Code can be generated from the models, ranging from system skeletons to complete, deployable products. With the introduction of the Unified Modelling Language (UML), MDE has become very popular today with a wide body of practitioners and supporting tools. More advanced types of MDE have expanded to permit industry standards which allow for consistent application and results. The continued evolution of MDE has added an increased focus on architecture and automation.

According to Schmidt, MDE technologies offer a promising approach to address the inability of third-generation languages to alleviate the complexity of platforms and express domain concepts effectively [2].

3.4.2 Model-Driven Architecture

Model-Driven Architecture, abbreviated to MDA, is a concept initially proposed and launched by the Object Management Group (OMG) in 2001. It is one of the most important research initiatives in MDE area. The main idea is to abstract kernel Platform Independent Model (PIM) which is independent on implementation technology and can complete describes business functions for system, defines transformation rules for different implementation technologies which aim to mapping to implementation technology related Platform Specific Model (PSM), and generates codes from enriched PSM. A main objective is separation of business modelling and the underlying platform technology, to protect the results of modelling from the impact of technological changes.

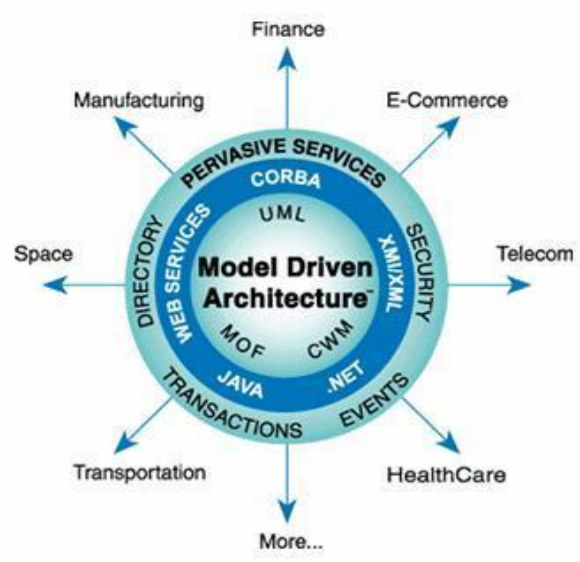


Figure 3-4. MDA Overview Structure [73]

Figure 3-4 is a diagram showing the structure of MDA which is specifically described as bellow:

- The inner part contains Meta Object Facility (MOF), Common Warehouse Metamodel (CWM), and Unified Modelling Language (UML). They are the kernel

technologies in MDA. A main task in MDA is to convert PIM which is formed based on these kernel technologies to PSMs corresponding with different middleware platforms.

- The middle ring shows currently main platforms for implementation. There is including Common Object Request Broker Architecture (CORBA), Extensible Markup Language (XML), Java, Web Services and. NET. Obviously, as technology development, this part contains more platforms than listed here.

- The outer ring is the public services provided by MDA such as “transactions”, “events”, “directory”, “security”, etc.

- The diverging arrows in the outside are MDA’s applications in various perpendicular domains, such as e-commerce, telecommunications and manufacturing, etc.

Overall, the MDA structure can be simply explained as Figure 2-5.

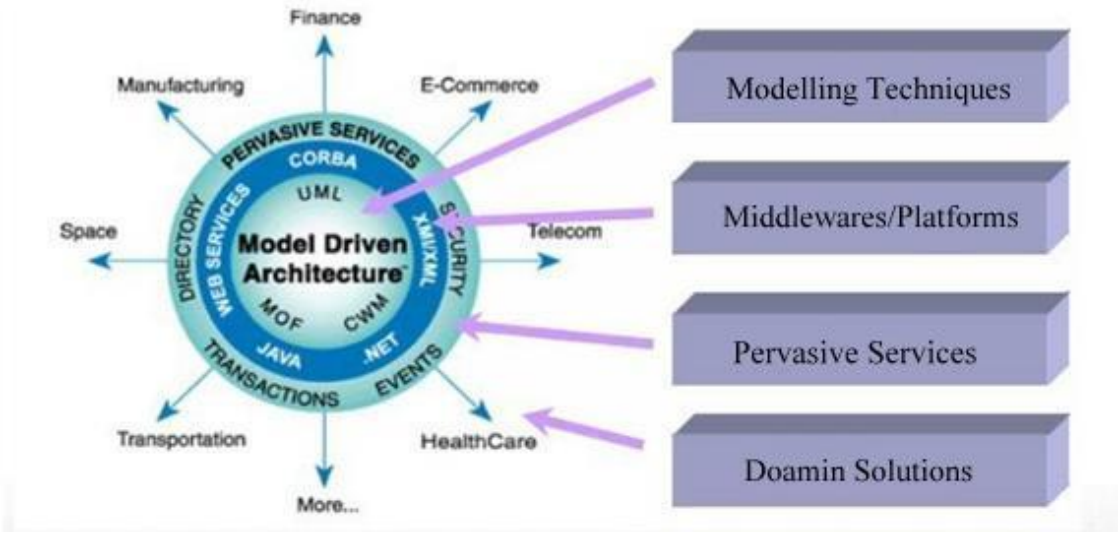


Figure 3-5. MDA Structure Explanation

In the home page on OMG's MDA official website, there is a presentation just besides the figure 2-4 which is never been changed since its first release in 2001. The presentation [73] titled with "How System Will Be Built" is as bellow,

"OMG's Model Driven Architecture (MDA) provides an open, vendor-neutral approach to the challenge of business and technology change. Based on OMG's established standards, the MDA separates business and application logic from underlying platform technology. Platform-independent models of an application or integrated system's business functionality and behaviour, built using UML and the other associated OMG modelling standards, can be realised through the MDA on virtually and platform, open or proprietary, including Web Services, .NET, CORBA, J2EE, and others. These platform-independent models document the business functionality and behaviour of an application separate from the technology-specific code that implements it, insulating the core of the application from technology and its relentless churn cycle while enabling interoperability both within and across platform boundaries. No longer tied to each other, the business and technical aspects if an application or integrated system can each evolve at its own pace-business logic responding to business needs, and technology taking advantage of new developments-as the business requires" [73].

Meanwhile, there is further description [70] on this topic represented by OMG as following,

"MDA provides an open, vendor-neutral approach to the challenge of interoperability, building upon and leveraging the value of OMG's established modelling standards: Unified Modelling Language (UML); Meta-Object Facility (MOF); and Common Warehouse Meta-model (CWM). Platform-independent Application descriptions built using these modelling standards can be realised using

any major open or proprietary platform, including CORBA, Java, .NET, XMI/XML, and Web-based platforms.

As new platforms and technologies emerge, MDA enables rapid development of new specifications that use them, streamlining the process of integration. In this way, MDA goes beyond middleware to provide a comprehensive, structured solution for application interoperability and portability into the future. Creating Application and Platform Descriptions in UML provides the added advantage of improving application quality and portability, while significantly reducing costs and time-to-market.

The architecture encompasses the full range of pervasive services already specified by OMG, including Directory Services, Event Handling, Persistence, Transactions, and Security. The core logic of many of these services is already available for multiple implementation technologies; for instance, Sun's J2EE platform uses Java interfaces to CORBA's long-established transactions and security services. MDA makes it easier and faster to design similar multiple-platform interfaces to common services.

Most importantly, MDA enables the creation of standardised Domain Models for specific vertical industries. These standardised models can be realised for multiple platforms now and in the future, easing multiple platform integration issues and protecting IT investments against the uncertainty of changing fashions in platform technology" [70].

In MDA Guide [4], MDA defined as an approach to using models in software development. Although MDA is not itself a technology specification, it represents an approach and a plan to achieve a cohesive set of model-driven technology specifications [79]. It is a new way of developing software systems which is combined by three parts.

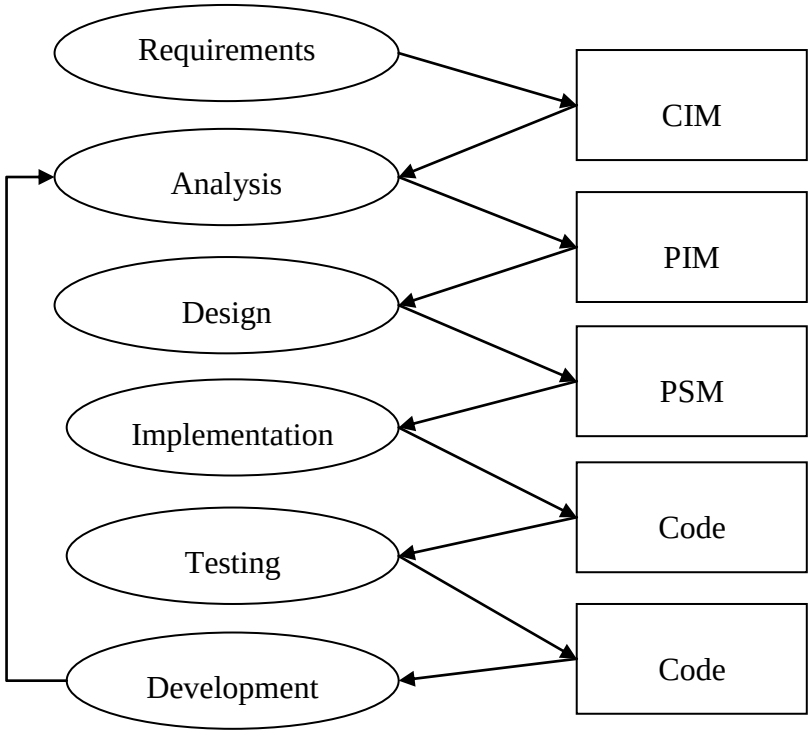


Figure 3-6. MDA Software Development Life Cycle

- **A development process**: briefly to say, MDA provides a model-driven development process including four steps, (1) Build computation independent model (CIM); (2) Build platform-independent model (PIM); (3) Generate one or more platform-specific model (PSM) from PIM using transformation tools; (4) Use transformation tools to generate specific codes. Figure 2-6 shows the corresponding software development life cycle.

MDA development process brings three benefits. (1) Productivity: developer can focus on business logic rather than technical details. (2) Portability: automated transformation tools available in different development phases and different platforms. (3) Maintenance and documentation: clarified models make maintenance and documentation easy and clear.

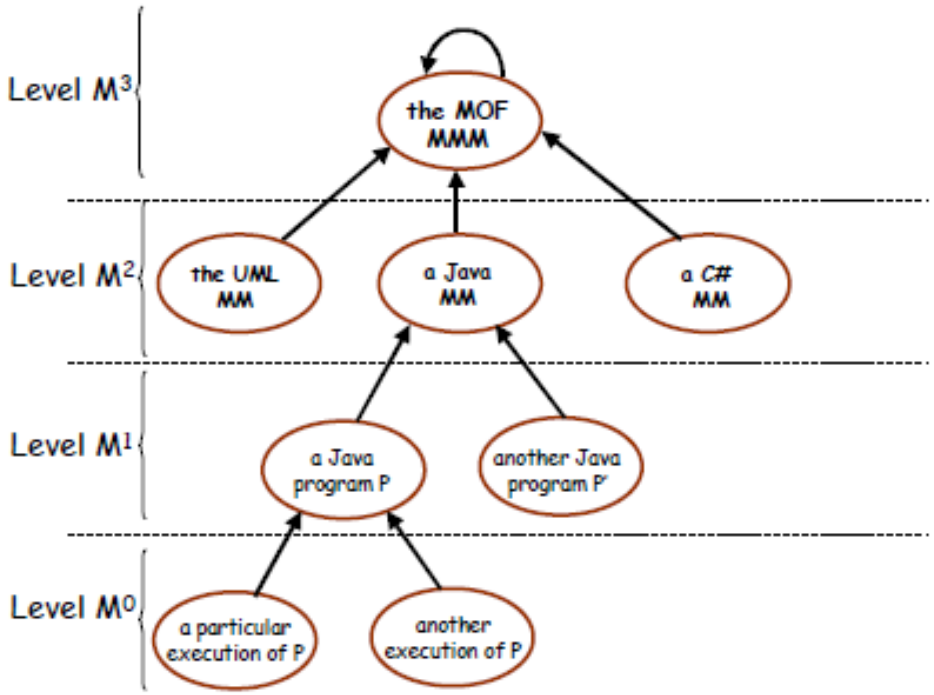


Figure 3-7. MDA Layered Architecture

- **A framework**: in MDA, a layered architecture has been defined with the following four levels which are showing in Figure 2-7.

M3: the meta-meta-model level, which only contains the Meta-Object Facility (MOF).

M2: the meta-model level, which contains any kind of meta-model including the Unified Modelling Language (UML) meta-model.

M1: the model level, which contains any model with a corresponding meta-model from M2.

M0: the concrete level, which contains any real situation, unique in space and time, described by a given model from M1.

Based on the four layers, MDA provides a software development framework which can be described as Figure 2-8. There are four blocks built in MDA framework.

(1) **Models.** It is including CIM, PIM, PSM, and source code.

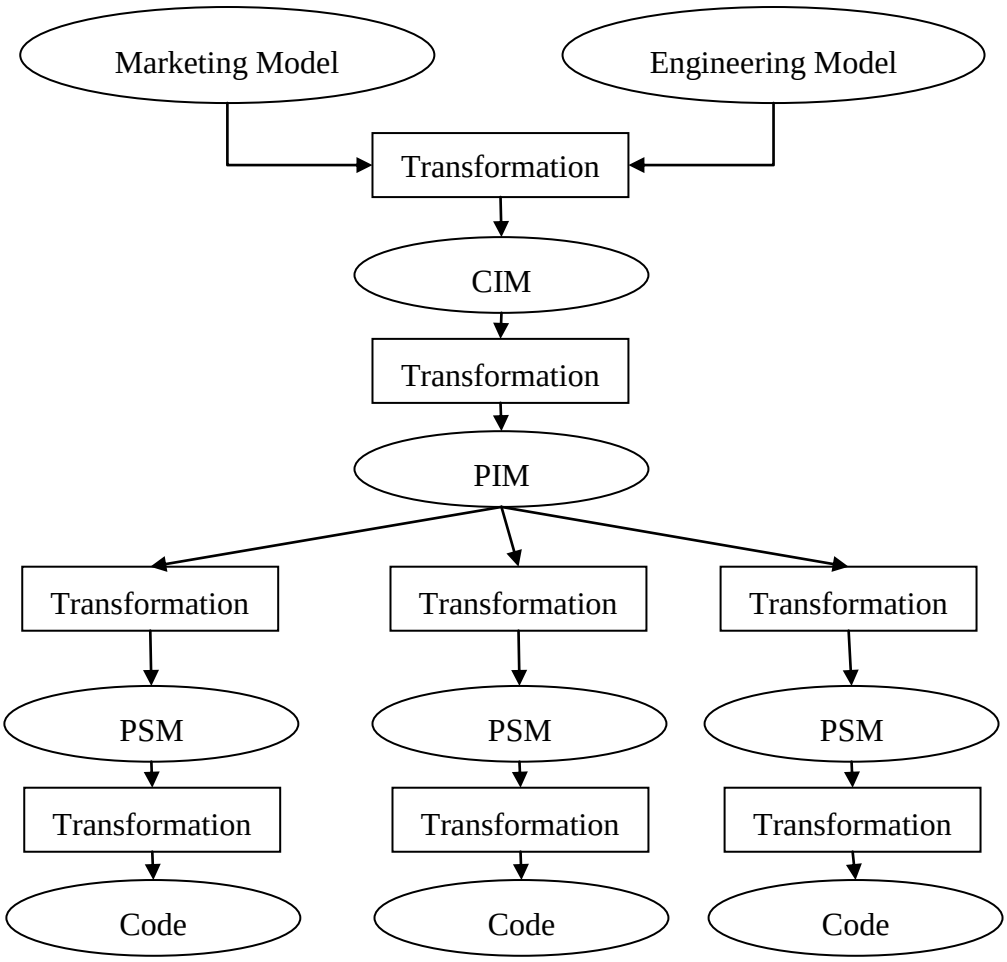


Figure 3-8. MDA Framework

(2) **Transformations.** It mainly includes "PIM to PSMs transformation" and "PSM to code transformation". Strictly, "CIM to PIM transformation" is a part of it as well.

(3) **MDA specifications.** Firstly, there are one or more standards and well-defined languages to build PIM. Secondly, there are standards and languages for PSM. Lastly, there is a language to write the definition of transformation between models.

(4) **Tools.** They are needed for implement the execution of the transformations.

- **A set of standards**: at the core of MDA there are a number of important OMG standards: the Unified Modelling Language (UML), Meta Object Facility (MOF), XML Metadata Interchange (XMI) and Common Warehouse Meta model (CWM).

These standards define the core infrastructure of the MDA, and have greatly contributed to modern systems modelling and development [4, 102].

MDA provides an approach that increases the power of models in system development by allow developers to create systems entirely with models. As the name says, it is model–driven because it provides a means for using models to direct the course of understanding, design, construction, development, operation, maintenances and modification.

3.4.3 Model-Driven Architecture Languages and Tools

All of the key parts of the MDA vision have already been standardised, including not only UML 2.0, the MOF, XMI and CWM, but also a number of middleware mappings including one to OMG's own CORBA and a middleware-independent mapping for enterprise systems called "UML Profile for Enterprise Distributed Object Computing" [71].

Even though UML is usually thought of as the basis for MDA because of its visibility, it is actually Meta-Object Facility (MOF) compliance that is formally required in order for a tool or tool chain to be labelled "MDA Compliant". The MOF is OMG's foundation specification for modelling languages; MOF compliance allows UML structural and behavioural models, and CWM data models, to be transmitted via XMI, stored in MOF-compliant repositories, and transformed and manipulated by MOF-compliant tools and code generators [71].

Although not formally required, UML is still a key enabling technology for the Model Driven Architecture and the basis for 99% of MDA development projects. (Work in some specialised fields requires specially tailored modelling languages, although the additional capabilities added to UML by the 2.0 revision satisfies this need in many

cases.) So, application development using the MDA is typically based on a normative, platform-independent UML model. By leveraging OMG's universally accepted MOF and UML standards, the MDA allows creation of applications that are portable across, and interoperate naturally across, a broad spectrum of systems from embedded, to desktop, to server, to mainframe, and across the Internet [71].

The basis on MOF was made official in August, 2004, when OMG's Object and Reference Model Subcommittee unanimously adopted the definition of MDA that is being used to revise the official MDA Guide. The August definition states [71]:

"Any modelling language used in MDA must be described in terms of the MOF language, to enable the metadata to be understood in a standard manner, which is a precondition for any ability to perform automated transformations" [71].

The main languages in MDA are briefly explained below [72] including Meta-Object Facility, Unified Modelling Language, Unified Modelling Language Profile, XML Metadata Interchange, and Common Warehouse MetaModel.

The Meta-Object Facility (MOF): In the MDA, models are first-class artefacts, integrated into the development process through the chain of transformations from PIM through PSM to coded application. To enable this, the MDA requires models to be expressed in a MOF-based language. This guarantees that the models can be stored in a MOF-compliant repository, parsed and transformed by MOF-compliant tools, and rendered into XMI for transport over a network. This does not constrain the types of models you can use - MOF-based languages today model application structure, behaviour in different ways, and data; OMG's UML and CWM are good examples of MOF-based modelling languages but are not the only ones.

The Unified Modelling Language (UML): Each MDA-based specification has, as its normative base, two levels of models: a Platform-Independent Model (PIM), and one or more Platform-Specific Models (PSM). For many specifications, these will be defined in UML, making OMG's standard modelling language a foundation of the MDA. (Use of UML, although common, is not a requirement; MOF is the mandatory modelling foundation for MDA.)

UML Profiles: UML Profiles tailor the language to particular areas of computing (such as Enterprise Distributed Object Computing) or particular platforms (such as EJB or CORBA). In the MDA, both PIMs and PSMs will be defined using UML profiles; OMG is well along our way defining a suite of profiles that span the entire scope of potential MDA applications. The current suite of profiles includes:

- The UML Profile for CORBA, which defines the mapping from a PIM to a CORBA-specific PSM.
- The UML Profile for CCM (the CORBA Component Model), OMG's contribution to component-based programming. Enterprise JavaBeans (EJBs) are the Java mapping of CCM; an initial take on a profile for EJB appears as an appendix of the UML 2.0 Superstructure specification, linked above.
- The UML Profile for EDOC is used to build PIMs of enterprise applications. It defines representations for entities, events, process, relationships, patterns, and an Enterprise Collaboration Architecture. As a PIM profile, it needs mappings to platform-specific profiles. A mapping to Web Services is underway now; additional mappings will follow.
- The UML Profile for EAI defines a profile for loosely-coupled systems - that is, those that communicate using either asynchronous or messaging-based methods. These modes are typically used in Enterprise Application Integration, but are used elsewhere as well.

- The UML Profile for Quality of Service (QoS) and Fault Tolerance defines frameworks for Real-time and high-assurance environments.
- The UML Profile for Schedulability, performance, and time supports precise modelling of predictable - that is, real-time - systems, precisely enough to enable quantitative analysis of their schedulability, performance, and timeliness characteristics.
- The UML Testing Profile provides important support for automated testing in MDA-based development environments.

XML Metadata Interchange (XMI): XMI defines an XML-based interchange format for UML and other MOF-based metamodels and models (since a metamodel is just a special case of a model); by standardising XML document formats, DTDs, and schemas. In so doing, it also defines a mapping from UML to XML. Because one of OMG's XMI updates reflects the incorporation of XML Schemas, while MOF point updates were made periodically through OMG's established maintenance process, numbering of XMI and MOF versions diverged.

Common Warehouse MetaModel (CWM): The CWM standardises a complete, comprehensive metamodel that enables data mining across database boundaries at an enterprise and goes well beyond. Like a UML profile but in data space instead of application space, it forms the MDA mapping to database schemas. The product of a cooperative effort between OMG and the Meta-Data Coalition (MDC), the CWM does for data modelling what UML does for application modelling.

In terms of products, MDA is being implemented by tools - or tool chains, which may come from a single vendor or a number of vendors - that integrate modelling and development into a single environment that carries an application from the PIM, through the PSM, and then via code generation to a set of language and configuration

files implementing interfaces, bridges to services and facilities, and possibly even business functionality [71]. Presently, there are many tools developed from both business and research areas to support model-driven development. However, it is cannot be easily defined as MDA tool if it is supporting model-driven approach. An MDA tool should be belong to one or more of the following types:

- Creation Tool: A tool used to elicit initial models and/or edit derived models.
- Analysis Tool: A tool used to check models for completeness, inconsistencies, or error and warning conditions. Also used to calculate metrics for the model.
- Transformation Tool: A tool used to transform models into other models or into code and documentation.
- Composition Tool: A tool used to compose (i.e. to merge according to a given composition semantics) several source models, preferably conforming to the same meta-model.
- Test Tool: A tool used to "test" models as described in Model-based testing.
- Simulation Tool: A tool used to simulate the execution of a system represented by a given model. This is related to the subject of model execution.
- Metadata Management Tool: A tool intended to handle the general relations between different models, including the metadata on each model (e.g. author, date of creation or modification, method of creation (which tool? which transformation? etc.) and the mutual relations between these models (i.e. one meta-model is a version of another one, one model has been derived from another one by a transformation, etc.)
- Reverse Engineering Tool: A tool intended to transform particular legacy or information artefact portfolios into full-fledged models.

Some tools perform more than one of the functions listed above. For example, some creation tools may also have transformation and test capabilities. There are other tools

that are solely for creation, solely for graphical presentation, solely for transformation, etc.

One of the characteristics of MDA tools is that they mainly take models (e.g. MOF models or meta-models) as input and generate models as output. In some cases however the parameters may be taken outside the MDA space like in model to text or text to model transformation tools.

Implementations of the OMG specifications come from private companies or open source groups. One important source of implementations for OMG specifications is the Eclipse Foundation (EF). Many implementations of OMG modelling standards may be found in the Eclipse Modelling Framework (EMF) or Graphical Modelling Framework (GMF), the Eclipse foundation is also developing other tools of various profiles as GMT. Eclipse's compliance to OMG specifications is often not strict. This is true for example for OMG's EMOF standard, which Eclipse approximates with its ECORE implementation. More examples may be found in the M2M project implementing the QVT standard or in the M2T project implementing the MOF2Text standard. Several vendors already provide tools that support integration at about this level, including substantial code generation. OMG has collected links to MDA products and vendors that we know about here. Today's tools are not supporting PIM to PSM transformation very well; because the industry got started on the second step much earlier, automation of the code generation is typically successful in many platforms [71].

Furthermore, it should be careful not to confuse of MDA tools and UML tools. Usually MDA tools focus rudimentary architecture specification, although in some cases the tools are architecture-independent or platform independent. This distinction can be made more general by distinguishing “variable metamodel tools” and “fixed metamodel tools”. A UML CASE tool is typically a “fixed metamodel tool” since it

has been hard-wired to work only with a given version of the UML metamodel. On the contrary, other tools have internal generic capabilities allowing them to adapt to arbitrary metamodels or to a particular kind of metamodels.

3.4.4 Model Driven Architecture and Transformation Today

The MDA is a new way of developing applications and writing specifications, based on a platform-independent model (PIM) of the application or specification's business functionality and behaviour. A complete MDA application consists of a definitive PIM, plus one or more PSMs and complete implementations, one on each platform that the application developer decides to support.

In order to benefit an industry, a standard must be used by a critical mass of companies. Technology-specific standards will have trouble getting established where platform incompatibility prevents achieving this critical mass. Sometimes the problem is even deeper than this: In some industries, architecturally excellent standards have been adopted in the formal sense but failed to gain hold because they were written for a platform that few companies were willing to support [71]. MDA aims to solve these issues. Under MDA, the functional description of every standard is technology independent, and the architecture is capable of producing interoperating implementations on multiple platforms. This allows an industry to define the business functionality and behaviour of its standards as a PIM, and then produce PSMs and implementations on whatever platforms the participants require.

In addition, technology-based standards become obsolete as their base platform ages; in fact they lose some of their appeal as soon as a new platform gets some play in the industry press. This is not a problem for MDA-based specifications: Because they are based on platform-independent PIMs and can be made to interoperate with new

platforms, or even re-implemented on them, through the MDA development process, MDA-based applications and specifications.

There are many advantages and benefits to using the MDA approach. Follows are the important there of them [71]:

- In an MDA development project, attention focuses first on the application's business functionality and behaviour, allowing stakeholders' investment to concentrate on the aspects that critically affect core business processes. Technical aspects, also critical but secondary to business functions, are well-handled by automated or semi-automated development tools.

- An architecture based on the MDA is always ready to deal with yesterday's, today's and tomorrow’s "next big thing" and makes it easier to integrate applications and facilities across middleware boundaries.

- Domain facilities defined in the MDA by OMG's Domain Task Forces will provide much wider interoperability by always being available on a domain's preferred platform, and on multiple platforms whenever there is a need.

Recent years, OMG established a domain facility-OMG's Domain Task Forces. It started to write their specifications, especially transformation rules, in the MDA in mid-2001, led by the Life Science Research Domain Task Force, working in Biotechnology, which was the first Domain Task Force to modify its Mission and Goals Statement to reflect its work in MDA [72]. Currently, there are many domains are covered as listed below with its specific projects [69],

- Transportation. It includes Air Traffic Control (ATC) and Surveillance User Interface (SURV).

- Command, Control, Communications, Computers, Intelligence (C4I): Includes Alert Management Service (ALMAS), Shared Operational Picture Exchange Services (SOPES), Information Exchange Data Model (IEDM), Unified Profile for the Dept. of Defence Architecture Framework (DoDAF) and the Ministry of Defence Architecture Framework (MODAF), and Application Management and System Monitoring for CMS Systems (AMSM).

- Telecommunications: It includes Audio / Visual Streams (AVSTR) and Telecom Service & Access Subscription (TSAS).

- Life Sciences Research. It includes Bibliographic Query Service (BQS), Bimolecular Sequence Analysis (BSA), Genomic Maps (GMAPS), Laboratory Equipment Control Interface Specification (LECIS), Phenotype and Genotype Object Model (PAGE-OM), Macromolecular Structure (MACSTR), Life Sciences Analysis Engine (LSAE), Life Sciences Identifiers (LIS), Chemical Structure and Access Representation (CSAR), and Gene Expression (GENE).

- Business: It includes Business Motivation Model (BMM), Business Process Definition Metamodel (BPDM), Production Rule Representation (PRR), Business Process Maturity Model (BPMM), and Business Process Modelling Notation (BPMN).

- Healthcare: It includes Clinical Decision Support Service (CDSS), Lexicon Query Service (LQS), Person Identification Service (PIDS), and Identity Cross-Reference Service (IXS).

- Manufacturing and Industrial Systems: It includes Computer Aided Design Services (CAD), Historical Data Acquisition from Industrial Systems (HDAIS), and Data Acquisition from Industrial Systems (DAIS).

- Finance: It includes Currency (CURR), Party Management Facility (PARTY), and General Ledger (LEDG).
- Simulation: It includes Distributed Simulation Systems (DSS).
- Space: It includes Ground Equipment Management Service (GEMS), Space Operations Language Metamodel (SOLM), and Reference Metamodel for the EXPRESS Information Modelling Language (EXPRESS).
- Cross-domain: It includes IT Portfolio Management Facility (ITPMF) and Task and Session (TSKSES).
- Government: It includes Metamodel for the Federal Transition Framework (FTF), Records Management Services (RMS), and Model for Performance-driven Government (MPG).
- Electronic Commerce: It includes Negotiation Facility (NEG) and Public Key Infrastructure (PKI).
- System Engineering: It includes OMG Systems Modelling Language (OMG SysML).
- Software-based Communications: It includes PIM and PSM for Smart Antenna (smart ant).
- Robotics Software Development: It includes Robotic Localisation Service (RLS) and Robotic Technology Component (RTC).

Generally, model transformation is a process of converting one type model into another type. Predefined rules are designed as the kernel part of this process. In MDA, transformation is mainly between PIM and PSM. Vertically, it includes

transformation from CIM to PIM, PIM to PSM, and PSM to code whereas horizontal transformations includes transforming a PSM into another PSM based upon implementing technologies [59, 93]. However, the important position of transformation rules is the same.

To generate PSM from PIM, different artefacts of the system are required to be mapped from one model to another. Therefore, it is necessary to formulate a set of transformation criteria that allows converting a source model in to the target model [93]. In last decade, there are many model transformation approaches have been proposed by researchers based on various requirements or platforms. For example, Kleppe [40] has provided mapping rules for generating EJB specific PSM. Also, there are a few works indirectly proposed a graphical notation for model transformation such as papers from Mazon et al [56, 57, 88]. There are two examples shown in this section to demonstrate the notation of transformation rules. First example is a transformation to get multidimensional data warehouse model with generalisation relationship from relational model which is shown as table 3-3 [57]. Next table presents the second example which is a transformation of Object Constraint Language (OCL) navigation call expression to various Prototype Verification System (PVS) expression.

Well-Formedness Rules	**Explanation**
∀se:SourceElement(instanceOf(se,SourceMetamodel))	All SourceElement must be instance of SourceMetamodel
∀re:ResultElement(instanceOf(re,ResultMetamodel))	All ResultElement must be instance of ResultMetamodel
∀se:SourceElement, n:Name (owned(se,n) ⇒ empty(n) ∨ notEmpty(n))	SourceElement's name can have a value or empty

∀se:ResultElement, n:Name (owned(re,n) ⇒ empty(n) ∨ notEmpty(n))	ResultElement's name can have a value or empty
∀tl:TransformationLink (∃ ms: MappingStatement (partOf(ms,tl)))	All TransformationLinks must have a MappingStatement
∀tl:TransformationLink, as:ActionStatement(partOf(as,tl) ⇒ (∃ gs:GuardStatement (partOf(gs,tl))))	ActionStatement can only appear in a TransformationLink that have a GuardStatement

Table 3-3. Explanation of Well-Formedness Rules [57]

Semantics	**Explanation**
∀ tl:TransformationLink, c:Condition (partOf(c,tl) ∧c=TRUE ⇒ execute(tl))	For all transformation link that have a condition, the transformation will happen if the condition is true
∀ tl:TransformationLink, ms:MappingStatement, gs:GuardStatement (gs=TRUE ∧ partOf(gs,tl) ∧ partOf(ms,tl) ⇒ exexute(ms) ∧ (∃ as:ActionStatement (partOf(as,tl) ⇒ execute (as))))	If GuardStatement true, execute MappingStatement and ActionStatement that is for the same TransformationLink as the GuardStatement

Table 3-4. Some of The Semantics for The Proposed Notation [57]

3.5 Software Engineering Creative Computing

Previous sections already demonstrated a relatively tight connection between software engineering and creative computing. Therefore, in this part, we would like to propose a method for software engineering creative computing, which is a solution on "how to develop creative computing as an academic discipline". Normally, it is a domain skill that all kinds of software have to be developed under the guidance of Software Engineering theory and methods. In this section, we are proposing a term, Software

Engineering Creative Computing (SECC), to promote a Software Engineering thinking helping with the development of creative computing, as an academic discipline.

3.5.1 General Discussion

Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software [12]. The most general view of software engineering is that software engineering is the analysis, design, construction, verification and management of technical (or social) entries [86]. Regardless of the entity that is to be engineered, it is answering follow questions [106]:

- What is the problem to be solved?
- What are the characteristics of the entity that is used to solve the problem?
- How will the entity (and the solution) be realised?
- How will the entity be constructed?
- What approach will be used to uncover errors that were made in the design and construction of the entity?
- How will the entity be supported over the long term, when corrections, adoptions and enhancements are requested by users of the entity?

Because characteristics of creative computing, there are some new development's issues generated which actually can be catalogued into above "What/How" questions' list. For example, in creative computing, "multimedia data" feature leads to a data structure problem which can be solved by software engineering's answer with "How will the entity be constructed?" question.

To sum up, software engineering creative computing can be a roadmap in development of creative computing in academic research and will help to enhance into a systematic disciplined, quantifiable development approach in industrial practice.

3.5.2 Classification of Software Engineering Principles

Software engineering can be divided into ten sub disciplines. They are [12]: Software requirements; Software development; Software maintenance; Software configuration management; Software engineering management; Software development process; Software engineering tools; and Software quality. Each sub discipline contains specific principles. Specifically, all the existing Software Engineering principles can be classified into four huge catalogues as Pressman [86] proposed,

- Managing software projects. It contains Software Process and Project Metrics, Software Project Planning, Risk Analysis and Management, Project Scheduling and Tracking, Software Quality Assurance, and Software Configuration Management.

- Conventional methods for software engineering. It contains System Engineering, Analysis Principles and modelling, Design Principles (including Architectural Design, User Interface Design, and Component-Level Design), and Software Testing Techniques and Strategies.

- Object-Oriented software engineering. It contains Object-Oriented Principles, Object-Oriented Analysis, Object-Oriented Design, and Object-Oriented Testing.

- Advanced Topics in software engineering. It contains Formal Methods, Cleanroom software engineering, Component-Based software engineering, Client and Server Ecommerce software engineering, Web Engineering, Reengineering, etc.

3.5.3 Software Engineering Creative Computing Application in an E-Learning System

In the previous research, there is a music-learning system [101, 102] involved as an experimental case, which is a creative computing project. It works as follows: there is a three-way approach that is to be presented interdependently. It consists of a "section" concerning music appreciation, one focusing on the understanding of musical, theoretical and technological concepts and another involved with music making. The heart is the understanding section as any tutor or learner-driven navigation starts here as all key terms and concepts are embedded in this section. Nonetheless, as the music type is not as well-known as, say, certain forms of commercial music, a didactic approach to repertoire development and learning what to listen for is essential. This takes place in the listening section and communicates directly with and is dependent on the related concepts in the understanding section. When it comes to how this music is made, certain specific means of sonic treatment will be introduced in the understanding section, for example, how a filter can alter the timbral quality of a given sound. To this end, some individual creative components are integrated, such as a real-time sound manipulation tool which is working online and allow multi-users' cooperation.

From above, this music-learning system is in the creative computing area. It mainly shows as following features (compared with characteristics of creative computing): Multimedia presentation, Art-technology collaboration, User cooperation, Multimedia data, and Technology innovation.

- Multimedia presentation. There are music, text, picture, flash and other presentations.

- Art-technology collaboration. This is a combination of music learning (with composition) and e-learning technology.

- User cooperation. There are multi-user's composition, connections between learner and tutor, and learner's cooperation with knowledge share and communications,

- Multimedia data. There are text, music/sound file, flash, picture, composition tool, etc.

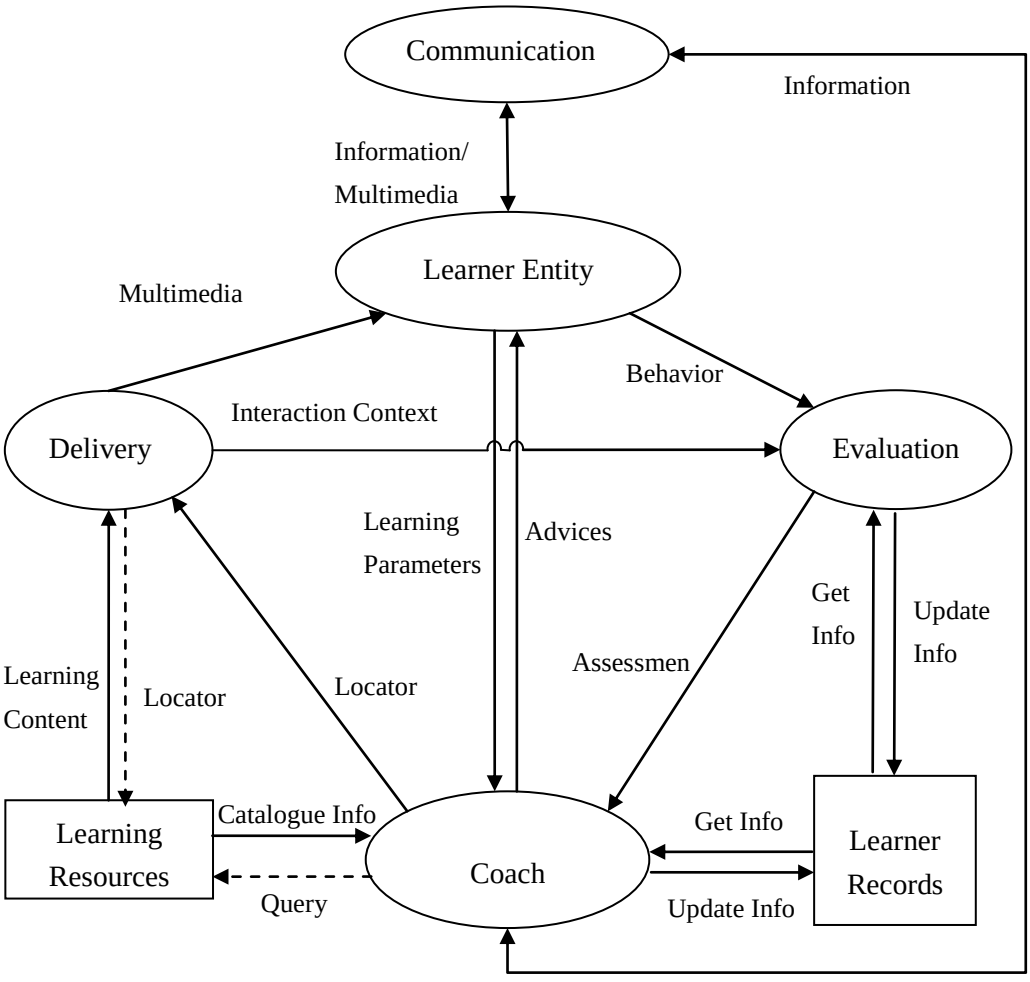


Figure 3-9. Component Structure of Interaction Learning Model

The first four features are discussed in last section that they can be engineered based on classic SE principles. Because the proposed “software engineer creative computing” concept is always applied in the research development, Figure 6-5, as a proposed learning model, shows the feasibility and applicability of proposed principles in section 6.3.

This components model contains some software engineering principles, such as “user interface design golden rules”, “data design principles”, and “metrics for object-oriented design class”.

- User Interface Design Golden Rules. When design components for user and their relative actions, the golden rules are concerned. Therefore, users are divided into two components, “Learner Entity” and “Coach”. Also, because of “Reduce user memory load” and “Make the interface consistent” rules, another component named “Delivery” is designed to manage presentation information.

- Data Design Principles. The “Learner Records” and “Learning Resources” are designed based on this principle, which makes them been divided with each other and also classifies methods between them and other components.

- Metrics for Object-Oriented Design Class. Although this model only shows components, this principle is applied in its design because we planned our development based on Object-Oriented. Classes Model will be mapped from this components model. Therefore, each component is calculated by those metrics.

Above all, the principles are working efficiency in the models designed for e-learning system.

3.6 Related Projects

The proposed research is trying to build a MDA-based software evolution framework, in which software modelling into CIM, PIM, and PSM, and transformation are designed following MOF QVT standard in order to facilitate software engineering and software evolution. Hence, there are a great range of related projects that this study has reviewed and discussed.

Gavras et al [27] have proposed an MDA-based development methodology. Applying MDA to enterprise computing have described in [24]. In [13] author has provided model driven software modernisation. They proved the practicability to apply MDA in general systems' development. Even in software evolution, MDA is an effective methodology [14]. However, those work mentioned above mainly focus on MDA's application but weak on PIM's establishment. Though Solms and Loubser [94] formulated a methodology to generate PIM, it aims at the system domain experts but not software technicians.

There are also researchers focusing on electronic learning with many results. Lots of researchers [5, 21, 38, 54, 58, 103, 105] have been working on learning content and part of the works focused on multimedia content management and delivery such as [54, 58]. Meanwhile, lots of researchers [15, 20, 25, 29] are working on software engineering issues. Indeed, there are some works covered both e-learning and software engineering. For example, in [109], the author has combined the areas of media streaming services, mobile devices, and manufacturing processes to a e-learning streaming framework. Especially, few studies are covering modelling theory and e-learning such as [108], which has proposed a multi-model ontology-based framework. However, it is rare to see research on multimedia electronic learning system combined with Model-Driven Architecture.

Previously, there are several works published related this research. The initial design for MDA-based development for a music learning system has been provided as [102], including lifecycle and pedagogical design. In [101], an ontology-based model-driven approach is proposed and applied in an electronic learning system, in which PIM generation method is designed. Besides, creative computing and software engineering is studied as [106]. In this paper, our proposal is based on those published works and can be considered as previous works' extension, improvement, and specification.

