使用 SBC 軟體架構整合 MVC 模式之研究
Integrating SBC Software Architecture and MVC Pattern

施凱中 [12],丁建文 [1]*
國立高雄應用科技大學資訊管理系 [1]
威聯通科技-數位看板事業處研發部 [2]
Kai-Chung Shih, Jen-Wen Ding*
KUAS-Information Management[1],QNAP-Digital Signage R&D[2]
Email: rxhivzero@gmail.com, jwding@cc.kuas.edu.tw*

摘　要

現今 MVC(Model-View-Controller)模式隨著軟體規模增加與專業分工趨勢，MVC 模式已被廣泛使用，成為一種主流的軟體模式，可以增加程式重複利用與擴充、分工的可能，並且降低軟體系統之複雜度，各種程式語言、開發工具皆大多支援 MVC 模式。本論文使用 SBC(Structure-Behavior Coalescence)軟體架構，注重其結構行為合一架構的六大模型來進行軟體分析與設計,但目前在 SBC 軟體架構中並沒有特別定義 MVC 模式的設計方法，必須由軟體開發者軟體實作時自行設計 MVC 模式，在軟體分析、設計與實作上皆會造成不一致。本論文將提出 SBC 軟體架構整合 MVC 模式之設計方法，並以電子商務網站為例，說明 SBC 軟體架構與 MVC 模式整合之方法。藉由本論文提出之方法，可在不違背 SBC 軟體架構設計原則的情況下將 SBC 軟體架構與 MVC 模式整合，增加軟體分析、設計與軟體實作一致性。
關鍵字: SBC 架構、MVC 模式、軟體工程、軟體架構

一、緒論

1.1 研究背景

軟體工程發展已久，著重於如何以系統性、組織化和量化方式的方法來有效率與規範的去分析、設計與維護軟體，將管理方法與技術整合，並且產出設計文件與模型來便於軟體開發:系統分析、設計、實作與後期維護、移交等，軟體工程發展之方法論中包含了流程(Process)及表示法(Notation)，流程是軟體開發流程：如瀑布模式、漸增模式等。表示法是以規則與圖形建構軟體模型。隨著軟體工程之發展產生了許多軟體工程方法,如 SBC(Structure-Behavior Coalescence) 或是 UML(Unified Modeling Language)統一建模語言軟體架構等表示法,也產生了軟體模式來解決軟體設計中重複出現常見的各種問題。

而現今 MVC(Model-View-Controller)模式隨著軟體規模的增加與專業分工的必要，MVC 模式已被廣泛使用，成為一種主流的軟體模式，可以增加程式重複利用與擴充、分工的可能，並且降低軟體系統之複雜度，各種程式語言、開發工具皆大多支援 MVC 模式。

1.2 研究動機與目的

目前在 SBC(Structure-Behavior Coalescence)軟體架構中並沒有特別定義 MVC 模式的設計方法，如果在 SBC 軟體架構塑模中並無整合 MVC 模式,就必須由軟體開發者於系統實作時自行設計 MVC 模式，如此一來在系統開發與維護上皆會造成不便，產出之文件與模型並無法整合與顯示出 MVC 模式，可能會導致系統設計與系統實作、維護上的不一致增加差異性。

因此本論文將提出 SBC 軟體架構整合 MVC 模式之設計方法，可以在不違背 SBC 軟體架構設計原則的情況下將 SBC 軟體架構與 MVC 模式整合，於系統設計階段即整合常用於系統實作上的 MVC 模式，增加系統設計與系統實作一致性，也因為產出之系統設計文件、模型即整合 MVC 涉及模式降低了系統實作分工的難度,並且因此增加了系統設計文件、模型對於後續維護與移交的可讀性，本論文將以電子商務網站為例，說明 SBC 軟體架構與 MVC 模式整合之方法。

