視覺化開發策略下軟體模式化技術之研究

A Research of Software Modeling Technology Suitable for Visual Development

程裕繁　　尤松文　　劉文卿
0931-409899
NantouBill@cttv.tinp.net.tw　　oilwarm@mis.nccu.edu.tw　　wliou@nccu.edu.tw

國立政治大學資訊管理學系
116 台北市文山區指南路二段 64 號

摘要

為了解決視覺化開發軟體專案缺乏模式化技術的問題，本文作者嘗試賦予傳統模式化工具新的意義與原則，其從結構化技術、物件導向技術，和人機介面技術的模式化工具中整理出一套適用於視覺化開發軟體專案的模式化技術。本文陳述這個模式化技術，說明視覺化開發軟體專案在軟體分析與軟體設計這兩個開發活動中應該模式化哪些部份以及應該使用哪些模式化工具來予以製作成軟體模式。

關鍵字：軟體模式化、圖表型式的模式化工具、元件基礎軟體工程、視覺化程式設計、資料中心企業資訊系統

ABSTRACT

Currently, there has no software modeling technology that is suitable for visual development. Therefore, a new software modeling technology was created by adjusting the modeling tools of structured technology, object-oriented technology, user interface technology and so forth. This paper presents the new software modeling technology, which describes the needs of modeling in analysis and design activities for visual developing software projects, as well as the suitable modeling tools for this.

Keywords: Software Modeling, Diagrammatical Modeling Tools, CBSE, Visual Programming, Data-Centric Business ISs

一、前言

軟體工程領域已經逐漸形成了沒有萬用軟體方法的共識,軟體專案應該視情況選用合適的軟體方法(權變理論)。Fowler〔5〕,Jacobson〔7〕,Sommerville〔18〕,van Slooten〔21〕,Wiegers〔24〕,Yourdon〔26〕等人都曾經在文獻中明白支持這個論點。

另一方面,刊登在 1999 年 Computerworld 的一篇調查報告指出,1998 年在美國業界有 33.5%的程式設計師選擇使用 Visual Basic,而且在未來幾年 Visual Basic 仍然會繼續保持領先地位〔11〕。如果再加上 Delphi, Power Builder 等工具的使用人口,可以合理相信,有為數眾多的軟體開發專案採取了視覺化開發策略,使用這些視覺化程式設計工具(Visual Programming Tools)來予以開發。

有一個不太被認知的嚴重問題是,軟體方法明顯缺了一塊。對於採取視覺化開發策略的軟體專案而言,目前並沒有一個完整的分析設計方法可供應用,仍然需要許多人的共同努力才可能解決。Llewellyn〔8〕,Orfali〔12〕,Philiph〔13〕,Pressman〔14〕,Wallnau〔22〕等人都曾經在文獻中指出這個問題,並嘗試提供部份問題的解決方案。

上述提及之文獻中,Llewellyn 與 Orfali 均是提出視覺化開發軟體專案的進行程序,藉以指引開發活動的進行。另外 Philiph 則嘗試為視覺化程式設計提供一些好的設計準則。而 Pressman 則提出了一些簡單的調校建議,企圖讓開發人員能夠自行彈性應用結構化方法或是物件導向方法於視覺化開發軟體專案。至於 Wallnau 則是認為以商用軟體元件組裝的做法來開發軟體已經是業界事實,然學界的軟體方法與實務之間出現了大落差,故而建議了一些軟體方法的改進方向。

流程與模式化工具[1]是軟體方法的兩個主要部份〔3〕。而文獻探討與實務經驗都顯示視覺化開發軟體專案的模式化仍然是一個需要被解決的問題。本文作者嘗試賦予傳統模式化工具新的意義與原則,其從結構化技術〔3,10,14,18,20,25〕、物件導向技術〔2,5,6,7,14,16,18〕,和人機介面技術〔14,17,18,23〕的模式化工具中整理出一套適用於視覺化開發軟體專案的模式化技術－VMVP(Visual Modeling technology suitable for Visual Programming)。本文先概念性探討視覺化開發策略(第二節),然後說明 VMVP 的基礎論述與實務(第三節),最後則說明在 VMVP 之下視覺化開發軟體專案在軟體分析與軟體設計這兩個開發活動中應該模式化哪些部份以及應該使用哪些模式化工具來予以製作成軟體模式(第四節)。

二、視覺化開發策略

當軟體專案決定在實作活動中採用視覺化程式設計工具並且以視覺化程式設計方式來撰寫程式時,這個軟體專案就是本文所謂的採取視覺化開發策略的軟體專案,簡稱視覺化開發軟體專案。

而本文所謂的視覺化程式設計工具係指 Visual Basic, Delphi, Visual C++, C++ Builder, Power Builder, JBuilder, Visual Object, Visual Cafe, VisualAge 等軟體開發工具。這些開發工具的主要組成為視覺化軟體元件(Visual Software Components)、視窗編輯器(Window Editor)、報表編輯器(Report Editor)、程式編輯器(Code Editor)、程式編譯器(Compiler)、程式連結器(Linker)、程式除錯器(Debugger)等。

雖然可以祇使用程式編輯器、程式編譯器、程式連結器、程式除錯器,並以逐行撰寫(Hard-Coding)的方式來寫程式,但這祇是使用這些方便的工具來進行程序性程式設計或是物件導向程式設計,並不是一種視覺化程式設計。本文所謂的視覺化程式設計係指程式設計師以現成的視覺化軟體元件為基礎,在視窗編輯器上組裝出軟體的程式設計方式。這種程式設計方式是以元件為基礎的(Component-Based),是視覺化開發的(Visual-Developed),是事件驅動的(Event-Driven)。其過程大致如下〔4,9,17〕:

1. 產生基礎視窗。
2. 在基礎視窗上放置元件。
3. 設定必要的屬性值。
4. 撰寫必要的事件處理程序(Event Handlers)。

有部份人士認為視覺化開發策略祇適用於比較簡單的軟體系統,其實像編譯器或是資料庫管理系統這種比較複雜的系統軟體如果有品質與數量都足夠的元件可供使用仍然適合採取視覺化開發策略。真正影響視覺化開發策略可行性的是該類型軟體系統有沒有足夠的市場利益來促進元件的產生。對於某些領域的軟體而言軟體元件的數量與品質都已經沒有問題。Llewellyn〔8〕, Maxim〔9〕, Sommerville〔18〕都認為資料中心企業資訊系統(Data-Centric Business Information Systems)已經可以使用視覺化開發策略來完成。資料中心企業資訊系統也有人叫做資料處理系統(Data Processing Systems)、資料密集企業程式(Data-Intensive Business Programs),或是資料庫應用(Database Applications)。這種軟體系統透過資料的處理來輔助企業的營運、控制,與管理。截至目前為止,企業對於這種軟體系統的需求

[1] 本文所指稱的模式化工具均為圖表型式的模式化工具(Diagrammatical Modeling Tools)。

量還是非常的龐大〔19〕。

也有部份人士對於物件導向技術與視覺化程式設計之間的關係存在著迷思，有必要予以說明清楚。物件導向技術是視覺化程式設計的核心，視覺化程式設計所使用的是物件導向或是物件基礎的程式語言，而且視覺化軟體元件根本就是一個類別。但是如前面所說明的，視覺化程式設計是一種以現成視覺化軟體元件為基礎並以視窗為開發單位的程式組裝過程，這種過程與物件導向技術的創造並使用類別的思維是截然不同的。而且在這種程式撰寫方式之下開發人員並不在乎是不是使用了物件導向技術，他們關心的是程式有那些功能需要完成，每個功能應該使用幾個視窗來實作，每個視窗需要使用哪些元件來組裝，哪些元件的屬性需要做設定，哪些元件的事件需要做處理。而指引程式撰寫與維護的藍圖－軟體設計文件（Software Design Document, SDD）在程式相關設計方面所應該規範的自然是這些事項。請檢視一下物件導向軟體設計文件的內容，很顯然的以類別圖（Class Diagram）為主體的它並不應該也無法使用於視覺化程式設計（從內容來看，以結構圖〔Structure Chart〕為主體的結構化軟體設計文件也無法使用於視覺化程式設計）。

另外，實務上發現相當數目的視覺化開發軟體專案沒有製作文件，就算是有文件的產出也幾乎是應付了事，純粹是因為客戶或是主管的要求而為了要有文件交差而做出來的文件。造成這個現象的原因之一是軟體工程教育之不足所導致的不瞭解與不重視，另一個是 VMVP 所要解決的技術不足問題，還有一個則是視覺化程式設計之獨特性質所導致的無設計逕行撰寫傾向：程式撰寫的困難度大幅度降低，再加上瑣碎的版面配置、屬性設定、事件處理等作業在開發工具上進行容易，但要單用紙和筆來予以事先規範則很困難，因此開發人員傾向於不做設計也不產出文件直接就進行程式的撰寫。

基於上述之視覺化程式設計的獨特性質，開發人員必需使用開發工具來輔助軟體設計文件的製作。可是在這種做法之下文件完成之時程式也可以完成了泰半，故而先做出軟體設計文件然後再據以撰寫出程式的理想化安排可能並不恰當，整併程式的設計與撰寫讓這兩者能夠平行進行並相互修訂是一個合理的改變。如此能夠讓程式擁有良好的思考與設計的過程，更可以因而留下有益於日後維護活動的設計文件。

三、VMVP 的基礎論述與實務

VMVP 採納了一些軟體工程文獻與軟體開發實務的建議。這些建議是 VMVP 研究的基礎，分項予以說明如下：

1. 軟體開發專案涉及了五項開發活動：**A.** 使用者需求分析（User Needs Analysis），產出使用者需求陳述（User Needs Statements）。**B.** 軟體規格制定（Software Specification），產出軟體需求規格（Software Requirements Specification, SRS）。**C.** 軟體設計（Software Design），產出軟體設計文件。**D.** 軟體實作（Software Implementation），產出程式與資料庫。**E.** 軟體測試（Software Testing），產出驗證過的程式與資料庫。上述的使用者需求分析與軟體規格制定兩項活動一般統稱為軟體分析（Software Analysis）。另外這五項開發活動之間存在著如圖 1 所示的銜接關係。要注意的是，圖 1 並沒有前一個開發活動要「全部」完成之後才可以執行下一個開發活動的意思，事實上這些開發活動應該會以反覆（iterative）與漸增（incremental）的方式來進行〔3,10〕。

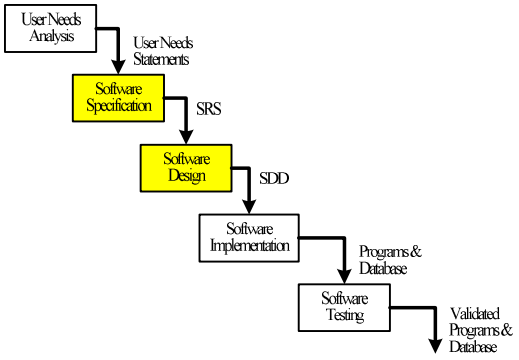

圖1 開發活動之間的銜接關係

2. 軟體需求規格顯示軟體應該具備的各種功能性與非功能性的特徵，軟體設計文件則呈現一個符合軟體需求規格的設計方案，而這兩者的核心內容都是使用模式化工具製作出來的軟體模式〔10〕。這種軟體模式具有下列數種用途：**A.** 促進溝通（開發人員與客戶或使用者之間的溝通、開發人員之間的溝通、開發人員與內部主管之間的溝通）。**B.** 避免錯誤的設計（扮演燈塔的角色，避免做出無法滿足客戶與使用者需求的錯誤設計）。**C.** 引領施工（扮演藍圖的角色，指引實作活動的執行）。**D.** 支援維護（減少因為忘記或是人員流動所造成的維護問題）。**E.** 幫助思考（進行模式化的過程本身就是一種思考的過程，這種過程有助於客戶與使用者需求的釐清，也能促進設計上的完善）〔3,10〕。