3.7 Summary

In this chapter, the related works of Model-Driven Architecture based evolution are introduced and discussed including technologies and projects:

- A brief overview of software engineering is introduced. Besides, three modern software engineering paradigms are reviewed including object oriented programming, component based development, and service oriented architecture.

- Software changes are introduced with three different maintenance activities, namely, maintenance, reengineering, and evolution. Additionally, the laws of software evolution are quoted to describe software evolution.

- Model driven engineering is introduced from background and basic concepts. Model driven architecture is summarised from three views. Meanwhile, current situation of MDA is analysed from PIM languages, PSM languages, transaction definitions, tools, and other standards. Related basic terms are also explained. Besides, a brief introduction to model-driven architecture based software evolution is given. MDA approach and methods are reviewed.

- UML is discussed with UML 2.x concept, UML and MDA relationship, and UML profile. UML structure, diagram, and elements are introduced. UML modelling tools are presented. Additionally, there is a discussion on UML's role in MDA approach to explain how UML 2 supports the most prominent concepts in the evolving MDA vision. Furthermore, UML profile specification and its application for particular domains and platforms are presented. In the meantime, there is an overview of XML, XMI and PHP with brief introductions on their background, definition, application, and features.

- Projects related to the MDA are reviewed and discussed. Existing works proved the practicability to apply MDA in general systems' development. Even in software evolution, MDA is an effective methodology. However, those works mainly focus on MDA's application but weak on PIM's establishment.

- E-learning related research results are reviewed and discussed. General researchers are working on content structure, management, and delivery. Few projects are covering e-learning and modelling theory. However, it is rare to see research focusing on multimedia electronic learning system combined with Model-Driven Architecture.

- Notion of transformation rules is studied and summarised. Related previous work is concluded. Meanwhile, examples of previous work are presented including "transformations to get multidimensional data warehouse model with generalisation relationship from relational model" and "transformation of Object Constraint Language (OCL) navigation call expression to various Prototype Verification System (PVS) expression".

Chapter 4
Proposed E-Learning Modelling

Objectives

- To present domain modelling for e-learning
- To employ LTSA standard into modelling
- To specific designed three models in domain modelling
- To present models with MVC structure
- To propose an e-learning domain framework

4.1 E-Learning Domain Modelling

4.1.1 E-Learning Standard: Learning Technology Systems Architecture

The standard of Learning Technology - Learning Technology Systems Architecture (LTSA) [35], specifies a high level architecture for information technology-supported learning, education and training systems that describes the high-level system design and the components of five refinement layers of architecture are specified in Figure 4-1. This architecture is applicable to a broad range of learning scenarios [35]. From the highest to the lowest level, these layers are named as below [35]:

- Learner and Environment Interactions (informative).

- Learner-Related Design Features (informative).

- System Components (normative).

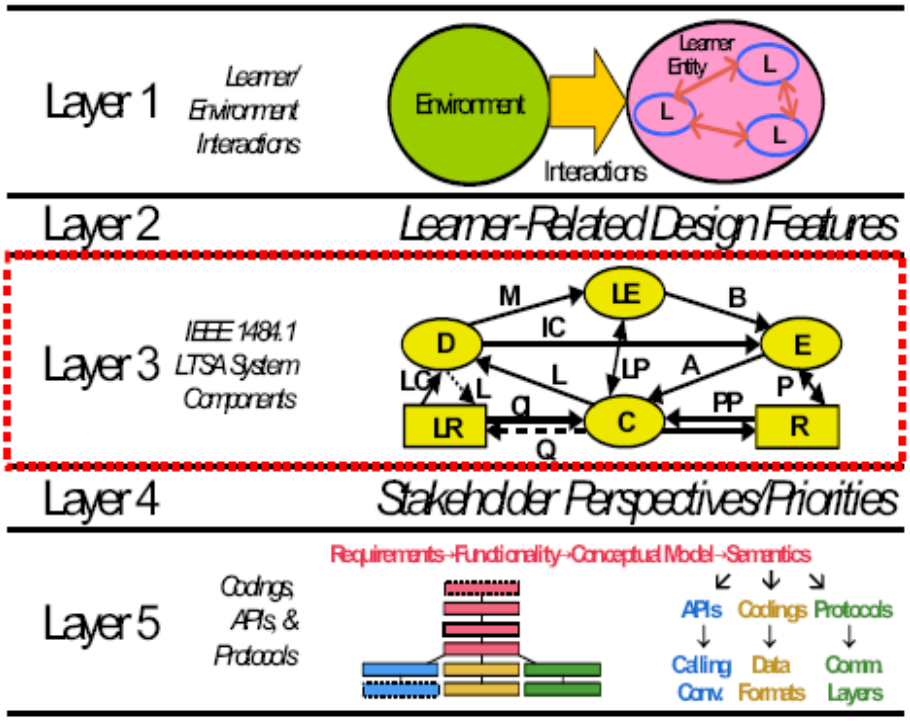


Figure 4-1. The LTSA Abstraction-Implementation Layers [35]

- Implementation Perspectives and Priorities (informative).

- Operational Components and Interoperability - coding, APIs, protocols (informative).

Concretely, the LTSA [35] identifies four processes: learner entity, evaluation, coach, and delivery process; two stores: learner records and learning resources; and thirteen information flows among these components: behavioural observations, assessment information, performance and preference information (three times), query, catalogue

info, locator (twice), learning content, multimedia, interaction context, and learning preferences. The System Core Component organisation is as follow:

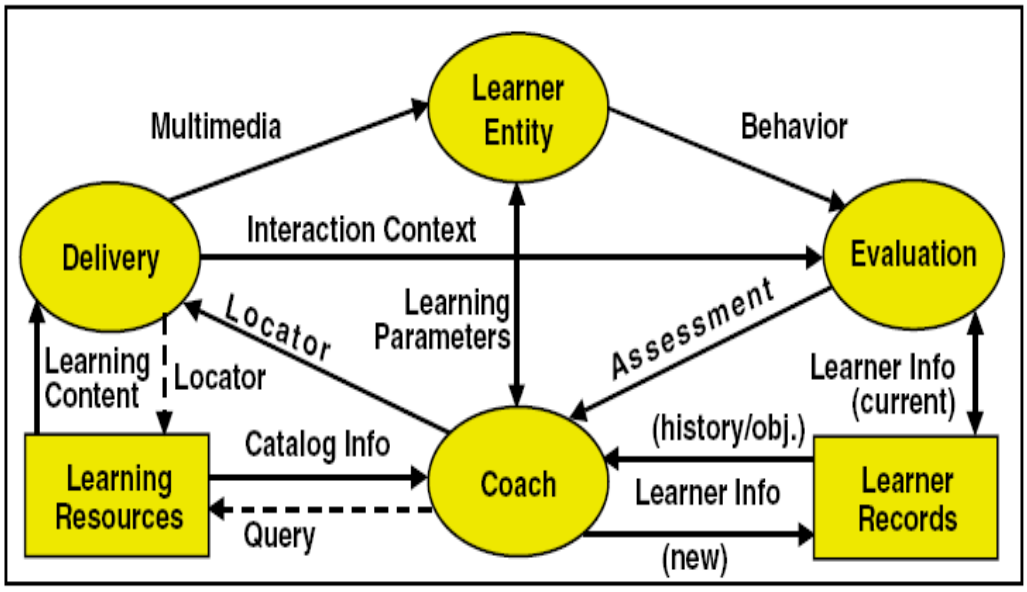


Figure 4-2. The LTSA System Components [35]

The LTSA system showing following features [35].

- Multiple role learning, team learning. Learners (of the collective learner entity) operating as teams in which learners have different roles. The interface to the learner entity is multimedia. The communication among the learners that represent the collective learner entity. There are support tools delivering via multimedia. A coach role is aiming to support collaboration activities.

- Collaboration and asynchronous learning. Learners are able to access the learning environment and/or collaborate at different times. Collaboration among the learners that present the collective learner entity. Collaboration among learners in different "time zones" or asynchronous access.

- Learner profiles. Learner information, such as performance information, preference information, and other important information are saved as learner profiles.

The proposed new version learning application is totally independent of any underlying platform. Therefore, it will follow the guidelines proposed in MDA [62]. Models of LTSA system will be structured explicitly into Platform Independent Models (PIMs) and Platform Specific Models (PSMs).

4.1.2 Pedagogical Strategy

This research is focus on educational e-learning system. Therefore, pedagogical strategy [16, 36, 50, 55] is the kernel methodology to support our requirement analyse. The kernel requirement based on the pedagogical strategy can be described as follow.

The pedagogical strategy that is being modelled is a holistic one. It works as follows: there is a three-way approach that is to be presented interdependently. The holistic approach that integrates concept development, listening and creative aspects related to electroacoustic music is one of the unique aspects of this project. Three more interesting aspects of the system are worthy of introduction. First of all, users will be entitled to have diagnostic information presented either to teachers or to the learners themselves. This information helps users to find out where their strengths lie and in which areas there is room for improvement. It can also signal concepts not being learned and offer advice as to how to achieve more satisfying results. This means that in any interactive aspect of the site, user reactions will be monitored and information on the acquisition of concepts presented. Applications of techniques developed in artificial intelligence should prove invaluable in this aspect of the research project.

A second aspect will be offered which has to do with user-generated feedback and content provision. To cite one example; the current EARS research site is based on a tree

structure to order the ca. 500 terms that it offers. The project presented here will continue to offer this traditional option, but will use state of the art tagging systems to offer another means of potential navigation through concept acquisition. This brings us the third important aspect.

The pedagogical environment will be able to be navigated in a variety of manners. These include:

- Previously organised paths designed by the development team will be on offer for individual users taking into account their previous knowledge. An avatar will act as teacher in such cases.
- Similarly, teachers may prefer to use one of the curriculum navigation systems on offer.
- The teachers may prefer to organise their own curriculum navigation system based on classroom needs.
- Individuals may be interested in specific subjects or aspects of electroacoustic music and therefore may want to create their own navigation.

All of the above will be implemented within the pedagogical architecture of this environment.

4.1.3 Three Models

Considered the requirement, the main functions are organised and designed into three groups, which present three individual learning approaches. Therefore, there are three "learning-models" are proposed, including "step by step learning model", "optional learning model", and "interacting learning model", to support each functional group.

The proposed three functional group and learn-models are based on pedagogical strategy and combined with LTSA. In the following sections, there are specifications for three functional groups and three learning models.

4.1.3.1 Step by Step Learning Model

This model is proposed to support the simplest learning method that is "step by step learning". This education method is known as a traditional teaching and learning process which is generally applied in class education. Learning path is the kernel issue in it which should be well designed in the beginning. Meanwhile, the learning path cannot be changed by learners. It is a process similar with reading a novel in which chapter's sequence is designed for every reader.

Obviously, this traditional education method is a typical approach in some educational process. It definitely has some advantages such as following list,

- Learning path is clear and easy to be followed. Because the path is static designed, learning progress is a predictable route. Hence, learners' only work is to follow the direction step-by-step.
- Learner records are simple information which means records are easy to manage by learner's manual update and automatic update from learning content path progress.
- Learning resources are easy to manage because of the static learning path.

However, there are many disadvantages in this method as well.

- Each learner cannot get specific set on learning path which cannot satisfy individual requirement. The best this method can do is to get a learning path designed that fit to the common requirement in an average level for the learner group.

- The successful degree in this method is more than 50% rely on the design of learning content and learning path. Which means it is taking a huge risk on data analysing.
- Changes on learning resources are not convenient because the contents are tightly bonded with static learning path. Any update for existing contents may cause path's change. Besides, any add or delete on the resources requires relevant change (add or delete) on the learning path.
- There is no guarantee for learner's successful on each learning stage. There is not including any independent assessment component. The judgement may depend on examinations on learning resources.
- There is no communication at all if it is self-study. Everyone is in an isolate learning environment which means learners cannot get effective help or advice if there is any problem they cannot solve by themselves.

There is a "step by step learning model components" figure shown as below,

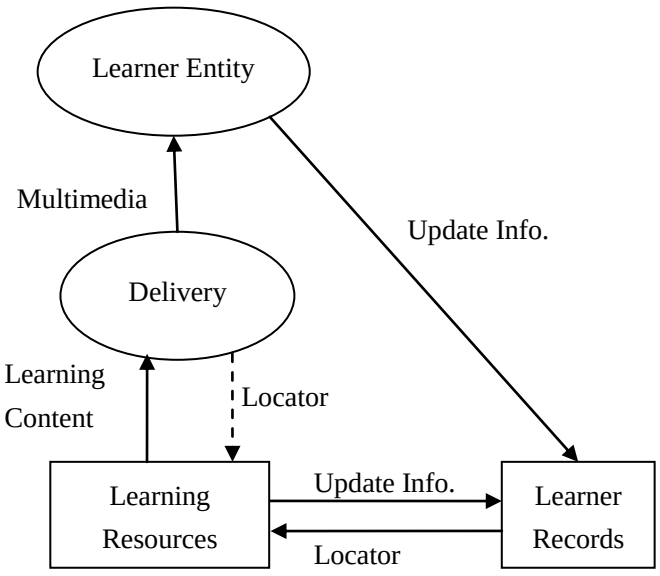


Figure 4-3. Step by Step Learning Model Components

Components' specification is explained as below:

- Learner Entity: It indicates a single learner, a group of learners learning individually, a group of learners learning collaboratively, a group of learners learning in different roles, and so on. For example, in a self-study environment, learner entity normally is individual learner. In a classroom-study or similar situation, it can be group learners learning individually, collaboratively, or in different roles.
- Delivery: It is a transmission process delivering learning content to learner entity. There are mainly two task in it, one is to receive learning content from learning resources, the other is to deliver received learning content to learner entity via multimedia. In addition, it may charges to locate learning resources aims to acquired relevant learning contents.
- Learning Resources: It contains all learning contents needed in system including various format such as text, pictures, images, sounds, and other files.
- Learner Record: It is a depository for every learner record that is necessarily to be remembered. It contains learner's basic information such as username and password. Meanwhile, learner's learning progress is a kernel part should be recorded.
- Multimedia: It indicates applications of many display methods realised by multimedia. It aims to support different formats' learning content. Besides, it can improve learners' interest via various presentation formats.
- Learning Content: It means individual file as learning content. All learning contents store at learning resources. It can be various format including text, image, sounds, and so on.
- Locator: This is a component working on learning content's location. Delivery component might need it to use it to acquire more content. It should locate learner required content from learning resources based on learning progress history in learner record.

- Update Info: It in charges to update any changes on learner's basic information and learning progress to learner record.

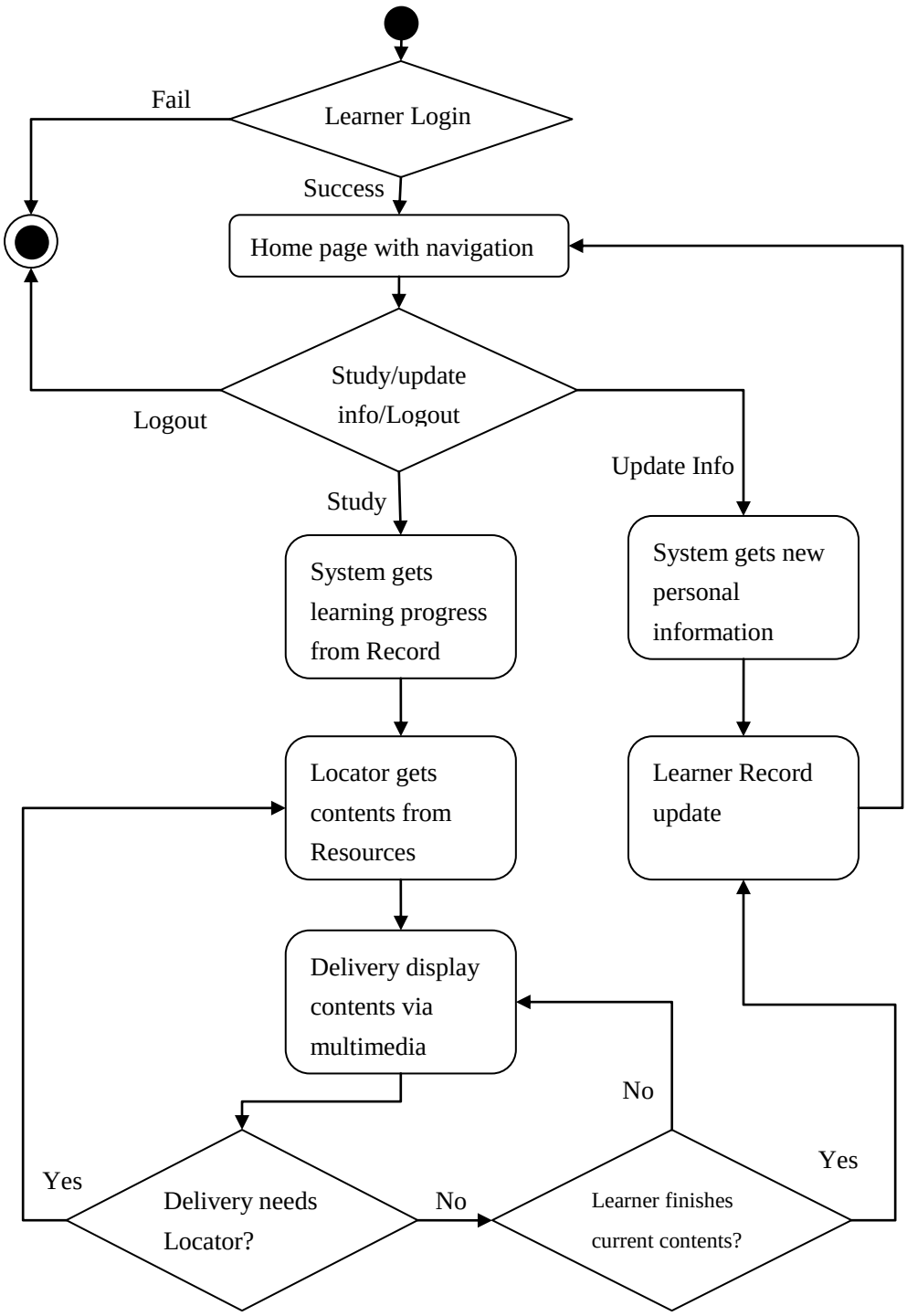


Figure 4-4. System Activity Diagram for Step by Step Learning Model

It can be seen from the figure 4-4 that the components in this model can be categorised into three catalogues,

- Processes: It contains “Learner Entity” and “Delivery” components.
- Storages: “Learner Records” and “Learning Resources” are two storages.
- Flows: It is including “Multimedia”, “Learning Content”, two “Update Info” and two “Locator”.

The user oriented working process in this model is quite clear. The Figure 4-4 is an activity diagram presents workflows of components in the step-by-step learning model.

In brief, all kinds of learning materials stored in Learning Resources. It includes text knowledge, sound materials, music example, and so on. Delivery is a process to transfer learning content into a presentation. Then, the presentation is represented by Multimedia, which delivery different type knowledge to the learner entity. Plus, the Learner Records store learner information, such as history work, study progress and so on. It is the basic data to locate where should be study.

4.1.3.2 Optional Learning Model

This model is proposed to support another learning method named as “optional learning”. This education method is a more flexible teaching and learning process which is suitable for both self-study and class-study environments. Learning path is not fixed but changeable. There is still a learning path designed on system. However, it is only a suggestion route considered general learner’s requirements. Learner entities can decide learning route by themselves via optional learning content selection. It is a process similar with reading a dictionary in which sequence is not necessarily to be followed.

This education method is a typical approach in some online self-study educational process such as The World Wide Web Consortium (W3C) Tutorials in W3Schools [3]. It definitely has some advantages such as following list,

- Learning path is flexible to change. Because the path is only designed as a suggestion, learners can create unique learning route to suit individual requirements via optional learning contents choice.
- Learner records are simple information which means records are easy to manage by learner's manual update and automatic update from learning contents' records.

However, there are some disadvantages in this method as well.

- The successful degree in this method is more rely on learner's own judgement for their finish assessment and learning path plan. It means it is taking a huge risk for rely on unpredictable user behaviour.
- Changes on learning resources are not very convenient because there is still a designed learning path bonding with learning contents. Any update for existing contents may cause path's change. Besides, any add or delete on the resources requires relevant change (add or delete) on the learning path.
- There is no guarantee for learner's successful on each learning stage. There is not including any independent assessment component. The judgement may depend on examinations on learning resources. However, learners can skip any unsuccessfully finished contents via optional learning function. It may cause failure of learning by incoherent knowledge.
- There is no communication at all if it is self-study. Everyone is in an isolate learning environment which means learners cannot get effective help or advice if there is any problem they cannot solve by themselves.

There is an "optional learning model components" shown as Figure 4-5, all the components' specifications in this model are same with specifications in last model – "step by step learning model" proposed in section 4.1.3.1.

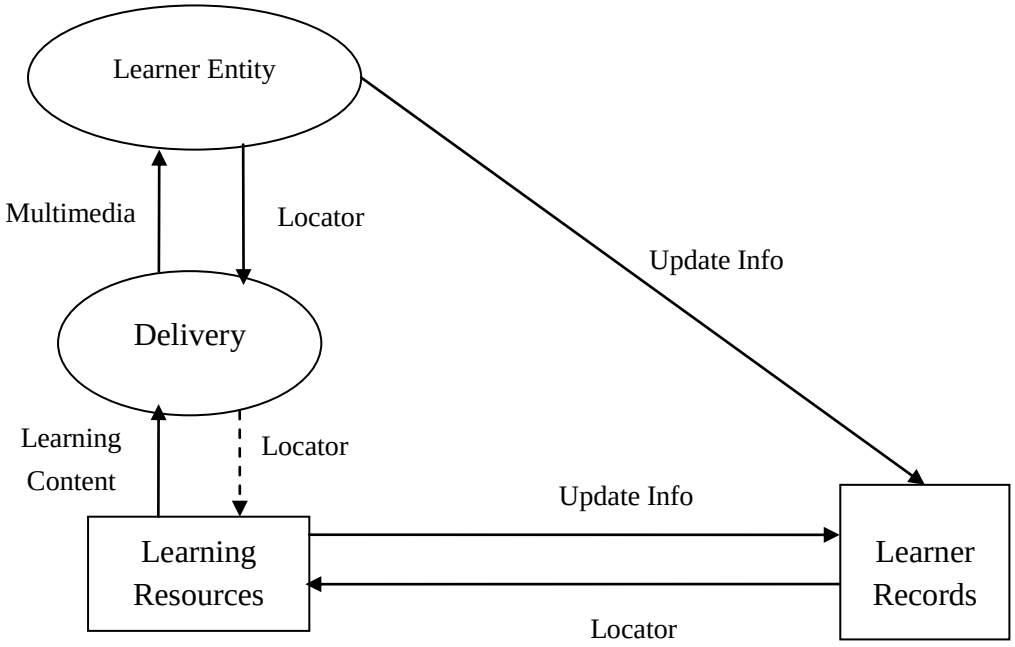


Figure 4-5. Optional Learning Model Components

It can be seen from the figure 4-5 that the components in this model can be categorised into three catalogues, which is just like last proposed learning model,

- Processes: It contains "Learner Entity" and "Delivery" components.
- Storages: "Learner Records" and "Learning Resources" are two storages.
- Flows: It is including "Multimedia", "Learning Content", two "Update Info" and three "Locator".

The user oriented working process in this model can be clearly shown as Figure 4-6 as a flowchart. It is an activity diagram presents workflows of components in the optional learning model.

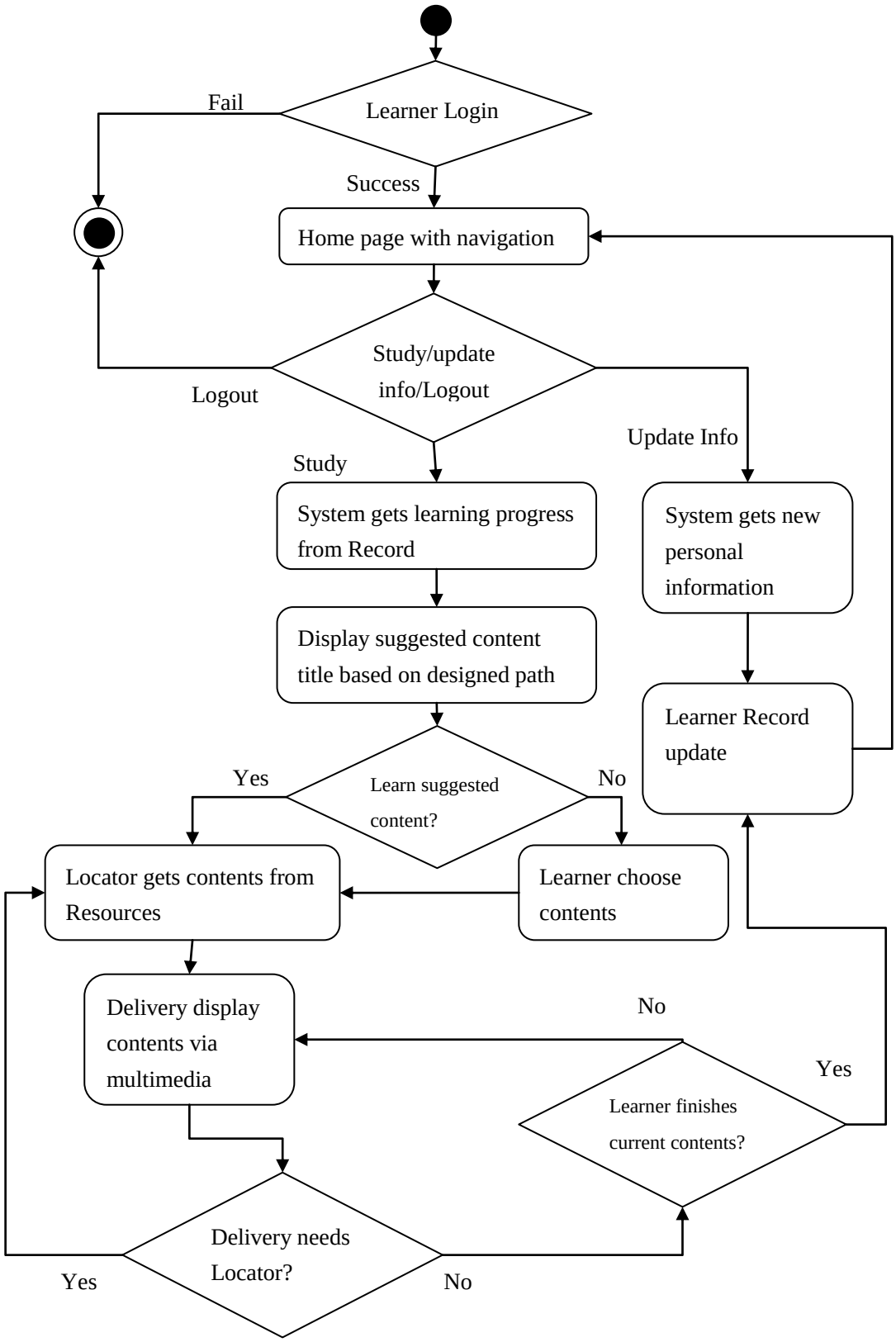


Figure 4-6. System Activity Diagram for Optional Learning Model

In brief, this model's progress is quite similar with step by step learning model. The main point in this model is that there is a locator flow comes from leaner entity. It means learners can choose learning content for themselves. In the activity diagram, it can be seen there is an option for learner to decide whether follow suggested content. The suggested content is recommended by system based on designed learning path. When learner decides do not follow suggestion, it is free to choose any content in the system as next learning stage. This is the "optional" point in this model.

4.1.3.3 Interacting Learning Model

Pedagogical knowledge includes generic knowledge about how students learn, teaching approaches, methods of assessment and knowledge of different theories about learning [28, 66, 92, 99]. It implies that it should be more considered on aspects of assessment, teaching approaches, learning process, etc.

At the meantime, pedagogical strategy proposed in section 4.1.2 is requiring a more complicated learning method to be realised. Considered about the interacting learning requirements listed in the pedagogical strategy, an interacting learning model is proposed to complete it. Therefore, an interacting learning model is proposed in the following paragraphs based on an introduced technology LTSA.

This education method is a very flexible teaching and learning process. Learning path is dynamic generated based on evaluation result and learning history record in which coach is involved as a new role to do judgement and advice for the learning path's location. Plus, coach is supporting communications with learner entity.

Based on the kernel points in this model, it definitely has more advantages than the last two proposed learning models. Here is a list of benefits below,

- Learning path is considered each learner entity's requirement. Therefore, learner entity's learning route can be unique and suitable for their specific situation.
- The successful rate in this method is more reliable based on evaluation function and coach's judgements and advices. It means learning under this model is low risk behaviour.
- Learner records are simple information which means records are easy to manage by learner's manual update, automatic update from learning contents' records, and from evaluation and coach's advices.
- Communication is realised no matter in self-study or group-study. Everyone is capable to communicate with others including other learners and coaches under any learning environment which means learners can get effective help or advice conveniently.

Compared with the other two learning models – "step by step learning model" and "optional learning model", the mainly disadvantage in this learning method is a higher system complexity. There is an "interaction learning model components" shown as Figure 4-7.

Components' specification is explained as below:

- Learner Entity: It indicates a single learner, a group of learners learning individually, a group of learners learning collaboratively, a group of learners learning in different roles, and so on. For example, in a self-study environment, learner entity normally is individual learner. In a classroom-study or similar situation, it can be group learners learning individually, collaboratively, or in different roles.

- Delivery: It is a transmission process delivering learning content to learner entity. There are mainly two task in it, one is to receive learning content from

learning resources, the other is to deliver received learning content to learner entity via multimedia. In addition, it may charges to locate learning resources aims to acquired relevant learning contents.

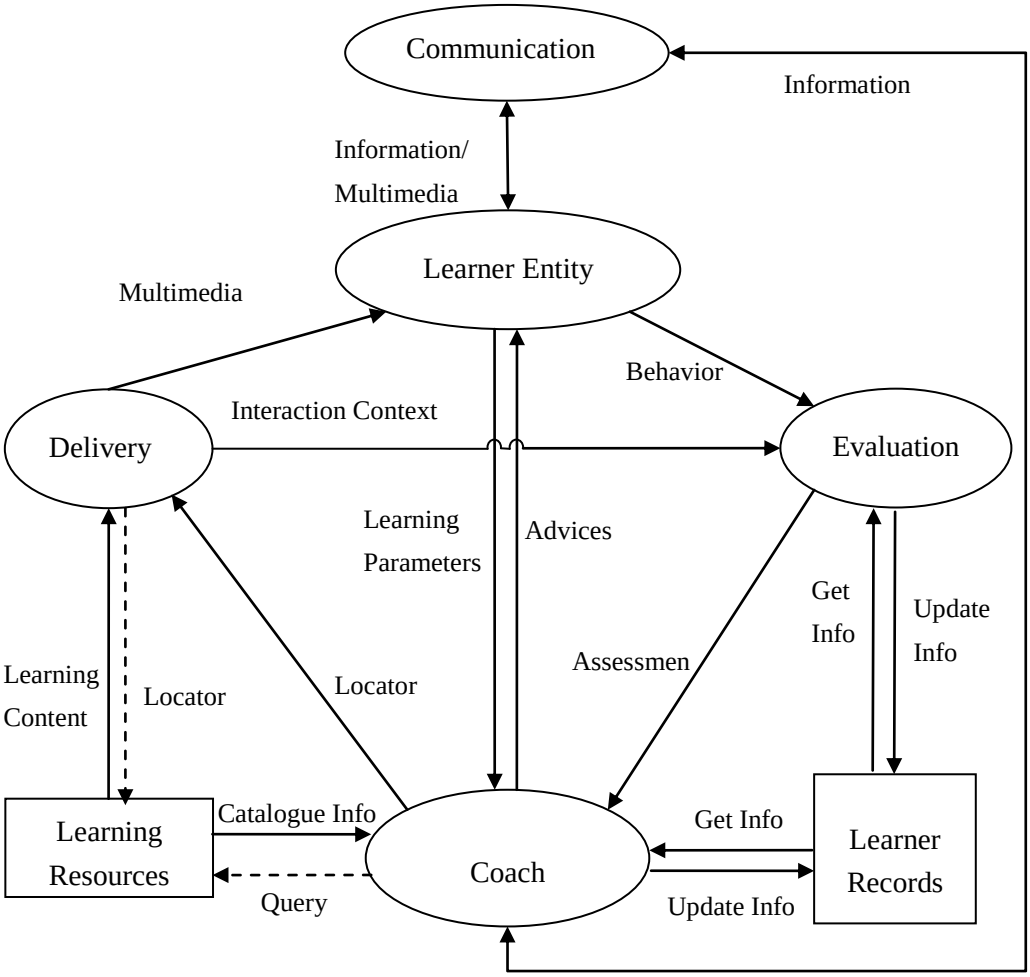


Figure 4-7. Interaction Learning Model Components

- Learning Resources: It contains all learning contents needed in system including various format such as text, pictures, images, sounds, and other files.

- Learner Record: It is a depository for every learner record that is necessarily to be remembered. It contains learner's basic information such as username and

password. Meanwhile, learner's learning progress is a kernel part should be recorded.

- Multimedia: It indicates applications of many display methods realised by multimedia. It aims to support different formats' learning content. Besides, it can improve learners' interest via various presentation formats.

- Learning Content: It means individual file as learning content. All learning contents store at learning resources. It can be various format including text, image, sounds, and so on.

- Locator: This is a component working on learning content's location. Delivery component might need it to use it to acquire more content. It should locate learner required content from learning resources based on learning progress history in learner record.

- Update Info: It in charges to update any changes on learner's basic information and learning progress to learner record.

- Get Info: It is a flow to acquire history information from learner record for evaluation and coach.

- Coach: It is a position for real or virtual tutors. Coach will analyse learner's existing information, assessment result, and learning resources, aims to support learner entity with reasonable advices with learning contents' location.

- Evaluation: It is a process to evaluate learner entity's learning achievement based on learner entity's behaviours, interaction context, and learner's history record.

- Interaction Context: It is information about current learning contents in delivery process.

- Learning Parameters: It is parameters about current learning situations come from learner entity, for example, learner entity's specific question, requirement, etc.

- Advices: It is advices that coach suggested to help learner entity. It can be advices for learner's questions or requirements, and advices for next step's learning contents.

- Behaviour: It includes every learner entities' behaviours related to evaluation such as online examination, exercises, demonstration, and so on.

- Communication: It includes public and private communication between learner entity and coach, and between different learner entities.

- Information: It is information comes from learner entity or coach for communication purpose.

- Assessment: It is evaluation result generated from evaluation process to support coach's work.

- Catalogue Info: It is catalogue information for contents in learning resources.

- Query: It is a coach's action to query about learning content's information from learning resources.

It can be seen from the Figure 4-7 that the components in this model can be categorised into three catalogues,

- Processes: It contains “Learner Entity”, “Delivery”, “Coach”, “Evaluation”, and “Communication” components.

- Storages: “Learner Records” and “Learning Resources” are two storages.

- Flows: It is including “Multimedia”, “Learning content”, two “Update info” and two “Locator”, two “Get info”, “Learning parameters”, “Interaction context”, “Advices”, “Assessment”, “Behaviour”, “Catalogue Info”, “Information”, and “Query”.

Although the working process in this model is more complex than the other two learning models, it still can be designed as a flowchart as below,

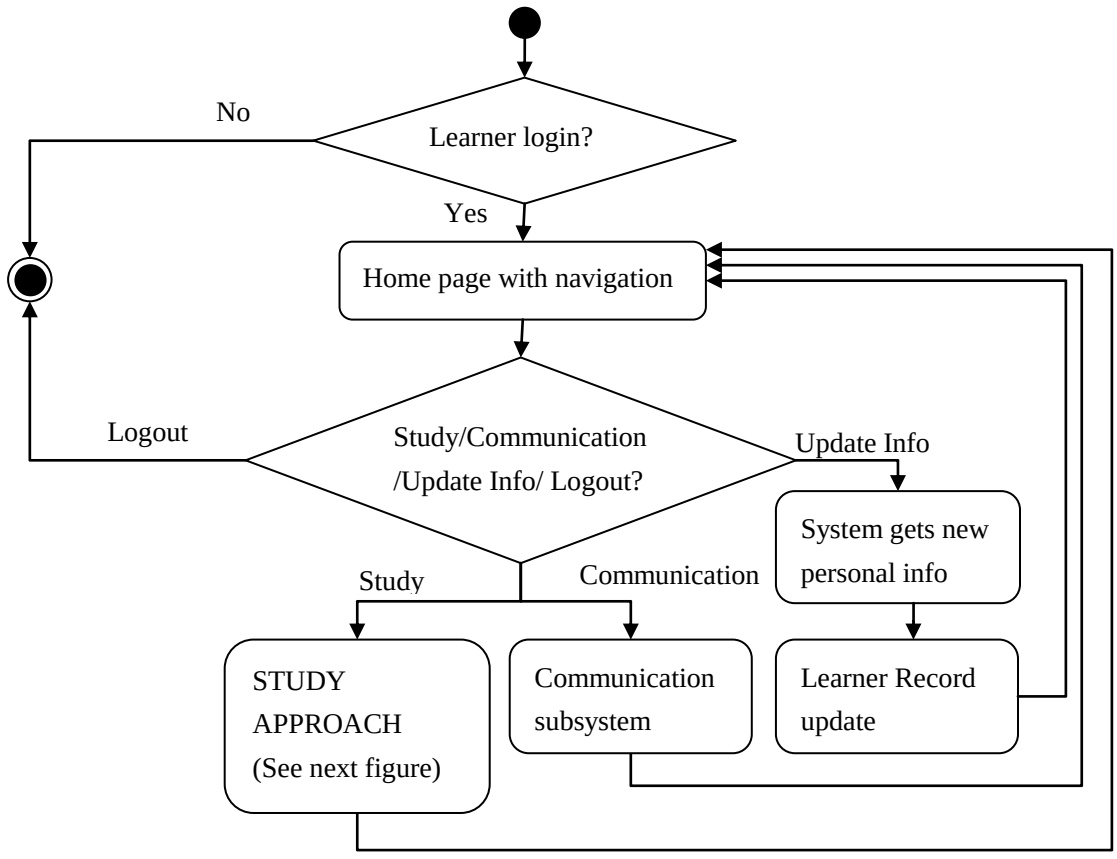


Figure 4-8. Overview System Activity Diagram for Interaction Learning Model

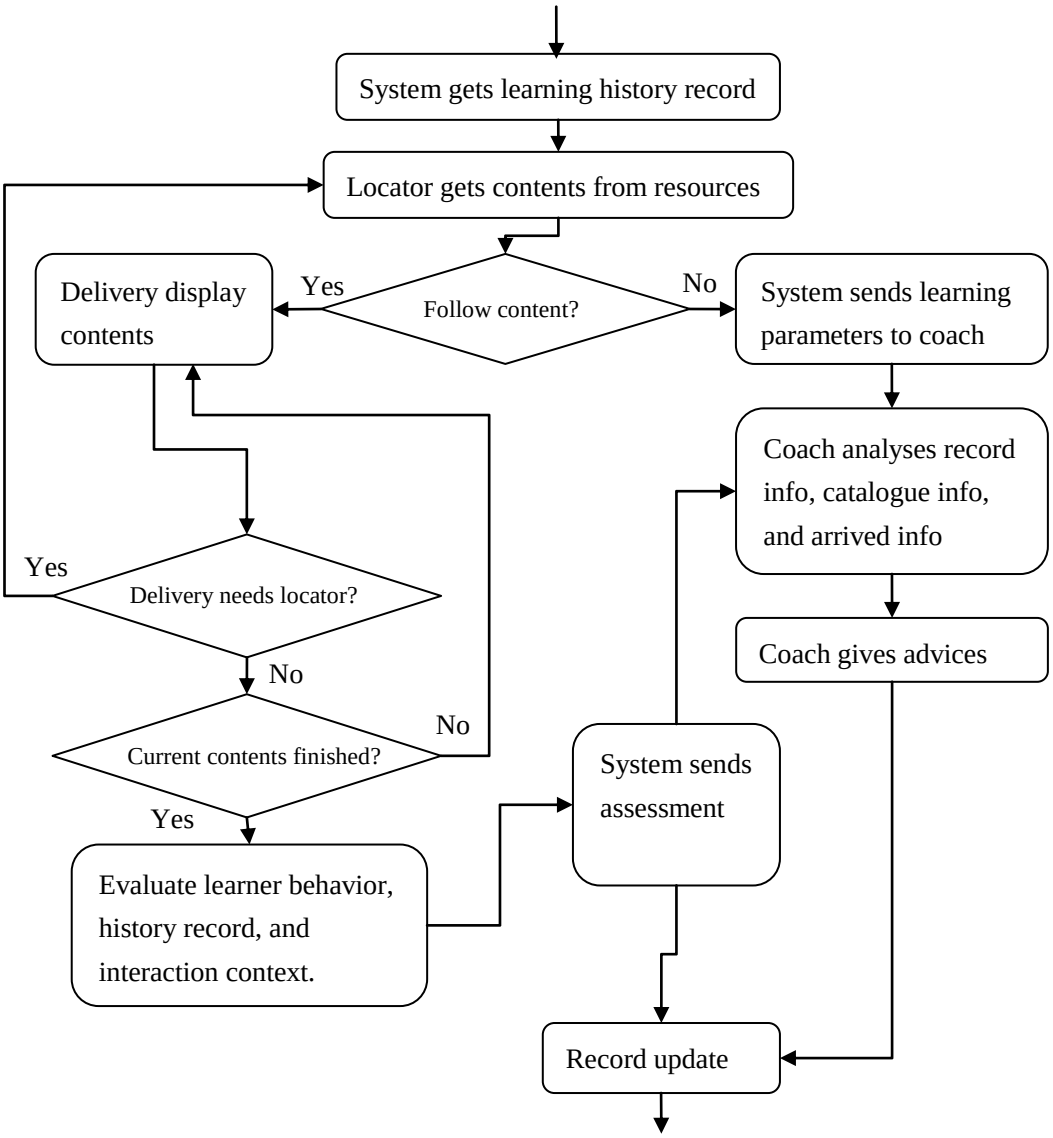


Figure 4-9. Study Approach Activity Diagram for Interaction Learning Model

The figure 4-8 is an activity diagram presents overview workflows of components in the interaction learning model. Because space limitation, specific study activities are presented as an individual diagram – figure 4-9. The two diagrams describe how the interaction learning model works.

Briefly, compare with the other two models, the key features in this model are communication, coach, and evaluation functions.

