在系統分析與設計階段導入設計樣式之研究

Adopting Design Pattern into System Analysis and Design

莊樹青 葉道明 胡正國

國立屏東科技大學資訊管理研究所 國立屏東科技大學資訊管理研究所 國立屏東科技大學資訊管理研究所

屏東縣內埔鄉學府路1號 屏東縣內埔鄉學府路1號 屏東縣內埔鄉學府路1號

Shuching@vip.url.com.tw dmyeh@mail.npust.edu.tw hujego@tpts4.seed.net.tw

摘要

設計樣式近年來在軟體工程研究領域日受重視。使用較嚴謹的樣式語言將一完整且良好的設計經驗封裝為設計樣式，有助於軟體設計理念之溝通與再使用，降低軟體設計與維護的成本。由於設計樣式著重於細部與局部之設計元件間互動與繼承關係，正好彌補物件導向設計方法論於此方面之不足。本研究針對此問題，開發一套輔助工具，利用設計樣式所衍生出功能相同的準樣式來搜尋系統設計圖，使系統分析師學習設計樣式的時間縮短，又能獲得設計樣式的種種好處，進而達成改善原有系統設計的目標。

關鍵詞彙：設計樣式、物件導向分析與設計、輔助工具、軟體再用、準樣式。

1. 簡介

設計樣式(Design Pattern)是近年來在物件導向分析設計上相當熱門的課題，其主要目的是憑藉著以往程式設計師累積的經驗，提供一些常用或簡潔的解法，來解決一再發生的問題。一旦熟悉這類的解法之後，就可以很快將這些方法應用在系統的設計上，而不用另外花時間構思解決之道，因此在開發複雜的系統或解決龐大的問題時，設計樣式的使用不但可以強化模組的結構，還可以降低系統分析設計所付出的時間和成本。除此之外，設計樣式還可以成為軟體開發人員間一種共通簡單的表達方式，減少彼此溝通時產生的問題[15]。但由於設計樣式的種類繁多，每隔一段時間就有新的設計樣式問世。因此在進行系統分析與設計時，在無法瞭解所有設計樣式詳細功能的情況下，可能會使用較繁複的設計方法，或誤用某些不好的設計樣式(Anti-Pattern)[16]。為了解決這個問題，本研究將開發一個輔助工具，在系統進行分析設計時，藉由物件間的關係，找出可能含有設計樣式的物件組合，並提出加入該設計樣式的建議，使系統分析設計的時間所需的時間及成本減少。但是本研究製作的系統僅為一決策支援系統，並不會直接以標準設計樣式取代準樣式，而關於以標準設計樣式取代準樣式之後所產生的邊際效應(Side Effect)，以及衍生的修改問題，都不在本研究的討論範圍之內。此外，因為設計樣式種類繁多，而且不斷有新的設計樣式問世，而且每個設計樣式都有其功能及特徵，不一定都能從外觀上看出其特性，因此本研究只選定數個設計樣式當做樣本進行系統軟體的開發。

2. 文獻探討

2.1 物件導向分析與設計方法論

物件導向分析與設計的方法論幾乎和物件導向技術同時誕生，但剛開始和物件導向技術一樣並未普遍流行。一直到80年代末期，在C++和Smalltalk等物件導向程式語言發展成熟後，使程式設計師可以使用物件導向技術來撰寫程式時，物件導向分析與設計的方法論才開始被廣泛的討論，於是90年代初期，便產生了一些較為流行的物件導向分析與設計方法論，包括Grady Booch所提出的物件導向開發方法[9]、James Rumbaugh所提出的OMT (Object Modeling Technique)[13]、Ivar Jacobson所提出的OOSE/Objectory[11]、D. Coleman所提出的Fusion Method[4]，以及由Grady Booch、James

Rumbaugh、Ivar Jacobson 所連合制定的 UML(Unified Modeling Language)[10]等等。從這些物件導向分析與設計的方法論演進，可以看出三個共同點和趨勢：

1. 使用圖形取代大量的文字說明。
2. 透過使用者的需求來建構系統，而不是由系統開發者的觀點去定義系統的需求。
3. 從不同的觀點來分析系統，讓使用者、程式設計師、系統開發人員和測試人員都各司其職又能互相溝通。

