VEIP：一個用於資訊處理之視覺化環境*
VEIP: a Visual Environment for Information Processing

廖珙洲　楊國樺　楊益誠　朱寶堂
朝陽科技大學 資訊工程系
Hsien-Chou Liao, Kuo-Hua Yang, Yi-Cheng Yang, and Pao-Tang Chu
Department of Computer Science and Information Engineering
Chaoyang University of Technology
E-mail: {hcliao, s8927045, s8927005, s8927059}@mail.cyut.edu.tw

摘要

由於網際網路的蓬勃發展，提供了一個便利的管道讓人們獲得各種資訊，而取得這些資訊後還需要進行各種處理動作，就這些處理動作而言，可以概分為五類：「擷取」、「分析」、「歸納」、「轉換」與「索引」。目前這些動作大都由人腦來進行，然而隨著資訊量的增加，資訊處理逐漸成為日常生活中沈重的負擔。

有許多研究提出方法來克服上面的問題，不過這些方法仍無法完全克服下面兩個主要問題，第一：處理動作的定義與調整不易，當面對新的資訊或者原有輸入資訊的格式或是內容改變時，無法快速定義所需的處理動作來滿足使用者的需求。

第二：設計過的處理動作無法重複利用，當使用者描述好資訊處理的動作後，沒辦法直接運用這些描述過的動作來處理其他的資訊，這使得應用範圍受限，也無法提升資訊處理的能力以滿足更複雜的需求。

為了克服上面的問題，在本論文中提出一個用於資訊處理的視覺化環境-VEIP。VEIP 特色之一在於提供了視覺化的操作介面來讓使用者設計處理動作，將整個動作以流程圖的方式來描述，並且可以透過 VEIP 中的執行模組來執行這些處理動作。另一個特色是 VEIP 中設計一個「階層式結構」，允許階層式的資訊處理流程，亦即某一個已設計好的資訊處理流程圖可以被包裝為一個流程元件，然後應用在另一個資訊處理流程圖中。

因此，藉由階層式結構的設計就可以讓使用者直接包裝設計過的流程圖並且運用在新的處理流程上，透過這樣的設計可以讓使用者設計出更強大的資訊處理流程，以因應越來越複雜的資訊處理需求，進而提升 VEIP 的應用範圍。除此之外，在本論文中也利用二個實例來展示 VEIP 的實用性。

關鍵詞：資訊處理，視覺化環境，階層式結構

* 本研究接受國科會 NSC 90-2213-E-324-008 研究計畫之補助

Abstract

Internet becomes a convenience way for peoples to access any kinds of information. When information is retrieved from Internet, peoples have to do further processing to acquire the needed part. According the functions of information processing, they can be classified into five categories: extract, analysis, induction, transform, and index. Currently, most of peoples perform these functions manually. The increasing of information also increases the load on information processing.

Many approaches are proposed to overcome the above problem. However, two drawbacks are still existed. One is that the design and adjustment of a process is difficult. That is, it is troublesome to define process or to adjust an existing process for new information. The other drawback is that existing processes are unable to reuse in a new process. It limits the application areas and the capability of these approaches.

In this paper, VEIP is proposed to overcome the above drawbacks. One of the VEIP's features is that users can utilize a visual tool to describe the process as a flow diagram. A flow diagram can be executed by VEIP's execution module. Another feature is that a layered structure is designed in VEIP. It allows an existing flow diagram can be packaged into a flow component. Such component can be used in another flow diagram. Users can thus design powerful flow diagrams to satisfy more and more complex requirement on information processing. In addition, two practical examples are used to demonstrate the VEIP's feasibility.

Keywords: Information process, visual environment, layered structure.

一、 簡介

從網際網路的出現至今，已經慢慢地在改變人們對於取得資訊的方法，而且由於網際網路深入到世界各地，方便了人們之間資訊的傳遞，所以人們也漸漸習慣地從網際網路上取得各式各樣的資訊。

由於網際網路與電腦的普及，雖然提供一個良好的資訊存取與分享的管道，但是卻也將世界引進了資訊爆炸的時代，每年資訊的成長量更以加倍的速度在增加中，因而有許多的研究領域分別從各個不同的角度來希望改進資訊與知識的獲得。不過，若是就「資訊處理」的方法來看，可以歸納為下列幾類：

1. 擷取(extract)：從資訊中擷取特定內容。
2. 分析(analysis)：分析出特定內容。
3. 歸納(induction)：歸納出特定的結果。
4. 轉換(transform)：將資訊進行特定轉換。
5. 索引(index)：對於資訊的某些或特定部分提供索引的動作。

這些不同的方法主要是為了拉近「資料來源」與「處理結果」之間的距離，也就是讓使用者更容易的處理一項資訊，其實在全球資訊網尚未廣泛受到重視之前，許多資訊處理的研究就已經開始進行了[9][10][1]，而當全球資訊網興起之後，資訊處理更是受到重視，處理的資訊除了文字之外，還包含影像、聲音或甚至是電子郵件[3]等，另外有些研究也從許多不同的角度來進行，例如：有些研究利用資料流程圖的方式來設定原先必須以程式撰寫的處理流程[5][7]，或是提供一個視覺化的程式設計環境[6]，也有人利用一個可調整的軟體代理人(software agent)來應付各種處理資訊時會面臨的狀況[8]，還有一些研究提供視覺化的操作介面讓使用者來處理資訊[4][2]。不過，這些研究還是無法完全克服下面二個主要的問題：

