國立交通大學

資訊管理研究所

碩士論文

以 MVC 設計樣式為基礎的應用程式產生器

An Application Generator based on the MVC Design Pattern

研究生：程鼎元

指導教授：羅濟群　教授

中華民國九十四年六月

以 MVC 設計樣式為基礎的應用程式產生器

An Application Generator based on the MVC Design Pattern

研 究 生：程 鼎 元　　Student: Ding-Yuan Cheng

指導教授：羅 濟 群　　Advisor: Chi-Chun Lo

國立交通大學

資訊管理研究所

碩士論文

A Thesis

Submitted to Institute of Information Management

College of Management

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Business Administration

in

Information Management

June 2005

Hsinchu, Taiwan, the Republic of China

中華民國　九十四 年 六 月

以 MVC 設計樣式為基礎的應用程式產生器

研究生：程鼎元　　指導教授：羅濟群 老師

國立交通大學資訊管理研究所碩士班

中文摘要

由於網際網路的快速發展，使用 Web 的方式進行建置多層式系統成了最佳選擇，由於其透過一致的 Web 介面與標準統一使得建置成本低廉等因素，新一代資訊系統發展均以此基礎做為發展之環境。但是以 Web 為應用程式系統則仍舊有許多問題需要克服，例如，程式碼撰寫後因結構鬆散使得管理不易、程式控制碼本身與 HTML 標記交錯於程式頁面當中，造成撰寫時與日後維護上的困難度增加。

本論文提出的程式產生器的架構是以 Web 應用程式開發環境，利用 MVC 設計樣式為基礎，將程式邏輯與顯示畫面分離，進而建立良好的程式開發環境，所產生的系統將可以大幅度的縮短開發的時間與成本。希望利用 Web 開放式環境的特性，協助程式設計開發團隊能夠快速、有效率的建立資訊系統。利用儲存 Meta-data 的概念，將已開發過的程式碼與文件等資料，以結構化的方式儲存於資料庫中。

關鍵字：產生器、MVC、Web-Based

An Application Generator based on the MVC Design Pattern

Student: Ding-Yuan Cheng Advisor: Dr. Chi-Chun Lo

Institute of Information Management

Nation Chiao Tung University

Abstract

Because of the rapid development of Internet, it is the best choice to build up the multi-layer system by using Web-based architecture. By the Web interface and unified standard, the next generation information systems will be based on it as development environment. But there are still many problems solved by using Web-based application. For examples: the program code was too loose to manage, the program control code mixed with HTML tag in the web-pages, and so forth. They will increase the difficulty in programming and maintaining in the future. The purpose of this thesis is to present a Web-based application development architecture.

The thesis proposed the application generator which was web-based environment by using MVC design pattern. It can separate the logical control and the presentation in order to construct a well programming development environment. The generated system could reduce the design time and cost. With meta-data concept, we store the program code and system analysis document which were structured in database.

Keyword: Generator、MVC

誌 謝

終於到了寫誌謝的時候，對於要感謝的人實在太多，首先最感謝的是我的指導老師羅濟群老師，在就讀研究所求學的這兩年間，老師教導我受用無窮的是為人處事的道理；在撰寫論文的研究上，讓我學習到做學問嚴謹的態度，更感謝老師對我的論文細心修改與指正，讓我得以順利完成本篇論文，衷心的感謝老師引領我進入這個浩瀚的殿堂，這篇論文的完成更是另外一階段學習的開始。

我要感謝因胃癌逝去陪伴著我十五年的摯友吳政霆，在我人生中最黑暗的時刻，與我一起衝向光明，曾經全世界的人轉過身，只有他面對我；而今，全世界的人轉回身，他已百年身，無法伴他走這最後一程甚至於無法見到最後一面，這份讓我成長的挫敗將會如影隨行地督促著我不斷成長，我的感謝將和思念一樣滿溢。

再來我要感謝 Netlab 所有的成員，感謝博士班的俊龍學長以及俊傑學長們的指導與建議，感謝吉宇學長與 nickle 學長以及碩二同學們的幫助，讓我的研究生活更為充實，也感謝碩一學弟們，為實驗室帶來歡樂的氣氛。感謝一起修課與我分享新知的緒杰，無私分享精神讓我在各方面更加充實；感謝一直陪伴著我的好室友仁德與宗仁，在我心情低落時為我加油打氣，你們的扶持，讓我重新拾起奮鬥的精神，同時也是籃球場上最好的伙伴，這份革命精神一直存在我心中。還有那些曾經指導過我，但未提及的同學和學長們，謝謝你們。

最後，我要特別感謝我的家人，我深愛與深愛我的爸媽，一直默默的在背後支持著我，為我不停的付出，在我犯錯時包容我，在我徬徨時給我希望，若不是你們為我撐起這一片天，我無法這麼幸福地求學，讓我無後顧之憂，我所有微薄的成就都要歸功於你們，如果有任何錯誤與疏失，都是我個人的問題，謹記在心的是你們給我的恩惠，無以為報的是我將不斷的努力。

目　錄

圖　目　錄

表　目　錄

第一章、緒論

在本章裡先討論本論文的研究動機、研究目的、研究方法及後續各章的簡單介紹。

1.1 研究動機

面對資訊快速變遷的時代，資訊科技的演變更是快速，從終端機與大電腦集中式的架構到 Client-Server 的 Two-tier 演變成今日的 Multi-tier、Thin-client 的分散式架構，一方面也受網路的普及與低價電腦所賜，資訊系統的開發技術也隨之不斷進步，企業也由 Two-tier 的架構轉而以多層次架構開發資訊系統，開發的複雜度也隨之提高。

在進行系統發展時，要面對者許多的問題要克服，在前人累積許多的經驗後，卻有許多的專案發展時，最後的結果常常會面臨到無法順利完成或是提前終止，實際上受限於技術層面的例子卻是在少數。因此過去在發展系統時常會依循著傳統 SDLC 方法來進行，的確讓系統在發展設計時解決了許多問題，像是人性因素、使用者的問題...等等。在軟體開發時能夠以塑模[24][25]的概念來設計，可以很快的轉換所應用的領域。

系統設計與開發一直是從事資訊類工程師所面對最重要的技術挑戰，由於缺乏一個有效的設計方法與平台，使得設計人員常常以一行接著一行的撰寫程式，雖然透過物件導向與其他設計方法的提出，增進了設計時的效率，但是對於多人開發團隊時所產生的問題，卻不是能夠靠個人的優異技術能力可以克服，這不但使得資訊系統的開發與建置無法有效率的提升，也使得系統在開發時，設計人員間的資訊無法交流；而在系統設計完成後，經常由於時程的壓力，在設計文件的更新往往並未更新，往往造成日後的維護上的困難，如何能讓系統開發的效率增

加，解決上述的問題，這便是本論文要探討的主題。

1.2 研究目的

本篇論文之研究目的是參照現有系統開發技術，提出一個具有彈性與快速建置系統的方法，透過把程式應用邏輯 (或稱商業應用邏輯) 與網頁呈現 (Layout) 邏輯分離，即利用 MVC(Model－View－Controller)設計樣式[1][2][3]為基礎，讓系統建置者能夠利用這個設計樣式來設開發產生器，過去的研究指出使用 Domain Patterns 在 Web Application 架構下開發系統已有越來越多的技術提供支援，可大幅節省開發的時間與成本，因此本論文希望利用設計樣式的方法應用在產生器的系統中，能夠改善資訊系統設計的效率。

1.3 研究方法

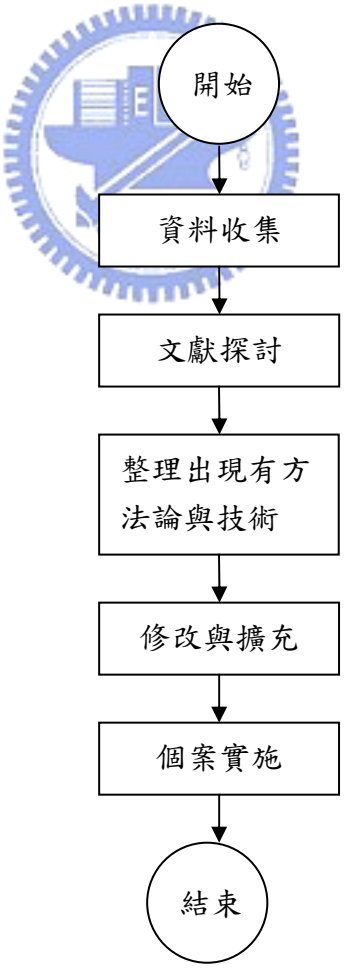

圖 1- 1 研究方法

1.4 章節說明

第二章將探討目前網路應用程式與相關技術，包括資訊系統開發架構，相關定義，並以範例說明。接著於第三章問題與改進構想中說明目前網路應用程式在設計時所面臨的問題，如何解決，並且提出改進的構想和模型；第四章實作與呈現實驗結果於第三章中所提出的模型；第五章、說明本研究所提出的設計方法與使用分析；第六章、結論與未來展望：作為本研究的總結與未來可行的研究方向。

第二章、文獻探討

在本章裡，主要介紹與說明與本論文主題相關的一些研究，一開始會先簡介應用系統產生器的概念，並說明傳統上程式產生器的種類，進而探討現有開發架構與方法；最後說明現有產生器的問題與分析目前採用技術的缺點。

2.1 應用系統產生器

產生器應用於系統的產生者，稱之為應用系統產生器，目前已有許多關於應用系統產生器系統的研究[4][14][15][16][17]，製作應用系統產生器雖然能夠有效提升軟體設計的產出，但是通常開發一個應用系統產生器時的成本遠高於直接開發個別的應用軟體，所以欲開發應用系統產生器時應考量到開發成本的因素[7]。採用產生器的優缺點如下(Cleaveland, 1998)[8]：

優點：

1.可降低程式碼的錯誤，能夠使得開發者專注於需求規格的錯誤。

2.可由非專業程式設計師建構與維護軟體。

3.易於建構與測試替代方案的雛型。

4.藉由產生器自動產生標準介面或輸出規格，易於將建構的過程標準化。

缺點：

1.單一個應用系統產生器僅能在一些特定環境中有效使用。

2.應用系統產生器很難建置，須要豐富的問題域知識、有能力去設計問題域中的原生元件、須仔細設計使用者介面與表達需求規格的語言。

3.發現應用系統產生器可用之處是很困難，且發現的時間點通常是在開發的末期，很難再激發動力而重新開發。

4.新技術的導入企業常引發管理上的問題。例如：如何且何時將應用系統產生器導入企業的流程中？誰才是考慮這些問題的人？開發軟體產生器的成本如何與未來所產生的效益作衡量？

由於基於以上的因素，所以Cleaveland 認為將應用系統產生器的概念與方法傳授給使用者，讓使用者發覺應用系統產生器可在工作流程中應用，由使用者自行開發應用系統產生器，因此應用系統產生器可落實使用者自建(End user computing)的概念。系統分析與設計的技術大概可以分為結構化、物件導向與元件導向三種，系統開發者也可以依產生器之不同特性進行系統分析與設計，若是產生器可以組裝出物件導向的應用系統，則可以利用物件導向的系統分析與設計的技術；若是以元件導向時的應用系統，須依元件導向的系統分析與設計的技術[14]；以結構化產生器的系統分析與設計則可見Wu等人[4]或張益嘉[15]。

程式產生器的分類依產生的方式可以分為主動型與被動型兩種。程式產生器分為三種[7]：(1)商業資料處理的應用系統產生器。(2)程式語言的文法與詞彙分析產生器。(3)CASE 工具中的程式產生器。依Jack Herrington[8]的分類則是以產生的方法分為六種(見2.1.1小節之描述)。產生器主要是將特定問題域的需求規格(Specification)轉換成程式片段、子程式(Subroutine) 或軟體系統，如圖2- 1所示；例如過去就一直最常使用的C preprocessor，就是一種最簡單的產生器，C preprocessor控制#include、#define、#if與#ifdef等宣告，這種利用代換與插入檔案產生的程式，也是最簡單的一種產生動作。