Communication is presented as an independent activity to support information exchange between leaner entities and coaches. It is free to be developed as any communication tools, for example, forum, blog, wiki, etc.

Evaluation and coach functions are mainly working in the study part. In the study approach, at the beginning, system will suggest content automatically based on learner's history record. Then it allows leaner entity to choose follow it or not. If learner has another idea other than the suggestion, learning parameters will be sent to coach which can be requirement, question, and other related information. Coach analyses parameters, learner's record, and catalogue information to make an advice for learner's request. The new advice and related information updated to learner record automatically to generate a new learning route. After those steps, learner will be suggested to new contents. When learner finishes current content, related interaction context and learner behaviours will be sent to evaluation part including test, exercises, demonstrations, examination, etc. Evaluation aims to generate an assessment based on received interaction context, learner behaviours, and learner's history record. Then, the assessment result will be saved into learner record and sent to coach to support coach's analysis.

4.2 Model-View-Controller E-Learning Modelling

Model-View-Controller (MVC) is a classic design pattern often used by applications that need the ability to maintain multiple views of the same data. The MVC paradigm hinges on a clean separation of objects into one of three categories – models for maintaining data, views for displaying all or a portion of the data, and controllers for handling events that affect the model or view. Because of this separation, multiple views and controllers can interface with the same model. Even new types of views and controllers that never existed before can interface with a model without forcing a

change in the model design. Therefore, MVC is selected as the main design pattern for presented e-learning modelling.

4.2.1 Model-View-Controller Modelling

There are three modules in MVC pattern: Model, View, and Controller. Figure 4-1 shows the basic associations between them.

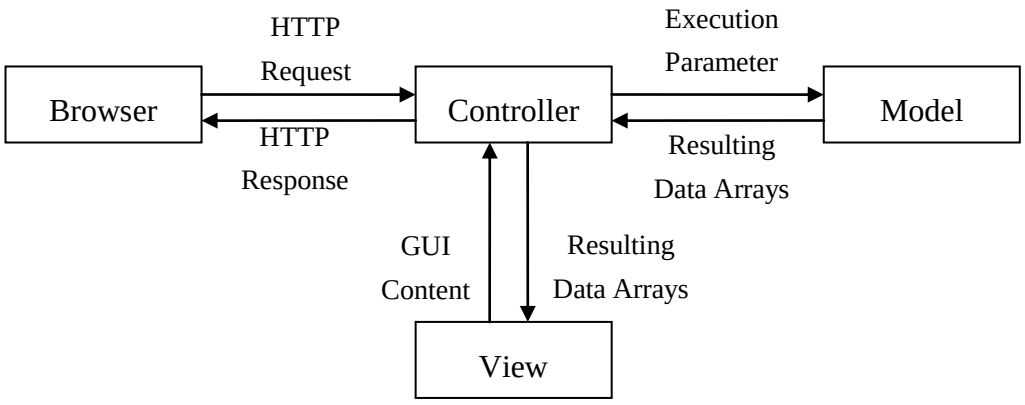


Figure 4-10. MVC Associations

Above figure is a specific browser based MVC example. Controller communicates with Models and Views directly. User gets browser results based on actions defined on Controllers. An important point here is separation of view and model. The communication between view and model is realised on controller.

Based on previous research results, basic functions on an e-learning system is summarised as Table 4-1.

Function	**User**
Login/Logout	Learner Entity, Coach, System Administrator

View Learning Contents	Learner Entity, Coach, System Administrator
Edit Learning Contents (Add/Edit/Del)	Coach
Navigation	Learner Entity, Coach, System Administrator
Attend Evaluation	Learner Entity
Evaluate Result	Coach
User Interface Layout Editing	System Administrator
Database management	System Administrator
Communication	Leaner Entity, Coach
Update Learner Record (personal information)	Learner Entity
Update Learner Record (add advice)	Coach

Table 4-1. Basic E-Learning Functions

The functions are user oriented. Users are including “learner”, “coach”, and “system administrator”. Because the system administrator’s functions are basically same with other general websites, the modelling research focuses on functions working for “learner” and “coach”. In Table 4-2, the functions are refined and classified based on MVC pattern.

	Learner	**Coach**
Views	Learner Login/Logout Pages	Coach Login/Logout Pages
	Leaner Information Pages	Coach Information Pages
	Learning Contents Pages	Learning Contents Editing Pages (View/Add/Edit/Del)
	Learner Navigation Bar	Coach Navigation Bar
	Evaluation Pages (e.g. Examination Pages, Result Pages, Homework pages, etc.)	Evaluation Pages (e.g. Homework Check Pages)
Controllers	Login/Logout Response	Login/Logout Response
	Learner Information Control	Coach Information Control
	Learning Contents Control	Learning Contents Editing (View/Add/Edit/Del) Control
	Navigation Generation	Navigation Generation
	Evaluation	Evaluation

Models	Databases Connection (Learning Resources DB, Learner Records DB, and Coach Records DB)
	Learning Resources Editing (Search, Add, Edit, Delete)
	Records Editing (Search, Add, Edit, Delete)

Table 4-2. Functions Refining and Categories Based on MVC Pattern

Views include and only include all kind of pages. Same function should not show in the exactly same format or content for different users, especially "learner" and "coach" in an e-learning system. Therefore, "Login/Logout", "Information", "Contents", "Navigation", and "Evaluation" views are divided into two groups to suit the difference.

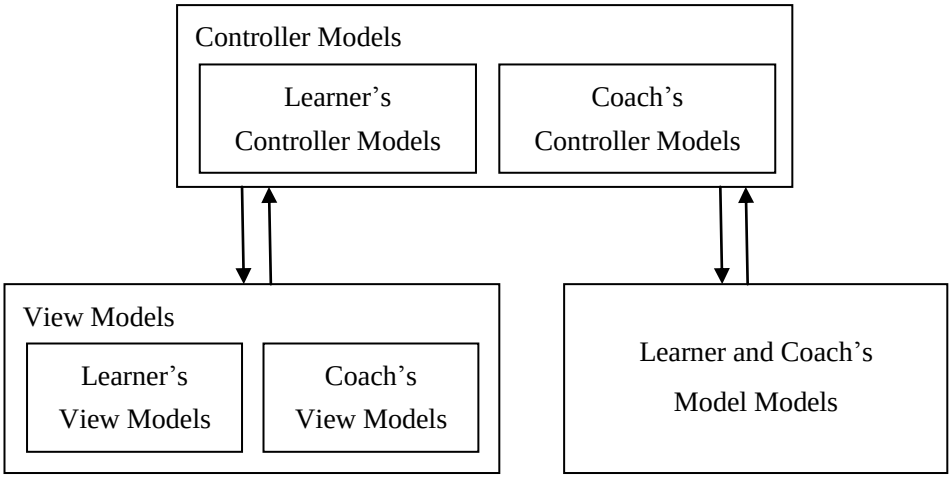


Figure 4-11. Advanced MVC-Based Modelling Structure

Controllers are the inside actions to connect models and views. There is at least one controller model corresponding to each view model. Consequently, "Login/Logout",

"Information", "Contents", "Navigation", and "Evaluation" models are divided into two groups as well.

Models are in charge of database operations. Hence, there are mainly "Databases connection", "Learning resources editing", and "Records editing" models. It is the basic database operation, so that there are no separate categories for different users.

Overall, the whole modelling structure is shown as Figure 4-11.

4.2.2 MDA-Based Modelling Structure

4.2.2.1 Modelling Structure

Based on the MVC structure, the MDA-based modelling is shown as following figure.

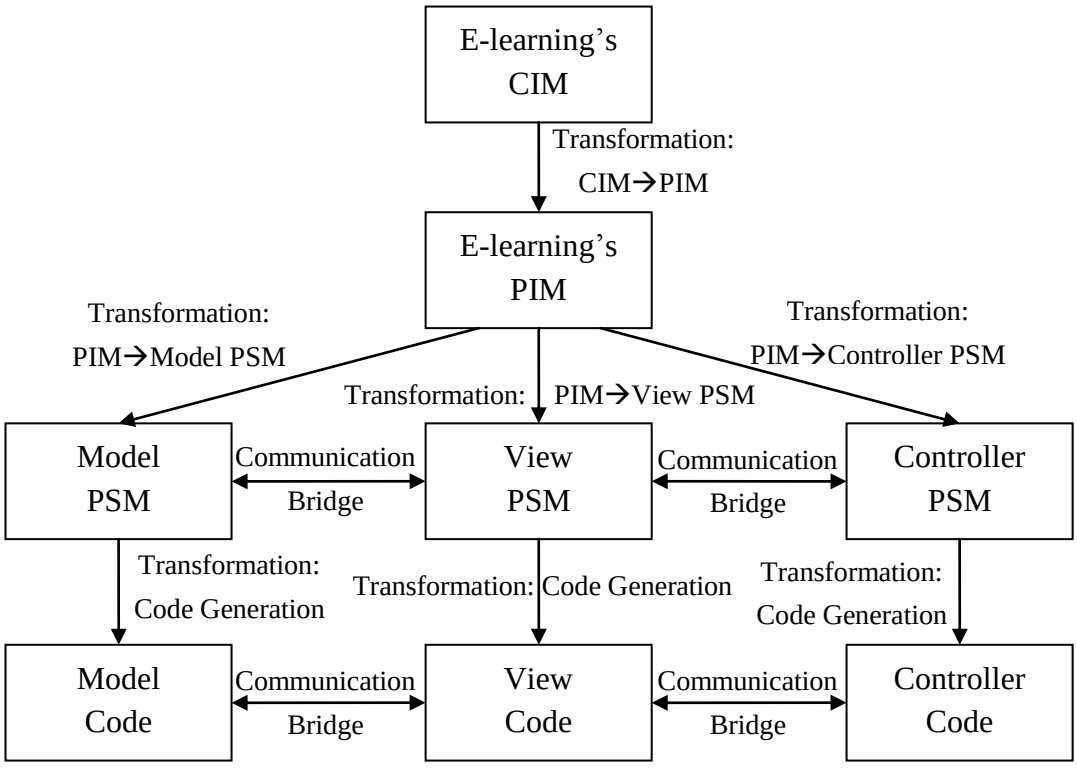


Figure 4-12. MDA-Based MVC E-Learning Structure

It merges MVC to the structure which is following MDA process.

4.2.2.2 Structure Layers

The structure is divided into four layers which are strictly following MDA principle.

- Layer 0: CIM. It is e-learning's CIM.
- Layer 1: PIM. E-learning's PIM is content of this level.
- Layer 2: PSM. The SPMs are divided into model PSM, view PSM, and controller PSM.
- Layer 3: Code. There are there group code models including model code, view code, and controller code.

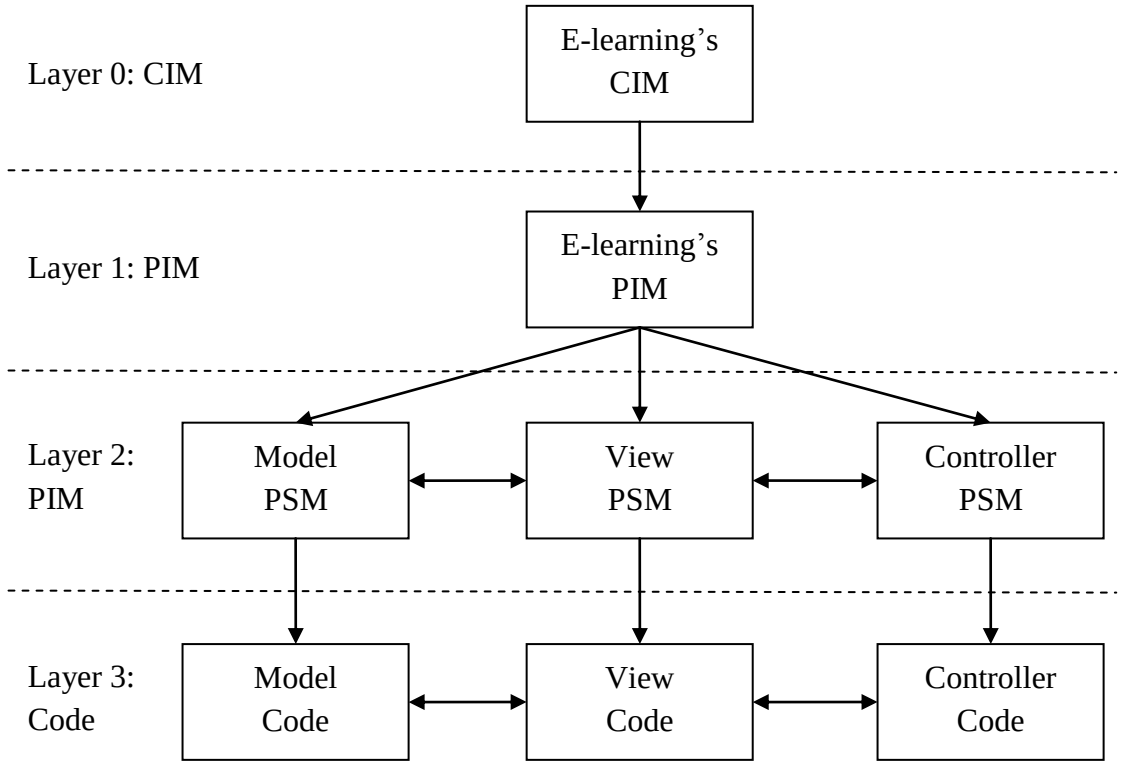


Figure 4-13. Modelling Structure Layers

4.2.2.3 Relationship and Communication on Models

As shown in figure 4-2, there are different relationships and communications between models. It can be categorised into two kinds: Transformation and Communication Bridge.

- Transformation CIM →PIM: it is CIM to PIM transformation working from layer 0 to layer 1.

- Transformation PIM → Model PSM: it is transformation from PIM specifically to model PSM. Therefore, this transformation will select Model elements from PIM to generate corresponding PSM.

- Transformation PIM → View PSM: it is transformation from PIM specifically to View PSM. The "specifically" means this transformation only mapping View elements from PIM to corresponding PSM.

- Transformation PIM → Controller PSM: it is transformation from PIM's controller elements to Controller PSM. The three PIM to PSM transformation are working between layer 1 and layer 2.

- Transformation – Code Generation: As the name shows, this transformation aims to generate corresponding codes for PSMs including Model PSM, View PSM, and Controller PSM. It is working on layer 2 to layer 3.

- Communication Bridge: it exists in layer 2 and layer 3. In layer 2, it shows the connections between PSMs. In layer 3, it presents communications between Model code, View code, and Controller code.

Overall, the transformations and communications are bonding models, codes, and layers to realise a united MDA-based MVC modelling structure.

4.3 E-Learning Domain Framework

Based on last sections in this chapter, e-learning domain architecture is proposed as figure 4-14 based on proposed MDA-based modelling structure.

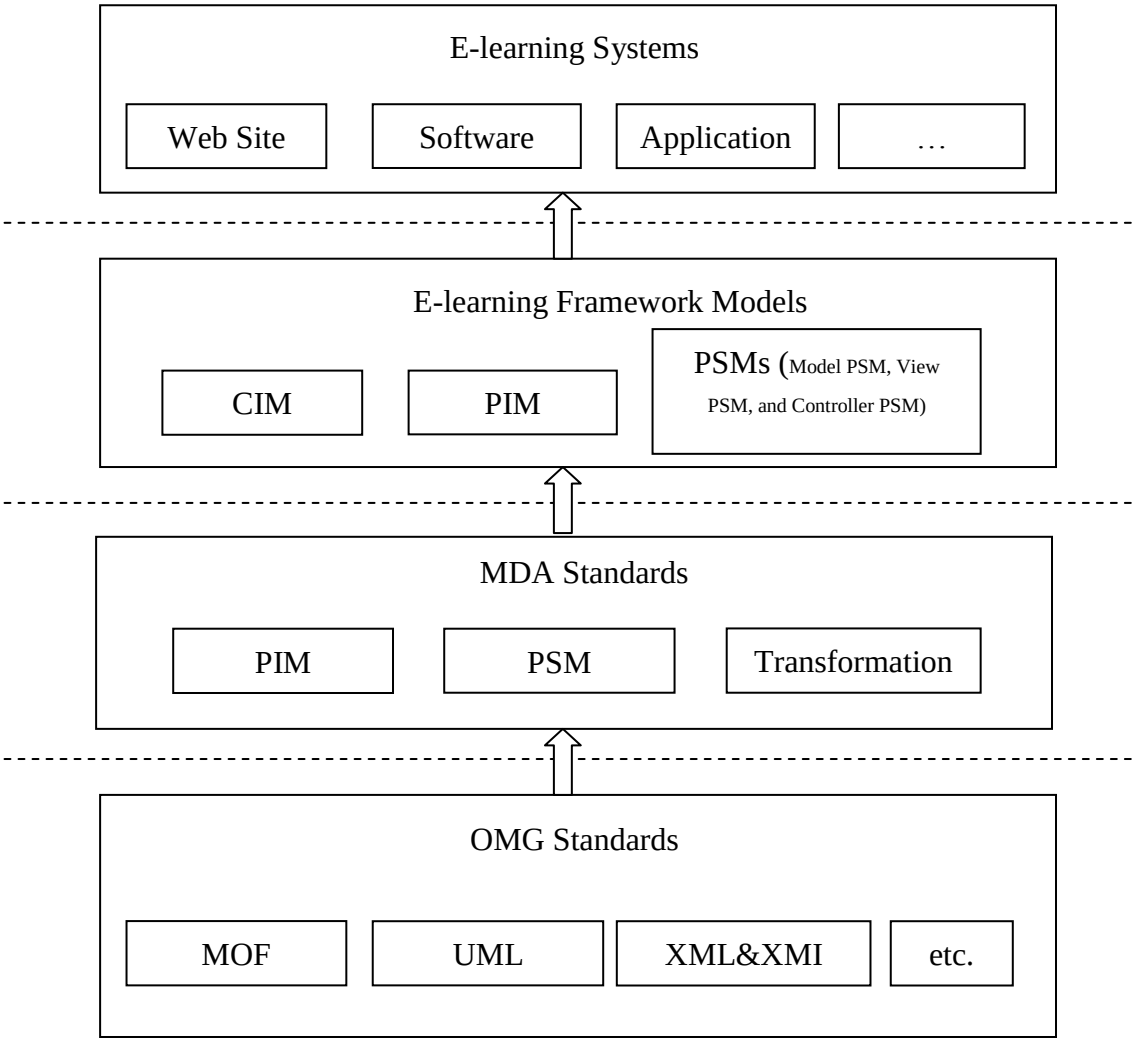


Figure 4-14. MDA-Based E-Learning Domain Architecture Overview

The above figure shows a logic view of MDA-based e-learning domain architecture. There are four layers in it specific as follow,

- OMG Standards: it is a layer including various standards proposed by OMG, i.e. MOF, UML, XML, XMI, etc. This is the foundation layer in this architecture.
- MDA Standards: this layer includes standards specifically proposed for MDA. They are PIM, PSM, and Transformation. It is a layer supported by OMG standards.
- E-learning Framework Models: it is a layer presenting framework models for e-learning. Because it is supported by MDA standards layer straightforward, there are mainly three group models: CIM, PIM, and PSM. Additionally, based on MVC structure, PSM is composed of "Model PSM", "View PSM", and "Controller PSM".
- E-learning Systems: this is system layer presenting developed achievements. The e-learning system can be web site, software, application, and so on, which is running on various platforms.

Combined above domain architecture with previous proposed domain modelling, MVC e-learning modelling, and MDA-based modelling, there is a framework for e-learning domain covering highly abstract structure and models in detail.

4.4 Summary

In this chapter, the e-learning modelling is proposed with specification:

- The LTSA standard is employed to support e-learning domain modelling as a kernel standard with its concept, architecture, and components. Firstly, a brief introduction of LTSA is presented. Secondly, the LTSA abstraction is explained with layers. Then, there are system components and features. Finally, it is stated that models of LTSA system will be structured explicitly into Platform Independent Models (PIMs) and Platform Specific Models (PSMs).

- Three models are designed for e-learning domain modelling supported by LTSA and pedagogical strategy. There are “step by step learning model”, “optional learning model”, and “interacting learning model”. These three models are presenting three individual learning approaches in order to meet the pedagogical requirement. Furthermore, there are specifications, advantages, and disadvantages stated for each model.

- MVC is applied to be the main design pattern for proposed e-learning modelling. Three modules and there associations are explained. Basic functions on general e-learning system is summarised, refined, and classified. Lastly, a whole modelling structure is designed for e-learning system based on MVC pattern.

- Based on MVC structure, a MDA-based modelling structure is proposed for e-learning domain with four layers. Meanwhile, each layer is explained with details. There are also specifications for relationships and communications between models. Moreover, there is an e-learning domain framework presented with layered structure and supporting relationships.

Chapter 5
A Proposed Approach to Model Driven Architecture based Evolution

Objectives

- To propose a Model-Driven Architecture based evolution method
- To present a Model-Driven Architecture based evolution process
- To explain development approach
- To specific vocabularies extraction method
- To present Platform-Independent Model generation

5.1 A Model-Driven Architecture based Evolution Method

A MDA-based System Development Lifecycle (MDA-SDLC) will be proposed in this section which has been published in a conference paper [102]. This lifecycle is still based on the traditional software development lifecycle, but combined with the MDA-based development approach. A software development methodology normally consists of two main parts: a Modelling Language (syntax and semantics) and a Process. Naturally, MDA-based methodologies use UML as their modelling language. The generic lifecycle is therefore mainly focused on the process part of the methodology.

The proposed lifecycle is not a concrete methodology, but a general process that defines the phases and activities expected to be present in an MDA-based methodology [9]. Therefore, it can be specialised to fit the exact project situation. MDA-SDLC consists of six phases (Figure 5-1): Project Initiation, CIM Development, PIM Development, PSM Development, Code Development, and Maintenance. As shown in Figure 5-1, returns to previous phases are usually necessary [9]. The PIM Development phase, the PSM Development phase, and the Code Development phase are typically performed in an iterative-incremental way. In section 5.1.1, the main phases will be described in detail.

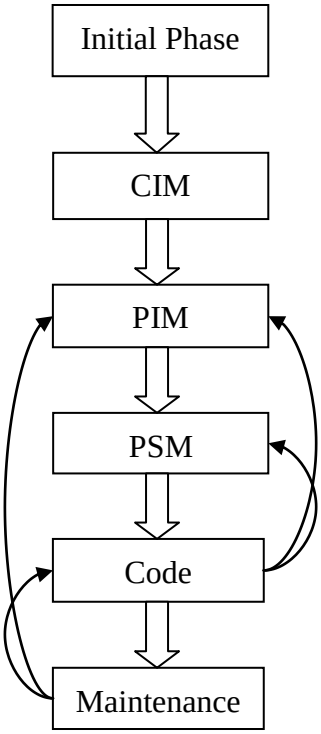


Figure 5-1. Proposed MDA-Based System Development Lifecycle

In our previous research [102], there is a basic MDA-based development approach been proposed by us. It is an initial research result, therefore, it is been improved after it's publish as following sections in this chapter. PIMs are normally modelled using the

UML to be UML diagrams. PSMs are generated by transformation tool that can translate the UML profile for target platforms.

5.1.1 Computation Independent Model

5.1.1.1 Requirements

The pedagogical strategy that is being modelled is a holistic one. It works as follows: there is a three-way approach that is to be presented interdependently. It consists of a "section" concerning music appreciation ("listening"), one focusing on the understanding of musical, theoretical and technological concepts ("understanding") and another involved with music making ("doing"). The heart is the understanding section as any tutor or learner-driven navigation starts here as all key terms and concepts are embedded in this section. Nonetheless, as this type of music is not as well known as, say, certain forms of commercial music, a didactic approach to repertoire development and learning what to listen for is essential. This takes place in the listening section and communicates directly with and is dependent on the related concepts in the understanding section. When it comes to how this music is made, certain specific means of sonic treatment will be introduced in the understanding section, for example, how a filter can alter the timbral quality of a given sound. To this end, the development team is working with the researchers who created JASS, a Java Audio Synthesis System for programmers at the University of British Columbia. JASS will allow for real-time sound manipulation tools to be placed online within the understanding section of the pedagogical environment in a manner that is platform independent. A software program called Sound Organiser is being developed currently and will be integrated into this system. This is the one element of the system that will exist as an independent program, but will also be adapted to work with specific concepts introduced in the understanding section. In this way users can focus

on a single concept instead of dealing with the full palette of means of sound organisation within this pedagogical environment, but will, at the same time, becoming increasingly able to use the Sound Organiser within and outside of the learning environment [102].

The holistic approach that integrates concept development, listening and creative aspects related to electroacoustic music is one of the unique aspects of this project. Three more interesting aspects of the system are worthy of introduction. First of all, users will be entitled to have diagnostic information presented either to teachers or to the learners themselves. This information helps users to find out where their strengths lie and in which areas there is room for improvement. It can also signal concepts not being learned and offer advice as to how to achieve more satisfying results. This means that in any interactive aspect of the site, user reactions will be monitored and information on the acquisition of concepts presented. Applications of techniques developed in artificial intelligence should prove invaluable in this aspect of the research project [102].

A second aspect will be offered which has to do with user-generated feedback and content provision. To cite one example; the current EARS research site is based on a tree structure to order the ca. 500 terms that it offers. The project presented here will continue to offer this traditional option, but will use state of the art tagging systems to offer another means of potential navigation through concept acquisition. This brings us the third important aspect [102].

The pedagogical environment will be able to be navigated in a variety of manners. These include:

- Previously organised paths designed by the development team will be on offer for individual users taking into account their previous knowledge. An avatar will act as teacher in such cases.

- Similarly, teachers may prefer to use one of the curriculum navigation systems on offer.

- The teachers may prefer to organise their own curriculum navigation system based on classroom needs.

- Individuals may be interested in specific subjects or aspects of electroacoustic music and therefore may want to create their own navigation.

All of the above will be implemented within the pedagogical architecture of this environment.

5.1.1.2 Models

In this MDA-based evolution method, there are models involved in CIM phase to support modelling the specific requirement in section 5.1.1.1. The three learning models proposed in last section includes “step by step learning model”, “optional learning model”, and “interaction learning model”. These models are applied as reference models for e-learning system based on educational pedagogical knowledge [66], Model-Driven Architecture (MDA) [62], and the Standard for Learning Technology — Learning Technology Systems Architecture (LTSA) [35].

Combined requirement with reference models, the CIM is created as the first layer model in MDA structure. Following the MDA-based System Development Lifecycle (MDA-SDLC) proposed in the beginning of section 5.1, CIM is going to be transformed into PIM which is specific described in next section.

5.1.2 Platform-Independent Model

In MDA structure, PIM is the model in layer 1 which is next to layer 0 – CIM and layer 2 - PSMs. There is a PIM development method proposed in one of our previous published conference paper [102]. This development phase aims to create a complete and exact model of the system which is presented as PIM.

PIM includes both structure and behaviour. It is derived from the Requirements Model which generated on the CIM phase. The core concept is that the model must be platform independent. For example, during the creation of this model nothing about the implementation platform is taken into account [9, 62, 63, 102]. Normally the results should be several UML diagrams, including activity diagrams, sequence diagrams, state machines, and class diagrams. Based on the research of M. Asadi, M. Ravakhah and R. Ramsin [9], there is an Activities Diagrame designed by us [102] for PIM development phases which is showing as figure 5-2.

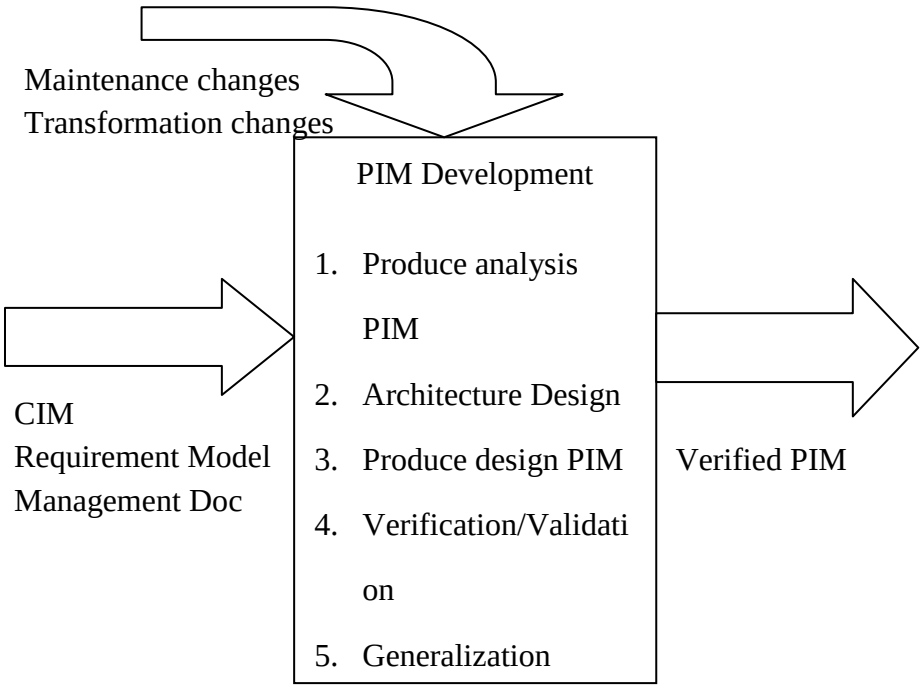


Figure 5-2. Main Activities in PIM Development [102]

- Produce analysis PIM [102]: it is the beginning work in this develop phase. In this activity, a model, which is platform independent, is defined through analysing the requirements model and other documents. System functionalities are described in the analysis PIM while maintaining traceability to the requirements model [9]. Developers use appropriate model elements to develop some parts of this PIM. Although this model is not the final PIM, it is quite important because it is the foundation for further producing to the final version. Conventional Object-Oriented analysis techniques can be used for this activity, which is typically executed in an iterative and incremental fashion [9].
- Architectural design [102]: in this activity, the system architecture is designed. If it is necessary, a review of the requirement and general plan is required. A main system framework is the result.
- Produce design PIM [9, 102]: in this activity, the model, which is generated in analysis PIM, is refined. The main function in this activity is models the detailed structure and behaviour of the system. Conventional OO design techniques can be used in this activity. The design PIM is derived from the analysis PIM and it is in an iterative-incremental style. Constraints, preconditions, post conditions, and invariants are defined using UML and OCL mechanisms [9].Meanwhile, reusable domain-dependent design model elements can also be retrieved from model repositories for composing the design PIM.
- Verification/Validation [102]: in this phase (PIM Development)，it is necessary to check whether the products of the modelling activities are free from defects. It is also need to be sure that those products are coincident with the requirements, which is generated in the requirements modelling activity. The mainly aim of this activity is to correct errors in design PIM. It is vital issue in PIM to PSMs transformation.
- Generalisation [102]: after creating models in the previous activities, it is naturally to execute the generalisation. In this activity, the models is going to be produced more

reusable. Reusable domain specific model elements are uploaded and categorised into repositories for reuse in future projects [9, 102]. The main productions of this activity are the verified design PIM and Management Documents.

It can be seen from the proposed MDA-based System Development Lifecycle (MDA-SDLC) that a completed PIM is going to be transformed into PSMs due to specific platform application. Then, corresponding codes are generated from PSMs. Details about method of these two development phases are presented in next section.

5.1.3 Platform-Specific Model and Code Development Method

The MDA separates certain key models of systems, and brings a consistent structure to these models. Models of different systems are structured explicitly into PIMs and PSMs. How the functionality specified in a PIM is realised is specified in a platform-specific way in the PSM, which is derived from the PIM via some transformation tools [23].

Once the last phase PIM Development is complete, it is stored in the suitable ways, i.e. UML diagrams. Then they will be the input resources to the mapping step, which will produce PSMs. One PIM can generate many PSMs, which are depended on different underlying platforms. In the beginning, UML only express PSM. However, since UML is independent of middleware technologies, UML is also popular to expresses PIM. Anyway, UML can be applied to store both of PIM and PSM. After transformation from PIM to PSM, code will be generated by some MDA tools. Based on the previous research [102], the activities of this phase are as Figure 5-3.

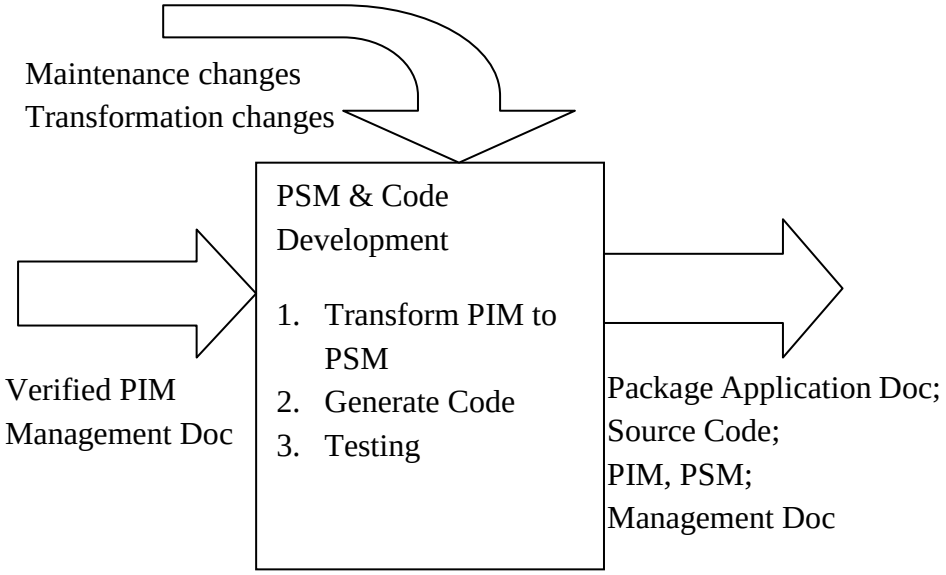


Figure 5-3. Main Activities in PSM and Code Development [102]

Transform PIM to PSM [102]: MDA tools are the main way to generate PSMs from PIM. The new and core technology provided by MDA is the automatically transformation. Conventional guidelines are used to guide the developer in performing the transformation using the selected tool [9]. These guidelines are provided by the tool or by the methodology itself.

Generate code [102]: Because good PSMs are already very close to the code, the generation is mainly worked by some MDA tools. Ideally，in this activity, the execution code is completely and automatically generated from the PSM using MDA tools. However，current MDA tools cannot work so far. The developers then have to manually complete the generated code. Plus, as mentioned in last activity, conventional guidelines can be used to guide the developer in performing the transformation using the selected tool [9, 102]. Currently, this activity aims to build organisation of the code. Meanwhile，it can executes the code unit tests, and do some integration work. All components and subsystems should be combined together shown as concrete code.

Testing [102]: this activity includes standard testing tasks such as: plan tests, prepare test model, prepare test cases and test scripts, execute tests, correct defects and document test results [9, 102]. As last activity, automatic testing is useful but still too ideally to complete all test tasks. Actually, manual testing is usually necessary. Developers should combine those two ways to finish this activity. There are several main production in this activity, which are packaged application, source code, PIM，PSM, and management documents.

5.2 A Model-Driven Architecture based Evolution Process

5.2.1 Approach Overview

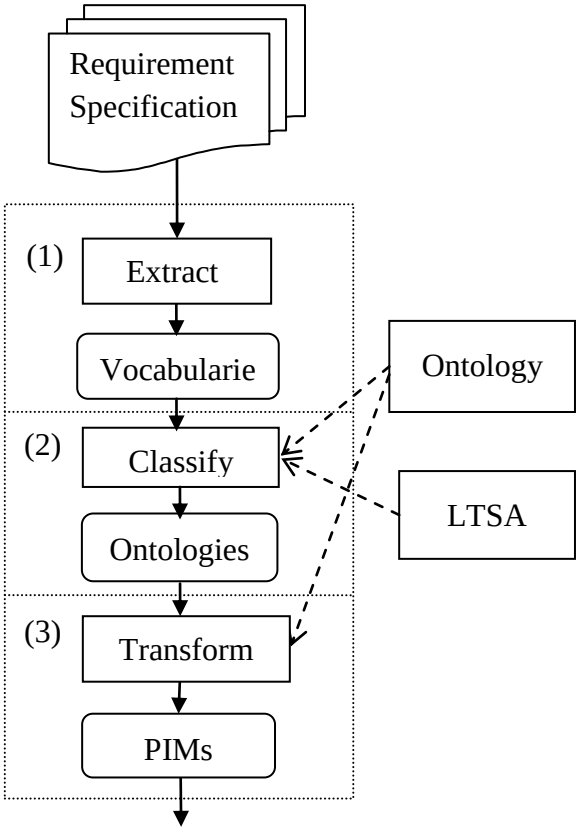


Figure 5-4. Ontology-Based PIM Modelling Approach

Based on proposed MDA-based evolution method in section 5.1, there is an ontology-based approach designed and presented in this chapter which is based on our published research result [101].

This ontology-based model driven approach was proposed for music learning system. However, this approach is not only limited on music-learning system but also suitable for general e-learning systems. Therefore, it is imported in this research to support the proposed MDA-based evaluation method. The main approach is shown as Figure 5-4.

It can be seen from Figure 5-4, there are mainly three steps in this approach:

- Extracting vocabularies: according to Natural Language Processing (NLP) technology, requirements are extracted into vocabularies.
- Classifying ontologies: LTSA is the basic structure for classify the vocabularies that come from previous step. First, a RO is involved in this phase, which designed based on LTSA. Then classify vocabularies into RO to be an AO. Next task is to add extra vocabularies into AO. Finely, if there are redundancies in AO, they are reduced in this step.
- Transforming into PIMs: Ontologies are transformed into Platform Independent Models following a set of transformation rules that we proposed. Considered the PIMs are showed as a set of UML diagrams generally, following the five rules, classes are generated with name, mandatory attributes, operations, interfaces, and relationships.

5.2.2 Vocabularies Extraction

Vocabularies extraction is always happened as a general activity in initial development such as requirement writing. Normally, developers extract them on mind with potential self-rules. In this approach, Natural Language Processing (NLP) theory is used as basic

technology for extraction. This activity aims to get simple vocabularies including Noun, Verb, and relevant explanation. Therefore, Natural language understanding (NLU) system is involved. However, we will not discuss specific methodology or tools about NLU since it is another research issue. The only rule here is to reduce redundancy after extraction. The result structure of vocabularies is organised as below,

Noun: Explanation	Verb: Explanation
…	…

Table 5-1. Format of a Set of Vocabularies

5.2.3 Ontologies Classification

There are a number of terms to be used to classify ontologies. There are Lightweight ontologies that only consist “if” concepts and their relationships, but without many axioms, additional conditions and restrictions; Application Ontologies (AO) contain the definitions specific to a particular application [31], while Reference Ontologies (RO) focus on clarifying the intended meaning of terms used in specific domains. RO is designed for general e-learning system based on proposed three learning models – “step by step learning model”, “optional learning model”, and “interaction learning model”.

The proposed three learning models has specified high level architectures for information technology-supported learning, education and training System. Therefore, components in those models are important supports in this phase. Additionally, because the “interaction learning model” is covering components in the other two learning models, it is selected to be the model to support RO’s design.

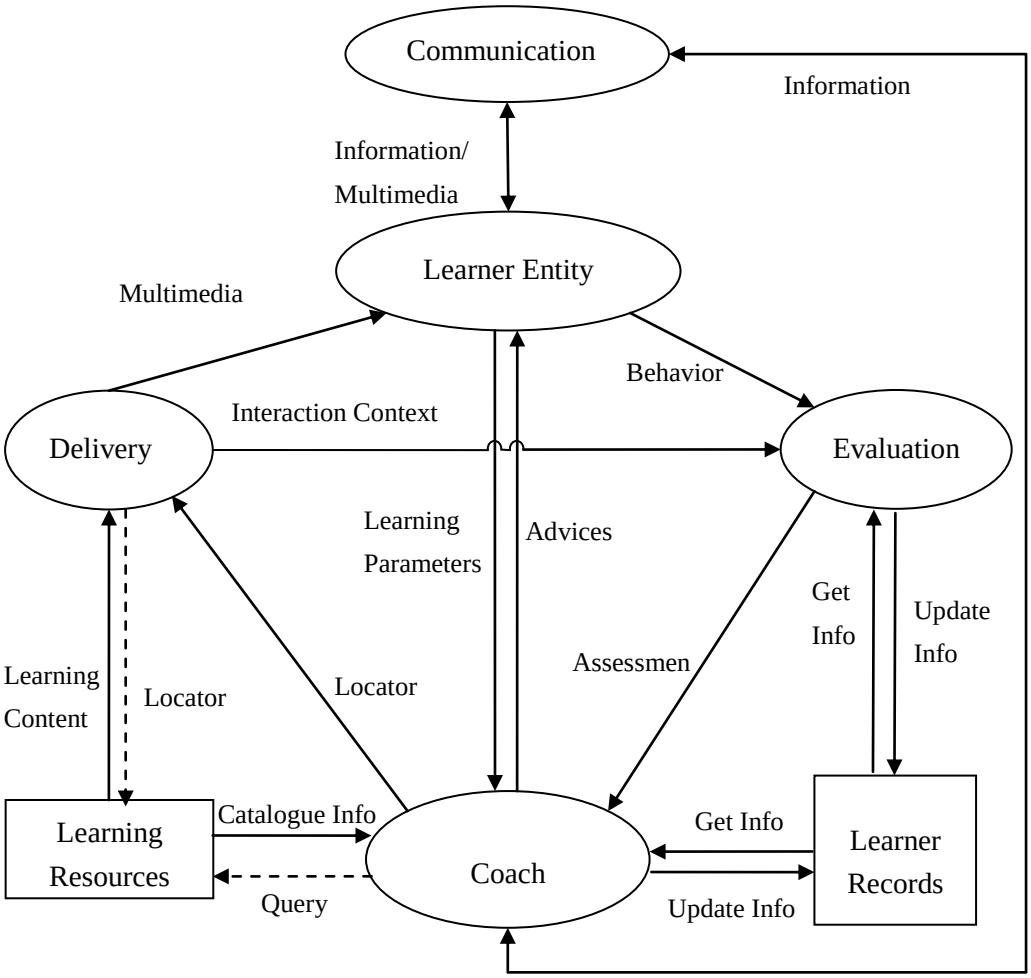


Figure 5-5. Interaction Learning Model Components' Structure

Based on components' structure in figure 5-5, a RO is designed and depicted as figure 5-6 with specification of concepts and relationships.

The notion of RO is defined as a 3-tuple RO=(C, A, Sc), where: C=Concept, A=Attribute, Sc=SubConcept. Attribute owns a specific definition A= (Do, Ra), where: Do=Domain, Ra=Range.