二、文獻探討

2.1 軟體工程

軟體工程(SoftwareEngineering)是一門研究生產開發軟體所需各方面知識的工程學科,最初是由 1968 年,NATO(北約)的科技委員會招集了將近 50 名一流的程式開發人員、計算機科學家與工業界人士，討論以及制定對於軟體危機的對策，在那次會議中第一次提出了軟體工程的概念。範圍從最開始的定義軟體需求到開發與實際上線後的軟體維護都包括在內。而軟體工程著重於如何以系統性、組織化和量化方式的方法來有效率與規範的去分析、設計與維護軟體，將管理方法與技術整合，並且產出系統設計文件與模型來便於軟體開發:系統分析、設計、實作與後期維護、移交等。軟體工程最主要包括了軟體開發技術以及軟體專案管理兩個內容。

2.2SBC 架構

本論文是使用架構導向的 SBC 軟體架構為表示法，以結構行為合一(Structure-Behavior Coalescence)的架構描述語言為工具，進行整合性的分析，找出軟體系統內的重要構件，藉由構件與

構件之間或與外部使用者之間的互動，建構出軟體系統的軟體架構模型。

軟體架構方法論是使用 SBC 架構描述語言來表示軟體架構，而 SBC 架構描述語言可以用來描述與表示軟體架構的多重觀點(Multiple Views)。SBC 軟體架構的多重觀點可以滿足軟體架構所需要的不同觀點，這些軟體架構觀點包括:結構觀點(View of Structure)、行為觀點(View of Behavior)和其他觀點(View of Others)三者。其他觀點包括了:組織觀點、管控管點、技術觀點、流程觀點、資源整合觀點等等。SBC 軟體架構是使用一個整合模型來描述與表達軟體架構的多重觀點，此模型就稱為軟體架構，如圖 1 所示。軟體架構可以投射出結構觀點；行為觀點；其他觀點等，也可以透過結構觀點、行為觀點和其他觀點的相互合作推導出軟體架構來，如圖 2 所示。

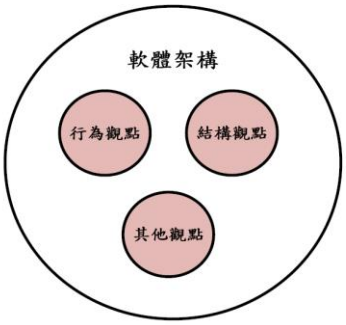

圖 1:軟體架構是一個整合模型
資料來源:系統分析與設計-使用軟體架構模型(趙善中,2008)

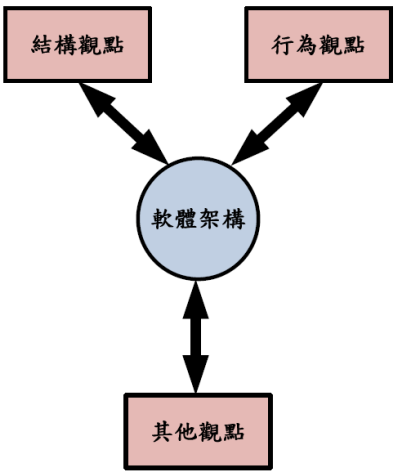

圖 2:軟體架構和觀點之間的投射關係
資料來源:系統分析與設計-使用軟體架構模型(趙善中,2008)

SBC 軟體架構除了以一個整合的軟體架構模型來描述與表達軟體架構的多重觀點之外，並以結構行為合一的六大金律來描繪出軟體結構與軟體行為兩者之間的關係。結構行為合一六大金律能夠以圖形化的描述方式來表示整個軟體架構模型，透過圖形化的表示，可以簡單、清楚的呈現出整個軟體的複雜系統，將結構觀點與行為觀點整合表達出一個完整的軟體架構。此六大金律分別是:架構階層圖(AHD)；框架圖(FD)、構件操作圖(COD)、構件連結圖(CCD)、結構行為合一圖(SBCD)、互動流程圖(IFD)。

