

基於模型驅動架構之系統開發研究—  
以 PHP 語言建置集中採購網站為例  
System Development Based on Model-Driven  
Architecture—An Example of Constructing A  
Centralized Purchasing Website with PHP



研究生：黃連豐 (Lan-Fon Huang)

指導教授：王永心 (Prof. Yung-Hsin Wang)

大同大學

資訊經營研究所

碩士論文

Thesis for Master of Business Administration  
Department of Information Management  
Tatung University

中華民國九十七年六月

June 2008

大同大學  
資訊經營研究所  
碩士學位論文

基於模型驅動架構之系統開發研究—  
以 PHP 語言建置集中採購網站為例

黃連豐

經考試合格特此證明

論文考試委員

指導教授

李光漢  
陳宗天  
王永心

王永心  
所長  
高南成

中華民國 97 年 5 月 2 日

## 致謝

本論文得以完成，首先必須先感謝我的指導教授王永生博士的指導，在我的論文寫作過程中，適時提出指正與引導，使研究內容有具體成果。

其次，也非常感謝口試委員：李興漢老師、陳宗天老師，給予本論文很多寶貴意見及指導。承蒙師長們在論文口試期間所提供的見解與指正，使得本論文得以更加嚴謹、完善，在此由衷感謝。

感謝黃永立同學熱心提供論文撰寫經驗，還有孫國倫、郭大華等同窗同學及學妹在就學期間的互相鼓勵與扶持。

最後，我要感謝我的家人，在我就讀大同的這段期間，給我最大的支持與鼓勵，謝謝你們。



## 摘要

本論文針對以模型驅動架構 (Model-Driven Architecture, MDA) 為基礎的系統開發作一研究探討。透過實作的過程驗證 MDA 類別之轉換—PIM (Platform Independent Model)、PSM (Platform Specific Model)與 CODE 之間轉換的規則與關係，並針對 PIM 到 PSM 的轉換過程，整理提出一個快速將統一塑模語言 (Unified Modeling Language, UML) 類別圖轉換為 PHP(Hypertext Preprocessor)程式樣版的方法與工具，節省人工轉換過程中需要了解 MDA 架構的時間，以及避免人為轉換的錯誤並且確保程式碼的品質。

在 UML 九種圖形之中，我們以最常用的兩種圖表類別圖 (Class Diagram) 與循序圖 (Sequence Diagram) 來作為系統分析時使用之圖表。系統分析之後遵循並落實 MDA 架構，將 UML 所產生的類別圖轉換為與 PHP 程式語言相關的程式樣版，並進一步的將程式樣版轉換為實際可執行的程式碼。本篇論文亦以集中採購網站建置為例來展示此系統開發之方法實作。

**關鍵字：**MDA、UML、PIM、PSM、PHP

# Abstract

This thesis aims at a study of system development based on the model-driven architecture (MDA). The MDA To utilizing the unified modeling language (UML) as an analytical tool for the system, comply and implement the MDA architecture, and then transform the Class Diagram which is created by the UML into the template related to PHP programming language. Furthermore, the said template would be transformed into the physically executable codes, and finally, the development method of this system will be utilized to construct a centralized purchasing website in an applicable implementation. Additionally, while referring to the transforming process of PIM-to-PSM, we address the relevant method and tools applicable to rapidly transform the UML Class Diagram into PHP template so as to dramatically save the time necessary for thoroughly understanding such MDA architecture during the artificial transformation process as well as to avoid the likely human errors.



# 目錄

致謝.....	II
摘要.....	III
ABSTRACT.....	IV
圖目錄 .....	VII
表目錄 .....	VIII
第一章 緒論 .....	1
1.1 研究背景.....	1
1.3 研究流程.....	3
1.4 論文架構.....	4
第二章 文獻探討 .....	5
2.1 統一塑模語言UML (UNIFIED MODELING LANGUAGE).....	5
2.1.1 UML的發展 .....	5
2.1.2 UML的定義及圖型分類 .....	6
2.2 模型驅動架構MDA (MODEL DRIVEN ARCHITECTURE).....	10
2.2.1. MDA介紹 .....	10
2.2.2. MDA之類別及轉換 .....	11
2.2.3. MDA程式的轉換工具 .....	14
2.2.4. MDA的優點 .....	15
2.3 PHP開發環境介紹.....	16
2.3.1 PHP(PHP Hypertext Preprocessor)介紹 .....	16



2.3.2 Web Server介紹 .....	17
2.3.3 MySQL介紹 .....	17
2.3.4 整合性環境Appserv 介紹 .....	18
第三章 系統分析與案例說明 .....	19
3.1 案例需求說明 .....	19
3.3 以UML建構系統模型 .....	21
3.3.1 類別圖(Class Diagram) .....	21
3.3.2 循序圖(Sequence Diagram) .....	21
第四章 系統實作與PIM至PSM之轉換 .....	23
4.1 PIM 轉關聯式 PSM .....	23
4.2 PIM 轉應用程式 PSM .....	27
4.3 應用程式PSM轉CODE .....	31
4.4 雛型系統展示與說明 .....	36
4.5 PSM TO PIM自動轉換工具程式 .....	38
第五章 結論及建議 .....	40
5.1 結論 .....	40
5.2 未來研究建議 .....	41
參考文獻 .....	42



## 圖目錄

圖 1.1 研究流程 .....	3
圖 2.1 UML 圖對照系統開發流程 .....	9
圖 2.2 Model Driven Architecture .....	11
圖 2.3 MDA 軟體發展生命週期 .....	12
圖 2.4 MDA 三個模式的轉換步驟 .....	14
圖 3.1 採購需求分析流程 .....	20
圖 3.2 以 UML 類別圖呈現之系統分析結果 .....	22
圖 3.3 以 UML 循序圖呈現之系統分析結果 .....	22
圖 4.1 本研究案例之 DMD .....	24
圖 4.2 集中採購物網站採購需求清單畫面 .....	37
圖 4.3 集中採購物網站新增採購品項畫面 .....	37
圖 4.4 PSM to PIM 轉換判斷流程.....	38
圖 4.5 PSM to PIM 自動轉換工具程式畫面.....	39



## 表目錄

表 2.1 MDA Tools List .....	15
表 4.1 MySQL 之資料型態 .....	24
表 4.2 Employee 之關聯式 PSM 與系統資料庫的對應及 DDL 語法 .....	25
表 4.3 Product 之關聯式 PSM 與系統資料庫的對應及 DDL 語法 .....	26
表 4.4 Order 之關聯式 PSM 與系統資料庫的對應及 DDL 語法 .....	27
表 4.5 UML 類別圖 Employee (PIM)轉換至程式樣板(PSM) .....	28
表 4.6 UML 類別圖 Product (PIM)轉換至程式樣板(PSM) .....	28
表 4.7 UML 類別圖 Order (PIM)轉換至程式樣板(PSM) .....	29
表 4.8 UML 類別圖 Quantity (PIM)轉換至程式樣板(PSM) .....	29
表 4.9 UML 類別圖 Cart (PIM)轉換至程式樣板(PSM) .....	30
表 4.10 Employee 程式樣板(PSM)轉換至 PHP 程式(CODE) .....	31
表 4.11 Product 程式樣板(PSM)轉換至 PHP 程式(CODE) .....	32
表 4.12 Order 程式樣板(PSM)轉換至 PHP 程式(CODE) .....	33
表 4.13 Quantity 程式樣板(PSM)轉換至 PHP 程式(CODE) .....	34
表 4.14 Cart 程式樣板(PSM)轉換至 PHP 程式(CODE) .....	34
表 4.15 UML 類別圖(PIM)轉換成 PHP 程式樣板(PSM)轉換規則.....	39

# 第一章 緒論

## 1.1 研究背景

近年來，企業組織不但以『資訊系統』做為幫助解決管理、決策及資訊處理上的問題，更可以改善組織作業的效率與效能，繼而提高企業的競爭優勢。但是企業內部佈署眾多資訊系統，涵蓋著舊與新的系統，因此要能使新舊系統能相容，降低錯誤率的產生並維持系統的處理效率，便成為一個必須面臨的複雜且重要議題。

物件管理組織 (object management group, OMG) 在 2001 年提出了模型驅動架構 (model-driven architecture, MDA) [12]，由於過去在塑模系統時，常是將系統的邏輯與實作平台的資訊一起建構於模型中，造成模型在轉換實作平台時往往無法再繼續使用。因此，在 MDA 的架構中，是將系統模型的發展分為兩階段，首先是建立只描述系統抽象邏輯設計的 Platform Independent Model (PIM)，再來是將系統的 PIM 結合實作平台的資訊來產生 Platform Specific Model (PSM)，最後才依照 PSM 來實作系統的程式碼。所以當系統需要轉移至新的實作平台時，不再需要重製系統的設計模型，只需要將系統的 PIM 依照新的實作平台來轉換成對應的 PSM，再從其中實作新平台的程式碼即可，藉此便可大幅增加模型的重複使用性 (reusability)[2]。對於開發現今越來越多樣化的平台與應用程式，MDA 提供了一個良好的解決方案。其期待的益處包括：減短系統開發週期、降低系統開發成本、增加系統品質、降低異質平台轉換困難度、增加模型的重複使用性，當然其中還包括文件的製作與維護。讓系統從分析、設計到實作，其自動生成的程式碼，甚至相關的文件，能夠在開發流程中無接縫的整合，提高可維護性。

OMG 提出 MDA 軟體開發的方法，強調開發者可依循技術與軟硬體之需求，進行系統的設計、開發及維護。當完成系統開發，也同時產生技術檔及相關系統資

訊，因此有利於未來系統的維護與系統的重新開發。而軟體再使用性及減少開發的成本與時間，正是軟體開發所關切的重要問題。若於系統開發的過程中，導入 MDA 開發方法，企業必能迅速開發出符合自己功能需求的系統，更可藉由 MDA 架構整合舊有的系統，達成跨技術平臺的目的。

## 1.2 研究動機與目的

即使 MDA 理論架構是對開發者有利，而且其系統開發方法能帶來許多的系統開發上的好處，但是目前利用 MDA 開發方法進行系統開發的企業或軟體廠商有限，所以潛在的問題必定影響企業及廠商使用的意願。

以往有關 MDA 系統開發的研究，大多在 Java 平台上利用 Java MDA 開發工具(例如 OptimalJ)所做的探討，事實上大部分的主機代管或是虛擬主機的服務，所提供的都是 Open Source 的環境(例如 Apache + PHP + MySQL)，甚少有提供 Java 平台的，所以本研究將會提出以 PHP 語言如何由一開始的 UML (Unified Modeling Language)系統分析產生與平台無關的 PIM，再由 PIM 轉換為與平台相關的 PSM 以及最後實際執行的 CODE。

為了驗證本架構的可行性，以 MDA 架構為基礎，在 Open Source 的環境下用 PHP 實作建置一個集中採購需求的網站，藉以瞭解在 Open Source 的環境下運用 MDA 時相關的問題，並提出問題的探討及建議。

另外以 MDA 的轉換工具而言，在 PHP 的語言中並無一適合之轉換工具，本研究整理提出一個快速將 UML (Unified Modeling Language) 類別圖轉換為 PHP(Hypertext Preprocessor)程式樣版的方法與工具，節省人工轉換過程中需要了解 MDA 架構的時間，以及避免人為轉換的錯誤並且確保程式碼的品質。

### 1.3 研究流程

本論文之研究流程如圖 1.1。由上而下首先設定研究主題與目的，接著進行 MDA、UML 與 PHP 相關文獻探討，再利用實作案例開始進行系統分析，以 PHP 語言進行模型驅動架構下之系統開發建置，並提出其 PIM 轉換到 PSM 之方法與工具。最後，歸納本研究之結論與相關建議。

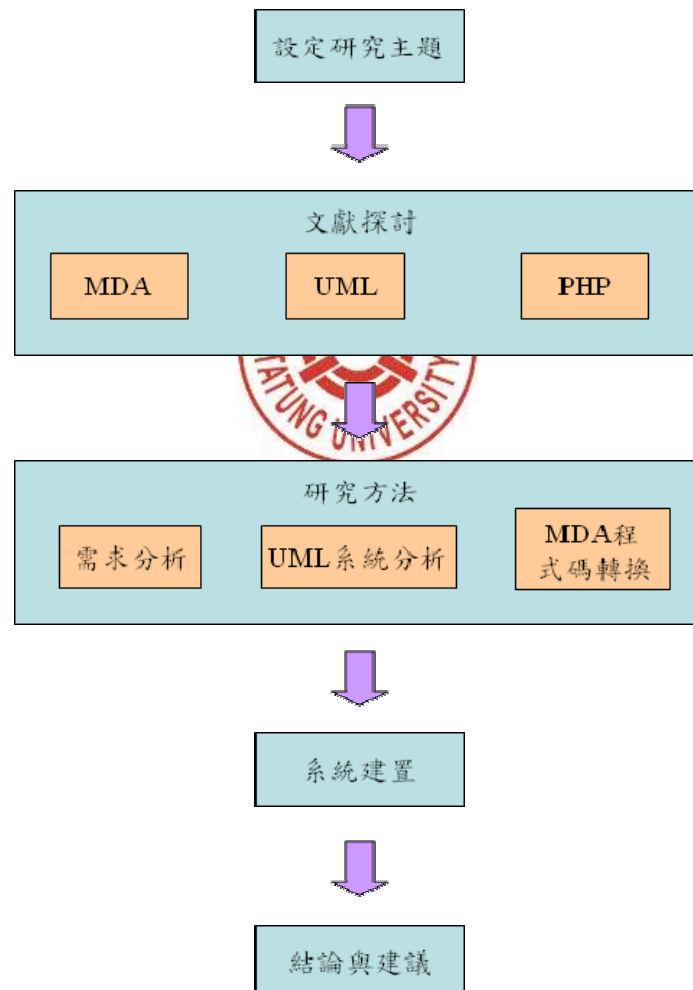


圖 1.1 研究流程

## 1.4 論文架構

本論文將分成五個章節，第一章序論描述本研究之研究背景、研究動機目的以及研究流程，並列出本研究之論文架構；第二章為文獻探討，描述與本研究所使用之相關理論 UML (Unified Modeling Language)與 MDA (Model-Driven Architecture) 架構以及介紹研究使用之 PHP(Hypertext Preprocessor)開發環境；第三章為系統分析與案例說明，說明本研究之研究方法並以實際案例作為系統分析；第四章系統實作以 PHP 語言進行系統開發實作並提出其 PIM(Platform Independent Model)轉換到 PSM(Platform Specific Model)之方法與工具；第五章作為本論文之結論，描述研究成果、研究貢獻及未來研究建議。

### 第一章 緒論

說明研究背景、研究動機與目的。

### 第二章 文獻探討

探討 UML 與 MDA 架構及介紹 PHP 開發環境。

### 第三章 系統分析與案例說明

介紹案例情境與需求分析，並進行系統分析。

### 第四章 系統實作與 PIM 至 PSM 之轉換工具

以 PHP 語言進行系統開發實作並提出其 PIM 轉換到 PSM 之方法與工具

### 第五章 結論及建議

描述研究成果、研究貢獻及未來研究建議。



## 第二章 文獻探討

本章將探討與 MDA 架構相關之研究理論，以及本研究所使用研究方法之相關應用。在第一節將介紹 MDA 架構所使用的系統分析方法 UML 之定義與圖形類別；第二節則說明 MDA 模型驅動架構及 PIM、PSM 與 Code 三階段之轉換關係；第三節是介紹 PHP 程式的開發環境及本研究所使用之開發環境。