2.2 設計樣式

設計樣式的主要精神在於設計方法的再使用，最早被廣泛應用在建築設計上。一位建築師 Christopher Alexander 對樣式(Pattern)下了一個定義：「每個樣式都描述了一個重複發生的問題，以及這個問題的解決方法。因此我們可以無限制的使用這個解法，而不需要再去思考這個問題的解決方法」[1]。

樣式的觀念應用到軟體工程領域是在 80 年代末期，Kent Beck 為了設計 Smalltalk 的圖形使用者介面所產生的樣式[14]，同一時期的 James Coplien 則在 C++ 上發展了一些樣式，並命名為 idiom[12]，之後 Eric Gamma 則在其關於物件導向軟體開發的論文中，確認類似觀念的重要性[5]。到了 1991 年，OOPLSA 開始了有關於樣式的討論，這些相關的討論產生了第一版設計樣式類別的草稿[6]。兩年後，這四位學者再以此為基礎，寫下了第一本深入探討設計樣式的專業書籍[7]。之後有相當多的學者們不斷開發出新的設計樣式，或針對單一設計樣式加以探討。簡單來說，設計樣式就是程式設計師在解決特定問題上經驗的累積，也可以視為是另一種軟體再利用的方法。

一個設計樣式有四個要素，分別描述如下：

1. 名稱(Name)：樣式名稱是一個我們用來描述問題、解決問題的方法和結果的字彙。
2. 問題(Problem)：問題就是描述使用該設計樣式的時機，主要是解釋問題和其內容。
3. 解法(Solution)：設計樣式讀解法則是描述設計的構成、要素之相互關係、相互職責和共同合作等。
4. 結果(Consequences)：結果則是應用此種設計樣式所要付出的代價和得到的結果。

設計樣式的分類依照 Design Pattern[7]書中所描述的標準，可依各個設計樣式的用途分為三類，分別為創造類(Creational)、結構類(Structural)及行為類(Behavioral)三種[17]。

設計樣式的優點很多，因為設計樣式本身是設計方式的再利用，所以一旦熟悉這類的解法之後，就可以很快將這些方法應用在系統的設計上，不需要另外花時間去構思解決之道。設計樣式還可以成為軟體開發人員間一種共通簡單的表達方式，減少彼此溝通時產生的問題[15]。此外，根據實驗證明，設計樣式的文件在特定的工作與程式之中，確實可以減少工作所需的時間與增加解法上的品質，而且可以增加系統維護上的效率[14]。

3. 準樣式與系統架構

3.1 準樣式

準樣式(Semi-patterns)[3]是指和其對應的標準設計樣式所欲解決的問題相同，但其解決方案不同的樣式，這些解決方式對於該問題而言大多是較差的方法，或者是由分析設計經驗較不足的系統分析師所設計出來的，因此也可以將準樣式視為一個尚未完成的樣式。

因為準樣式在解決問題的方式上和標準樣式有所差異，所以準樣式的設計圖和標準設計樣式的設計圖便略有不同。準樣式的設計圖可能很接近標準樣式，也可能只具其對應的標準樣式其中一兩項特徵。而因為準樣式與標準樣式的功能相同，因此我們可以在已完成的系統分析設計圖中尋找準樣式，進而建議將準樣式的設計方法改為標準樣式的設計方式。

理論上一個標準樣式可以對應出相當數個乃至數十個的準樣式，光是考慮 Design Patterns 書中所有標準樣式所衍生的準樣式，其數量就已經相當可觀了。因此，本研究只挑選結構類的樣式當做本研究開發工具的測試樣本，因為結構類的設計樣式較容易由樣式的類別、物件、與關係式中存在的特性來加以定義其準樣式。此外，因為每種設計樣式都有其特性，不一定單靠靜態的類別圖就能看出其特性，可能需要搭配動態的循序圖或是合作圖才能了解該設計樣式的運作方式。而因為結構類的設計樣式主要是表達物件與物件之間的關係，只需要靜態的類別圖便可表達出其設計理念，所以本研究只從結

構類的設計樣式中選擇 Adapter、Bridge、Composite、Decorator、Flyweight 當做樣本進行研究。

Design Patterns 書中除了定義設計樣式以外，還有指出部份準樣式的形態，例如其中有一個關於橋樑(Bridge)樣式的範例，其樣式的原始結構便是巢狀繼承關係(nested generalization)。此樣式的功能是利用繼承關係，將數個不同功能的類別綁在一個抽象類別(abstract class)上，而此抽象類別則定義出一個共同的介面。橋樑樣式主要是提供其層級以下所屬類別一個共同的介面，但是每一個抽象類別所延伸出去的類別都需要定義出不同的適應規格，一旦類別的數量激增，整個設計便會變得混亂不堪，如圖 1 所示。

