# 國 立 交 通 大 學

資訊科學與工程研究所

**碩 士 論 文**

以主要功能需求導引臨摹學習
網路應用系統的軟體架構設計

Major-Requirement-First Imitating Learning for
Software Architecture Design of Web Based System

研 究 生：薛祖淵
指導教授：曾憲雄　博士

**中 華 民 國　九 十 八　年 七 月**

# 以主要功能需求導引臨摹學習 網路應用系統的軟體架構設計

# Major-Requirement-First Imitating Learning for Software Architecture Design of Web Based System

研 究 生：薛祖淵　　　　Student：Tsu-Yuan Hsueh

指導教授：曾憲雄　　　　Advisor：Dr. Shian-Shyong Tseng

國 立 交 通 大 學
資 訊 科 學 與 工 程 研 究 所
碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

In partial Fulfillment of the Requirements

for the Degree of

Master

In

Computer Science

July 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年七月

# 以主要功能需求導引臨摹學習
# 網路應用系統的軟體架構設計

學生：薛祖淵　　　指導教授：曾憲雄 博士

國立交通大學資訊學院
資訊科學與工程研究所

## 摘　要

軟體架構設計對於把設計概念應用在實作能力上，並且透過解構、抽象和封裝等概念去簡化整個系統功能需求的複雜度是非常重要的議題。在本篇論文中，我們以「錄影帶租借管理系統」當作我們教學網路應用系統邏輯層設計的教材，來教導學生軟體架構設計的概念。我們主要的教學策略導入了鷹架式教學的理論，導引學生從系統的主要功能需求往細部的元件一一去臨摹，並且在過程中提供必要的鷹架作為輔助。依照這個想法我們發展了一套臨摹式學習系統，我們定義了一個系統需求的知識本體去維護教材的知識架構，並且設計了一套對話式問答的機制去擷取老師設計教材的知識，然後產生出知識本體。接下來，根據我們的教學策略去規劃出臨摹的流程，然後透過教學專家系統去呈現教材內容和提供個人化的學習。最後，實驗結果顯示我們提出的教學策略對於學習軟體架構設計是有效的，而且我們會在未來的研究上加入適性化學習以提供更合適的教學。

**關鍵字：**軟體架構設計，物件導向程式設計，臨摹式學習，網路應用系統，鷹架式教學

# Major-Requirement-First Imitating Learning for Software Architecture Design of Web Based System

Student: Tsu-Yuan Hsueh    Advisor: Dr. Shian-Shyong Tseng

Department of Computer Science
National Chiao Tung University

## Abstract

Software architecture design is an important issue to transform design concepts into implementation ability, where the ideas of decomposition, abstraction, and encapsulation of functionality are usually used to simplify the complexity of the system's requirements. This thesis focusing on the logic tier of web-based system uses "Video Rental Management System (VRMS)" as our teaching case to teach learners how to design the software architecture. Major-Requirement-First strategy (MRFS), the main idea of this thesis, applies scaffolding instruction theory to guide learners imitating the system architecture design from major functionalities of VRMS to detailed components by providing all the necessary scaffolds. Based upon MRFS, we develop the Major-Requirement-First Imitating Learning System (MRFILS), where a knowledge structure of VRMS is constructed and maintained according to System Requirement Ontology (SRO), and the Recursive Descent Dialog Approach (RDDA) is proposed to acquire the teacher's knowledge of teaching cases and generate the SRO. Accordingly, the schedule of an imitating procedure can present the learning materials and provide personalized learning for learners with guidance using an Object-Oriented Learning Activity (OOLA) System. Finally, the experimental results show that our teaching approach is useful for learners to gain the software architecture design concepts of VRMS, and we will add adaptive learning in MRFILS for the future work.

**Keywords:** Software architecture design, object-oriented programming, imitating learning, web-based system design, scaffolding instruction.

# 致謝

在交大碩士班的兩年中，首先誠摯的感謝指導教授：曾憲雄博士。無論是在學術研究或是為人處世方面，皆讓我受益匪淺，尤其是我學到了對一個知識領域的研究方法、邏輯思考及表達能力的訓練，這將使我終生受用不盡，特別要在這對教授獻上十二萬分的感謝。同時也要特地感謝我的口試委員，賀嘉生教授、孫春在教授、廖岳祥教授和葉耀明教授，他們給了我相當多的寶貴意見，讓本論文更有意義與價值。

再來要感謝的是實驗室的林喚宇學長、劉怡利學姊、翁瑞峰學長、蘇俊銘學長、曲衍旭學長，在論文研究期間，不厭其煩的給予我非常多的建議及協助，讓我的論文更加完整。此外，我也從他們身上學到了不少生活態度以及系統實作上的技巧和經驗，在此深表感激。同時也要非常感謝實驗室的同窗夥伴們，惠君、靖雅、啟珺、士緯和世恒以及學弟妹們：金龍、嘉祥、國彰、杰峰、紹宜和佳榕，也要特別謝謝你們這段期間的互相鼓勵及支持，我才能堅持努力完成這篇碩士論文，謝謝你們。

祖淵

2009 年 7 月于新竹

# Table of Content

# List of Figures

# List of Tables

Page

# List of Definitions

Page

# List of Algorithms

# List of Examples

Page

# Chapter 1. Introduction

With the growth of Internet, many Internet service providers, such as Google, Yahoo, etc., who have provided a huge amount of web applications for various web services. In order to develop such web applications, the system architecture design becomes a crucial task. Currently, the most popular web system architecture design mechanism, named *Three-Tier Architecture* [1], consists of presentation tier, logic tier, and data tier to implement user interface, business logic, and data I/O, respectively. Among these three tiers, logic tier is responsible to handle complicated computations and program control-flows, so it usually becomes a bottleneck when novice programmers who just begin to learn web application design.

In the construction process of web application logic tier, designers are supposed to clearly understand the system requirements, and then the concepts of decomposition, abstraction, and encapsulation of functionality [1-4] are usually used to transform the requirements into the system design and simplify the complexity of requirements to each program component. We take a web application "*Video Rental Management System (VRMS)*" as an example, whose logic tier design contains various requirements of member management, video management, online item management and data passing, and this teaching case is simple and suitable for novice learners to learn the software architecture design. Thus, a learner, who wants to construct a VRMS, needs to consider the following design problems: How to divide the requirements into different components to simplify the individual computation logics? How to merge the computation results from different components to generate the final outputs? How to cope with data encapsulation and passing? In other words, the concept of *software architecture design* [3, 4] is a very important learning objective for constructing web application.

In the traditional software architecture design learning, lectures and imitating learning are the major methods for learners to gain the design idea. Among them, lecturers can teach learners the abstract concept of software architecture design, but learners usually can not easily apply the concepts to a real software design. As we know, imitating learning lets learners imitate the design of teaching cases [5] or class diagrams [1-3] to learn the real design experience, but it is still difficult to help learners understanding the major idea of the previous design from the detailed component implementation without any guidance and explanation. Another researches use project-based learning [6, 7] to teach learners software architecture design, where teachers guide learners to construct the main program in the project and gain the real design concepts and construction experience. This teaching approach can contribute significant learning efficacy, but it is very time consuming for teachers to run a project.

Most computer-assisted programming learning systems [8-11] aim to assist learners learning the syntax and semantic rules of programming language, but software architecture design is still inadequate for learning because the abstract design concepts are difficult to be implemented as the fixed learning rules in learning system. Therefore we propose a learning system, which can guide learners to imitate the design of a given learning project case to reduce teachers' loading of guiding learners to complete a project.

In order to help learners understanding why and how the software architecture design can satisfy the abstract requirements, **Major-Requirement-First Strategy (MRFS)** is proposed to guide learners to imitate program architectures from major functionalities of the system to detailed components. According to *scaffolding instruction theory* [12, 13], all the required scaffolds, including usable library packages and suggestions, are provided for learners to assist them completing the imitating tasks in the top-down imitating manner. This teaching approach can make learners understand the overall configuration of the system first, so they will easily comprehend the task and the design of each detailed component. Accordingly, we develop the **Major-Requirement-First Imitating Learning System (MRFILS)**, where a knowledge structure of our teaching case, *Video Rental Management System (VRMS)*, is constructed and maintained based on **System Requirement Ontology (SRO)**, and the **Recursive Descent Dialog Approach (RDDA)** is proposed to acquire the teacher's knowledge of teaching cases to generate the SRO by a series of dialogues. Based upon SRO, an imitating procedure which is scheduled by MRFS can present the learning materials and provide personalized learning for learners using an *Object-Oriented Learning Activity (OOLA) System* [14], which guides the learners to learn the software architecture design of VRMS.

Finally, we have done a real experiment to evaluate the learning efficacy and satisfaction of learners. The experimental results show that most of learners think this top-down imitating learning can make them understand the software architecture design effectively, and implementing the task with library packages supporting will easily comprehend how to design these used objects afterward.

The remainder of this thesis is organized as follows. In Chapter 2, we introduce some related works about the web-based system design learning and programming learning in e-learning. Chapter 3 describes our main idea, Major-Requirement-First Strategy (MRFS), and we propose a Major-Requirement-First Imitating Learning

System (MRFILS) in Chapter 4. Next, Chapter 5 discusses the system implementation and experiment. Finally, we give the conclusion and future works in Chapter 6.

# Chapter 2. Related Work

In general Computer Science curriculums, computer programming and practical techniques for implementation and application in computer system are important abilities for students. Figure 1 shows the typical catalogs of Computer Science curriculums [15]. From the coding's aspects such as semantic and syntax rules of programming language, data structure and algorithm, to the requirement' aspects like requirement analysis, system design, testing and deployment, the learners can gain the software development ability through taking lessons in a series of Computer Science curriculums.

However, as a result of the difficulty for learners to combine design concepts with implementation ability in construction phase [16], there still exists a gap between basic programming skills and software engineering abilities. Thus, the *software architecture design learning* [3, 4] aims to solve the issues about how to design the program architecture from requirement's aspects to coding's aspects. And in this thesis, we focus on the software architecture design learning of web-based system. The following sections will introduce related studies about present programming learning.

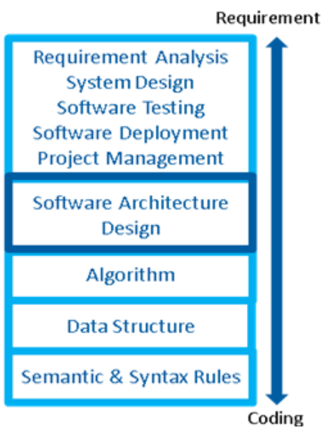


Figure 1. Typical catalogs of Computer Science curriculums

## 2.1 Web-Based System Design Learning

### *2.1.1 Traditional Teaching Approaches*

General programming capability on web-based system design can be learned by lectures, case study, or project-based learning. Lecturer gives lessons about basic concepts of web-based system development to learners, but the concept may be too abstract to link the concept with their implementation ability. Case study [1-3] is to consult relevant books, such as codes tracing to imitate related design concepts or use class diagram to figure out the whole system architecture. These imitating learning approaches are very difficult for helping learners understanding the design procedure and the physical meaning of the whole system architecture design without the ability of guidance and explanation. In project-based learning [6, 7], learners who explore real-world problems and challenges can be inspired to obtain a deeper knowledge of the subjects they're studying, but the teachers usually can not afford one-on-one instruction.

Therefore, there are some issues which are required to be proved in this thesis as follows: combine learner's implementation ability with design concepts, provide an appropriate guiding mechanism, make teaching cases contain semantic meaning, and reduce the teacher's loading for one-on-one instruction.

### *2.1.2 Design Methodologies*

In web-based system design [17], there are many methodologies such as Client-Server Architecture [18], Model-View-Controller Architecture, Component Based Methodology [19], and Model Driven Architecture [20], etc. Among these methodologies, three-tier architecture is a popular way for web-based system design [1]. Apart from the usual advantages of modular software with well defined interfaces, three-tier architecture also allows any of the three tiers to be upgraded or replaced independently as requirements or technology change.