3. 不同抽象層次、不同觀點，不同功能的模式化工具可以讓我們從不同的角度來瞭解軟體的重要特徵。而一個軟體需要從多個不同的觀點來進行模式化（功能觀點、資料觀點、依時行為觀點、結構觀點、物件觀點等），亦即必需同時使用多種的模式化工具才能夠對軟體做到完善的表示〔2,3,26〕。
4. 軟體工程領域研究已經陸續提出了很多的模式化工具，例如結構化技術的資料流程圖（Data Flow Diagram, DFD）、結構圖、狀態移轉圖（State Transition Diagram, STD）、實體關連圖（Entity-Relationship Diagram, ERD）等〔3,10,25〕，物件導向技術的使用案例圖（Use-Case Diagram）、類別圖、物件互動圖（Object Interaction Diagram, OID）等〔2,3,5〕都是廣泛被使用的模式化工具。數量龐大的模式化工具應該足以應付各種軟體系統在模式化方面的各種需求。
5. 遵循標準將有利於溝通與維護。物件導向技術的模式化工具應該遵循 Unified Modeling Language（UML）標準〔2,5〕（國際標準）。在結構化技術的模式化工具方面則可以遵循 Yourdon 於 1989 年所著現代結構化分析一書〔25〕所規範的符號標準（業界標準）。
6. 使用模式化工具製作完成的軟體模式是抽象概約而不夠精細的，其目的在於顯示某種抽象層次某個觀點下軟體的重要特徵。因此軟體需求規格與軟體設計文件都需要再使用文字來解釋與補充〔18〕。
7. 採取視覺化開發策略的資料中心企業資訊系統多數使用兩層主從架構（Two-Tier Client/Server Architecture），系統由程式與資料庫組成。程式以現成軟體元件於視窗編輯器上組裝完成，負責與使用者互動並依據企業規則進行必要的資料處理。資料庫則委託給關連式資料庫管理系統管理，負責儲存軟體系統運作所需要的所有資料〔12〕。
8. 結構化技術比物件導向技術更適合使用在資料中心企業資訊系統〔18〕。而且採取視覺化開發策略的資料中心企業資訊系統可以使用資料流程圖來表示系統範圍、高層次功能、資料儲存，但應該避免過度的功能分解〔14〕。
9. 如果對軟體系統在未來將提供的功能，以及在軟體系統完成之後使用者將會如何使用這些功能（功能的使用情境〔Scenarios〕）等事項予以推導並清楚陳述，則可以幫助客戶與使用者理解與確認尚未開發完成的軟體系統，並能夠促進開發人員對於功能之設計的思考〔6,7,16〕。
10. 結構化技術以第三代程式語言為基礎，所以使用函數的觀點來建立程式，而物件導向技術則以物件導向程式語言為基礎，所以使用物件的觀點來建立程式〔26〕。本文所指的視覺化開發策略則以視覺化程式設計工具為基礎，故而以視窗的觀點來建立程式。依據上述論述可知在視覺化開發策略下，一個軟體系統會有若干個程式，而每個程式負責若干個功能，每個功能則以若干個相互合作的視窗來予以實現。
11. 資料庫的設計過程包含三個步驟：**A.** 概念設計（Conceptual Design）：瞭解資料的儲存需求，然後據以提出理想情況下的資料庫設計方案。**B.** 邏輯設計（Logical Design）：依據所採用資料庫模式的限制，對概念設計的結果進行調整。**C.** 實體設計（Physical Design）：依據效能、成本、安全、整合等方面的考量對邏輯設計的結果再進行調整〔1〕。
12. 在資料庫的模式化方面，視覺化開發專案可以與結構化技術一樣也使用實體關連圖來進行資料庫的模式化。然而實體關連圖對於實體型態（Entity Types）的詳細特徵（例如實體型態的狀態和狀態之間的移轉）與資料庫程序的重要特徵（例如資料庫程序的處理邏輯和複雜的決策規則）等項目並無法予以表示，需要額外使用其它的模式化工具來補強〔14〕。
13. 使用者介面已經成為軟體系統的關鍵重點，然而目前的軟體方法過於不重視使用者介面設計環節〔17〕。在視覺化程式設計下每個功能由一群視窗來實作，所以特定視窗可以當做是功能的一個特殊狀態。據此可以使用狀態移轉圖來做功能的模式化〔24〕。

四、VMVP 軟體模式化技術

VMVP 涵蓋軟體分析與軟體設計這兩個開發活動，並適用於符合下列情況的軟體專案：

1. 是一資料中心企業資訊系統。
2. 採用兩層主從架構。
3. 採取視覺化開發策略。
4. 使用關連式資料庫管理系統。

從建議 4 可知 VMVP 並不需要創造新的模式化工具，而是應該從已經存在的豐富模式化工具中去整理與調整。而且應該注意的是，使用 VMVP 時應該接受建議 5 與建議 6，遵循 UML 之類的符號標準並使用適當的文字說明對軟體模式做補充。