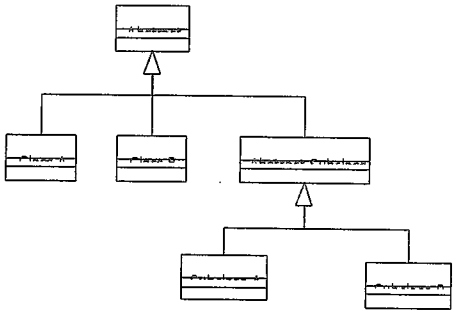


圖 1 Bridge 的準樣式圖

在一些資歷較淺或是經驗不足的系統分析師的系統分析設計中，經常可以發現準樣式的存在，會造成這種原因是因為他們只了解部份的物件導向設計技術，但還無法善用完整優良的設計技巧，特別是抽象化物件的設計概念。以[7]中的 Composite 樣式為例，一種準樣式設計方式為階層式設計，如圖 2 所示。在設計圖的圖形繪製上，這種階層式的準樣式圖形並沒有辦法徹底的描繪出來，因為這種圖形理論上可以有無限深的層級。要解決這種設計方式的準樣式，只能從準樣式中遞迴的找尋出最基本的兩層，就像圖 2 中所列出的部份。

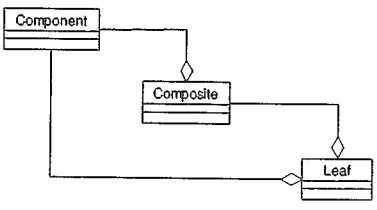


圖 2 階層式的 Composite 準樣式圖

3.2 系統架構

本研究所提出的系統架構圖如圖 3 所示，輸入的部份是系統分析圖，其用途可分為資料庫所需的標準樣式以及準樣式。所需建構的系統部份為其中虛線框起來的範圍。除了上一小節所提的設計樣式資料庫以外，系統中還包含了三個重要的模組，分別是格式轉換模組、樣式搜尋模組，以及樣式提示模組。格式轉換模組主要的工作是輸入 Rose 繪製的圖形，轉換成系統內部所使用的編碼法；樣式搜尋模組是依據設計樣式資料庫中所定義準樣式，以及準樣式的特徵條件去搜尋；樣式提示模組則是依照樣式搜尋模組所搜尋的結果，將準樣式對應系統設計圖中的類別名稱輸出給系統設計人員參考。這三個模組的運作方式及流程於下一節說明之。

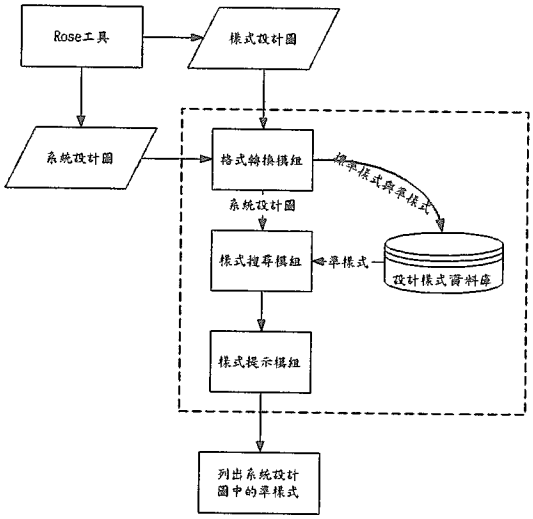


圖 3 系統架構圖

設計樣式資料庫是將本研究所使用到的標準樣式及準樣式加以建檔，以供後續的搜尋以及查詢所需。本研究所有的系統分析設計圖形都使用 UML 的當做標準，而資料庫中所使用到的設計圖，在減少儲存的空間及縮短後續設計圖分析時間的考量下，本研究將物件圖中的各個元素，以及其對應的關係加以編碼處理之後放入資料庫中。因此，本研究中的設計樣式資料庫可分為兩大部份：標準樣式的詳細說明，包括該設計樣式的名稱、問題、解法、及結果等。另一部份則是準樣式的相關說明，包括對應的標準樣式名稱、準樣式的圖形編碼、以及特徵條件等。