### 2.1 統一塑模語言 UML (Unified Modeling Language)

本節將介紹統一塑模語言 UML(Unified Modeling Language)的發展過程及圖形類別，以及 UML 與系統開發流程之對照。

#### 2.1.1 UML 的發展

物件導向語言(Object Oriented Language)於 70 年代中期及 80 年代後期間出現，而從 1989 年到 1994 年之間，語言的數量從不到十種增加到超過五十種。在眾多的模式語言中，語言的創造者雖然努力推崇自己的產品，並且不斷的在實作中改善。但是，物件導向方法的使用者並無法瞭解不同模式語言的優缺點及彼此的差異性，以致於難以選擇合適的語言來符合其需求，於是有了令人注目的 Grady Booch 與 Ivar Jacobson 的 OOSE (Object-Oriented Software Engineering)及 Jim Rumbaugh 的 OMT(Object Modeling Technique)的方法[3]。

1994 年 10 月，Booch 和 Rumbaugh 開始將 Booch 93 和 OMT-2 (Object Modeling Technique) 統一起來，並於 1995 年 10 月發佈了第一個公開版本，稱之為統一方法 UM 0.8 (United Method)。在 1995 年秋天，OOSE 的創始人 Jacobson 並加入一起工作。終於經過 Booch、Rumbaugh 和 Jacobson 三人的共同努力，在 1996 年 6 月和 10





月分別發佈了兩個新的版本，即 UML 0.9 和 UML 0.91，並將 UM 重新命名為 UML(Unified Modeling Language)。至 1996 年一些機構將 UML 作為其商業策略已十分明顯，因此 UML 的開發者倡議成立 UML 成員協會，以促進、加強和完善 UML 的定義工作。當時的成員有 DEC、HP、I-Logix、Intellicorp、IBM、ICON Computing、MCI Systemhouse、Microsoft、Oracle、Rational Software、TI 以及 Unisys。由於這些公司的合作參與，也使得 UML 1.0(1997 年 1 月)及 UML 1.1(1997 年 11 月)的定義更完整，並且具有較佳的表現性與適用性[3]。

接著在 1998 年六月發表 UML 1.2 版本，當年秋天隨即發表出 UML 1.3 版本，因此開發過程遵循 UML 已經進入第六年，軟體工具研發廠商和系統分析人員對於 UML 語言都有了更多的經驗，更能瞭解其優缺點，並且軟體業在這些年也發生了很多變化，需要去支援一些新的技術，例如基於元件的開發及可執行的模式等。這些需求仍無法順利用現行的 UML 適當地處理，加上 MDA 理論架構的推行，因此對 UML 大幅的修訂，於 2002 年七月發表 UML 2.0 版本[3,6]。

### 2.1.2 UML 的定義及圖型分類

UML 是一種定義良好、易於表達、功能強大且普遍適用的模式語言，它融入了軟體工程領域的新思想、方法與技術，不限於支援物件導向的分析與設計，更可支持從需求分析開始的軟體發展的全部開發過程[3]。

UML 的定義包括 UML 語義和 UML 表示法兩個部分：

- **UML 語義**：描述基於 UML 的精確 meta-model 定義。meta-model 為 UML 的所有元素在語法和語義上提供了簡單、一致、通用的定義性說明，使開發者能在語義上取得一致，解決因人而異的最佳表達方法所造成的影響。此外 UML 還支援對 meta-model 的擴展定義。

- **UML 表示法：**定義 UML 符號的表示法，為開發者或開發工具使用這些圖形符號和文本語法為系統建模提供了標準。這些圖形符號和文字所表達的是應用程級的模式，在語義上它是 meta-model 的實例。

UML 的重要內容可以由下列五類圖(共 9 種圖形)來定義[3]：

**第一類：**使用者案例圖(Use Case Diagram)從使用者的角度描述系統的行為者與系統間之互動行為與關係。從內部觀點來看可描述系統在做什麼，而從外部觀點來看可描述行為者與系統如何互動。

**第二類：**靜態圖(Static diagram)，包括類別圖(Class Diagram)及物件圖(Object Diagram)。其中類別圖描述系統中類別的靜態結構，主要用以表示系統存在之物件類別及類別間之邏輯關係。不僅定義系統中的類別，表示類別之間的聯繫如關聯、依賴、聚合等，也包括類別的內部結構(類別的屬性和操作)。類別圖描述的是一種靜態關係，在系統的整個生命週期都是有效的。而物件圖可說是類別圖的實例，用來描述一系統於某依時間點的靜態資料，幾乎使用與類別圖完全相同的標識。他們的不同點在於物件圖顯示類的多個物件實例，而不是實際的類別。一個物件圖是類別圖的一個實例。由於物件存在生命週期，因此物件圖只能在系統某一時間段存在。

**第三類：**行為圖(Behavior diagram)，包括狀態圖(State Diagram)與活動圖(Activity Diagram)，用以描述系統的動態模式和組成物件間的交互關係。狀態圖描述類別的物件所有可能的狀態以及事件發生時狀態的轉移條件，主要用來輔助類別圖。在實際上並不需要為所有的類別畫上狀態圖，只需為有多個狀態其行為受外界環境的影響並且發生改變的類別畫狀態圖。而活動圖描述滿足用例要求所要進行的活動以及活動間的約束關係，有利於識別並行活動。



**第四類：交互圖(Interactive diagram)**，包括循序圖(Sequence Diagram)與合作圖(Collaboration diagram)，用以描述物件的交互關係。其中循序圖顯示物件之間的動態合作關係，它強調物件之間訊息發送的順序;合作圖描述物件間的協作關係，而合作圖跟循序圖相似，顯示物件間的動態合作關係。除顯示資訊交換外，合作圖還顯示物件以及它們之間的關係。如果強調時間和順序，則使用循序圖;如果強調上下級關係，則選擇合作圖。

**第五類：實現圖(Implementation diagram)**，包括元件圖(Component Diagram)與部署圖(Deployment Diagram)。其中元件圖描述程式碼部件的物理結構及各元件之間的依賴關係。一個元件可能是一個資源程式碼元件、一個二進位元件或一個可執行元件。它包含邏輯類或實現類的有關資訊，有助於分析和瞭解元件之間的相互影響程度。部署圖定義系統中軟硬體的物理體系結構，它可以顯示實際的電腦和設備(用節點表示)以及它們之間的連接關係，也可顯示連接的類型及元件之間的依賴性。在節點內部放置可執行部件和物件，以顯示節點與可執行軟體單元的對應關係。

UML 提供的九種視圖從不同應用層次和不同角度，自系統分析、設計直到實現均提供有力支援，可以在不同的階段建立不同的模式，每個類型的圖其目的也各不相同，見圖 2-1。但在以上的九種圖形中，最主要還是以類別圖(Class Diagram)及循序圖(Sequence Diagram)為基本，也是應用在系統分析時最常使用之圖形[3]。

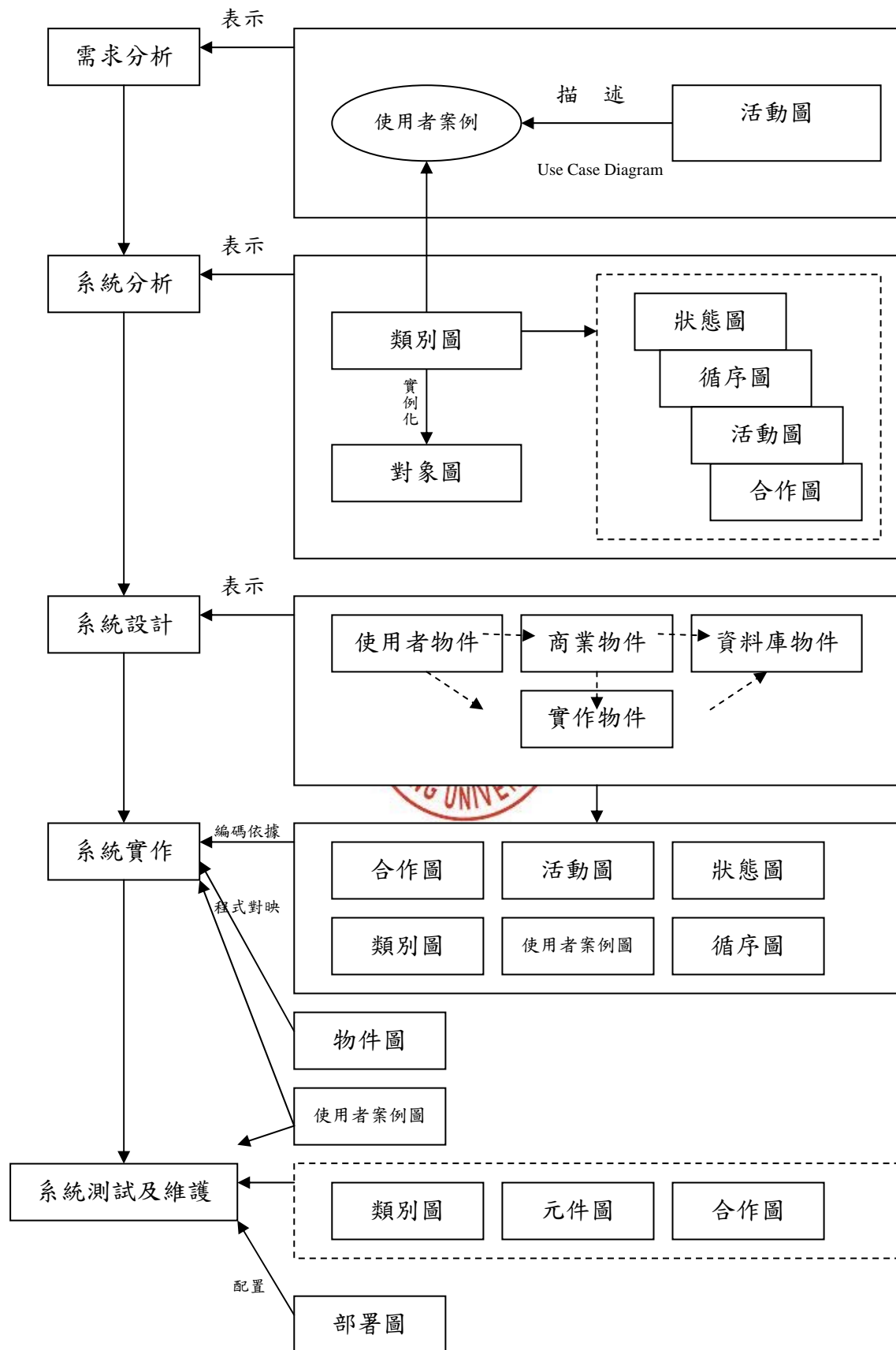


圖 2.1 UML 圖對照系統開發流程[5,8]

## 2.2 模型驅動架構 MDA (Model Driven Architecture)

本節將就模型驅動架構 MDA (Model Driven Architecture) 的發展過程及類別轉換的模式作一介紹，並描述 MDA 運用在系統開發之優點說明。

### 2.2.1. MDA 介紹

Object Management Group(OMG)組織的 Unified Modeling Language(UML)規範了標準的模型符號(Model Notations)來讓系統開發人員做系統設計的描述。以 UML 來塑模系統，除了因為有共同的溝通符號能減少溝通時的誤會外，也能以系統模型(model)的資訊來驗證設計是否符合系統需求；另外對於後續的系統維護，或是讓專案新進人員了解現有系統的設計，都有相當大的幫助。然而可惜的是 UML 文件只能成為開發人員在開發系統時的參考，對於加速系統的開發則是很難有助益的。

於是 OMG 組織在 2001 年提出了 Model Driven Architecture(MDA)[12]的軟體發展標準。在 MDA 的架構中，包含了兩種重要的模型(model)，一種是只描述系統抽象邏輯的 Platform Independent Model(PIM)，另一種則是將系統抽象邏輯投射到實作平台後的模型，稱為 Platform Specific Model(PSM)[4]。由於過去在系統塑模時，通常是直接塑模 PSM 的資訊，然而這樣做的結果卻會造成系統將來在轉移實作平台時的困難。其原因為系統的抽象邏輯已經緊緊地跟實作平台的資訊綁在一起，使得模型的資訊無法重複使用，需要再依新的實作平台重新製作 PSM，形成資源的浪費，而這也就是 MDA 將系統的抽象邏輯獨立出成為 PIM 的理由。

圖 2.2,MDA 以 UML、MOF (Meta Object Facility)及 CWM (Common Warehouse Meta-model)三個主要核心功能，整合不同仲介軟體平臺(如 CORBA、.NET、EJB、SOAP 等)，並能在兼具各個領域知識條件下，發展出高整合性及高效能的軟體[12]。

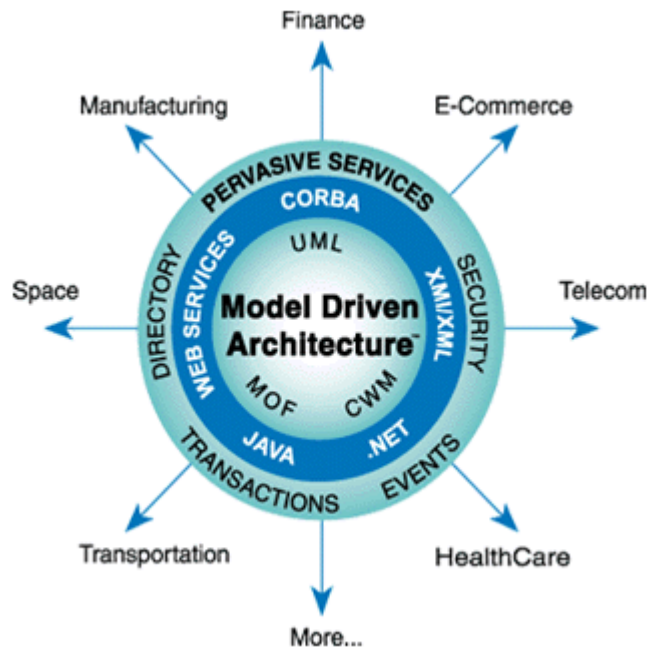


圖 2.2 Model Driven Architecture [12]

### 2.2.2. MDA 之類別及轉換

MDA 這個新的軟體開發架構其關鍵是軟體開發過程中的每個階段之產出均須建構出模式，且該模式之產出是下一個階段的輸入。而 MDA 的生命週期與其它系統開發模式的系統發展生命週期並沒有差別(請參考圖 2.3)，包括需求(Requirements)、分析(Analysis)、低階設計(Low-level Design)、程式編輯(Coding)、測試(Testing)與部署(Deployment)等，但是主要的差別之一是在發展過程中步驟之產出，強調該產出是由電腦可理解的正式模式(Formal Model)表達 [1]。