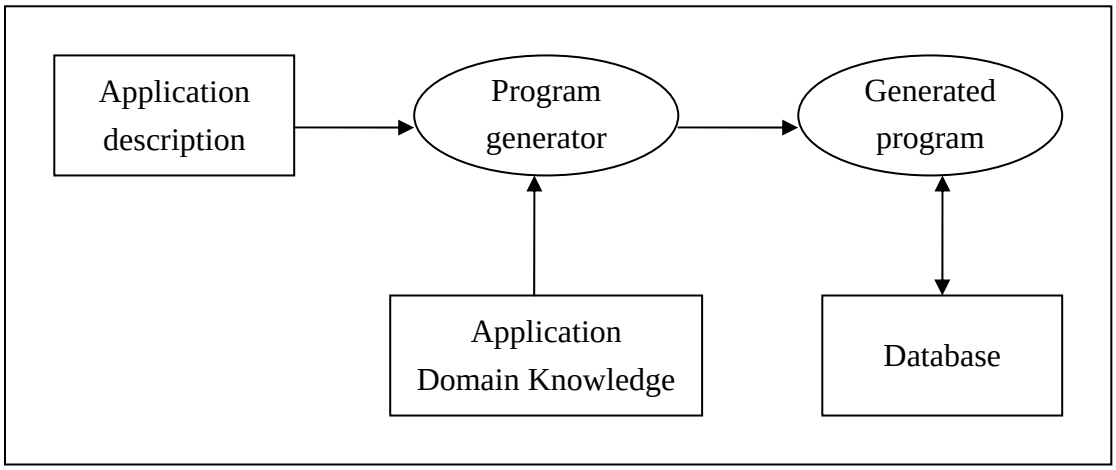

圖 2- 1 程式產生器(取自[7])

依據Code Generator的性質可以分為兩類，以下就針對Code Generator的性質做簡單的分類與說明：

I Passive Style: One-time code generation

稱為被動型態的產生器，該產生器只會產生一次程式碼，產生過後就不再更動。

II Active Style: Integrated code generation

而主動型的 Code Generator 則可分類成六個種類，分別於下各小節中介紹：

2.1.1 型 1: Code munger

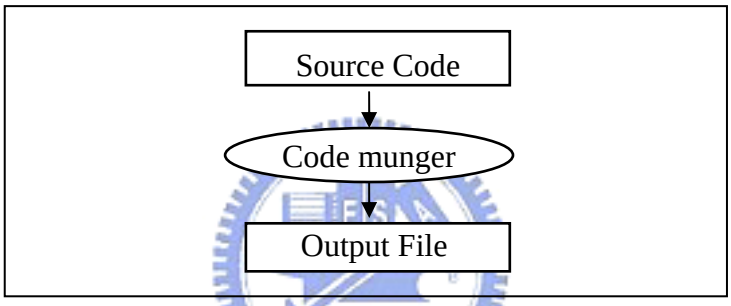

圖 2- 2 Code munger 產生器架構圖(取自[7])

Code Munger 是將給定的程式碼轉依據其特性換成另一種形式的輸出，其流程如圖 2- 2 所示，通常是依正規表示語法使用簡單的來源分析後，轉換輸出成另外一個輸出的檔案，可以用在許多地方，例如用它來產生文件資料或是函數。以 C 語言中#define 為例，我們定義 PI 如：#define PI 3.14159

於是在程式中當遇到變數為 PI 時，便會將 PI 替換成 3.1419 這個數值，使用此架構的優點為：

1. 從單一個檔案中讀入資料，在設計上較為簡易。
2. 從單一個路徑下中讀入資料，不容易錯亂且管理方便。
3. 從單一個標準輸入讀入資料，較節省讀取時間。

缺點則是功能太過簡易，無法推廣使用在大型的專案系統開發中。

2.1.2 型 2: inline-code generator

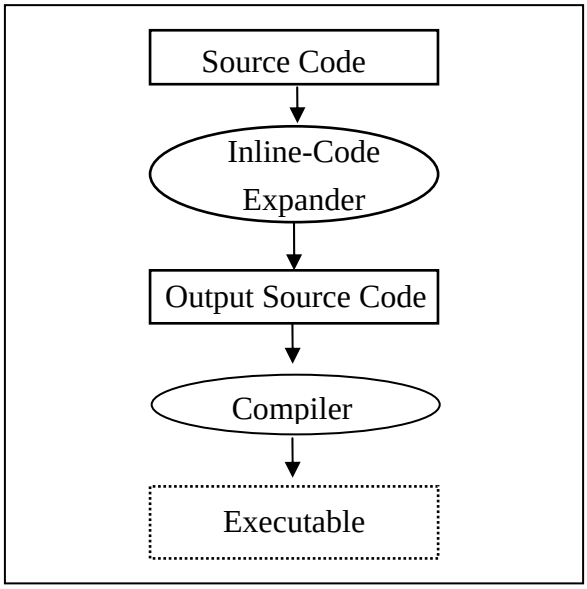

圖 2- 3 inline-code 產生器架構圖(取自[7])

Inline-code 擴展是一種簡化程式碼的方式，是以種特殊語法的方式將原本的程式碼展開，而該語法則是透過產生器依據不同的需求提供轉譯，轉譯出的程式碼再經過編譯器編譯成為可執行檔。該產生器運作的流程則如圖2- 3所示，茲舉一範例如下表1所示：

表 1 inline-code 展開前範例

```
void main( int argc, char *argv[] ){
<sql-select: SELECT first, last FROM names>
return;}
```

經過inline-code generator讀入展開後，最後的產出結果如下表2所示：

表 2 inline-code 展開後範例

```
void main( int argc, char *argv[] ){
struct {
char *first;
char *last;
} *sql_output_1;
   {
      db_connection *db = get_connection();
      sql_statement *sth = db->prepare( "SELECT first, last FROM names" );
      sth->execute();
      sql_output_1 = malloc( sizeof( *sql_output_1 ) * sth->count() );
      for( long index = 0; index < sth->count(); index++ ){
         // ... marshal data}
   }
```

2.1.3 型 3: mixed-code generator

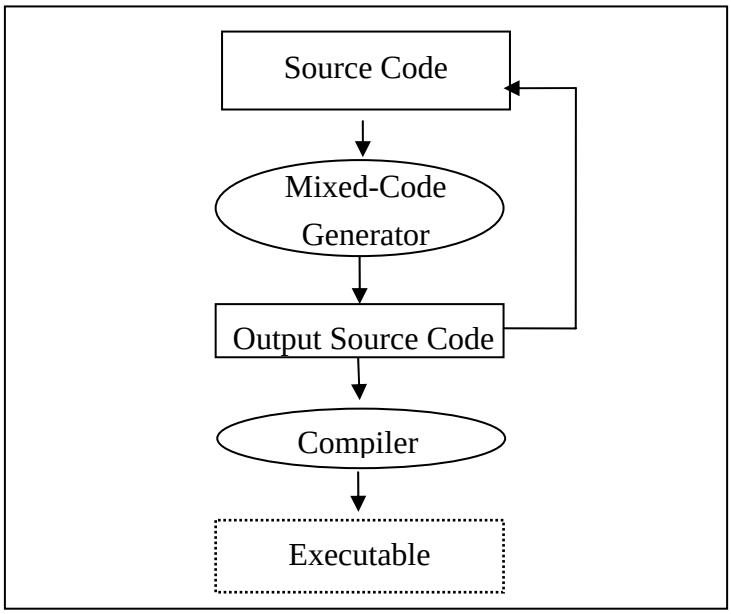

圖 2- 4 mixed-code 產生器架構圖(取自[7])

混合式產生器從單一個檔案讀入來源資料，依據不同問題的需求，做了部份的修改後再回存到該檔案，和 in-line code 最主要的不同則是在於 I/O 的部份，mixed-code 所產生的檔案和原始的檔案是同一個檔案，其流程如圖 2- 4 所示，以 2.1.2 節中的例子變化如下：

表 3 mixed-code 轉換前範例

```
void main( int argc, char *argv[] )
{
   // sql-select: SELECT first, last FROM names
   // sql-select-end
   return;
}
```

其中<sql-select: …> 程式碼的部份將會轉換成為如表 4 所示：

表 4 mixed-code 轉換後範例

```
void main( int argc, char *argv[] ){
   // sql-select: SELECT first, last FROM names
   struct {
      char *first;
       char *last;
   } *sql_output_1;
   {
      db_connection *db = get_connection();
      sql_statement *sth = db->prepare( "SELECT first, last FROM names" );
      sth->execute();
      sql_output_1 = malloc( sizeof( *sql_output_1 ) * sth->count() );
      for( long index = 0; index < sth->count(); index++ ){
         // ... marshal data
      }
   }
       // sql-select-end
       return;
}
```

2.1.4 型 4: partial-class generator

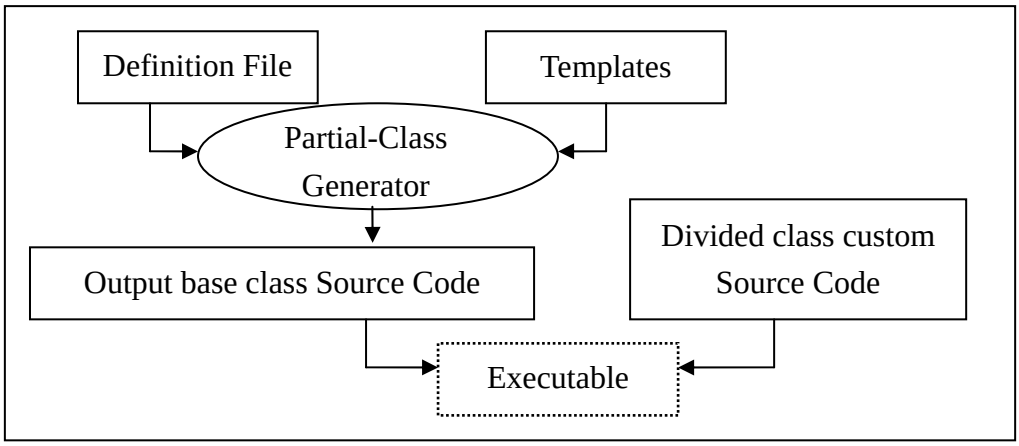

圖 2- 5 partial class 產生器架構圖(取自[7])

Partial-class generator 首先會從一個定義檔中讀入所需要的資訊，緊接著會使用一個 templates 格式建立目的類別檔，而此類別檔案產生後，將由開發者自行編譯成可用的目的檔，圖 2- 5 為產生的流程，舉一範例如下：我們在某一個文字檔中定義一類別為 Person，且類別的屬性有 first、middle、與 last 三個，經過 Partial-Class Generator 的處理後產生如下的結果：

表 5 partial class 轉換後範例

```
public class PersonBase {
    protected String _first;
    protected String _middle;
    protected String _last;
    public PersonBase()
    {
    _first = new String();
    _middle = new String();
    _last = new String();
    }
    public String getFirst() { return _first; }
    public String getMiddle() { return _middle; }
    public String getLast() { return _last; }
}
```

2.1.5 型 5: tier generator

此類的產生器通常可用來建立完整的系統，有別於前面幾節所討論的產生器都是以產生個別檔案或某些功能的函數與片段程式碼，而最常見的方式是使用模式導向分析方式分析並產生整個系統。由圖 2-6 可知 tier generator 的輸入和輸出是和 partial-class generator 相同，tier generator 讀入定義檔並且使用樣板的方式產生檔案，可以分別產生使用者介面、商業邏輯與資料連結三個部份達成多層次的 Web 應用系統。

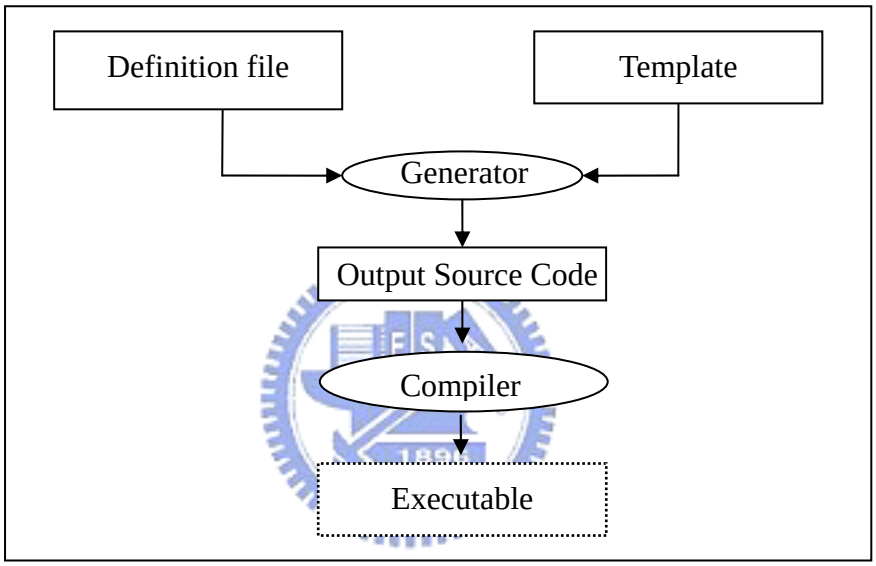

圖 2- 6 tier 產生器架構圖(取自[7])