There are three steps designed to generate AO from RO:

- To generate AO by mapping vocabularies into RO;

- To add extra vocabularies into ontologies as an AO;

- To reduce redundancies for AO.

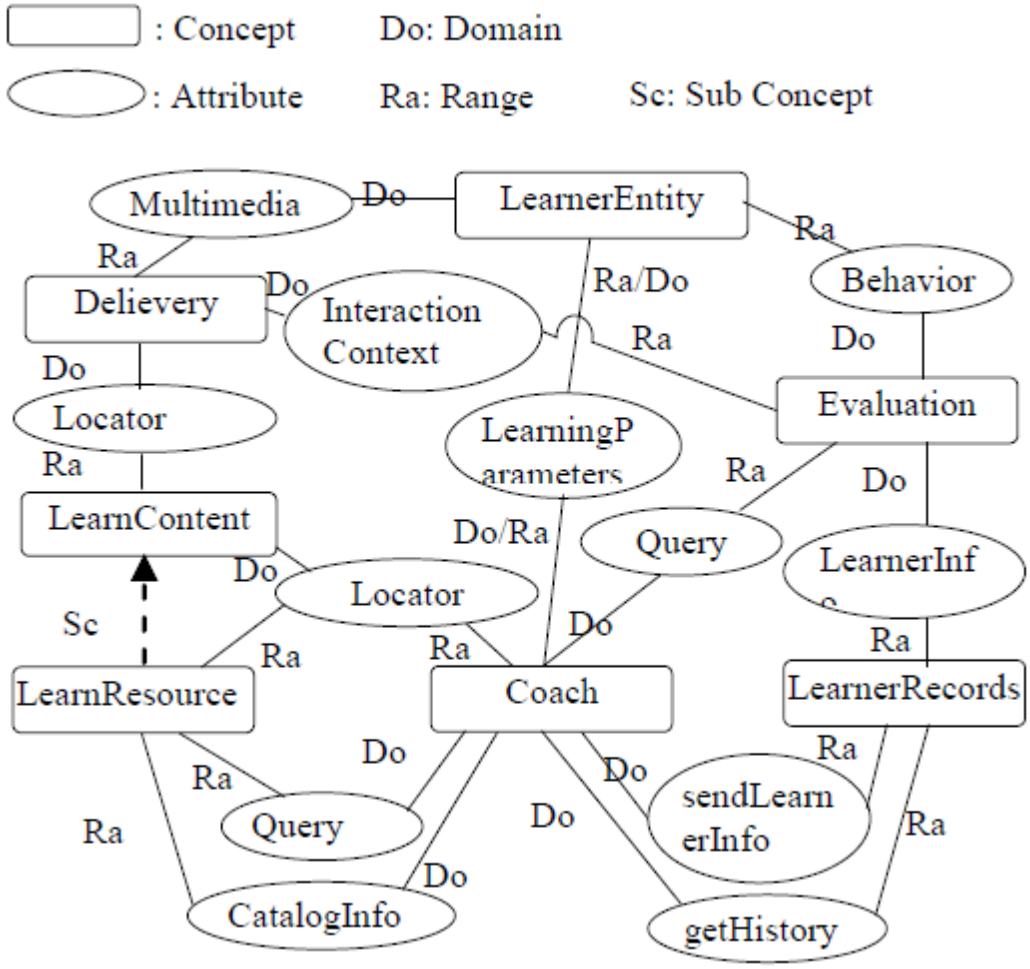


Figure 5-6. Reference Ontologies Structure

The notation of AO is defined as a 4-tuple AO=(C, A, O, Sc) that A=Attribute, O=Object= (Name, Domain, Range, Value), Sc=SubConcept. There are more details for e-learning system on Table 5-2. Besides, the result is a “good” AO following Gruber’s criteria [30] which describes what “good” ontology should meet: terms clarity, axioms coherence, extensibility, and suitability.

Concepts [C]	**Attributes(Range) [A(Ra)]**	**SubConcept [Sc]**
LearnerEntity	leID (String); lePassword(String); Login(Boolean); Logout(Boolean); Multimedia(Delievery); LearningParameters(Coach).	
LearnerRecords	LroID(String); learnerInfo(Evaluation).	
LearnResource	lrID(String); lrContents(X[1]);	LearnContent

LearnContent	lcID(String);
	lcBegin(X);
	lcEnd(X).	

Delivery(GUI)	deID(String); locator(LearnContent);	

[1] 'X' indicates range is uncertain but depends on specific system.

Evaluation	LearnerInfo(LearnerRecord); Evaluate(LearnerEntity).	

Coach	coID(String);
	coPassword(String);
	Login(Boolean);
	Logout(Boolean);
	sendLearnerInfo(LearnerRecord);
	getHistory(LearnerRecord);
	LearningParameters(LearnerEntity);
	Locator (Coach, LearnResource);
	Query(LearnResource);
	CatalogInfo(LearnResource).	

Table 5-2. Reference Ontologies List for General E-Learning System

The followings are the specific classification steps:

(1) To map vocabularies into AO.

In the vocabularies, noun words map to “Concept”, “SubConcept”, or “Object” under its explanation. Verb maps to “Attribute” only. This step results in Table 5-3 as below,

Concepts [C]	**Attributes(Range) [A(Ra)]**	**Object(O)**	**SubConcept [Sc]**
N1 …	V1; V2; V3 …	N11 …	 …

Table 5-3. Application Ontologies Result of Step (1)

(2) To add extra vocabularies into ontologies.

If there are some vocabularies left, a step to add them properly in AO is necessary. The Table 5-4 shows the Result of step2, where Ex is Extra Vocabulary.

Concepts [C]	**Attributes(Range)[A(Ra)]**	**Object(O)**	**SubConcept [Sc]**
N1	V1; V2; V3	N11	
…	…	…	…
Ex_N1	Ex_V	Ex_N2	Ex_N3
…	…	…	…

Table 5-4. Application Ontologies Result of Step (2)

(3) To reduce redundancy for AO.

There are many reasons to introduce redundancy, such as synonyms, verb and noun with same meaning, vocabularies under inclusive relationship, etc. Table 5-5 shows the result of this step with strikethrough on redundancy.

Concepts [C]	**Attributes(Range)[A(Ra)]**	**Object(O)**	**SubConcept [Sc]**
N1	V1; V2; ~~V3~~	N11	
…	…	…	…
~~Ex_N1~~	Ex_V	Ex_N2	Ex_N3
…	…	…	…

Table 5-5. Application Ontologies Result of Step (3)

5.2.4 PIM Generation

To transform AO to PIM, a set of transformation rules are proposed. The rules working structure is shown as figure 5-7 which presents how they are mapping AO to PIMs.

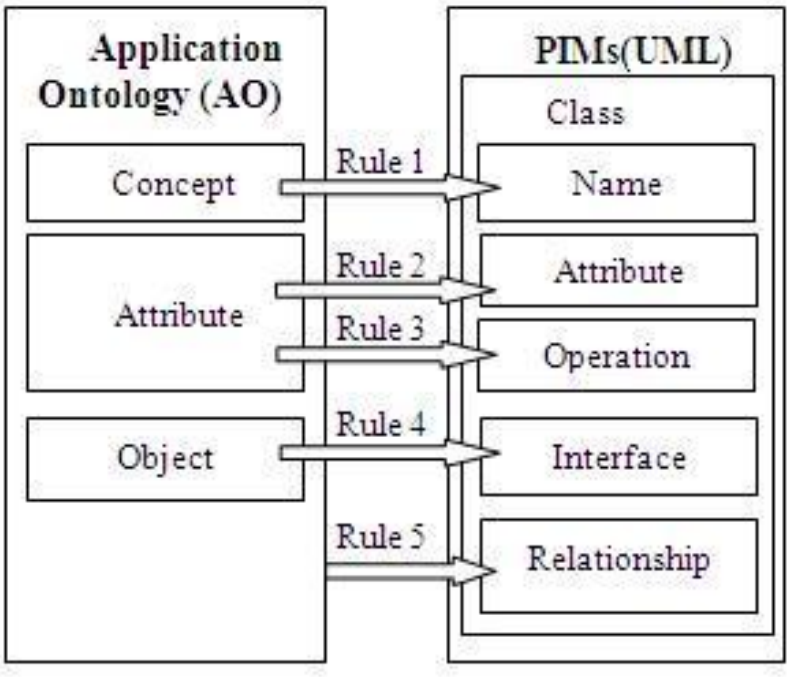


Figure 5-7. Transformation Rules -- AO to PIMs

Notations of the transformation rules are defined as follows.

(1) AO=(C, A, O, Sc); C=Concept, A=Attribute= (Name, Domain, Range, Value), O=Object= (Name, Domain, Range, Value), Sc=SubConcept.

(2) M=PIM= (Cl, In, Re (Clx)). Cl=Class= (Na, At, Op), where, Na=Name, At=Attribute, Op=Operation; In=Interface= (Na, At, Op) where Na=Name, At=Attribute, Op= Operation; Re (Clx) =Relationship with Classes Clx, including "Generalisation", "Association", and "Composition".

Rule 1: Mapping Class

In AO, each Concept maps to a Class in PIM. Because every Concept is a noun, Class's name simply is valued by Concept.

```
 M.Cl{
Check AO.C;
Force M.CL.Na=AO.C;
}
```

Rule 2: Mapping mandatory attributes

In AO, if Attribute's Range is not any Concept or Object, it equals to Class's mandatory attribute.

```
 M.Cl.At{
Check AO.A;
If (Not AO.A.Ra==AO.C and AO.O){
 M.Cl.At=AO.C.O.A;
}}
```

Rule 3: Mapping Operations

In AO, if Attribute's Value is verb, it maps to Option of Class.

```
M.Cl.Op{
Check AO.A;
if(AO.A.Value==Verb){
 M.Cl.Op=AO.A;
}}
```

Rule 4: Mapping Interfaces

In AO, if Attribute's Range is a Concept, AO's Object is valued as an Interface in Class.

```
M.In{
Check AO;
if (AO.A.Range==AO.C){
 M.In=AO.O;
}}
```

Rule 5: Mapping Relationships

In PIM, three relationships are necessary: Generalisation, Association, and Composition. They are mapped separately.

(1) Generalisation

Generalisation is "a-kind-of" relationship. Checking AO, if SubConcept is not empty, there must have a generalisation relationship. One Class is valued by SubConcept which is generated from the other Class that valued by Concept.

```
M.Re{
Check AO.Sc;
if (Not AO.Sc==None){
  M.Re=Generalisation (M.Cl1==AO.C, M.Cl2==AO.Sc);//Cl2 generated from Cl1.
}}
```

(2) Association

Association is a kind of semantic relationship between classes. In AO, if value of one Attribute is another Concept, the mapped classed are associated.

```
M.Re{
Check AO.A;
if (have (AO1.A.Value==AO2.C)){
  M.Re=Association (M.Cl1==AO1.C, M.Cl2==AO2.C);//Cl1 associate withCl2.
}}
```

(3) Composition

Composition is a particular association relationship showing components. If the value of an AO's attribute is a sum of many other AO's Concept, this AO valued Class is composed by other Classes that mapped from those other AO's Concepts.

```
M.Re{Check AO.A;
if (have (AO1.A.Value==∑_{i=2}^{n} AOi . C)){
```

M.Re=Composition(M.Cl1==AO1.C, M.Cl2==AO2.C,…, M.Cln=AOn.C);
}}

Following above rules, AO can be transformed into PIM as UML diagrams. Most Classes are mapped from AO's Concepts under Rule 1. Mandatory attributes and operations come from AO's Attribution by Rule 2 and 3. AO's Object been leaded to Interface following rule 4. Also, there are main relationships generated from AO depends on rule 5. Particularly, in rule 5, Generalisation, Association, and Composition are the three necessary relationships we considered. To sum up, PIM is transformed from AO under proposed rules.

5.3 Summary

In this chapter, there is a proposed approach to MDA-based evolution includes evolution method and process:

- A MDA-based evolution method is proposed. A MDA-based System Development Lifecycle (MDA-SDLC) is presented to support the proposed method with six phases: Project Initiation, CIM Development, PIM Development, PSM Development, Code Development, and Maintenance.

- There are explanations about CIM, PSM, PSM, code development methods. Firstly, the pedagogical requirement is presented and analysed. Previously proposed three models in last chapter are employed into CIM as reference models for e-learning system. Then, main activities in PIM development are explained containing "produce analysis PIM", "architectural design", "produce design PIM", "verification/validation", and "generalisation". Finally, PSM and code development method is presented with transformation technology.

- Based on proposed MDA-based evolution method, there is an ontology-based approach designed and presented. It is basically an ontology-based model driven approach proposed for e-learning system. There are mainly three steps in this approach including “Extracting vocabularies”, “Classifying ontologies”, and “Transforming into PIMs”.

- A RO is designed and depicted with specification of concepts and relationships based on components’ structure of “Interaction Learning Model”. Classification steps are specified including “to map vocabularies into AO”, “to add extra vocabularies into ontologies”, and “to reduce redundancy for AO”.

- There is a set of transformation rules proposed and specified to transform AO to PIM. Following proposed rules, AO can be transformed into PIM as UML diagrams.

Chapter 6
PIM to PSM Transformation

Objectives

- To propose transformation architecture
- To design source model and metamodel for source model
- To design target model and metamodel for target model
- To define transformation rules for transformation model and metamodel for transformation model
- To implement proposed mapping rules

6.1 Scope

This chapter outlines a PIM to PSMs transformation. It defines mapping methodology and rules for elements of a MOF 2.0 compliant PIM to the MOF 2.0 compliant PSM. The mapping rules basically depend on the state of the MOF 2.0 Core documents [75] and the adopted UML Infrastructure [81]. The transformation methodology is proposed in section 7.2. The set of mapping rules is outlined defined in detail in section 7.5.

The MOF 2.0 compliant models are chosen as PIM and PSMs aiming at the generation of highly preferment, highly scalable, and reliability, which are automatically deployable. It is believed that this specification strongly supports e-learning domain modelling using metadata technology and MVC architecture. Furthermore, it is a

contribution on the current technical disadvantages on PIM to PSMs transformation, which is the very heart of the MDA.

For the definition of the mapping rules themselves, a combined model is derived from the MOF 2.0 model and the MVC-based e-learning domain-specific model. Based on this combined model, the mapping rules are defined using OCL. The methodology is covered by section 7.2, including assumptions of the MOF model that served as the basis for this specification.

6.2 Transformation Architecture

Based on MDA and proposed evolution method, the transformation structure is designed as following figure with models and processes.

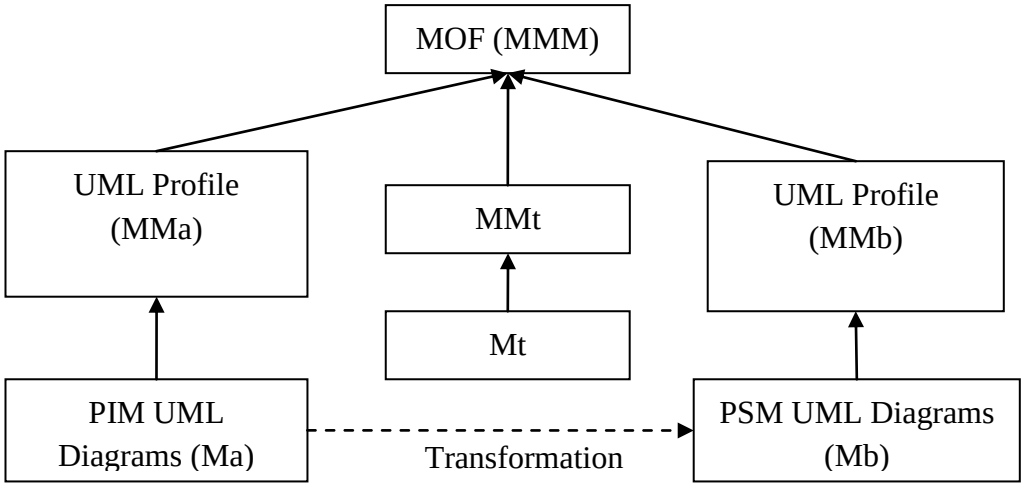


Figure 6-1. Transformation Architecture

The specification of this transformation architecture is as following,

- Ma: Source Model. It is the PIM for designers, which is described as a UML Object Diagram.

- MMa: MetaModel of Ma. It is proposed as a framework designed based on our research. It is shown as UML Class Diagram. Of course it is Platform independent.
- MMM: It is the meta-metamodel that is the kernel of MMa, MMb, and MMt.
- Mb: Target Model. It is the PSM for designers, which is described as a PHP5-aimed UML Object Diagram.
- MMb: MetaModel of Mb. It is proposed as a framework designed based on our research. It is shown as UML Class Diagram. Specifically, it is PHP5 dependent.
- Mt: Transformation Model. It is designed based on the proposed Transformation Rules.
- MMt: It is the MetaModel of Mt.

The proposed transformation architecture can be divided into three layers as figure 6-2 shows,

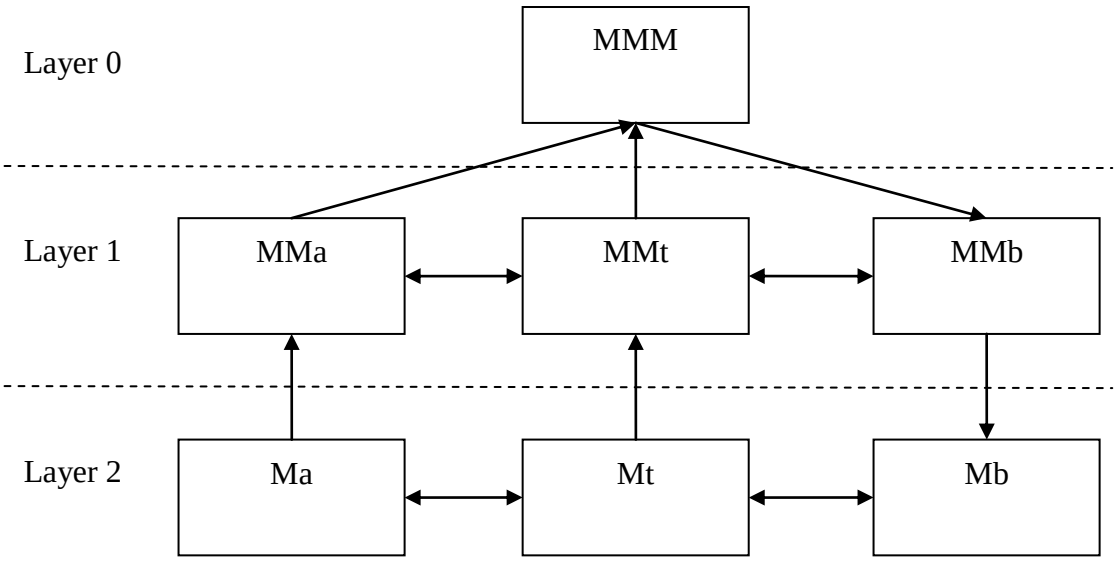


Figure 6-2. Layers of Transformation Architecture

Models and meta-models in different levels are catalogued into three layers,

- Layer 0: it is the kernel layer which supporting the other two layers. MMM is the content in this layer.
- Layer 1: it is a layer including meta-models for source model, target model, and transformation model. Specifically, they are MMa, MMb, and MMt.
- Layer 2: it can be seen as a user layer where shows source model, target model, and transformation model. Models in this layer are capable to edit if it is necessary.

6.3 Source Model and Metamodel of Source Model

In this section, the Platform Independent Model (PIM) is the source model (Ma) representing a platform-independent definition of system. The PIM is defined in the accompanying normative UML model. The metamodel (MMa) is also provided on a UML profile basis. Elements of this model are presented in this clause to clarify and provide guidance on this model.

6.3.1 Fundamental Model Elements for Metamodel of Source Model

Based on UML Profile extension method, there are fundamental model elements defined for MMa, including *"System", "User", "Content", "Navigation", "Resource"*, and *"Record"*.

- **System**: It is a class describes the system communicating users and contents.

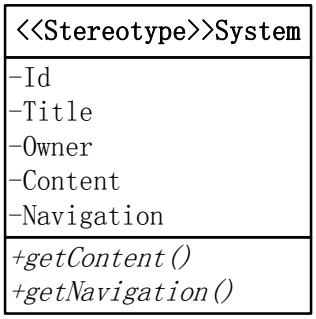


Figure 6-3. Model for System

- **User**: It is a class indicates various users. For example, it can be learners, coaches, and so on in an e-learning system.

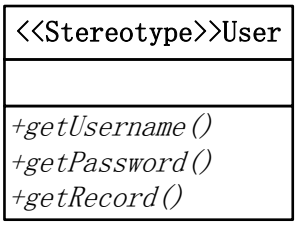


Figure 6-4. Model for User

- **Content**: It is a class to describe contents combined by various formatted resources in the system, including text, pictures, sound files, etc formats.

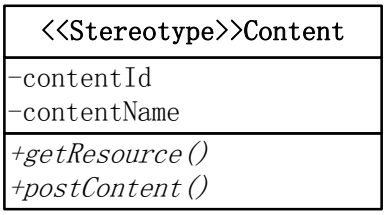


Figure 6-5. Model for Content

- Navigation: It is a class presenting navigation function aiming to support a path for user.

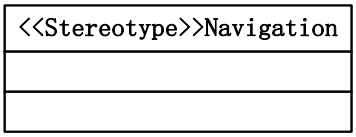


Figure 6-6. Model for Navigation

- **Resource**: It is a database stocking resources for system. The specific resources are text, pictures, sound files, etc formats. Different resources gathered together become content.

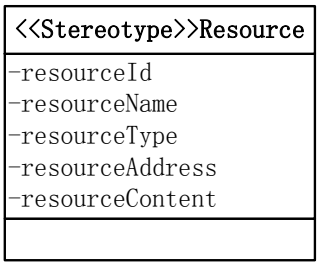


Figure 6-7. Model for Resource Database

- **Record**: It is a database stocking and providing user records for system.

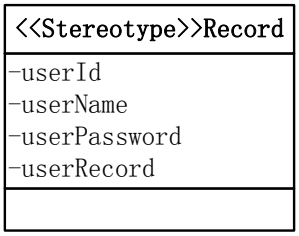


Figure 6-8. Model for Record Database

The relationship between fundamental elements is defined as figure 6- 9. A “system” depends on “Content” and “User” which shows as two “Dependency” associations. There is a “self-association” in “Content” element indicates it is an element can contains self-format elements. Two “Aggregation” relations present that “Resource” is part of “Content”, and “Record” is a part of “User”.

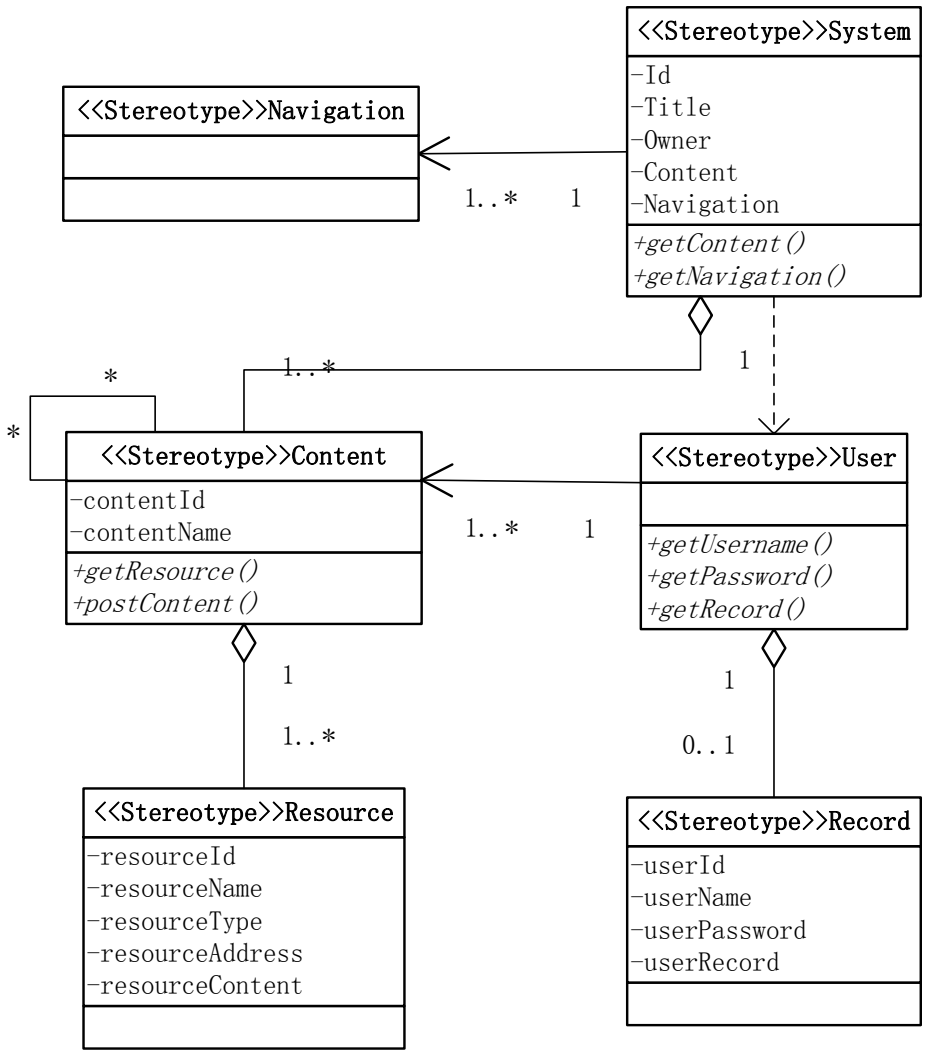
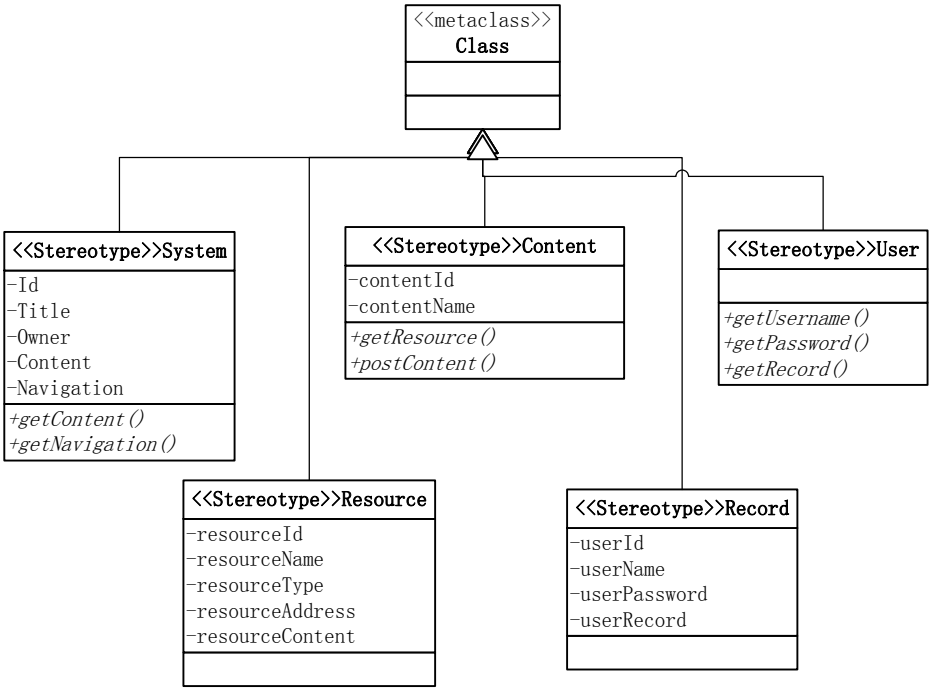
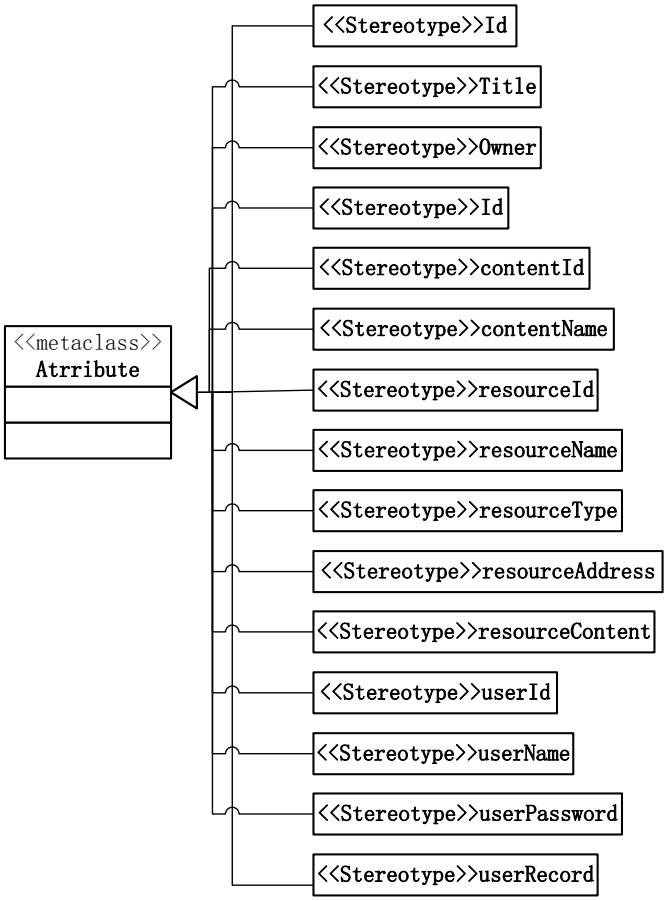


Figure 6-9. Stereotype Classes Relationship on Metamodel for Source Model

UML Profile is an extension based on UML aiming to present different models for specific domain. Therefore, the stereotype elements are related to UML original elements.

The details are shows as below figure including “metaclass:class”-based stereotype, “metaclass:attribute”-based stereotype, “metaclass:operation-based” stereotype, and “metaclass:association”-based stereotype.


<<metaclass>>
Class
<<Stereotype>>System
-Id
-Title
-Owner
-Content
-Navigation
+getContent()
+getNavigation()
<<Stereotype>>Content
-contentId
-contentName
+getResource()
+postContent()
<<Stereotype>>User
+getUsername()
+getPassword()
+getRecord()
<<Stereotype>>Resource
-resourceId
-resourceName
-resourceType
-resourceAddress
-resourceContent
<<Stereotype>>Record
-userId
-userName
-userPassword
-userRecord


<<metaclass>>
Atrribute
<<Stereotype>>Id
<<Stereotype>>Title
<<Stereotype>>Owner
<<Stereotype>>Id
<<Stereotype>>contentId
<<Stereotype>>contentName
<<Stereotype>>resourceId
<<Stereotype>>resourceName
<<Stereotype>>resourceType
<<Stereotype>>resourceAddress
<<Stereotype>>resourceContent
<<Stereotype>>userId
<<Stereotype>>userName
<<Stereotype>>userPassword
<<Stereotype>>userRecord

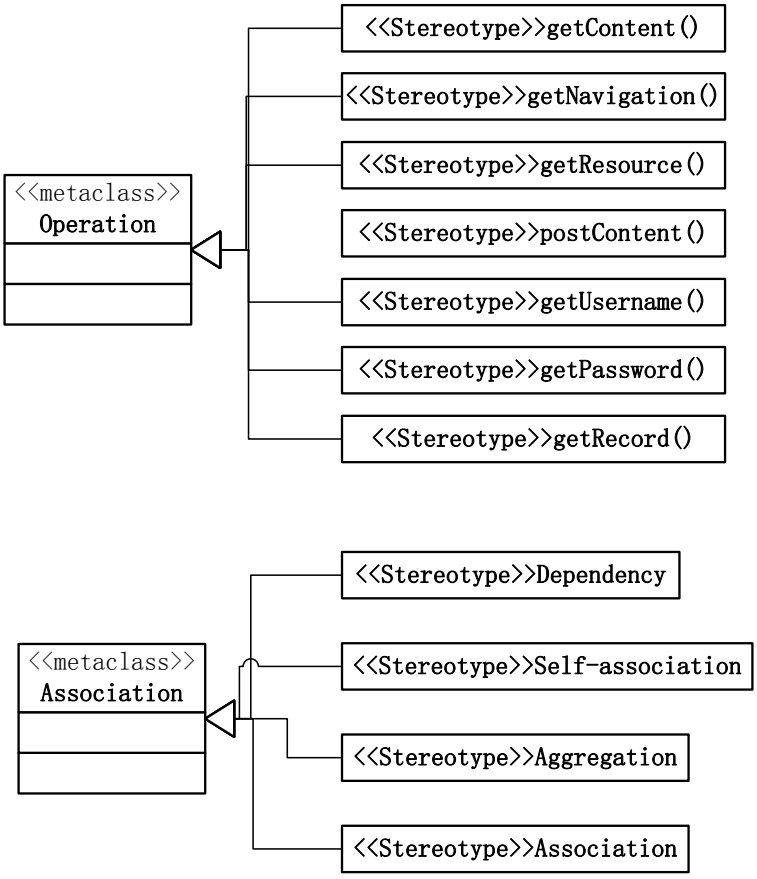


Figure 6-10. Level Structure of Metamodel for Source Model Stereotype Classes

6.3.2 Fundamental Model Elements for Source Model

In this section, there are definitions and specifications for Ma's fundamental model elements, including "E-learning System (eLearningSys)", "Learner", "Coach", "Administrator (Admin)", "Learner Record (LearnerRec)", "Coach Record (CoachRec)", "Adminidtrator Record (AdminRec)", "Content", "Resource", and "Navigation". Because elements are based on MMa's elements, MMa's class name is shown before Ma's class name in format as "MMa's name: Ma's name" to indicate each element model's metamodel. Plus, element's attributes and operations are omitted if they are inherited from metamodel. Only new properties are presented in the following models.

- E-learning System (eLearningSys): it is system top level abstract class means whole e-learning system.

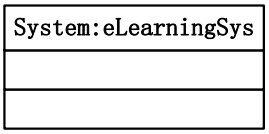


Figure 6-11. Model for eLearningSys

- Learner: it is class for learner entities.

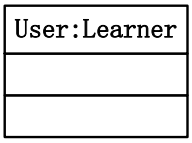


Figure 6-12. Model for Learner

- Coach: it is class to describe coach.

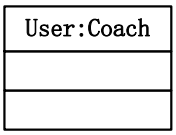


Figure 6-13. Model for Coach

- Administrator (Admin): it is class presenting system administrators.

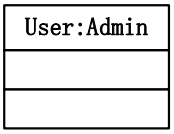


Figure 6-14. Model for Admin

- Learner Record (LearnerRec): it is storage for learner's records including general information, learning history, and learning progress.

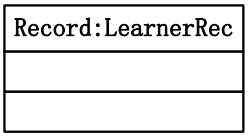


Figure 6-15. Model for LearnerRec

- Coach Record (CoachRec): it is storage for coach's records basically including general information and relative learners' information.

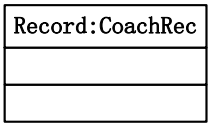


Figure 6-16. Model for CoachRec

- Administrator Record (AdminRec): it is storage for system administrator's record including general information and operation history.

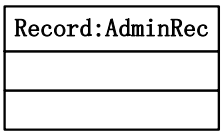


Figure 6-17. Model for AdminRec

- Content: it is class presenting learning contents such as pages with text, picture, and sound as learning material.

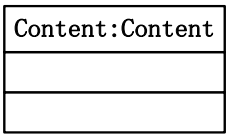


Figure 6-18. Model for Content

- Resource: it is class to store learning resources including various format files and contents.

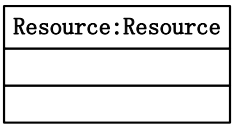


Figure 6-19. Model for Resource

- Navigation: it is a class to indicate system navigation method.

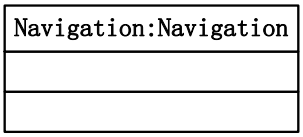


Figure 6-20. Model for Navigation

The models' relationships are based on MMa's relationship. It can be simply shown as below figure.

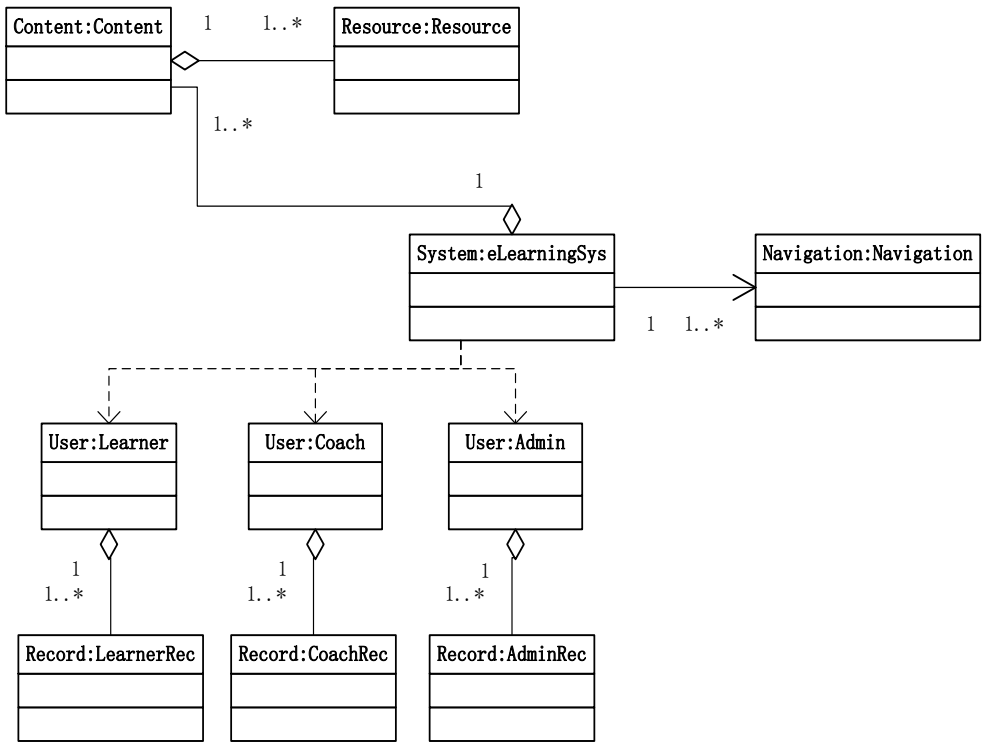


Figure 6-21. Relationship in Source Model

6.4 Target Model and Metamodel of Target Model

In this section, the Platform Specific Model (PSM) is the target model (Mb) representing a platform-independent definition of e-learning system. The PSM is defined in the accompanying normative UML model and UML profile based metamodel. The metamodel of target model (MMb) is also defined based on UML profile and MVC structure. Elements of this model are presented in this clause to clarify and provide guidance on this model.

6.4.1 Fundamental Model Elements for Metamodel of Target Model

Based on MVC structure, the MMb is divided into three catalogues: view, controller, and model. The three parts of MMb will be presented as following three sections including MMb:View, MMb:Controller, and MMb:Model.

6.4.1.1 MetaModel for Target Model: View

Based on UML Profile extension method, there are fundamental model elements defined for MMb:View, including *"Login", "Logout", "User Information (UserInfo)", "Content"*, *"Navigation Bar (NavBar)"* and *"View"*.

- **Login**: it is a view allow user to login system with correct username and password.

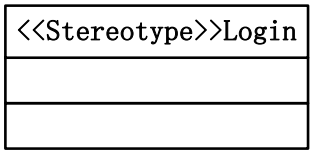


Figure 6-22. Model for Login

- **Logout**: it is a view allow user to logout system.

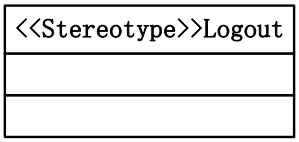


Figure 6-23. Model for Logout

- **User Information (UserInfo)**: It is a view capable to show user information.

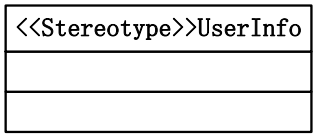


Figure 6-24. Model for UserInfo

- **Content**: It is a view to show specific contents.

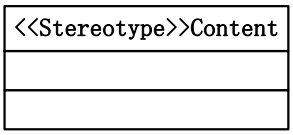


Figure 6-25. Model for Content

- **Navigation Bar (NavBar)**: it is a navigation function bar based on navigation method support.

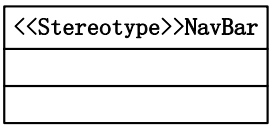


Figure 6-26. Model for NavBar

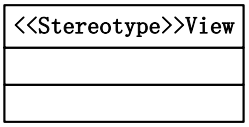


Figure 6-27. Model for View

- **View**: it is a class including rest of views including *"Login", "Logout", "User Information (UserInfo)", "Content"*, and *"Navigation Bar (NavBar)"*.

"View" is an elements combination including *"Login", "Logout", "User Information (UserInfo)", "Content"*, and *"Navigation Bar (NavBar)"*. The follow figure shows this relationship with "Composite" association.

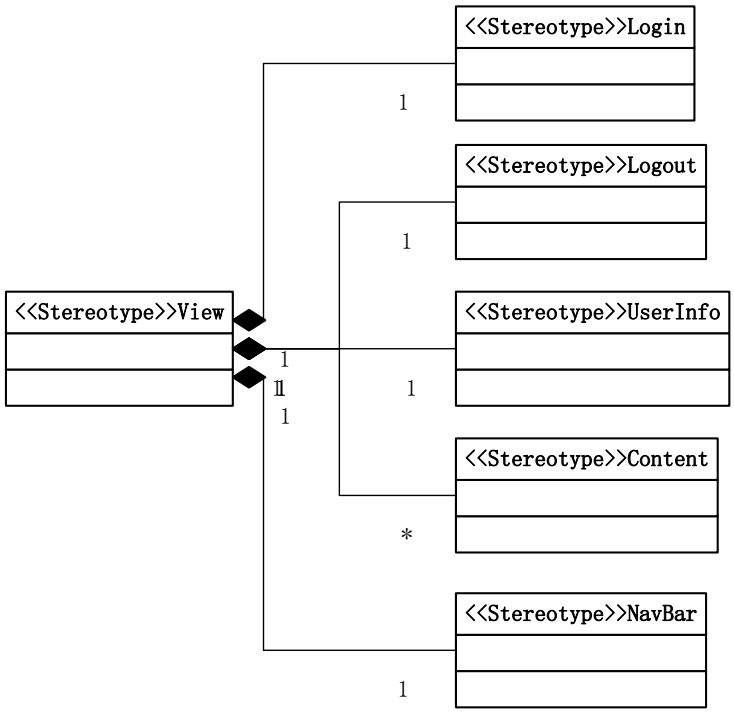


Figure 6-28. Metamodel of Target Model: View Models' Relationship

6.4.1.2 MetaModel for Target Model: Controller

Based on UML Profile extension method, there are fundamental model elements defined for MMb:Controller, including *"Login Response (LoginRsp)", "Logout Response (LogoutRsp)", "User Information Controller (UserInfoCtr)", "Content Controller (ContCtr)", "Navigation Bar Controller (NavBarCtr)", "Evaluation"* and *"Controller"*.