MDA 需求階段的產出主要是文字之描述，這些產出是進行分析工作的主要資訊來源(也就是主要的輸入)，該階段的產出主要是建構出 PIM。同理 PIM 是進行低階設計階段的主要資訊來源，該階段的產出是要建構出 PSM，而 PSM 是進程式編輯階段的主要資訊來源，該階段的產出主要是建構出程式碼。

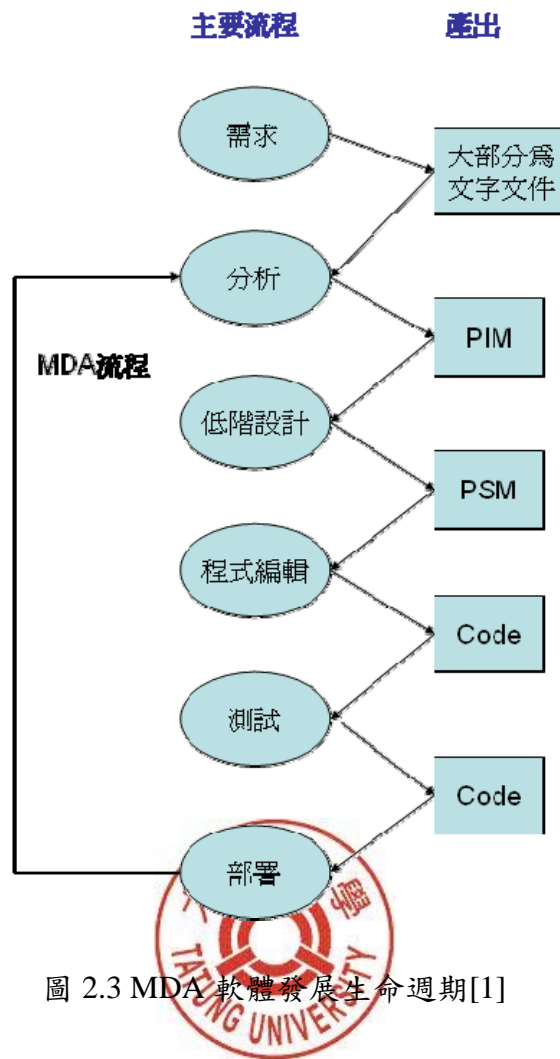


圖 2.3 MDA 軟體發展生命週期[1]

所以 MDA 有三個核心模式：PIM、PSM 及 Code。茲將分別說明如下[1]：

- PIM：是一種高階抽象的模式，該模式與開發技術獨立無相關。PIM 是分析與設計結果的重要產出，主要根據需求描述的結果，從如何支援企業運作的觀點描述一個軟體系統，並不涉及描述系統開發與運作之平台。PIM 需有完整定義 (Well-Defined) 的語言來描述，一個具有完整的語言具有完整定義的語法 (Syntax) 與語義，且適合用電腦來自動解譯 (Automated Interpretation)。因此以 UML 來描述 PIM 應是目前最好的選擇。
- PSM：是一種特定平台的模式，也就是該模式相依於軟體開發技術。對某一種 PSM 而言，可能僅具有該特定平台知識的開發者才能理解。一個 PSM 主要是以

開發工具的架構描述一個軟體系統。一個 PIM 可被轉換成一個或多個 PSM，因為一個系統可能包含幾種技術，對每一個特定的技術平台需產生一個與其他技術分開的 PSM，PSM 間可藉由溝通橋樑(Communication Bridge)的機制來互動[11]。

- Code：每一個 PSM 需被轉成程式碼，因為一個 PSM 相依於其開發技術，因此 PSM 轉成程式碼之步驟非常直接。若有個 PSM 則會轉出多種程式碼，不同的程式碼間也需藉由溝通橋樑的機制來互動。

MDA 除了定義以上模型外，也描述了四種模型間的轉換(Mapping)。這四種轉換分別為[12]：

- PIM to PIM：此種轉換發生於精練(refine)系統的 PIM。由於設計一個良好穩定的系統並非一蹴可及，而是以漸進演化的方式來建構。而這演化的過程所隱含的意義就是 PIM to PIM 的轉換。
- PIM to PSM：此種轉換發生於當系統的 PIM 已經完成，並要投射至某個實作平台上時。
- PSM to PSM：此種轉換發生於精練(refine)系統的 PSM。由於每個實作平台都可能會有各自的特性，所以我們可能會需要將 PIM 投射出的 PSM 做微調的動作來發揮該平台的優勢，而這微調的動作就屬於 PSM to PSM 的轉換。
- PSM to PIM：此種轉換發生於精練(refine)某個舊有的系統(Legacy System)。一個企業應用系統的開發，常無可避免地必須與舊有系統做整合，一般比較常用的方式是以 Adapting 的方式建立溝通的介面，像是在新系統與舊系統之間建立 Message Oriented Middleware(MOM) 伺服器做資訊交換。另外一種整合方式則是以反向工程(reverse engineering)來反轉產生舊有系統的 PIM，經過修改後，再重新部署至新的平台，而這過程就是屬於 PSM to PIM 的轉換。





MDA 的每一個轉換(例如 PIM→PSM、PSM→Code)需有清楚的轉換定義，且該轉工作主要是藉由 CASE 工具來執行，也就是 PIM 可藉由 CASE 工具轉換成 PSM，再轉換成 Code(如圖 2.3) [1]。

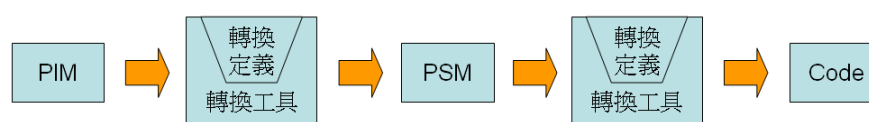


圖 2.4 MDA 三個模式的轉換步驟[1]

### 2.2.3. MDA 程式的轉換工具

目前已經有不少軟體供應商推出可支援 MDA 程式轉換的 CASE 工具，但這些工具仍然有進步之空間，雖然 PIM 至 PSM 與大部分的 PSM 至 SQL Code 之轉換已能自動化，但是 PSM 至應用系統程式碼之轉換僅能轉出一些樣版程式(Template Code)，系統開發者仍須以人工補強[1]。

市面上的 MDA Tools 整理如下表 2.1，大致可分為 Open Source Tools 與 Commercial Tools 兩類。比較知名的例如 Rational Rose、Power Designer 與 Together 等，大部分的 Tools 均是以轉換 Java 或 .NET 等程式語言為主，並未見到可以轉換 PHP 之工具，因此本研究提出一個快速將 UML (Unified Modeling Language) 類別圖轉換為 PHP 程式樣版的方法與工具，節省人工轉換過程中需要了解 MDA 架構的時間，以及避免人為轉換的錯誤並且確保程式碼的品質。

表 2.1 MDA Tools List

Open Source Tools	Commercial Tools
MOFScript (Eclipse plugin)	ArcStyler(MDA tool from Interactive Objects)
MTF(IBM Model Transformation Framework)	MCC (Model Component Compiler,J2EE)
ATL (Eclipse plugins)	OptimalJ(PSM transformations, integrated UML tool)
MTL Engine (Integrates with Netbeans MDR and Eclipse EMF)	Xactium XMF Mosiac(model-based mapping, generation and execution tool)
ModFact (MOF Repository and QVT-like engine)	SosyInc (Modeler and Transformation Engine)
GMT (Generative Model Transformer)	Model-in-Action(code generation and model to model transformation)
KMF (Kent Modelling Framework)	MetaEdit+ (modeling and metamodeling tool)
OpenArchitectureWare (template-based generator framework)	MDWorkbench (text and model transformation toolset)
OpenMDX(integrates with several tools, J2EE, .Net)	iQgen 3.0 (template-based generator)
AndroMDA (J2EE code generation)	Rational Rose
XDoclet (attribute based code generation tool for J2EE)	Power Designer
Middlegen (database driven code generator)	Together

#### 2.2.4. MDA 的優點

架構在 MDA 上的開發流程大致上是先將系統的抽象邏輯塑模成 PIM，再透過 PIM to PSM 轉換投射成某個實作平台的 PSM，之後再經過 PSM to Code 的轉換，產出系統的程式碼。這些轉換的動作可以是手動，或是透過工具程式做半自動甚至是全自動的轉換，且目前市面上也已有許多廠商開發出了 MDA Compliant 的開發工具。以 MDA 精神搭配 MDA 工具來開發系統，具有下列的好處[10]：

- 減短系統開發的週期：由於系統的程式碼可以從模型產生出來，所以可以降低系統實作的時間。
- 降低系統開發的成本：由於部份程式的實作可以透過工具產出，所以可以減少專案對人力的需求。



- 增加系統的品質：由於程式碼可以是從 PSM 模型自動或是半自動的產出程式碼，意味著可以降低人為的疏失以及確保程式碼的品質。
- 降低異質平台轉換的困難度：由於 MDA 可以從系統的 PIM 直接投射到某個實作平台上的 PSM 後再產出程式碼，所以可以降低異質平台轉換的困難度。
- 增加模型的重複使用性：由於系統的 PIM 描述的僅是系統的抽象邏輯，所以將來若有其他專案系統的設計跟原有系統類似，就可以直接使用或是修改原有系統的 PIM，所以透過 MDA 可以增加模型的重複使用性。

## 2.3 PHP 開發環境介紹

PHP (PHP Hypertext Preprocessor)是目前網頁常用的程式語言之一，本節除了介紹 PHP 程式語言的發展以外，更探討適合 PHP 程式語言開發之環境，例如網頁伺服器 (Web Server) 以及常搭配使用之資料庫 MySQL，此外也提出本研究所使用之 PHP 開發環境說明。

### 2.3.1 PHP(PHP Hypertext Preprocessor)介紹

PHP (PHP Hypertext Preprocessor，最初稱為"Personal Home Page Tools"，也稱為"Professional Homepages"，或者"Pre-Hypertext Processor")，是一種開放源代碼的腳本編程語言，主要用於 Web 伺服器的伺服器端應用程式，用於動態網頁設計。PHP 可以用於替代微軟的 ASP/VBScript/JScript 體系、Sun 微系統公司的 JSP/Java 體系，以及 CGI/Perl 等。它是一種嵌入 HTML 頁面中的腳本語言。PHP 特別適合用來開發網站程式，可以內嵌在 HTML 碼。PHP 程式的原始碼是純文字，所以可以用任何可處理純文字檔的文字編輯器，如：記事本、vi、emacs 等，來撰寫 PHP 程式。

PHP 基本上是用來製作「互動式」或是「CMS (Content Management System)」等網站的網路程式語言，PHP 是一個在網路普遍被伺服器利用的語言，它是一種開放式來源碼，非常有效率極適合用來應用在網頁製作上 [7]。

### 2.3.2 Web Server 介紹

Web 技術的獨特之處是採用超鏈接和多媒體信息。Web Server 使用超文本標記語言（HTML-HyperText Marked Language）描述網絡的資源，創建網頁，以供 Web 瀏覽器閱讀。HTML 文檔的特點是交互性。不管是一般文本還是圖形，都能通過文檔中的鏈接連接到 Server 上的其他文檔，從而使客戶快速地搜尋他們想要的資料。HTML 網頁還可提供表單供用戶填寫並通過服務器應用程序提交給資料庫。這種資料庫一般是支持多媒體數據類型的。

在 1995 年之前就有蠻多的 Web 架設伺服器軟體的出現，不過，真正到了 1995 年之後，由國際超級電腦應用中心（National Center for Supercomputing Applications, NCSA）主導並克服了一些 Web 主機的 bug 之後，才讓這個 http 協定的 WWW 套件得到了更廣泛的應用！而因為這個釋出的版本是來自於一些 bug 的克服，因此，這個 WWW 套件被戲稱為『A patchy server』，意思就是說，一個經過更新後的 Server 的意思！後來，因為要將名字確定下來，乾脆就直接取其諧音，用『Apache』，這也是目前使用最普遍的 web server [9,13]。

### 2.3.3 MySQL 介紹

MySQL 是一個真正的多用戶、多線程 SQL 數據庫服務器。SQL（結構化查詢語言）是世界上最流行的和標準化的數據庫語言。MySQL 主要目標是快速、健壯和易用。最初是因為們需要這樣一個 SQL 服務器，它能處理與任何可不昂貴硬件平

台上提供數據庫的廠家在一個數量級上的大型數據庫，但速度更快，MySQL 就開發出來。自 1996 年以來，我們一直都在使用 MySQL，其環境有超過 40 個數據庫，包含 10,000 個表，其中 500 多個表超過 7 百萬行，這大約有 100 GB 的關鍵應用數據 [7]。

MySQL 是一個資料庫系統，它是免費且非常好的系統，它通常與 PHP 語言一起使用。MySQL 主要被使用在開發「互動式」與「CMS」等網站系統。

#### 2.3.4 整合性環境 Appserv 介紹

基於上述說明 PHP、Apache 與 MySQL 均為目前(尤其 Open Source)最常應用之技術平台，但三者之間尚有諸多版本及相容性問題，為求一較完善之系統開發環境，傾向以完整測試之整合環境為優先考量。

Appserv 是一個在微軟視窗作業系統下，包含 Apache、MySQL 與 PHP 的整合套件，可以快速建立適合 PHP 語言的開發環境。使用者可由網站[9]免費下載，目前提供兩個版本，分別是 2.4.6 與 2.5.6，若有新版本釋出網站也會更新檔案以供下載。本研究乃採用 AppServ 2.5.6 作為 PHP 程式之開發環境，可以快速安裝至個人電腦上，並且軟體經過整合性測試，減少因版本或不同環境配合而產生之開發環境問題。

## 第三章 系統分析與案例說明

本章將說明需求案例與系統分析之流程，並且描述如何利用 UML 工具建構系統模型。在 UML 九種圖形之中，我們以最常用的兩種圖表類別圖 (Class Diagram) 與循序圖 (Sequence Diagram)來做為系統分析時所使用之圖表。

### 3.1 案例需求說明

本研究所採用之案例需求情境簡述如下：

A 公司經常進行儀器耗材的採購，但是目前需求分散提出，採購人員零星採購不僅耗費人力，在數量及總價上更無法與廠商議價。為解決這個問題，因此欲規劃一個集中採購的網站，可以將使用者的需求集中，順利統計正確數量與需求，並且讓採購人員可以批次作業，節省人力，而且集中採購便於與廠商議價，降低採購成本。

未來可以考慮與供應商網站資料相結合，或是將收集之需求配合內部自動作業流程，將需求數量與規格通知相關作業人員，包括供應商可以直接線上報價送畫等功能，但是初期以先收集使用者的採購需求為考量。

系統建置的原則希望在不影響現有系統的提條件下開發，並希望能有跨平台及保有未來修改的彈性，因此選擇在 Open Source 的環境中以 PHP 語言開發，而對於未來功能修改時不致於重複撰寫程式，故運用 MDA 架構以保有系統容易維護與修改的特性。

### 3.2 需求與系統分析

依照案例情境開始進行系統分析，首先確認使用者需求，規劃程序為使用者依實際需要提出採購需求，若此需求已經在系統需求清單之中，就直接增加需求數量。而若是使用者之需求未在系統需求清單之中，則自行新增一項需求，記錄所需採購品項以及集體採購起、迄時間。待使用者設定之期限截止，採購人員將可以統計並查詢出此次採購之規格與數量，統一與廠商議價並進行採購作業，可以降低採購成本並將採購流程集中作業，節省人力並提高工作效率。圖 3.1 為其採購需求分析流程。