在 UML 的類別圖中，常見的模型元素有五種，分別是 Class、Aggregation、Association、Dependency、

Generalization，其對應的編碼如表 1 所示。

以[7]的 Composite 準樣式為例，經過轉換後的編碼如表 2 所示。而隨著不同的設計方式與習慣，會產生出不同的類別編號方式，例如表 2 中灰色部份便是另一種編碼結果。

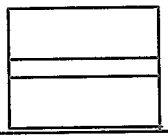
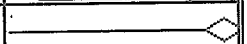
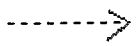
元素名稱	編碼	對應圖形
Class	n(n 為一整數)	
Aggregation	a	
Association	b	
Dependency	c	
Generalization	d	

表1 類別圖中的模型原素及其編碼

類別名稱	Client	Leaf	Composite
類別編碼	1	2	3
關係編碼	1b2, 1b3	2a3	3a3
類別名稱	Client	Leaf	Composite
類別編碼	3	2	1
關係編碼	3b2, 3b1	2a1	1a1

表2 Composite 的準樣式編碼

4. 系統分析與實作

本系統採用 Microsoft Visual Basic 6.0 當作前端開發工具，連結 Microsoft Access 97 當作後端的資料庫，在物件導向分析設計的開發工具則是使用 Rational Rose 4.0。

4.1 格式轉換模組

格式轉換模組主要的功能就是將 Rose 繪製好的圖形檔案加以剖析、轉換成樣式資料庫所使用的編碼，任何輸入系統的圖形，包括標準樣式、準樣式、分析設計完成的圖形都要經過這一個步驟，轉換成系統內其它模組所能辨認的編碼方式。此外，如果輸入的是標準樣式或準樣式，經過轉換之後將被存入設計樣式資料庫中，而分析設計完成的類別圖將送往樣式搜尋模組中找尋可能存在的模組。

格式轉換模組的執行步驟如下：

1. 開啟 Rational Rose 檔案。
2. 搜尋類別位置，付予編號並加以記錄。
3. 搜尋關係及串接的兩個類別編號。
4. 圖形分析完以後，由使用者定義圖形用途：標準樣式、準樣式、欲比對的目標圖形。

以 Composite 樣式為例，經過轉換後的編碼為「1b2 2a3 2b3 2b4」，不過這並不是唯一的編碼方式，因為隨著不同的設計方式與習慣，會處生出不同的類別編號方式，在格式轉換模組中並不針對此狀況加以處理，而只是將圖形依特定的形式轉換成其它各模組所需的編碼。

4.2 樣式搜尋模組

樣式搜尋模組主要功能是在分析設計完成的類別圖中，找出準樣式的位置，是整個系統中最複雜的部份。本研究捨棄從圖形的觀點來解決搜尋上的問題，而將圖形中的每個元素給予適當編碼處理，將整個問題簡化為字串比對，並以此為基礎設計出直接搜尋法和特徵搜尋法等兩種搜尋演算法。

直接搜尋法是以搜尋樣式為基準，針對搜尋的目標做徹底的比對，直到所有可能的情況都比對完成為止，主要的步驟如下：

1. 將資料庫中取出的準樣式和輸入欲比對的設計圖轉換成兩個字串。
2. 進行字串比對，檢查準樣式是否存在。
3. 排列準樣式的類別值，重複第二步驟，直到所有的類別值排列結束為止。如果準樣式的個數為 m 個時，則其排列次數為 $m!$。
4. 變更準樣式的類別值，重複第二步驟，直到所有的類別值變更結束為止。變更類別值主要是以準樣式的類別值為主，其規則如下：

(1.) 將準樣式的類別值由小至大排列。

(2.) 每次變更類別值時，便將最大的類別值加 1，一直加到與欲比對的設計圖類別最大值相同為止。

(3.) 將準樣式中次大的類別值加 1，最大的類別值為次大值加 1，然後回上一個累加步驟，直到無法再增加為止。

(4.) 依上一個步驟遞增下一個類別值，一直反覆到準樣式中所有的類別值都無法增加為止。假如準樣式的

類別個數為 m 個，欲比對圖形的類別個數為 n 個，則其變更次數為 C_{n-m}^{n}。

直接搜尋法可以達到完全比對，但缺點就是太浪費時間，假設樣式資料庫中總共有 l 個準樣式，平均每個準樣式的類別個數有 m 個，輸入的比對圖形有 n 個類別，總計比對次數高達為 $l \times m! \times C_{n-m}^{n}$。如此多的比對次數，還不考慮到兩個字串間的比對程序，可以想見隨著樣式資料庫的加大，搜尋的時間也將隨之大幅上升。為了縮短搜尋所需的時間，本研究針對每一個準樣式提出至少一個對應的「特徵條件」，並以此設計出特徵搜尋法。