2.1.6 型 6: Full-Domain Language

Full-Domain Language 可以允許工程師輕易用該語言表達所達不同專業領域裡的概念，該語言包括了對變數的宣告、邏輯判斷、分支跳躍等不同語法的支援，設計者只需要對目前開發的系統，了解專業領域上的知識即可產生系統，例如：行政事務系統、人事差勤系統...。但是這種高階語言在先天上就會有一些限制，例如在數學分析計算方面的支援就會不如傳統上的 C 或是 C++，要用到大量的矩陣運算或是最佳化，效能的表現就不會太好。

2.2 MVC 設計樣式

本節主要介紹 MVC 設計樣式的概念，但在介紹 MVC 之前，會介紹兩個相關的軟體開發架構，常用於系統設計時的塑模方法，分述於 2.2.1 節的 Model Driven Architecture(MDA)以及 2.2.2 節的 Presentation-Abstraction-Control(PAC)是用於使用者介面分析的表達工具。

2.2.1 Model Driven Architecture

模式驅動結構（Model Driven Architecture, MDA）是由 OMG 定義的一種軟體開發架構，主要是用來解決製作與維護高階文件時，文件品質低落的問題。

整個 MDA 發展的生命週期則是如圖 2- 7 所示，MDA 在發展過程中步驟之產出，強調其產出是由電腦可理解的正式模式（Formal Model）表達。

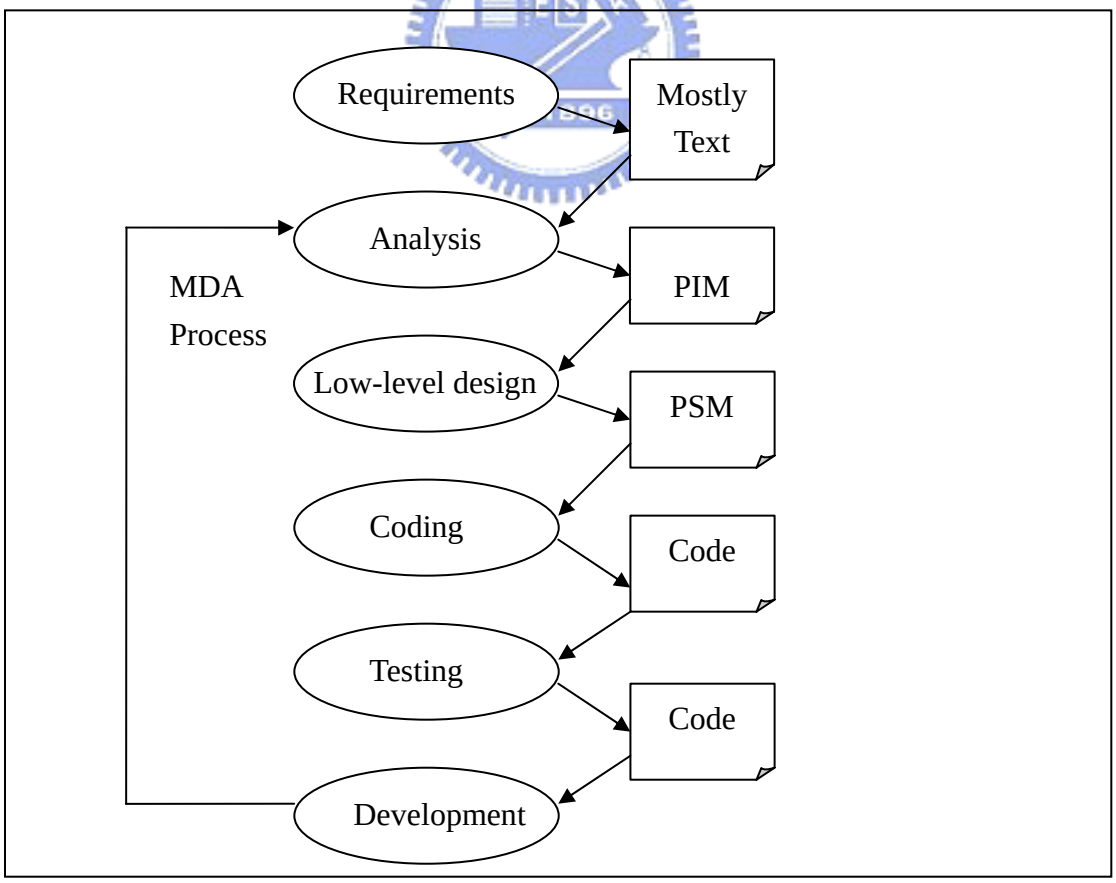

圖 2- 7 MDA 軟體開發之生命週期(取自[22])

在 MDA 有三個核心模式[9]：平台獨立模式、特定平台模式與程式模式，以下就分別介紹之。

一、平台獨立模式(Platform Independent Model, PIM)

PIM 是一種高階抽象的模式，該模式與所採用何種開發技術的獨立，PIM 是分析與設計後的結果，主要是根據需求分析的結果，得到如何支援企業運作的觀點來描述一個軟體系統，但是並不涉及描述系統開發與運作之平台。

二、特定平台模式(Platform Specific Model, PSM)

PSM 則是一種特定平台的模式，該模式是相依於軟體開發技術，PSM 的描述方式主要是以所使用的開發工具來描述一個軟體系統，在描述時便會大量使用該特定平台中的技術辭彙，例如：對關聯式資料庫 PSM 而言，辭彙包括 “table”、“column”、 “foreign key”。一個 PIM 的描述則會包含一個以上的 PSM，因為開發的系統可能包含數種技術，PSM 內的溝通則利用溝通橋樑來達成訊息交換。

三、程式模式(Code Model, CM)

每個 PSM 最終會被轉成使用開發工具的程式碼，由於 PSM 是相依於其開發技術，因此 PSM 轉成程式碼之過程非常直接。如果開發的系統中有多個 PSM 時，此時將會轉出多種的程式碼，而這些程式碼間的互動是藉由溝通橋樑來達成。

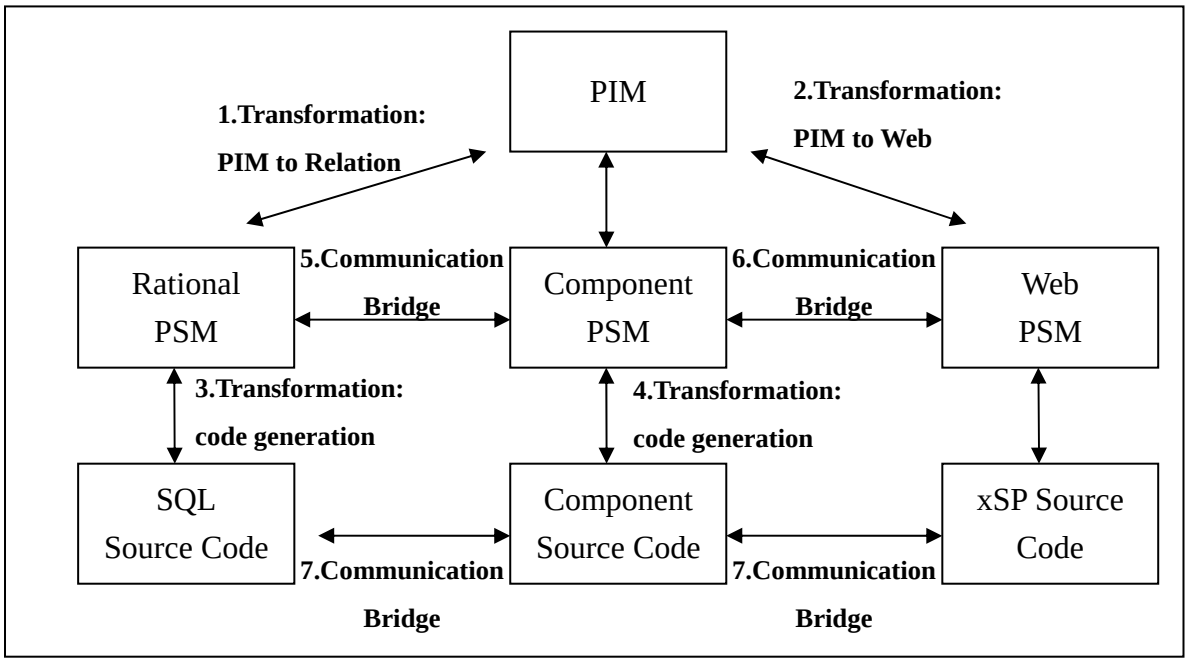

圖 2- 8 MDA 架構圖(取自[10])

MDA 轉換的架構圖則是如圖 2- 8 所示，首先由使用者需求分析中製作出 PIM，接著由此描述出的 PIM 轉成一個或多個的 PSM，例如圖 2- 8 的 Web PSM、Rational PSM 與 Component PSM，系統利用到此三者技術進行開發，則會產生三個 PSM 的描述，將 PIM 轉換至 Rational PSM 與程式樣版的研究(圖 2-8 中步驟 1)可見[20]，類別圖轉換至物件關聯模式則可見[21]。最後則是將這些 PSM 分別轉成對應的程式碼，以此例來說，三個 PSM 的程式碼分別會轉出 JSP 程式碼、EJB Component 程式碼與 SQL 程式碼，而這些不同的程式碼則是可以利用溝通橋樑來達成相互間的溝通。

2.2.2 Presentation-Abstraction-Control

另外一個常用於使用者介面分析的表達工具則是利用 PAC 架構，良好的介面架構圖可以使得介面間的連接與控制關係清楚地表現，而 PAC 是個常見的介面架構表達的工具，PAC 架構圖則如圖 2- 9　PAC 架構圖所示。

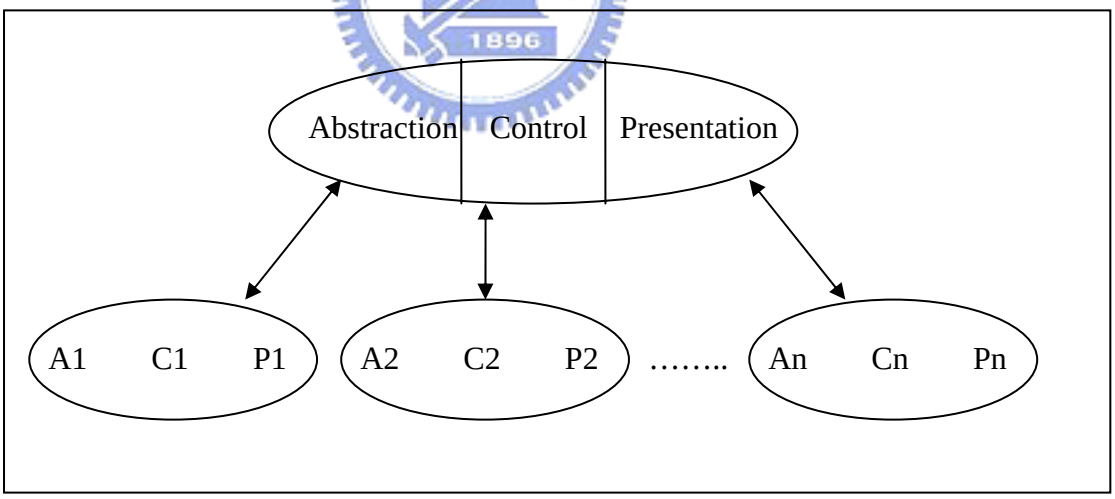

圖 2- 9　PAC 架構圖(取自[22])

PAC 將使用者介面的設計細分成許多個子介面，其中的每個子介面都可視為一個物件，每一個物件是展示(Presentation)、摘要(Abstraction)與控制(Control)三個部份所組成。其中展示部份是定義該物件的外觀，並處理訊息的輸入及輸出；摘要則是定義物件的功能及概念；控制部份則是展示與摘要間溝通的橋樑，

也是其它的物件相互連繫的管道。利用 PAC 進行使用者介面塑模[18]，這種作法可讓模式只單純的作資料處理的動作，因此可利用 PAC 模式提升在設計與維護的效率。

2.2.3 Model-View-Controller 設計模式

MVC是一種相當著名的而且廣泛被採用的設計樣式，最早可追溯到Smalltalk[1]語言的Smalltalk-80 版本，在Smalltalk-80 中，使用者介面(UI)其背後主要的概念就是MVC。

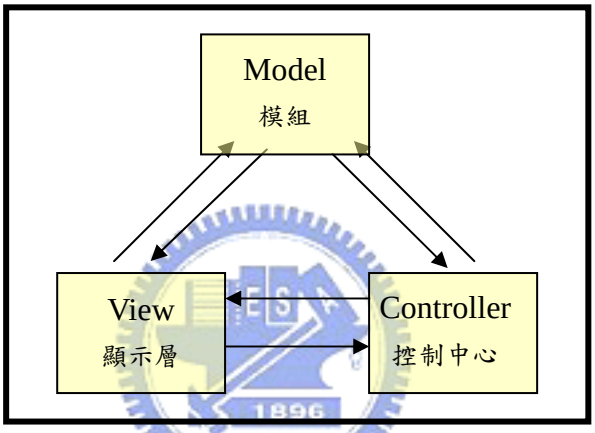

圖 2- 10 MVC 運作模式

"Model"所代表的是應用程式的商業邏輯，是用來執行資料的處理或轉換、回應請求資料的要求，以及回應變更資料的要求，Model 不用去管 View 與 Controller 的設計方式，它們可以透過特定的介面相互溝通。