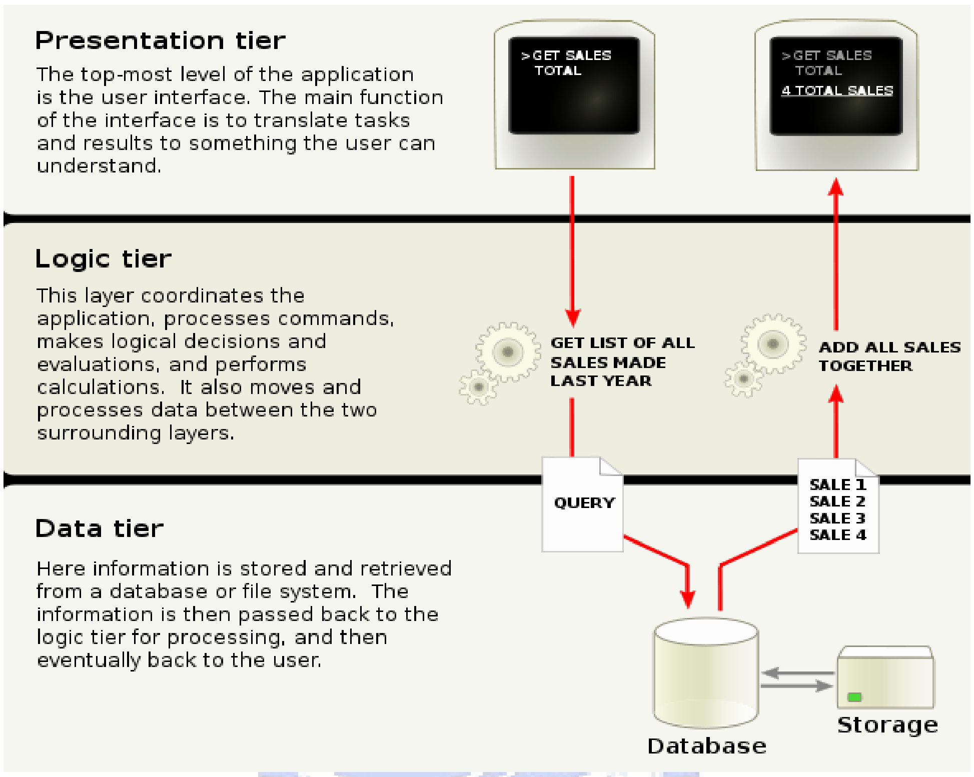

Presentation tier
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.
>GET SALES TOTAL
>GET SALES TOTAL
4 TOTAL SALES
Logic tier
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.
GET LIST OF ALL SALES MADE LAST YEAR
ADD ALL SALES TOGETHER
Data tier
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.
QUERY
SALE 1
SALE 2
SALE 3
SALE 4
Database
Storage

Figure 2. Three-tier architecture

Three-tier architecture is composed of *presentation tier*, *logic tier* and *data tier*. Figure 2 briefly describes the utility of each tier. Presentation tier converts and displays application data into a human-legible form, and it also provides an application's user controls. Logic tier implements business logic and translates the reality into programming objects. Data tier provides data storage and data access mechanisms to the application. Except for the user interface of presentation tier and the data I/O of data tier, the logic tier is responsible to handle complicated computations and program control-flows. For novice programmers who just begin to learn web application design, the logic tier design usually becomes a bottleneck for them. Thus, in this thesis we focus on the software architecture design of logic tier.

### *2.1.3 Software Architecture Design*

Designing, developing, and evolving complex software systems require a mastery of analytical and technical skills, as well as knowledge of appropriate processes, architectures and design patterns. Software architects building complex systems must create the illusion of simplicity through decomposition, abstraction, and encapsulation of functionality. The following issues will be discussed in software architecture design learning: the way to distribute the functionalities of software into different components, the components design to handle the logic computation, the responsibility of each component, the data and logic encapsulations, and the program control-flow, etc.

Surveying the researches about software architecture design learning, most researches adopt project-based learning to teach the design concepts by implementation. For example, Woei-Kae Chen and Yu Chin Cheng [6] designed an object-oriented programming (OOP) laboratory course where the students are required to implement a small-to-medium scale interactive computer game with framework-assisted. The students use the pre-define objects and source codes from teachers to modify and re-design the game scenario. The teaching assistants need to give all the students one-on-one instructions for guiding them to complete the lab, and it is a very time-consuming job.

## 2.2 Programming Learning in e-Learning

### *2.2.1 Object-Oriented Programming Learning*

Object-oriented programming (OOP) is a programming paradigm which uses objects and their interactions to design applications and computer programs, and it is very popular to be used in web-based system development. In the OOP learning researches of e-learning domain, most issues are relevant to teach the basic coding abilities and OO concepts such as inheritance, abstraction, encapsulation and polymorphism. The followings are relevant literature reviews about OOP learning

Stelios Xinogalos et al. [11] proposed a programming environment, objectKarel, which incorporates e-lessons, hands-on activities, an easy to use structure editor for developing and editing programs, program animation, explanatory visualization, highly informative and friendly error messages for novice programmers to develop programs.

J.Baltasar Garcia Perez-Schofield et al. [8] developed an interactive object-oriented environment, Visual Zero, which allows learners to concentrate on objects and their relationships from the very beginning, and thus helps the learners achieving a high degree of knowledge about the object-oriented programming paradigm.

Matthew Conway et al. [10] developed a 3D programming environment, Alice, to teach entry level Computer Science students basic programming concepts without exposing them to all the intricate details of a full-blown programming language. Using Alice, students accomplish their programming tasks by a series of mouse clicks and drag-and-drop maneuvers, and it eliminates from syntax and logical errors of programming.

The Lifelong Kindergarten research group at the MIT Media Lab [21] developed a new programming language, Scratch, to help the learners creating their own interactive stories, animations, games, music and arts. The learners can learn important mathematical and computational ideas, while also learn to think creatively, reason systematically, and work collaboratively.

### *2.2.1 Other Programming Learning Issues*

In addition to basic coding ability, there are some researches about data structure, algorithm, structured query language (SQL) and design pattern.

Jaime Galvez et al. [22] presented a blended e-learning experience consisting of supplying an undergraduate student population with a learning tool called OOPS and a testing system called SIETTE to teach data structure.

Maria Kordaki et al. [9] presented a modeling, student-centered methodology for the design of a constructivist computer environment, the SORTING environment, for the learning of sorting algorithms, within a context consisting of interactive, multiple and linked representation systems.

Claus Pahl and Clair Kenny [23] presented an automated learning and skills training system for a database programming environment that promotes procedural knowledge acquisition and skills training. The system provides meaningful knowledge-level feedback such as correction of student solutions and personalized guidance through recommendations.

Zoran Jeremic et al. [24] presented the evaluation of DEPTHS (Design Patterns Teaching Help System), an intelligent tutoring system (ITS) for teaching software design patterns. Furthermore, there are other studies of software development in general showing that adherence to common software design principles, guidelines, and rules [25].

However, most of the aforementioned researches provide computer-assisted programming learning systems or assisting tools to help learners learning the basic coding abilities of object-oriented programming, but there is no research providing the learning system to assist the software architecture design learning. In order to provide the learners one-on-one instruction in a cost effective manner and reduce the teacher's loading, we decide to construct an intelligent tutoring system for software architecture design learning.

# Chapter 3. Imitating Learning Guidance Approach

## 3.1 General Program Assignments

Generally, after a series of lectures in the programming course, the teacher usually assigns homework to make learners obtain a deeper knowledge of the design concepts they have learned. Figures 3 ~ 5 show three different implementation processes of program assignment, where the teacher uses the *Video Rental Management System (VRMS)* project as an example to make the students think how to design this project. The horizontal axis means the assignment iteration, which is set by teacher according to the project's features like the amount of functionality or component. And the vertical axis means the expected completeness degree of the project, called expected program progress.

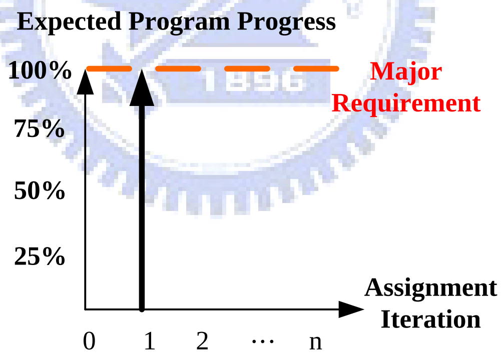


Figure 3. Traditional program assignment

Figure 3 shows that *traditional program assignment* assigns a whole VRMS project to students in one time. The teacher requests students to apply the knowledge which have been learned from the course to accomplish this project, but the scope of VRMS project is too big for students to know how to start implementing it. Hence, the students usually have trouble with this wide-scope assignment.

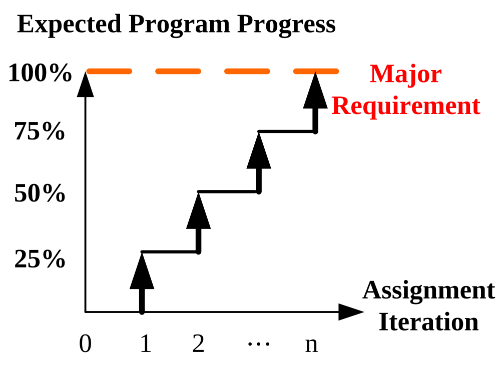


Figure 4. Progressive program assignment

Figure 4 shows that in *progressive program assignment*, the teacher segments the VRMS project into several assignments according to the constructing procedure from basic components to major requirements, and the learners are asked to complete parts of programs progressively. This bottom-up manner can make learners understand the constructing process, but they have no idea about why and how the design of basic components and its details.

## 3.2 Major-Requirement-First Strategy (MRFS)

Hence, we propose a teaching strategy named **Major-Requirement-First Strategy (MRFS)** to help learners easily understanding how to design the VRMS project. This teaching strategy adopts a top-down imitating learning mechanism which guides the learners from the major requirements of VRMS to basic components and apply the ideas of *Scaffolding Instruction Theory* [12, 13].

As mentioned in scaffolding instruction theory, the *Zone of Proximal Development (ZPD)* [12, 13] noted that it is also very efficient for learners to use their prerequisite knowledge to help themselves developing new knowledge. So we intend to combine the learners' implementation ability and request them to complete the project by imitating the software architecture which is designed by teachers in advance.

Moreover, scaffolding instruction theory also mentioned that building some scaffolds for learners and assisting them to complete the target they can’t accomplish before, and then gradually dismantling the scaffolds they have learned, is an effective way to get the learning objectives. So we provide the necessary scaffold which called *program scaffolds*, such as component requirements, interface and usable objects of each component, relationships between the components, library packages and source codes to assist learners to implement from the major requirements with these imitating information and library supporting. The detailed information of program scaffolds is shown in Table 1.

Table 1. The detailed information of program scaffolds

| Program Scaffold | Description |
|---|---|
| System Requirement | Describe the main target of this system. |
| Class Requirement | Describe the requirement and design concepts of this class. |
| Method Requirement | Describe the requirement and utility of this method. |
| Interface | Describe what methods are included in this class. |
| Usable Objects | Describe what objects will be used in this method. |
| Relationship | Describe the relationship between two components. |
| Library Package | A program library used for this component implementation. |
| Source Code | A solution code created by teacher for this component. |
| Class Diagram | An overall configuration of this system. |

Therefore, based on scaffolding instruction theory, MRFS guides learners to imitate the program architectures from major functionalities of system to detailed components through a series of implementations with program scaffolds supporting. As shown in Figure 5, *top-down program assignment* applies the idea of MRFS, where the teachers have to construct the necessary program scaffolds and segment the VRMS project into progressive assignments in advance, and then assign a task which is the major functionality of VRMS to students. After the students finish the assignment with program scaffolds supporting like library, they will continue to finish the detailed components of major functionality. Consequently, the learners can figure out the overall configuration of VRMS first, and then they will easily comprehend the design concepts of each detailed component. The learning efficacy of MRFS will be compared with general program assignments in our experiment in Chapter 5.