特徵條件是指構成某個標準樣式或準樣式中不可或缺的一個或多個關係。例如在 Composite 樣式中，Component 和 Composite 兩個類別之間的 Aggregation 關係便是一個不可或缺的關係式，可視為該樣式的特徵條件。準樣式中的特徵條件可能不止一個，因此在特徵條件建檔時，須將所有的特徵條件加以建檔，以便於依所有的特徵條件加以搜尋準樣式的位置。

特徵條件搜尋法的執行步驟如下：

1. 利用準樣式的特徵條件搜尋欲比對的設計圖。
2. 如果找到與特徵條件相同的關係式，則將搜尋到的關係式和相鄰的類別關係分離成另一個小圖形。
3. 如果特徵條件為具方向性之關係，包括 Aggregation、Dependency、和 Generalization，則將特徵條件與萃取的圖形中相同關係的類別值鎖定，再依照類別值與類別間的關係循序比對剩餘的部份。
4. 如果特徵條件並不具備方向性(Association)，則將欲比對的圖形改成步驟二所分離出的小圖形，代入步驟一中執行，直到所有的特徵條件比對完畢。
5. 如果第二步驟和第四步驟所分離出的結果小於準樣式大小或為空集合，則換下一個準樣式。如果產生的圖形大於準樣式，則將分離出的小圖形，利用直接搜尋法搜尋。

由上述執行步驟可知，特徵搜尋法的運作方式有兩種，一是將欲比對目標圖形中的類別與關係數量減少，達到簡化圖形的目的，然後再將萃取後的圖形代入直接搜尋法之中，以藉此減少比對的次數；二是利用找到的特徵條件為基準加以擴散搜尋範圍，迅速並直接找出是否存在準樣式。

在一般情況下，特徵搜尋法雖然可以比直接搜尋法減少較多的比對次數，但是如果目標圖形中存在太多與準樣式相同的特徵條件，可能會造成因萃取過多重複的小圖，導致總比對次數高於直接搜尋法，尤其是當準樣式中並沒有存在具方向性特徵條件時。例如準樣式形式為 1b2 2b3 3b4，其主特徵條件為 1b2，欲比對的圖形為 1a2 2b3 3b4 4b5 5b6 6b7，直接搜尋法須執行 $4!*C_3^7$=840 次字串比對動作。

而在同一個例子中，特徵搜尋法則會萃取出 5 組小圖形，如表 3 所示，代入直接搜尋法之後須執行 984 次字串比對。假如將準樣式中每一項關係都當成特徵條件加以搜尋，則其萃取的過程如表 4 所示，總比對次數降至 240 次。若是取準樣式中的 2b3 為主特徵條件，不但萃取出的小圖形減為 4 個，而且字串比對次數大幅縮減至 96 次，如表 5 所示。

從上述的例子中可以清楚的看出圖形萃取不當所產生的後遺症，要避免這種情況發生的方式有三種，一是儘量以具方向性的關係當作特徵條件，因為不具方向性的特徵條件並無法直接鎖定目標圖形的類別值，因此沒辦法直接依準樣式的關係與類別值直接擴散範圍來搜尋。二是在準樣式的形式允許的情況下，儘可能的多設數個特徵條件以達到縮小比對圖形的目的，甚至整個準樣式中的所有關係都是特徵條件也可以。三是如果準樣式中所有的關係都沒有方向性，那就儘量選擇準樣式圖形的中心關係為第一特徵條件，如此可讓特徵搜尋法在做第一次搜尋時，不會取出範圍太大的圖形。

搜尋結果	類別個數	字串比對次數
1a2 2b3 3b4 4b5	5	$4!*C_1^5$=120
1a2 2b3 3b4 4b5 5b6	6	$4!*C_2^6$=360
2b3 3b4 4b5 5b6 6b7	6	$4!*C_2^6$=360
3b4 4b5 5b6 6b7	5	$4!*C_1^5$=120
4b5 5b6 6b7	4	4!=24
		總計 984 次