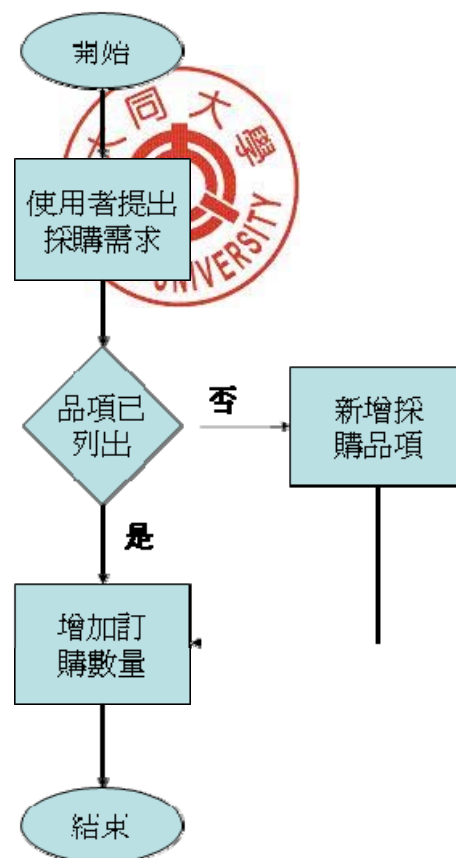


圖 3.1 採購需求分析流程

### 3.3 以 UML 建構系統模型

在 UML 九種圖形之中，我們以最常用的兩種圖表來做為系統分析時使用之圖表。這兩種圖分別為類別圖 (Class Diagram) 與循序圖 (Sequence Diagram)，茲將這兩個圖內容詳細說明如下：

#### 3.3.1 類別圖(Class Diagram)

類別圖描述系統中類別的靜態結構。不僅定義系統中的類別，表示類別之間的聯繫如關聯、依賴、聚合等，也包括類別的內部結構(類別的屬性和操作)。舉例說明，Employee class 有五個 attribute 分別是 ID, Name, Dept, Tel 和 email。並且有一個 method 就是 Employee()。Employee class 與 Cart class 是一個一對多的關係，如此就可以畫出完整的 Employee class 類別圖，以及 Employee class 與 Cart class 的關係。若是將其他所有的 class (Cart, Employee, Quantity, Product 及 Order) 的 attributes、methods 以及相互關係完成後，就會得到一完整的類別圖，如圖 3.2。

#### 3.3.2 循序圖(Sequence Diagram)

循序圖顯示物件之間的動態合作關係，它強調物件之間訊息發送的順序;合作圖描述物件間的協作關係，而合作圖跟循序圖相似，顯示物件間的動態合作關係。除顯示資訊交換外，合作圖還顯示物件以及它們之間的關係。如果強調時間和順序，則使用循序圖;如果強調上下級關係，則選擇合作圖。而本案例若以循序圖來表示，總共會有 5 個 objects，舉例說明若要新增訂購品項，使用者在進入使用者介面之後，首先會看到目前所有訂購項目，如需使用系統，則須先經驗證為員工之後，才可以新增訂購品項或數量。如需要採購之品項已在清單之中，直接增加數量即可，若要採購之品項不再現有清單之中，則需新增採購品項，可以填寫規格數量及需求收集截止日期。將所有相關的物件與訊息整理之後可以得到一完整的循序圖如圖 3.3。

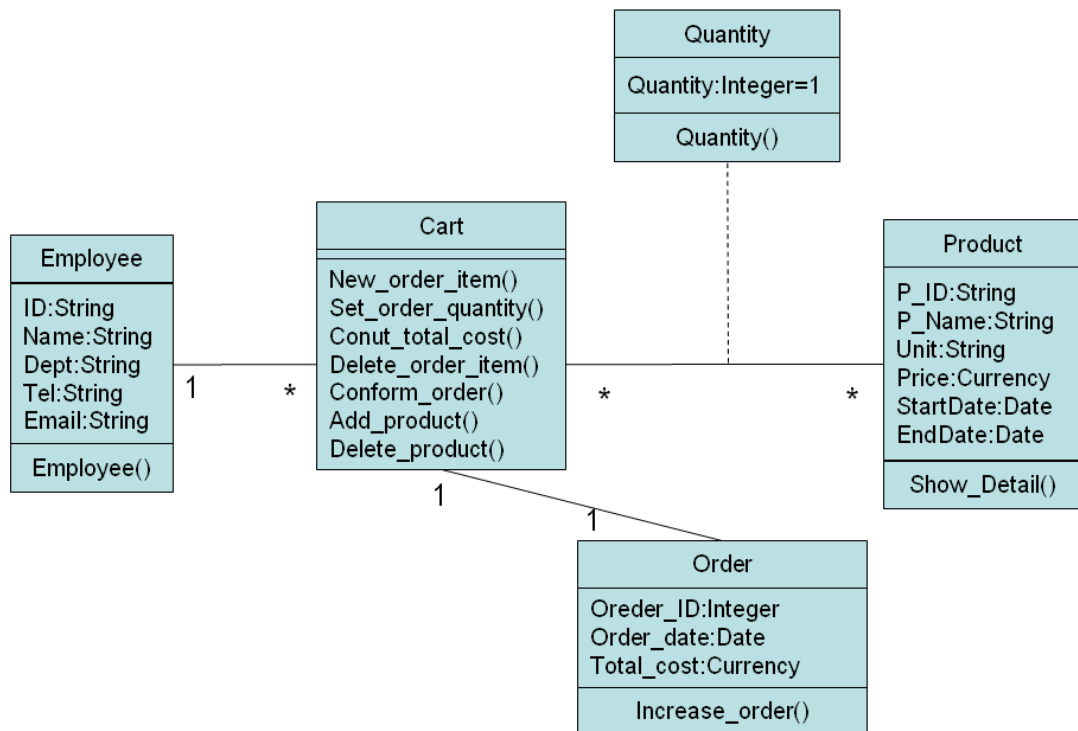


圖 3.2 以 UML 類別圖呈現之系統分析結果

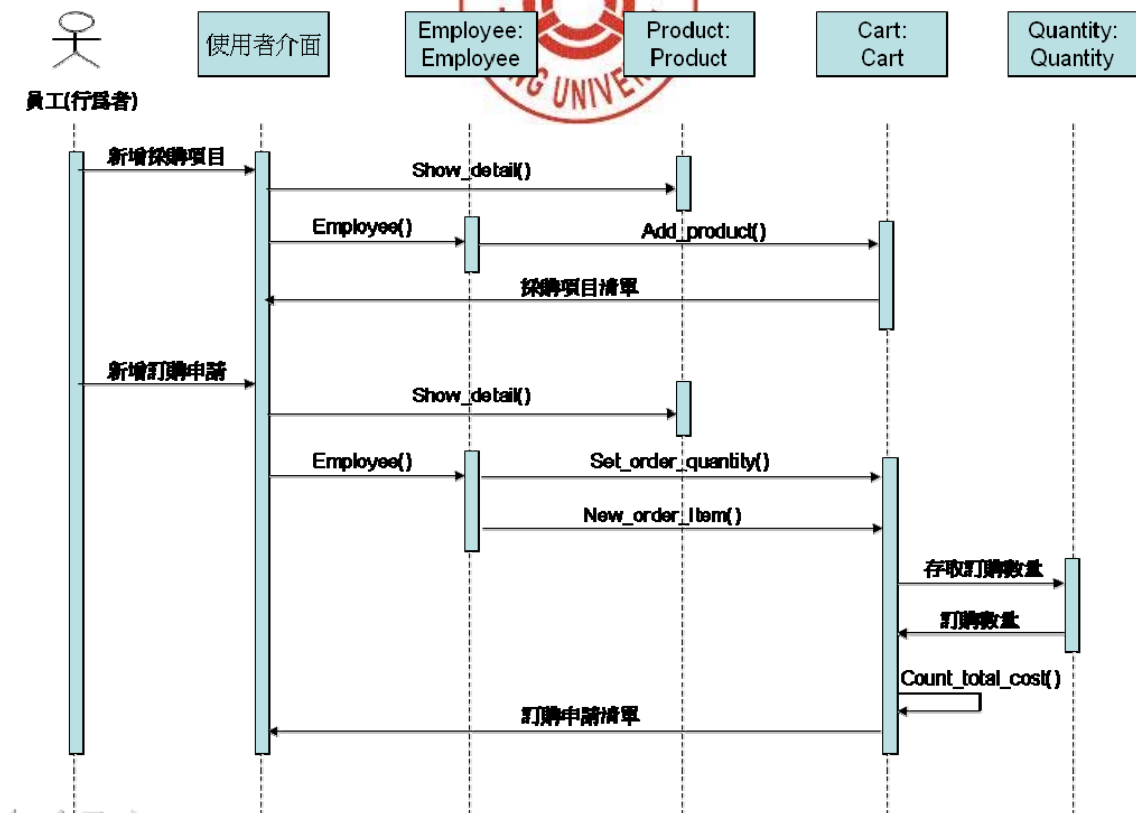


圖 3.3 以 UML 循序圖呈現之系統分析結果



## 第四章 系統實作與 PIM 至 PSM 之轉換

本章將進行 MDA 系統開發實作，以 PHP 語言建置集中採購網站為例，透過實作的過程驗證 MDA 類別之轉換--PIM、PSM 與 CODE 之間轉換的規則與關係。另外以雛形系統展示方式說明本研究開發系統之操作介面與運作方式。

並且針對 PIM 到 PSM 的轉換過程，整理提出一個快速將 UML 類別圖轉換為 PHP 程式樣版的方法與工具，節省人工轉換過程中需要了解 MDA 架構的時間，以及避免人為轉換的錯誤。

### 4.1 PIM 轉關聯式 PSM

當類別圖 (PIM) 產出後，可以按照關聯式資料庫平台的特性，將此類別圖轉換成符合該平台特性的資料表，接下來設計資料庫。本研究使用之資料庫為 MySQL，將各個類別圖之間的關係基數(一對一、一對多、多對一)整理之後，可以得到一資料模式圖(Data Model Diagram, DMD)，如圖 4.1。

接下來開始設計資料庫，基本上一個關聯表須建立一個資料庫的 Table，逐一依屬性定義其欄位名稱、型態、可否為 Null 與其資料存取之功能。完成上述圖 4.1 DMD 之後，必須先做資料類型轉換，可進一步將類別圖轉換為該模式的 DDL (Data Definition Language)。在本研究的案例中採用的關聯式資料庫是 MySQL，其資料型態如表 4.1，在定義其欄位名稱、型態及資料長度之後，可以產出類別圖與資料庫之對應關係，並依 SQL 語法產生 MySQL 資料庫之 DDL。以 Employee 資料表為例，有五個欄位，資料型態皆為文字 (String)，除 ID 欄位資料長度為 10Byte 以外，其餘欄位長度都是 50Byte，在對應表資料型態及 SQL 語法之後結果如表 4.2。本研究案例使用之其他關聯式 PSM 與系統資料庫的對應及 DDL 語法如表 4.3, 4.4。



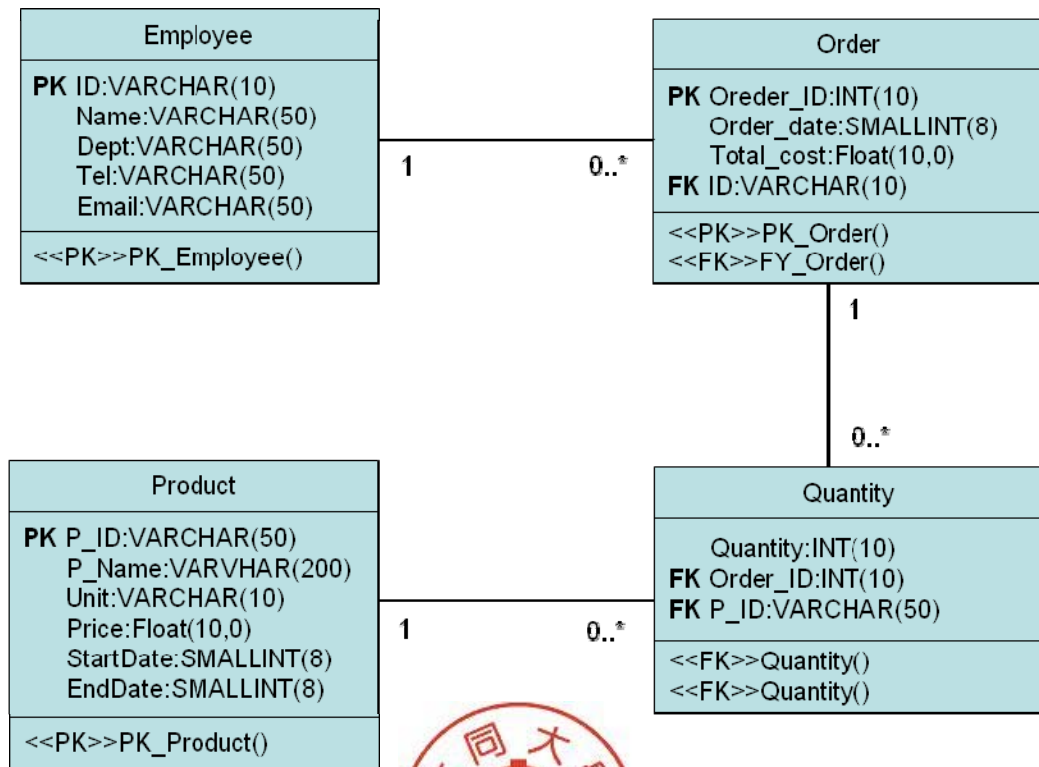


圖 4.1 本研究案例之 DMD

表 4.1 MySQL 之資料型態[1]

Attribute	Data type in MySQL
String	VARCHAR()
Integer	INT()
Date	SMALLINT()
Currency	FLOAT()

表 4.2 Employee 之關聯式 PSM 與系統資料庫的對應及 DDL 語法

Employee					
欄位	型態	NULL?	鍵值	預設值	類別圖
ID	varchar(10)		P		<div>Employee</div> <div>ID:String Name:String Dept:String Tel:String Email:String</div> <div>Employee()</div>
Name	varchar(50)				
Dept	varchar(50)				
Tel	varchar(50)				
Email	varchar(50)				
P:PRIMARY					

SQL Script:  
CREATE TABLE `Employee` (  
  `ID` varchar(10) NOT NULL default "  
  `Name` varchar(50) NOT NULL default "  
  `Dept` varchar(50) NOT NULL default "  
  `Tel` varchar(50) NOT NULL default "  
  `Email` varchar(50) NOT NULL default "  
  PRIMARY KEY (`ID`)  
) DEFAULT CHARSET=big5;



表 4.3 Product 之關聯式 PSM 與系統資料庫的對應及 DDL 語法

Product					
欄位	型態	NULL?	鍵值	預設值	類別圖
P_ID	varchar(50)		p		<div>Product</div> <div>P_ID:String P_Name:String Unit:String Price:Currency StartDate:Date EndDate:Date</div> <div>Show_Detail()</div>
P_Name	varchar(200)				
Unit	varchar(10)				
Price	Float(10,0)			0	
StratDate	varchar(10)				
EndDate	varchar(10)				
P:PRIMARY					