- **Login Response (LoginRsp)**: it is a controller taking responsibility for users' login. There are mainly two operations: to check if the user is capable to login via communication with the corresponding model in MMb:Model, and to response successful or failed GUI which is got from view function in MMb:View.

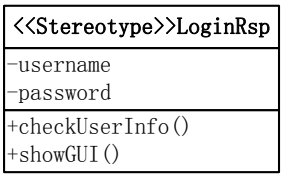


Figure 6-29. Model for LoginRsp

- **Logout Response (LogoutRsp)**: it is a controller taking responsibility for users' logout. There are mainly two operations including to update user's record in database via MMb:Model, and to response logout GUI.

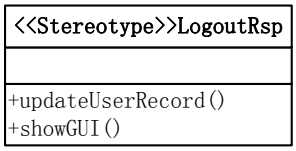


Figure 6-30. Model for LogoutRsp

- **User Information Controller (UserInfoCtr)**: it is a controller in charge of changes in user's record. The operation "getUserInfo()" is to achieve users' record database via model in MMb:Model. The result in "getUserInfo()" is parameter in "showGUI()". When user requests an update for his/her information, the opearation "updateUserInfo()" is applied to request a database update with MMb:Model.

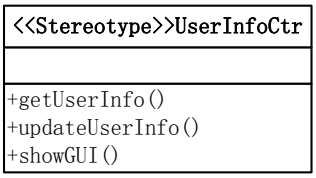


Figure 6-31. Model for UserInfoCtr

- **Content Controller (ContCtr)**: it is a controller working on learning contents. Operation "getContents()" in charge of achieve requested contents from learning resources database. The results of "getContents" are parameters of "showGUI" to respond user corresponding interface with learning contents.

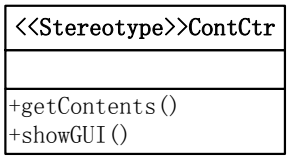


Figure 6-32. Model for ContCtr

- **Navigation Bar Controller (NavBarCtr)**: it is a controller to generate a navigation bar. Operation "getList()" is to organise navigation list based on learning resource database and user record. The results in "getList()" are parameters in "showGUI()" to respond user.

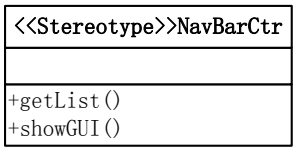


Figure 6-33. Model for NavBarCtr

- **Evaluation**: it is a controller working on evaluation. Operation "doEvaluation()" is in charge of finish evaluation process via automatic settings and communication with Coach. Evaluation results are parameters in "updateUserRecord()" to update corresponding database. User will get evaluation results via "showGUI()".

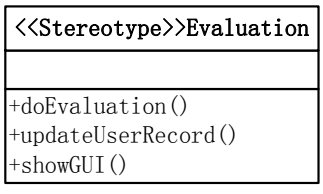


Figure 6-34. Model for Evaluation

- **Controller**: it is an elements combination including *"Login Response (LoginRsp)", "Logout Response (LogoutRsp)", "User Information Controller (UserInfoCtr)", "Content Controller (ContCtr)"*, *"Navigation Bar Controller (NavBarCtr)",* and *"Evaluation"*. There are two kernel operations: "showGUI()" is to transfer data to view and request specific GUI from MMb:View; and "connectModel()" is a operation to communicate with MMb:Model.

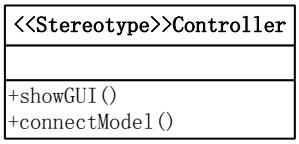


Figure 6-35. Model for Controller

The following figure shows above controllers' relationship with "Composite" association.

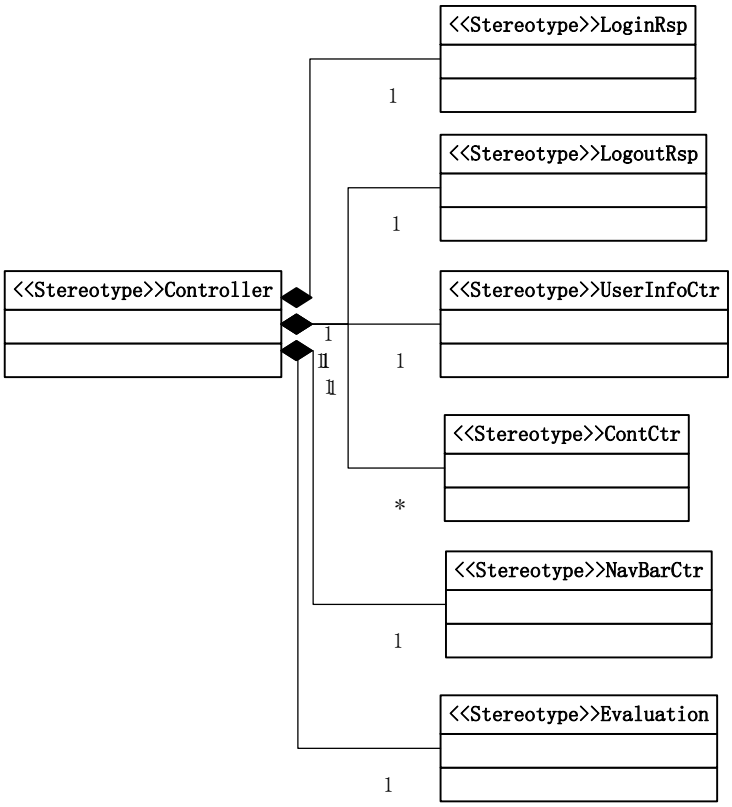


Figure 6-36. MetaModel for Target Model: Controller Models' Relationship

6.4.1.3 MetaModel for Target Model: Model

Based on UML Profile extension method, there are fundamental model elements defined for MMb:Model including *"Database Connection (DBconn)", "Record Database Connection (RecDBConn)", "Resource Database Connection (ResDBConn)", and "Model"*.

- **Database Connection (DBconn)**: it is aims to response communication with databases. Mainly, there are two operations to connect and to disconnect database. This is a parent class of *"Record Database Connection (RecDBConn)"* class and *"Resource Database Connection (ResDBConn)"* class.

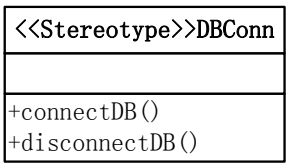


Figure 6-37. Model for DBConn

- **Record Database Connection (RecDBConn)**: it is a model specific for record databases. It directly inherits operation "connectDB()" and "disconnectDB()" from its parent model *"DBConn"*.

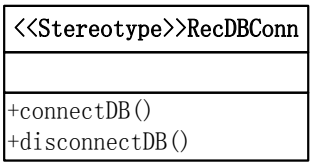


Figure 6-38. Model for RecDBConn

- **Resource Database Connection (ResDBConn)**: it is a model specific for resource databases. Same as *"RecDBConn"*, it directly inherits operation "connectDB()" and "disconnectDB()" from its parent model *"DBConn"*.

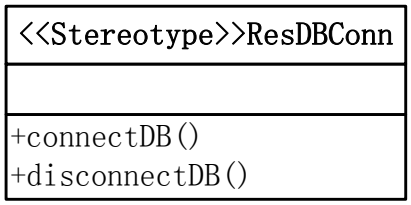


Figure 6-39. Model for ResDBConn

- **Database Edit (DBEdit)**: it is a model in charge of edits on databases including "insert()", "update()", and "delete()".

Figure 6-40. Model for DBEdit

- **Model**: it is an elements combination including *"DBConn"* with its children models, and *"DBEdit"*.

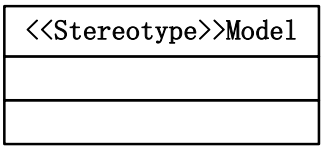


Figure 6-41. Model for MVC's Model

The Figure 6-42 shows above models' relationship with "Composite" and "Generalisation" associations.

Figure 6-43 shows relationship between MMb: View, MMb: Controller, and MMb: Model in one system. Controller is communicating with view and model. View and model cannot communication directly.

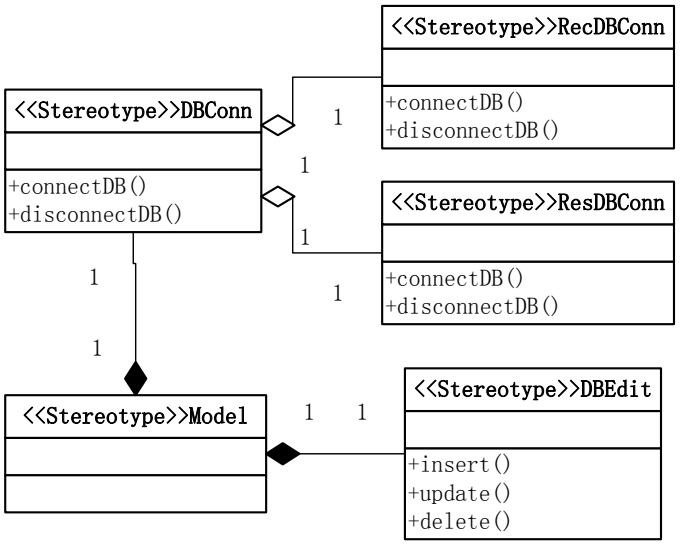


Figure 6-42. MetaModel for Target Model: Model Elements Relationship

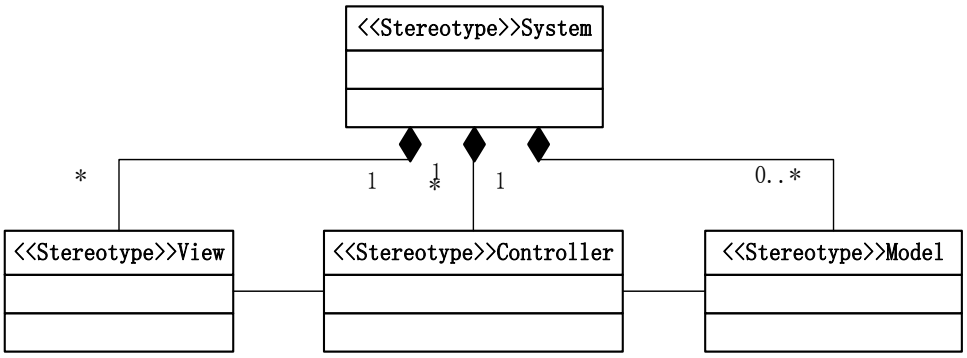


Figure 6-43. View, Controller, and Model Metamodel Relationship

6.4.2 Fundamental Model Elements for Target Model

In this section, there are definitions and specifications for Mb's fundamental model elements. Based on MMb's definition, there are three groups of Mb elements, including "Mb: View", "Mb: Controller", and "Mb: Model".

6.4.2.1 Target Model: View

The Mb:View have to be divided into three groups according to different user's character in e-learning system. The three groups are: Learner's View, Coach's View,

and Administrator's View (Admin's View). Each group contains specific models to achieve users' requirements.

- **Learner View**: it is a view working for learners. It is composed by *"LearnerLogin"*, *"LearnerLogout"*, *"LearnerInfo"*, *"LearnerContent"*, and *"LearnerNavBar"*. The following figure shows each view component and the associations between them.

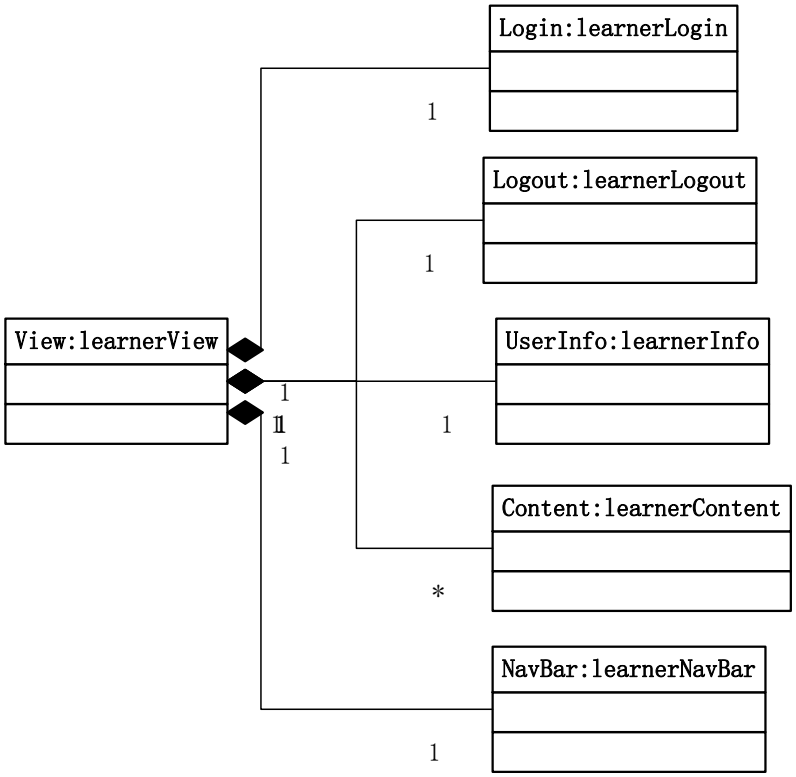


Figure 6-44. Model for Learner's View

- **Coach View**: it is a view working for coaches. It is composed by *"CoachLogin"*, *"CoachLogout"*, *"CoachInfo"*, *"CoachContent"*, *"CoachNavBar"*, and *"CoachEvaluation"*. The following figure shows each view component and the associations between them.

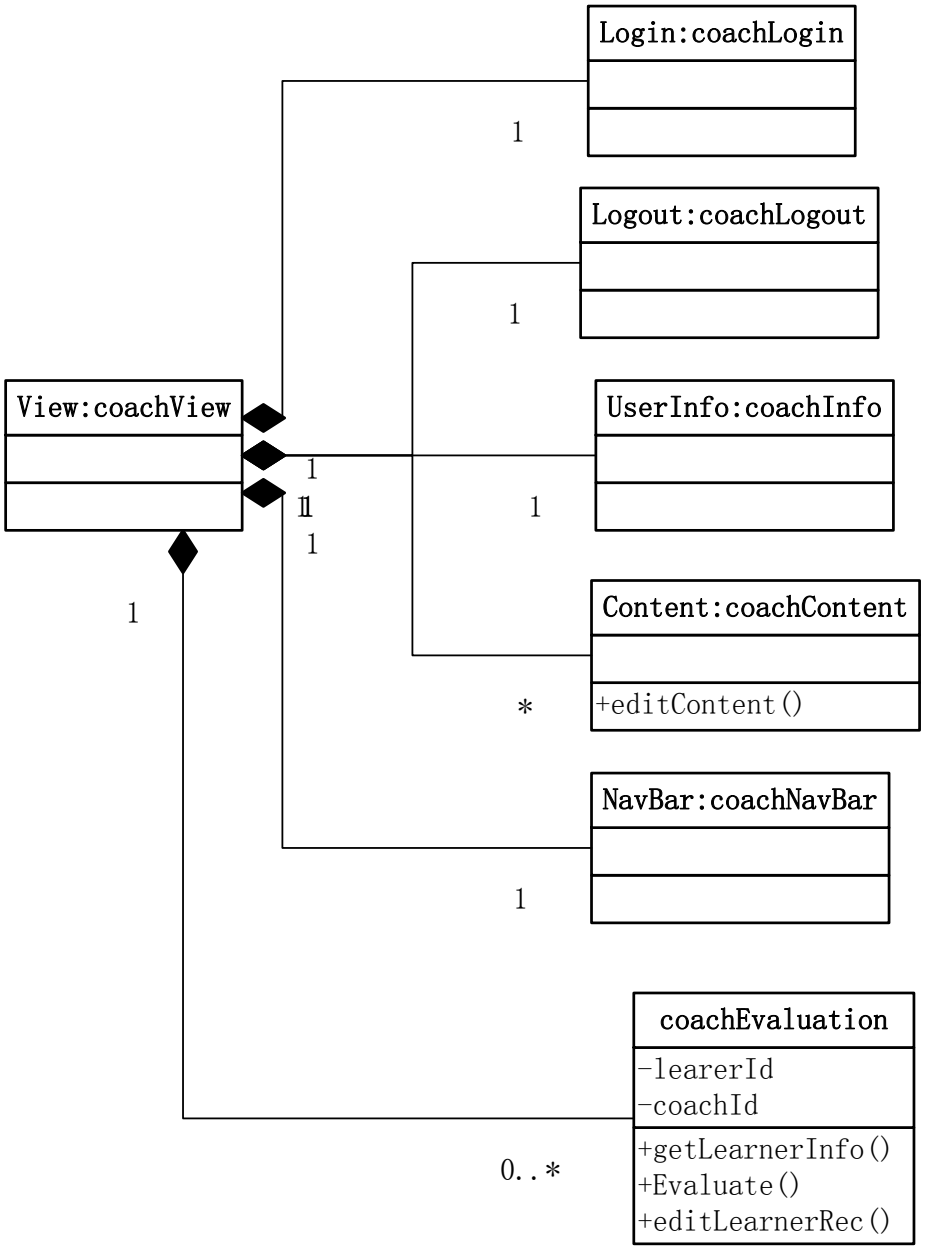


Figure 6-45. Model for Coach's View

- **Admin View**: it is a view working for learners. It is composed by *"AdminLogin"*, *"AdminLogout"*, *"AdminInfo"*, *"AdminContent"*, *"AdminNavBar"*, and *"SystemMaintenance"*. The following figure shows each view component and the associations between them.

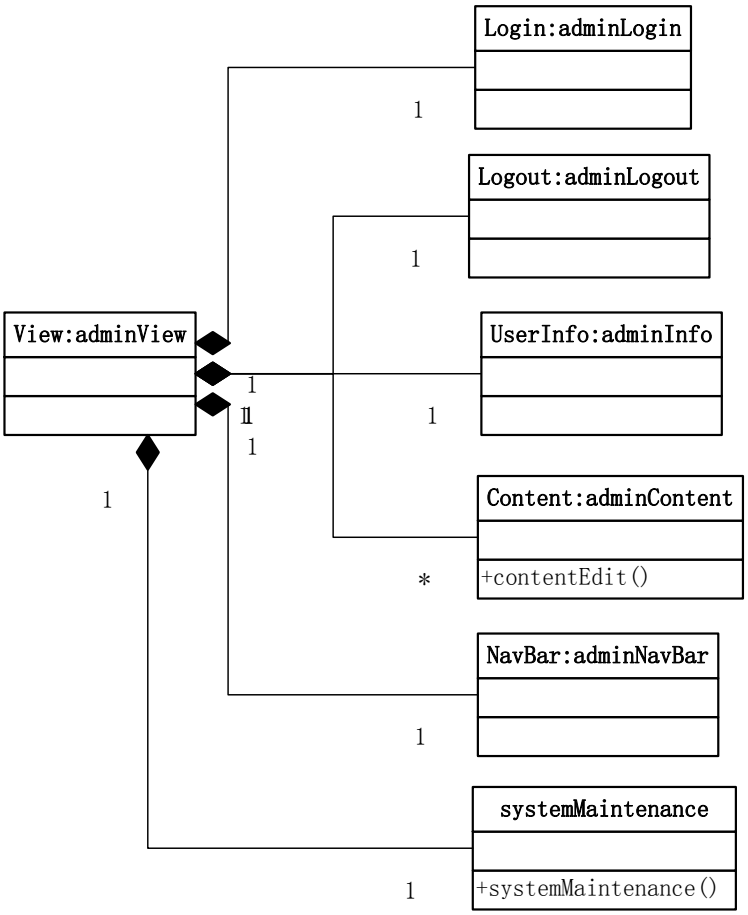


Figure 6-46. Model for Admin's View

6.4.2.2 Target Model: Controller

- E-learning controller is in charge of communication between user and the rest two parts in MVC which is model and view. Based on UML Profile extension method, there are fundamental model elements defined for Mb:Controller including *"Login Response (LoginRsp)", "Logout Response (LogoutRsp)", "User Information Controller (UserInfoCtr)", "Content Controller (ContCtr)", "Navigation Bar Controller (NavBarCtr)", "Evaluation"* and *"elearningController"*. The following figure shows each view component and the associations between them.

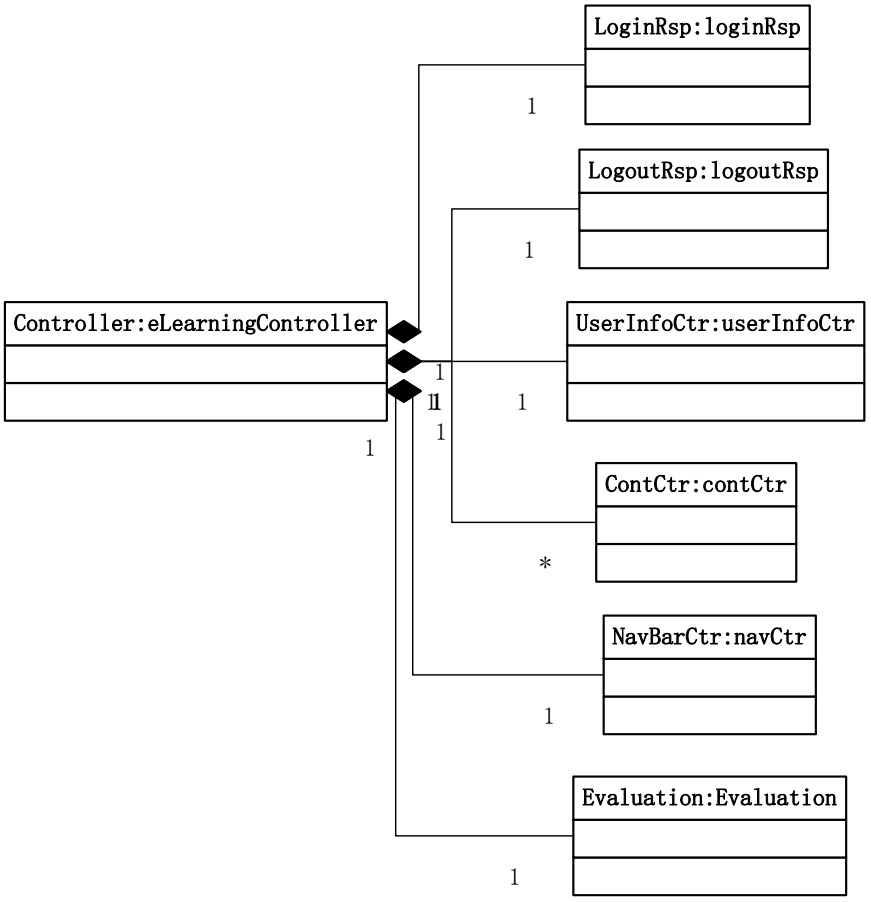


Figure 6-47. Model for Mb:Controller

6.4.2.3 Target Model: Model

- Based on UML Profile extension method, there are fundamental model elements defined for Mb:Model including *"E-learning Database Connection (elarningDBconn)", "Admin Record Database Connection (adminRecDBConn)", "Coach Record Database Connection (coachRecDBConn)", "Learner Record Database Connection (learnerRecDBConn)", "Learning Resource Database Connection (learningResDBConn)", "Coach Database Edit (coachDBEdit)" and "elearningModel"*. The following figure shows each view component and the associations between them.

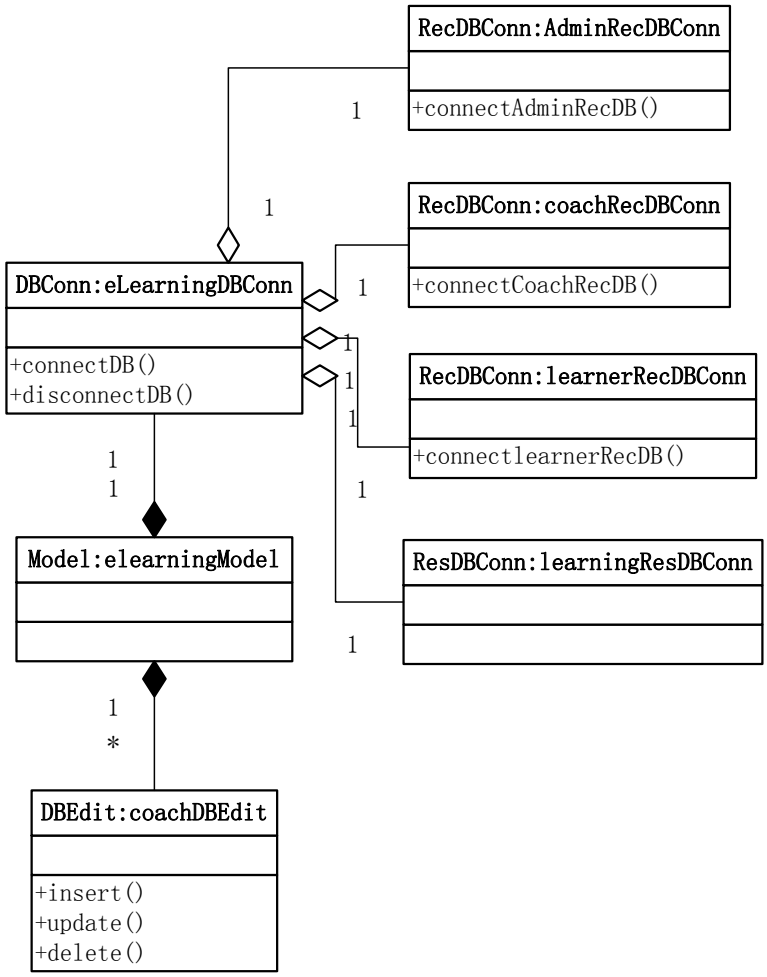


Figure 6-48. Model for Mb:Model

6.5 Transformation Rules

This section specific mapping rules for defined source model and target model based on QVT. The rules are described as formulas with constraints.

6.5.1 Metamodel for Transformation Model

In this section, the metamodel of transformation model is defined with specifications. Because proposed MMa and MMb are based on UML profile, the Metamodel for

Transformation Model (MMt) is mainly taking the responsibility for transformations between defined UML profiles.

In MMa, there are stereotype classes including system, user, record, content, resource, and navigation. Each class includes its own attributes and operations. There are also various associations between those stereotype classes. All of them are the elements to be mapped into MMb.

The transformation rules in this stage is catalogued based on individual stereotype classes describing as following groups.

Group 1. Stereotype class: System

It is mapping to MMb as a stereotype class named "System" as well. In addition, there are three stereotype classes derived automatically including "View", "Controller", and "Model" based on MMb's basic structure.

No.	Well-formed Rules	Explanations
1	$\mathbf{Sc} \Rightarrow \mathbf{S'c}(\mathbf{Vc}, \mathbf{Cc}, \mathbf{Mc})$ $Sc = MMa{:}System.Class$ $S'c\ (Vc) = MMb{:}System.Class,$ $Vc = MMb{:}View.Class,$ $Cc = MMb{:}Controller.Class,$ $Mc = MMb{:}Model.Class.$	MMb's "System" is generated from MMa's "System" which including View, Controller and Model.
2	$\mathbf{Sa}(\mathbf{Id}) \Rightarrow \mathbf{S'a}(\mathbf{Id})$ $Sa = MMa{:}System.Attribute$ $S'a = MMb{:}System.Attribute$	Attribute "Id" in MMb's "System" comes from MMa's "System" attribute "Id" directly considering they are the basic system information.
3	$\mathbf{Sa}(\mathbf{Owner}) \Rightarrow \mathbf{S'a}(\mathbf{Owner})$	Attribute "Owner" in MMb's "System"

	$Sa = MMa: System. Attribute$ $S'a = MMb: System. Attribute$	comes from MMa's "System" attribute "Owner" directly considering they are the basic system information.
4	$\mathbf{Sa(Title)} \Rightarrow \mathbf{Va(Title)}$ $Sa = MMa: System. Attribute$ $Va = MMb: View. Attribute$	Attribute "Title" in MMa's "System" is the element to generate attribute "Title" in MMb's "View" class because the title value is an element in users' GUI.
5	$\mathbf{Sa(Content)} \Rightarrow \mathbf{Va(Content)}$ $Sa = MMa: System. Attribute$ $Va = MMb: View. Attribute$	Attribute "Content" in MMa's "System" is the element to generate attribute "Content" in MMb's "View" class because the content value is an element in users' GUI.
6	$\mathbf{Sa(Navigation)} \Rightarrow \mathbf{Va(NavBar)}$ $Sa = MMa: System. Attribute$ $Va = MMb: View. Attribute$	Attribute "Navigation" in MMa's "System" is the element to generate attribute "NavBar" in MMb's "View" class because this attribute's value is an element in users' GUI.
7	$\mathbf{So(getContent)} \Rightarrow \mathbf{M(Mo)}$ $Sa = MMa: System. Operation$ $M = MMb: ContCtr$ $Mo = MMb: ContCtr. Operation$	Operations in MMa's "System" maps to MMb as stereotype classes belongs to "Controller" class. There for the operation "getContent()" is transformed to MMb's "ContCtr".
8	$\mathbf{So(getNavigation)} \Rightarrow \mathbf{N(No)}$ $Sa = MMa: System. Operation$ $N = MMb: NavBarCtr$ $No = MMb: NavBarCtr. Operation$	Operations in MMa's "System" maps to MMb as stereotype classes belongs to "Controller" class. There for the operation "getNavigation()" are transformed to MMb's "NavBarCtr".

Table 6-1. Group 1 Rules and Its Explanations: Stereotype class: System

Based on rules' explanation on group 1, it is easy to understand the following group rules. The rest groups are defined and explained as following tables for stereotype classes "Content", "User", "Record", "Resource" and "Navigation".

Group 2. Stereotype class: Content

No.	Well-formed Rules	Explanations
9	$\mathbf{Cc} \Rightarrow \mathbf{V(C'c)}$ $Cc = MMa: Content.Class$ $V = MMb: View$ $C'c = MMb: Content.Class$	"Content" class on View of MMb is generated from MMa's "Content".
10	$\mathbf{Ca(Id)} \Rightarrow \mathbf{C'a(Id)}$ $Ca = MMa: Content.attribute$ $C'a = MMb: Content.attribute$	Attribute "Id" in MMa's "Content" is the element to generate attribute "Id" in MMb's "Content" class.
11	$\mathbf{Ca(Name)} \Rightarrow \mathbf{C'a(Name)}$ $Ca = MMa: Content.attribute$ $C'a = MMb: Content.attribute$	Attribute "Name" in MMa's "Content" is the element to generate attribute "Name" in MMb's "Content" class.
12	$\mathbf{Co(getResource)} \Rightarrow \mathbf{Ro(getResource)}$ $Co = MMa: Content.operation$ $Ro = MMb: ContCtr.operation$	Operations in MMa's "Content" is mapping to MMb as stereotype classes belongs to "ContCtr" (Content Controller) class. There for the operation "getResource()" is transformed to MMb's "ContCtr".
13	$\mathbf{Co(postContent)} \Rightarrow \mathbf{Ro(postContent)}$ $Co = MMa: Content.operation$	Operations in MMa's "Content" is mapping to MMb as stereotype

	$Ro = MMb: ContCtr.operation$	classes belongs to "ContCtr" (Content Controller) class. There for the operation "postContent()" is transformed to MMb's "ContCtr".

Table 6-2. Group 2 Rules and Its Explanations: Stereotype class: Content

Group 3. Stereotype class: User

No.	Well-formed Rules	Explanations
14	$\mathbf{Uc \Rightarrow V(Xc) + C'(Yc)}$ $Uc = MMa: User.class$ $V = MMb: View$ $Xc = MMb: UserInfo.class$ $C' = MMb: Controller$ $Yc = MMb: UserInfoCtr.class$	Details of "User" class on View and Controller of MMb is generated from MMa's "User".
15	$\mathbf{Ua \Rightarrow Xa}$ $Ua = MMa: User.attribute$ $Xa = MMb: UserInfo.attribute$	Attributes on MMb's "UserInfo" class is mapped from attributes on MMa's "User" class.
16	$\mathbf{Uo \Rightarrow Yo}$ $Uo = MMa: User.operation$ $Yo = MMb: UserInfoCtr.opeartion$	Each operation on MMb's "UserInfoCtr" class is mapped from corresponding operation on MMa's "User" class.

Table 6-3. Group 3 Rules and Its Explanations: Stereotype class: User

Group 4. Stereotype class: Record

No.	Well-formed Rules	Explanations

17	**Rc ⇒ M(D(RDc))** $Rc = MMa: Record.class$ $M = MMb: Model$ $D = MMb: DBConn$ $RDc = MMb: RecDBConn.class$	"RecDBConn" (Record Database Connection) class on MMb is generated from MMa's "Record".
18	**Ra ⇒ EDo** $Ra = MMa: Record.attribute$ $EDo = MMb: DBEdit.operation$	Some operations on MMb's "DBEdit" class are mapped from corresponding attributes on MMa's "Record" class.
19	**Ro ⇒ RDo** $Ro = MMa: Record.operation$ $RDo = MMb: RecDBConn.operation$	Some operations on MMb's "RecDBConn" class are mapped from corresponding operation on MMa's "Record" class.

Table 6-4. Group 4 Rules and Its Explanations: Stereotype class: Record

Group 5. Stereotype class: Resource

No.	Well-formed Rules	Explanations
20	**Qc ⇒ M(D(QDc))** $Qc = MMa: Resource.class$ $M = MMb: Model$ $D = MMb: DBConn$ $QDc = MMb: ResDBConn.class$	"ResDBConn" (Resource Database Connection) class of MMb is generated from MMa's "Resource" class.
21	**Qa ⇒ EDo** $Qa = MMa: Resource.attribute$ $EDo = MMb: DBEdit.operation$	Some operations on MMb's "DBEdit" class are mapped from corresponding attributes on MMa's "Resource" class.

22	**Qo ⇒ QDo** $Qo = MMa: Resource.operation$ $QDo = MMb: ResDBConn.operation$	Some operations on MMb's "RecDBConn" class are mapped from corresponding operation on MMa's "Resource" class.

Table 6-5. Group 5 Rules and Its Explanations: Stereotype class: Resource

Group 6. Stereotype class: Navigation

No.	Well-formed Rules	Explanations
23	**Nc ⇒ V(NBc) + C′(N′c)** $Nc = MMa: Navigation.class$ $V = MMb: View$ $NBc = MMb: NavBar.class$ $C' = MMb: Controller$ $N'c = MMb: NavBarCtr.class$	"Navigation" class on MMa will generate MMb's "NavBarCtr" (Navigation Bar Control) class and MMb's "NavBar" (Navigation Bar) class.
24	**Na ⇒ NBa** $Na = MMa: Navigation.attribute$ $NBa = MMb: NavBar.attribute$	Attributes of "NavBar" on MMb are mapped from attributions on MMa's "Navigation" class.
25	**No ⇒ N′o** $No = MMa: Navigation.operation$ $N'o = MMb: NavBarCtr.operation$	Operations of "NavBarCtr" on MMb are mapped from operations on MMa's "Navigation" class.

Table 6-6. Group 6 Rules and Its Explanations: Stereotype class: Navigation

Group 7. Compulsory Rules

Besides the other six specific group rules, there are also 13 compulsory rules predefined general constraints of the proposed transformation process.

No.	Well-formed Rules	Explanations
26	$\exists!\ \mathbf{L1}:\mathbf{V}(\mathbf{L1})$ $L1 = MMb:Login$ $V = MMb:View$	There is existing and only existing one “Login” on MMb’s “View”.
27	$\exists!\ \mathbf{L2}:\mathbf{V}(\mathbf{L2})$ $L2 = MMb:Logout$ $V = MMb:View$	There is existing and only existing one “Logout” on MMb’s “View”.
28	$\exists\ \mathbf{X}:\mathbf{V}(\mathbf{X})$ $X = MMb:UserInfo$ $V = MMb:View$	There is at least one “UserInfo” class existing on MMb’s “View”.
29	$\exists\ \mathbf{T}:\mathbf{V}(\mathbf{T})$ $T = MMb:Content$ $V = MMb:View$	There is at least one “Content” class existing on MMb’s “View”.
30	$\exists\ \mathbf{NB}:\mathbf{V}(\mathbf{NB})$ $NB = MMb:NavBar$ $V = MMb:View$	There is at least one “NavBar” class existing on MMb’s “View”.
31	$\exists!\ \mathbf{D}:\mathbf{M}(\mathbf{D})$ $D = MMb:DBConn$ $M = MMb:Model$	There is existing and only existing one “DBConn” (Database Connection) on MMb’s “Model”.
32	$\exists\ \mathbf{ED}:\mathbf{M}(\mathbf{ED})$ $ED = MMb:DBEdit$ $M = MMb:Model$	There is at least one “DBEdit” (Database Edit) class existing on MMb’s “Model”.
33	$\exists!\ \mathbf{L1'}:\mathbf{C'}(\mathbf{L1'})$ $L1' = MMb:LoginRsp$ $C' = MMb:Controller$	There is existing and only existing one “LoginRsp” (Login Response) on MMb’s “Controller”.

34	$\exists!\ \mathbf{L2'}: \mathbf{C'}(\mathbf{L2'})$ $L2' = MMb: LogoutRsp$ $C' = MMb: Controller$	There is existing and only existing one "LogoutRsp" (Logout Response) on MMb's "Controller".
35	$\exists\ \mathbf{Y}: \mathbf{C'}(\mathbf{Y})$ $Y = MMb: UserInfoCtr$ $C' = MMb: Controller$	There is at least one "UserInfoCtr" (User Information Controller) class existing on MMb's "Controller"
36	$\exists\ \mathbf{R}: \mathbf{C'}(\mathbf{R})$ $R = MMb: ContCtr$ $C' = MMb: Controller$	There is at least one "ContCtr" (Content Controller) class existing on MMb's "Controller"
37	$\exists\ \mathbf{N'}: \mathbf{C'}(\mathbf{N'})$ $N' = MMb: NavBarCtr$ $C' = MMb: Controlle$	There is at least one "NavBarCtr" (Navigation Bar Controller) class existing on MMb's "Controller"
38	$\exists\ \mathbf{E}: \mathbf{C'}(\mathbf{E})$ $E = MMb: Evalutaion$ $C' = MMb: Controller$	There is at least one "Evaluation" class existing on MMb's "Controller"
39	$\mathbf{S'} = \mathbf{V} \cup \mathbf{C'} \cup \mathbf{M}$ $S' = MMb: System$ $V = MMb: View$ $C' = MMb: Controller$ $M = MMb: Model$	MMb's "System" is the combination of "View", "Controller" and "Model".

Table 6-7. Group 7 Rules and Its Explanations: Compulsory Rules

6.5.2 Transformation Model

The transformation model Mt is designed to map Ma to Mb. It is based on MMt proposed in last section. The idea about transformation model (Mt) is to support

additional information for MMt. Therefore, it is a model allows developer to modify under specific conditions.

There are few rules defined as a core Mt.

Rule 1: if there are more than one class that are inherited from stereotype class "user", then create specific "view" for each of them. View is named in this format: "userclass.name&View"

Rule 2: Each individual "View" should contains sub-views.

Rule 3: if there are more than one class that are inherited from stereotype class "user", then create specific "DBConn" for each of them. The name format is "userclass.name&RecDBConn"

Compulsory Rules:

Rule 4: $\exists!\ V{:}S'(V)$

$$V = Mb{:}View, S' = Mb{:}System.$$

Rule 5: $\exists!\ C'{:}S'(C')$

$$C' = Mb{:}Controller, S' = Mb{:}System.$$

Rule 6: $\exists!\ M{:}S'(M)$

$$M = Mb{:}Model, S' = Mb{:}System$$

Rule 7: $S' = \sum V \cup C' \cup M$

$$S' = Mb{:}eLearningSystem$$

$$\sum V = Mb{:}Views$$

$$C' = Mb: Controller$$

$$M = Mb: Model.$$

6.6 Mapping Rules Implementation

In this chapter, "Class Diagram" are applied to present simple source PIM and target PSMs.

This is a simple Metamodel for PIM represented as a class diagram. There are mainly two classes: system and content. The relationships show that system is a combination of 1 or more contents. Meanwhile, content can be a part of content.

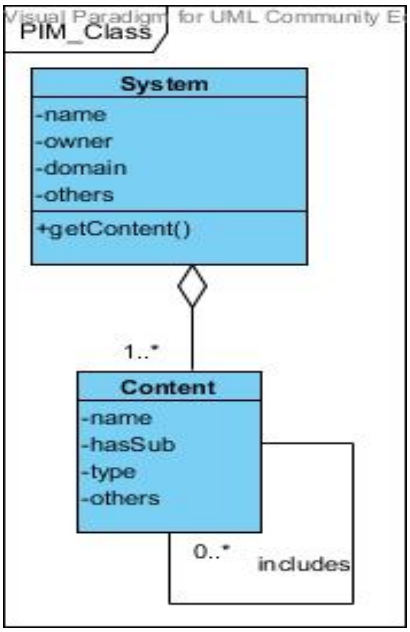


Figure 6-49. A Simple Metamodel for Source Model

It is not using the MMa defined in last section; because this section is aiming to prove proposed mapping rules are suitable for general situations.