表 3 特徵搜尋法範例之一

搜尋結果			類別個數	字串比對次數
特徵 1b2	特徵 2b3	特徵 3b4		
1a2 2b3 3b4 4b5	2b3 3b4 4b5	2b3 3b4 4b5	4	24
1a2 2b3 3b4 4b5 5b6	1a2 2b3 3b4	1a2 2b3 3b4	4	24
	2b3 3b4 4b5	2b3 3b4 4b5	4	24
	3b4 4b5 5b6	3b4 4b5 5b6	4	24
2b3 3b4 4b5 5b6 6b7	2b3 3b4 4b5	2b3 3b4 4b5	4	24
	3b4 4b5 5b6	3b4 4b5 5b6	4	24
	4b5 5b6 6b7	4b5 5b6 6b7	4	24
3b4 4b5 5b6 6b7	3b4 4b5 5b6	3b4 4b5 5b6	4	24
	4b5 5b6 6b7	4b5 5b6 6b7	4	24
4b5 5b6 6b7	4b5 5b6 6b7	4b5 5b6 6b7	4	24
				總計 240 次

表 4 特徵搜尋法範例之二

搜尋結果	類別個數	字串比對次數
1a2 2b3 3b4	4	24
2b3 3b4 4b5	4	24
3b4 4b5 5b6	4	24
4b5 5b6 6b7	4	24
		總計 96 次

表 5 特徵搜尋法範例之三

4.3 樣式提示模組

樣式提示模組的主要功能是依照樣式搜尋模組所搜尋的結果，列出比對目標圖形中所包含的準樣式名稱及位置，但是樣式提示模組並不會將準樣式所定應的標準樣式插入原本設計圖中，考量的因素有三：

1. 設計樣式中所使用的類別名稱主要是便於解釋該樣式的功能，不一定適用於每一個不同的系統設計。
2. 因為準樣式和標準樣式的類別和關係個數不盡相同，假設在已完成的類別圖中插入標準樣式，如何將插入圖形部份以外的類別與關係做完整的串接會是一個相當複雜的問題。
3. UML 的所有設計圖並不止有類別圖，而在整體設計規劃時可能會有一部份資訊是屬於數個圖形所共有的，雖然這部份可以在圖形轉換時加以萃取，但是插入標準樣式之後會對其它圖形造成何種影響並無法預估。

基於上面的考量因素，所以本研究中的樣式提示模組只輸出準樣式的位置給系統分析人員參考。而為了讓系統分析人員可以直接針對搜尋結果加以改進，樣式提示模組另外提供標準樣式的查詢功能。樣式提示模組的執行步驟如下：

1. 讀入圖形搜尋模組所產生的結果。
2. 列出準樣式在目標圖形中的位置，包括目標圖形中對應準樣式的類別名稱，以及準樣式所屬標準樣式的名稱等資訊，同時提供直接查詢搜尋到的準樣式對應的標準樣式詳細資料之功能。

5. 實例測試

本節主要描述系統實作的成果，主要分為兩個部份，第一部份是測試系統是否能正確無誤的找出準樣式的位置，以一個較為簡單的虛擬範例來加以探討，列出其搜尋結果並探討其正確性。第二部份以一個較大的實際系統設計圖中的部份來加以測試，以其原始功能來加以探討。

為了測試本研究所開發的輔助工具是否能正確無誤的找出準樣式的位置，本研究繪製了一張虛擬的系統設計圖，其類別圖如圖 4 所示。其中共包含了 9 個類別、15 個關係式，如其類別與關係的編碼如表 6 所示。而設計樣式資料庫中共有 7 個準樣式，包括一個 Adapter 準樣式、兩個 Bridge 準樣式、兩個 Composite 準樣式、一個 Decorator 準樣式、以及一個 Flyweight 準樣式，

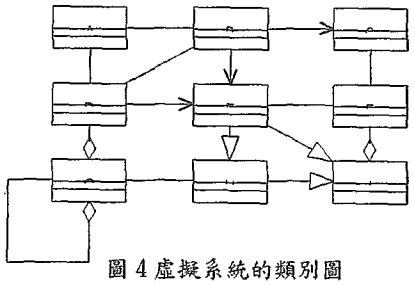


圖 4 虛擬系統的類別圖

類別名稱	類別編碼	關係編碼
A	1	1-b-2 1-b-3
B	2	2-c-4 2-c-7 2-b-3
C	7	7-b-8
D	3	3-c-4 3-a-5
E	4	4-d-6 4-b-8 4-d-9
F	8	8-a-9
G	5	5-a-5 5-b-6
H	6	6-d-9
I	9	