SQL Script:  
CREATE TABLE `Product` (  
`P\_ID` varchar(50) NOT NULL default "",  
`P\_Name` varchar(200) NOT NULL default "",  
`Unit` varchar(10) NOT NULL default "",  
`Price` Float(10,0) NOT NULL default 0,  
`StratDate` varchar(10) NOT NULL default "",  
`EndDate` varchar(10) NOT NULL default "",  
PRIMARY KEY (`P\_ID`)  
) DEFAULT CHARSET=big5;

表 4.4 Order 之關聯式 PSM 與系統資料庫的對應及 DDL 語法

Order					
欄位	型態	NULL?	鍵值	預設值	類別圖
Order_ID	int(10)		P		<div>Order</div> <div>                     Oorder_ID:Integer                      Order_date:Date                      Quantity:Integer=1                 </div> <div>Increase_order()</div>
Quantity	int(10)			0	
ID	varchar(10)				
P_ID	varchar(50)				
P:PRIMARY					
SQL Script: CREATE TABLE `order` ( `Order_ID` int(10) NOT NULL auto_increment, `Quantity` int(10) NOT NULL default '0', `ID` varchar(10) NOT NULL default "", `P_ID` varchar(50) NOT NULL default "", PRIMARY KEY (`Order_ID`) ) DEFAULT CHARSET=big5;					



## 4.2 PIM 轉應用程式 PSM

本節將說明如何由 PIM 轉換成 PSM，僅舉例 Employee 類別作為說明。首先要由系統分析時的類別圖作為轉換基礎，但是目前轉換的工具皆僅限於 Java 或 .NET 等語言，目前尚未有關 PHP 的轉換工具。而 PIM to PSM 的轉換在沒有工具輔助時必須一步一步自行轉換，轉換步驟主要是要將所有變數宣告，並定義 Function。以 Employee 類別來說，有五個 attributes (ID, Name, Dept, Tel, Email) 以及一個 method (Employee)，所以我們需要有一個 Employee class，包含這五個變數與一個 function，轉換結果如表 4.5。其餘的類別依照上述說明方式轉換，轉換之結果如表 4.6, 4.7, 4.8 與 4.9。

表 4.5 UML 類別圖 Employee (PIM)轉換至程式樣板(PSM)

Employee	
類別圖(PIM)	程式樣板(PSM)
<div>Employee</div> <div> ID:String  Name:String  Dept:String  Tel:String  Email:String </div> <div>Employee()</div>	<pre>class Employee {     var \$ID;     var \$Name;     var \$Dept;     var \$Tel;     var \$Email;     function Employee()     {         //Add PHP code here     } }</pre>

表 4.6 UML 類別圖 Product (PIM)轉換至程式樣板(PSM)

Product	
類別圖(PIM)	程式樣板(PSM)
<div>Product</div> <div> P_ID:String  P_Name:String  Unit:String  Price:Currency  StartDate:Date  EndDate:Date </div> <div>Show_Detail()</div>	<pre>class Product {     var \$P_ID;     var \$P_Name;     var \$Unit;     var \$Price;     var \$StartDate;     var \$EndDate;     function Product ()     {         //Add PHP code here     }     function Show_Detail()     {         //Add PHP code here     } }</pre>

表 4.7 UML 類別圖 Order (PIM)轉換至程式樣板(PSM)

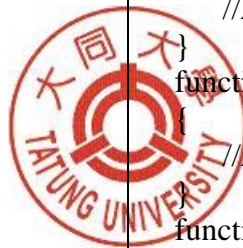
Order				
類別圖(PIM)	程式樣板(PSM)			
<table><tr><td>Order</td></tr><tr><td>Oreder_ID:Integer Order_date:Date Total_cost:Currency</td></tr><tr><td>Increase_order()</td></tr></table>	Order	Oreder_ID:Integer Order_date:Date Total_cost:Currency	Increase_order()	<pre>class Order {     var \$Order_ID;     var \$Order_date;     var \$Total_cost;     function Order()     {         //Add PHP code here     }     function Increase_order()     {         //Add PHP code here     } }</pre>
Order				
Oreder_ID:Integer Order_date:Date Total_cost:Currency				
Increase_order()				

表 4.8 UML 類別圖 Quantity (PIM)轉換至程式樣板(PSM)

Quantity				
類別圖(PIM)	程式樣板(PSM)			
<table><tr><td>Quantity</td></tr><tr><td>Quantity:Integer=1</td></tr><tr><td>Quantity()</td></tr></table>	Quantity	Quantity:Integer=1	Quantity()	<pre>class Quantity {     var \$Quantity=1;     function Quantity()     {         //Add PHP code here     } }</pre>
Quantity				
Quantity:Integer=1				
Quantity()				

表 4.9 UML 類別圖 Cart (PIM)轉換至程式樣板(PSM)

Cart	
類別圖(PIM)	程式樣板(PSM)
<div> <div>Cart</div> <div> New_order_item()  Set_order_quantity()  Conut_total_cost()  Delete_order_item()  Confirm_order()  Add_product()  Delete product() </div> </div>	<pre> class Cart {     Var \$theProductType[];     var \$theOrder;     var \$theEmployee     function New_order_item()     {         //Add PHP code here     }     function Set_order_quantity()     {         //Add PHP code here     }     function Count_total_cost()     {         //Add PHP code here     }     function Delete_order_item()     {         //Add PHP code here     }     function Confirm_order()     {         //Add PHP code here     }     function Add_product()     {         //Add PHP code here     }     function Delete_product()     {         //Add PHP code here     } } </pre>



### 4.3 應用程式 PSM 轉 Code

本節將說明如何由 PSM 轉換成 Code，同樣的舉 Employee 的轉換做說明。在上一節將 PIM 轉為程式樣板(PSM)之後，已經可以看出 PHP 程式語法的雛形，接下來是要將程式樣板轉換為實際可以執行的 PHP 程式語言(CODE)。目前仍無相關工具可以使用，其轉換過程需要人工作業，依照 MDA 轉換規則轉換之後，結果如表 4.10。其餘的類別轉換之結果如表 4.11, 4.12, 4.13 與 4.14。

表 4.10 Employee 程式樣板(PSM)轉換至 PHP 程式(CODE)

Employee	
程式樣板(PSM)	PHP 程式(Code)
<pre>class Employee {     var \$ID;     var \$Name;     var \$Dept;     var \$Tel;     var \$Email;     function Employee()     {         //Add PHP code here     } }</pre>	<pre>class Employee {     var \$ID;     var \$Name;     var \$Dept;     var \$Tel;     var \$Email;     function Employee(\$ID)     {         settype(\$this-&gt;\$ID,"String")         settype(\$this-&gt;\$Name,"String")         settype(\$this-&gt;\$Dept,"String")         settype(\$this-&gt;\$Tel,"String")         settype(\$this-&gt;\$Email,"String")     } }</pre>



表 4.11 Product 程式樣板(PSM)轉換至 PHP 程式(CODE)

Product	
程式樣板(PSM)	PHP 程式(Code)
<pre> class Product {     var \$P_ID;     var \$P_Name;     var \$Unit;     var \$Price;     var \$StartDate;     var \$EndDate;     function Product ()     {         //Add PHP code here     }     function Show_Detail()     {         //Add PHP code here     } } </pre>	<pre> class Product {     var \$P_ID;     var \$P_Name;     var \$Unit;     var \$Price;     var \$StartDate;     var \$EndDate;     function Product (\$P_ID)     {         settype(\$this-&gt;\$P_ID,"String")         settype(\$this-&gt;\$P_Name,"String")         settype(\$this-&gt;\$Unit,"String")         settype(\$this-&gt;\$Price,"String")         settype(\$this-&gt;\$StartDate,"String")         settype(\$this-&gt;\$EndDate,"String")     }     function Show_Detail (\$P_ID)     {         \$handle=         mysql_connect(\$server,\$user,\$password);         \$result=mysql_db_query("\$db","select *         from product where P_ID='         \$P_ID_number'");         \$row=mysql_fetch_array(\$result);          echo '編號:.'.\$row['P_ID'];         echo '品名:.'.\$row['P_Name'];         echo '單位:.'.\$row['Unit'];         echo '價錢:.'.\$row['Price']; echo         '起:.'.\$row['StartDate'];         echo '迄:.'.\$row['EndDate'];     } } </pre>

表 4.12 Order 程式樣板(PSM)轉換至 PHP 程式(CODE)

Order	
程式樣板(PSM)	PHP 程式(Code)
<pre> class Order {     var \$Order_ID;     var \$Order_date;     var \$Total_cost;     function Order()     {         //Add PHP code here     }      function Increase_order()     {         //Add PHP code here     } } </pre>	<pre> Class Order {     var \$Order_ID;     var \$Order_date;     var \$Total_cost;     function Order(\$Cart,\$ID)     {         settype(\$this-&gt;\$Order_ID,"String")         settype(\$this-&gt;\$Order_date,"String")         settype(\$this-&gt;\$Total_cost,"String")     }     function Increase_order()     {         mysql_connect(\$server,\$user,\$password);         mysql_select_db ("db");         \$query = "INSERT INTO Order         (Order_ID, Order_date , Total_cost , ID)         VALUES         (",NOW() ,",         ".\$this-&gt;Cart-&gt;Count_total_cost() .",         ".\$this-&gt;ID.")";         mysql_query(\$query);         \$Order_ID = mysql_insert_id();         return \$Order_ID;     } } </pre>

表 4.13 Quantity 程式樣板(PSM)轉換至 PHP 程式(CODE)

Quantity	
程式樣板(PSM)	PHP 程式(Code)
<pre> class Quantity {     var \$Quantity = 1;     function Quantity()     {         //Add PHP code here     } } </pre>	<pre> class Quantity {     var \$Quantity = 1;     function Quantity(\$Quantity)     {         \$this-&gt;Quantity = \$Quantity     } } </pre>

表 4.14 Cart 程式樣板(PSM)轉換至 PHP 程式(CODE)

Cart	
程式樣板(PSM)	PHP 程式(Code)
<pre> class Cart {     Var \$theProductType[];     var \$theOrder;     var \$theEmployee     function New_order_item()     {         //Add PHP code here     }      function Set_order_quantity()     {         //Add PHP code here     }      function Count_total_cost()     {         //Add PHP code here     }      function Delete_order_item()     {         //Add PHP code here     } } </pre>	<pre> class Cart {     Var \$theProductType = array();     var \$theOrder;     var \$theEmployee     function New_order_item()     {         \$handel=         mysql_connect(\$server,\$user,\$password);         \$sql = "SELECT         P_ID,P_Name,Price FROM Product         WHERE P_ID ='P_ID'";         \$search =         mysql_db_query("db",\$sql);         list(\$P_ID,\$P_ame,\$Price) =         mysql_fetch_array(\$search);         \$new_P_ID =         array(\$P_ID=&gt;\$P_ID);         \$this-&gt;P_ID =         array_merge(\$this-&gt;P_ID,\$newP_I         D);         \$new_P_Name =         array(\$P_ID=&gt;P_Name);     } } </pre>

表 4.14 Cart 程式樣板(PSM)轉換至 PHP 程式(CODE) (續)

Cart	
程式樣板(PSM)	PHP 程式(Code)
<pre> function Confirm_order() {     //Add PHP code here }  function Add_product() {     //Add PHP code here }  function Delete_product() {     //Add PHP code here }         </pre>	<pre> \$this-&gt;P_Name = array_merge(\$this-&gt;P_N,\$new_P_Name ); \$new_UnitPrice = array(\$P_ID=&gt;\$UnitPrice); \$this-&gt;UnitPrice = array_merge(\$this-&gt;UnitPrice,\$ UnitPrice); \$new_quantity = array(\$P_ID=&gt;\$quantity); \$new_quantity = array(\$P_ID=&gt;\$Quantity); \$this-&gt;quantity = array_merge(\$this-&gt;Quantity,\$new_qua ntity); \$Price = \$Quantity * \$UnitPrice; \$new_price = array(\$P_ID=&gt;\$Price); \$this-&gt;Price = array_merge(\$this-&gt;price,\$new_price); \$this-&gt;total_cost = array_sum(\$this-&gt;Price); }  function Set_order_quantity(\$P_ID,\$Quantity) {     \$this-&gt;Quantity[\$P_ID] = \$Quantity;     \$this-&gt;Price[\$P_ID] = \$quantity *     \$this-&gt;UnitPrice[\$P_ID];     \$this-&gt;total_cost =     array_sum(\$this-&gt;Price); }  function Count_total_cost() {     return \$this-&gt;total_cost; }  function Delete_order_item() {     unset(\$this-&gt;P_ID[\$P_ID]);     unset(\$this-&gt;P_Name[\$P_ID]);     unset(\$this-&gt;UnitPrice[\$P_ID]);     unset(\$this-&gt;Quantity[\$P_ID]);     unset(\$this-&gt;Price[\$P_ID]);     \$this-&gt;Total_cost =     array_sum(\$this-&gt;Price);         </pre>



表 4.14 Cart 程式樣板(PSM)轉換至 PHP 程式(CODE) (續)

Cart	
程式樣板(PSM)	PHP 程式(Code)
	<pre> } function Confirm_order() {     include_once("Order.php"); } function Add_product() { } function Delete_product() { } } </pre>

#### 4.4 雛型系統展示與說明

本研究的 MDA 基本架構經過下列的三個步驟：建立 PIM 、PIM 轉換 PSM 及 PSM 轉換 Coding，經過上述的轉換步驟之後，實作完成一個集中採購網站。在開發的過程中，系統的 PIM 直接投射到實作平台 PHP 上的 PSM 後，再轉換產出完整 PHP 程式碼，程式樣板 PSM 可以由工具產出(詳見下節介紹)，的確可以達到 MDA 所強調的優點，例如減短系統開發的週期，可節省人力、成本並增加系統的品質。

茲將實作之雛形系統功能說明如下：

1. 進入系統之後可以看到需求清單(圖 4.2)，若想加訂只要輸入數量與員工帳號即可。
2. 如無需要之品項，可以自行新增項目(圖 4.3)。
3. 採購人員可至歷史紀錄中，查閱需求進行採購。



Purchase List

Requirement List New Requirement History List About

History List

#	ID	Item	Unit	Price	Start Date	End Date	Count
1	AD017	AD017	set	6	2007-01-23	2007-01-23	4
2	AD016	AD016	set	13	2007-01-23	2007-01-23	13
3	AD015	AD015	piece	5	2007-01-23	2007-01-23	100
4	AD014	AD014	set	10	2007-01-23	2007-01-23	15
5	AD013	AD013	set	500	2007-01-23	2007-01-23	10
6	AD012	AD012	can	24500	2007-01-23	2007-01-23	1
7	AD011	AD011	set	3	2007-01-23	2007-01-23	1000
8	AD010	AD010	piece	134	2007-01-23	2007-01-23	101
9	AD009	AD009	set	3500	2007-01-23	2007-01-23	7
10	AD006	AD006	set	234	2006-11-10	2006-11-10	2
11	AD005	AD005_Name	set	1000	2006-11-11	2007-01-01	1
12	AD004	AD004Name	set	600	2006-11-01	2006-12-10	1
13	AD003	AD003Name	set	1000	2006-09-09	2006-09-20	1
14	AD002	AD002Name	set	200000	2006-09-09	2006-09-30	1
15	AD001	AD001Name	set	300	2006-09-01	2006-09-10	12