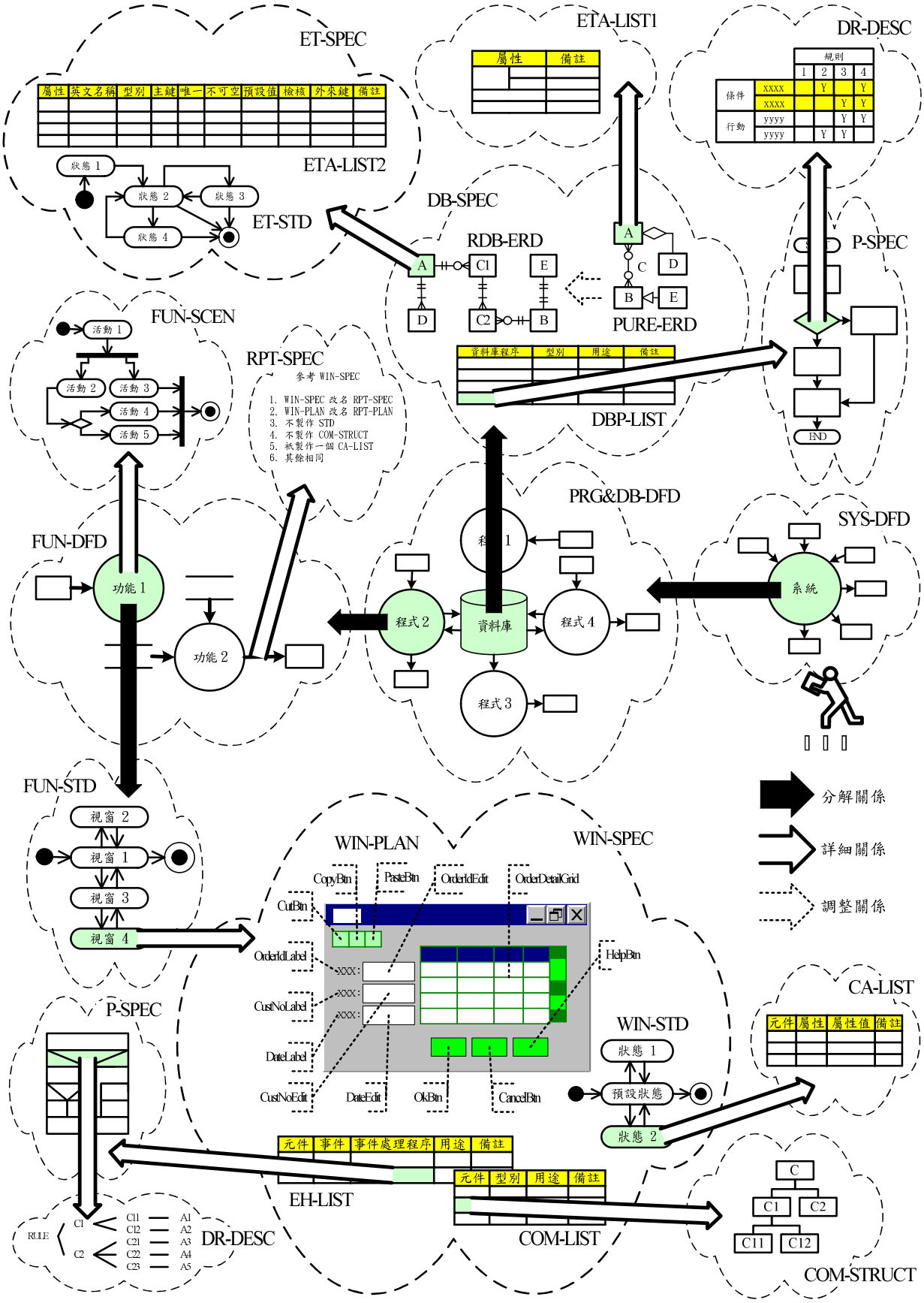
ET-SPEC
屬性 英文名稱 型別 主鍵 唯一 不可空 預設值 檢核 外來鍵 備註
ETA-LIST2
狀態 1
狀態 2
狀態 3
狀態 4
ET-STD
ETA-LIST1
屬性 備註
DR-DESC
規則 1 2 3 4
條件 XXXX Y Y
XXXX Y Y
行動 yyyy Y Y
yyyy Y Y
DB-SPEC
RDB-ERD
A C1 E D C2 B
C D E
PURE-ERD
P-SPEC
END
FUN-SCEN
活動 1
活動 2
活動 3
活動 4
活動 5
RPT-SPEC
參考 WIN-SPEC
1. WIN-SPEC 改名 RPT-SPEC
2. WIN-PLAN 改名 RPT-PLAN
3. 不製作 STD
4. 不製作 COM-STRUCT
5. 祇製作一個 CA-LIST
6. 其餘相同
資料庫程序 型別 用途 備註
DBP-LIST
PRG&DB-DFD
程式 1
程式 2
資料庫
程式 4
程式 3
SYS-DFD
系統
FUN-DFD
功能 1
功能 2
FUN-STD
視窗 2
視窗 1
視窗 3
視窗 4
WIN-PLAN
WIN-SPEC
CopyBtn
PasteBtn
OrderIdEdit
OrderDetailGrid
CutBtn
OrderIdLabel
CustNoLabel
DateLabel
CustNoEdit
DateEdit
OkBtn
CancelBtn
HelpBtn
XXX:
分解關係
詳細關係
調整關係
CA-LIST
元件 屬性 屬性值 備註
WIN-STD
狀態 1
預設狀態
狀態 2
P-SPEC
元件 事件 事件處理程序 用途 備註
元件 型別 用途 備註
EH-LIST
COM-LIST
RULE
C1 C11 C12 A1 A2
C2 C21 C22 C23 A3 A4 A5
DR-DESC
C
C1 C2
C11 C12
COM-STRUCT

圖2 VMVP 的概念圖

表1 VMVP之模式化工具的彙總

類別	工具		來源工具	製作數量	主要用途
分析	SYS-DFD		DFD	1個	顯示系統與外部實體之間的互動
	PRG&DB-DFD		DFD	1個	顯示構成系統的程式與資料庫
	FUN-DFD		DFD	1個程式1個	顯示程式應該實作的功能
	FUN-SCEN		AD	1個功能1個	描述使用者觀點的功能使用情境
程式設計	FUN-STD		STD	1個功能1個	顯示功能的實作視窗
	WIN-SPEC	WIN-PLAN		1個視窗1個 *	顯示視窗的外觀
		COM-LIST		1個視窗1個	表列視窗的所有元件
		COM-STRUCT	TD	1個複雜結構元件1個	顯示複雜結構元件的階層結構（聚合關係）
		WIN-STD	STD	1個複雜視窗1個	顯示視窗的各種狀態
		CA-LIST		1種狀態1個	表列應該設定的屬性
		EH-LIST		1個視窗0或1個	表列應該撰寫的事件處理程序
		P-SPEC	FC,NC,AD,PC	1個複雜程序1個	顯示事件處理程序的處理邏輯
		DR-DESC	DT1,DT2	1個複雜決策規則1個	顯示事件處理程序的決策規則
	RPT-SPEC	RPT-PLAN		1張報表1個	顯示報表的外觀
		COM-LIST		1張報表1個	表列報表的所有元件
		CA-LIST		1張報表1個	表列應該設定的屬性
		EH-LIST		1張報表0或1個	表列應該撰寫的事件處理程序
		P-SPEC	FC,NC,AD,PC	1個複雜程序1個	顯示事件處理程序的處理邏輯
		DR-DESC	DT1,DT2	1個複雜決策規則1個	顯示事件處理程序的決策規則
資料庫設計(DB-SPEC)	PURE-ERD		ERD	1個	顯示資料庫的實體型態與實體型態之間的關係
	ETA-LIST1			1個實體型態1個 #	表列實體型態的所有屬性
	RDB-ERD		ERD	1個	顯示資料庫的實體型態與實體型態之間的關係
	$	ET-STD	STD	1個複雜實體型態1個	顯示實體型態的各種狀態
		ETA-LIST2		1個實體型態1個	表列實體型態的所有屬性
	DBP-LIST			0或1個	表列所有的資料庫程序
	P-SPEC		FC,NC,AD,PC	1個複雜程序1個	顯示資料庫程序的處理邏輯
	DR-DESC		DT1,DT2	1個複雜決策規則1個	顯示資料庫程序的決策規則