1. 處理動作的定義與調整不易：當使用者面對新的資訊或者因為資訊處理上需求的變動而需要調整處理動作時，例如：對於不同的資訊要得到相同的結果，或是針對相同的資訊要得到不同的結果，這些工具並無法讓使用者容易的定義與調整資訊處理的動作。
2. 設計過的處理動作無法重複利用：當使用者描述好一個資訊處理的動作後，沒辦法直接運用這些描述過的動作來處理類似的資訊，這使得方法的應用範圍受限，也無法提升資訊處理的能力以滿足更複雜的需求。

為了克服上面的問題，在本論文中提出一個用於資訊處理的視覺化環境-VEIP。VEIP 特色之一在於提供了視覺化的操作介面來讓使用者設計處理動作，將整個動作以流程圖的方式來描述，並且可以透過 VEIP 中的執行模組來執行這些處理動作。另一個特色是 VEIP 中設計一個「階層式結構」，允許階層式的資訊處理流程，亦即某一個已設計好的資訊處理流程圖可以被包裝為一個流程元件，然後應用在另一個資訊處理流程圖中。

因此，藉由階層式結構的設計就可以讓使用者直接運用設計過的流程圖在新的處理流程上，透過這樣的設計可以讓使用者設計出強大的資訊處理流程，以因應越來越複雜的資訊處理需求，進而提升 VEIP 的應用範圍。另外，在本論文中也利用二個實例來展示 VEIP 的實用性。

二、 階層式的資訊處理流程

底下使用一個實例來說明這樣的觀念。假設使用者想購買某一項產品（例如：液晶螢幕 BENQ FP581s），並且已知某個網站有提供價格的資訊（例如：www.arclink.com.tw），當每次檢視網頁內容時，使用者只想知道目前價格，並且希望可以將價格資訊記錄到檔案中，以瞭解價格的變化情形。下圖是使用者所看到的網頁畫面：

圖 1：查詢價格的網頁畫面

一般使用者面對上述的需求時，大都會定時上網察看網頁內容，然後再擷取出價格資訊，這樣的動作看起來似乎很簡單，不過若是一次要察看好幾個網站，這樣的動作就會造成使用者的負擔。

若是以流程圖來描述上面的資訊處理動作，一個假想的流程圖將會如下圖所示：

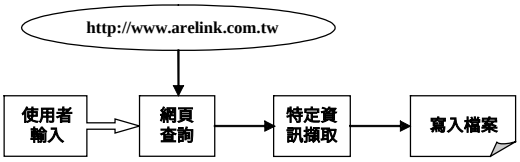


圖 2：一個假想的資訊處理流程圖

圖中說明在「使用者輸入」方塊是讓使用者定義想要詢價的產品名稱，接著，透過「網頁查詢」來取得價格的網頁，這些網頁依據使用者輸入的產品名稱進行「特定資訊擷取」的處理，取出價格的資訊，最後透過「寫入檔案」的動作將價格寫入檔案中。

從上述的範例可以清楚的了解，若是以視覺化的方式來設定資訊處理流程的話，將大幅地降低使用者定義流程上的複雜度。

接著說明「階層式結構」的概念，如下圖所示：

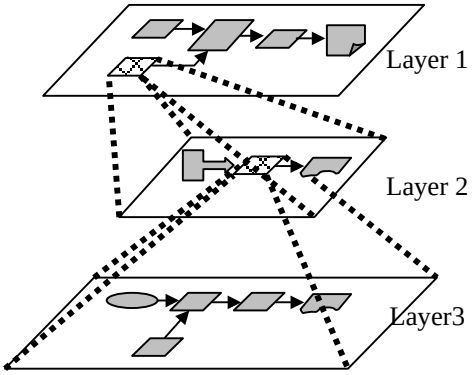


圖 3：階層式結構示意圖

圖中共有三個「階層」(layers)，每一個階層表示一個設計好的「資訊處理流程」，而「階層式結構」指的就是一個設計過的資訊處理流程可以成為更上面一層的處理流程中的一個動作，就像是 Layer 3 是 Layer 2 流程中網狀部分的動作，而 Layer 2 則是 Layer1 流程中網狀部分的動作。因此，一個最上層的處理流程所包含的動作，可能由底下好幾層的流程所建構上來的。

我們再回到前面圖 2 的實例來更清楚說明處理流程的階層式結構。在前面的例子中，「特定資訊擷取」的動作必須取出 HTML 網頁表格中特定的列與欄，並且去除原先網頁中 HTML 的標籤，因此，「特定資訊擷取」的動作可以進一步描述為下面的流程：