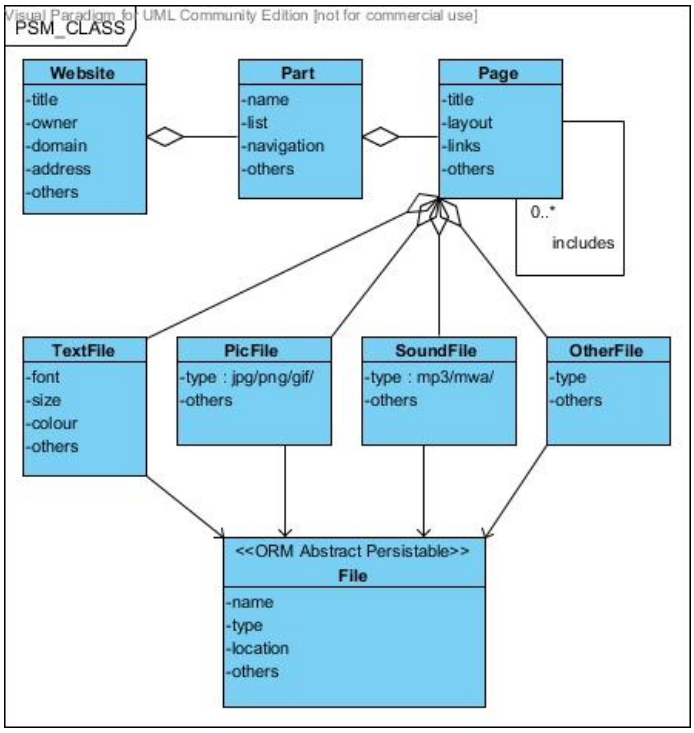


Figure 6-50. A Simple Metamodel for Target Model

Figure 6-50 is a simple Metamodel for PSM represented as a class diagram. Figure 6-51 is a PIM represented as a class diagram based on metamodel in Figure 6-49. There are many classes with attributes and operations. Mainly they are two groups based on dependent metaclass.

- System based: EARSII is a class based on metaclass system.
- Content based: The rest of classes are based on metaclass content including “Listening”, “Understanding”, and “Doing”. The “Understanding” class contains more contents as a tree structure.

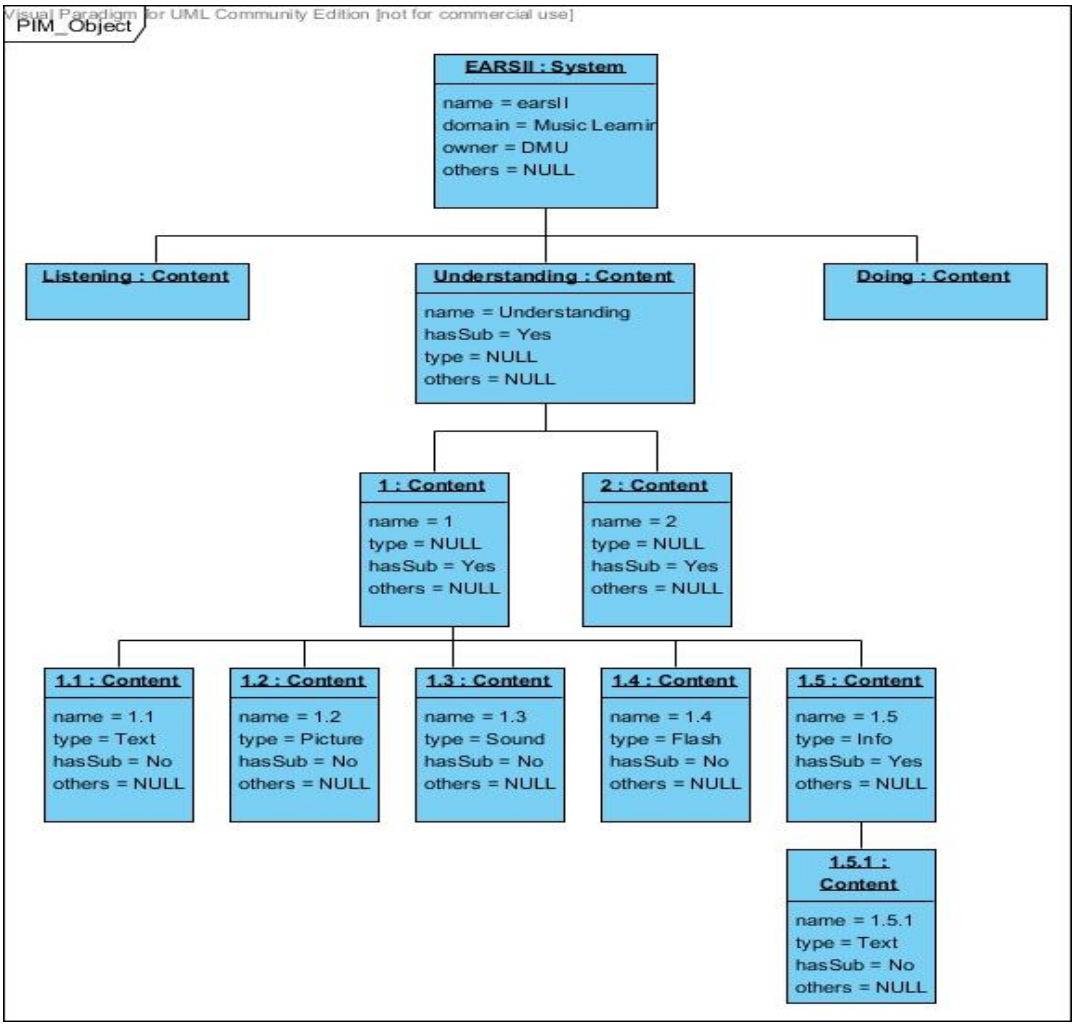


Figure 6-51. A Source Model Example

Mapping Process

The proposed rules are the kernel of transformation model, Mt, which can be presented as below to execute in this implementation,

```
If (PIM.hasSub=Yes) {
    If (ClssName=”System”) {
        New Object.Website;
        Website.Object.name=System.Object.name;
        Website.Object.owner=System.Object.owner;
```

```
            Website.Object.domain=System.Object.domain;
            Website.Object.others=System.Object.others;
            Website.Object.address=NULL;
        }
        Elseif (Classname=”Content”){
            If (Content.Object.subcontent doesn’t include leaf node) {
            New Object.Part;
            Part.Object.name=Content.Object.name;
            Part.Object.list=None (options: type1/type2/type3/…/None);
            Part.Object.navigation=None (Options: Nav1/Nav2/…/None);
            Part.Object.others=Content.Object.others;
            }
            Elseif (Content.Object.subcontent includes leaf node) {
            New Object.Page;
            Page.Object.title=Content.Object.name;
            Page.Object.layout=None (Options: Layout1/2/…/None);
            Page.Object.links=NULL;
            Page.Object.others=Content.Object.others;
            }
            Else {
            Error 2;
            }
        }
        Else { Error 1; }
    } Elseif (PIM.hasSub=”No”) {
        If (Content.Object.type=”text”) {
        New Object.TextFile;
```

```
TextFile.Object.name=Content.Object.name;
TextFile.Object.type=Null (Optinal: .txt/.doc/.docx/…);
TextFile.Object.location=Null;
TextFile.Object.Font=Null (Op: Calibri/Arial/…);
TextFile.Object.size=Null (Op: 8/9/10/…/72);
TextFile.Object.colour=black(default) (Op: black/red/…);
TextFile.Object.others=Content.Object.others;
}
If (Content.Object.type=”picture”) {
New Object.PicFile;
PicFile.Object.name=Content.Object.name;
PicFile.Object.type=Null (Optinal: .jpg/.gif/.png/…);
PicFile.Object.location=Null;
PicFile.Object.size=Null (Op: Length*Width);
PicFile.Object.others=Content.Object.others;
}
If (Content.Object.type=”sound”) {
New Object.SoundFile;
```

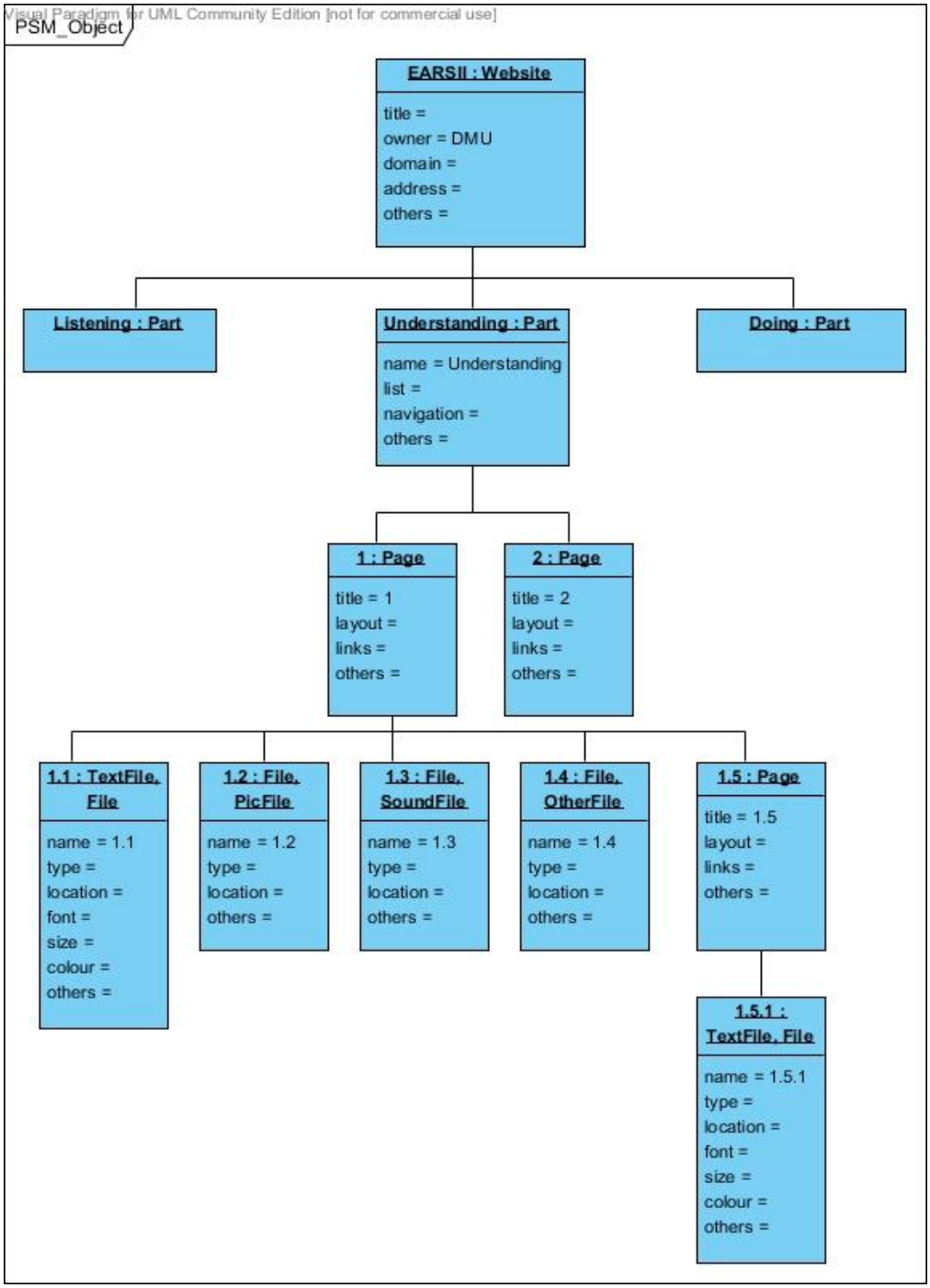


Figure 6-52. A Target Model Example

SoundFile.Object.name=Content.Object.name;

SoundFile.Object.type=Null (Optinal: .mp3/.wma/…);

SoundFile.Object.location=Null;

SoundFile.Object.others=Content.Object.others;

} Else {

New Object.OtherFile;

OtherFile.Object.name=Content.Object.name;

OtherFile.Object.type=Null (Optinal: .*);

OtherFile.Object.location=Null;

OtherFile.Object.others=Content.Object.others; }}

Followed above process, Figure 6-52 is generated as a target model which is represented as a class diagram.

6.7 Summary

In this chapter, the PIM to PSMs transformation is specified:

- A transformation architecture is proposed based on MDA and proposed evolution method. The transformation structure is designed and specified with models and processes.

- Source model (Ma) is defined in the accompanying normative UML model. The metamodel (MMa) is also provided on a UML profile basis. Fundamental elements and relationships are clarified for each model.

- Target model (Mb) is defined in the accompanying normative UML model. The metamodel (MMb) is also provided on a UML profile basis. Based on proposed structure, there are “View”, “Model”, and “Controller” models on both Mb and MMb. Fundamental elements and relationships are clarified for each model.

- Mapping rules for defined source model and target model are defined and specified. The rules are described as formulas with constraints. There are rules

working on transformation model (Mt) and metamodel for transformation model (MMt) based on its level.

- There is a piece of simply source model employed as an implementation to prove defined mapping rules' feasibility.

Chapter 7
Case Study: ElectroAcoustic Resource Site II

Objectives

- To introduce the ElectroAcoustic Resource Site II project
- To analyse issues on the ElectroAcoustic Resource Site II system
- To present MDA-based development approach on the ElectroAcoustic Resource Site II
- To research toolkit supporting
- To present screen shot of prototype

7.1 Overview

The MDA-based evolution approach focuses on domain-specific modelling and model to model and model to code transformations. In this chapter, a music e-learning system named the ElectroAcoustic Resource Site II (EARSII) is presented as a case study to illustrate presented approach and implementation.

The proposed MDA-based evaluation approach includes the following steps, from CIM modelling to PIM Generation, then transformation from PIM to PSMs, finally codes are generated from PSMs. The CIM and PIM are constructed based on e-learning domain models and ontology-based vocabularies extraction. PSMs are

transformed from PIM via designed transformation method. Lastly, codes are generated manually instead of tools.

The Pedagogical ElectroAcoustic Resource Sit (EARS II), a music learning system, is demonstration project in this research, showing the implementation of the proposed approach.

7.2 EARSII System Background and Issues

7.2.1 EARSI Basic Introduction

The ElectroAcoustic Resource Site project was first established in 2001, with the aim of providing academic information for researchers in the realm of electroacoustic music. It has been co-ordinated by Leigh Landy and Simon Atkinson at the Music, Technology and Innovation Research Centre (MTI) of De Montfort University Leicester and is supported by the MTI as well as an international consortium.

In 2002 the first version went online containing an English language initial glossary and an index. Today, the website presents in a multi-lingual glossary with definitions of about 500 terms, which are organised under six headers (Discipline of Studies, Genre and Categories, Musicology of Electroacoustic Music, Performance Practise and Presentation, Sound Production and Manipulation, Musical Structure), as well as an extensive bibliography of over three thousand items and a thesaurus. Furthermore, it provides online publications. Currently, the website is translated in five languages: English, German, French, Italian and Spanish. In collaboration with the University of Beijing a Chinese version of the website is in progress (the project is called CHEARS, which stands for Chinese Electroacoustic Resource Site). A Greek translation is also being prepared.

Although in the description of the ElectroAcoustic Resource Site it is stated, "The project will cite or directly link to texts, titles, abstracts, images, audio and audiovisual files, and other relevant format" (EARS 1), the project is mainly text based. Audio files or other media formats are only available via links. As the EARS website addresses researchers and experienced listeners (such as composers) in particular, this is sufficient information.

Further information about EARS I can be found in Landy's paper [10, 44] and article [43].

7.2.2 EARS II

7.2.2.1 EARS II Basic Introduction

In a second part of the project (from now on called EARS II) a new web-based environment will be developed, which addresses the inexperienced listeners, in the first instance children at key stage 3 of the National Curriculum in the United Kingdom (aged 11-14). As text is not appropriate as the sole means of presenting information to children in that age group, EARS II will use the information from the current EARS website (EARS I), but will present it now didactically within a multimedia environment. While the information on EARS I is organised as a reference, EARS II is based on a pedagogical approach, which means that the organisation and presentation of the information must be adapted. Not only is knowledge transfer related to electroacoustic music necessary; also an understanding must be gained of how it is composed and of course how it sounds: this leads to a better acceptance of electroacoustic music. To realise this aim, the project contains three important components: learning about this music (understanding), listening to it (appreciation) and making it themselves with the help of the software, Sound Organiser (creating). In

combining these elements EARS II is unique amongst the few other pedagogical projects about electroacoustic music.

7.2.2.2 Aims – Content – Method for Understanding/Learning

a) Aims

In these times of far-reaching isolation of a great deal of contemporary arts within society, the project EARS II is designed to help close the gap, at least as far as electroacoustic music is concerned. In being an equally challenging and in equal shares entertaining platform it is intended to broaden the mind of all users (pupils and teachers). Presented as a mix of learning, listening and making electroacoustic music is planned to be placed on the curricula of secondary schools internationally (plans for translation and cultural adoption go beyond this project, but are already in an advanced stage).

The most important aim of EARS II is to introduce children and other inexperienced users to electroacoustic music, as it is an important part of our music culture (see also Problems of Contemporary Music). Not only is knowledge about this music necessary, but also understanding of basic theories and concepts, information on how the music is composed and of course how it sounds, leads to a greater acceptance of electroacoustic music. To realise this aim, the project contains three important components: to provide knowledge, to explain, how to make this music and allows users to listen to this music.

b) Learning Outcomes for EARSII

The following list presents the learning outcomes a user can achieve in learning with EARS II. As the amount of outcomes is huge, naturally not every user will achieve every outcome. However, it shows the wide ranges of skills and knowledge which can be gained within the EARS II environment.

Broad aim: A society whose members is not afraid of electroacoustic music, but instead understand it and can appreciate it and/or make it.

Concrete social objectives:

- familiarity with the body of electroacoustic works
- broad but discriminating musical tastes
- awareness of basic musical design and the general outline of its evolution
- ability to compose or improvise with sounds
- participation in musical activity appropriate to one's interests and talents

Program objectives:

- Knowledge of
    a. a repertoire of electroacoustic music
    b. basic functions of the Sound Organiser
    c. musical vocabulary and meanings in the realm of electroacoustic music
    d. electroacoustic music's development
    e. the principal concepts and key figures
- Understanding of
    a. issues in electroacoustic music concerning technology and theoretical discussions (e.g. performance, notation, …)
    b. the general concepts related to the construction of electroacoustic music
- Skills in
    a. running the basic functions of the Sound Organiser
    b. being aware of the sonic environment
    c. reading a graphic representation of sound
    d. speaking about electroacoustic music
    e. hearing and identifying the main elements of musical compositions

f. being aware of listening strategies

- Attitudes of
  a. musical broadmindedness and the discrimination of quality
  b. respect for electroacoustic music as an art and a profession
  c. intention to improve one's musicianship
- Appreciation of
  a. skilled and tasteful performance
  b. good music in any medium, style or genre
- User initiatives
  a. frequent and efficient visits to EARS II
  b. searching for more about electroacoustic compositions on the Internet, libraries, radio, TV, and attending – where possible – concerts

Content: Although history is more integral to the pedagogical approach of EARS II than in EARS I, the content of the website will not be presented chronologically. Not only has the development of this music been very rapid, there have been also different styles developed at the same time. The plan is to introduce electroacoustic music by focussing on its key concepts. A concept can be for example the sonic material used in the production (e.g. using real-life sounds or generated sounds) or how it is performed (e.g. live performance or recorded on a fixed medium) or both. That said, history will not be neglected. In EARS II music history will be presented by way of a timeline. This timeline contains everything that is introduced in the website. By clicking on a date the users get links to every topic that is related to that year. So the chronology of everything that happened during this time is still present.

7.3 MDA-Based EARSII System Development Approach

In this section, the EARSII prototype system is shown as a case study to be developed following proposed MDA-based e-learning system evolution approach. CIM, PIM, PSMs, and codes are carried out to show the feasibility of the proposed method.

7.3.1 Computation Independent Model

A piece of the customers' original requirement is showed as follows, which comes from the EARS II music-learning system by Landy [43]:

"The pedagogical strategy that is being modelled is a holistic one. It works as follows: there is a three-way approach that is to be presented interdependently. It consists of a "section" concerning music appreciation ("listening"), one focusing on the understanding of musical, theoretical and technological concepts ("understanding") and another involved with music making ("doing"). The heart is the understanding section as any learner-driven navigation starts here as all key terms and concepts are embedded in this section" [43].

Based on proposed Ontology-based CIM modelling method, the CIM for above requirement can be generated following below modelling processes.

7.3.1.1 Extracting Vocabularies

Noun	**Verb**
Music appreciation.	Listen
Understanding	Make music
Concept	
Term	
Learner	

Navigation	
Doing	

Table 7-1. Extracted Requirement Vocabularies

According to Natural Language Processing (NLP) technology, requirements are extracted into vocabularies. As an example here, extraction is done manually- picked key nouns and verbs- and the result is shown as Table 7-1.

7.3.1.2 Classifying Ontologies

LTSA is the basic structure for classify the vocabularies that come from previous step. First, a RO is involved in this phase, which designed based on LTSA. Then classify vocabularies into RO to be an AO. Next task is to add extra vocabularies into AO. Finely, if there are redundancies in AO, they are reduced in this step.

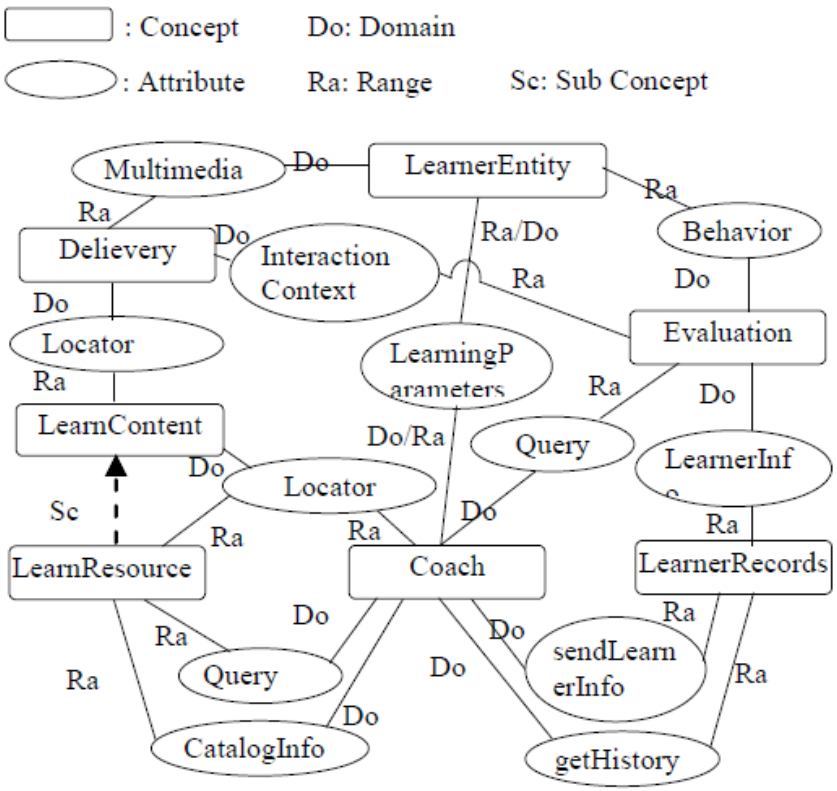


Figure 7-1. Reference Ontologies (RO) Structure

There are three steps designed to generate AO from RO: to generate AO by mapping vocabularies into RO; to add extra vocabularies into ontologies as an AO; and to reduce redundancies for AO.

Step 1: To map vocabularies into AO

Concepts [C]	**Attributes(Range) [A(Ra)]**	**SubConcept[Sc]**
Leaner==LearnerE ntity	leID ( ); lePassword( ); lrID().	
Music==LearnReso urce	lrID( ); lrContents( );	LearnContent

term, concept ==LearnContent	lcID( );
	lcBegin( );
	lcEnd( ).	

Delivery(GUI)	deID( ); locator(LearnContent);	Listening/Unde rstanding/Doin g

Table 7-2. Application Ontologies Result of Step 1

Step 2: to add extra vocabularies into ontologies as an AO

In table 7-2, Navigation is an extra vocabulary from requirement. Besides, based on LTSA, there is a potential vocabulary, LearnerRecord.

Noun: Explanation
Learner Record: to record Learner's information. **Navigation:** to navigate the learning route.

Table 7-3. Extra Vocabularies

Table 7-3 is the AO with above vocabularies.

Concepts [C]	**Attributes(Range) [A(Ra)]**	**SubConcept [Sc]**
Leaner	leID ( );	

	lePassword( ); lrID( ).	
LearnerRecords	LrdID( ); lrdContents( ).	
LearnResource	lrID( ); lrContents( );	LearnConten t
LearnContent	lcID(String);	

Delivery(GUI)	deID(String);
	locator(LearnContent);
	hasNavigation(Navigation).
		Listening/Un
		derstanding/
		Doing

Navigation		

Table 7-4. Application Ontologies Result of Step 2

Step 3: to reduce redundancies for AO

After check, there is no redundancy in Table 7-4, so it is the final AO for example requirement on EARS II.

7.3.2 Platform-Independent Model

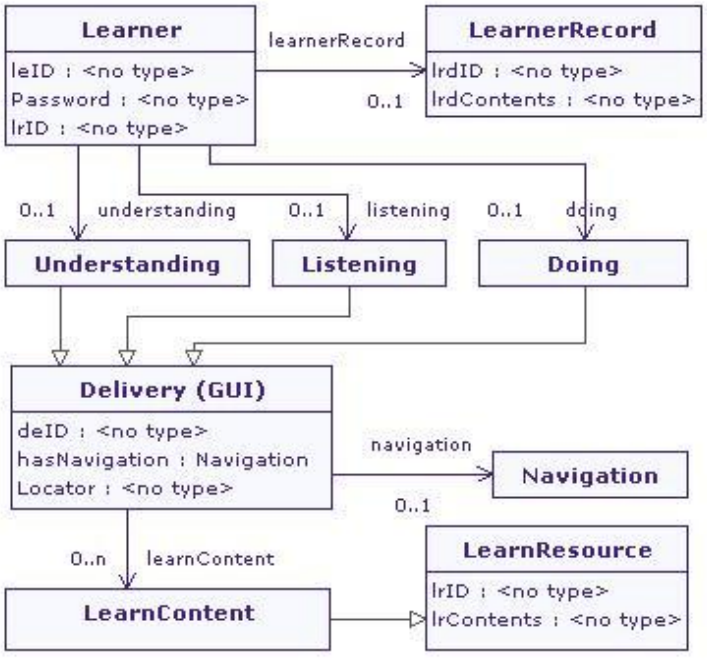


Figure 7-2. Generated PIM for EARS II

Ontology-based CIM is transformed into Platform Independent Models following a set of transformation rules that proposed in Chapter 5. Considered the PIMs are showed as a set of UML diagrams generally, following the five rules, classes are generated with name, mandatory attributes, operations, interfaces, and relationships.

Figure 7-2 shows PIM is generated properly. Under the proposed rules, classes, attributes, operations, and relationships are transformed from AO successfully.

7.3.3 Platform-Specific Model

In section 8.3.2, there is a PIM generated based on proposed ontology-based modelling approach. Therefore, in this section, this PIM will be transformed into a PSM based on proposed transformation method in chapter 7.

Based on MVC structure, PSM should be divided into three models including “View Model”, “Controller Model”, and “Model Model”.

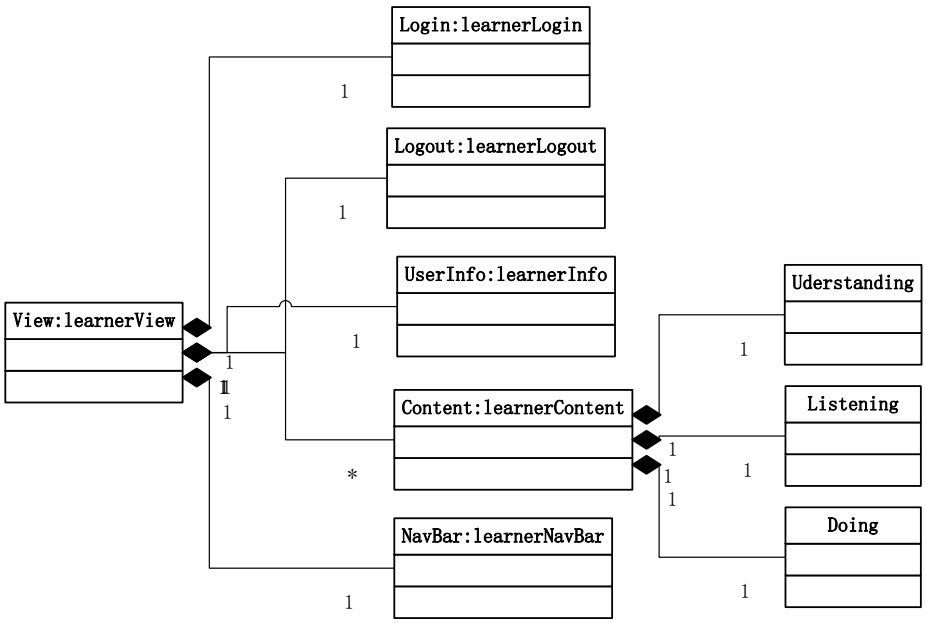


Figure 7-3. PSM View Model for EARS II

- PSM View Model: it is a model in charge of users' view. Based on the PIM generated in last section, the PSM should include "learner's view" model but without "coach's view" and "administrator's view". The view models are shown as Figure 7-3.
- PSM Controller Model: it is a model in charge of system's controller. The controller model is shown as following figure.

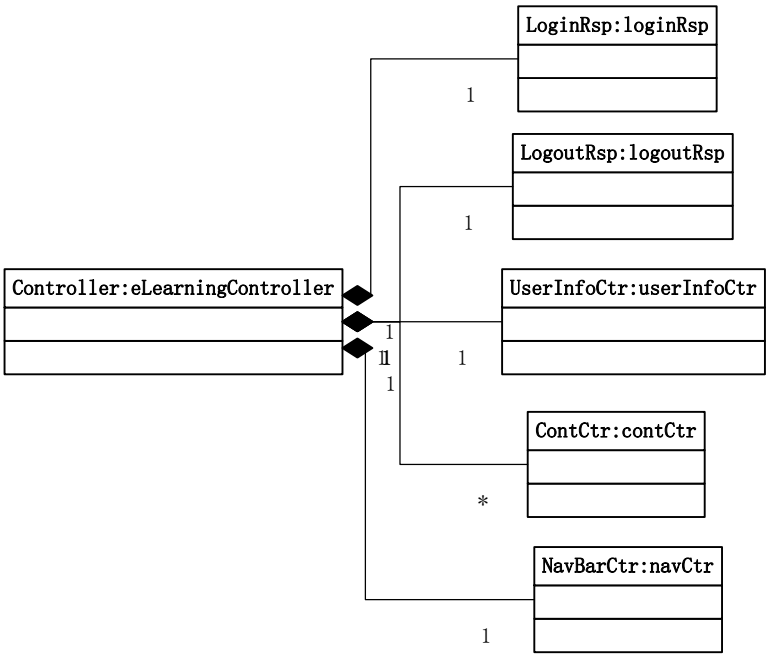


Figure 7-4. PSM Controller Model for EARS II

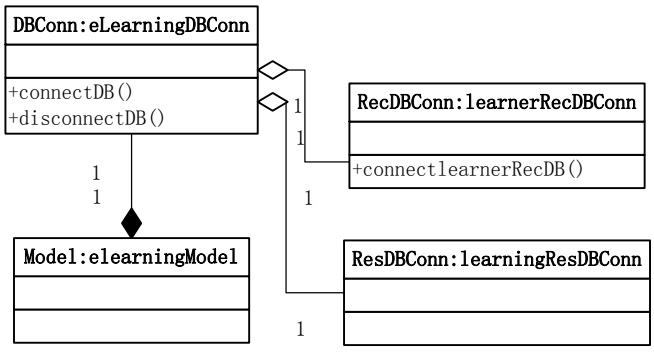


Figure 7-5. PSM Model Model for EARS II

- PSM Model's Model: it is a model in charge of system's communications with databases. The model's model is shown as following figure.

The "View Model", "Controller Model", and "Mode Model" are presented. To be a unitary PSM for EARS II, the three models are grouped together with associations which is shown as below,

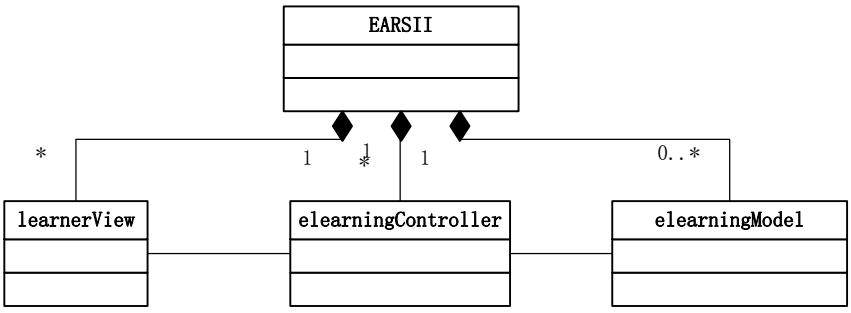


Figure 7-6. PSM Structure for EARS II

7.3.4 Code Packages

As the last phase of proposed approach, code generation is necessary to realise functions based on PSM which is presented in last section. According to structure in the PSM, there are mainly there group codes to be generated. They are "View Code", "Controller Code", and "Model Code" to corresponding "View Model", "Controller Model", and "Model Model" in PSM. Each group is defined as a package. This section is focus on showing the results with packages rather than specific code due to code's amount. Three short pieces of code are selected to give an example for each package. The full code is attached in this thesis as appendix A.

7.3.4.1 View Package

The view package includes four php files – "learnerLogin.php", "learnerLogout.php", "learnerInfo.php", and "learnerNavBar.php", and one sub-package named

"learnerContent". There are three php files in the sub-package "learnerContent" including "understanding.php", "listening.php", and "doing.php". A brief structure for this package is shown in below image with red mark.

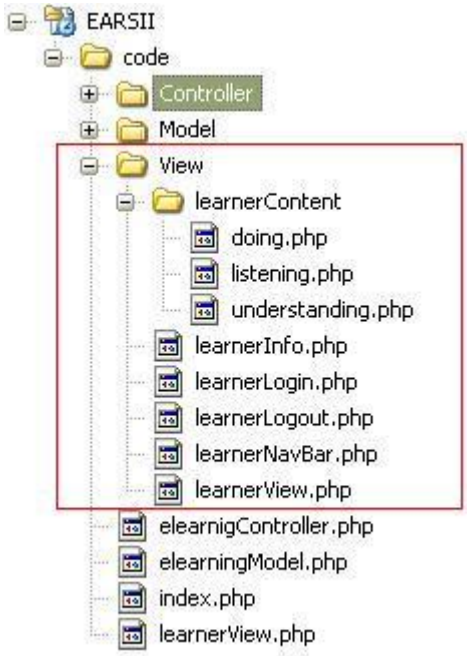


Figure 7-7. Structure of View Package

The codes in each file are shown in appendix A. Here is a piece selected as an example to show the result.

learnerLogin.php
<html> <title>EARS2:Login</title> <style type="text/css" > form { width:500px;} input { font-family:Arial; } </style> <body>

```
<div align="center"><img src="./files/images/LogoV1.jpg" alt="EARSII"
longdesc="http://www.dmuchina.com/ears2"></div>
<?php
echo “Username: <input type="text" name="username"/><br/>”;
echo “Password: <input type="password" name="password" /><br/>”;
echo “<input type="submit" name="submit" value="Submit" align="right" />”;
echo “<input type="reset" name="cancel" value="Cancel"    align="right" />”
?>    <form action="checkNewLogin.php" method="post">
</body>
</html>
```

Table 7-5. Codes in “learnerLogin.php”

7.3.4.2 Controller Package

The controller package includes five php files – “loginRsp.php”, “logoutRsp.php”, “userInfo.php”, “contentCtr.php”, and “navCtr.php”. A brief structure for this package is shown in below image with red mark.

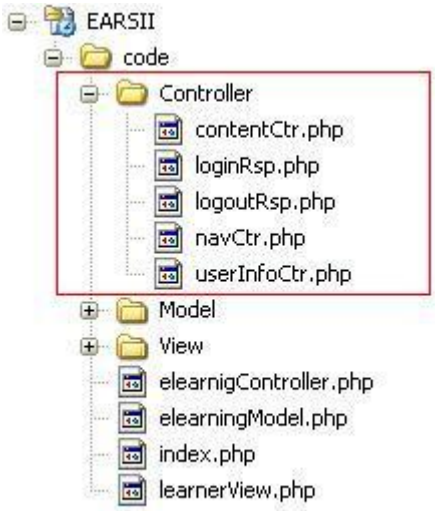


Figure 7-8. Structure of Controller Package

All codes are attached in appendix A including every file. One short file, “logoutRsp.php”, is selected to show as an example of the code result.

logoutRsp.php

<?php
function updateRecord(){
//to update user record database information
echo("<meta http-equiv=refresh content='0; url=./Model/learnerRecDBConn.php'>");
}
function showLogoutView(){
echo("<meta http-equiv=refresh content='0; url=./View/learnerLogout.php'>");}?>

Table 7-6. Codes in “logoutRsp.php”

7.3.4.3 Model Package

The controller package includes three php files – “elearningDBConn.php”, “learnerRecDBConn.php”, and “learningResDBConn.php”. A brief structure for this package is shown in below image with red mark.

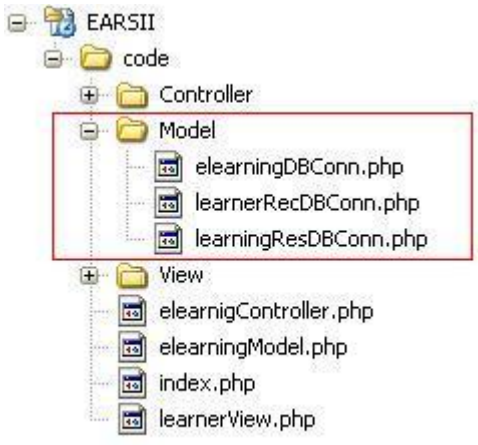


Figure 7-9. Structure of Model Package

The codes in each file are listed in appendix A. Here is a short piece selected showing as an example of the code result.

learnerRecDBConn.php
<?php $db_ears2="learnerRec"; mysql_select_db($db_ears2, $db); //$sql = "SELECT adminUsername, adminPassword FROM admin"; $sql = "SELECT * FROM admin"; $result = mysql_query($sql); ?>

Table 7-7. Codes in “learnerRecDBConn.php”

7.4 Toolkit

Presently, there are many case tools available provided by modelling vendors. However, it is not enough for the automation requirement in OMG’s MDA. On one hand, most of transformation works on PSMs to codes are realised by various tools by different mapping methods, however, it is only capable to generate template codes as the automatic transformation result generally. In this case, it is necessary to enrich and improve code by developers to accomplish system development, which is not fully achieving MDA’s automation ambition. On the other hand, the transformation from PIM to PSMs is lack of tools’ support. A large number of modelling vendors is not providing support for all the features specified by the MDA. For example, many vendors decided to skip PSM step on their tools but designed to generate codes from PIM directly. In the following content, there is a summary and catalogue for most common UML tools on market researched till December 2011, in which the issues shows clearly.

Tool (Latest Ver.)	UML Ver.	Code Enviorments	XMI	Platform	Note
AndroMDA (3.4)	2.2	J2EE/EJB, Spring, Hibernet, Struts, .NET, Web Service	YES		supporting UML2 and EMF based tools
AmaterasUML (1.3.2)	2	Java	Yes	Java	A Eclipse plug-in; Feature on analysis of Class Diagram and Sequence Diagram
Astade (0.10.1)	2	C++	No		UML to C++ Transformation Tool
Cohesion (1.0.2)	2		No	Java	Metamodel Modelling Tool. UML, OCL, and other languages can be used.
Dia (0.97)	2	C++, Java, CORBA IDL	No	Linux (Debian, Redhat), Windows	Similar with Visio. Supporting E-R diagram.
Fujaba Tool Suite (5.0.4)	2	Java	No	Java	Supporting Patterns.
miUML	2		No		Executable UML tool
NetBeans UML (6.7)	2				NetBean Plug-in UML tool
OpenAmeos (10.2)	2	Java, C, C++, Ada95	Yes	Linux, Solaris, Windows	A real-time embedded system modelling tool.
OpenArchitectureWare (4.3.1)	2		No	Java	A frame of MDA/MDD generator. Supporting tools as MagieDraw, Eclipse UML2, Enterprise Architet, etc. A part of Eclipse Modelling Project.
Open ModelSphere (3.1)	2	Java	Yes	Java	Supporting the business process modelling, data modelling, and UML modelling.
Papyrus	2		No	Java	Eclipse-based

UML (1.12)					modelling tool and code generator.
StarUML (5.0)	2	C++, Java, C#	Yes	Windows	UML/MDA platform
Topcased (5.0.0)	2.1	Java	No	Java	Eclipse's UML plug-in. Focus on critical system modelling.
Umbrello UML Modeller (2.0)	2	PHP, Ada2005, Perl	Yes	Linux/KDE	Supporting most UML diagrams, code export (C++ and Java) and reverse engineering [89].
UMLCanvas	2	HTML	No		Transform HTML5 Canvas to dynamic interactive UML diagram.
UML Graph (5.4)	2	Java	No	Java	Automated drawing of UML diagrams. It allows the declarative specification and drawing of UML class and sequence diagrams [96].
Umlify (1.2.6)	2	Ruby	No		Generate UML diagrams from Ruby code.
ArgoUML (0.32.2)	2	Java, C#	Yes	Java	A leading open source UML modelling tool and includes support for all standard UML 1.4 diagrams [8].

Table 7-8. Open Source UML Tools

Above table lists currently common UML tools as open source projects. There are also many commercial vendors supporting modelling tools. Therefore, there is also a table summarising commercial UML tools with details such as UML version, code environments, XMI available, running platform, etc.