DFD=Data Flow Diagram, STD=State Transition Diagram, ERD=Entity Relationship Diagram, FC=Flow Chart, NC=N-S Chart, AD=Activity Diagram, PC=Pseudo Code, DT1=Decision Table, DT2=Decision Tree, TD=Tree Diagram

* 使用分頁元件（Page Control）的多頁面視窗應該增加視窗藍圖的製作數量，務必讓所有的介面元件都能夠顯示出來

\# 擁有屬性的關係予以視同實體型態，也要予以製作屬性清單

$ ET-SPEC

由建議3可知VMVP與其它的軟體模式化技術一樣，必需使用多種的模式化工具來做軟體的模式化。VMVP涵蓋了軟體分析與軟體設計兩個部份，而依據建議7可知軟體設計還可以再細分成程式設計與資料庫設計兩個部份。另從第8,9,10,12,13項建議可知VMVP應該以高層次的資料流程圖和功能的使用情境描述來做為軟體分析活動的模式化工具（建議8,9），以狀態移轉圖來表示出每個功能的構成視窗與這些視窗之間的各種可能移轉（建議10,13），以實體關連圖來表示資料庫的實體型態與實體型態之間的關係（建議12）。

以下將分別就軟體分析、程式設計、資料庫設計等三個部份的模式化工具做進一步的說明（請參考圖1、圖2與表1來幫助瞭解），並在最後彙總性說明VMVP與結構化技術和物件導向技術之間的不同之處。

4.1 軟體分析方面的模式化工具

在軟體分析活動中，開發人員將先瞭解客戶與使用者的需要（Needs，使用文字方式記錄），然後據以開立出軟體系統的需求規格。而軟體需求規格規範著為了滿足客戶與使用者的需要軟體系統應該具備的各種功能性特徵（或稱為功能需求）與非功能性特徵（或稱為限制）。非功能性特徵可以使用文字予以清楚說明，功能性特徵則需要使用以模式化工具製作的功能需求模式才能夠予以清楚說明（請參考建議2）。

建議8與實際開發經驗都顯示適當應用的資料流程圖〔3,10,25〕將是視覺化開發策略下資料中心企業資訊系統很理想的功能需求模式化工具，據此VMVP予以採納。VMVP固定使用三個層次的資料流程圖，而這三者應該平行製作、反覆精鍊，並且相互修正，如此才能夠很容易就予以正確且一致的製作完成。

最高層的環境圖（SYS-DFD）與結構化技術的環境圖（Context Diagram）完全相同，用來顯示軟體系統與外部實體之間的互動。圖中包含了軟體系統（圓形）、資訊需求者（外部實體之一，有箭頭線段指入的矩形）、

資料提供者（外部實體之一，有箭頭線段流出的矩形）、軟體系統產出的資訊（從軟體系統流出的箭頭線段）、軟體系統維護的資料（流入軟體系統的箭頭線段）等符號，用來顯示軟體系統應該提供哪些資訊給誰，而為了產出這些資訊，必需維護哪些資料，這些資料又應該由誰提供。需要特別小心的是，外部實體所指的不盡然全是程式的操作人員，而應該是軟體系統所產出資訊的真正使用人員與軟體系統所維護資料的真正提供者，並不需要考慮其是否直接操作軟體系統（請參考圖 2 的 SYS-DFD 部份）。

第二層的系統架構圖（PRG&DB-DFD）是顯示在環境圖中之軟體系統的進一步功能分解圖，用來顯示軟體系統的架構，亦即用來顯示軟體系統由哪些程式與資料庫組成。有兩個地方與傳統的資料流程圖不太相同。第一個不同處是 VMVP 明顯區分軟體系統與程式，所以會有這個以程式和資料庫為主體的特殊資料流程圖。第二個不同處是 VMVP 引進了資料庫符號（圓柱形），所有與資料庫相關的資料流均流向資料庫或由資料庫流出，藉以避免資料庫的設計方案過早顯示在這種高層次的資料流程圖之中，亦即在系統架構圖之中將不會出現以平行線表示的資料儲存（請參考圖 2 的 PRG&DB-DFD 部份）。

第三層的程式功能圖（FUN-DFD）是系統架構圖中每一個程式的進一步功能分解圖，用以顯示程式應該實作的所有功能。要注意的是，這個層次的資料流程圖雖然已經顯示了資料儲存，但顯示在這裡的目的係在於表示出每個功能的資料儲存需求並不涉及資料庫設計活動，而且有可能會與設計活動所完成的實體關連圖有些地方不一致。當資料庫設計活動完成之後可以回過頭來修改程式功能圖的資料儲存，使兩者之間能夠保持一致（請參考圖 2 的 FUN-DFD 部份）。

另一方面，上述之三個層次的資料程流圖製作完成之後，建議再從使用者的角度來對程式功能圖裡頭的每一個功能予以製作使用情境的描述（FUN-SCEN），藉以清楚顯示程式完成之後使用者在執行特定功能之時，從開始執行到結束執行，使用者與程式之間的一系列互動過程[2]。VMVP 採納 Jacobson 的做法，以文字陳述做為使用情境的主要描述方式〔6〕，然而如果使用情境過於複雜之時則建議額外製作活動圖（Activity Diagram）〔2,5〕來予以輔助（請參考圖 2 的 FUN-SCEN 部份）。

4.2 程式設計方面的模式化工具

在軟體設計活動中，開發人員依據軟體分析活動所完成的軟體需求規格提出符合要求的軟體設計方案（記錄在軟體設計文件之中），藉以指引軟體實作活動的程式撰寫與資料庫建置及輔助維護活動之程式與資料庫的變動。