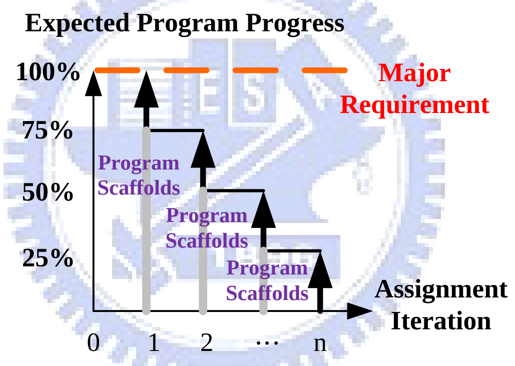


Figure 5. Top-down program assignment

**Example 1. Major-requirement-first strategy for program assignments**

Figure 6 shows our teaching case in this thesis, *Video Rental Management System (VRMS)*, which is represented by class diagram. VRMS is a common web-based system which can offer user convenient services about online video rental, such as video introduction, video search, popular video recommendation, subscribe and relend for video, online comment and member management, etc. Besides the user interface and database I/O, the learners are supposed to focus on logic tier design of VRMS.

The teacher will segment the project into several assignments, and ask the learners to complete the assignments from "*MemberMgr*", "*VideoMgr*" or "*OnlineMgr*", the major functionalities of VRMS, to detailed components. If the "*MemberMgr*" task is assigned to learners, all the information about the task like design concepts, component requirements, interface, usable objects and library packages like "*DBMgr*", "*MemberInfo*", etc. should also be provided to assist learners implementing this class. The learners can imitate the software architecture design of VRMS through a sequence of implementations with guidance and program scaffolds supporting in this top-down imitating manner.

Major-requirement-first strategy is not only adaptive for VRMS but also suitable for use in logic tier design of web-based system, and the learners can get the software architecture design concepts easily in this top-down imitating manner.

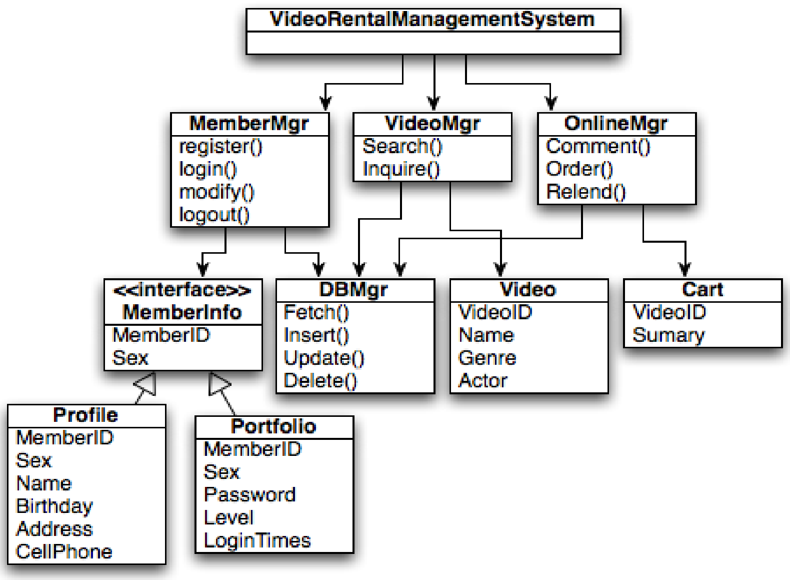


Figure 6. The class diagram of Video Rental Management System (VRMS)

# Chapter 4. Major-Requirement-First Imitating Learning System (MRFILS)

For the purpose to provide the learners one-on-one instruction in a cost effective manner and reduce the teacher's loading, we develop the **Major-Requirement-First Imitating Learning System (MRFILS)**. MRFILS provides guidance and necessary assistance for learners and assists them imitating the software architecture design of teaching cases which are constructed by teachers.

In order to apply MRFILS, the teachers have to construct necessary teaching materials for learners to imitate the software architecture design, and the learners need to have enough coding ability of object-oriented programming for implementation. The goal for this intelligent tutoring system is to help teachers constructing a knowledge structure of teaching case, and then applying a guiding mechanism to generate an imitating procedure, next guiding learners to imitate the software architecture design of teaching case through a series of implementations with program scaffolds supporting.

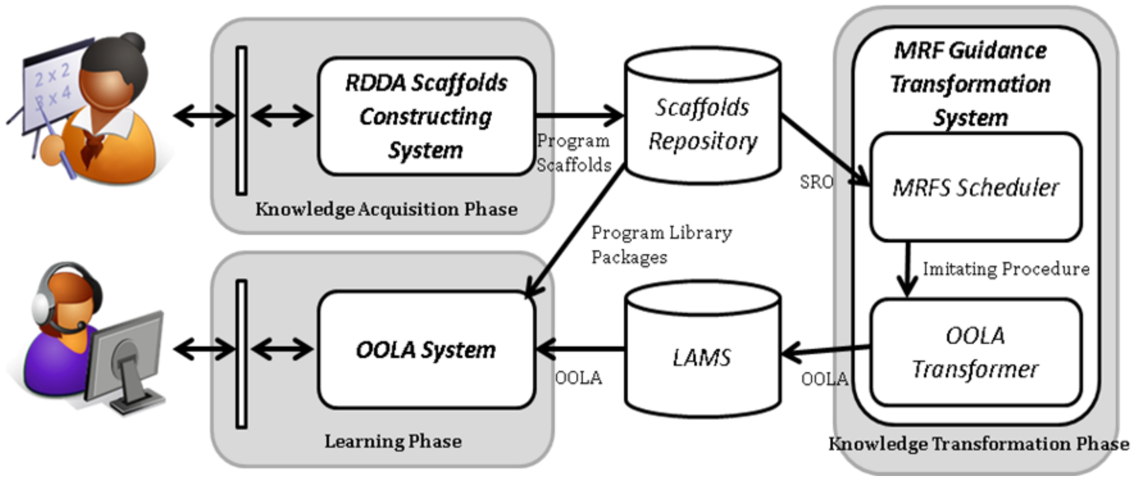


Figure 7. The system architecture of MRFILS

The MRFILS is composed of three phases, including *knowledge acquisition phase*, *knowledge transformation phase* and *learning phase*. The overall picture of system architecture is shown in Figure 7.

Firstly, in knowledge acquisition phase, the teachers construct the program scaffolds by the *RDDA Scaffolds Constructing System*, which applies the **Recursive Descent Dialog Approach (RDDA)** to acquire the teacher's knowledge of teaching case through a series of dialogues. After the construction, the program scaffolds that are stored in scaffolds repository will be composed into a knowledge structure of teaching case, named **System Requirement Ontology (SRO)**.

Next, in knowledge transformation phase, *MRFS scheduler* applies **Major-Requirement-First Strategy (MRFS)** on SRO to schedule an imitating procedure, and then *OOLA Transformer* arranges different *learning guidance templates* for imitating procedure to transform an *Object-Oriented Learning Activity (OOLA)* [14].

At last, in learning phase, *Object-Oriented Learning Activity (OOLA) System* retrieves learning activities and necessary program library packages from *Learning Activity Management System (LAMS)* [14] and scaffolds repository to present the guiding information and learning contents for learners. The following sections will introduce the detail of each phase in MRFILS.

## 4.1 Knowledge Acquisition Phase

### *4.1.1 System Requirement Ontology (SRO)*

For imitating learning, we need to construct teaching cases in advance. According to the features of logic tier design of web-based system, we define a knowledge structure named **System Requirement Ontology (SRO)** to maintain our teaching case, *Video Rental Management System (VRMS).*

Traditional learning content of programming case doesn't take the physical meanings of components and the relationships between components into account, so it's difficult for learners to get the design concepts of system. Thus, we think the SRO should contain semantic meanings like component's utility and physical meaning of interaction, and then we can apply MRFS to schedule an appropriate imitating procedure based on the information. Accordingly, the knowledge representation of SRO is defined as Definition 1, where p means the total number of node set and q means the total number of relationship set.

**Definition 1. System Requirement Ontology (SRO)**

- SRO = (N, R) : Composed of a set of nodes denoting to N and a set of relationships denoting to R.
- N = $\{n_1, n_2, n_3, \ldots, n_p\}$ : A set of nodes denoting the downstream nodes of n.
- R = $\{r_1, r_2, r_3, \ldots, r_q\}$ : A set of relationships denoting the downstream relationships of r.
- $n_i$ = ($nType_i$, $Name_i$, $Requirement_i$, $Interface_i$, $ClassQuantity_i$, $UsableObject_i$) : Composed of component type, component name, component requirements,

component interface, the quantity of class under the component, and the names of usable objects, accordingly.

- $r_i$ = ($rType_i$, $From_i$, $To_i$) : Composed of relationship type, relationship original node, relationship terminal node, accordingly.
- $nType_i \in$ {System, Class, Method} :
  - *System* node is the root of SRO.
  - *Class* node is the class name of system.
  - *Method* node is the method name of class.
- $rType_i \in$ { Be-composed-of, Contain, Use-data, Use-function, Such-as} :
  - *Be-composed-of* relationship describes what major functional classes constitute *System* node.
  - *Contain* relation describes what *Method* nodes are included in *Class* node.
  - *Use-data* relation indicates which data-storage *Class* node is used by *Method* node.
  - *Use-function* relation indicates which functional *Class* node is used by *Method* node.
  - *Such-as* relation is similar to inheritance, and it is connected with two *Class* nodes.

**Example 2. Parts of VRMS_SRO**

Figure 8 presents the utilities of three types of nodes and five types of relationships in VRMS_SRO, which is instantiated from SRO and used to maintain the knowledge structure of VRMS. Figure 8 (a) shows that VRMS is composed of major functionalities like "*MemberMgr*" class and "*VideoMgr*" class. Figure 8 (b) shows that "*MemberMgr*" class contains the interface such as "*Register*" method and "*Login*" method. Figure 8 (c) shows that "*Register*" method uses a data-storage object called “*MemberInfo*” which stores the information about member to handle data encapsulation. Figure 8 (d) shows that “*Register*” method uses a functional object called “*DBMgr*” which communicates with database to handle logic computation. And Figure 8 (e) shows that “*Profile*” class and “*Portfolio*” class inherit from “*MemberInfo*” class.