"View" 是應用程式的表現畫面，也就是把資料做視覺化的呈現在使用者面前，View 透過特定的介面與 Model 溝通，設計時也不會相互影響。

"Controller" 是應用程式的處理程序控制，提供改變 Model 中資料的機制，透過 Controller 抓住由使用者所發出的鍵盤或滑鼠事件，以便告知 Model 資料已改變的事實，亦是透過特定的介面與 Model 溝通。可以透過這種設計模型把應用邏輯、處理過程和顯示邏輯分成不同的單元分別實作，使得程式在維護上開發

[1] Smalltalk，一種物件導向程式語言，由Alan Kay所發明。

時有相當大的助益。

圖 2-11 為一個簡單 MVC 範例，假設欲設計一個時鐘程式，依據顯示方式的不同可區分為數位式與指針式時鐘兩種，兩者皆顯示相同的時間，其中計時的部份就是 MVC 中的 Model，該 Model 負責保存時間，也允許將時鐘設定成新的時間；而數位式與指針式兩種呈現的方式不同，數位式的時鐘是利用數字的改變來顯示時間，而指針式的時鐘則是以指針的移動來顯示時間，所表示的是相同的時間；使用者可以利用滑鼠或是鍵盤操作時鐘，藉由事件的觸發，可以向 Model 送出重新設定時間的請求，而利用滑鼠或是鍵盤發出改變 Model 的狀態，這部份就是 MVC 中的 Controller。

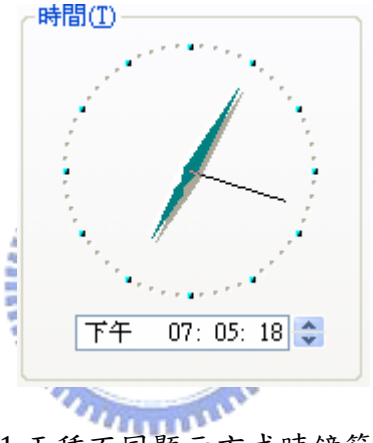

圖 2- 11 兩種不同顯示方式時鐘範例

目前以MVC設計模式開發系統的例子，則是以Struts框架為首；Struts是Jakarta下的一個子計劃[11]，其目的是以一個開放式的架構來建構Java的網路應用程式，Struts就是以MVC設計模式而發展出來的架構。在Struts中的Model是利用java beans 實作，JSP[2]對應到View，而Servlets[3] 對應到Controller與URI互相對映，使得資料處理、系統功能以及外觀展現的處理能夠區分出來，整個的運作模式如圖 2- 12 所示：

[2] JSP，Java Server Page，http://java.sun.com/products/jsp/
[3] Servlet，Java Servlet Technology，http://java.sun.com/products/servlet/

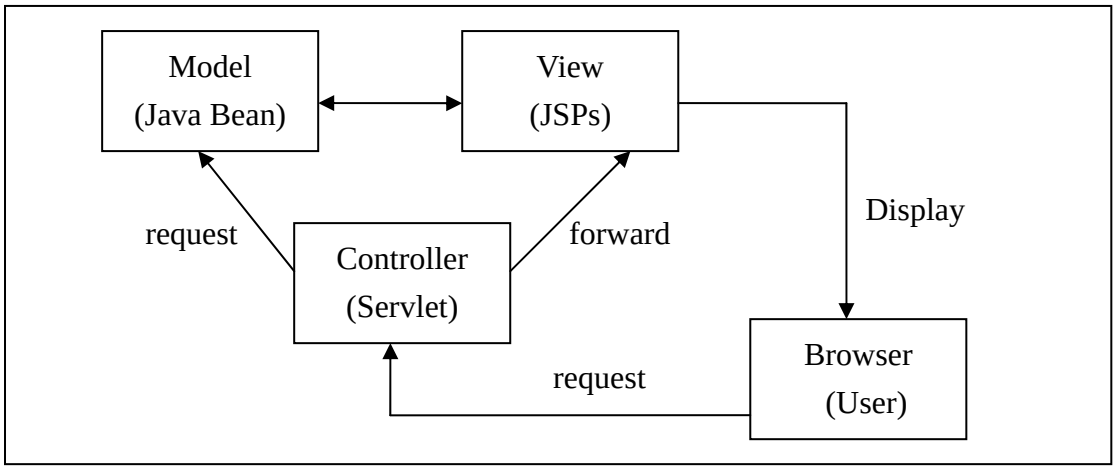

圖 2- 12 MVC 運作模式 in java

1. Model 部分：

採用 JavaBean 和 EJB 元件，設計和實作系統的商業邏輯。根據不同的請求從 Action 衍生具體 Action 處理物件。完成"做什麼"的任務來 呼叫由 Bean 構成的商業組件。

2. Controller 部分：

提供應用程式的處理程序控制(使用 Servlet)，負責處理從用戶端傳送的請求，分析請求後決定適切的處理者，交給該執行者處理，最後交給適當的 View 元件展現；控制器的作用是從用戶端接受請求，並且選擇執行對應的商業邏輯，然後把結果送回到用戶端。Struts 提供了核心控制部分的實作，只需要配置 ActionMapping 物件。

3. View 部分：

Struts 應用中的 View 部分是透過 JSP 技術實作的。為了使用 Model 中的 ActionForm 物件，必須用 Struts 提供的自定義標籤建立 HTML Form，利用 Struts 提供的自定義標籤庫編寫使用者介面把應用邏輯和顯示邏輯分離，Struts 框架透過這些自定義標籤建立了 View 和 Model 之間的聯繫。

基於 MVC Framework 的設計、觀念與技巧，java opensource 也提出各式各樣的解決方案，例如：apache 就實作了 jakarta turbine、jakarta struts、jakarta

tapestry、avalon…等；opensymphony 的 webwork、webwork2；其他的還有 Maverick、JPublish、JApple… 等等；也有包含更多的 J2EE 架構的實作 framework，例如：expresso、SOFIA、Spring 等。所以 MVC 設計模式在許多重要軟體開發的場合中，都會出現它的足跡，也可以想見它的重要性。

2.3 潛在問題與改進

本節所要討論的是產生器的潛在問題與改進的構想。首先說明現有產生器的潛在問題，再探討應用系統產生器的組成元件，最後並提出幾個改進的構想，與應用系統建構的方法論。

本研究以市面上的 CASE Tool - Rational Rose 為例，簡介 Rose 所提供的功能以及其優缺點做出如下整理：

1. 簡介：

Rational Rose 是一個可以輔助進行物件導向軟體系統分與設計的視覺化工具，在尚未撰寫任何程式之前，可以利用該軟體為系統塑模工具，利用 UML 描述系統規格，以確保系統的周嚴性與完整性。

2. 所支援程式碼：

Ration Rose 提供了 C++、XML DTD、Visual Basic、C#、CORBA、Web Application 等程式語言的程式碼產生，設計人員可以利用 UML 語言塑模出系統的設計規格書與系統文件，並利用 Rose 所提供的還原工程，產生程式碼。

3. 多數只支援單機版本：

Ration Rose 只提供了單機版產生程式碼，無法讓產生的程式碼與多人共享，設計階段的資訊無法在程式設計人員間流通，資訊的交流必須透過網路分享或其他的方式進行。

4. 在 Web Application 上的不足：

其中 Web Application 所產生的程式碼只是針對網頁中的元件產生對應的程式碼，這些程式碼只包括 User Interface 的外觀，對於建立整個應用系統的完整

性來說，尚欠許多功能。

從 2.2 節討論的例子中，我們可以發現不同類型的產生器與的各個產生器的特性，也可以發現不同的產生器在產生程式碼上不同的地方，將各類型產生器功能比較、優缺點整理如表 6 各類型產生器比較表所示：

表 6 各類型產生器比較表

特性	Code munger	Inline-code	Mixed-code	Partial-class	Tier/layer
產生架構	單機使用	單機使用	單機使用	單機使用	單機使用
資料來源	單一檔案	單一檔案	單一檔案	多個檔案	多個檔案
產生結果	單一輸出	不同檔案	單一輸出	單一輸出	單一輸出
環境設定	手動設定	手動設定	手動設定	手動設定	手動設定
Template	無	無	無	有	有
產生型態	靜態產生	動態產生	靜態產生	靜態產生	動態產生
資料儲存	無	無	無	無	無
優點	1.容易使用 2.單一個標準輸入讀入	1.簡化設計時的複雜度 2.使用不同語言，易於實作	1.所產生輸出的檔案是不同檔 2.易於實作	1.利用已定義與樣版的檔案產生輸出	1.有定義與樣版的檔案 2.動態產生輸出的檔案
缺點	設計過的規則不易於重複利用	沒有一套良好管理 inline-code 的方法	沒有一套良好管理已產生程式碼的方法	需要良好的分析出物件類別	架構討論較缺乏完善方法論
範例	Define in C	Macro in C	C	J2EE	-

本論文依照產生器的不同特性，分別說明在設計產生器時，可能會遭遇到的問題，並提供解決方式來改進產生器：

問題 1：只支援單機版本的產生器佔大多數

解決方式：產生器系統架構改用以 Web-Based 的架構

過去程式產生器的系統架構大多都為單機式的版本，一次只能作業在同一台機器上，對於多人使用的系統開發需求並無法滿足，實際的運作方式只好將工作切割，把不同功能或子系統分別交付，最後再匯集個別產生的程式後組合。本論文改進的方法，是透過 Web-Based 的架構，利用網路分享的方式，讓具有共同任務的小組同時作業，如圖 2- 13 產生器系統架構所示：

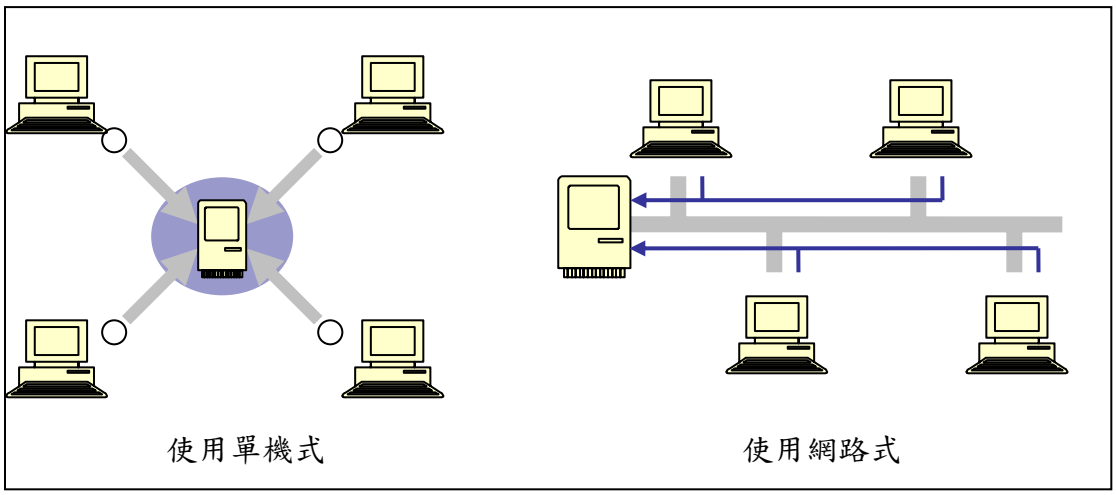

圖 2- 13 產生器系統架構

問題 2：樣版資料來源只來自於單一檔案

解決方式：以資料庫作為資料來源

讀取的範本檔或是樣版資料的來源可以分為：單一檔案與多個檔案；前者通常讀入資料較簡短，有時讀取的來源就在相同的檔案中，如 inline code 類型；另外一種改進方式則是從多個來源檔讀取，在系統的設計上較有彈性，對於模組或功能的描述也可以放在不同的檔案中。本論文所提出的方法是以資料庫做為資料來源的儲存體，將定義好的樣版資料儲存於資料庫中，更有助管理，產生器資料來源方式表示如圖 2-14。