圖 4.2 集中採購物網站採購需求清單畫面

Purchase List

Requirement List New Requirement History List About

New Request

ID:

Item:

Unit:

Price:

Start Date:  (yyyy mm dd)

End Date:  (yyyy-mm-dd)

User ID:

Quantity:

圖 4.3 集中採購物網站新增採購品項畫面

## 4.5 PSM to PIM 自動轉換工具程式

本研究針對 PHP 程式開發平台，提出之符合 MDA 轉換之 CASE 工具雛型，目前可以做 PIM 到 PSM 之轉換，不僅可以節省程式設計人員自行轉換之時間與人力，加快系統開發流程，更可以讓不熟悉 MDA 架構的人也可以享受 MDA 帶來的好處，避免人為錯誤，確保程式品質。

PIM 到 PSM 之轉換判斷流程如圖 4.4，首先確認 PIM UML 類別圖之 class name，先就 class name 轉換並置於最外層。接下來判斷是否有 variable，如果有就進行 variable 的轉換。最後再判斷是否有 function，如果有就進行 function 的轉換。

而 class, variable 及 function 的轉換規則整理如表 4.15。

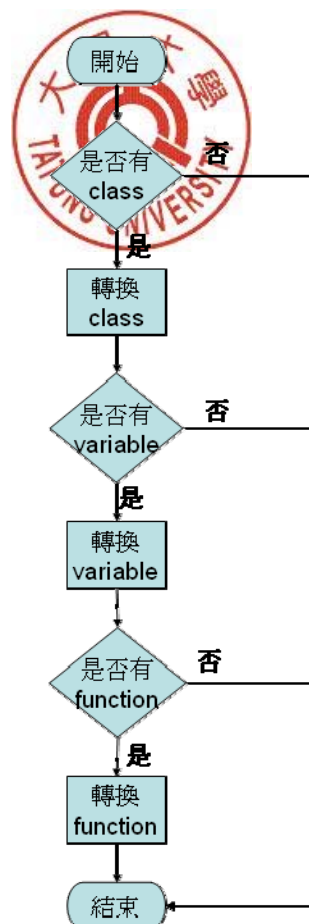


圖 4.4 PSM to PIM 轉換判斷流程

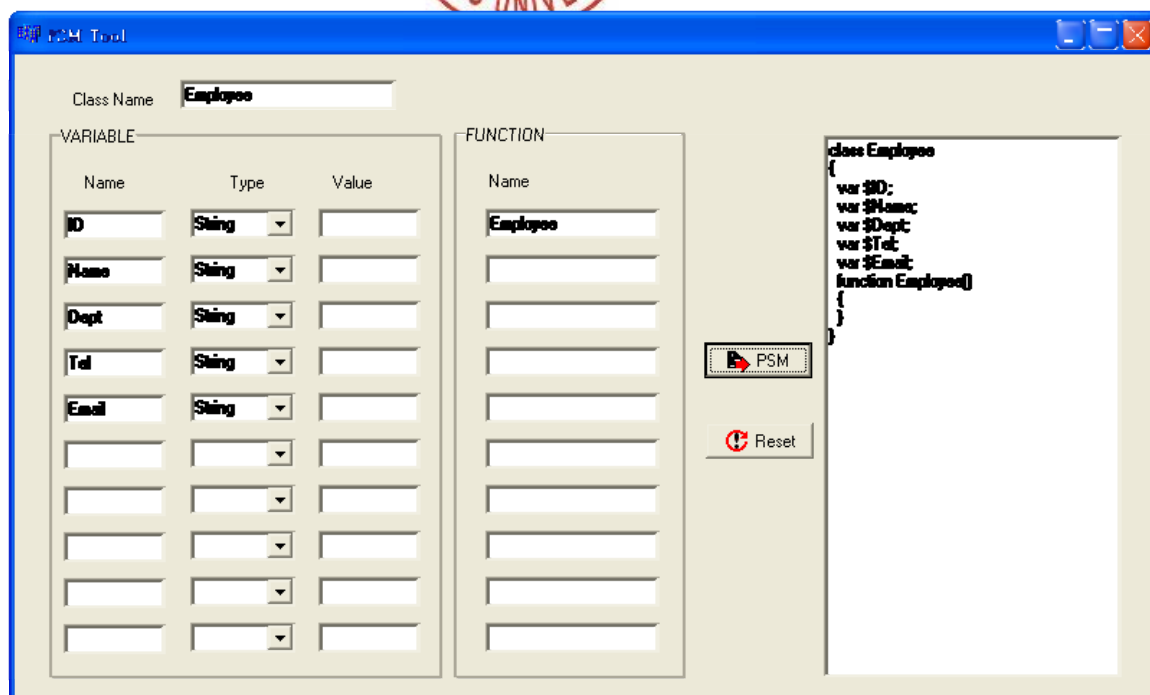


表 4.15 UML 類別圖(PIM)轉換成 PHP 程式樣板(PSM)轉換規則

UML 類別圖(PIM)	PHP 程式樣板(PSM)
Class name	class <u>Class name</u> { }
Variable name	var <u>\$Variable name</u> ;
Function name	function <u>Function name</u> () { }

本研究提供的轉換程式可以將 UML Class 圖，自動轉換成 PHP 程式樣板，其操作方式說明如下：

1. 打開轉檔介面，將 class 圖的 variable 及 function 輸入。
2. 按下 PSM 按鈕，即可快速產生 PHP 程式樣版。



The screenshot shows a software window titled "PSM Tool". It contains a "Class Name" field with the value "Employee". Below this, there are two main sections: "VARIABLE" and "FUNCTION".

The "VARIABLE" section is a table with three columns: "Name", "Type", and "Value". It contains five rows of input fields. The first row has "ID" in the Name field, "String" in the Type dropdown, and an empty Value field. The subsequent rows have "Name", "Dept", "Tel", and "Email" in the Name field, all with "String" in the Type dropdown and empty Value fields. There are three more empty rows below.

The "FUNCTION" section has a "Name" field with the value "Employee" and several empty input fields below it.

To the right of these sections are two buttons: "PSM" (with a right-pointing arrow) and "Reset" (with a circular arrow). Further to the right is a text area displaying the generated PHP code:

```
class Employee
{
    var $ID;
    var $Name;
    var $Dept;
    var $Tel;
    var $Email;
    function Employee()
    {
    }
}
```

圖 4.5 PSM to PIM 自動轉換工具程式畫面

## 第五章 結論及建議

本章結論與建議將分為兩部分，第一節說明本論文之研究成果與貢獻，第二節提出未來相關研究之建議。

### 5.1 結論

本研究的個案環境是以 MySQL 5 版本為物件關聯資料庫，並以 PHP 為軟體程式平台，最後整理出應用系統開發中以 UML 進行系統分析塑模的工具，以及由類別圖轉換成 PIM 及 PSM 的程式樣版。並藉由「集中採購網站」個案的案例實作，將整個模型驅動架構三部份的塑模及轉換作一個實證範例。

本研究最主要的研究成果如下：

1. 提出由 UML 系統分析、建立類別圖轉換至 MDA 中 PIM、PSM 及 Code 各階段的原則與步驟。
2. 於實作個案中，依循模型驅動架構的模式將包含應用程式及資料庫部份由 PIM 到 PSM 到 Code 的塑模及轉換作了完整的實作驗證。
3. 提出了 PHP 程式符合 MDA 轉換之 CASE 工具雛型，可以協助由 PIM 至 PSM 轉換時程式樣版之快速建立。

更具體而言本研究主要的貢獻如下：

1. 提供使用 UML 進行應用系統開發時，系統分析與設計的塑模步驟，可供 PHP 使用者進行系統分析與設計之參考。
2. 整理出進行應用系統開發時，依循模型驅動架構由 PIM 到 PSM 到 Code 三個層次中包含資料庫及應用程式詳細作業程序，可以提供運用 UML 進行軟體開發工作的人員參考，以提供較快速的實作開發程序。



3. 本研究以「集中採購網站」的範例來進行模型驅動架構模式的整體系統開發，可提供業界應用 Open Source 環境下進行應用系統開發之參考。
4. 針對 PHP 程式開發平台有關資料庫及應用程式，提出了符合 MDA 轉換之 CASE 工具雛型，可供後續研究參考。
5. 本研究中有關模型驅動架構的塑模及轉換方法，以及應用個案的實作過程，可以作為學習模型驅動架構及 UML 系統開發方式的參考資料，以提供新學習者清楚的應用程序，能更容易地學習應用相關的系統開發模式。

## 5.2 未來研究建議

在本研究的過程中，發現在 UML 的塑模及 MDA 的轉換上，仍有許多可進一步深入探討的主題。

1. 在本研究的「集中採購網站」實作案例中，對於在使用者介面轉換之方式並無探討，因此，在應用系統開發過程中使用者介面轉換要如何進行，是一個有待研究的議題。
2. 本研究是由類別圖轉換到程式碼的研究，運用了類別圖及循序圖作為輔助分析轉換，對於是否使用其他圖類協助進行 UML 的轉換分析，則是另一項有待探討的題目。
3. 針對 PHP 程式開發平台，提出之符合 MDA 轉換之 CASE 工具雛型，僅能做 PIM 到 PSM 簡單之轉換，而 PSM 到 Code 的轉換，仍有相當大的落差，如何在 UML 塑模及轉換過程中，提高程式樣板的程式碼可供直接成為最終程式之比例，節省人工撰寫程式的時間，提高系統開發的效率，是未來極待研究的課題。

## 參考文獻

- [1] 吳仁和，「物件導向系統分析與設計—結合MDA與UML」，智勝文化，2005。
- [2] 李佳璋，「利用模型驅動架構技術所發展的工作流程架構」，私立東海大學資訊工程與科學研究所碩士論文，2003年10月。
- [3] Grady Booch, James Rumbaugh , and Ivar Jacobson著，張裕益譯，「UML使用手冊」，博碩文化，2001。
- [4] 陳鏞阡，「基於MDA建立網路服務知識本體化之元模型」，台北大學資訊管理研究所碩士論文，2006年7月。
- [5] 解燕豪，「應用模式導向架構技術進行系統開發之研究-以知識管理系統為例」，國立臺北大學資訊管理研究所碩士論文，2004年7月。
- [6] Martin Fowler著，趙光正譯，「UML 精華第三版—標準物件模型語言」，基峰資訊，2004。
- [7] 摩理資訊，「網路常見專有名詞」，<http://www.askmorris.org>，存取時間：2006年6月。
- [8] 模式驅動架構網，<http://www.mdasky.com>，存取時間：2006年6月。
- [9] HTTP Server Project, “Apache Web Server Project,” available at <http://www.apache.org>, access time: 2006 June.
- [10] Frankel, D. S., *Model Driven Architecture: Applying MDA to Enterprise computing*, Canada: Wiley Publishing, Inc, 2003.
- [11] Nadarajan, R. and Manavalan, R., “Design of Platform Specific Model and Transformation to Platform Independent Model for Persistent Services,” Department of Mathematics and computer Applications PSG College of Technology, 2004 June.

[12] OMG, “Model Driven Architecture (MDA)”, available at <http://www.omg.org>,  
access time: 2006 June.

[13] vbird web site, “Web Server Introduction,” available at <http://www.vbird.org>,  
access time: 2006 June.



# **SYSTEM DEVELOPMENT BASED ON MODEL-DRIVEN ARCHITECTURE—AN EXAMPLE OF CONSTRUCTING A CENTRALIZED PURCHASING WEBSITE WITH PHP**

## **ABSTRACT**

This paper aims at a study of system development based on the model-driven architecture (MDA). The MDA To utilizing the unified modeling language (UML) as an analytical tool for the system, comply and implement the MDA architecture, and then transform the Class Diagram which is created by the UML into the template related to PHP programming language. Furthermore, the said template would be transformed into the physically executable codes, and finally, the development method of this system will be utilized to construct a centralized purchasing website in an applicable implementation. Additionally, while referring to the transforming process of PIM-to-PSM, we address the relevant method and tools applicable to rapidly transform the UML Class Diagram into PHP template so as to dramatically save the time necessary for thoroughly understanding such MDA architecture during the artificial transformation process as well as to avoid the likely human errors.

Keyword: Model Driven Architecture, MDA, UML, PIM, PSM, PHP

## **INTRODUCTION**

The information system is commonly used to assist an enterprise organization not only in resolving problems related to the management, decision and information processing, but also in improving the operational efficiency and effectiveness. However, there are compatible problems among various information systems of the legacy and new systems. The system logic and platform information are used to be mingled together during modeling process, which hamper the model reusability when implementation platform changed.

The Object management group (OMG) addressed the model-driven architecture (MDA) in 2001 (OMG, 2001) providing a good solution for the development of the diversified platforms and applications nowadays. The primary goal of MDA is interoperability between tools and the long-term standardization of models for popular application domains. In the MDA architecture, there are two kinds of key models thereof, one of them is the Platform Independent Model (PIM) that merely describes the abstract logic of the system, and another model is used to project the system's abstract logic onto an implementation platform, which is called the Platform Specific Model (PSM).

In addition, MDA describes the mapping rules of models transformation. Each transformation needs clear mapping definition. The developing constructs on MDA architecture firstly would model the abstract logic into PIM, and then project PSM to a certain implementation platform with the PIM-to-PSM mapping, which afterward would create the system code through the mapping of PSM-to-Code. All these relevant mapping operations can be done artificially, or done using tools with the semi-automatic or even full-automatic transformation. Currently, there are many MDA compliant development tools on the market, such as OptimalJ, Codagen Architect, Sossy Inc Modeller,

Transformation Engine, etc. to name a few (Lano, 2005). On the other hand, a famous CASE tool is Unified Modeling Language (UML) (OMG, 2003) which is a kind of properly defined, easily expressible, functionally powerful and generally applicable type of modeling language to supporting the object-oriented analysis and design from the requirement analysis to the overall developing process of the software (Booch, Rumbaugh, & Jacobson, 2001).

Although MDA has been introduced for years and its theory architecture is favorable to the developer with many benefits, there are some misinterpretations in practice of certain software industries. The prior studies related to MDA system development were mostly aiming at the utilization of Java MDA development tool (e.g., OptimalJ) on top of the Java platform. However, most of the services, such as that of the dedicated server hosting or the virtual hosts, are provided with the Open Source environment (e.g., Apache+PHP+MySQL) and very few are with the Java platform. Therefore, the objective of this study is to present MDA-based system development on the Open Source environment.

In this paper, we will demonstrate how to produce a PIM starting from UML system analysis as well as how to accomplish the mapping of PIM-to-PSM transformation and then the final execution code of PHP. To validate the feasibility of our MDA-based system development, we shall establish a centralized purchasing website focusing on PHP realization under an Open Source environment and investigate some related problems while using the MDA under the Open Source environment (Apache+PHP+MySQL in this study). Besides, since there is no MDA compliant transformation tool available for PHP, this study works up a method and propose a tool that can fast transform the UML Class Diagram into a PHP code template. This saves the time needed in understanding the MDA mapping rule during artificial transformation process and avoids manual transformation error that the code quality can be assured.