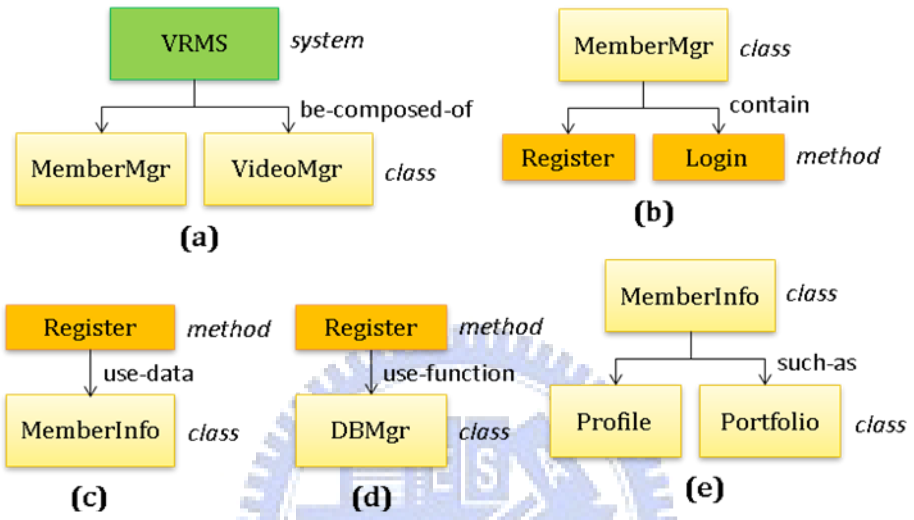


Figure 8. The examples of VRMS_SRO

### *4.1.2 Recursive Descent Dialog Approach (RDDA)*

After defining the SRO, we have to help teachers constructing this ontology. We propose the **Recursive Descent Dialog Approach (RDDA),** which is a systematic procedure to allow teachers to input relevant program scaffolds of VRMS. RDDA adopts a top-down dialogue mechanism built from a set of mutually-recursive procedures to construct SRO from major functionalities of VRMS to basic components by asking a series of questions about teacher’s design concepts of VRMS. The question templates described in Table 2 are defined based upon basic object-oriented design (OOD) process [5], which is shown in Figure 9. And the detailed algorithm of RDDA is shown in Algorithm 1.

Table 2. Question templates

| Node Type | Question Template |
|---|---|
| System | 請輸入這套系統的名稱和主要的功能需求 |
|  | 請輸入這套系統主要功能的類別名稱 |
| Class | 請問這個類別主要的功能需求以及Interface爲何 |
| Method | 請問這個方法的功能需求、會使用到的物件和使用的關係爲何 |
|  | 請問在這個方法所使用到的物件中，會使用到哪些方法 |

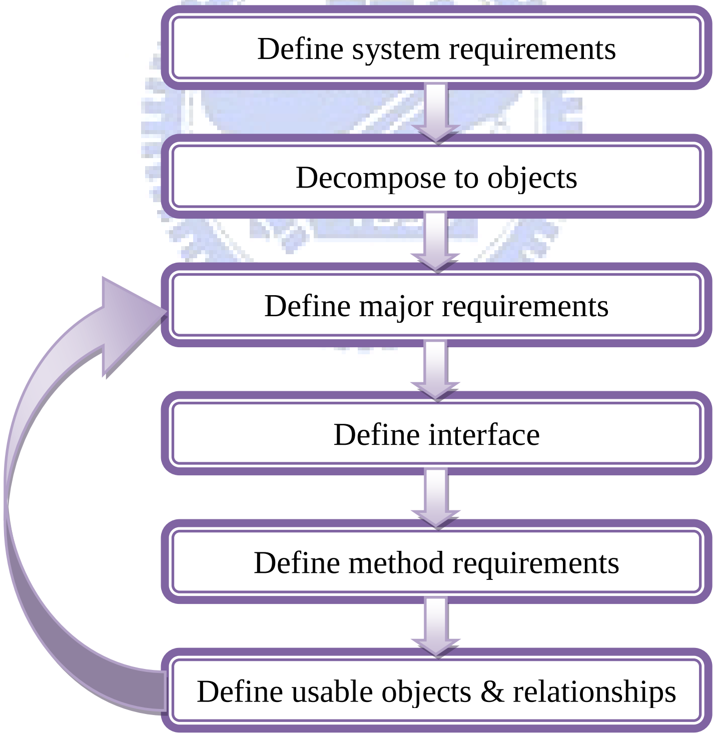

Define system requirements
Decompose to objects
Define major requirements
Define interface
Define method requirements
Define usable objects & relationships

Figure 9. Object-oriented design process

**Algorithm 1: Recursive Descent Dialog Approach (RDDA)**

**Denotation :**

**S :** The stack which is used for Depth-First Search (DFS)

**Input:** The program scaffolds of teaching case, Question Templates (QT)

**Output:** System Requirement Ontology (SRO)

```
While (There exists node $n_i$ which needs to be constructed for teacher)
{
    Push node $n_i$ into S.
    While (S is not empty)
    {
        Pop a node $n_i$ from S.
        Based on $n_i$ type, use the corresponding QT to compose the dialogue and
        ask all necessary information about node $n_i$ from teacher.
        Acquire the teacher's answer and construct a node
                $n_i = (nType_i, Name_i, Requirement_i, Interface_i, 0, UsableObject_i)$.
        Add $n_i$ to $N$.
        For each relationship $r_j \in$ { The output relationships of $n_i$ }
        {
            Based on $n_i$ type, use the corresponding QT to compose the dialogue
            and ask all necessary information about $r_j$ from teacher.
            Acquire the teacher's answer and construct a relationship
            $r_j = (rType_j, From_j, To_j)$.
            Add $r_j$ to $R$.
            Push $To_j$ into S.
        }
    }
Construct a $SRO = (N, R)$.
For each node $n_k \in N$
{
    According to the relationships and nodes of $SRO$ to compute the class quantity
            under node $n_k$, and then modify $ClassQuantity_k$.
}
Return $SRO$.
```

Algorithm 1. Recursive Descent Dialog Approach (RDDA)

**Example 3. RDDA Scaffolds Constructing System**

RDDA Scaffolds Constructing System adopts a series of dialogues to acquire the knowledge from the teacher and construct a SRO of teaching case. Figure 10 shows the example of VRMS_SRO construction process, where VRMS_SRO is an instance of SRO. Firstly, RDDA Scaffolds Constructing System asks the system name, system requirement and major functionalities from the teacher. And then according to major functionalities, where we use "*MemberMgr*" for example to ask the class requirement, class interface and their relationships. Next, based upon the interface of "*MemberMgr*", we choose a method "*Register*" for example to ask the method requirement, usable objects like "*MemberInfo*" and "*DBMgr*" and their relationships like "*use-data*" and "*use-function*", accordingly. Rules like that, at last RDDA Scaffolds Constructing System will request the teacher to upload the library package, source codes and class diagram. According to this constructing process, the teacher can construct a complete VRMS_SRO by RDDA Scaffolds Constructing System.

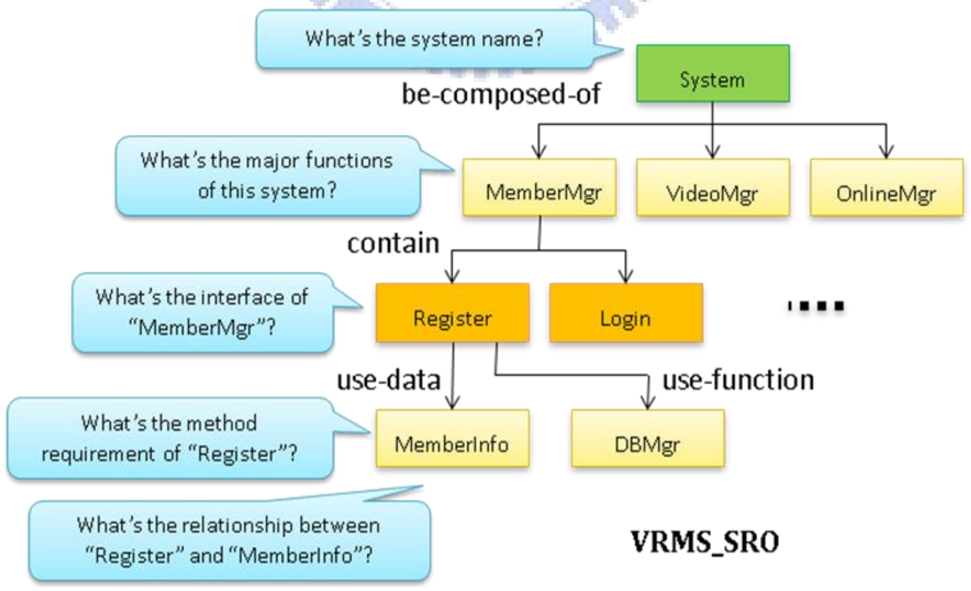

What's the system name?
System
be-composed-of
What's the major functions of this system?
MemberMgr
VideoMgr
OnlineMgr
contain
What's the interface of "MemberMgr"?
Register
Login
....
use-data
use-function
What's the method requirement of "Register"?
MemberInfo
DBMgr
What's the relationship between "Register" and "MemberInfo"?
VRMS_SRO

Figure 10. The example of VRMS_SRO construction process

## 4.2 Knowledge Acquisition Phase

### *4.2.1 Algorithm of Major-Requirement-First Strategy*

The algorithm of MRFS is shown in Algorithm 2 and represented by the flowchart in Figure 11. The main algorithm of MRFS is Depth-First-Search (DFS) with heuristic functions to determine the search path. DFS needs a stack to store temporal nodes, and a queue to store the imitating procedure as output. The heuristic functions are based on the class quantity under each node and the weight of each relationship type, where we define the weight of "*Use-function"* relationship is 2, the weight of "*Use-data"* relationship is 1, and the others are 0 to schedule the priority of imitation. We think that imitating the functional components first will easily comprehend what is supposed to do in next step, so we set "*Use-function*" with high priority than "*Use-data*". If the relationship types are equal, we will compare the class quantity under the node, where the larger quantity possesses the higher priority.

**Algorithm 2: Major-Requirement-First Strategy (MRFS)**

**Denotation :**

**S :** The stack which is used for Depth-First Search (DFS)

**A :** The array which is used for heuristic computation

**Q :** The queue which stores the imitating procedure

**Input:** System Requirement Ontology (SRO)

**Output:** Q (Imitating Procedure)

```
If (There exists a root node $n_i$ of SRO)
{
    Push $n_i$ to Q.
    While ($n_i$ contains any output relationship)
    {
        Do
        {
            For each relationship $r_j \in$ { The output relationships of $n_i$ }
            {
                Insert the relationship $r_j$ into A.
                Compare the $rType_j$ with the type of relationships in A, and sort these
                        relationships according to relationship type from max to min.
                        ("Use-function" = 2, "Use-data" = 1, others = 0)
                If (A contains relationships r[] whose type is equal to $rType_j$)
                    Compare $ClassQuantity_j$ with the class quantity of nodes which
                            are terminal nodes of relationships in r[], and sort these
                            relationships according to class quantity from max to min.
            }
            For each relationship $r_j$ in A
                Push $To_j$ into S.
        }While (S is not empty)
        Pop one node $n_k$ from S.
        Get a node $n_i = n_k$.
        Push $n_k$ to Q.
    }
}
Return Q.
```

Algorithm 2. Major-Requirement-First Strategy (MRFS)