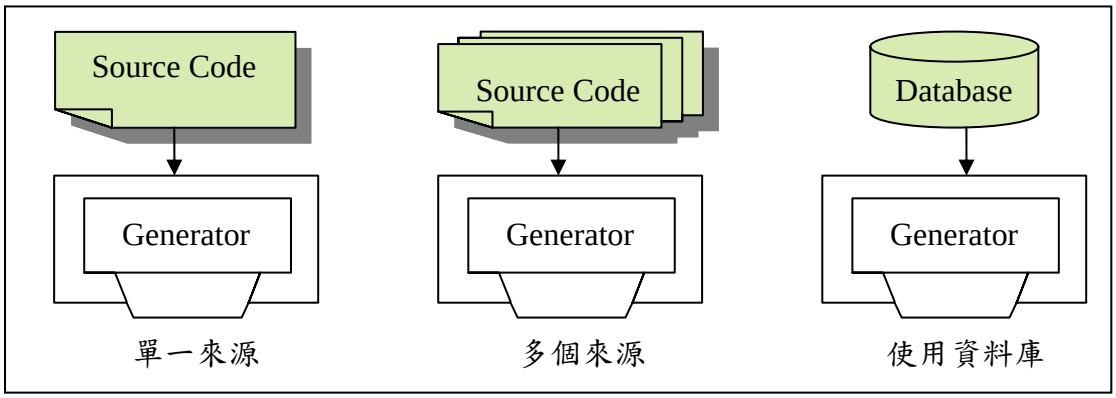

圖 2- 14 產生器資料來源

問題 3： 畫面設計與程式無法重覆利用

解決方式：利用 Template 定義輔助設計

利用樣版所定訂出的資料可以節省撰寫與開發的時間，進而可以重複利用

已經開發過的程式碼或是使用者介面等部份，雖然不同程式語言有不同的支援方式及語法，透過 MVC 設計樣式，將常用的函數程式碼與使用者介面定義成 Template，將 Model 與 View 的設計分離，可以快速的進行設計工作。圖 2- 15 畫面樣版的例子是顯示一個使用者介面樣版之範例系統畫面；畫面上方的選項是做為選擇子系統的分類，畫面中所顯示的五個選項：首頁、資源管理、資源瀏覽、用戶管理、系統管理等；當指標移動到該選項時，則會跳出一下拉式選單，讓使用者選擇各個子功能，畫面中用戶管理下就提供了個人信息、修改個人信息、權限申請、用戶組管理、設定用戶組、修改用戶權限…等九個子功能。

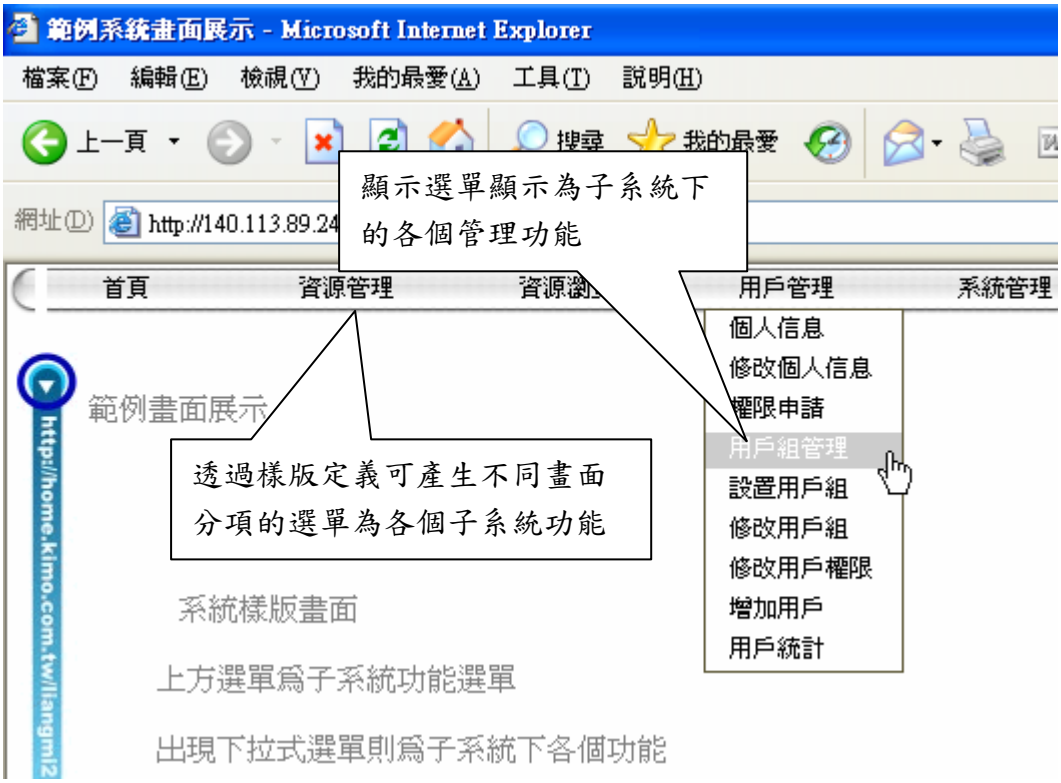

圖 2- 15 畫面樣版範例

第三章、應用系統程式產生器之設計

經由第一章的研究動機說明，可以了解本論文要做的方向，於是收集、研究了許多相關的文獻，如第二章所介紹，在探討的過程中，了解到何謂應用系統產生器，了解產生器的設計方式，也了解到使用 MVC 設計模式的好處，接著本論文提出改良後應用系統產生器之設計，在此章中，3.1 節先對系統設計的原則做說明，3.2 節則是描述系統操作的整個流程，接著再對本論文提出的應用系統產生器做詳細的設計描述，3.3.1 節則是針對系統架構說明，3.3.2 則是說明所提供系統模組。

3.1 系統設計原則

實作支援所有的程式語言並不是本篇論文的主要目的，本論文最主要是提出彈性的設計架構，藉由產生良好的程式碼快速完成資訊系統建置。因此建議可選擇網路應用程式中最常用的三種語言：JSP、PHP、ASP 之一來進行設計，系統設計原則需符合以下功能系統特性：

A. 以開放式網路環境為基礎架構，可使得多人同時進行設計工程

B. 採用 MVC 概念為架構發展 CASE Tool

C. 以 XML[13]為資料儲存方式，可彈性設計語言格式

D. 以設計樣板[2]的方式，藉以應用在不同領域之快速轉換

E. 將分析階段之 meta-data 直接按照正式規格輸出發展手冊

F. 定義綱要資料可以使得多語言的程式設計環境，快速達成

目前大多數的程式語言所提供的圖形使用者介面都將 View 與 Controller 合併，若是將 View 與 Controller 合併則稱之為 Delegate 元件[10][11]，由 Delegate 負責 View 與 Controller 的責任，即接收使用者發出的請求，並與 Model 溝通，在 Model 狀態改變時，Delegate 就會向 Model 取得最新的資料來更新自己的狀

態。本論文的設計架構中，MVC 的部份將以 Model-Delegate 設計架構來實現，如圖 3-1 所示，在 3-2 節中說明產稱器運作操作的流程，如何利用 MVC 設計模式支援應用程式產生器於 3-3 節中描述系統架構與系統模組的設計。

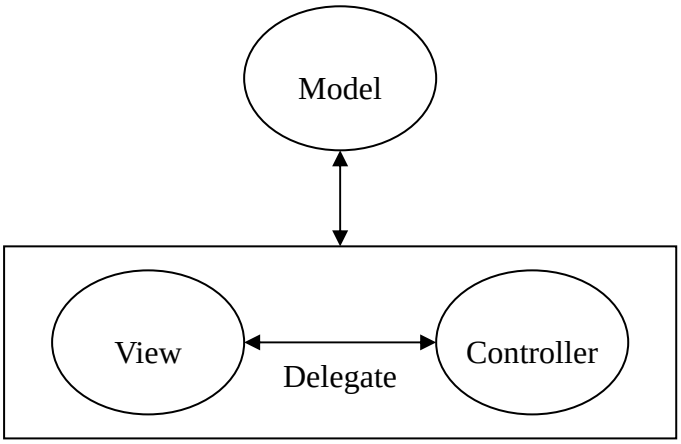

圖 3- 1 Model-Delegate 的架構圖(取自[26])

3.2 系統流程

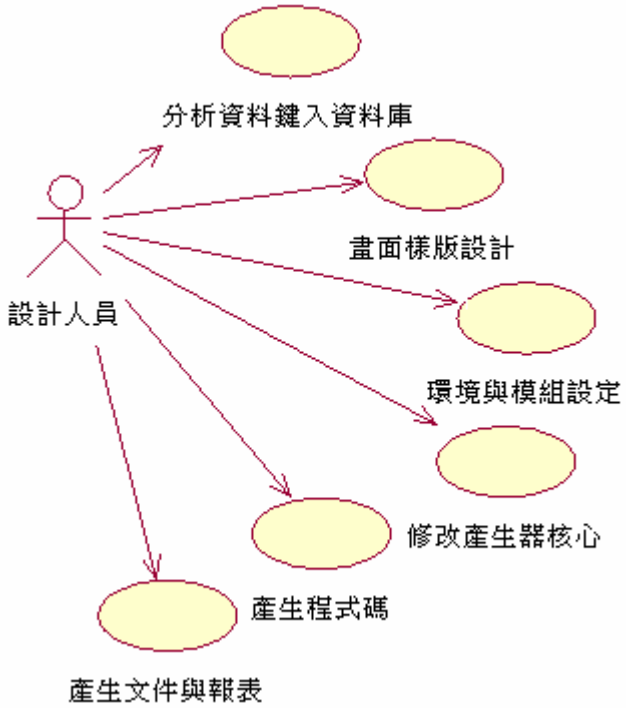

圖 3- 2 系統使用個案圖

圖 3-2 為利用 UML 繪製之系統使用個案圖，說明了設計人員在建構系統時，可以先從建立檔案物件開始，也就是以關聯式資料表的 Table。頁面之間流

程的轉變與控制權的轉移則依照各個子系統功能的需求來設定執行的先後順序。樣版的格式可以由檔案或資料庫中讀入，搭配使用者自行定義的標記，可以將部份 Script language 輸出至外部版面控制頁面；資料庫連線控制的程式碼則輸出到中心邏輯控制檔案中；系統中 meta-data 則是包括負責儲存所要產生的應用系統相關的資料表與功能函式，該資料表可經由使用者介面將應用系統之需求塑模建立至產生器中，功能函式則是將系統常使用到的程式模組化，將其儲存於資料庫中，做為輸出時的系統 include file 或是 library。

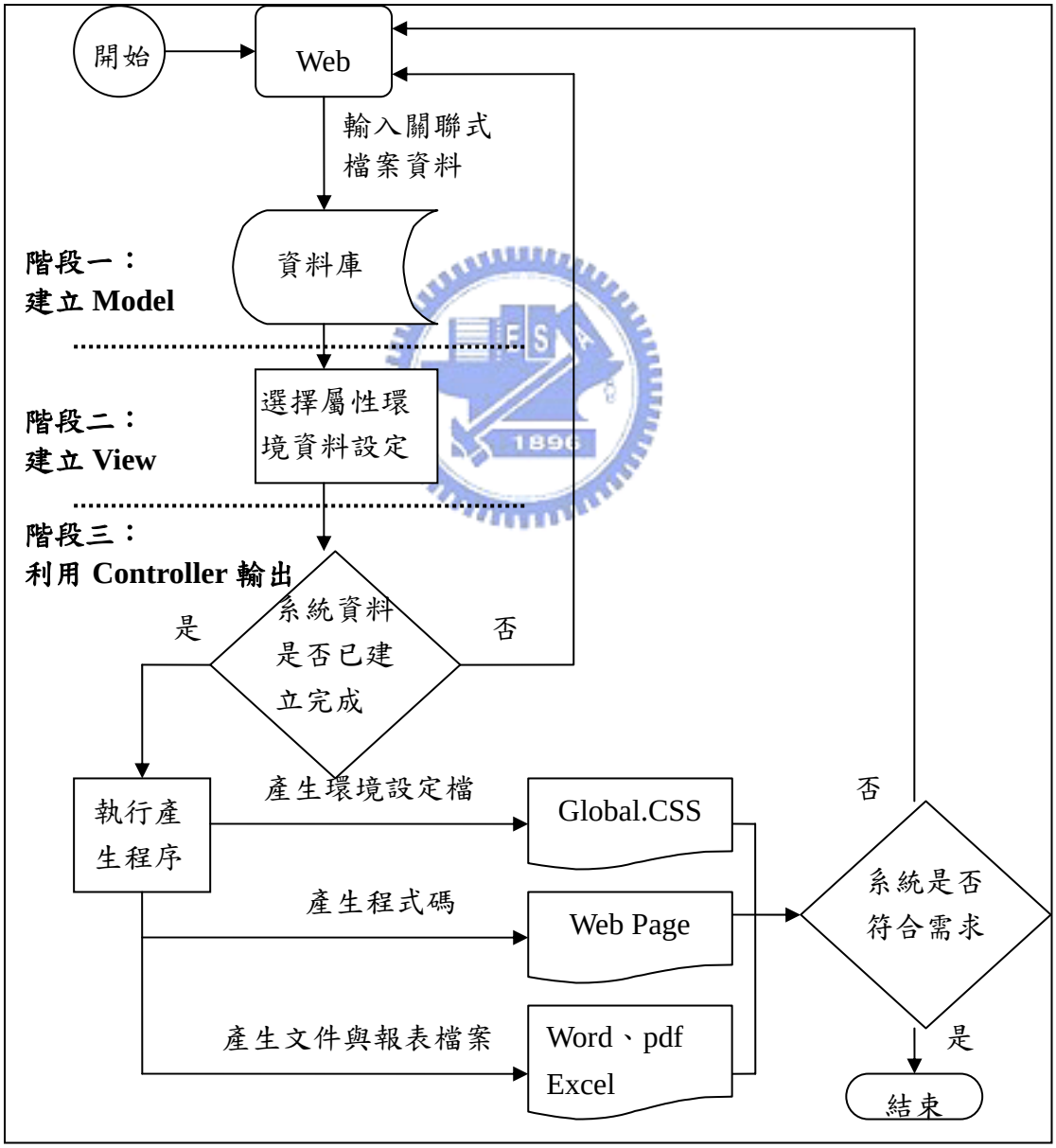

圖 3- 3 以 MVC 設計模式應用系統產生流程圖

整個系統產生可以分成三個階段，分別是所需要的應用系統資訊輸入階段，包括建立 Model 與 View Page，以及最終的應用系統產生階段，產生流程如圖 3- 3 所示，運用 Code Generator 產生 Application 可產生完整程式碼，資料庫是資料的來源；輸入與產生階段的步驟則分別如下所列：