2.3 MVC 模式

MVC 是 Model、View、Controller 的縮寫。MVC 模式是由特里夫.里斯高(Trygve Reenskaug)於 1978 年所提出的，MVC 模式源自於 Smalltalk 程式語言，然而 MVC 模式隨著不斷的演變，現在最主要用來實現畫面顯示與程式邏輯以及資料的分離，隨著軟體規模的增加與專業分工的必要，MVC 模式現今已被廣泛使用。透過 MVC 模式，可以將 Model、View、Controller 清楚分工，可以縮短維護的時間和降低軟體系統的複雜度，並有效率的分工合作，增加使用者介面之彈性與重覆使用性，將軟體系統透過 MVC 模式分開後，可以在不影響介面(View)的情況下修改模型(Model)，也可在不動到模型(Model)下去修改介面的呈現(View)，降低軟體系統的耦合性。

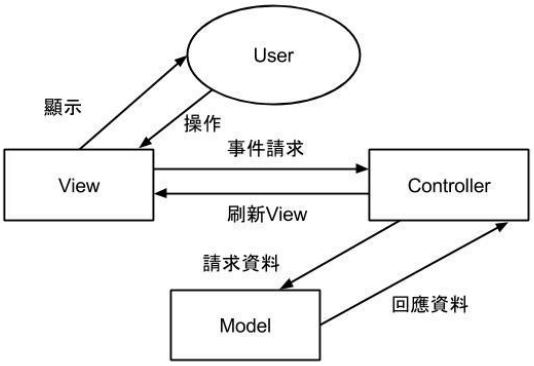

圖 3:MVC 模式的結構圖

三、建置 SBC 軟體架構整合 MVC 模式架構模型

本章節將說明 SBC 軟體架構整合 MVC 模式之塑模方法與步驟，以電子商務行動 APP 為範例，實作完整的 SBC 軟體架構整合 MVC 模式之架構模型。本研究使用 SBC 軟體架構為基礎，在依照其設計原則與步驟的情況下整合 MVC 模式，產出電子商務行動 APP 之 SBC 架構模型。

3.1 塑模步驟與方法

本節將以依照 SBC 軟體架構之設計原則的情況下說明建置 SBC 軟體架構整合 MVC 模式之塑模步驟與方法。

3.1.1 塑模步驟

SBC 軟體架構整合 MVC 模式之塑模步驟依序為:架構階層圖、框架圖、構建操作圖、構建連結圖、結

構行為合一圖、互動流程圖，並在塑模中即可整合 MVC 模式。

3.1.2 塑模方法

SBC 軟體架構之模型將包含:架構階層圖、框架圖、構建操作圖、構建連結圖、結構行為合一圖、互動流程圖。整合 MVC 模式後各項塑模方法如下:

1. 架構階層圖(AHD)塑模方法:

架構階層圖(AHD)可以看出系統之多階層(Multi-Level)的階層與分解組合，在此將母系統進行分解，並且依照架構階層定義之系統分解與組合，在母系統下定義 MVC 之 Model、View、Controller 三類子系統，再分解各類子系統至所需之基本構件。

2. 框架圖(FD)塑模方法:框架圖(FD)可以看出系統之多層級(Multi-Layer)的分解與組合，在此將層級(Multi-Layer)依照 View、Controller、Model 使用者操作傳遞之順序分類。

3. 構建操作圖(COD)塑模方法:構建操作圖(COD)可以看出系統內所有構件之操作與傳送、回傳之參數，在此依照 SBC 軟體架構之定義，對構件的操作與傳送、回傳之參數定義。

4. 構建連結圖(CCD)塑模方法:構建連結圖(CCD)可以看出一個系統的結構概觀，在此依照 SBC 軟體架構定 義，將有關連之構件連結。

5. 結構行為合一圖(SBCD) 塑模方法:結構行為合一圖(SBCD)可以同時看出系統內的構件與使用者行為，可以了解到使用者之行為會使用到之構件與操作行為傳遞之順序，在此依照 SBC 軟體架構定義出使用者對於系統之行為並且依照 View、Controller、Model 使用者操作傳遞之順序進行塑模。