Tool (Latest Ver.)	UML Ver.	Code Environments	XMI	Platform	Note
PowerDesigner (16.0)	2	C++, Java, C#, VB.Net, XML	No	Windows	Enterprise modelling to modelling, and data modelling combined.
Enterprise Architect (9.2)	2.4	C++, C#, Java, Delphi, SQL-DDL, VB.NET, VB	Yes	Windows, Linux	It provides full life cycle modelling for business and IT systems, software and systems engineering, and real-time and embedded development [95].
Acceleo (3.0)	2	.Net, J2EE, PHP, Python	No	Java	Integration with Eclipse and EMF.
Apollo for Eclipse (2.0)	2.1	Java	No	Java	A UML extension under Eclipse. Supporting Round-trip engineering for java5.
ARTiSAN Studio (7.2)	2.1	C++, Java, C#, Ada83, Ada95, C, SQL-DDL	Yes	Windows, Solaris	Capable to combine with PVCS, VSS, ClearCase, CM Synergy. Supporting OMG UPDM-DoDAF and MODAF.
Astah UML (6.5)	2.1	Java, C, C#	No	Java	Supporting Mind Map. A UML tool for round-trip engineering for Java.
BOUML 4.23	2	C++, Java, IDL, PHP	No	Unix/Linux/Splaris, MacOS X, Windows	Supporting plug-in on C++ and Java.
Cadifra UML Editor (1.3.2)	2		No	Windows	
CoFluent	2	C, C++	No		UML/SysML

Studio					development
EclipseUML	2.2	JavaEE	Yes	Java	A UML tool integrated Eclipse and CVS. Capable on reverse engineering from the byte code to class diagrams and sequence diagrams.
Edraw (6.1)	2		No	Windows	Visio file can be imported.
Eiffel Studio (6.8)	2	Eiffel	No	Linux, MacOS, Windows, FreeBSD	A plug-in for Visual Studio based on UML and Eiffel.
eUML2 (3.7.0)	2.1	Java	Yes	Java	A branch product of Eclipse UML tools.
Gaphor (0.14.0)	2		No	GTK	A UML tool developed by Python.
Gliffy	2				An online UML tool.
GModeler	2	AS2.0	Yes	Browser with Flash available	An online Flash UML tool.
IntelliUML Teresa (2.1.1)	1.5		Yes	Java	Tightly integrated with IntelliJ IDEA. It has not been updated since 2008.
Javelin (7.3.0.2)	2	Java	No	Windows	A graphic programming platform based on UML. Automatically maintain the class diagram and Java code synchronisation. Supporting Hibernate.
JDeveloper	2	Java	No	Java	A Java tool combined with UML.

LumiCode (3.0)	2	.NET	No		Reverse engineering. Generate sequence diagram and class diagram from .Net program.
MacA&D (7.4)	2	C++, Java, Delphi	Yes	Mac	Supporting UML modelling, structured modelling, and data modelling.
MagicDraw (17.0.1)	2.2	C++, Java, C#, IDL	Yes	Java	Integrated with Eclipse, VS2005/2008. Supporting RUP and WAE design pattern, DoDAF framework, and SysML.
MetaEdit+ (4.5)	2	Smalltalk, C++, Java, Delphi, SQL, CORBA, IDL	No	Linux, Windows	A Domain-Specific Modelling (DSM) tool.
Metamill (6.0)	2.3	C++, Java, C#	Yes	Linux, Windows	Supporting multi-users modelling.
Modelio (1.2.2)	2	C#, Java	Yes		
Modelmaker (11.2.0)	2	Delphi 4 to Delphi XE, C# to VS 2003, VS 2005, VS 2008, and VS 2010.	No	Windows	A Delphi and C# Visual modelling and Refactoring tool based on UM 2 technology [64].
Objecteering (6.0)	2	Java, C++, C#, IDL, SQL, Oracle	Yes	Windows, Linux, Solaris	It provides a dedicated graphical modelling tool to help you easily develop the tools you need when implementing a model-driven MDA approach in your projects [68].

ObjectiF (5.0)	2	Visual C++, JBuilder, Visual Cafe, IDL, SQL, VB	Yes	Windows	Combined with VS.NET and Eclipse.
Rhapsody (7.6)	2.1	IDL, Java, C++, Ada, C	Yes	Linux, Windows, Solaris	It focuses on MDD for real-time embedded systems.
Rational Software Architect (8.0.2)	2.1	Java, C++, VB, ADa, IDL, Delphi, SQL, Oracle	Yes	Windows, Linux, Unix	An IBM product.
Together R3	2	CORBA, IDL, C++, Java, C#, COM IDL, EJB, VS.NET, SAP, WebSphere	Yes	Java	Supporting DSL, OCL 2.0, and PEL4WS.
Visio 2010	2	IDL, C++, C#, VB	Yes	Windows	A Microsoft diagram tool supporting UML.

Table 7-9. Commercial UML Tools

From above tables, there are a large amount UML tools on the market. Functions and aiming systems are different showing in the details specification. Considering MDA specification, there is not much tools supporting it. To sum up, there are following tools are related to MDA paradigm closely,

- AndroMDA 3.4 [7]: is an open source code generation framework that follows the Model Driven Architecture paradigm. It takes models from CASE-tools and generates fully deployable applications and other components.
- Objecteering 6.0: is a model-driven development tool. Objecteering MDA Modeller provides a dedicated graphical modelling tool to help you easily develop the tools you need when implementing a model-driven MDA approach in your projects [68].

- ObjectiF 5.0: is a UML-Tool for Model-Driven Development in C#, C++ and Java with model transformations for standard .NET and Java technologies, and integrated technology for developing specific model transformations [60, 61].
- OpenArchitectureWare (oAW) 4.3.1: is a part of the Eclipse Modelling Project as a MDA/MDD generator framework implemented in Java(TM) [84].
- Select Solution for MDA: is a ground breaking modelling and transformation tool designed to generate, reverse engineer and synchronise all your model viewpoints and your code, based upon UML designs within Select Architect [91].
- Together R3: is a modelling platform that gives enterprise teams leading-edge design capabilities which enable the visualisation and continued maintenance of IT architectures [22].
- StarUML: is an open source project to develop fast, flexible, extensible, featureful, and freely-available UML/MDA platform aiming to build a software modelling tool and also platform that is a compelling replacement of commercial UML tools such as Rational Rose, Together and so on. [98].

Overall, there are many tools can be applied on the proposed MDA-based evolution approach especially on code generation step. The available tools might not as strong as expected; however, they are supports on current research stage and can be valuable references for the future improvement on transformation.

7.5 Prototype Screen Shot

There are few screen shots showing the learner-oriented interfaces in EARSII prototype system.

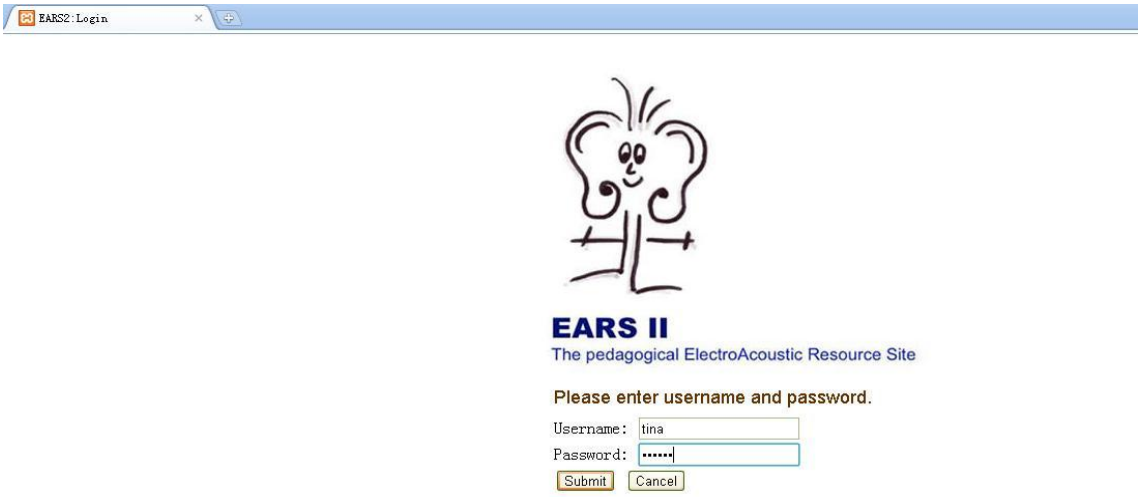


Figure 7-10. Learner Login Interface in EARSII Prototype

Figure 7-10 is leaner login page. Home page is next page shown to learner after successfully login which is shown as Figure 7-11. Navigation in EARSII is shown as Figure 7-12, Figure 7-13, and Figure 7-14.

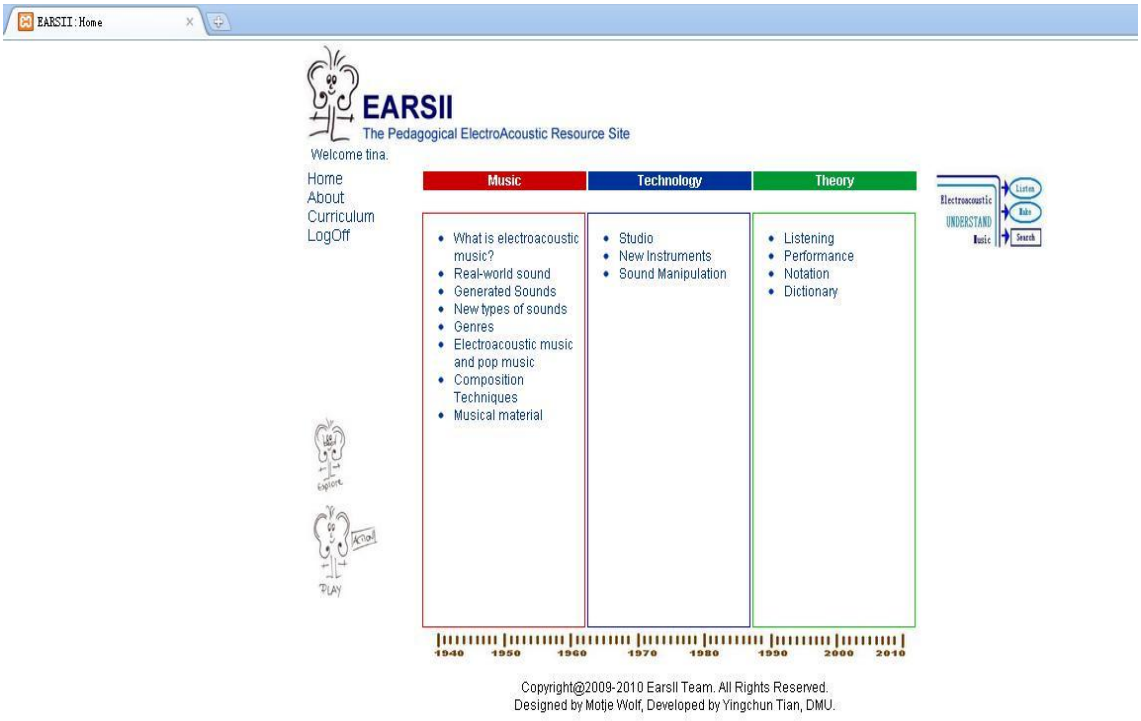


Figure 7-11. Home Page in EARSII Prototype

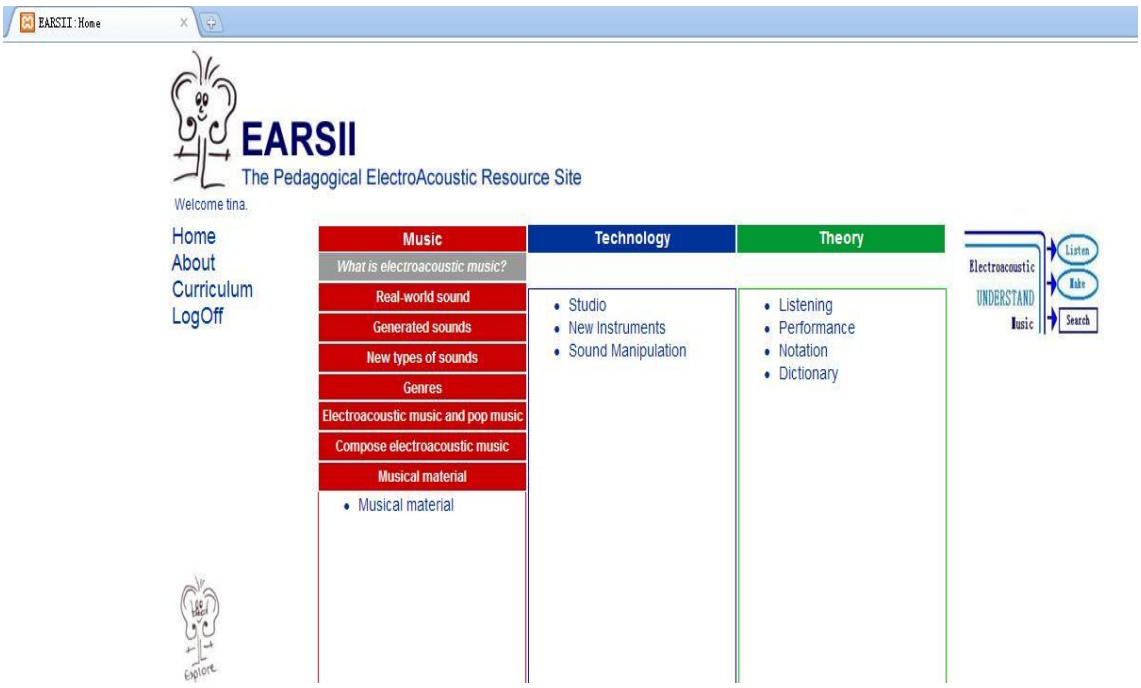


Figure 7-12. Navigation Interfaces in EARSII Prototype 1

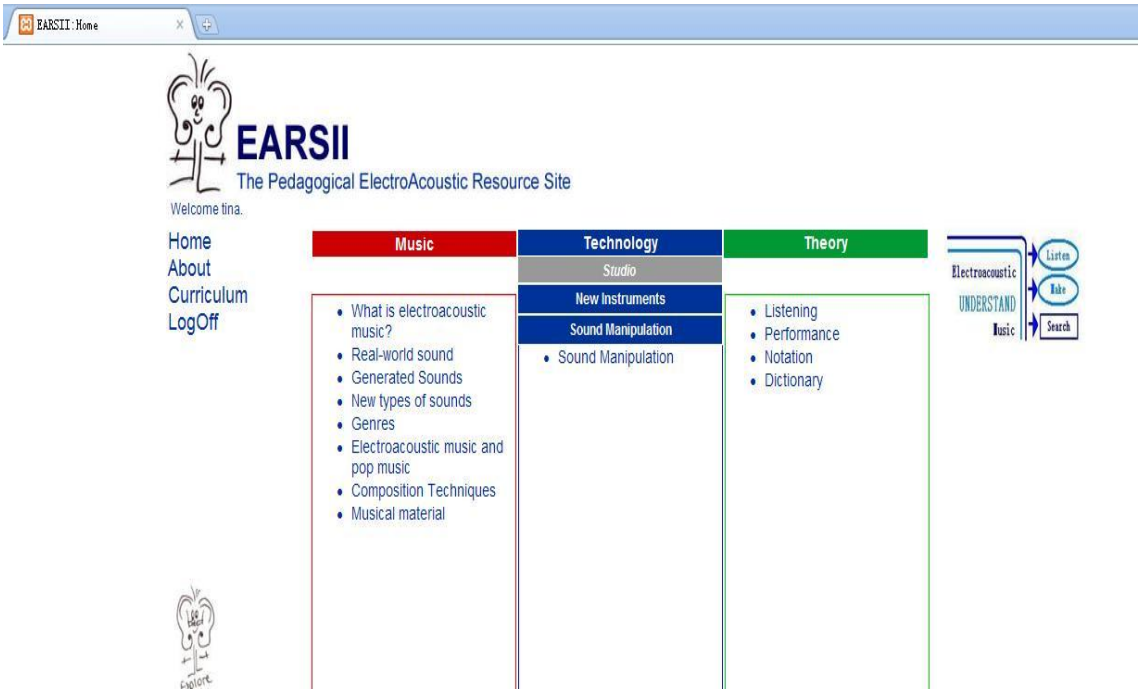


Figure 7-13. Navigation Interfaces in EARSII Prototype 2

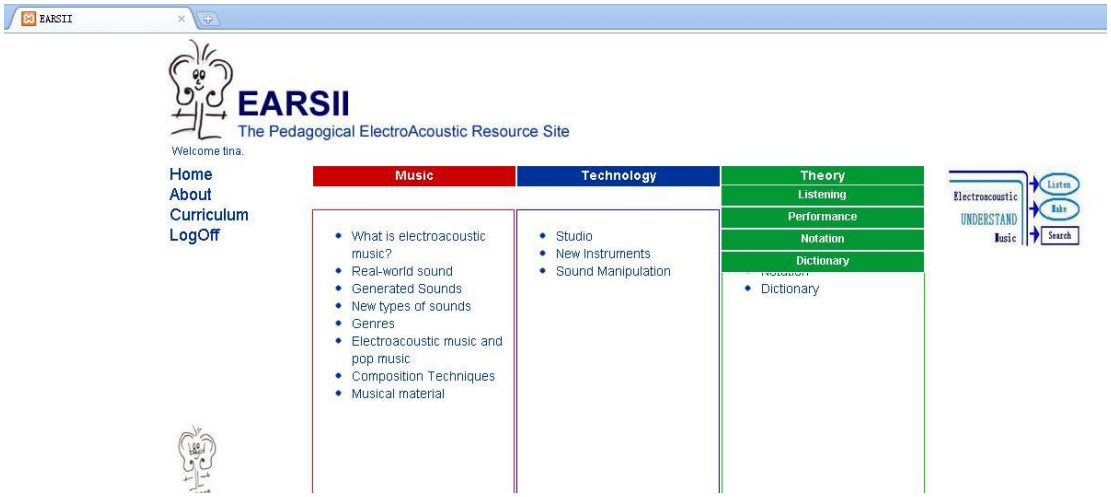


Figure 7-14. Navigation Interfaces in EARSII Prototype 3

Figure 7-15 is a content page with sound player. There is a content page with picture showing as Figure 7-16.

Screen shots showing here are samples pages. Content's representation can be multimedia. There is possible to add video, audio, image, application, etc into this system to enhance content pages.

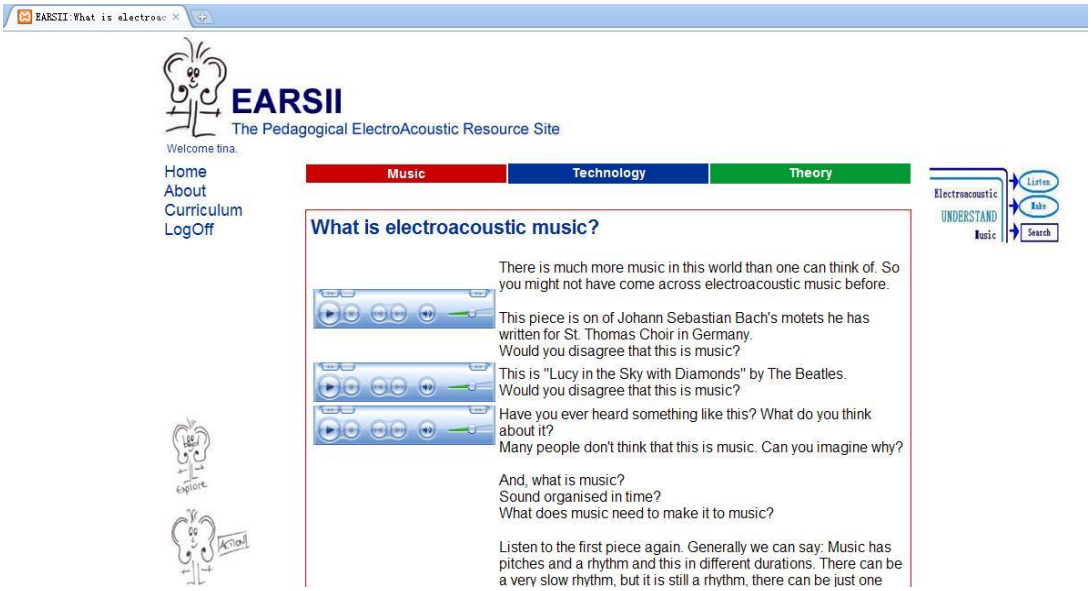


Figure 7-15. Content Page with Sound Player

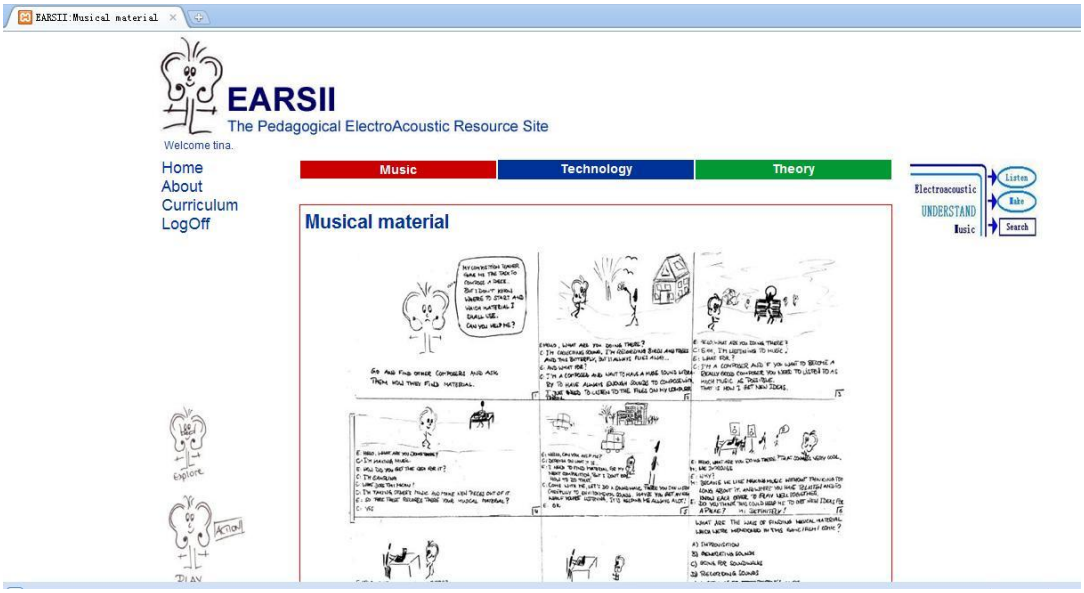


Figure 7-16. Content Page with Picture

Interfaces for coach are showing as Figure 7-17, Figure 7-18, and Figure 7-19.

- Coach login page: username and password are required to login.
- Home page: it shows navigation tree in the left column. Right column is an area for main contents. Initially it is designed to display notices or general information for system.
- Edit page: it provides spaces for coach to edit learning contents.

Figure 7-17. Coach Login Interface in EARSII Prototype

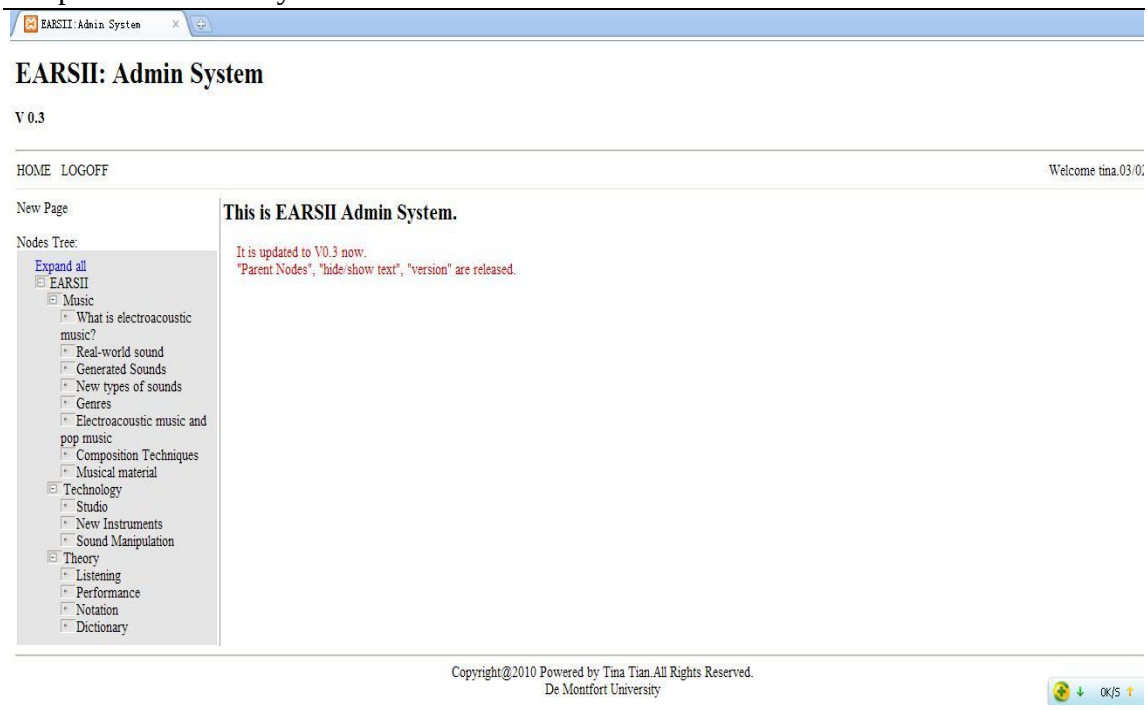


Figure 7-18. Coach Home Page in EARSII Admin System

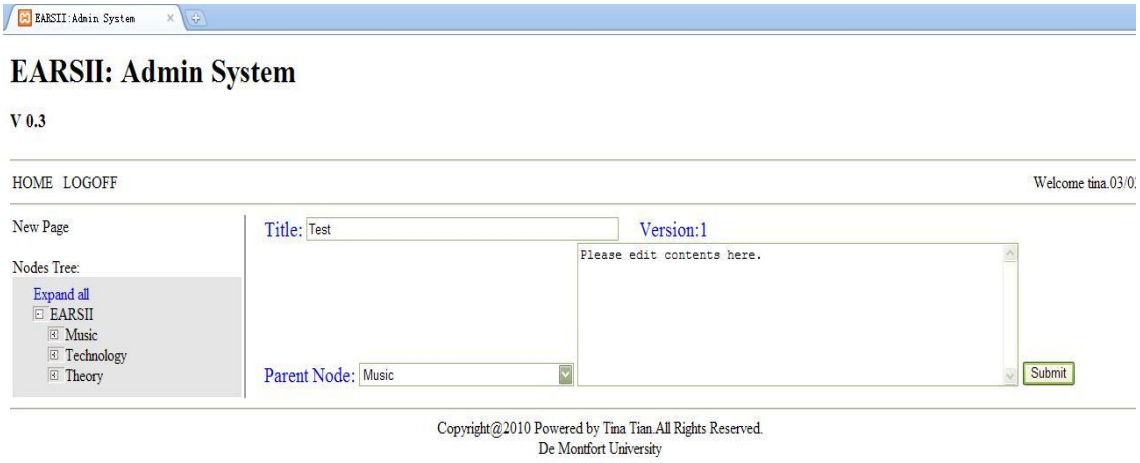


Figure 7-19. Coach Edit Page in EARSII Admin System

7.6 Summary

In this chapter, the proposed method is applied in a project named ElectroAcoustic Resource Site II as a case study.

In the beginning, Background and issues of EARSII are introduced. Then, it is a development approach in which proposed MDA-based method is applied. Proposed development phases are demonstrated from CIM to PHP code. Additionally, there are research on toolkit to explain UML tools, MDA tools, and current situations. Finally, there are prototype screen shots to prove approach's feasibility.

Chapter 8
Conclusions

Objectives

- To summarise the thesis and draw conclusions
- To revisit original contributions
- To evaluate the research by answering the research questions and revisiting the success criteria
- To illustrate the limitations of the work
- To propose future work

8.1 Summary of Thesis

This thesis aims to improve the software evolution methods in e-learning domain by proposing a Model-Driven Architecture based evolution approach via domain modelling and transformation technique. The basic idea is to employ MDA standard to lead software lifecycle, and as a result, create a semi-automated process to enhance models' role in software development and maintenance and to create more space for designers and developers on creative development.

The research described in this thesis is postulated in the context of model-driven engineering and e-learning domain. UML and UML profile are selected to support modelling presentation of PIM, PSMs, and their metamodels. The proposed research can be divided into four main stages, namely, e-leaning modelling, MDA-based evolution method, transformation, and approach's application. The e-learning

modelling stage is supported by Learning Technology Systems Architecture standard and Model-View-Controller design pattern. The evolution method stage is based on OMG's Model-Driven Architecture standard and supported by ontology knowledge. The transformation stage is implemented by UML and UML profile based MOF mapping algorithm. The approach's application stage deploys the proposed approach into an e-learning system. In order to guarantee that the proposed research is systematic and well-structured, reference has been made to software taxonomy. The main research subjects in this study are Model-Driven Architecture and e-learning domain system. A project named ElectroAcoustic Resource Site II is performed as use case to validate the proposed approach. The research of toolkit is presented to support and facilitate use case.

8.2 Revisiting Original Contributions

This thesis proposes Model-Driven Architecture based evolution method to e-learning system, as observed in Chapter 1. An e-learning domain modelling is proposed in Chapter 4. Proposed approach to Model-Driven Architecture based evolution is specified in Chapter 5. Meanwhile, in Chapter 3, Software Engineering Creative Computing is proposed as a new concept. Furthermore, the kernel contribution is a set of predefined transformation rules and its application in electronic learning system which is shown in Chapter 7.

This section will revisit and extend the seven expected original contributions presented in Chapter 1 as follows:

- C1: In Chapter 5, a Model-Driven Architecture based evolution method and approach has been proposed. The method is specified individually in CIM, PIM, PSM, and code development.

- C2: In Chapter 5, ontology technique has been applied into PIM generation process which is divided into two phases: vocabularies extraction and PIM generation.

- C3: In Chapter 6, a PIM to PSM transformation has been designed with methodology and processes.

- C4: In Chapter 6, models in layers have been defined based on UML and UML profile. There are source model, metamodel for source model, target model, metamodel for target model, transformation model, and metamodel for transformation model.

- C5: In Chapter 6, a set of mapping rules have been clarified with formulas and constraints for transformation model and metamodel for transformation model.

- C6: In Chapter 4, E-learning system has been modelled as domain framework. LTSA standard and pedagogical strategy has been applied into modelling to support e-learning system. As a result, three models have been designed: “step by step learning model”, “optional learning model”, and “interacting learning model”.

- C7: In Chapter 4 and 5, Model-View-Controller design pattern has been applied into e-learning domain models and MDA-based evolution approach.

- C8: In Chapter 3, Software Engineering Creative Computing has been proposed as a new concept. Four examples have been presented: user interface design - the golden rules, data design principles, metrics for object-oriented design class, and framework for web engineering. Besides, an application in e-learning system has been implemented.

- C9: In Chapter 7, an application in an e-learning system has been deployed. A prototype of project ElectroAcoustic Resource Site II is the result.

- C10: In Chapter 7, a set of tools is summarised and analysed to support the proposed MDA-based evolution method for e-learning system.

8.3 Evaluation

8.3.1 Answering Research Questions

The evaluation of this study starts by answering the proposed research questions. The global research question presented in Chapter 1 was:

How can software systems, especially electronic learning system, be developed maximum automatically by Model-Driven Architecture theory?

This question has been answered by proposing a Model-Driven Architecture based evolution method and process. The ontology-based PIM generation approach supports an automatic process for CIM and PIM phases. E-learning modelling is supplying reference models and basic architecture for e-learning system, specifically PIM and PSMs. Transformation definitions are employed to mapping PIM to PSMs. Moreover, Software Engineering Creative Computing provides an idea to enhance the whole process.

A set of research questions was defined subsequently to refine this global question in detail.

RQ1: What is the proper development approach to achieve the requirement of Model-Driven Architecture standard?

Based on Model-Driven Architecture standard, there are three group models are required including Computation Independent Model, Platform-Independent Model, and Platform-Specific Model. A MDA based life cycle is proposed showing the recommended development approach. (Section 2.3)

- *Which models are necessary based on Model-Driven Architecture standard?*

Platform-Independent Model and Platform-Specific Model are two group models necessarily required. Computation Independent Model is optional but highly recommended to be realised in Model-Driven Architecture standard. (Section 2.3)

- *How to represent those models to meet standard's requirements?*

There are different ways to describe models including UML, UML profile, XML, XMI, etc. In this thesis, UML and UML profile are chosen to represent models. (Section 3.3 and 3.4)

- *How to organise models' structure following Model-Driven Architecture standard?*

Models are categorised into groups including Computation Independent Model, Platform-Independent Model, and Platform-Specific Model. The three groups are organised in layers based on layered architecture for MDA. (Section 2.3)

RQ2: How to realise transformations for Model-Driven Architecture based electronic learning system development approach?

QVT is the basic standard in transformation definition. Transformation architecture is proposed to abstract the realisation. (Section 7.2)

- *What is transformation method in the approach?*

The transformation method contains QVT-based transformation definition and e-learning domain modelling based transformation architecture. (Section 7.2)

- *Which phases need transformations?*

In the proposed MDA-based approach, transformation is required in PIM to PSMs phase. (Section 5.1.3)

- *Which mapping rules should be followed in each transformation step?*

Mapping rules are defined with well-formed formulas and constraints supporting models in two layers: transformation model and metamodel for transformation model. (Section 7.5)

RQ3: What is the proper way to modelling electronic learning system?

LTSA standard and pedagogical strategy has been applied to support electronic learning domain modelling. Model-View-Controller has been employed as basic design pattern. MDA standard should be followed strictly with CIM, PIM, and PSM phases. (Section 4.1 and 4.2)

- *What are basic functions required in electronic learning domain modelling?*

With standard and strategy supporting, functions are selected and summarised for different learning models. There are three learning models proposed in this thesis including “step by step learning model”, “optional learning model”, and “Interacting learning model”. Each learning model contains a set of functions fitting its features. (Section 4.1)

- *Which standards, strategies, or patterns should be involved to support domain modelling for electronic learning?*

Based on this study, there are LTSA standard, pedagogical strategy, and Model-View-Controller pattern chosen to support domain modelling for electronic learning in this thesis. (Section 4.1 and 4.2)

- *How to organise domain models to meet requirements on Model-Driven Architecture based development approach?*

To meet Model-Driven Architecture requirement, electronic learning domain models are organised into a four-layered structure with CIM, PIM, and PSM expression. (Section 4.2)

RQ4: What is the Model-Driven Architecture based development process for electronic learning system?

The development process is based on electronic learning domain modelling and Model-Driven Architecture standard. A Model-Driven Architecture based evolution method is proposed as kernel theory. (Section 5.1, Section 5.2, and Section 6)

- *Which phases are necessary in this development process?*

Regarding Model-Driven Architecture standard, the kernel phases in this development process are CIM creation, transformations from CIM to PIM, mapping PIM to PSMs, and code generations. (Section 5.1)

- *What elements should be presented in each phase?*

Specific elements on CIM, PIM and PSM are defined and represented through modelling. (Section 5.2 and Section 6.2)

- *How to get each model's elements automatically?*

Transformation is the kernel method to generate each model's elements automatically. Related methodology and notations are presented. Specific mapping rules are defined with well-formatted formulas and explanations. (Section 6.3, Section 6.4 and Section 6.5)

- *How to define and represent the relationships between models?*

Proposed MDA-based modelling structure provides relationships between models in levels. Specific relationships between PSMs are provided based on transformation method. (Section 4.2.2 and Section 6.5)

RQ5: What kind of tools is capable to support the proposed Model-Driven Architecture based approach?

There is a toolkit summarisation and analysis including two categories: general UML tools and specific MDA-related tools with their capability of supporting the proposed MDA-based evolution approach. (Section 7.4)

8.3.2 Revisiting the Measure of Success

In Chapter 1, a set of measures are defined to validate the success of the proposed research described in this thesis. This section will revisit the predefined measure of success.

- *How to develop electronic learning system following Model-Driven Architecture standard?*

The development process is based on electronic learning domain modelling and Model-Driven Architecture standard. A Model-Driven Architecture based evolution method is proposed as kernel theory.

- *How many standards or theories are supporting electronic learning domain modelling?*

In this thesis, there are LTSA standard, pedagogical strategy, and Model-View-Controller pattern chosen to support domain modelling for electronic learning.

- *How many phases are automatic or semi-automatic in proposed development process?*

By proposed Model-Driven Architecture based evolution method, every phase is automatic or semi-automatic in proposed development process because models are generated by transformation tool or rules, which is including CIM establish, PIM generation, PIM to PSM transformation, and code generation.

- *How to realise transformation for models in different levels?*

Models are generated in steps following Model-Driven Architecture and resulted as CIM, PIM, and PSMs. Transformations are based on UML profile and mapping rules. Therefore, both models and transformations are complied with MDA standard.

- *How many rules are supporting Platform-Specific Model generation?*

Proposed mapping rules are working for models in two levels: transformation model and metamodel for transformation model. They are supporting PSMs' generation from metamodel layer to model layer. There are totally 39 kernel rules categorised into seven groups in metamodel level. Meanwhile, 7 kernel rules are defined in transformation model level. Each level's rules' amount is capable to be extended following kernel rules. Besides, because mapping rules are specified as formulas and constrains, they are capable to be programmed as a tool.

- *How many electronic learning systems are suitable to be developed based on proposed method?*

The proposed Model-Driven Architecture based evolution approach is able to be applied to a use case - ElectroAcoustic Resource Site II project, which is an existing project including every basic function in e-learning system.

M2: The models and their transformations in different levels should be exactly complied with OMG's MDA standards.

Models are generated in steps following Model-Driven Architecture and resulted as CIM, PIM, and PSMs. Transformations are based on UML profile and mapping rules. Therefore, both models and transformations are complied with MDA standard.

M3: Designed transformation rules should support PIM to PSM properly, meanwhile, they should be capable to be realised as a mapping tool.

Predefined transformation notions are represented based on studies of related projects. A set of transformation rules is designed with well-formatted formulas and explanations. Therefore, it is capable to be realised as a concrete tool or a set of tools.

M4: The proposed MVC structure should be support the general e-learning system based on education pedagogical knowledge.

Three learning models have been designed based on pedagogical strategy for general e-learning system. MVC modelling is presented based on the three models. Moreover, models presented in Chapter 6 are showing application of MVC pattern.

8.4 Limitations

Having discussed the original contributions and success criteria, the proposed research described in this thesis also has following limitations:

The PIM to PSM transformation is currently presenting as models and rules but not a finished transformation tool.

The fundamental elements of proposed PIM to PSM transformation method are transformation models and rules. Models are presented by UML and UML profiles. Rules are specified as formulas and constrains. At the current stage, they are working as standards to be followed manually. A transformation tool is not developed to realise automatic PSM generation.

Proposed MDA-based evolution approach supports process for forward engineering but not reverse engineering.

Development approach is well presented in proposed MDA-based evolution method including phases: CIM, PIM, PSMs, code, and maintenance. It is capable to be applied into forward engineering. Reverse engineering is not concerned in this study but left as further work to be an improvement of this method.

8.5 Further Work

Based on the discussions regarding research questions, research propositions, original contributions, success criteria, and limitations in the previous sections, the conclusions can be drawn. The Model-Driven Architecture based evolution method and its application in an e-learning system, described in this thesis, is a novel, systematic, and practical methodology for software engineering in e-learning domain. The prototype of use case and researches on toolkits have supported and verified the success of the

approach. The fact that human interventions are still required indicates that it is only semi-automatic process. However, given the fact the manual efforts' capability to be programmed as tools, this semi-automatic approach will improve the software engineering process in e-learning domain greatly.

The research presented in this thesis is not the terminus. The following further work can be suggested to be pursued based on the present work.

- Based on transformation method proposed in Chapter 6, the PIM to PSM transformation can be programmed and realised as a tool or a toolkit to achieve maximum automatic.
- Based on proposed e-learning modelling in Chapter 4, more specific functions can be considered to enhance e-learning domain models.
- Proposed MDA-based evolution approach can be improved to support reverse engineering. Mapping rules can be enhanced for reverse engineering. Because models are the kernel elements in this method, it is capable to realise support on reverse engineering without changes in major processes.

References

[1] ISO/IEC 14764:2006: Software Engineering - Software Life Cycle Processes - Maintenance. 2006, pp.1-44.

[2] *Unified Modelling Language from FOLDOC* [Online]. Available: http://foldoc.org/UML, 2005, pp.1-20.

[3] *W3schools.com,* Avaliable: http://www.w3schools.com/w3c/default.asp, 2011, pp. 12-59.

[4] *Model Driven Architecture, MDA Guide 1.0.1*, OMG Standard, 2003, pp. 5-40.

[5] Y. Akpinar, V. Bal, and H. Simsek, "An E-Learning Content Development System on The Web: BU-LCMS," in *Information Technology Based Higher Education and Training, 2004 (ITHET 2004). Proceedings of the FIfth International Conference on*, 2004, pp. 239-243.

[6] S. S. Alhir, *A Guide to Successfully Applying the Uml*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2002, pp. 24-63.

[7] AndroMDA.org. *Generate Components Quickly with AndroMDA* [Online], 2011, pp.1-9, Avaliable: http://www.andromda.org.

[8] *ArgoUML-Overview* [Online], 2008, pp. 1-3, http://argouml.tigris.org/.

[9] M. Asadi, M. Ravakhah, and R. Ramsin, "An MDA-Based System Development Lifecycle," in *Second Asia International Conference on Modeling and Simulation, AICMS 08*, 2008, pp. 836-842.

[10] S. Atkinson and L. Landy, "The ElectroAcoustic Resource Site (EARS): Philosophy, Foundation and Aspirations," *Organised Sound* vol. 9, 2004, pp. 79-85.

[11] D. Batory, "Multilevel Models in Model-Driven Engineering, Product Lines, and Metaprogramming," *IBM Systems Journal*, vol. 45, July 2006, pp. 527-539.

[12] E. Bersoff, V. Henderson, and S. Siegel, "Software Configuration Management," in *Proceedings of The Software Quality Assurance Workshop on Functional and Performance Issues*, 1978, pp. 9-17.

[13] F. Chen, "*Model Driven Software Modernisation*," PhD Thesis, Software Technology Research Laboratory, De Montfort University, Leicester, 2007, pp. 202-287.

[14] F. Chen, B. Qiao, H. Yang, and W. C. Chu, "A Formal Model Driven Approach to Dependable Software Evolution," in *30th Annual International Computer Software and Application Conference (COMPSAC'06)*, Chicago, 2006, pp. 205-212.

[15] J. Chen and Q. Lu, "A Complex Adaptive E-Learning Model Based on Semantic Web Services," in *Knowledge Acquisition and Modeling, 2008. KAM '08. International Symposium on*, 2008, pp. 555-559.

[16] C. Christian, "Pedagogical Pattern Selection Strategies," *Neural Networks*, vol. 7, 1994, pp. 175-181.

[17] D. Deridder, "A Concept-Oriented Approach to Support Software Maintenance and Reuse activities," presented at *the Workshop on Knowledge-Based Object-Oriented Software Engineering at 16th European Conference on Object-Oriented Programming (ECOOP 2002)*, Málaga,Spain, 2002, pp. 40-49.

[18] M. G. B. Dias, N. Anquetil, and e. al, "Organizing the Knowledge Used in Software Maintenance," *Journal of Universal Computer Science*, vol. 9, 2003, pp. 641-658.