階段一：建立 Model

步驟1. 關聯式檔案資料輸入

步驟2. 函數與副程式碼建立

階段二：建立 View Page

步驟1. 使用者頁面建立

步驟2. 圖檔與其他資料

步驟3. 環境資料設定

階段三：利用 Controller 產生程式

步驟1. 產生系統目錄

步驟2. 產生環境設定檔

步驟3. 產生函數檔案

步驟4. 產生使用者頁面

步驟5. 產生處理頁面(Process Engine or EJB)

步驟6. 產生報表(System Report)

以圖 3-4 來說明使用者與系統的互動關係，使用者透過 Delegate 將資料輸入到資料庫中，是建立 Domain Entity，儲存在資料庫中的 meta-data 是產生器 model 的資料來源，等待資料建立完成後，就可以進行資料與程式的產生，使用者可以直接瀏覽或下載產生的結果，進一步的修改核心 model，以符合功能需求。

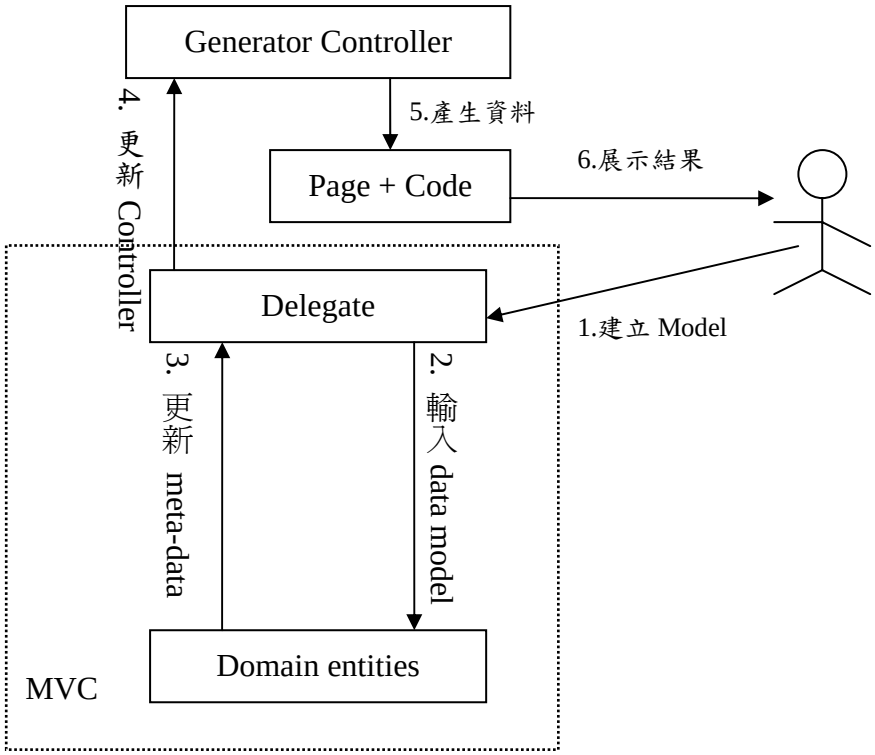

圖 3- 4 使用者與系統的互動關係圖

3.3 系統設計

本小節是針對本論文提出的應用系統產生器設計內容做詳細的設計描述，3.3.1 節則是針對應用系統產生器的架構與元件做說明，3.3.2 則是說明系統的模組，並解釋各個模組如何達成 MVC 設計樣式之支援。

3.3.1 系統架構

本應用程式產生器系統的設計主要是以 Web-based 環境為基礎，使用者可以透過 HTTP 協定連線至產生器的引擎中心，透過此一引擎進行整個應用系統的產生；而產生的程式碼，則是儲存在主機端的檔案庫。

當使用者端(即設計人員)透過網路連線後，將想要產生的應用系統相關資料鍵入後，產生器的引擎便會根據已經設計好的樣版資料，包括 Presentation Tier 的使用者介面端程式、Model Tier 的商業邏輯與 Control Tier 的程式流程控制等三個部分，產生輸出至檔案系統中儲存，整體運作架構則如圖 3- 5 所示：

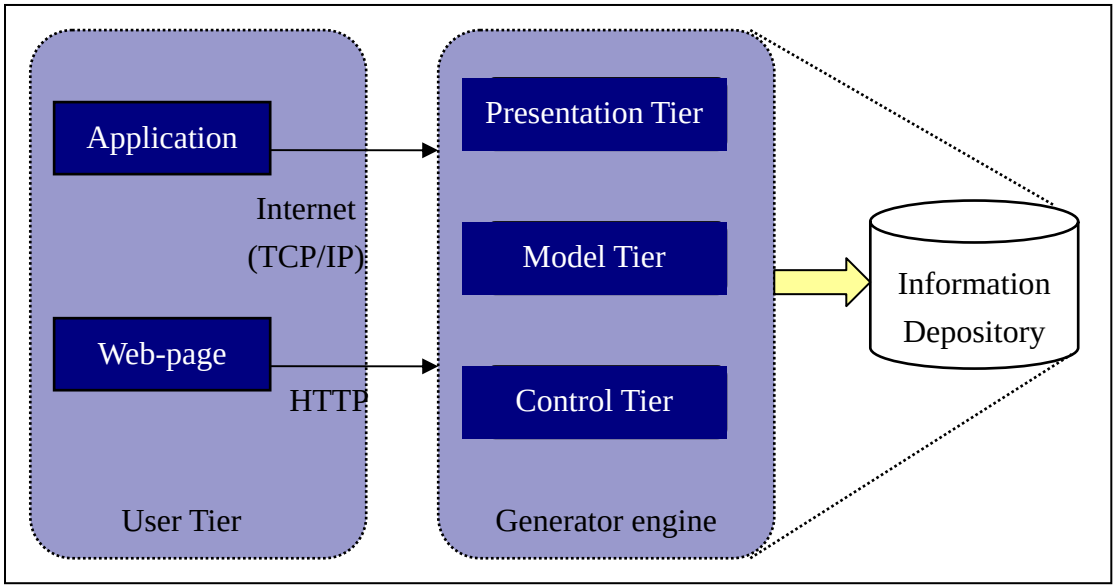

圖 3- 5 系統運作架構

系統組成元件則包括使用者介面頁面(View Page)、功能函式庫(Deployment and function)、檔案物件庫(Other Files)、meta-system(Database Schema)、應用系統功能(xSP)等部份。其中使用者介面部份為系統與使用者互動的操作介面；功能函式庫提供應用系統運算或處理的功能函式，將常用的功能函式儲存於資料庫中，例如：西元日期與民國日期轉換運算、匯率轉換等常用運算功能；而 meta-system 為記錄應用系統架構與運作流程的資料庫，meta-system 可視為整個應用系統運作的驅動中心，利用此一 Table driven 的方式將系統環境設定、檔案物件、系統報表等資訊以資料表格(Table)的方式儲存，系統於建置時，就可以依據 Table 中的資料產生該系統的環境設定檔，如 Config.xml 或 Config.CSS 等檔案，系統組元件則如圖 3-6 所示。

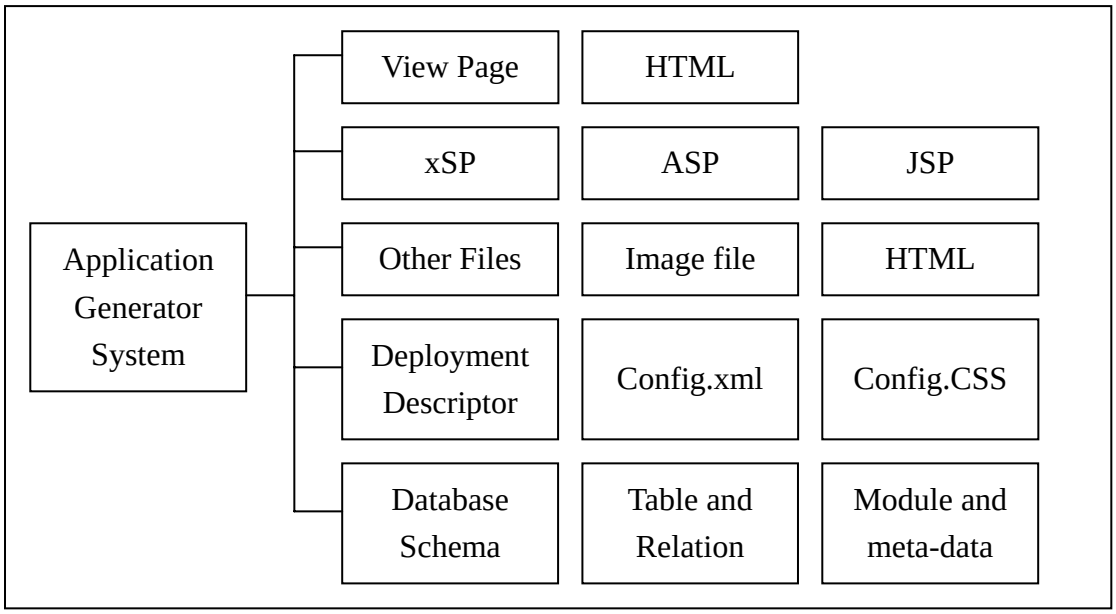

圖 3- 6 應用程式產生器系統組成元件

3.3.2 系統模組

本小節依據本論文中設計之系統所提供的功能，將應用系統產生器設計成七大模組，系統模組架構圖則如圖 3- 7 所示，其中檔案物件模組與功能函數模組是系統的核心，負責 MVC 中的 Model；而樣版定義模組與環境設定模組，是負責 MVC 中的 View；結果輸出模組與流程物件模組是負則 MVC 中的 Controller。以下就針對各個模組所提供的功能進一步描述：

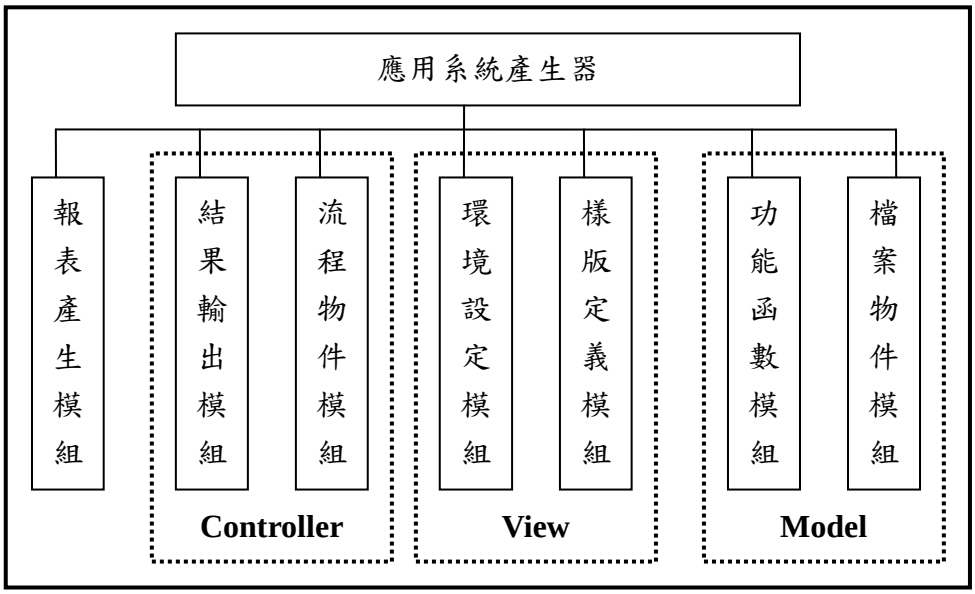

圖 3- 7 系統模組架構圖

1. 檔案物件模組

利用檔案物件模組，將關聯表與資料字典定義，並將這些資料建立至資料庫系統中。除了設定資料表格鍵值、每個欄位的型態與長度，也從關聯表建立檔案物件間的關聯，表 7 則是檔案物件的資料庫表格。