6. 互動流程圖(IFD) 塑模方法:互動流程圖(IFD)可以看出構件與構件的互動與操作行為之事件與參數先後傳遞之順序，在此我們依照 SBC 軟體架構之定義分別製作不同操作行為之互動流程圖(IFD)，並且依照 View、Controller、Model 以及使用者操作事件與參數傳遞之順序進行塑模

3.2 SBC 軟體架構整合 MVC 模式塑模-以行動電子商務 APP 為例

本節將以行動電子商務 APP 為範例，並依照 3.1 敘述的 SBC 軟體架構整合 MVC 模式之塑模步驟進行塑模，逐步地描述 SBC 軟體架構整合 MVC 模式之架構模型與塑模方法。

3.2.1 架構階層圖(AHD)塑模

在架構階層圖中，我們可以看到電子商務行動 APP 系統的階層分解過程，經過第一次的分解，可以得到符合 MVC 模式的主要層面構件;再經過第二次的分解，可以得到細部功能的構件。透過 SBC 之架構階層圖可以讓本來不明確的系統架構，依照 MVC 與功能去分解，讓整個架構變得清晰與明確。

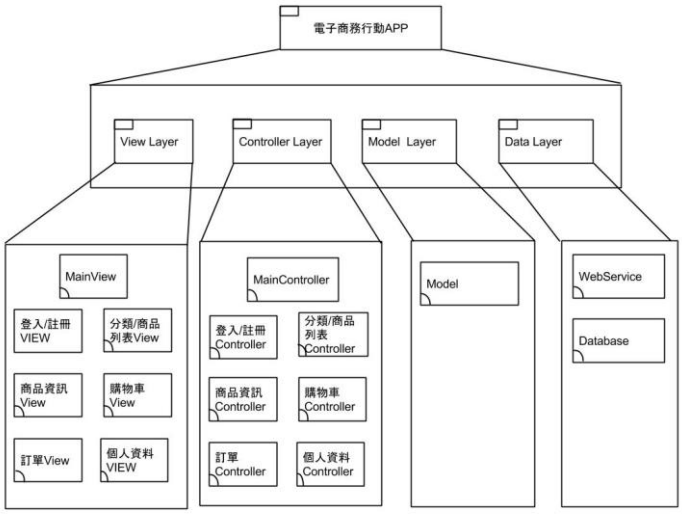

圖 4:架構階層圖(AHD)

3.2.2 框架圖(FD)塑模

框架圖忽略階層的關係，而只考慮構件的組合，可以用來解釋構件組合的系統架構，在此依照「View Layer」、「Controller Layer」、「Model Layer」、「Data Layer」使用者操作傳遞之順序分類，並清楚顯示 MVC 模式之 View、Controller、Model 框架。

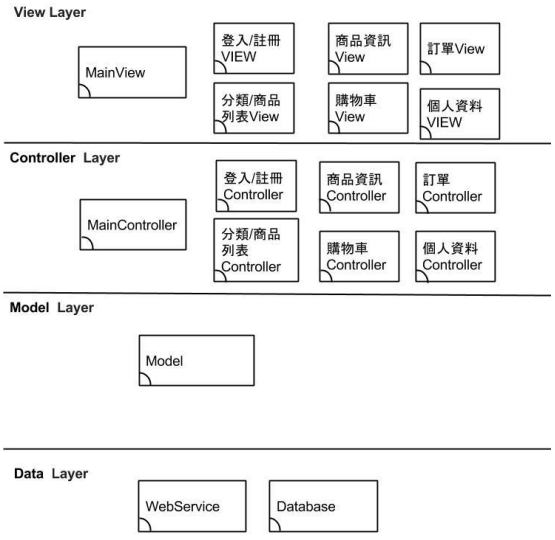

圖 5:框架圖(FD)