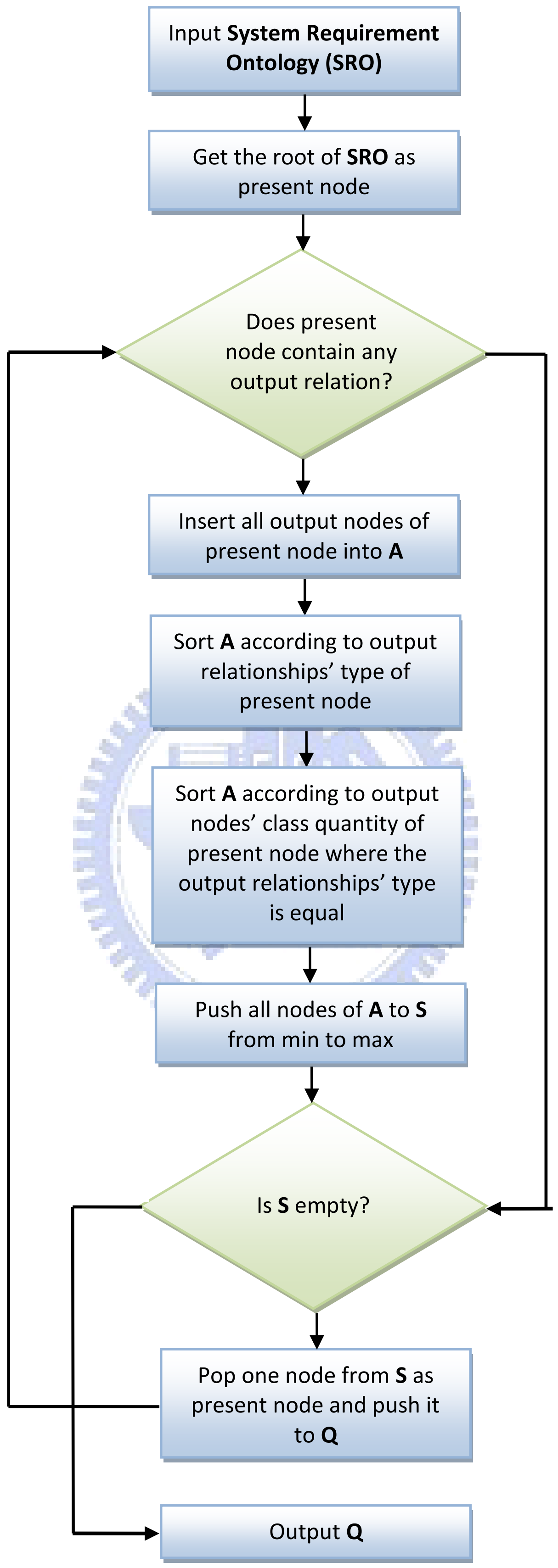

Input **System Requirement Ontology (SRO)**
Get the root of **SRO** as present node
Does present node contain any output relation?
Insert all output nodes of present node into **A**
Sort **A** according to output relationships' type of present node
Sort **A** according to output nodes' class quantity of present node where the output relationships' type is equal
Push all nodes of **A** to **S** from min to max
Is **S** empty?
Pop one node from **S** as present node and push it to **Q**
Output **Q**

Figure 11. The flowchart of Major-Requirement-First Strategy (MRFS)

### *4.2.2 Major-Requirement-First Guidance Transformation System (MRFGTS)*

In knowledge transformation phase, MRFGTS is composed of *MRFS Scheduler* and *OOLA Transformer*. *MRFS Scheduler* uses the SRO which constructed by *RDDA Scaffolding Constructing System* to apply MRFS to schedule an imitating procedure, and then *OOLA Transformer* transform this imitating procedure into a learning activity of *OOLA system* [14]. The internal procedure of MRFGTS is shown in Figure 12.

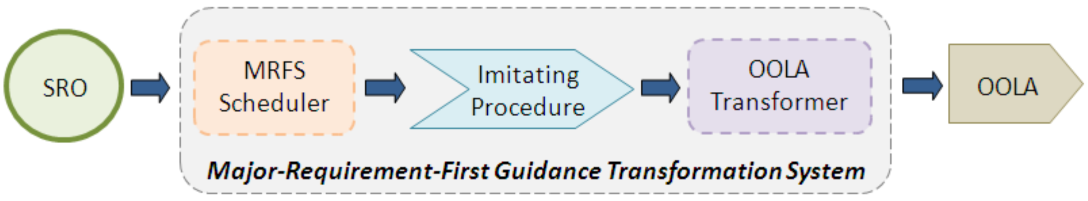


Figure 12. Major-Requirement-First Guidance Transformation System

**Example 4. MRFS Scheduler**

Figure 13 presents the example of MRFS Scheduler. Based upon the MRFS algorithm, we get the "*VRMS*" node which is the root of VRMS_SRO at first and push it into the imitating procedure. And then we check the output nodes and output relationships of "*VRMS*" node, where the three major functional nodes "*MemberMgr*", "*VideoMgr*" and "*OnlineMgr*" contain the same relationships "*be-composed-of*". So we will check the class quantity of these output nodes, and find out the largest one. Suppose that "*MemberMgr*" node has the largest class quantity of the three output nodes, we push it into the learning precedure, and check the output nodes and relationships again. Suppose there exists "*Register*" and "*Login*" method nodes and the relationships are as same as "*Contain*" relationship, and the "*Register*" node has large class quantity than "*Login*" node, so we push the "*Register*" node into the imitating procedure. Accordingly, there are two objects will be used in the "*Register*" method, so we choose the "*use-function*" relationship which has high priority than "*use-data*" relationship and push "*DBMgr*" node into the imitating procedure. Rules like that, then we will push "*InsertProfile*", "*InsertPortfolio*", "*MemberInfo*", "*Profile*", "*Portfolio*" nodes and so on into the learning queue sequentially based on MRFS.

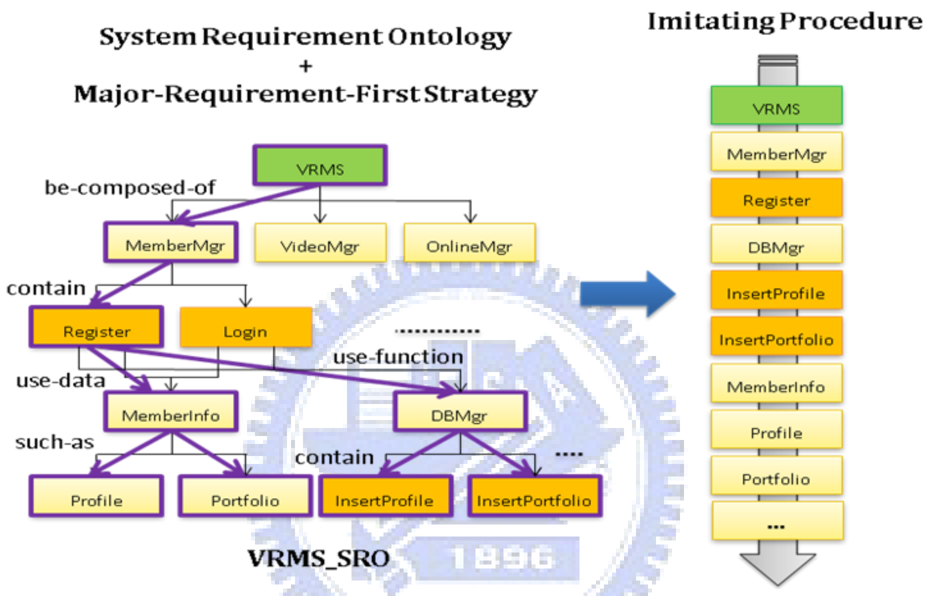


Figure 13. The example of MRFS Scheduler

**Example 5. OOLA Transformer**

In order to use OOLA System to present the learning contents, we need to transform the imitating procedure into an Object-Oriented Learning Activity (OOLA) in advance. As shown in Figure 14, after getting the imitating procedure from MRFS Scheduler, we arrange different *learning guidance templates* which are presented in Table 3 according to the type of nodes. The imitating procedure will attach different learning contents and learning resources which are constructed in knowledge acquisition phase to create the learning units, and then transform the imitating procedure into OOLA rules, which are used to arrange the learning sequence.

Afterwards, OOLA Transformer can generate an OOLA based on learning units and OOLA rules.

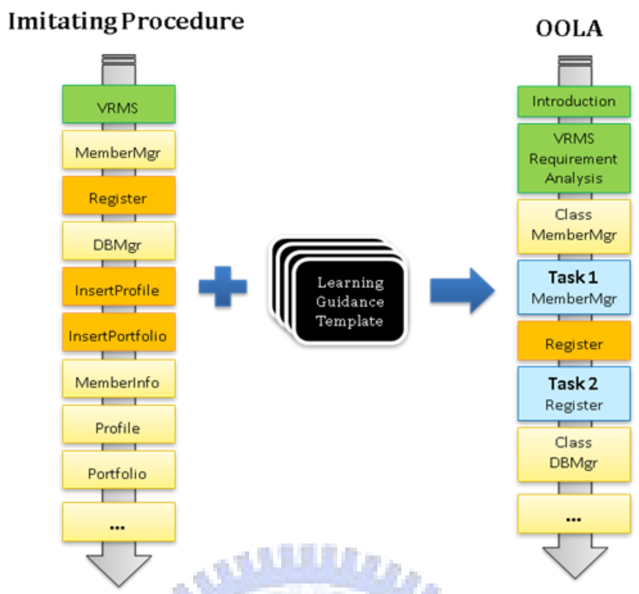


Figure 14. The example of OOLA Transformer

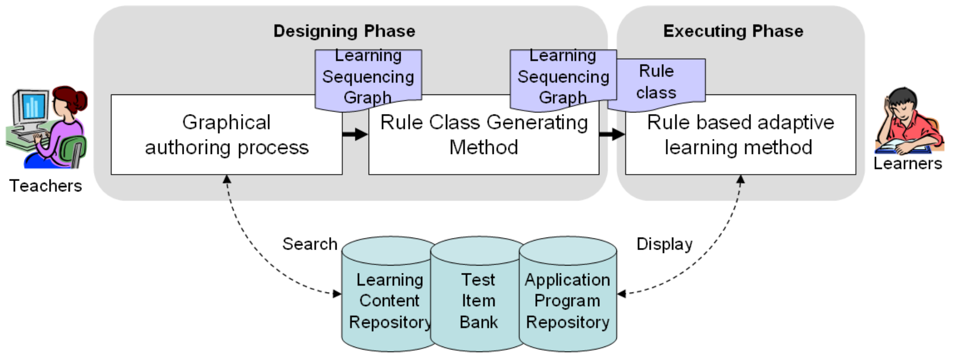
Table 3. Learning guidance templates

| Node Type | Learning Guidance Template | |
|---|---|---|
| System | Learning Content | Introduction, system requirement analysis, major functionality requirements, and guiding information. |
| | Learning Resource | Class diagram. |
| Class | Learning Content | Class requirement, interface, method requirements, impending method tasks, complete method list, guiding information, and task information. |
| | Learning Resource | Class diagram, source code, library package. |
| Method | Learning Content | Method requirement, usable object interface, usable object requirements, guiding information, and task information. |
| | Learning Resource | Class diagram, source code, library package. |

## 4.3 Learning Phase

### *4.3.1 Object-Oriented Learning Activity (OOLA) System*

**Object-Oriented Learning Activity (OOLA) System** [14] is an expert system with a graphical authoring tool, which can assist teachers to design adaptive learning activity, and a web based learning system, which can steer learners learning using a rule inference engine. The teacher can arrange learning sequence, edit learning contents and manage the learning resources to provide a learning activity to learners.



Designing Phase
Executing Phase
Learning Sequencing Graph
Learning Sequencing Graph
Rule class
Teachers
Graphical authoring process
Rule Class Generating Method
Rule based adaptive learning method
Learners
Search
Display
Learning Content Repository
Test Item Bank
Application Program Repository

Figure 15. The system architecture of OOLA System

Figure 15 shows designing phase and executing phase in OOLA System. In the designing phase, an adaptive learning activity can be designed via the graphical authoring process to teachers easily integrating various learning resources, which are provided by learning resource repositories, and a rule class generating method is also proposed to generate the rule class of learning sequencing controls. In the executing phase, the learning sequencing graph and the rule class are used in rule based adaptive learning method to conduct an adaptive learning to learners.

Therefore, in the learning phase of MRFILS, an OOLA which is already attached necessary learning contents and learning resources will guide the learners to learn the software architecture design.

# Chapter 5. System Implementation and Experiment

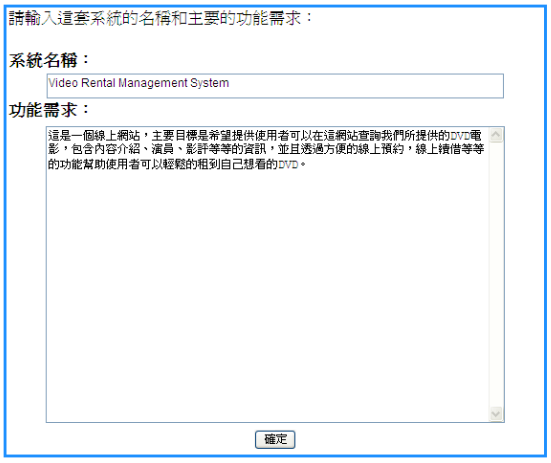
## 5.1 System Implementation

We have implemented a Major-Requirement-First Imitating Learning System (MRFILS) on web-based environment to evaluate the proposed ideas. The following sections will demonstrate the *Teaching Materials Construction Phase* and the *Software Architecture Design Learning Phase*.