表 7 「檔案物件」資料庫表格

資料元素名稱	欄位名稱	欄位型態	欄位長度	主鍵	外鍵
編號	ID	Numeric	9	*	
中文名稱	Chi_Name	Varchar	50		
英文名稱	Eng_Name	Varchar	100		
功能說明	Fun_Desc	Varchar	255		

2. 功能函數模組

功能函數模組則是將系統共用的函數由專人設計完成後，建立至資料庫系統中，透過統一集中的方式管理，可以使得設計的工作更容易進行，表 8 是 include 檔資料庫表格，預覽程式碼會顯示類似表 9：抓取現在函數範例。

表 8 include 檔資料庫表格

資料元素名稱	欄位名稱	欄位型態	欄位長度	主鍵	外鍵
編號	ID	Numeric	9	*	
類別	varchar	50			
名稱	varchar	50			
內容	text	16			

表 9 抓取現在函數範例

```
<%
function webtime()' 抓取現用時間(包括時分秒)
   webtime = date & " " & hour(time) & ":" & minute(time) & ":" & second(time)
end function
%>
```

3. 流程物件模組

流程物件模組則是設定頁面間轉移的先後順序，可以透過系統分析文件中有關於環境圖、流程圖(Flow Chart)、處理描述與藍圖(Drawing)等資訊進行設定。

4. 樣版定義模組

透過樣版定義模組設定，顯示於使用者介面的 View 的呈現畫面，每一個物件都能夠設定其屬性，並可以與檔案物件相互搭配。

表 10 「使用者介面」CSS 設定檔案範例

```
a:active
{
    FONT-FAMILY: 新細明體;
    FONT-SIZE: 11pt;
    TEXT-DECORATION: none;
}
a:hover
{
    FONT-FAMILY: 新細明體;
    FONT-SIZE: 11pt;
    TEXT-DECORATION: none;
}
```

表 11 CSS 檔資料庫表格

資料元素名稱	欄位名稱	欄位型態	欄位長度	主鍵	外鍵
id	id	numeric	9	NOT NULL,PK	
形態	形態	varchar	50		

5. 結果輸出模組

結果輸出模組最後將會輸出包括：外部環境組態檔(XML file)、外部版面(Web Page)、共用函數(include file)與商業邏輯程式碼(Servlet or Bean/ Engine)。結果輸出模組最後將會輸出包括：外部環境組態檔(XML file)、外部版面(Web Page)、共用函數(include file)與商業邏輯程式碼(Servlet or Bean/ Engine)。

6. 環境設定模組

環境設定模組則是提供了對於系統模組設定、系統版面設定、使用者帳號管理等環境資訊設定，在 Web 互動式應用程式的操作環境，本模組負責整個 MVC 設計模式中 View 的展現，使用介面可以自由更改顯示的樣式，但不會影響系統其他的功能。其中系統模組是依照系統分析規格描述所產生，系統版面設定則與使用者帳戶管理相互搭配。

表 12 模組檔資料庫表格

資料元素名稱	欄位名稱	欄位型態	欄位長度	主鍵	外鍵
id	id	decimal	9	*	
模組	模組	nvarchar	255		
排序	排序	decimal	9		
形態	形態	smallint	2		
預設	預設	smallint	2		
連結	連結	nvarchar	255		
顏色	顏色	nvarchar	50		
底色	底色	nvarchar	50		

7. 報表產生模組

應用系統多半需要具備報表產生功能，而報表產生模組則是設計用來產生與應用系統相關資料，提供使用者儲存為一般文件或是試算表型態文件，報表的內容則依據使用者需求塑模階段產生規格，結構化塑模主要的工具可以利用事件、環境圖、流程圖(Flow Chart)、處理描述、藍圖(Drawing)、資料詞彙(Data Glossary)、實體關係圖、關聯表、正規化關聯表、資料流程圖(Data Flow Diagram)、結構圖、HIPO 與處理規格描述等工具，詳細討論可見[23]。

第四章、系統實作與評估

本章節將以一個行政事務管理系統為實際個案，利用本論文提出的系統完成實際個案的實驗，透過 meta-data 輸入的建立與分析的結果實作該應用系統，並產生該系統的文件、程式頁面等資料，輸出的結果均能在很短的時間立即產生。第一節為本研究所採用的實際個案之介紹，說明該案例的內容與需求；第二節則是利用第三章中所描述的方法建構個案之應用系統，透過建構的過程能夠更了解與驗證本論文所設計的產生器；第三節為評估與結論。

4.1 研究個案說明

本個案主要是以提供竹福企業日常行政庶務管理的電腦系統為例，其管理內容包括了：人員的每日出缺勤管理，能夠將出缺勤的資料予以電腦化，並且處理人員請假的核定與申請等作業，以了解人員的出缺勤狀況，可做為考績與獎懲的參考資料來源。物品採購與庫存管理則是以企業日常性的庶務性用品為主，若是有任何需要的文具、紙張等用品，提供採購與庫存電腦化的作業，讓物品的使用資料更透明，申請的手續與管理也更為簡便，而物品採購則是提供總務人員管理聯絡廠商資料、採買記錄。而車輛派遣管理則是讓竹福企業的公務車輛的使用，預約與登記電腦化，各單位人員若是需要使用公務車或派遣等，都可向總務單位登記與預約。而會議室使用管理則是預約登記未來某時間想要使用的會議室，透過在網路上提前登記與申請，可以將申請的流程透明且簡便，讓管理者在管理會議室時，也省下若干溝通與協調的時間，增進管理與作業上的效率。

該行政事務管理系統的系統目標為：

1. 建構企業內部線上申請作業與管理的資訊系統，有助於日常營運的作業。
2. 資訊系統能夠產生相關報表，以利管理作業以及相關決策的制定。
3. 系統能夠方便擴充，讓維護作業簡易節省經費。

該行政事務管理系統所提供的功能為：

1. 系統管理

提供系統管理者針對系統經常性需要管理之功能，包括使用者帳號管理、組織管理、系統權限設定、資料異動管理以及系統版面設定…等功能。

2. 出勤管理

出勤管理功能主要是處理人員假別的核定、申請、查詢、報表列印、上下班出缺勤的資料管理，可做為其他資訊作業處理的資料來源，如薪資計算與獎金發放等功能。

3. 採購管理

採購管理功能主要是提供採購人員將文具辦公用具以及日常庶務用品資料，做歸檔與彙整的功能。

4. 物品管理

當訂購後的物品購入後，將其資料鍵入彙集，於發放給使用者時，做有效的清點與庫存管理等功能。

5. 車輛管理

車輛管理功能係為讓公務車的派遣與使用，得到妥善的管理與控制，有使用需求的人便登記預約車輛。

6. 會議室管理

對於目前可使用的會議室，依據所登記的先後順序與閒置狀況，排訂給需要預定的使用者，管理者可透過適當的協調將會議室出借給需要的使用者。

4.2 建構個案-以行政事務管理系統為例

本小節主要是以 4.1 節中所描述的個案為例，利用 3.2 節中所設計的產生器為工具，建構竹福企業行政事務管理系統，圖 4-1 建構應用系統狀態圖是用來說明本小節建構個案的流程，登入本系統後，便可以進行系統資料的建置，而建立的資料包括該應用系統的 Meata-data 與 Page View，接著是將建立函式庫與環境變數資料，直到這些前置作業都完成後，便可以進行程式碼產生工作。

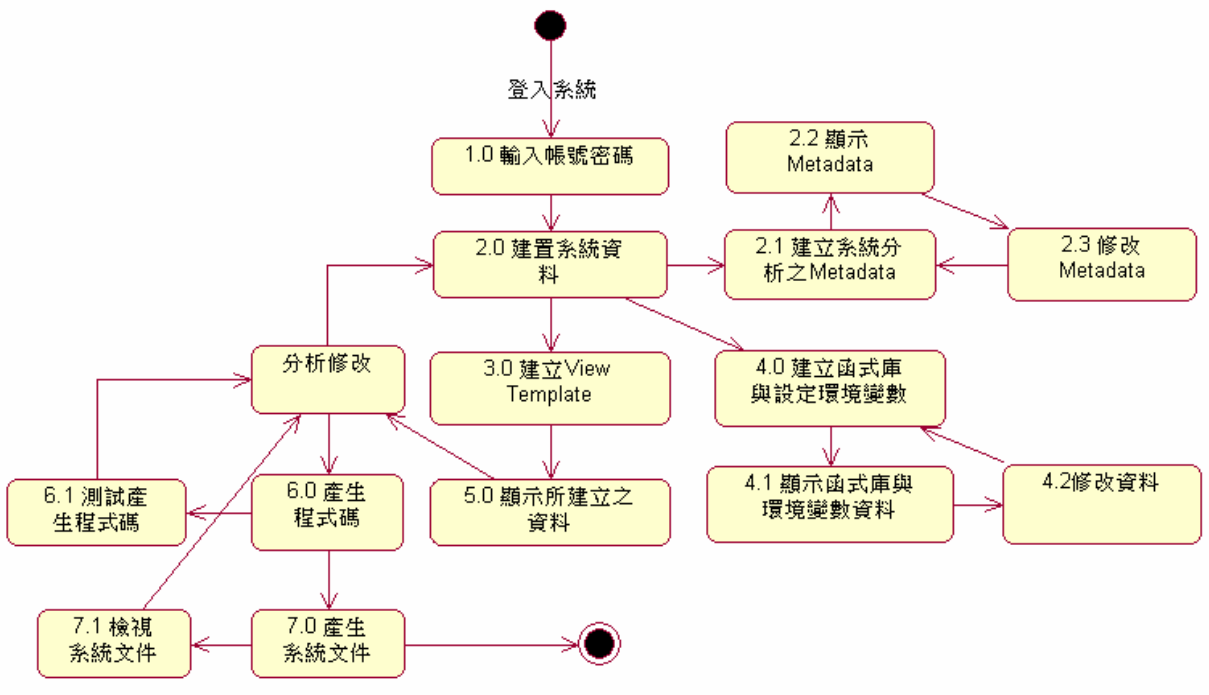

圖 4- 1 建構應用系統狀態圖

我們將使用者與系統的互動關係以圖 4-2 表示，首先把竹福企業行政事務管理系統的 data model 輸入到資料庫中，建立行政事務管理系統的 domain entity，當資料建置完成後，Delegate 便將請求傳給 Generator Controller，由 Controller 產生資料，再將結果展現給使用者。

以下我們依照 3.2 節中的系統流程說明整個案例的產生過程：

階段一：建立 Model

步驟1.　依照關聯式檔案資料輸入至 Model 中

本步驟是將資料表新增至資料庫中，包括設定資料表的名稱，資料欄位的名稱、屬性與型態，資料來源則可由系統分析後的類別圖所取得，竹福企業行政事務管理系統的類別資料與設計規格由於礙於篇幅，也並不是本論文的研究重點，故在此並不詳細列出，圖 4-2 是資料表格的新增建構畫面。

圖 4- 2 資料表格的新增建構畫面

步驟2.　函數與副程式碼建立

本步驟是將常使用的函數建立至資料庫中，函數的管理與分類則依據建立人員的判斷，以圖 4-3 為例，便是設計一個函數將阿拉伯數字轉換成大寫國字，而建立的程式都會在輸出階段一起產生。

類別	轉換類
名稱	將數字轉為大寫國字
內容	<% function chang3(num) %> <% num = replace(num,"1","壹") %> <% num = replace(num,"2","貳") %> <% num = replace(num,"3","參") %> <% num = replace(num,"4","肆") %> <% num = replace(num,"5","伍") %> <% num = replace(num,"6","陸") %> <% num = replace(num,"7","柒") %> <% num = replace(num,"8","捌") %> <% num = replace(num,"9","玖") %> <% if len(num) >= 2 and not right(num,1) = "0" then %> <% num = left(num,1)&"拾"&right(num,1) %> <% end if %> <% if left(num,1) = "壹" then %> <% num = right(num,2) %> <% end if %> <% chang3 = num %> <% end function %>