3.2.3 構件操作圖(COD)塑模

構件操作圖可以將每一個構件的操作與所傳送的參數都清楚標示出來，而操作是由其他構件或外部環境(使用者等...)呼叫使用的，以下將完整之構件操作圖依照使用者操作之順序分割顯示，並以表 2 構件操作表說明行動電子商務 APP 的構件操作與傳送之參數。

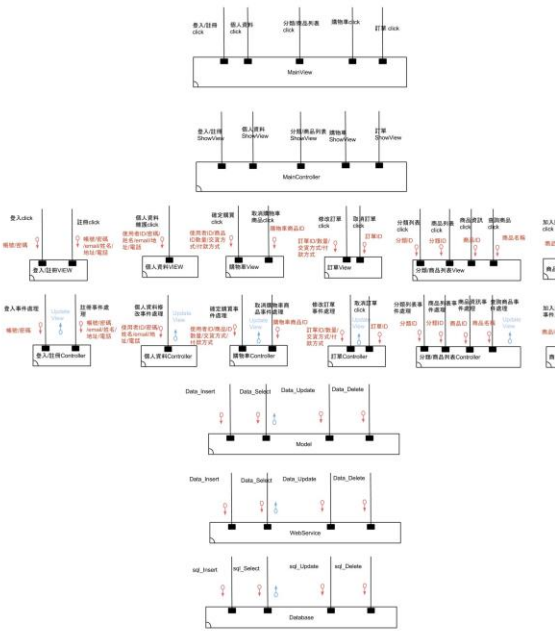

圖 6: 構件操作圖(COD)

3.2.4 構件連結圖(CCD)塑模

構件連結圖可以顯示系統當中各個構件彼此之間的連結，連結就是外部環境與構件以及構件與構件之間的關係，透過構件之間的連結關係，可以讓軟體架構變得更清楚。行動電子商務 APP 構件連結圖的外部環境有 User(使用者)，並且顯示各個構件之連結關係。

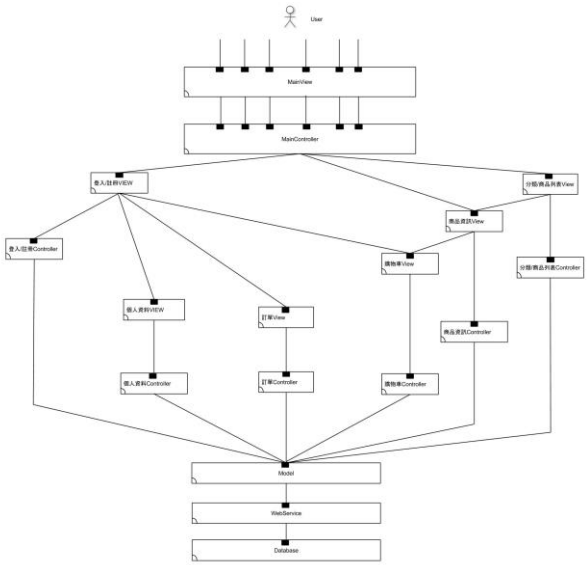

圖 7: 構件連結圖(CCD)

3.2.5 結構行為合一圖(SBCD)塑模

結構行為合一圖可以同時顯示整個系統當中所有的構件，以及與外部環境互動的操作行為，結構行為合一就是將系統中的構件與行為操作合一，並且看出行為會使用到之構件與行為操作傳遞的順序，產生一個結構與行為整合的模型，在此也可以看到 MVC 模式完整的融入了 SBC 軟體架構之結構行為合一圖當中。