### *5.1.1 Teaching Materials Construction Phase*

This phase we use a dialogue mechanism to acquire teachers' design knowledge of Video Renal Management System (VRMS). MRFILS will ask teachers a series of questions and request them to input the program scaffolds by a systematic process.

As shown in Figure 16, these pictures present the construction process for the teaching materials of VRMS. At first, the teacher is requested to input the system name, system requirement (see Figure 16 (a)), and names of major functionalities (see Figure 16 (b)). Based upon one of major functionalities, MRFILS continues to ask this class requirement, interfaces and relationships (see Figure 16 (c)). Next, choosing one method among the interfaces to ask the method requirement, usable objects, relationships (see Figure 16 (d)), and usable methods (see Figure 16 (e)). And then keeping on asking the questions about the class requirement, interfaces and relationships of usable objects (see Figure 16 (f)). After a series of questions, the teacher is demanded to commit the source codes, and MRFILS will compile them to library packages automatically.


請輸入這套系統的名稱和主要的功能需求：

**系統名稱：**

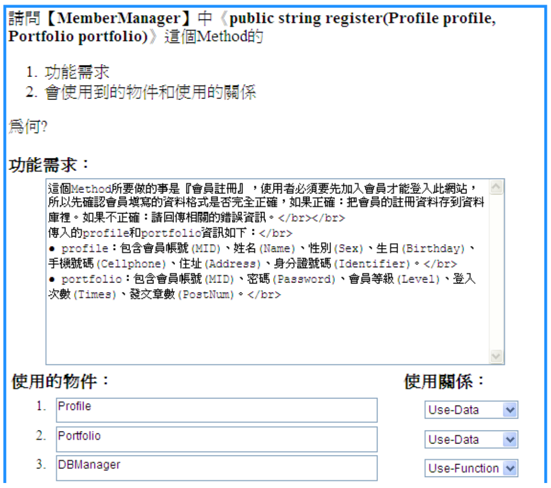
Video Rental Management System

**功能需求：**

這是一個線上網站，主要目標是希望提供使用者可以在這網站查詢我們所提供的DVD電影，包含內容介紹、演員、影評等等的資訊，並且透過方便的線上預約，線上續借等等的功能幫助使用者可以輕鬆的租到自己想看的DVD。

確定

(a)


請問【**MemberManager**】中《**public string register(Profile profile, Portfolio portfolio)**》這個Method的

1. 功能需求
2. 會使用到的物件和使用的關係

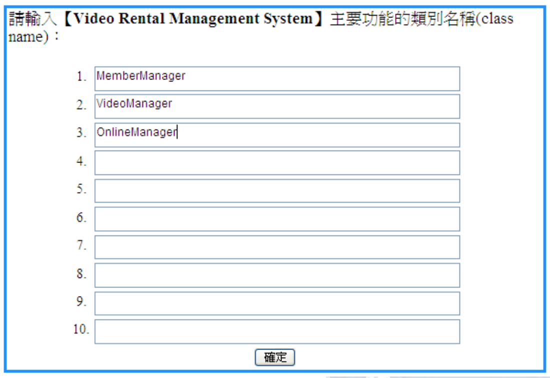
爲何?

**功能需求：**

這個Method所要做的事是『會員註冊』，使用者必須要先加入會員才能登入此網站，所以先確認會員填寫的資料格式是否完全正確，如果正確：把會員的註冊資料存到資料庫裡。如果不正確：請回傳相關的錯誤資訊。</br></br>
傳入的profile和portfolio資訊如下：</br>
● profile：包含會員帳號(MID)、姓名(Name)、性別(Sex)、生日(Birthday)、手機號碼(Cellphone)、住址(Address)、身分證號碼(Identifier)。</br>
● portfolio：包含會員帳號(MID)、密碼(Password)、會員等級(Level)、登入次數(Times)、發文章數(PostNum)。</br>

| | **使用的物件：** | **使用關係：** |
|---|---|---|
| 1. | Profile | Use-Data |
| 2. | Portfolio | Use-Data |
| 3. | DBManager | Use-Function |

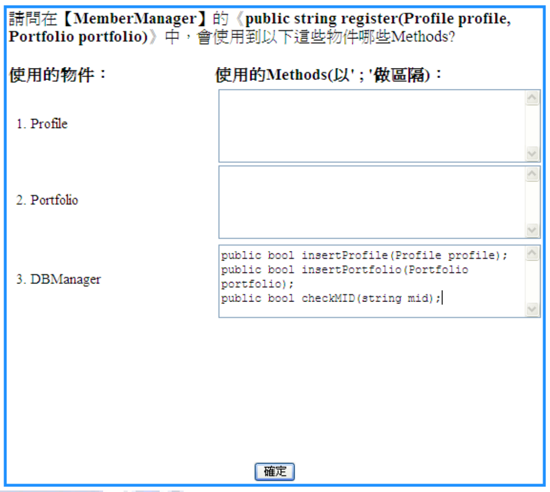
(d)


請輸入【**Video Rental Management System**】主要功能的類別名稱(class name)：

1. MemberManager
2. VideoManager
3. OnlineManager
4. 
5. 
6. 
7. 
8. 
9. 
10.

確定

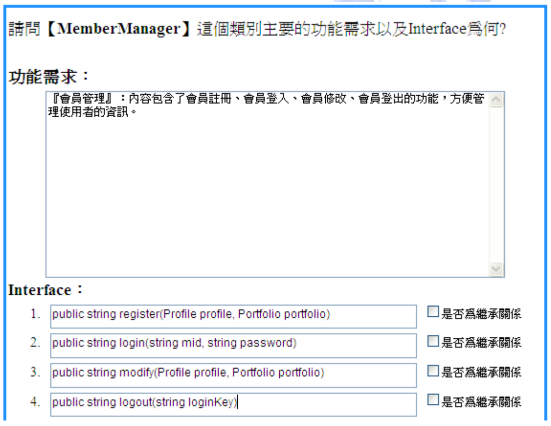
(b)


請問在【**MemberManager**】的《**public string register(Profile profile, Portfolio portfolio)**》中，會使用到以下這些物件哪些Methods?

| **使用的物件：** | **使用的Methods(以' ; '做區隔)：** |
|---|---|
| 1. Profile | |
| 2. Portfolio | |
| 3. DBManager | public bool insertProfile(Profile profile);<br>public bool insertPortfolio(Portfolio portfolio);<br>public bool checkMID(string mid); |

確定

(e)


請問【**MemberManager**】這個類別主要的功能需求以及Interface爲何?

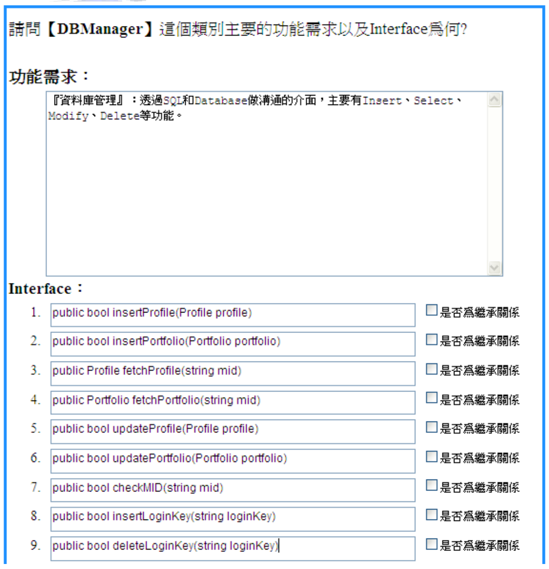
**功能需求：**

『會員管理』：內容包含了會員註冊、會員登入、會員修改、會員登出的功能，方便管理使用者的資訊。

**Interface：**

| | | |
|---|---|---|
| 1. | public string register(Profile profile, Portfolio portfolio) | ☐是否爲繼承關係 |
| 2. | public string login(string mid, string password) | ☐是否爲繼承關係 |
| 3. | public string modify(Profile profile, Portfolio portfolio) | ☐是否爲繼承關係 |
| 4. | public string logout(string loginKey) | ☐是否爲繼承關係 |

(c)


請問【**DBManager**】這個類別主要的功能需求以及Interface爲何?

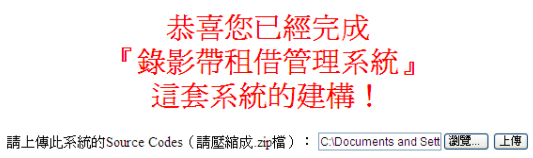
**功能需求：**

『資料庫管理』：透過SQL和Database做溝通的介面，主要有Insert、Select、Modify、Delete等功能。

**Interface：**

| | | |
|---|---|---|
| 1. | public bool insertProfile(Profile profile) | ☐是否爲繼承關係 |
| 2. | public bool insertPortfolio(Portfolio portfolio) | ☐是否爲繼承關係 |
| 3. | public Profile fetchProfile(string mid) | ☐是否爲繼承關係 |
| 4. | public Portfolio fetchPortfolio(string mid) | ☐是否爲繼承關係 |
| 5. | public bool updateProfile(Profile profile) | ☐是否爲繼承關係 |
| 6. | public bool updatePortfolio(Portfolio portfolio) | ☐是否爲繼承關係 |
| 7. | public bool checkMID(string mid) | ☐是否爲繼承關係 |
| 8. | public bool insertLoginKey(string loginKey) | ☐是否爲繼承關係 |
| 9. | public bool deleteLoginKey(string loginKey) | ☐是否爲繼承關係 |

(f)


恭喜您已經完成
『錄影帶租借管理系統』
這套系統的建構！

請上傳此系統的Source Codes（請壓縮成.zip檔）： C:\Documents and Sett 瀏覽... 上傳

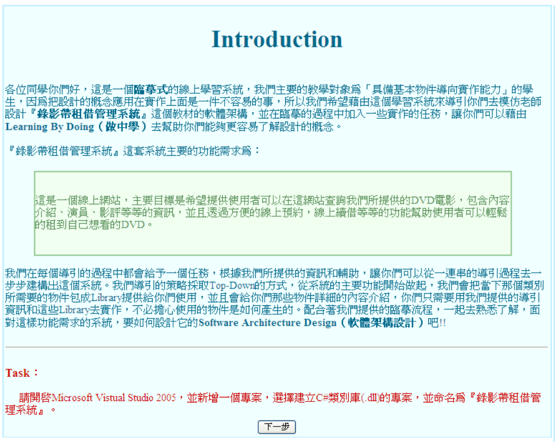
(g)

Figure 16. The snapshot of teaching materials construction phase

## *5.1.2 Software Architecture Design Learning Phase*

This phase we use Object-Oriented Learning Activity (OOLA) System to present the learning target "VRMS", which is constructed by teacher in Teaching Materials Construction Phase. OOLA System will provide necessary learning contents and learning resources for learners and guide them to design the system architecture by imitating the compositions of each component and component's design.

Figure 17 shows the learning process, where OOLA System offers guiding information to instruct the learners why and how the component design is in each step. First of all, the learning system provides the introduction about this imitating learning approach and a task to create a project (see Figure 17 (a)). And then the learners can get the system requirement analysis (see Figure 17 (b)), one of major functional classes introduction with a task and a library download link for learners to start implementing this class with library package supporting (see Figure 17 (c)). After that, the OOLA System provides the method information of this class and a task (see Figure 17 (d)) for learners to complete this method. Next, it continues to provide the class information which was used by previous method and a library package for task implementation (see Figure 17 (e)). Such as this top-down imitating learning procedure, the learners can gain these software architecture design concepts through a sequence of implementations.