In the rest of this paper, we shall have a brief review of the related techniques describing the UML and the MDA architecture along with the introduction to the PHP developing environment. We then describe the case example of this study and its system analysis. System implementation with PHP as well as the method and tool for PIM-to-PSM transformation will be presented. Finally, conclusion of this study is made including future work suggestions.

### **BRIEF REVIEW OF RELATED TECHNIQUES**

The UML has integrated the new thought, methodology and technology of software engineering field; without limitation to support the analysis and design of object-oriented, instead it even can support overall software development process starting from the requirement analysis. Therefore, it is a kind of properly defined, easily expressible, functionally powerful and generally applicable type of modeling language. In the MDA architecture, UML is applicable to play a role of system analysis, which is able to specifically define various features of PIM.



## **UML (Unified Modeling Language)**

The UML is the result of a successful unification of the many object-oriented analysis and design notations which arose during the 1980s and 1990s. In 1994 work began on unifying the widely-used OMT (Object Modeling Technique) and Booch et al's methods into what became UML, with OOSE also integrated in 1995 (Booch, Rumbaugh, & Jacobson, 2001). From version 1.1 UML was endorsed by the Object Management Group (OMG), representing many companies and organizations, as a standard for object-oriented analysis and design, and it has now become the primary notation in this field. A major revision of UML, to version 2.0, was published in 2004 (Lano, 2005).

UML 2.0 defines thirteen types of diagrams, divided into three categories where six diagram types represent static application structure, three represent general types of behavior, and four represent different aspects of interactions (OMG, 2003):

- Structure Diagrams include the Class Diagram, Object Diagram, Component Diagram, Composite Structure Diagram, Package Diagram, and Deployment Diagram.
- Behavior Diagrams include the Use Case Diagram (used by some methodologies during requirements gathering); Activity Diagram, and State Machine Diagram.
- Interaction Diagrams, all derived from the more general Behavior Diagram, include the Sequence Diagram, Communication Diagram, Timing Diagram, and Interaction Overview Diagram.

## **Model Driven Architecture (MDA)**

The Unified Modeling Language (UML) organized by the Object Management Group (OMG) has specified the standard Model Notations which is utilized to describe the system design by the system development personnel. While modeling the system with UML, except reducing the communication misunderstandings by means of the common communication symbols, the model information also can be used to identify the relevant designs to make sure whether it meets the system requirements or not; furthermore, it will greatly facilitate the successive system maintenance, or assist the new project members to have a better understanding on the current systematic design. However, it is a little bit regretful that UML documentation is merely acting as a reference to the developing personnel upon the system development period; so it is helplessly to promote the speed of system development.

Thus, OMG addressed the software development standard of Model-driven Architecture (MDA) in 2001. In the MDA architecture, there are two kinds of key models thereof, one of them is the Platform Independent Model (PIM) that merely describes the abstract logic of the system, and another model is used to project the system's abstract logic onto an implementation platform, which is called the Platform Specific Model (PSM). While modeling the system in the past, it is usually modeling the PSM information directly; however, it will result in a difficulty for the system to transform the implementation platform in the future. Since that the system's abstract logic has been tied closely with the information of the implementation platform already, and in this way the repetitious usage of model information will become impossible, it is required to produce PSM again according to the new implementation platform as well as the wastage of resources. This is an actual reason why the MDA would like to independently extract the abstract logic from the system and make it as the PIM.

In addition to the definition of two models, MDA has also described the mappings of four models. These four mappings comprise (Lee, 2003):

- PIM to PIM: PIM-to-PIM: This mapping takes place while defining the PIM of the system. It is not an easy work at all to design a very stable system, which needs to be constructed by a progressive-based method. And the implicit meaning of this progressive process is the PIM-to-PIM transformation.
- PIM-to-PSM: This mapping occurs when the PIM of the system has already been accomplished, and is ready to be projected onto one certain implementation platform.
- PSM-to-PSM: This mapping takes place while defining the PSM of the system. Because each implementation platforms may have individual characteristic, so we might need to have a fine adjustment the PSM projected by the PIM so as to reveal the advantage of such platform. The action of this fine adjustment belongs to the PSM-to-PSM transformation.
- PSM-to-PIM: This mapping takes place while refining a certain Legacy System. The application system development for an enterprise often can only be integrated with the Legacy System, and a common method generally utilizes Adapting Method to set up a communication interface, that is, setting up a Message Oriented Middleware (MOM) server in-between new system and old system for information exchange. Additionally, there is another integration method, that is, utilizing the reverse engineering to reversely transform the PIM which is created by the old system, and after modification, it could be re-implemented to a new platform again. This particular process is PSM-to-PIM transformation.

Figure 1 shows the clear mapping steps (Wu, 2005). Note that the model transformations from PIM to PSM and from PSM to the Code can be done artificially, or done with CASE tools with semiautomatic or even full-automatic transformation. At present MDA Tools may be categorized into Open Source Tools and Commercial Tools as we listed some in Table 1.

The advantages of using tools include shortening the cycle of system development, improved the system quality, reduced cost of system development, reduced difficulty in the operation of heterogeneous platform transformation, and increased repetitive usage of models.



**Figure 1: MDA mapping steps of PIM, PSM, and Code**

**Table 1: MDA Tools List**

Open Source Tools	Commercial Tools
MOFScript (Eclipse plugin)	ArcStyler (MDA tool from Interactive Objects)
MTF (IBM Model Transformation Framework)	MCC (Model Component Compiler, J2EE)
ATL (Eclipse plugins)	OptimalJ (PSM transformations, integrated UML tool)
MTL Engine (Integrates with Netbeans MDR and Eclipse EMF)	Xactium XMF Mosiac (model-based mapping, generation and execution tool)
ModFact (MOF Repository and QVT-like engine)	SosyInc (Modeler and Transformation Engine)
GMT (Generative Model Transformer)	Model-in-Action (code generation and model to model transformation)
KMF (Kent Modelling Framework)	MetaEdit+ (modeling and metamodeling tool)
OpenArchitectureWare (template-based generator framework)	MDWorkbench (text and model transformation toolset)
OpenMDX (integrates with several tools, J2EE, .Net)	iQgen 3.0 (template-based generator)
AndroMDA (J2EE code generation)	Rational Rose
XDoclet (attribute based code generation tool for J2EE)	Power Designer
Middlegen (database driven code generator)	Together

### PHP Develop Environment

PHP (recursive acronym for “PHP: Hypertext Preprocessor”) is a widely-used Open Source general-purpose scripting language that is especially suited for Web development and can be embedded into HTML. PHP is mainly focused on server-side scripting, so you can do anything any other CGI program can do, such as collecting form data, generating dynamic page content, or sending and receiving cookies. But PHP can do much more. There are three main areas where PHP scripts are used, server-side scripting, command-line scripting, and writing desktop applications.

The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows NT. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards.

On the other hand, MySQL, the most popular Open Source SQL database management system, is developed, distributed and supported by MySQL AB. MySQL AB is a commercial company, founded by the MySQL developers. It is a second generation Open Source company that unites Open Source values and methodology with a successful business model. Its database server is very fast, reliable, and easy to use. In addition, MySQL Server works in client/server or embedded systems.

## DESCRIPTION AND SYSTEM ANALYSIS OF THE CASE EXAMPLE

We now proceed to describe the case example and system analysis process in this study, as well as describe how to utilize the UML tool to construct the system models.

### Requirement Scenarios

A company usually performs the purchasing operation for some consumable materials; however, diverse purchasing method not only wastes the manpower, but also nonnegotiable with manufacturers for the sake of quantity. To resolve this problem, therefore it is necessary to draw up a plan for constructing a centralized purchasing website so as to accumulate all the requirements of users and also allow the purchasing personnel to purchase batch by batch.

The principle of the system construction is hoping to develop a new system under a condition which won't influence the current system at all, in addition, also hoping to have an across-platform function ability and the flexibility for future modification. Therefore, they select to develop the relevant system with the PHP language under the Open Source environment along with a consideration that it is no need to rewrite all related programs while proceeding the future modification. Then they decide to utilize the MDA architecture in order to preserve the features of current system, such as easy maintenance and modification.

### System Requirement Analysis

While a user addresses a purchasing requirement, if it has already existed in the system requirement list, then the system would add-on the required quantity directly. However, if such user's requirement does not exist in the system requirement list, then the said user can add a requirement by self, and meanwhile also record the purchasing items and the starting/closing time of such collective purchasing operation. The requirement analysis process is shown in Figure 2.

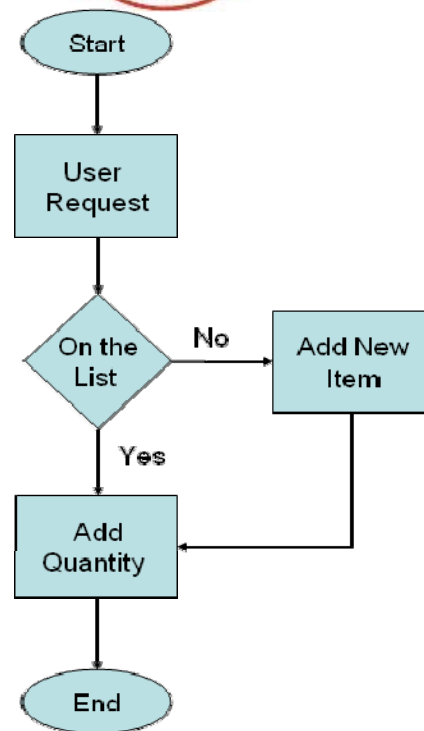


Figure 2: The requirement analysis process

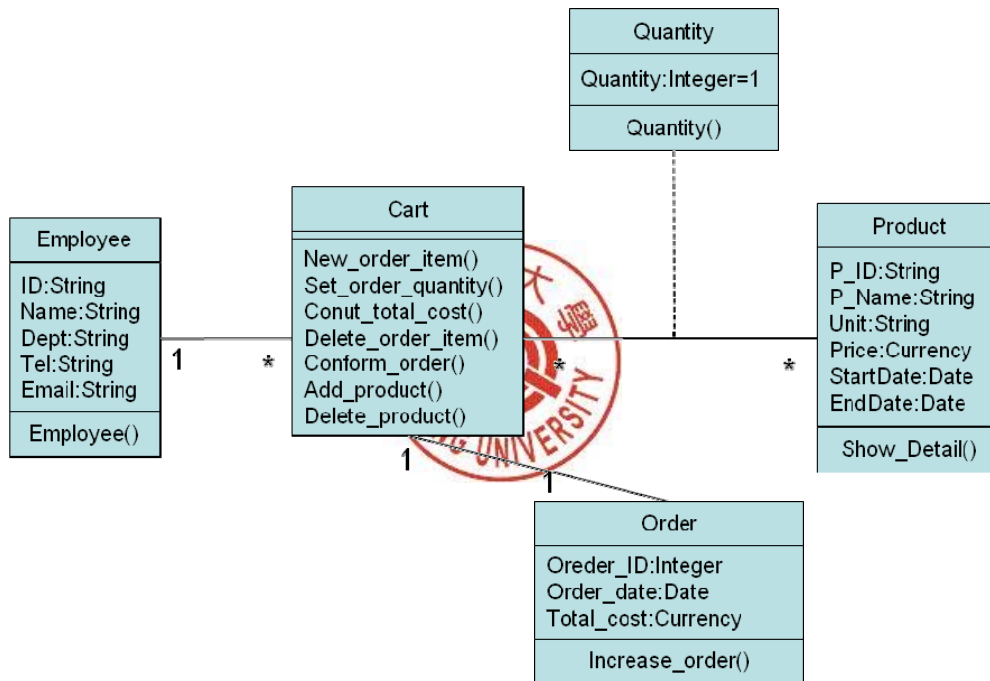
### Constructing System Model (PIM) with UML

In nine diagrams of UML, we adopt two of the most common diagrams as the analytical tools for the system. These two are Class diagram and Sequence Diagram, which we will describe respectively as follows:

#### ● Class Diagram

The Class Diagram describes the quiescent structure of category in the system. It not only defines the system Class, indicating the association between Classes, such as correlation, dependence, aggregation, etc., but also comprises the Class' internal architecture (the attribute and operation of the Class).

For example, Employee class has five attributes ID, Name, Dept, Tel and email. And there is one method Employee() in it. Employee class and Cart class is a one-many association. If all attributes, methods and correlation of the class have been accomplished, then we will acquire a complete series of Class Diagram as shown in Figure 3.



**Figure 3: Class diagram of case example**

#### ● Sequence Diagram

The Sequence Diagram represents the dynamic cooperative relationship among objects. It emphasizes the transferring sequence of the information of objects. The Collaboration Diagram describes the synergistic relationship among objects; however, similar to Sequence Diagram, the Collaboration Diagram can show the dynamic cooperative relationship among objects. In addition to the presentation of information exchange, Collaboration diagram also can show objects as well as the relationship thereof. If it needs to emphasize the time and sequence, then adopt the Sequence Diagram; but if it is important to emphasize the hierarchical relationship, then it is required to select the Collaboration Diagram.

If we utilize the Sequence Diagram to present this case example, there are altogether five objects, in the illustration. If someone would like to add a new purchasing item, the user

will see all the purchasing items after entering into the User's Interface, and after the verification procedure, ha can immediately add-on the purchasing items. Its related objects and Messages are organized in Figure 4.