表 6 虛擬系統的類別圖編碼表

比對結果如表 7 所示，一共找出一組準樣式，其對應到原本的類別圖中可得到圖 5 中虛線所框起來的部份，顯示出搜尋的結果正確無誤。

準樣式名稱	對應關係編碼	對應類別
Bridge1	4-d-9 6-d-9	E-d-I H-d-I

表 7 虛擬系統的搜尋結果表

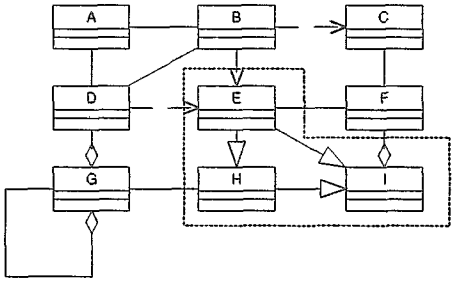


圖 5 虛擬系統的搜尋結果對應圖

本研究所採用的實際範例系統為 UML Toolkit 一書中所附的範例，功能為圖書館的電腦系統，其中包括書籍管理、租借、歸還、建檔等等功能。從其系統分析的角度來看，該系統所採用設計方式是階層式的設計法，在最外層先定義出整個架構和每一個部份物件之間的關係，然後再接著定義每一個物件中的內容。這種設計法適用於系統較為龐大，且內部功能獨立性較高的系統設計之中。

本研究取其中關於書籍管理的部份類別圖來加以測試，其類別圖（為了簡化圖形，故將所有的屬性及運作方式隱藏）如圖 6 所示，對應編碼如表 8 所示，其執行結果如表 9 所示。

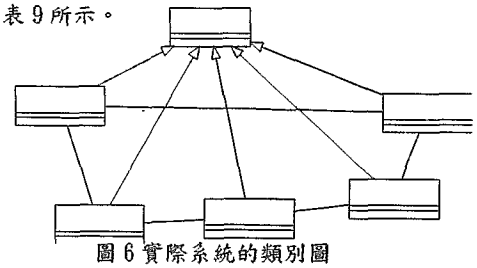


圖 6 實際系統的類別圖

類別名稱	類別編碼	關係編碼
Persistent	1	
Reservation	2	2-d-1 2-b-3 2-b-4
Title	3	3-d-1 3-b-5
BorrowerInformation	4	4-d-1 4-b-6
Item	5	5-d-1 5-b-6
Loan	6	6-d-1

表 8 實際系統的類別圖編碼表

準樣式名稱	對應關係編碼
Bridge1	2-d-1 3-d-1
Bridge1	2-d-1 4-d-1
Bridge1	2-d-1 5-d-1
Bridge1	2-d-1 6-d-1
Bridge1	3-d-1 4-d-1
Bridge1	3-d-1 5-d-1
Bridge1	3-d-1 6-d-1
Bridge1	4-d-1 5-d-1
Bridge1	4-d-1 6-d-1
Bridge1	5-d-1 6-d-1

表 9 實際系統的搜尋結果表

從搜尋結果中可以明顯的看出，整個搜尋結果都是 Bridge 的準樣式。就設計的角度來看，除了 Persistent 類別以外，其它的類別大略可以分為兩類，一類是 Title 和 Item 兩個類別，另一類是 Loan、Borrower Information、和 Reservation 等三個類別，前一類的功能是關於書籍的基本資訊，屬於較靜態不常變動的資料；後者的功能是關於書籍的租借、歸還等資訊，其資料的變動與更新較為頻繁。在整個大物件中的每一個類別都需要繼承自 Persistent 類別，因此可將此類別視為整個設計中其它類別的 super class。但是 Bridge 樣式的功能是從數個應用實例中分離出一組抽象設計，使其對應的每一個應用設計之間擁有共同標準又能彼此獨立運作，所以就功能而言，這個類別圖中並沒有 Bridge 準樣式的存在。

從上面誤判準樣式的例子可以看出，準樣式定義的優劣對搜尋結果有相當重要的影響。因為本研究只從系統設計圖形中取類別圖當做實驗對象，探討其類別與類別之間的關係，目前並無法直接依照類別圖去衡量出系統分析與設計的原義，是否真的符合搜尋結果所對應的標準樣式，因此，一切還是要靠系統設計人員根據輸出的結果自行判斷與衡量，進而考慮是否加以改善原先的系統設計。此外，本研究所提供的建議只針對類別圖，其它設計圖會不會受到影響，以及影響的層面有多大並不在考量的範圍之內，因此這一部份也需要由系統設計人員參照建議修改的部份，一步步檢查可能影響到的範圍。