[19] S. Downes. *E-learning 2.0*, 2011. Avaliable: http://www.downes.ca/post/31741.

[20] J. D. Edwards and M. A. Preston, "Forces in Screen-Secondary Linear Reluctance Motors," *IEEE Transactions on Magnetics*, vol. 24, 1988, pp. 2913-2915.

[21] J. O. Entzinger, K. Morimura, and S. Suzuki, "Developing E-Learning Content to Raise Global Awareness in a Seminar Style Course," in *2011 IEEE International Professional Communication Conference (IPCC)*, 2011, pp. 1-8.

[22] M. Focus. *Together-Visual Modeling for Software Architecture Design* [Online], Avaliable: http://www.borland.com/us/products/together/index.aspx. [Access: 09/01/2012]

[23] D. S. Frankel, *Model Driven Architecture: Applying MDA to Enterprise Computing*, 1st ed. New York, NY, USA.: John Wiley & Sons, Inc., 2002, pp. 98-165.

[24] D. S. Frankel, *Model Driven Architecture: Applying MDA to Enterprise Computing*, 3rd ed. New York, USA: John Wiley & Sons, Inc., 2007, pp. 21-60.

[25] C. Gang, "Improving System Usability Model in E-Learning Based on System-Reasonable Theory," in *International Conference on E-Learning, E-Business, Enterprise Information Systems, and E-Government, (EEEE '09)*, 2009, pp. 72-75.

[26] D. Gasevic, D. Djuric, and V. Devedzic, *Model Driven Engineering and Ontology Development*: New York, USA: Springer Publishing, Inc., 2009, pp. 13-79.

[27] A. Gavras, M. Belaunde, L. F. Pires, and J. P. A. Almeida, "Towards an MDA-Based Development Methodology," in *Lecture Notes in Computer Science: Software Architecture*. vol. 3027, 2nd ed. Berlin: Springer, 2004, pp. 230-240.

[28] K. C. Graber, "The Influence of Teacher Education Programs on the Beliefs of Student Teachers: General Pedagogical Knowledge, Pedagogical Content Knowledge, and Teacher Education Course Work.," *Journal of Teaching in Physical Education*, vol. 14, Jan. 1995, pp. 157-178.

[29] P. Grew, F. Giudici, and E. Pagani, "Specification of a Functional Architecture for E-Learning Supported by Wireless Technologies," in *Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, 2006. (PerCom Workshops'06)*, 2006, pp. 215 -220.

[30] T. R. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," *International Journal of Human-Computer Studies - Special Issue: The Role of Formal Ontology in The Information Technology*, vol. 43, issue 5-6, Duluth, MN, USA: Academic Press, Inc., 1995, pp. 907-928.

[31] N. Guarino, "Understanding, Building and Using Ontologies," *International Journal of Human-Computer Studies*, vol. 46, issue 2-3, Duluth, MN, USA: Academic Press, Inc., 1997, pp. 293-310.

[32] B. Haslhofer. *Meta Object Facility* [Online]. 2008, pp. 12-54. Available: http://metadaten-twr.org/2008/09/22/mof/#more-3

[33] J. Holt, *UML for Systems Engineering: Watching the Wheels IET*, 2nd ed. Philadephia, PA: Institution of Engineering and Technology, 2004, pp. 201-208.

[34] IBM. *IBM Software Development Library: Industry Standards*. 2008, pp. 98-107.

[35] *Draft Standard for Learning Technology — Learning Technology Systems Architecture (LTSA)* [Online], IEEE Standard, Available: http://ieeeltsc.org. 2007, pp. 30-59.

[36] D. Jaffee, "Asynchronous Learning: Technology and Pedagogical Strategy in a Distance Learning Course," *Teaching Sociology*, vol. 25, Oct. 1997, pp. 262-277.

[37] K.H.Benett and V. T. Rajlich, "Software Maintenance and Evolution: a Roadmap," presented at *the The Future of Software Engineering*, 2000, pp. 31-43.

[38] J. S. Kim and Y. G. Park, "Mapping Method of SCORM Content Aggregation Model for E-Learning Content Design," in *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*, 2009, pp. 2010-2015.

[39] B. A. Kitchenham, G. H. Travassos, A. v. Mayrhauser, F. Niessink, N. F. Schneidewind, J. Singer, S. Takada, R. Vehvilainen, and H. Yang, "Towards an Ontology of Software Maintenance," *Journal of Software Maintenance*, vol. 11, 1999, pp. 365-389.

[40] A. Kleppe, J. Warmer, and W. Bast, *MDA Explained: the Model Driven Architecture: Practice and Promise*: Pearson Education, Inc., 2003, pp. 29-100.

[41] A. G. Kleppe, J. Warmer, and W. Bast, *MDA Explained: The Model Driven Architecture: Practice and Promise*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003, pp. 52-59.

[42] P. Kumar, S. G. Samaddar, A. B. Samaddar, and A. K. Misra, "Extending IEEE LTSA E-Learning Framework in Secured SOA Environment," in *Education Technology and Computer (ICETC), 2nd International Conference on 2010*, pp. V2-136-V2-140.

[43] L. Landy, "The ElectroAcoustic Resource Site (EARS)," *Journal of Music, Technology and Education*, vol. 1, 2007, pp. 69-81.

[44] L. Landy, "The ElectroAcoustic Resource Site (EARS) Approaches Its Next Phase: Going Global and Addressing the Young," in *International Computer Music Conference 2007 Proceedings*, Copenhagen, 2007, pp. 141-144.

[45] M. M. Lehman, "Laws of Software Evolution Revisited," in *Proceedings of the 5th European Workshop on Software Process Technology (EWSPT'96)*, London, UK: Springer-Verlag, 1996, pp. 108-124.

[46] M. M. Lehman, "Programs, Life Cycles, and Laws of Software Evolution," *Proceedings of the IEEE*, vol. 68, issue 9, 1980, pp. 1060-1076.

[47] M. M. Lehman, D. E. Perry, and J. F. Ramil, "On Evidence Supporting the FEAST Hypothesis and the Laws of Software Evolution," in *Software Metrics Symposium. Metrics 1998. Proceedings. Fifth International*, 1998, pp. 84-88.

[48] M. M. Lehman and J. F. Ramil, "Software Evolution and Software Evolution Processes," *Annals of Software Engineering*, vol. 14, 2002, pp. 275-309.

[49] M. M. Lehman and J. F. Ramil, "Software Evolution in the Age of Component-Based Software Engineering," *Software, IEE Proceedings*, vol. 147, 2000, pp. 249-255.

[50] M. W. Lewis and G. E. Dehler, "Learning through Paradox: A Pedagogical Strategy for Exploring Contradictions and Complexity," *Journal of Management Education*, vol. 24, Dec. 2000, pp. 708-725.

[51] B. P. Lientz and B. E. Swanson, *Software Maintenance Management: A Study of the Maintenance of Computer Application Software in 487 Data Processing Organizations*. Boston, Ma, USA: Addison-Wesley Longman Publishing Co., Inc, 1980, pp. 12-32.

[52] M. M. Lehman and J. F. Ramil, "Towards a Theory of Software Evolution And its Practical Impact," in *Proceeding of Principles of Software Evolution, International Symposium on*, 2000, pp. 2-11.

[53] M. M. Lehman and L. A. Belady, *Program Evolution: Processes of Software Change*. San Diego, CA, USA: Academic Press Professional, Inc, 1985, pp. 223-241.

[54] T. Mandel, *The Elements of User Interface Design*, 1st ed. New York, USA: John Wiley & Sons, Inc., 1997, pp. 70-89.

[55] J. M. Matthews and S. Jahanian, "A Pedagogical Strategy for Gradual Enhancement of Creative Performance of the Students," *European Journal of Engineering Education*, vol. 24, 2012/01/09 1999, pp. 49-58.

[56] J. N. Mazon and J. Trujillo, "A Model Driven Modernization Approach for Automatically Deriving Multidimensional Models in Data Warehouses," in *Proceedings of the 26th international conference on Conceptual modeling (ER'07)*, Auckland, New Zealand, 2007, pp. 56-71.

[57] J. N. Mazon, J. Trujillo, and J. Lechtenborger, "A Set of QVT Relations to Assure the Correctness of Data Warehouses by Using Multidimensional Normal Forms," in *Proceeding of The 25th International Conference on Conceptual Modelling*, Tucson, AZ, 2006, pp. 385-398.

[58] M. McCool, "Adapting E-Learning for Japanese Audiences Tutorial," *Professional Communication, IEEE Transactions on*, vol. 49, 2006, pp. 335-345.

[59] T. O. Meservy and K. D. Fenstermacher, "Transforming Software Development: An MDA Road Map," *IEEE Computer*, vol. 38, 2005, pp. 52-58.

[60] MicroTool. *Model-Driven Development (MDD) with ObjectiF* [Online], 2011, pp. 2-13. Avaliable: http://www.microtool.de/objectif/en/mdd.asp. [Access: 09/08/2011]

[61] MicroTool. *ObjectiF -- the UML-Tool for Model-Driven Developemtn in C#, C++ and Java*, 2011, pp.1-5. http://www.microtool.de/objectif/en/index.asp. [Access: 10/21/2011]

[62] J. Miller and J. Mukerji, *Model Driven Architecture, MDA Guide*, 1.0.1, OMG Standard, 2003, pp. 33-97.

[63] S. J. Miller, K. Scott, A. Uhl, and D. Weise, *MDA Distilled: Principles of Model-Driven Architecture*, 1st ed. Boston, USA: Addison-Wesley Professional, 2004, pp. 12-105.

[64] ModelMakerTools. *ModelMaker Tools: Delphi and C# .Net Refactoring & UML Modelling* [Onlin], 2011, pp.1-5. Avaliable: http://www.modelmakertools.com/. [Access: 20/07/2011].

[65] S. Mohammad, "Effectiveness of E-Learning System," in *Computer Engineering and Technology, 2009 (ICCET '09) International Conference on*, 2009, pp. 390-394.

[66] G. Morine-Dershimer, T. Kent, J. Gess-Newsome, and N. Lederman, "The Complex Nature and Sources of Teachers' Pedagogical Knowledge Examining Pedagogical Content Knowledge." in *Examining Pedagogical Content Knowledge*, vol. 6, section 2, Netherlands: Springer, 2002, pp. 21-50.

[67] A. Nagy, "The Impact of E-Learning," in *E-Content: Technologies and Perspectives for the European Market.* , P. A. Bruck, A. Buchholz, Z. Karssen, and A. Zerfass, Eds., Berlin: Springer-Verlag, 2005, pp. 79-96.

[68] ObjecteeringSoftware. *Objecteering MDA Modeler* [Online], 2008, pp.2-4. Avaliable: http://www.objecteering.com/products_mda_modeler.php.

[69] OMG. *Domain Specifications* [Online], OMG Standard, 2009, pp. 60-74. Avaliable:*http://www.omg.org/technology/documents.htm*.

[70] OMG. *Executive Overview, http://www.omg.org/mda/executive_overview.htm*. 2010, pp. 23-90. [Access: 20/12/2011]

[71] OMG. *MDA FAQ,* Avaliable: http://www.omg.org/mda/faq_mda.htm. 2010, pp. 10-43. [Access: 25/09/2011]

[72] OMG. *MDA Specifications,* Avaliable: http://www.omg.org/mda/specs.htm. 2007, pp. 101-129. [Access: 18/10/2011]

[73] OMG. *MDA,* 2003, pp. 10-146. Avaliable: *http://www.omg.org/mda/*. [Access:20/09/2009]

[74] OMG. *Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT), v1.1* [Online], OMG Standard, Available: http://www.omg.org/spec/QVT/1.1/, 2004, pp. 13-40. [Access:20/09/2009]

[75] OMG, "*MOF 2.0 Core RFP,*" OMG Document ad/2001-11-052001, pp.1-2.

[76] OMG. *MOF 2.0/XMI Mapping Specification v2.1*, OMG Document ad/2010-06-022005 , 2010, pp. 32-45.

[77] OMG. *MOF Core Specification* [Online], v2.0, OMG Standard, 2007, pp. 49-69. Available: http://www.omg.org/spec/MOF/2.0/PDF/. [Access:20/09/2009]

[78] OMG. *OMG's Executive Overview of MDA* [Online], 2008, pp. 1-2. Available: www.omg.org/mda/executive_overview.htm.[Access:16/09/2009]

[79] OMG. *OMG Unified Modeling Language(OMG UML), Infrastructure*, v2.4, OMG Documentation, 2007, pp. 13-45.

[80] OMG. *OMG Unified Modeling Language(OMG UML), Superstructure*, v2.4. OMG Documentation, 2007, pp. 10-79.

[81] OMG, "*U2P UML Infrastructure*," OMG Document ad/03-03-01, 2003, pp.10-15.

[82] OMG. *XML Metadata Interchange (XMI)*, v2.0, OMG Document ad/07-10-01 2007, pp. 1-14.

[83] OMG and Linda. *UML Profile Specification*. OMG Document ad/10-06-05, 2010, pp. 1-3.

[84] OpenArchitectureWare. *Official OpenArchitectureWare Homepage*, 2009, pp.1. Avaliable: *http://www.openarchitectureware.org/*. [Access:03/01/2011]

[85] php.net. *PHP: What Can PHP Do?* [Online], 2011, pp.1. [Access:06/10/2011]

[86] S. R. Pressman, *Software Engineering: A Practitioner's Approach*, 6th ed. Boston: Mass: McGraw-Hill, 2005, pp. 71-85.

[87] H. Qian jin and W. Min, "LTSA-Based intelligent E-Learning System of the personalized interactive mechanism," in *Information Engineering and Computer Science, 2009 (ICIECS 2009). International Conference on*, 2009, pp. 1-3.

[88] L. A. Rahim and S. B. R. S. Mansoor, "Proposed Design Notation for Model Transformation," in *Software Engineering, 2008 (ASWEC 2008). 19th Australian Conference on*, 2008, pp. 589-598.

[89] J. Riddell. *Umbrello UML Modeller*, http://uml.sourceforge.net/. 2008, pp.1-3.

[90] F. Ruiz, A. V. Barceló, and e. al, "An Ontology for the Management of Software Maintenance Projects," *International Journal of Software Engineering and Knowledge Engineering*, vol. 14, 2004, pp. 323-349.

[91] SelectBusinessSolutions. *Select Solution For MDA*, 2010, pp. 1-2, Avaliable: http://www.selectbs.com/analysis-and-design/select-solution-for-mda. [Access:05/10/2011]

[92] L. S. Shulman, "Teacher Development: Roles of Domain Expertise and Pedagogical Knowledge," *Journal of Applied Developmental Psychology*, vol. 21, 2000, pp. 129-135.

[93] Y. Singh and M. Sood, "Models and Transformations in MDA," in *Computational Intelligence, Communication Systems and Networks, 2009. CICSYN '09. First International Conference on*, 2009, pp. 253-258.

[94] F. Solms and D. Loubser, "Generating MDA's Platform Independent Model Using URDAD," *Knowledge-Based Systems*, vol. 22, 2009, pp. 174-186.

[95] SparxSystems. *Enterprise Architect Overview*, 2010, pp. 1-2, Avaliable: http://www.sparxsystems.com/products/ea/index.html. [Access:15/11/2011]

[96] D. Spinellis. *UMLGraph: Automated Drawing of UML Diagrams*, 2009, pp. 1-3, Avaliable: http://www.umlgraph.org/. [Access:11/06/2010]

[97] C. Stark, K. J. Schmidt, L. Shafer, and M. Crawford, "Creating E-Learning Programs: A Comparison of Two Programs," in *Frontiers in Education, 2002. FIE 2002. 32nd Annual*, vol.1, 2002, pp. T4E-1-T4E-6.

[98] StarUML. *StarUML - The Open Source UML/MDA Platform*, 2009, pp.1-4. Avaliable: http://staruml.sourceforge.net/en/. [Access:05/10/2011]

[99] P. Tamir, "Subject Matter and Related Pedagogical Knowledge in Teacher Education," *Teaching and Teacher Education*, vol. 4, 1988, pp. 99-110.

[100] D. Tavangarian, M. Leypold, K. Nölting, and M. Röser, "Is E-Learning the Solution for Individual Learning?," *Journal of e-Learning*, 2004, pp. 21-29.

[101] Y. Tian, F. Chen, H. Yang, and L. Landy, "An Ontology-Based Model Driven Approach for a Music Learning System," in *21st International Conference on Software Engineering and Knowledge Engineering (SEKE'09)*, Boston, USA, 2009, pp. 739-744.

[102] Y. Tian, H. Yang, and L. Landy, "MDA-based Development of Music-Learning System," presented at *the Proceedings of the 14th Chinese Automation & Computing Society Conference in the UK*, Brunel University, West London, UK, 2008.

[103] M. Tong, Q. Liu, and X. Liu, "A Service Context Model based on Ontology for Content Adaptation in E-Learning," in *Frontiers in Education Conference (FIE), 2010 IEEE*, 2010, pp. S1D-1-S1D-5.

[104] M. Weisfeld, *Object-Oriented Thought Process, The*. 2nd ed. Indianapolis, IN, USA: Sams, 2003, pp. 113-150.

[105] S. Xu and H. Shen, "QoS-Oriented Content Delivery in E-Learning Systems," in *IT in Medicine & Education, 2009. ITIME '09. IEEE International Symposium on*, 2009, pp. 665-670.

[106] H. Yang and Y. Tian, "Software Engineering Creative Computing," presented at *the Conference on Computer Science and Software Engineering 2010 (CSSE 2010)*, Taiwan, 2010.

[107] H. Yang and M. Ward, *Successful evolution of software systems*, Norwood, MA, USA: Artech House, Inc., 2003, pp. 59-149.

[108] L. Zhuhadar, O. Nasraoui, R. Wyatt, and E. Romero, "Multi-Model Ontology-Based Hybrid Recommender System in E-Learning Domain," in *Web Intelligence and Intelligent Agent Technologies, (WI-IAT '09), IEEE/WIC/ACM International Joint Conferences on*, 2009, pp. 91-95.

[109] M. Zimmermann, "Adaption of Multimedia E-Learning Services to Mobile Environments," in *IEEE Global Engineering Education Conference (EDUCON)* , 2011, pp. 671-678.

Appendix A
Prototype of EARS II

This section presents php files of the manually generated code for EARSII system. It is only a prototype, which does not cover full details of functions and user interfaces.

learnerLogin.php

```
<html>
<title>EARS2:Login</title>
<style type="text/css" >
form { width:500px;}
input { font-family:Arial; }
</style>
<body>
<div align="center"><img src="./files/images/LogoV1.jpg" alt="EARSII"
longdesc="http://www.dmuchina.com/ears2"></div>
<?php
echo “Username: <input type="text" name="username"/><br/>”;
echo “Password: <input type="password" name="password" /><br/>”;
echo “<input type="submit" name="submit" value="Submit" align="right" />”;
echo “<input type="reset" name="cancel" value="Cancel"    align="right" />”
?>   <form action="checkNewLogin.php" method="post">
</body>
</html>
```

learnerLogout.php

```
<?php
$t="EARSII";
echo "You have been logged out. Thank you for using the". t ."system.:<br />";
?>
```

learnerInfo.php

```
<?php
echo “User Information:<br/>”;
echo Info;
?>
```

understanding.php

```
<div id="content">
        <div id="understanding">
          <table width="100%"><tr align="left"><td>
          <?php
          $barW="99%";
          include ("../OldnavBar.php");
          ?>
               </td></tr></table>
          </div>
          <div id="centerFrame"><table cellspacing=0 cellpadding=5 border=2
style="border-collapse: collapse" bordercolor="#FF0000" width="578">
```

```
            <tr><td>
          <div id="mainCon">
          <?php
          include ($content);
          ?>
          </div>
          <table width="100%"    border="0">
            <tr>
              <td align="center"><img name="timeline" src="../contents/pics/timeline.jpg"
width="590" height="30" alt=""></td>
            </tr>
          </table> </td></tr></table></div>
   </div>
```

listening.php

```
<div id="content">
        <div id="listening">
          <table width="100%"><tr align="left"><td>
          <?php
          $barW="99%";
          include ("../OldnavBar.php");
          ?>
              </td></tr></table>
          </div>
          <div id="centerFrame"><table cellspacing=0 cellpadding=5 border=2
style="border-collapse: collapse" bordercolor="#FF0000" width="578">
```

```
            <tr><td>
          <div id="mainCon">
          <?php
          include ($content);
          ?>
          </div>
          <table width="100%"    border="0">
            <tr>
              <td align="center"><img name="timeline" src="../contents/pics/timeline.jpg"
width="590" height="30" alt=""></td>
            </tr>
          </table>
          </td></tr></table></div>
   </div>
```

Doing.php

```
<div id="content">
        <div id="doing">
          <table width="100%"><tr align="left"><td>
          <?php
          $barW="99%";
          include ("../OldnavBar.php");
          ?>
              </td></tr></table>
          </div>
          <div id="centerFrame"><table cellspacing=0 cellpadding=5 border=2
```

```
style="border-collapse: collapse" bordercolor="#FF0000" width="578">
            <tr><td>
          <div id="mainCon">
          <?php
          include ($content);
          ?>
          </div>
          <table width="100%"    border="0">
            <tr>
              <td align="center"><img name="timeline" src="../contents/pics/timeline.jpg"
width="590" height="30" alt=""></td>
            </tr>
          </table>
          </td></tr></table></div>
  </div>
```

learnerNavBar.php

```
<div id="content">
        <div id="navBar">
          <table width="100%"><tr align="left"><td>
          <?php
          $barW="99%";
          include ("../OldnavBar.php");
          ?>
              </td></tr></table>
          </div>
```

```
        <div id="centerFrame"><table cellspacing=0 cellpadding=5 border=2
style="border-collapse: collapse" bordercolor="#0000FF" width="600">
            <tr><td>
        <div id="mainCon">
        <?php
        include ($content);
        ?>
        </div>
        <table width="100%"    border="0">
            <tr>
                <td align="center"><img name="timeline" src="../contents/pics/timeline.jpg"
width="590" height="30" alt=""></td>
            </tr>
        </table>
        </td></tr></table></div>
    </div>
```

loginRsp.php

```
<?php
$savePath="./admin/temp/";
session_save_path($savePath);
session_start();
$_SESSION["admin"] = 0;
$error_txt="";
?>
<?php
```

```
$username=$_POST["username"];
$password=$_POST["password"];
require ("./conn/dbconn.php");
$db=dbconn();
if (!$db)
{
	die('Could not connect: ' . mysql_error());
}else {
	//echo "Connected! <br />";
}
$db_ears2=dbname();
mysql_select_db($db_ears2, $db);
$sql = "SELECT adminUsername, adminPassword, adminLevel FROM admin";
//$sql = "SELECT * FROM admin";
$result = mysql_query( $sql );
$level="admin";
//$userInfo = @mysql_fetch_array($result);
	while ($userInfo = mysql_fetch_array($result)) {
		if ($userInfo["adminUsername"] == $username && $userInfo["adminPassword"] ==
$password && $userInfo["adminLevel"] == $level) {
			$_SESSION["admin"] = 1;
			$_SESSION["username"]=$username;
			$_SESSION["password"]=$password;
			echo "Session admin = ".$_SESSION["admin"] . "<br/>";
			echo("<meta http-equiv=refresh content='0; url=edit.php'>");
			$flag=1;
```

```
            } else {
            //die("Username or password not correct!");
            //echo "Username or Password is not correct!111";
            //$error_txt="Username or Password is not correct!";
            }
        }
        if(!($flag==1)){
        echo("<meta http-equiv=refresh content='0; url=adminLogin.php'>");
        }
mysql_close($db);
function error_txt()
{
        return $error_txt;
}
?>
```

logoutRsp.php

```
<?php
function updateRecord(){
        //to update user record database information
        echo("<meta http-equiv=refresh content='0; url=./Model/learnerRecDBConn.php'>");
}
function showLogoutView(){
         echo("<meta http-equiv=refresh content='0; url=./View/learnerLogout.php'>");
         }
?>
```

userInfo.php

```
<?php
$savePath="./temp/";
session_save_path($savePath);
session_start();
$_SESSION["admin"] = 0;
$error_txt="";
?>
<?php
$flag=0;
$username=$_POST["username"];
$password=$_POST["password"];
require ("./conn/dbconn.php");
$db=dbconn();
if (!$db)
{
	die('Could not connect: ' . mysql_error());
}else {
	//echo "Connected! <br />";
}
$db_ears2=dbname();
mysql_select_db($db_ears2, $db);
$sql = "SELECT adminUsername, adminPassword FROM admin";
$result = mysql_query( $sql );
//$userInfo = @mysql_fetch_array($result);
```

```
    while ($userInfo = mysql_fetch_array($result)) {
        if ($userInfo['adminUsername'] == $username && $userInfo['adminPassword'] ==
$password ) {
            $flag=1;
            $_SESSION["admin"] = 1;
            $_SESSION["username"]=$username;
            $_SESSION["password"]=$password
        } else {
        }
    }
    if($_SESSION["admin"]==1){
            echo "Session admin = ".$_SESSION["admin"] . "<br/>";
            echo("<meta http-equiv=refresh content='0; url=page.php'>");
    }else{
    echo("<meta http-equiv=refresh content='0; url=login.php'>");
    }
mysql_close($db);
function error_txt()
{return $error_txt;
}
?>
```

contentCtr.php

```
<?php
$edTitle=$_POST['edTitle'];
$edCont=$_POST['edCont'];
```

```
$parentNode=$_POST['parentNode'];
$flag=0;
require ("./conn/dbconn.php");
$db=dbconn();
if (!$db)
{
	die('Could not connect: ' . mysql_error());
}else {
	//echo "Connected! <br />";
}
$db_ears2=dbname();
mysql_select_db($db_ears2, $db);
$sql="SELECT * FROM contents WHERE Cont_Title='$edTitle'";
$result = mysql_query( $sql );
$userInfo = @mysql_fetch_array($result);
$sql2="SELECT * FROM contents WHERE Cont_Title='$parentNode'";
$result2 = mysql_query( $sql2 );
$subOfNo=$result2['Cont_Id'];
	if (!empty($userInfo)) {
	$sql = "UPDATE contents SET Cont_Text = '$edCont', subOf=$subOfNo WHERE
Cont_Title='$edTitle';";
	mysql_query( $sql );
	$msg="Page"."'".$edTitle."'"." is successfully updated.";
		}else{
	$sql = "INSERT INTO contents (Cont_Title, Cont_Text, Version, subOf) VALUES
('$edTitle','$edCont', '1', '$subOfNo');";
```

```
	mysql_query( $sql );
	$msg="Page"."'".$edTitle."'"." is added as a new page.under folder".$parenNode;
	//mysql_query("INSERT INTO contents (Cont_Title, Cont_Txt) VALUES
('$edTitle','$edCont');";);
	}
	echo("<meta http-equiv=refresh content='0; url=edit.php?t=HOME&m=".$msg."'>");
if ( isset( $_POST['edCont']) )
	$postArray = &$_POST ;			// 4.1.0 or later, use $_POST
else
	$postArray = &$HTTP_POST_VARS ;	// prior to 4.1.0, use HTTP_POST_VARS
foreach ( $postArray as $sForm => $value )
{
	if ( get_magic_quotes_gpc() )
		$postedValue = htmlspecialchars( stripslashes( $value ) ) ;
	else
		$postedValue = htmlspecialchars( $value ) ;
}
?>
```

navCtr.php

```
<style type="text/css">
/* common styling */
.menu {font-family: arial; width:630px; position:relative; margin:0; font-size:11px; padding:0;}
.menu ul li a {display:block; text-decoration:none; color:#003399;width:220px; height:20px;
text-align:center; border:1px solid #003399; background:#fff; line-height:20px; font-size:12px;
overflow:hidden;}
```

```
.menu ul li a:visited {display:block; text-decoration:none; color:#003399; width:220px;
height:20px; text-align:center; border:1px solid #003399; background:#fff; line-height:20px;
font-size:12px; overflow:hidden;}
.menu ul {padding:0; margin:0;list-style-type: none; }
.menu ul li {float:left; margin-right:1px; position:relative;}
.menu ul li ul {display: none;}
/* specific to non IE browsers */
.menu ul li:hover a.red {color:#fff; background:#ff0000;}
.menu ul li:hover a.blue {color:#fff; background:#0000ff;}
.menu ul li:hover a.green {color:#003399; background:#00ff00;}
.menu ul li:hover ul {display:block; position:absolute; top:21px; left:0; width:105px;}
.menu ul li:hover ul li a.hide {background:#ff0000; color:#003399;}
.menu ul li:hover ul li:hover a.hide {background:#ff0000; color:#000;}
.menu ul li:hover ul li ul {display: none;}
.menu ul li:hover ul.red li a{display:block; background:#ff0000; color:#fff;}
.menu ul li:hover ul.blue li a {display:block; background:#0000ff; color:#fff;}
.menu ul li:hover ul.green li a {display:block; background:#00ff00; color:#fff;}
.menu ul li:hover ul li a:hover {background:#ff00000; color:#003399;}
.menu ul li:hover ul li:hover ul {display:block; position:absolute; left:105px; top:0;}
.menu ul li:hover ul li:hover ul.left {left:-105px;}
/*Red Menu*/
/*
.menuRed {font-family: arial; width:220px; position:relative; margin:0; font-size:11px;
padding:0;}
.menuRed ul li a {display:block; text-decoration:none; color:#fff;width:220px; height:22px;
text-align:center; border:1px solid #fff; background:#cc0000; line-height:20px; font-size:14px;
```

```
overflow:hidden; font-weight:bold; font:Arial;}
.menuRed ul li a:visited {display:block; text-decoration:none; color:#fff; width:220px;
height:20px; text-align:center; border:1px solid #fff; background:#cc0000; line-height:20px;
font-size:14px; overflow:hidden; font-weight:bold;}
.menuRed ul {padding:0; margin:0;list-style-type: none; }
.menuRed ul li {float:left; margin-right:1px; position:relative;}
.menuRed ul li ul {display: none;}
/* specific to non IE browsers *//*
.menuRed ul li:hover a {color:#fff; background:#cc0000;}
.menuRed ul li:hover ul {display:block; position:absolute; top:21px; left:0; width:105px;}
.menuRed ul li:hover ul li a.hide {background:#cc0000; color:#003399; font-size:12px;}
.menuRed ul li:hover ul li:hover a.hide {background:#cc0000; color:#000; font-size:12px;}
.menuRed ul li:hover ul li ul {display: none;}
.menuRed ul li:hover ul li a {display:block; background:#cc0000; color:#fff; font-size:12px;}
.menuRed ul li:hover ul li a:hover {background:#cc00000; color:#003399; font-size:12px;}
.menuRed ul li:hover ul li:hover ul {display:block; position:absolute; left:105px; top:0;}
.menuRed ul li:hover ul li:hover ul.left {left:-105px;}*/

.menuRed {font-family: arial; width:220px; position:relative; margin:0; font-size:11px;
padding:0;}
.menuRed ul li a {display:block; text-decoration:none; color:#fff;width:220px; height:22px;
text-align:center; border:1px solid #fff; background:#cc0000; line-height:20px; font-size:14px;
overflow:hidden; font-weight:bold; font:Arial;}
.menuRed ul li a:visited {display:block; text-decoration:none; color:#fff; width:220px;
height:20px; text-align:center; border:1px solid #fff; background:#cc0000; line-height:20px;
font-size:14px; overflow:hidden; font-weight:bold;}
```

```
.menuRed ul {padding:0; margin:0;list-style-type: none; }
.menuRed ul li {float:left; margin-right:1px; position:relative;}
.menuRed ul li ul {display: none;}
/* specific to non IE browsers */
.menuRed ul li:hover a {color:#fff; background:#cc0000;}
.menuRed ul li:hover ul {display:block; position:absolute; top:21px; left:0; width:105px;}
.menuRed ul li:hover ul li a.hide {background:#cc0000; color:#003399; font-size:12px;}
.menuRed ul li:hover ul li:hover a.hide {background:#cc0000; color:#000; font-size:12px;}
.menuRed ul li:hover ul li ul {display: none;}
.menuRed ul li:hover ul li a {display:block; background:#cc0000; color:#fff; font-size:12px;}
.menuRed ul li:hover ul li a:hover {background:#999999; font-style:italic;/*color:#003399;*/
font-size:12px;}
.menuRed ul li:hover ul li:hover ul {display:block; position:absolute; left:105px; top:0;}
.menuRed ul li:hover ul li:hover ul.left {left:-105px;}

/* Green Menu */
.menuGreen {font-family: arial; width:220px; position:relative; margin:0; font-size:11px;
padding:0;}
.menuGreen ul li a {display:block; text-decoration:none; color:#fff;width:220px; height:22px;
text-align:center; border:1px solid #fff; background:#009933; line-height:20px; font-size:14px;
overflow:hidden;font-weight:bold;}
.menuGreen ul li a:visited {display:block; text-decoration:none; color:#fff; width:220px;
height:20px; text-align:center; border:1px solid #fff; background:#009933; line-height:20px;
font-size:14px; overflow:hidden;font-weight:bold;}
.menuGreen ul {padding:0; margin:0;list-style-type: none; }
.menuGreen ul li {float:left; margin-right:1px; position:relative;}
```

```
.menuGreen ul li ul {display: none;}
/* specific to non IE browsers */
.menuGreen ul li:hover a {color:#fff; background:#009933;}
.menuGreen ul li:hover ul {display:block; position:absolute; top:21px; left:0; width:105px;}
.menuGreen ul li:hover ul li a.hide {background:#009933; color:#003399; font-size:12px;}
.menuGreen ul li:hover ul li:hover a.hide {background:#009933; color:#000; font-size:12px;}
.menuGreen ul li:hover ul li ul {display: none;}
.menuGreen ul li:hover ul li a {display:block; background:#009933; color:#fff; font-size:12px;}
.menuGreen ul li:hover ul li a:hover {background:#999999; font-style:italic; /*color:#003399;*/
font-size:12px;}
.menuGreen ul li:hover ul li:hover ul {display:block; position:absolute; left:105px; top:0;}
.menuGreen ul li:hover ul li:hover ul.left {left:-105px;}

/* Blue Menu */
.menuBlue {font-family: arial; width:220px; position:relative; margin:0; font-size:11px;
padding:0;font-weight:bold;}
.menuBlue ul li a {display:block; text-decoration:none; color:#fff;width:220px; height:22px;
text-align:center; border:1px solid #fff; background:#003399; line-height:20px; font-size:14px;
overflow:hidden;font-weight:bold;}
.menuBlue ul li a:visited {display:block; text-decoration:none; color:#fff; width:220px;
height:20px; text-align:center; border:1px solid #fff; background:#003399; line-height:20px;
font-size:14px; overflow:hidden;}
.menuBlue ul {padding:0; margin:0;list-style-type: none; }
.menuBlue ul li {float:left; margin-right:1px; position:relative;}
.menuBlue ul li ul {display: none;}
/* specific to non IE browsers */
```

```
.menuBlue ul li:hover a {color:#fff; background:#003399;} /*when mouse move on
MUSIC/TECH/THEORY*/
.menuBlue ul li:hover ul {display:block; position:absolute; top:21px; left:0; width:105px;}
.menuBlue ul li:hover ul li a.hide {background:#003399; color:#fff; font-size:12px;}
.menuBlue ul li:hover ul li:hover a.hide {background:#003399; color:#fff; font-size:12px;}
.menuBlue ul li:hover ul li ul {display: none; font-size:12px;}
.menuBlue ul li:hover ul li a {display:block; background:#003399; color:#fff; font-size:12px;}
.menuBlue ul li:hover ul li a:hover {background:#999999; font-style:italic;
/*color:#000;*/font-size:12px;}
.menuBlue ul li:hover ul li:hover ul {display:block; position:absolute; left:105px; top:0;}
.menuBlue ul li:hover ul li:hover ul.left {left:-105px;}
.STYLE3 {color: #FFFFFF}

#tableNav { border-collapse:collapse; border:1px; border-color:#003399;}
</style>
<!--[if lte IE 6]>
<style type="text/css">
.menu ul li a.hide, .menu ul li a:visited.hide {display:none;}
.menu ul li a:hover ul li a.hide {display:none;}
.menu ul li a:hover {color:#fff; background:#36f;}
.menu ul li a:hover ul {display:block; position:absolute; top:21px; left:0; width:105px;}
.menu ul li a:hover ul li a.sub {background:#6a3; color:#fff;}
.menu ul li a:hover ul li a {display:block; background:#ddd; color:#000;}
.menu ul li a:hover ul li a ul {visibility:hidden;}
.menu ul li a:hover ul li a:hover {background:#6fc; color:#000;}
.menu ul li a:hover ul li a:hover ul {visibility:visible; position:absolute; left:105px; top:0;
```

```
color:#000;}
.menu ul li a:hover ul li a:hover ul.left {left:-105px;}
</style>
<![endif]-->
<table id="tableNav">
<tr>
<td>
<div class="menuRed">
<ul>
    <li><?php $t="Music";       echo"<a class='hide' href='page.php?t=".$t."'>".$t."</a>";?>
        <ul>
        <li><?php
                $t="What is electroacoustic music?";
                echo"<a href='page.php?t=".$t."'>".$t."</a>";
            ?>
        </li>
        <li><?php
                $t="Real-world sound";
                echo"<a href='page.php?t=".$t."'>".$t."</a>";
                ?>
            </li>
            <li><?php
                $t="Generated sounds";
                echo"<a href='page.php?t=".$t."'>".$t."</a>";
                ?>
            </li>
```

```
<li><?php
    $t="New types of sounds";
    echo"<a href='page.php?t=".$t."'>".$t."</a>";
    ?>
</li>
<li><?php
    $t="Genres";
    echo"<a href='page.php?t=".$t."'>".$t."</a>";
    ?>
</li>
<li><?php
    $t="Electroacoustic music and pop music";
    echo"<a href='page.php?t=".$t."'class='style5'>".$t."</a>";
    ?>
</li>
<li><?php
    $t="Compose electroacoustic music";
    $t2="Composition Techniques";
    echo"<a href='page.php?t=".$t2."'>".$t."</a>";
    ?>
</li>
<li><?php
    $t="Musical material";
    echo"<a href='page.php?t=".$t."'id='bottomLine'>".$t."</a>";
    ?>
</li>
```

```
            </ul>
        </li>
</ul>
</div>
</td>
<td>
<div class="menuBlue">
<ul>
        <li><?php $t="Technology";       echo"<a class='hide'
href='page.php?t=".$t."'>".$t."</a>";?></a>
            <ul>
            <li><?php
                    $t="Studio";
                    echo"<a href='page.php?t=".$t."'>".$t."</a>";
                    ?>
                </li>
                <li><?php
                    $t="New Instruments";
                    echo"<a href='page.php?t=".$t."'>".$t."</a>";
                    ?>
                </li>
                <li><?php
                    $t="Sound Manipulation";
                    echo"<a href='page.php?t=".$t."'id='bottomLine'>".$t."</a>";
                    ?>
                </li>
```

```
            </ul>
        </li>
</ul>
</div>
</td>
<td>
<div class="menuGreen">
<ul>
    <li><?php $t="Theory";      echo"<a class='hide' href='page.php?t=".$t."'>".$t."</a>";?>
            <ul>
            <li><?php
                    $t="Listening";
                    echo"<a href='page.php?t=".$t."'>".$t."</a>";
                    ?>
                </li>
                <li><?php
                    $t="Performance";
                    echo"<a href='page.php?t=".$t."'>".$t."</a>";
                    ?>
                </li>
                <li><?php
                    $t="Notation";
                    echo"<a href='page.php?t=".$t."'>".$t."</a>";
                    ?>
                </li>
                <li><?php
```

```
                    $t="Dictionary";
                    echo"<a href='page.php?t=".$t."'>".$t."</a>";
                    ?>
                </li>
            </ul>
        </li>
</ul>
</div>
</td>
</tr>
</table>
<p>
   <!--
<div class="menu">
<ul>
        <li><a class="red" href="#">Music</a>
            <ul class="red">
            <li><a href="#" title="The zero dollar ads page">What is M?</a></li>
            </ul>
        </li>
        <li><a class="blue" href="#">Music</a>
            <ul class="blue">
            <li><a href="#" title="The zero dollar ads page">What is M?</a></li>
            </ul>
        </li>
</ul>
```

```
</div>
-->
```

elearningDBConn.php

```
<?php
function dbconn(){
	$db=mysql_connect("localhost","root","");
	//$db=mysql_connect("mysql04.iomart.com","tiannz231","dmuchina100");
	if (!$db)
	{
	die('Could not connect: ' . mysql_error());
	}else {
	return $db;
	}

}
function dbname(){
	return "ears2v1";
	//return "tiannz231";
}
function dbdisconnect(){
	mysql_close($db);
}
?>
```

learnerRecDBConn.php

```
<?php
$db_ears2="learnerRec";
mysql_select_db($db_ears2, $db);
//$sql = "SELECT adminUsername, adminPassword FROM admin";
$sql = "SELECT * FROM admin";
$result = mysql_query( $sql );
?>
```

learningResDBConn.php

```
<?php
$db_ears2="learningRes";
mysql_select_db($db_ears2, $db);
$sql = "SELECT * FROM admin";
$result = mysql_query( $sql );
?>
```

Appendix B
List of Publications by Candidate

[1] Y. Tian, H. Yang and L. Landy, "MDA-Based Development of Music-Learning System," presented at *the Proceedings of the 14th Chinese Automation & Computing Society Conference in the UK*, Brunel University, West London, UK, 2008.

[2] Y. Tian, F. Chen, H. Yang and L. Landy, "An Ontology-Based Model Driven Approach for a Music Learning System," in *21st International Conference on Software Engineering and Knowledge Engineering (SEKE'09)*, Boston, USA, 2009, pp. 739-744.

[3] H. Yang and Y. Tian, "Software Engineering Creative Computing," presented at *the Conference on Computer Science and Software Engineering 2010 (CSSE'10)*, Taiwan, 2010.

[4] J. Kang, J. Li, J. Huang, Y. Tian and H. Yang, "Automating Business Intelligence Recovery from a Web-Based System," in *21st International Conference on Software Engineering and Knowledge Engineering (SEKE'09)*, Boston, USA, 2009, pp. 262-267.

[5] T. K. Chen, H. P. Fang, Y. Tian, H. L. Fang, Y. J. Li, S. H. Tseng, and S. E. Miao, "Design and Evaluation of Social Interfaces for Cultural Exhibitions of Chinese Shadow Puppetry," in *IEEE 35th Annual Computer Software and Applications Conference (COMPSAC'11)*, Munich, Germany, 2011, pp. 346-347.