圖 4- 3 建立函數範例

階段二：建立 View Page

步驟1.　使用者頁面建立

本步驟是使用者頁面設計人員將畫面設計好，也就是 HTML 的 Web-Page，這些顯示的頁面將做為輸出時統一的畫面，頁面層次的設定方式可以依據子系統別做群組管理，分析的方法建議可以利用 2.2.2 節中所介紹的 PAC 架構來描述，以圖 4- 4 為例就是提供使用者進行系統畫面的設定。

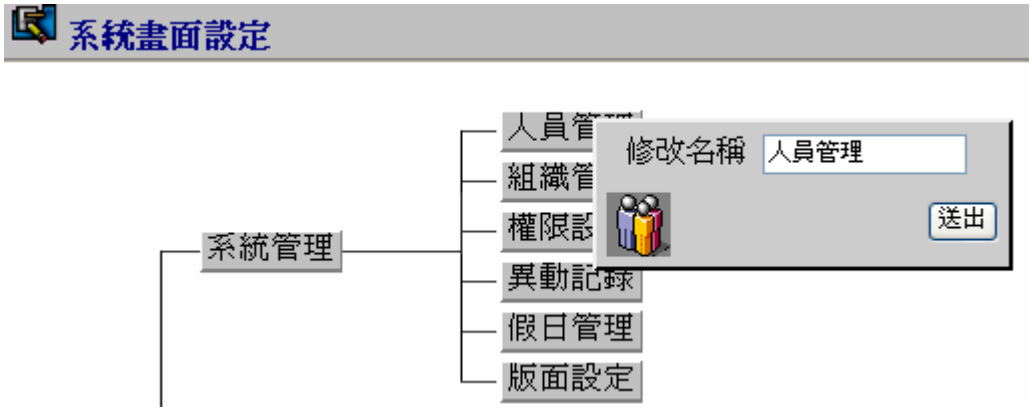

圖 4- 4 系統畫面設定範例

步驟2.　圖檔與其他資料

圖檔資料建立是提供使用者介面的顯示樣式的設定，使用者介面與程式邏輯間的分離，能夠讓 View 的設計與 Model 的設計分離，以達到 MVC 設計樣式的好處。

分類	檔名	
其他類	11.gif	
其他類	12.gif	
其他類	13.gif	
其他類	14.gif	
其他類	15.gif	
列印類	16.gif	
其他類	17.gif	
列印類	18.gif	

圖 4- 5 設定引用圖檔資料

步驟3.　環境資料設定

此處的環境資料設定，依據所使用的系統與語言而不同，常見的設定如 Web Server 的目錄環境、環境設定檔(Config.xml 或 Config.CSS)…等。

階段二：產生階段

步驟1.　產生系統目錄(System Directory)

步驟2.　產生環境設定檔(Configuration)

步驟3.　產生函數檔案(Function and sub-routine)

4.3 評估與結論

本系統測試平台實作了「應用系統程式產生器」，資料庫建立完成後，依據產生的不同檔案類別產生文件與程式碼。測試資料庫則總共產生 54 表格、653 個欄位，及 393 個程式頁面。

藉由 View 與 Model 分離的概念，便能在設計產生器時快速的支援系統開發的工作，雖然在支援功能上功能尚未能完備，但是透過 MVC 設計模式的概念，應用到程式產生器中，可提供未來的開發人員一個設計的參考；由 1.12.2.1 一節中討論可知，PSM 轉出 Code 部份尚未有相關之研究，本論文可提供未來研究者於設計時的一個參考。

在實作的過程中也以實際的案例顯示了本論文提出方法的可行性，產生出的應用系統通常無法一次滿足使用者需求，可以依據 3.2.3 節中所描述的流程，不斷的修改，直到符合使用者需求為止。實作功能的完整性並不是本論文的研究重點，以 Rose Enterprise 為例，雖然使用 UML 進行塑模分析，但是該軟體的價格要 40 多萬台幣，對於預算短缺的中小企業而言一筆龐大的費用，再者，Rose 所產生的程式碼，多以類別、介面、方法等宣告產生為主，在企業邏輯的輔助設計上並不完整。透過產生器設計的改良後自行設計，輔以其他軟體工程上的分析方法，相信能夠提供未來設計者與企業一個更好的選擇。

第五章、結論與未來研究方向

本章節主要分為兩個部份，第一節總結本研究的成果與貢獻，第二節提出研究的限制與未來的研究方向，期盼後續研究者能夠有進一步的研究。

5.1 研究成果與貢獻

本研究所提出的設計架構，研究成果整理如以下幾點說明：

1. 應用系統程式產生器概念與設計

本研究由文獻與研究報告中整理出應用系統產生器的各種類型，分別討論不同類型的產生器設計的架構，以範例分別說明，並討論現行相關軟體技術的發展，設計人員可以依據各別的需求設計開發出合適的產生器。

2. 應用系統程式產生器的改良

本研究提出一個以 MVC 架構設計為基礎的產生器，透過模組化與樣版程式的方式，可以使得產生出的程式邏輯與畫面分離，設計人員可以依據分析人員所塑模出的分析結果，直接將塑模資料鍵入產生器系統中，用以產生所需的應用系統。

3. 個案實作

本研究以一個實際個案，透過系統實驗的操作產生實際的程式碼與文件，做為範例系統。

本研究的貢獻可由以下幾點說明：

1. 由於編譯式的程式語言牽涉到編譯程式的設計與其他的問題，故本研

究中便以直譯式語言為主的應用程式產生器，改進目前市面上只以編譯式語言的產生器的不足。

2. 發展一個以 Web-based 環境的網路應用程式，這也是目前資訊系統開發的主流之一，在實務上十分需要良好的方法增進開發的效率；網路應用程式的環境又可以克服單機版本一次只能由單一使用者者操作的缺點。

3. 將系統設計綱要資料儲存於資料庫，藉由維護綱要資料來完成系統設計與開發。

4. 分析各個產生器類型之優劣，並提出以 MVC 架構為基礎，作為設計與開發的方法，提升設計的速度。

5. 結合 2.4.1 節中所討論的 MDA 設計模式，利用本研究之設計平台架構做為 Web PSM to Source Code 的基礎架構，使得整體系統開發的方法論更臻於完備。

5.2 研究限制與未來研究方向

本研究的研究限制可以分為以下幾點：

1. 由於網路應用程式環境下與系統連線時，需要考量到伺服器逾時的問題，例如：SQL Server 預設至多 600 秒、Web Server 預設至多 900 秒而 http connection 則則是約 20 秒。當連線逾時發生，無法預期是因為系統忙碌或是網路壅塞造成當機。

2. 每秒所能產生的檔案個數與時間為線性關係，在系統執行大量 IO 時，需要花費一定的時間，測試結果如圖 6- 1 所示，故當資料量超過 Web Server 或資料庫伺服器所能負荷時，便會停止運作，故於產生時必須考量輸出檔案的個數；建議解決的方式可以依據子系統功能別劃分，一次只產生一個子系統，或是利用 2.2.2 節中的 PAC 模式的階層式方式

分批輸出產生資料。

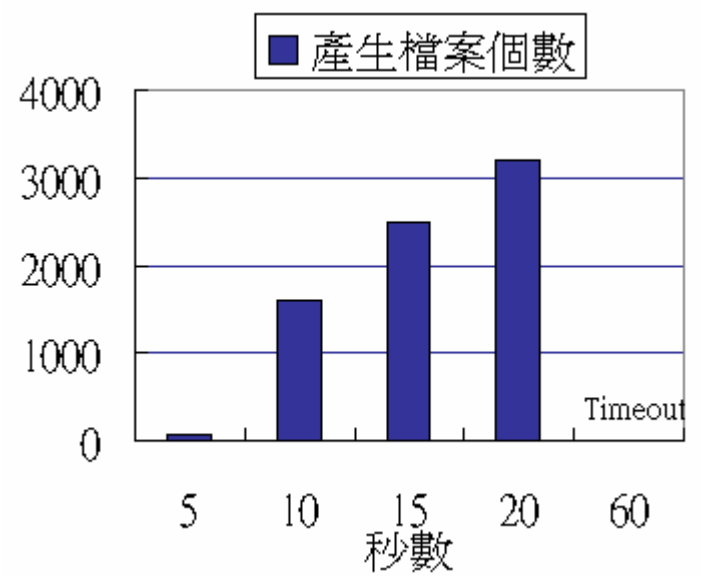

圖 6- 1 產生檔案個數

未來的研究方向

本研究主要的目的是以 MVC 架構和 Web-Based 環境下設計建置應用程式產生器，讓開發人員能夠透過這樣的平台與機制可以加速開發的工作。以此一機制為起點，仍有許多問題需要進一步探討，希望未來能夠在此架構上，朝以下幾個方面進行：

1. 目前產生器主要多半針對單一語言為主，未來應結合 UML 方式減少使用不同語言時開發設計的時間。
2. 將儲存於資料庫中的 meta-data，以 XML 的格式儲存，以建立一個完善的 Tag Library，使得在未來資訊系統開發時，能夠利用更完整的函式庫進行開發工作。
3. 期望能夠在未來提供一個以 Web Services 為基礎的環境，將產生的程式或文件資訊透過以 SOAP 協定，分享並傳送系統分析資訊於所需要的使用者，讓產生器中的各個功能提供給開發團對中的成員，建立一個更完善應用系統產生環境。

參考文獻

英文文獻

[1]. Chen S., Lu F. Y. “Web-based simulations of power systems”, IEEE Computer Applications in Power , Volume 15, p.37 , Jan. 2002.

[2]. Ladislav Seredi, “A DOM-based MVC multi-modal e-business”, IEEE International Conference on Multimedia and Expo, 2001.

[3]. Satoh Uehara, Osamu Mizuno, Yumi Itou and Tohru Kikuno, “An MVC-based analysis of object-oriented system prototyping for banking related GUI applications-correlationship between OO metrics and efforts for requirement, 1999.

[4]. Jen-Her Wu; Tse-Chih Hsia; I-Chia Chang; Sun-Jen Tsai; Application generator: a framework and methodology for IS construction ,System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on , 6-9 Jan. 2003 Pages:263 – 272.

[5]. Thibault, S. and Consel, C., “A Framework for Application Generator Design,” Proceedings of the 1997 Conference on Software Reusability, 1997.

[6]. Cleaveland, J.C., ”Building Application Generators,” IEEE Software, Vol. 5, No. 4, 1988, pp.25-33.

[7]. Sommerville, I., Software Engineering, Massachusetts: Addison-Wesley, 2000.

[8]. “Code Generation in Action”, Jack Herrington, Manning, July 2003.

[9]. Mellor, S. J., Scott, K., Uhl, A., and Weise, D., MDA Distilled: Principles of Model-Driven Architecture,Addison-Wesley, Boston, 2004.

[10]. Jen-Her Wu, Yen-Chieh Huang and Shin-Shing Shin, “Object-Oriented Analysis and Design: Transformation from Class Diagram to Relational Table and Application Template”, 2004 Journal of Internet Technology.

[11]. RFC 1945 – Hypertext Transfer Protocol – HTTP/1.0

[12]. Extensible Markup Language (XML) 1.1

[13]. Feng XIA, Zhi WANG, Youxian SUN, “A design pattern for holonic manufacturing system in the IEC61499-based model-view-controller framework”, 2003.

中文文獻

[14]. 王證宜，元件式應用系統產生器：應用系統建構方法論，國立中山大學資訊管理學系研究所碩士論文，2003

[15]. 張益嘉，應用系統產生器：之架構與資訊系統塑模方法論，國立中山大學資訊管理學系研究所碩士論文，2001

[16]. 連文達，東海大學資訊與科學研究所，”實作一個電腦輔助軟體工程工具(Case Tool)以提昇軟體發展效率及軟體可維護性”。

[17]. 王有利，元智大學資訊工程所碩士論文，“軟體工程技術標準實際應用效用之分析方法”。

[18]. 蔡舜仁,使用者介面塑模:整合 Net-PAC Model 與 UML 於元件式介面開發之研究，國立中山大學資訊管理學系研究所碩士論文，2001

[19]. 吳仁和、林信惠，系統分析與設計，智勝出版社，台北，2002。

[20]. 黃彥結,類別圖轉關聯表與程式樣版之研究,國立中山大學資訊管理學系研究所碩士論文，2004

[21]. 廖漢君，UML：類別圖轉換至物件關聯模式之研究，國立中山大學資訊管理學系研究所碩士論文，2003

[22]. 吳仁和，物件導向分析方法，智勝出版社，台北，2005。

[23]. 吳仁和、林信惠，系統分析與設計，智勝出版社，台北，2002。

[24]. 曾光輝,軟體元件塑模方法研究,國立中山大學資訊管理系所碩士論文,2002

[25]. 陳鴻明，元件塑模方法論：一個植基於 UML 的方法，2002

[26]. Any Fowler "A Swing Architecture Overview" http://java.sun.com/products/jfc/tsc/articles/architecture [17 March 2003].