## Introduction

各位同學你們好，這是一個**臨摹式**的線上學習系統，我們主要的教學對象爲「具備基本物件導向實作能力」的學生，因爲把設計的概念應用在實作上面是一件不容易的事，所以我們希望藉由這個學習系統來導引你們去模仿老師設計**『錄影帶租借管理系統』**這個教材的軟體架構，並在臨摹的過程中加入一些實作的任務，讓你們可以藉由**Learning By Doing（做中學）**去幫助你們能夠更容易了解設計的概念。

『錄影帶租借管理系統』這套系統主要的功能需求爲：

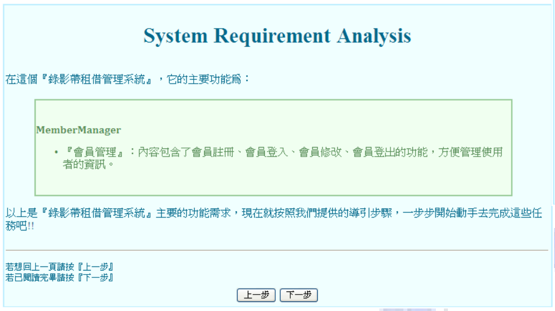
這是一個線上網站，主要目標是希望提供使用者可以在這網站查詢我們所提供的DVD電影，包含內容介紹、演員、影評等等的資訊，並且透過方便的線上預約，線上續借等等的功能幫助使用者可以輕鬆的租到自己想看的DVD。

我們在每個導引的過程中都會給予一個任務，根據我們所提供的資訊和輔助，讓你們可以從一連串的導引過程去一步步建構出這個系統。我們導引的策略採取Top-Down的方式，從系統的主要功能開始做起，我們會把當下那個類別所需要的物件包成Library提供給你們使用，並且會給你們那些物件詳細的內容介紹，你們只需要用我們提供的導引資訊和這些Library去實作，不必擔心使用的物件是如何產生的。配合著我們提供的臨摹流程，一起去熟悉了解，面對這樣功能需求的系統，要如何設計它的**Software Architecture Design（軟體架構設計）**吧!!

**Task：**

請開啓Microsoft Vistual Studio 2005，並新增一個專案，選擇建立C#類別庫(.dll)的專案，並命名爲『錄影帶租借管理系統』。

(a)



## System Requirement Analysis

在這個『錄影帶租借管理系統』，它的主要功能爲：

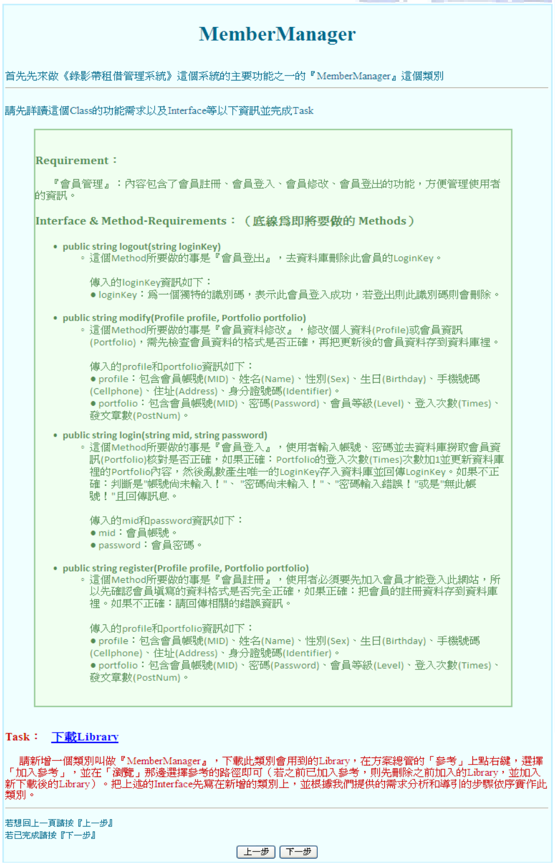
**MemberManager**

- 『會員管理』：內容包含了會員註冊、會員登入、會員修改、會員登出的功能，方便管理使用者的資訊。

以上是『錄影帶租借管理系統』主要的功能需求，現在就按照我們提供的導引步驟，一步步開始動手去完成這些任務吧!!

若想回上一頁請按『上一步』
若已閱讀完畢請按『下一步』

(b)



## MemberManager

首先先來做《錄影帶租借管理系統》這個系統的主要功能之一的『MemberManager』這個類別

請先詳讀這個Class的功能需求以及Interface等以下資訊並完成Task

**Requirement：**

『會員管理』：內容包含了會員註冊、會員登入、會員修改、會員登出的功能，方便管理使用者的資訊。

**Interface & Method-Requirements：（底線爲即將要做的 Methods）**

- **public string logout(string loginKey)**
  - 這個Method所要做的事是『會員登出』，去資料庫刪除此會員的LoginKey。

    傳入的loginKey資訊如下：
    ● loginKey：爲一個獨特的識別碼，表示此會員登入成功，若登出則此識別碼則會刪除。
- **public string modify(Profile profile, Portfolio portfolio)**
  - 這個Method所要做的事是『會員資料修改』，修改個人資料(Profile)或會員資訊(Portfolio)，需先檢查會員資料的格式是否正確，再把更新後的會員資料存到資料庫裡。

    傳入的profile和portfolio資訊如下：
    ● profile：包含會員帳號(MID)、姓名(Name)、性別(Sex)、生日(Birthday)、手機號碼(Cellphone)、住址(Address)、身分證號碼(Identifier)。
    ● portfolio：包含會員帳號(MID)、密碼(Password)、會員等級(Level)、登入次數(Times)、發文章數(PostNum)。
- **public string login(string mid, string password)**
  - 這個Method所要做的事是『會員登入』，使用者輸入帳號、密碼並去資料庫撈取會員資訊(Portfolio)核對是否正確，如果正確：Portfolio的登入次數(Times)次數加1並更新資料庫裡的Portfolio內容，然後亂數產生唯一的LoginKey存入資料庫並回傳LoginKey。如果不正確：判斷是"帳號尚未輸入！"、"密碼尚未輸入！"、"密碼輸入錯誤！"或是"無此帳號！"且回傳訊息。

    傳入的mid和password資訊如下：
    ● mid：會員帳號。
    ● password：會員密碼。
- **public string register(Profile profile, Portfolio portfolio)**
  - 這個Method所要做的事是『會員註冊』，使用者必須要先加入會員才能登入此網站，所以先確認會員填寫的資料格式是否完全正確，如果正確：把會員的註冊資料存到資料庫裡。如果不正確：請回傳相關的錯誤資訊。

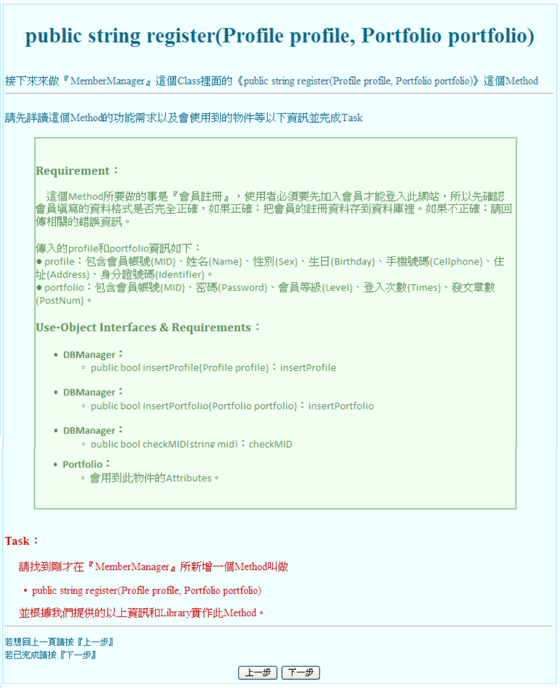
    傳入的profile和portfolio資訊如下：
    ● profile：包含會員帳號(MID)、姓名(Name)、性別(Sex)、生日(Birthday)、手機號碼(Cellphone)、住址(Address)、身分證號碼(Identifier)。
    ● portfolio：包含會員帳號(MID)、密碼(Password)、會員等級(Level)、登入次數(Times)、發文章數(PostNum)。

**Task：** 下載Library

請新增一個類別叫做『MemberManager』，下載此類別會用到的Library，在方案總管的「參考」上點右鍵，選擇「加入參考」，並在「瀏覽」那邊選擇參考的路徑即可（若之前已加入參考，則先刪除之前加入的Library，並加入新下載後的Library）。把上述的Interface先寫在新增的類別上，並根據我們提供的需求分析和導引的步驟依序實作此類別。

若想回上一頁請按『上一步』
若已完成請按『下一步』

(c)



## public string register(Profile profile, Portfolio portfolio)

接下來來做『MemberManager』這個Class裡面的《public string register(Profile profile, Portfolio portfolio)》這個Method

請先詳讀這個Method的功能需求以及會使用到的物件等以下資訊並完成Task

**Requirement：**

這個Method所要做的事是『會員註冊』，使用者必須要先加入會員才能登入此網站，所以先確認會員填寫的資料格式是否完全正確，如果正確：把會員的註冊資料存到資料庫裡。如果不正確：請回傳相關的錯誤資訊。

傳入的profile和portfolio資訊如下：
● profile：包含會員帳號(MID)、姓名(Name)、性別(Sex)、生日(Birthday)、手機號碼(Cellphone)、住址(Address)、身分證號碼(Identifier)。
● portfolio：包含會員帳號(MID)、密碼(Password)、會員等級(Level)、登入次數(Times)、發文章數(PostNum)。

**Use-Object Interfaces & Requirements：**

- **DBManager：**
  - public bool insertProfile(Profile profile)：insertProfile
- **DBManager：**
  - public bool insertPortfolio(Portfolio portfolio)：insertPortfolio
- **DBManager：**
  - public bool checkMID(string mid)：checkMID
- **Portfolio：**
  - 會用到此物件的Attributes。

**Task：**

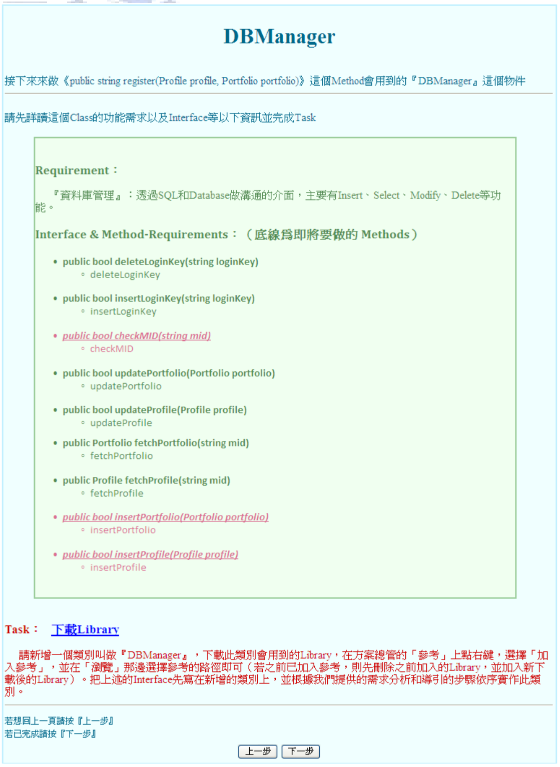
請找到剛才在『MemberManager』所新增一個Method叫做

- public string register(Profile profile, Portfolio portfolio)

並根據我們提供的以上資訊和Library實作此Method。

若想回上一頁請按『上一步』
若已完成請按『下一步』

(d)