6. 結論

本研究所建立的輔助工具，可以協助系統設計人員找出準樣式的位置來加以改善原有設計，進而增加軟體設計方法再使用的機會，減少系統開發人員重新思考設計方式所花費的時間，相對縮短軟體的開發時間，進而增加軟體系統的競爭力。而設計樣式資料庫的建立，除了可以協助本研究開發的輔助工具搜尋準樣式以外，還可以讓系統開發人員直接線上查詢及使用未曾使用過的設計樣式。一旦有新的設計樣式問世時，可以馬上加入資料庫中直接使用，減少系統開發人員學習新設計樣式所需的時間與精力。

關於如何在系統分析設計階段導入設計樣式，本研

究所提出的方法僅為此領域中的第一步，關於後續的研究方向整理歸納如下：

1. 準樣式的萃取範圍可再加以擴大，並考慮定義出屬於動態結構的準樣式。
2. 本研究中並不探討以設計樣式取代準樣式之後所衍生的問題，未來可以考慮朝這個方向加以研究，如此將會使工具本身更具有使用價值。
3. 在搜尋結果處理上，可以考慮透過設計過程產生的其它資訊，例如文件說明或是類別物件名稱等資訊，來加以判斷與衡量是否搜尋到準樣式，藉此過濾掉誤判準樣式的機會。

國科會計劃編號 NSC 89-2213-E-020-004

參考文獻

[1] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahlking, Shlomo Angel, "A Pattern Language", Oxford University Press, New York, 1977.

[2] Dirk Riehle, "Composite Design Patterns", OOPLSA, pp. 218-228, 1997.

[3] Dowming Yeh, and Shuching Chuang, "A Supporting Tool for Introducing Design Patterns into the Object-Oriented Development", ISAS/SCI' 99.

[4] D. Coleman, P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes, P. Jeremes, "Object-Oriented Development : The Fusion Method", Prentice Hall, 1994.

[5] Erich Gamma, "Object-Oriented Software Development Based on ET++", PhD thesis, University of Zurich, Institut for Infomatik, 1991.

[6] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, "Design Patterns : Abstraction and Reuse of Object-Oriented Design", European Conf. On Object-Oriented Programming, Kaiserslautern, Germany, July 1993.

[7] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, "Design Patterns - Elements of Reusable Object-Oriented Software", Addison Wesley, 1995.

[8] F. J. Budinsky, M. A. Finnie, J. M. Vissides, and P. S. Yu, "Automatic Code Generation from Design Patterns", IBM System Journal, Vol 35, No. 2, 1996.

[9] Grady Booch, "Object-Oriented Analysis and Design with Applications", Benjamin/Cummings, Redwood City CA, 1994, Second Edition.

[10] Hands-Erik Eriksson and Magnus Penker, "UML Toolkit", John Wiley & Sons, Inc, 1998.

[11] Ivar Jacobson, Magnus Christerson, Patrik Jonsson, Gunnar Overgard, "Object-Oriented Software Engineering- A Use Case Driven Approach", Addison Wesley, Workingham, England, 1992.

[12] James O. Copien, "Advanced C++ : Programming Styles and Idioms", Addison Wesley, 1992.

[13] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorenson, "Object-Oriented Modeling and Design", Prentice Hall, Englewood Cliffs, NJ, 1991.

[14] Ken Beck, "Using a Pattern Language for Programming", Addendum to the Proceeding of OOPLSA '87, vol. 23, 5 of ACM SIGPLAN Notices, p. 16, May 1988.

[15] Robert T. Monroe, Andrew Kompanek, Ralph Melton, David Garlan, "Architectural Styles, Design Patterns, and Objects", IEEE Software, pp. 43-52, Jan 1997.

[16] William J. Brown, Raphael C. Malveau, Hays W. McCormick III, Thomas J. Mowbray, "Anti

Patterns – Refactoring Software, Architectures, and Projects in Crisis", John Wiley & Sons, Inc., 1998.

[17] 賀元，劉燈，賴明宗，“世紀末軟體革命2”，資訊人文化事業有限公司。1996.