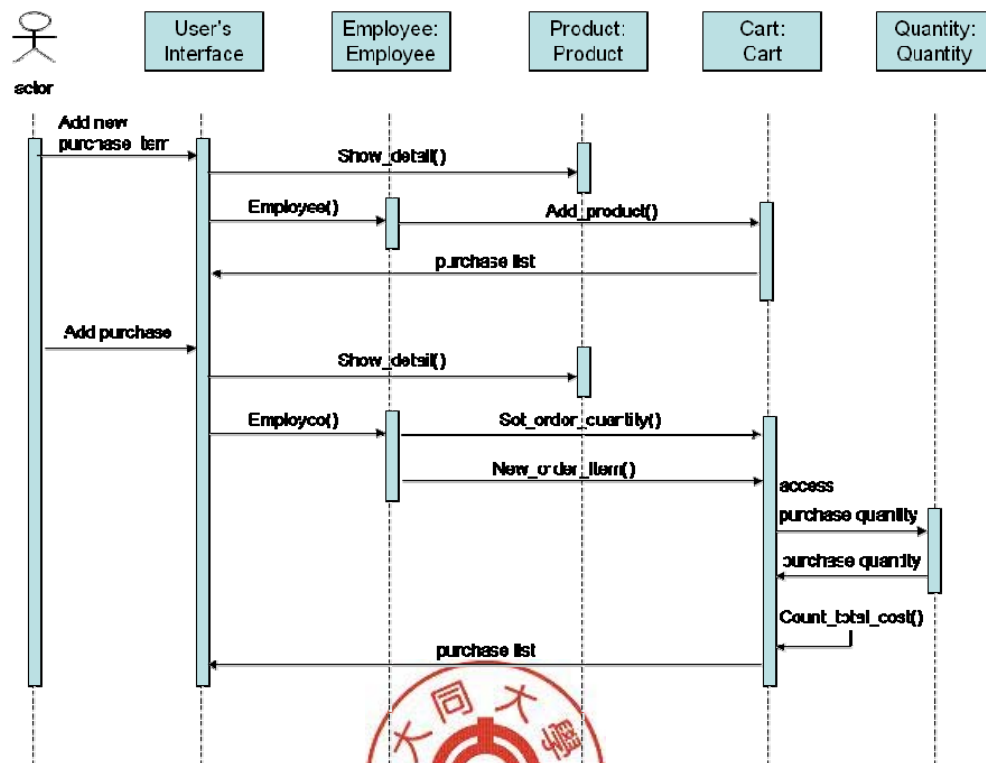


Figure 4: Sequence diagram of case example

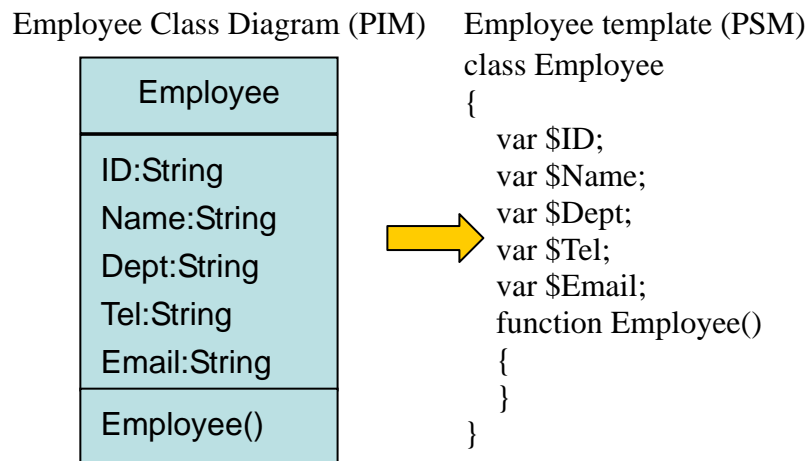
## SYSTEM IMPLEMENTATION AND PIM-TO-PSM TRANSFORM TOOL

We now present the implementation of MDA system, with an example to construct a centralized purchasing website with PHP languages, and identify the PIM-to-PSM transformation for MDA Class as well as the rules and relationship upon the process of Code transformation. In addition, the prototype system demonstration is utilized to expound both the operational interface and application of the present study development system along with the transformation tools for the PHP program languages in the process of PIM-to-PSM transformation.

### The Application Related to PIM-to-PSM Transformation

In this paper we only set forth the Employee Class as the descriptive example. Firstly, it is required to utilize the Class Diagram as the transformation basis in system analysis process, but the general transformation tools currently used are only covering the languages of Java, C++, .NET, etc., and at the time being there is no one has ever utilized any transformation tool related to PHP yet. However, the process of PIM-to-PSM translation without tool assistance can only be transformed step by step, wherein the primary operations comprising variable declaration and function definition, e.g. attributes ID, Name, Dept, Tel, Email and Employee method of the Employee Class; therefore, we do need a class which is able to comprise all variables and the function. The transformation result is as shown in Figure 5.

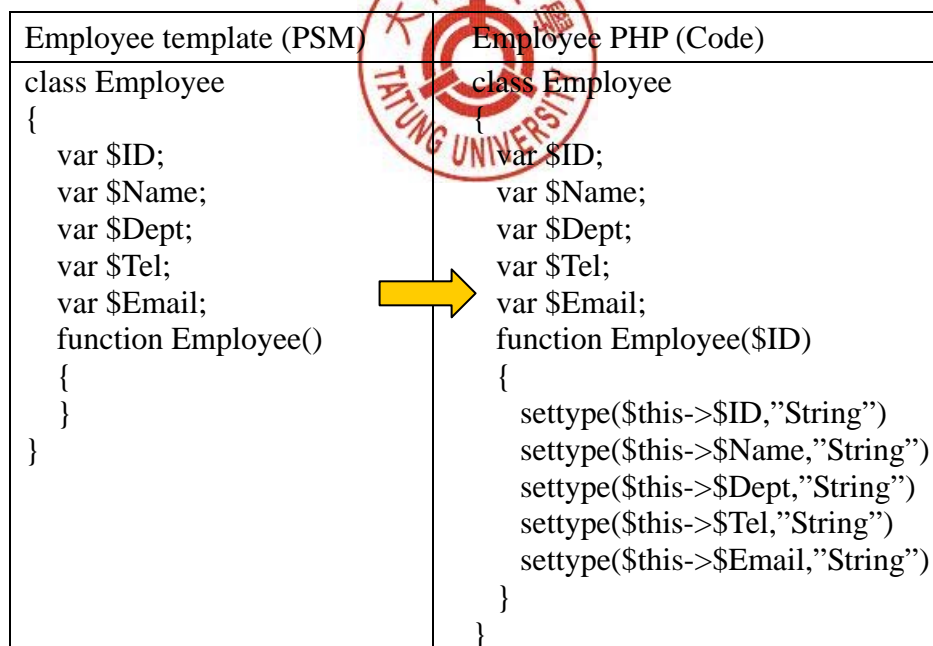




**Figure 5: Transfer UML class diagram (PIM) to template (PSM)**

### Transforming Template into PHP Code

After being transformed into template (PSM), it is available for us to see the embryo of PHP program syntax, and the next step is to transform the template into the PHP programming language (Code) that could be run physically. Due to the fact that there is no any available tool currently, its transformation process must be performed manually. The transformed result can be referenced as that shown in Figure 6.



**Figure 6: Transfer template (PSM) to PHP Code**

### Demonstration and description of the prototype system

The other Class Diagrams will be processed repeatedly with the above-mentioned method and will be transformed into the template (PSM) one by one, then retransformed again into PHP program (Code), along with the setting for MySQL database. In this way we will gradually accomplish the prototype of the centralized purchasing website system. Implementing a centralized purchasing requirement website according to the MDA

architecture is actually able to fulfill all advantages emphasized by MDA in the process of development. For instance, the system codes can be generated from the model so that it is possible to shorten the cycle of the system development. In addition, due to that MDA can be directly projected from the PIM of the system to the PSM on a certain implementation platform and then to create all related codes in order to save the manpower and cost while improving the system quality at the same time.

The functions of prototype system are described as follows (refer to Figures 7 and 8):

1. While entering into the system, the user can easily find out the detailed requirement lists; if he wants to add-on the order, he might just input the quantity along with employee's account number then.
2. If he finds there is no available items meet his requirement, he might add-on a new item by himself.
3. The purchasing personnel is allowed to check the historical records, and performs the purchasing operation based on the statistical results of such requirements.

**Figure 7: Demonstration of the prototype system--New Requirement**

#	ID	Item	Unit	Price	Start Date	End Date	Count
1	AD006	AD006	set	234	2006-11-10	2006-11-10	2
2	AD005	AD005_Name	set	1000	2006-11-11	2007-01-01	1
3	AD004	AD004Name	set	600	2006-11-01	2006-12-10	1
4	AD003	AD003Name	set	1000	2006-09-09	2006-09-20	1
5	AD002	AD002Name	set	200000	2006-09-09	2006-09-30	1
6	AD001	AD001Name	set	300	2006-09-01	2006-09-10	12

**Figure 8: Demonstration of the prototype system--History List**



### PSM-to-PIM Auto-Transformation Utilities

In order to shorten the time spent in the system development, this paper aims at the PHP program development platform, addressing the CASE tool which is applicable to the MDA transformation, and for the time being it is able to process the PIM-to-PSM transformation; thereat in addition to the saving of the programming time, for any one who is not too familiar to the MDA architecture will still be able to enjoy all the advantages of MDA. Furthermore, it is capable of reducing the human errors and hence ensuring the quality of the codes accordingly. The PSM to PIM mapping process is shown in Figure 9.

The class, variable and function of UML class diagram to PHP template mapping rules are shown in Table 2. The transforming program provided in this paper is applicable to transform the diagram of UML Class into the PHP template automatically. The operation is described as follows:

1. Open the file transformation interface.
2. Input both the variable and function of such class diagram.
3. Press the PSM button, then the PHP template will be generated at once.

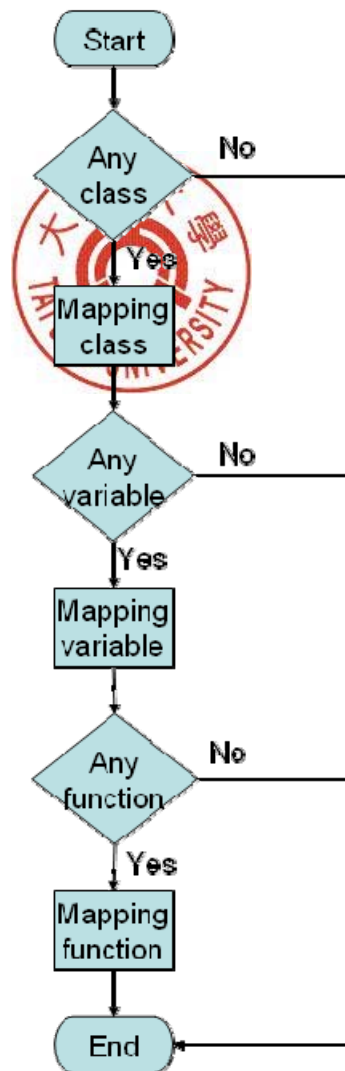
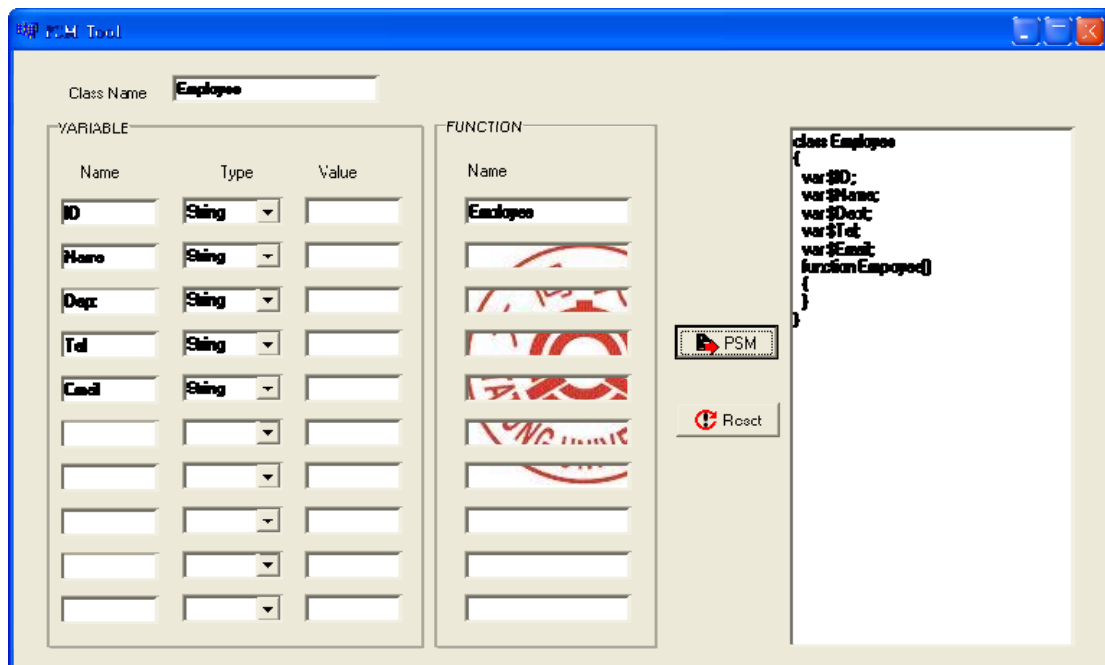


Figure 9: PIM to PSM mapping process

**Table 2: UML class diagram to PHP template mapping rules**

UML class diagram (PIM)	PHP template (PSM)
Class name	class <u>Class name</u> { }
Variable name	var <u>\$Variable name</u> ;
Function name	function <u>Function name</u> () { }



**Figure 10: Tools for transforming PIM to PSM**

## CONCLUSION

The environment for the individual case according to this study is utilizing the revision of MySQL 5 as the object relational database along with the PHP software program platform. We summarize the specific steps for modeling analysis of user's interface with UML upon the application system development, as well as program template for transforming the Class Diagram into user's interface and control class. At final, we set forth a case example implementation of a centralized purchasing website, wherein it is an identified MDA example comprising the three modeling processes and transformation of the entire MDA.

The primary results of this study comprise the following:

- Address the specific principles and procedures for each stage in MDA concerning UML system analysis, Class Diagram establishment and its transformation into PIM, PSM and Code.

- Complete a case implementation demonstration that summarizes detailed operational procedure in the MDA-based system development with database, application program and user interface, covering modeling and transformation from levels of PIM to PSM to Code.
- Provide CASE transformation tool for PHP program complying with the MDA transformation, which can assist to rapidly create template upon the PIM-to-PSM transformation process.

During the process of this study, it is found that there are still a lot of topics deserved further investigation on the UML modeling and the MDA model transformation. Thus, the following list some of our suggestions for future work:

- In the case example of a centralized purchasing website, there is no discussion related to the transformation method concerning user's interface transformation; therefore, how to proceed with the transformation of the user interface has become an expected issue.
- This study has aimed at the transformation from the Class Diagram to user's interface, wherein it is utilizing the Class Diagram and the Sequence Diagram as the assistance in the analysis of transformation process. Therefore, whether there are other kind of UML Diagrams can be used to assist such transformation analysis would become another interesting issue for future study.
- As to the PHP program development platform, we are addressing the CASE tool complying with the MDA transformation, which is merely applicable in the process of the PIM-to-PSM transformation. However, there is still a big gap in the process of the PSM-to-Code transformation, and hence how to upgrade the code of template to be directly used and save the manual programming time as well as promote the efficiency of system development in the process of UML modeling and the transformation should also be challenging.

## REFERENCES

- Booch, G., Rumbaugh, J., & Jacobson, I. 2001. *Unified Modeling Language User Guide*. Addison-Wesley Professional.
- HTTP Server Project, <http://www.apache.org/>
- Lano, K. 2005. *Advanced Systems Design with JAVA, UML and MDA*, Elsevier.
- Lee, J.-W. 2003. *A Workflow Solution Based on Model Driven Architecture Technology*. Unpublished master thesis, Computer Science and Information Engineering, Tunghai University, Taichung, Taiwan.
- MySQL 5.1 Reference Manual, <http://dev.mysql.com/doc/refman/5.1/en/>
- OMG. 2001. *Model Driven Architecture (MDA)*. OMG Document number ormsc/2001-07-01, <http://www.omg.org>.
- OMG. 2003. *Unified Modeling Language Specification*. OMG Document number formal/2003-03-01, <http://www.omg.org>.
- PHP Manual, <http://www.php.net/>.
- Wu, J.-H. 2005. *Object-Oriented Systems Analysis and Design: An MDA Approach with UML*. Taipei, Taiwan: Best-Wise Publishing Co., Ltd.