## DBManager

接下來來做《public string register(Profile profile, Portfolio portfolio)》這個Method會用到的『DBManager』這個物件

請先詳讀這個Class的功能需求以及Interface等以下資訊並完成Task

**Requirement：**

『資料庫管理』：透過SQL和Database做溝通的介面，主要有Insert、Select、Modify、Delete等功能。

**Interface & Method-Requirements：（底線爲即將要做的 Methods）**

- **public bool deleteLoginKey(string loginKey)**
  - deleteLoginKey
- **public bool insertLoginKey(string loginKey)**
  - insertLoginKey
- ***public bool checkMID(string mid)***
  - checkMID
- **public bool updatePortfolio(Portfolio portfolio)**
  - updatePortfolio
- **public bool updateProfile(Profile profile)**
  - updateProfile
- **public Portfolio fetchPortfolio(string mid)**
  - fetchPortfolio
- **public Profile fetchProfile(string mid)**
  - fetchProfile
- ***public bool insertPortfolio(Portfolio portfolio)***
  - insertPortfolio
- ***public bool insertProfile(Profile profile)***
  - insertProfile

**Task：** 下載Library

請新增一個類別叫做『DBManager』，下載此類別會用到的Library，在方案總管的「參考」上點右鍵，選擇「加入參考」，並在「瀏覽」那邊選擇參考的路徑即可（若之前已加入參考，則先刪除之前加入的Library，並加入新下載後的Library）。把上述的Interface先寫在新增的類別上，並根據我們提供的需求分析和導引的步驟依序實作此類別。

若想回上一頁請按『上一步』
若已完成請按『下一步』

(e)

Figure 17. The snapshot of software architecture design learning phase

## 5.2 Experiment

To evaluate the efficacy of Major-Requirement-First Strategy (MRFS) in software architecture design learning, we choose 10 graduate students who have enough coding ability of object-oriented programming (C#) from Computer Science in National Chiao Tung University. We design the Video Rental Management System (VRMS) as the teaching case and use Major-Requirement-First Imitating Learning System (MRFILS) to teach software architecture design. After a series of guidance by MRFILS, we use a questionnaire which is shown in Table 4 to evaluate the learning efficacy and satisfaction of learners. We apply the five-level Likert Scale which is the most widely used scale in survey research and it rates from 1 to 5 to specify the respondent's agreement level to a statement. The five-level Likert degree is shown in Table 5 and the results for learning efficacy and satisfaction is shown in Figure 18.

Table 4. The questionnaire items

| Questionnaire Items |
|---|
| Q1. Is the guiding procedure useful to construct the VRMS? |
| Q2. Is the information provided in each implementation process sufficient to construct the task? |
| Q3. Do you think that the library provided in each implementation process is useful to complete the task? |
| Q4. Do you think that using the library first and then construct these objects later will easily comprehend what should be considered in these objects? |
| Q5. Do you think that this top-down imitating manner will easily understand why and how the software architecture design is than traditional bottom-up implementation? |
| Q6. Do you think that the learning efficacy of this system is better than other learning approaches such as lectures or case study? |
| Q7. Please give a grade (1~5) for this learning system. |

Table 5. The format of a typical five-level Likert degree

| Degree | Meaning |
|---|---|
| 1 | Strongly disagree |
| 2 | Disagree |
| 3 | Neither agree nor disagree |
| 4 | Agree |
| 5 | Strongly agree |

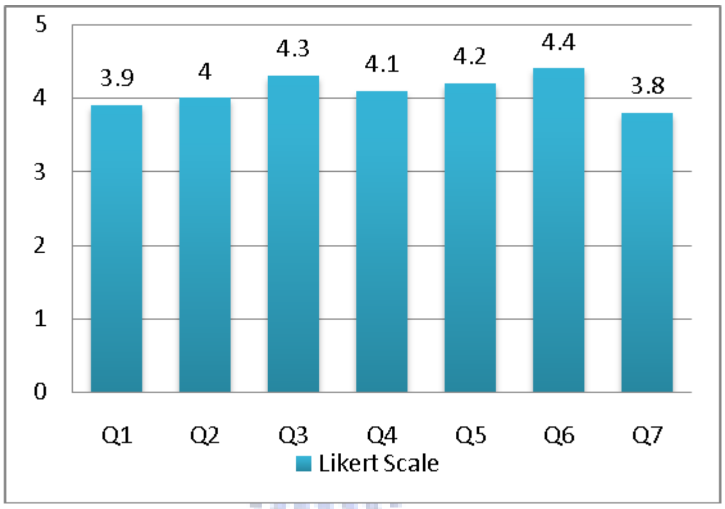


Figure 18. The results for learning efficacy and satisfaction

The results show that most of learners think this top-down imitating learning can help them understanding the software architecture design effectively than bottom-up implementation and traditional learning approaches like lectures or case study. And implement the programs with library packages supporting will easily comprehend how to design these used objects. Otherwise, some learners think the information of learning contents need to be more sufficient, and some learners don't agree with the guiding procedure where the system provided.

## 5.3 Discussion

Finally, we also interview the learners and ask the impressions for MRFILS, there are some comments on this system as below:

- **Major-Requirement-First Strategy (MRFS)**

  Several learners think that this top-down imitating manner is a good idea for learning system design. Because traditional implementation manner is bottom-up, even they have the system requirements, architectural blue print, and UML diagrams for reference, it is still difficult to understand how to implement the system for learners who just begin to learn the web-based system design.

- **Learning contents**

  Several learners think that we should provide more UML diagrams rather than only one Class Diagram, such as Use Case Diagram, Activity Diagram, and Sequence Diagram, etc., and point out the present status and what they have done. The learners believe that the information of UML diagrams will make themselves clearly understand the complete degree of system and the design procedure.

- **Guiding procedure**

  Because the guiding procedure is scheduled by MRFILS, where we design some heuristic algorithms to prioritize the imitating sequence, several learners think that we can provide more imitating sequences for them to choose rather than only one choice.

The above comments are very significant for improving this computer-assisted learning system, and we will take these comments into account in the future work.

# Chapter 6. Conclusion and Future Work

In this thesis, we propose a top-down imitating learning manner named Major-Requirement-First Strategy (MRFS) and apply this teaching approach to construct an intelligent tutoring system named Major-Requirement-First Imitating Learning System (MRFILS) to teach software architecture design of logic tier in web-based system. The teachers can use MRFILS to construct teaching materials, where we adopt a web-based system called "Video Rental Management System (VRMS)" as our teaching case. MRFILS provides learners one-on-one instruction and assistance to guide them imitating the software architecture design of VRMS through a series of implementations with scaffolding instruction, and it will significantly reduce the loading of teachers. Finally, we have done a real experiment to evaluate the learning efficacy and satisfaction of learners. The experimental results show that MRFILS can help learners gaining the software architecture design concepts of VRMS effectively.

In the near future, we would like to embed diagnostic tests in the guiding process to evaluate the learning performance and learning status, and add some imitating learning guidance rules to provide adaptive learning for learners. Thus, the learners can get more flexible guiding procedure and more appropriate learning contents in the learning process.

# References

1. Grady Booch, *Building Web Applications with UML*. 2000: Addison Wesley.
2. Grady Booch, *Visual Modeling With Rational Rose 2002 AND UML*. 2003: Addison Wesley.
3. Jeff Garland, Richard Anthony, *Large-Scale Software Architecture: A Practical Guide using UML*. 2003: Wiley.
4. David Garlan, Mary Shaw, *An Introduction to Software Architecture.* Advances in Software Engineering and Knowledge Engineering, January 1994.
5. Grady Booch, Robert A.Maksichuk, *Object-Oriented Analysis And Design With Applications*. 2007: Addison Wesley.
6. Woei-Kae Chen, Yu-Chin Cheng, *Teaching Object-Oriented Programming Laboratory With Computer Game Programming.* IEEE Transactions on Education, AUGUST 2007. **VOL.50, NO.3**.
7. Liang-Yi Li, Gwo-Dong Chen, *A Coursework Support System for Offering Challenges and Assistance by Analyzing Students' Web Portfolios.* Educational Technology & Society, 2009. **Vol. 12**: p. 205-221.
8. J. Baltasar Garcia Perez-Schofield, Emilio Garcia Rosello, Francisco Ortin Soler, Manuel Perez Cota, *Visual Zero: A persistent and interactive object-oriented programming environment.* Journal of Visual Languages and Computing 2008. **Vol. 19** p. 380-398.
9. Maria Kordaki, Micael Miatidis, George Kapsampelis, *A Computer Environment For Beginners' Learning of Sorting Algorithms: Design and pilot Evaluation.* Computers & Education, 2008. **Vol. 51**: p. 708-723.

10. Matthew Conway, Steve Audia, Tommy Burnette et al., *Alice: Lessons Learned From Building A 3D System For Novices*, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. April 2000: The Hague, The Netherlands. p. 486-493.

11. Stelios Xinogalos, Maya Satratzemi, Vassilios Dagdilelis, *An Introduction to Object-Oriented Programming with A Didactic Microworld: ObjectKarel.* Computers & Education, 2006. **Vol. 47**: p. 148-171.

12. Rachel Van Der Stuyf, November 2002: *Scaffolding As A Teaching Strategy*.

13. Lev Semenovich Vygotskiĭ, Michael Cole, Vera John-Steiner, Sylvia Scribner, *Mind in Society: The Development of Higher Psychological Processes*. 1978: Harvard University Press.

14. Huan-Yu Lin, Shian-Shyong Tseng, Jun-Ming Su, Jui-Feng Weng, *Design and Implementation of an Object Oriented Learning Activity System*, in *Proc. of the 10th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI)*. July, 2006: Orlando, Florida, USA.

15. *Computer Science (Catalog).* 2006; Available from: http://beloit.edu/~huss/cs/CS%20Catalog.html.

16. Jungwoo Ryoo, Frederico Fonseca, David S.Janzen, *Teaching Object-Oriented Software Engineering through Problem-Based Learning in the Context of Game Design*, in *21st Conference on Software Engineering Education and Training, IEEE Computer Society*. 2008.

17. Michael Lang, Brian Fitzgerald, *Web-based Systems Design: A Study of Contemporary Practices and an Explanatory Framework Based on "Method-in-Action"*. Requirements Engineering, 2007. **Vol. 12**.

18. Robert Orfali, Dan Harkey, *Client/Server Programming with Java and CORBA*. 2nd ed. 1998: John Wiley & Sons, Inc.

19. Seung C. Lee, Ashraf I. Shirani, *A Component Based Methodology for Web Application Development.* Systems and Software, 2004. **Vol. 71**.

20. Dimitrios A. Kateros, Georgia M. Kapitsaki, Nikolaos D. Tselikas and Iakovos S. Venieris, *A Methodology for Model-Driven Web Application Composition*, in *IEEE International Conference on Services Computing*. 2008.

21. Lab, T.L.K.R.G.a.t.M.M. *SCRATCH*. Available from: http://llk.media.mit.edu.

22. Jaime Galvez, Eduardo Guzman, Ricardo Conejo, *A Blended E-Learning Experience in A Course of Object Oriented Programming Fundamentals.* Knowledge-Based Systems, 2009.

23. Claus Pahl, Claire Kenny, *Interactive Correction and Recommendation for Computer Language Learning and Training.* IEEE Transactions on Education and Data Engineering, June 2009. **Vol. 21**.

24. Zoran Jeremić, Jelena Jovanović, Dragan Gašević, *Evaluating an Intelligent Tutoring System for Design Patterns: the DEPTHS Experience.* Educational Technology & Society, 2009. **Vol. 12**: p. 111-130.

25. L. Briand, C. Bunse, and J. Daly, *A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs.* IEEE Transactions on Software Engineering, 2001. **Vol. 27**: p. 513-530.