使用以模式化工具製作的設計模式來對軟體設計方案做表達，有其好處與必要性（請參考建議 2），故而 VMVP 予以採用。而在視覺化開發策略下資料中心企業資訊系統由程式與資料庫構成，因此包含了程式與資料庫這兩個部份的設計。本小節先探討程式設計方面的模式化工具，下一小節則再探討資料庫設計方面的模式化工具。

程式功能圖中的每個功能都將以一個以上的視窗來實作，因此表示出每個功能的所有實作視窗與每個實作視窗的設計內容是程式設計活動的重點。VMVP 採用了多種的程式設計方面的模式化工具，分別說明如后。

對於程式的每一個功能 VMVP 建議都先使用狀態移轉圖〔2,3,5,10,25〕（功能架構圖，FUN-STD）來表示出特定功能的所有實作視窗、這些視窗之間的可能移轉，以及造成這些移轉的事件與因應行動（亦即事件處理程序的執行）（請參考圖 2 的 FUN-STD 部份），然後再使用視窗規格（WIN-SPEC）對每一個實作視窗做詳細的設計說明[3]（請參考圖 2 的 WIN-SPEC 部份）。而視窗規格包含了八個部份：視窗藍圖（WIN-PLAN）、元件清單（COM-LIST）、元件結構圖（COM-STRUCT）、視窗狀態移轉圖（WIN-STD）、元件屬性清單（CA-LIST）、事件處理程序清單（EH-LIST）、程序規格（P-SPEC）、決策規則陳述（DR-DESC）。

視窗藍圖用來顯示視窗的外觀。可以徒手繪製，也可以使用 VISIO 之類的流程圖工具來繪製，當然也可以是螢幕擷取影像（Screen-Shot Image）。然而無論其製作的方式為何，都應該標註視窗所使用之介面元件的名稱（請參考圖 2 的 WIN-PLAN 部份）。

在視覺化程式設計方式之下視窗是使用現成元件組裝出來的，所以必需表示出視窗應該使用的所有元

[2] 一個功能可能會有多個使用情境，而且一個使用情境除了正常的途徑（Path）之外，也可能還會有若干個例外的途徑。製作使用情境描述時應該予以考慮週詳不要有所缺漏。

[3] 除了每個功能的實作視窗之外，完整的程式還必需製作選單（Menu）、啟動（Spalsh）、關於（About）等獨立於功能之外的視窗。這些獨立視窗的設計也需要使用視窗規格來予以清楚說明。

件。VMVP 以元件清單來說明這些元件，而元件清單應該有的欄位如圖 3 所示(請參考圖 2 的 COM-LIST 部份)。

元件	型別	用途	備註

圖3 元件清單應該有的欄位

視窗所使用的某些元件之間可能存在著聚合關係(Aggregation Relationship)，而這種關係大致上可以再區分成整體與部份關係 (Whole-Part Relationship)：某個複合元件由多個零組件元件所構成。容器與內容物關係(Container-Content Relationship)：某個容器元件容納著多個內容元件。而如果這些關係無法在視窗藍圖中呈現，或是無法使用元件清單的備註欄位來予以清楚說明之時，那麼就要替具有複雜結構的元件繪製元件結構圖(採用六標準差的 Tree Diagram〔15〕)，藉以清楚顯示出元件之間的聚合關係(請參考圖 2 的 COM-STRUCT 部份)。

視窗可能是複雜的，擁有一種以上的狀態。而這些狀態之間的不同即在於有著不一樣的元件屬性設定。每個視窗最少會有一種在開發時期就已經決定好了的預設狀態 (Default State)，也可能會有一種以上因為某一個事件與因應行動對元件屬性做了更動而進入的其它狀態 (Other States)。例如一個比較複雜的客戶資料維護視窗也許會有瀏覽(預設狀態)、編輯，與新增等三種狀態。使用者開始使用這個視窗時視窗處於祇可以唯讀資料的瀏覽狀態，而當使用者點選了編輯按鈕後則進入了可以對現有資料做修改的編輯狀態，又如果使用者點選的是新增按鈕則進入的是可以增加一筆新資料的新增狀態，另外當視窗在編輯狀態或是新增狀態時使用者可以點選確定按鈕或是取消按鈕重新回到預設的瀏覽狀態。如果一個視窗擁有了一個以上的狀態，那麼就應該予以製作視窗狀態移轉圖，藉以顯示該視窗的所有可能狀態、這些狀態之間的可能移轉，以及造成這些移轉的事件與因應行動(請參考圖 2 的 WIN-STD 部份)。

視窗所使用元件之屬性的設定必需予以適當的說明。除了說明在開發時期應該做的元件屬性設定之外(預設狀態之元件屬性設定的說明)，還要說明在執行時期應該做的元件屬性設定(各種其它狀態之元件屬性設定的說明)。VMVP 建議使用元件屬性清單來做各種視窗狀態之元件屬性設定的說明，而元件屬性清單應該有的欄位如圖 4 所示(請參考圖 2 的 CA-LIST 部份)。

事件處理程序清單說明應該予以關心與處理的元件事件，亦即說明應該撰寫的事件處理程序，而事件處理程序清單應該有的欄位如圖 5 所示(請參考圖 2 的 EH-LIST 部份)。

元件	屬性	屬性值	備註

圖4 元件屬性清單應該有的欄位

元件	事件	事件處理程序	用途	備註

圖5 事件處理程序清單應該有的欄位

程序規格的用途類似結構化技術所使用的處理規格 (Process Specification)，其描述著特定程序(事件處理程序或是資料庫程序)的處理邏輯，而事件處理程序清單上的每一個複雜的事件處理程序都必需製作一個程序規格。VMVP 建議採用活動圖、N-S 圖 (N-S Diagram)〔10〕、虛擬碼 (Pseudo Code)〔3,10〕、流程圖 (Flow Chart)〔3,10〕等工具來表達複雜程序的處理邏輯(請參考圖 2 的 P-SPEC 部份)。

如果程序規格裡頭有複雜的決策規則，那麼必需再製作決策規則陳述。VMVP 建議採用決策樹 (Decision Tree)或是決策表 (Decision Table)〔3,10〕來表達複雜的決策規則(請參考圖 2 的 DR-DESC 部份)。