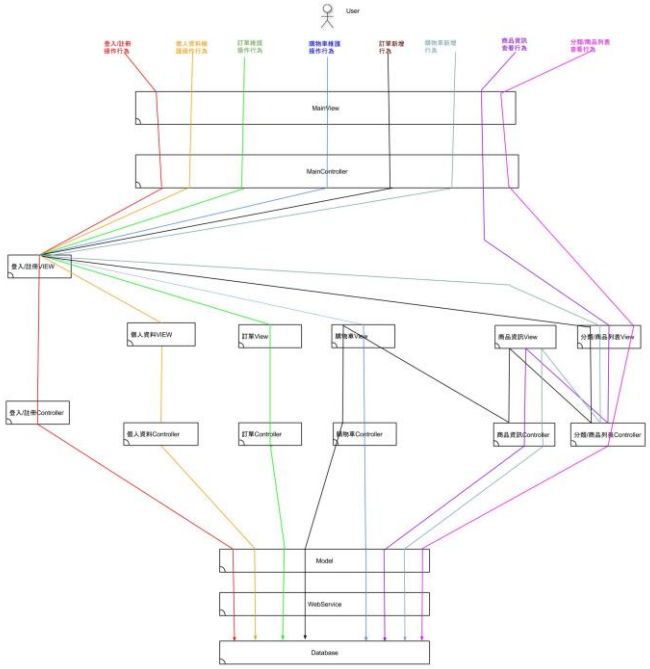

圖 8:結構行為合一圖(SBCD)

在行動電子商務 APP 結構行為合一圖當中，完整的顯示出使用者對於系統的行為連線，而每一條連線就代表著一個系統的操作行為，並且看出所以經過與使用的構件與行為順序與流程，完整的表達 SBC 軟體架構結構行為合一圖的觀點。

3.2.6 互動流程圖(IFD)塑模

互動流程圖是以行為觀點，詳細描述每一個系統的外界環境與構件之間如何互動以及如何操作、資料的流向等，清楚顯示出操作以及執行的順序關係。並且以使用者行為觀點依照 MVC 模式之運作流程，由 View 到 Controller 至 Model，省略後端 Data Layer 由 Model 呈現資料，顯示完整的使用者觀點之 MVC 模式互動流程，互動流程圖對圖 8:結構行為合一圖當中當中八項操作行為進行塑模，得出八組互動流程圖，並能清楚的顯示出這八組操作行為對於系統的操作、使用的構件、執行順序等互動，以下以登入/註冊操作行為、個人資料維護作行為兩組互動流程圖為例。

1. 登入/註冊操作行為

(1). 使用者由主畫面「MainView」點選登入/註冊按鍵後，由「MainController」 進行 ShowView 事件將畫面轉換至「登入/註冊 View」。

(2). 使用者由「登入/註冊 View」點選註冊按鍵，並輸入註冊所需參數:帳號、密碼、email、姓名、電話、住址，後由「登入/註冊 Controller」進行註冊事件處理並將參數傳送至「Model」進行註冊帳號，而「Model」對後端進行新增資料後回傳註冊成功/失敗訊息給「登入/註冊 Controller」,「登入/註冊 Controller」再將訊息回傳至「登入/註冊 View」並刷新 View 通知使用者註冊成功/失敗。

(3). 使用者由「登入/註冊 View」點選登入按鍵，並輸入註冊所需參數:帳號、密碼，後由「登入/註冊 Controller」進行登入事件處理並將參數傳

送至「Model」進行登入資料查詢，而「Model」對後端進行查詢登入資料後回傳登入成功/失敗訊息給「登入/註冊 Controller」,「登入/註冊 Controller」再將訊息回傳至「登入/註冊 View」並刷新 View 通知使用者登入成功/失敗。

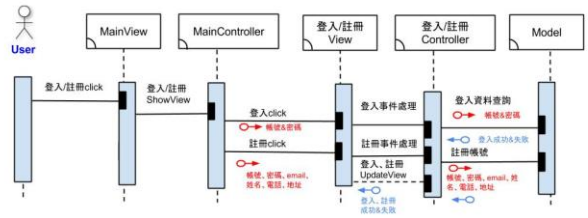

圖 1:互動流程圖(IFD)-登入/註冊操作行為

2.個人資料維護操作行為

(1).使用者由主畫面「MainView」點選個人資料維護按鍵後，由「MainController」進行登入判斷事件處理來判斷使用者是否已經進行登入， 如果尚未登入就執行登入/註冊 ShowView 事件將畫面轉換至「登入/註冊 View」進行登入或是註冊，而如果已經登入則執行登入/註冊 ShowView 事件將畫面轉換至「個人資料 View」。