另一方面，雖然大多數的資訊都可以使用視窗在螢幕上顯示，然而紙張性資訊輸出(亦即報表)對於現今的資料中心企業資訊系統而言仍然有其不可取代性。而使用視覺化程式設計工具來製作一張報表的方式與製作一個視窗的方式大致上相同，因此每張報表都應該使用與視窗相同的模式化工具。不同的五個地方是：**1.** 視窗規格改名成報表規格 (RPT-SPEC)。**2.** 視窗藍圖改名成報表藍圖 (RPT-PLAN)。**3.** 報表固定為單一種狀態並不需要製作報表狀態移轉圖。**4.** 報表不需要製作元件結構圖。**5.** 報表固定祇製作一個元件屬性清單(請參考圖 2 的 RPT-SPEC 部份)。

4.3 資料庫設計方面的模式化工具

雖然不同軟體系統的資料庫之間必需予以整合，而且帳號管理與安全防護等也都是資料庫相關的重要事項，但是在開發階段中祇需要考慮理想情況下單一軟體系統的資料儲存需求，與開發無關的事項不需予以理會。據此可知開發階段中祇涉及了概念設計與邏輯設計這兩項資料庫設計步驟(請參考建議 11)。

在資料庫設計活動中，開發人員主要以系統架構圖上面所顯示出來的資料儲存需求來做資料庫的設計(思考圖中資料庫的流入與流出資料)，而 VMVP 建議使用資料庫規格 (DB-SPEC)來詳細說明開發人員所提

出來的資料庫設計方案（請參考圖 2 的 DB-SPEC 部份）。資料庫規格包含七個部份：原始的實體關連圖（PURE-ERD）、調整後的實體關連圖（RDB-ERD）、資料庫程序清單（DBP- LIST）、第一種實體型態屬性清單（ETA-LIST1）、第二種實體型態屬性清單（ETA-LIST2）、程序規格、決策規則陳述。

原始實體關連圖用來顯示理想情況下一個最能反應真實資料儲存需求的資料庫設計方案（請參考圖 2 的 PURE-ERD 部份）。調整後實體關連圖則用來顯示考量關連式資料庫的各種限制而修訂完成的資料庫設計方案[4]（請參考圖 2 的 RDB-ERD 部份）。調整後實體關連圖裡頭的每一個實體型態都將映射成一個關連式資料庫的資料表格。

如果在實體關連圖中顯示屬性將會讓實體關連圖變得很雜亂很不容易閱讀，故而 VMVP 不建議在實體關連圖中顯示任何的屬性，所有的屬性都建議使用實體型態屬性清單來說明。而適用於原始實體關連圖的第一種實體型態屬性清單應該有的欄位如圖6所示（請參考圖 2 的 ETA-LIST1 部份），適用於調整後實體關連圖的第二種實體型態屬性清單應該有的欄位則如圖 7 所示（請參考圖 2 的 ETA-LIST2 部份）。

屬性	備註

圖6 第一種實體型態屬性清單應該有的欄位

屬性	英名文稱	型別	主鍵	唯一	不可空	預設值	檢核	外來鍵	備註

圖7 第二種實體型態屬性清單應該有的欄位

VMVP 也建議對調整後實體關連圖裡頭擁有多種狀態的複雜實體型態[5]額外製作一個狀態移轉圖，藉以讓實體型態的各種狀態與狀態之間的移轉能夠比較清楚地顯示出來（請參考圖 2 的 ET-STD 部份）。

除了資料之外，現代的關連式資料庫管理系統也支援了預存程序（Stored-Procedures）、驅動（Triggers）、使用者自訂函數（User-Defined Functions）等儲存在資料庫並由資料庫管理系統執行的資料庫程序，需要對應該撰寫的資料庫程序進行說明。VMVP 建議採用資料庫程序清單來說明應該撰寫的資料庫程序，而資料庫程序清單應該有的欄位如圖 8 所示（請參考圖 2 的 DBP-LIST 部份）。另外資料庫程序之處理邏輯與決策規則的說明則使用和事件處理程序相同的方式，以程序規格（請參考圖 2 的 P-SPEC 部份）和決策規則陳述來說明（請參考圖 2 的 DR-DESC 部份），請參考前一小節關於事件處理程序的說明。

資料庫程序	型別	用途	備註

圖8 資料庫程序清單應該有的欄位

4.4 與結構化技術和物件導向技術的不同之處

相較於結構化技術 VMVP 有下列數個不同之處（以 Yourdon 方法〔25〕為比較的標的）：

1. VMVP 祇使用三個高層次的資料流程圖，這不同於結構化技術的不斷分解到都是最基本處理（Primitive Process）才停止。並且在第二層資料流程圖（系統架構圖）上引進了資料庫符號，增加資料流程圖的表達能力。另外 VMVP 也不製作最基本處理的處理規格。
2. 雖然 VMVP 也使用資料字典（Data Dictionary）的符號以 BNF（Backus-Naur Form）的方式來表示資訊流與資料流的內涵。但並不要求使用者編製細密詳微的集中式資料字典，使用者可以直接在資料流程圖上標註，也可以在軟體需求規格中使用補充文字來說明。
3. VMVP 在分析活動之中不使用狀態移轉圖（系統依時行為的模式化）與實體關連圖（資料庫結構的模式化）。在設計活動之中也不使用結構圖（程序導向之系統結構的模式化）。
4. VMVP 在分析活動之中額外引入了程式功能的使用情境描述，藉以幫助客戶與使用者理解與確認尚未完成的軟體系統。

另一方面，相較於物件導向技術 VMVP 有下列數個不同之處（以 Rational 方法[6]〔7〕為比較的標的）：

1. 程式功能圖這種高層次的資料流程圖在功用上類似物件導向技術的使用案例圖[7]。但是程式功能圖

[4] 擁有屬性的關係轉化成實體型態。繼承、包含、多對多結合等關係都轉化成一對一或是一對多的結合。所有的實體型態都正規化成 Boyce/Codd 型式。

[5] 以某一班級圖書館的藏書為例，其有七種可能的狀態：編目中、展示中、暫停流通、可借閱、已被預約、已借出、已遺失。

[6] 最新版本的 Rational 方法又稱為 RUP，其標榜是一個可以被客製化的流程框架（Process Framework），所有軟體專案的適用流程都可以從 RUP 中衍生出來。然而本質是物件導向的 RUP，其衍生出來的也會是物件導向的流程，但是並不盡然所有的軟體專案都適合使用物件導向流程，故而可知 RUP 仍然有其在適用性上的侷限。