(2).使用者由「個人資料 View」點選個人資料修改按鍵，並輸入個人資料修改所需參數:密碼、姓名、email、地址、電話並且加上不可修改之使用者 ID，後由「個人資料 Controller」進行個人資料修改事件處理並將參數傳送至「Model」進行個人資料修改，而「Model」對後端進行個人資料修改後回傳修改成功/失敗訊息給「個人資料 Controller」,「個人資料 Controller」再將訊息回傳至「個人資料 View」並刷新 View 通知使用者修改成功/失敗。

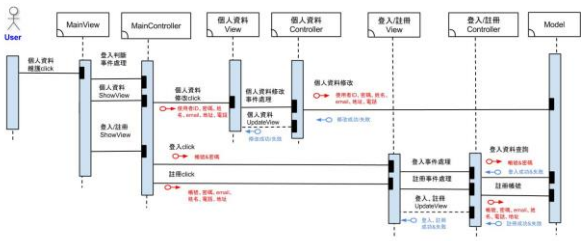

圖 2:互動流程圖(IFD)-個人資料維護作行為

四、效益評估

本論文提出的使用 SBC 軟體架構整合 MVC 模式方法可以在不違背 SBC 設計原則的情況下整合 MVC 模式，以軟體流程來說同樣是依照需求分析、軟體設計、軟體實作、測試、維護等五個軟體開發階段來進行軟體開發，而本論文提出的方法可以將軟體分析與設計階段產出之 SBC 軟體架構模型與軟體實作階段使用的軟體技術 MVC 模式整合，提高軟體分析與設計階段與軟體實作的一致性，也降低了後續維護難度，並使軟體分析與設計產出之模型即可表示出 MVC 模式，增加了分析與設計階段之專業分工可行性，以下將詳細解釋本論文方法可能產生的優點與缺點。

4.2.1 本論文方法之優點

1.增加軟體設計與實作一致性:於軟體分析與設計階段產出之 SBC 架構模型即可顯示 MVC 模式，如此一來程式開發者可以直接依照 SBC 軟體架構進行軟體實作，並不需要另外設計 MVC 模式，會增加軟體分析、設計與軟體實作以及維護上的一致性。

2.專業分工與提升效率: 於軟體分析與設計階段產出之 SBC 架構模型即可顯示 MVC 模式，可以於軟體分析與設計階段便依照 MVC 模式來進行分工，這樣可以更清楚地進行專業分工與提升效率。

3.程式的可再利用性:MVC 的特點就是利用 Model、View、Controller 的三層式概念來進行分工，與提高程式的可再利用性，本論文提出之方法讓軟體分析與設計階段便整合 MVC 模式，可以於分析與設計階段設計最大共同利用之 Model、View、Controller，降低軟體實作階段多人分工開發之複雜度。

4.2.1 本論文方法之缺點

1.並無量化資料驗證:本論文提出之 SBC 軟體架構整合 MVC 模式之方法，雖預期產生上述之效益，但目前尚無量化統計資料可以驗證，僅依照少數案例來進行推測評估，尚無法確定在各式各樣的軟體專案上使用本論文之方法是否會產生同樣的預期效益。

2.軟體實作自由度:使用本論文之方法可能降低了軟體開發者進行軟體實作時的自由度，讓軟體實作依照於 SBC 軟體架構來實作 MVC 模式與軟體系統。

表 1:效益評估表

	現行開發方式	本論文提出方法
SBC 軟體架構使用 MVC 模式之方法	於軟體實作階段由軟體開發者設計與實作 MVC 模式	SBC 軟體架構模型即可顯示出 MVC 模式
軟體實作自由度	較高	較低
軟體設計與實作一致性	較低	較高
專業分工難度	較不易	較容易
程式的可再利用性	較低	較高
對應各種案例	經過驗證，能對應多數案例	對應特定案例
方法驗證	經過驗證	尚無量化資料