[7] 儘管物件導向技術的擁護者極力辯證使用案例（Use Case）並不等同於功能。可是從定義〔2,5,6,7〕來看，

關心的是資訊的真正需求者與資料的最終提供者，而使用案例圖重視的是與程式直接互動的操作者（物件導向技術稱之為行為者，Actors）。另外程式功能圖沒有功能間之包含(Include)與使用(Use)等關係的概念，但程式功能圖多了功能與外部實體之間之資訊流動與資料流動的表示。

2. 資訊與資料是資料中心企業資訊系統的重心，大多數的程式功能都應該是從這兩者所衍生出來的。據此 VMVP 建議先做外部實體、資訊需求、資料需求、系統架構等方面的模式化（使用環境圖與系統架構圖），然後才做程式功能的模式化（使用程式功能圖與使用情境描述）。這不相同於物件導向技術之直接產出使用案例圖的做法。
3. 視覺化開發軟體專案並不執行類別的識別和實作，所以 VMVP 在設計活動之中不予以製作類別圖等物件導向技術的核心設計模式。

五、結論

視覺化開發已經成為一種重要的軟體開發策略，但其在軟體工程技術層面上的各種問題並沒有受到應該有的重視。本文作者嘗試在這一方面做一些努力。

本文作者彙整了文獻上零散的建議並結合了多年的軟體開發實務經驗，從豐富的現有模式化工具中去尋找與整理，建議了一個適合使用在視覺化開發軟體專案的軟體模式化技術－VMVP。

VMVP 在分析活動之中結合賦予新意義與用法的高層次資料流程圖以及程式功能的使用情境描述來做功能需求的模式化。在程式相關設計方面則採用人機介面技術所建議的狀態移轉圖來表示出每個程式功能的實作視窗，並使用視窗規格來顯示每個視窗的詳細設計內容。在資料庫設計方面則延用實體關連圖並輔以實體型態規格等額外工具的補充。

VMVP 曾經請十四位業界專家[8]予以審視過，並依據這些專家的建議做了部份修訂。另外 VMVP 已經在某些大學資管系和資訊類公共職訓班級[9]的軟體工程或系統分析設計課程上教授，也已經使用在這些單位的某些學生軟體開發專題上。在未來本文作者將對 VMVP 進行科學性實證，並將以實證的結果對 VMVP 進行改善。

被物件導向技術用來模式化軟體之功能性需求的使用案例圖，它的使用案例就是一個功能，是一個比較高層次比較大的功能。

[8] 服務於鼎新電腦、台中商銀、中國石油、電信研究所等機構。

[9] 僑光學院資管系、南亞學院資管系、中區職訓中心電子商務班等單位。

參考文獻

1. Batini, C., Ceri, S. and Navathe, S.B., Conceptual Database Design: An Entity-Relationship Approach, Benjamin/Cummings, 1992.
2. Booch, G., Rumbaugh, J. and Jacobson, I., The Unified Modeling Language User Guide, Addison-Wesley, 1999.
3. Budgen, D., Software Design, Addison-Wesley, 1994.
4. Dukovic, J.M. and Joyce, D.T., "An Evaluation of Object-Based Programming with Visual Basic", Proceedings of the IEEE 14th Annual International Phoenix Conference, 1995, pp. 346-351.
5. Fowler, M. and Scott K., UML Distilled: Applying the Standard Object Modeling Language, Addison-Wesley, 1999.
6. Jacobson, I. et al., Object-Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley, 1995.
7. Jacobson, I., Booch, G., and Rumbaugh, J., The Unified Software Development Process, Addison-Wesley, 1999.
8. Llewellyn, E.L.., Stanton, M. and Roberts, G., "Nine-Step Approach to Designing Successful Visual Programming Applications", Computing and Control Engineering Journal, April 2002, pp. 82-86.
9. Maxim, M., "Microsoft Windows Programming Strategies", Crossroad, Vol. 6, No. 4, 2000, pp. 19-21.
10. Martin, J. and Mcclure, C., Structured Techniques: The Basis for CASE, Prentice-Hall, 1988.
11. Orenstein, D., "Java, Visual Basic seen as languages of future", Computerworld, Vol. 33, No. 13, 1999, pp. 6.
12. Orfali, R., Harkey, D. and Edwards, J., The Essential Client/Server Survival Guide, Wiley, 1999.
13. Philip, G.C., "Software Design Guidelines for Event-Driven Programming", The Journal of Systems and Software, Vol. 41, No. 6, 1998, pp. 79-91.
14. Pressman, R.S., Software Engineering: A Practitioner's Approach, McGraw-Hill, 2001.
15. Pyzdek, T., The Six Sigma Handbook: A Complete Guide for Greenbelts, Blackbelts and Managers at All Levels, McGraw-Hill, 2001.
16. Rumbaugh, J. et al., Object-Oriented Modeling and Design, Prentice-Hall, 1991.
17. Shneiderman, B., Designing the User Interface, Addison-Wesley, 1998.
18. Sommerville, I., Software Engineering, Addison-Wesley, 2001.
19. Tan, H.B.K. and Ling, T.W., "Components Reuse for Data-Intensive Business Programs through an Object-Oriented Architecture", The Journal of Systems and Software, Vol. 34, No. 1, 1996, pp. 3-20.
20. Thayer, R.H. and Dorfman, M., Tutorial: System and Software Requirements Engineering, IEEE, 1990.
21. van Slooten, K. and Schoonhoven, B., "Contingent Information Systems Development", Journal of Systems and Software, Vol. 33, No. 2, 1996, pp. 153-161.
22. Wallnau, K.C., Hissam, S.A. and Seacord, R.C., Building Systems from Commercial Components, Addison-Wesley, 2002.
23. Weinschenk, S., Jamar, P., and Yeo, S.C., GUI Design Essentials, Wiley, 1997.
24. Wiegers, K.E., Software Requirements, Microsoft, 1999.
25. Yourdon, E., Modern Structured Analysis, Prentice-Hall, 1989.
26. Yourdon, E., Decline & Fall of the American Programmer, Prentice-Hall, 1993.