五、結論

本論文回顧軟體工程、SBC 軟體架構與 MVC 模式，並提出 SBC 軟體架構整合 MVC 模式之方法，此方法可以在不違背 SBC 設計原則的情況下整合 MVC 模式，並將軟體開發流程當中軟體分析與軟體設計階段之軟體架構模型與通常於軟體實作階段使用的軟體技術 MVC 模式整合，如此一來預期可以提高軟體分析與設計階段與軟體實作的一致性，並且讓軟體分析與設計產出之文件模型即可表示出 MVC 模式，增加了分析與設計階段之專業分工可行性，也降低了後續維護難度。

參考文獻

[1] 趙善中、康中倫、余遠航,2008,系統分析與設計-使用軟體架構模型,碩博文化。

[2] 趙善中、孫述平、韓孟麒,2013,系統學 2.0:使用 SBC 架構描述語言,阜盛文教。

[3] 趙善中、郭麗齡、尤炳文,2009,軟體工程-使用軟體架構模型,儒林。

[4] 吳能和,2010,物件導向系統分析與設計結合 MDA 與 UML,知勝。

[5] Ian Sommerville, 2007, "Software Engineering_" , Addison-Wesley.

[6] Chao,W.S,2011, "Software Architecture:SBC Architecture at Work",National Sun Yat-sen University Press.

[7] Shams Mukhtar, 2004 , "Applying Robustness Analysis on the Model-View-Controller (MVC) Architecture in ASP.NET Framework, using UML" ,http://www.codeproject.com/Articles/8058/Applying-Robustness-Analysis-on-the-Model-View-Con.

[8] Wei-Ming Ma, 2010 , "Study on Architecture-Oriented Information Security Risk Assessment Model " , Second International Conference, ICCCI 2010, Kaohsiung, Taiwan, November 10-12, 2010. Proceedings, Part III.

[9] Syun Sheng Jhan., et al , 2003 , "Analysis and Design of Semantic Web Services Using MDA and SBC Structure, Applied Mechanics and Materials (Volumes 284 - 287)

[10] Erich Gamma., et al, 1995, "_Design Patterns", Addison-Wesley.

[11] Shuh-Ping Sun, William S. Chao," An Architecture-Oriented Design Method For Innovative Service Systems", http://www.aea-taiwan.org

[12]WIKIPEDIA, "Software architecture", http://en.wikipedia.org/wiki/Software_architecture

[13] A Leff, JT Rayfield , 2001, " Web-application development using the model/view/controller design pattern", Enterprise Distributed Object Computing Conference,2001. EDOC'01. Proceedings. Fifth IEEE International,pp118-127.

[14] Collins, D, 1995, " Designing Object-Oriented User Interfaces ", Redwood City, CA: Benjamin/Cummings.

[15] Erich Gamma., et al., 2008, "What Are The Benefits of MVC", http://blog.iandavis.com/2008/12/09/what-are-the-benefits-of-mvc/.

[16] Martin Fowler, 2006, "history of UI Architectures and the evolution of MVC",http://martinfowler.com/eaaDev/uiArchs.html.

[17] David Lowe, Brian Henderson-Sellers, Alice Gu , 2003, " Web extensions to UML: Using the MVC Triad ", Lecture Notes in Computer Science Volume 2503, pp 105-119.

[18] " ASP.NET MVC Overview", Microsoft MSDN, http://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx

[19] " Cocoa Core Competencies ", iOS Developer Library, https://developer.apple.com/library/ios/documentation/general/conceptual/devpedia-cocoacore/MVC.html

[20] " Intro to Applications with Sencha Touch ", Sencha, http://docs.sencha.com/touch/2.3.1/#!/guide/apps_intro