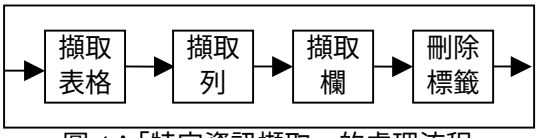
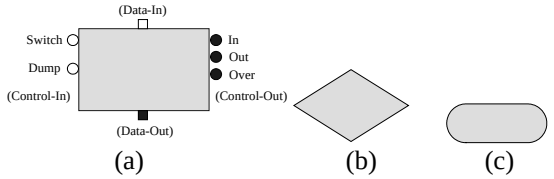


圖 4：「特定資訊擷取」的處理流程

所以，圖 2 與圖 4 就形成了一個階層式結構，並且圖 4 還可以用在其他的網頁中表格的擷取，只需調整要擷取的列或者欄就可以了。因此，透過上面例子的說明，可以清楚的了解到「階層式結構」的好處，如同硬體的設計一般，可以使用基本的資訊處理動作來設計更高階的處理流程，一旦使用者面臨新的資訊處理需求時並不需要從頭開始，可以利用已經設計過的流程，大幅增加流程描述上的彈性以及擴充性。

三、 流程圖表示法

為了滿足使用者在描述資訊處理流程上的各種需求，因此定義不同的符號，這些符號包含「元件」、「輸入」、「判斷」、「連接線」與「連接點」，如下圖所示：



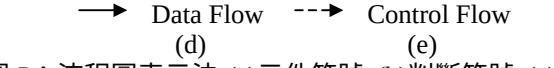


圖 5：流程圖表示法-(a)元件符號 (b)判斷符號 (c)輸入符號 (d)資料流 (e)控制流

1. 「元件」：不管是基本元件或是流程元件都如圖 5(a)所示，元件中央顯示元件型態與名稱。
2. 「判斷」：判斷特定內容以控制流程運作。
3. 「輸入」：使用者可以用輸入符號來定義元件或者判斷符號所需的數值。
4. 「連接線」：連接線區分為資料流與控制流，分別以實線與虛線來表示。
5. 「連接點」：連接點分為資料輸入、資料輸出、控制輸入與控制輸出四種。這裡利用方形與圓形來區別資料與控制，並且用空心與實心來區別輸入與輸出，如圖 5(a)中的標示。這裡特別說明一個元件必備的五個控制連接點，其中有二個控制輸入點(Switch, Dump)，與三個控制輸出點(In, Out, Over)。Switch 用來控制元件的啟動與否，Dump 設定為“On”時，有資料輸入這個元件或者這個元件輸出某個資料，都會將元件目前所有連接點的內容顯示在一個特別設計的監控視窗中，這些資料可以用來找出流程圖上的錯誤。另外，當元件接收到資料時會輸出 In 的訊號，當輸出一項資料時會輸出一個 Out 訊號，當某個輸入完全處理完畢時會輸出一個 Over 訊號，這些訊號可以提供給使用者來控制流程的執行順序。

使用者可以利用上面這些表示法來描述出所需的各種流程，下圖表示一個流程圖的範例：

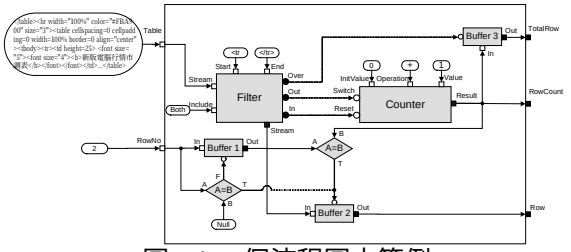


圖 6：一個流程圖之範例

上圖表示擷取 HTML 表格中的每一列的流程圖，具有二個輸入與三個輸出，並包含三個基本元件 Filter, Counter 與 Buffer，其中 Filter 會將輸入的 Stream，擷取出 Start 字串到 End 字串的這一段資訊，而 Include 是表示輸出時要不要包含 Start 與 End 這二個子字串（Both 表示兩者都要），Counter 元件則是用來進行計數的動作，可以設定初始值(InitValue)，向上計數或者向下計數(Operation)，以及每次計數的增減值(Value)，Buffer 元件則只是一個暫存的動作，依照 Switch 來決定要不要將輸入 In 送到輸出 Out。

所以在圖 6 中根據輸入的表格，指定要取出第 2 個列的內容，並且輸出時要包含左與右兩邊的標籤。因此，當表格輸入到 Filter 的輸入 Stream 時，Filter 的控制訊號 In 會啟動，並且傳送到 Counter 的 Reset 來將 Counter 歸零。接著，每當 Filter 輸

出一個列時，Out 控制訊號會啟動 Counter 來加 1，並且每次 Counter 的結果都送到中間的 “A=B”的判斷符號來判斷是否等於 2，一旦等於 2 的話，就會啟動下方 Buffer 2 的 Switch 來將目前這一個列的資料輸出到右下角的 Row。特別注意的是，在流程圖的左下角有一個 “A=B” 的判斷符號，用來判斷指定列的輸入是否為 Null，由於目前的例子並不是 Null，因此，會啟動其上方的 Buffer 1 將輸入值 2 傳送給右邊的判斷符號。若是等於 Null 時，會一直啟動其右邊的 Buffer 2，讓每個列都會依序輸出去。當所有的列都輸出後，Filter 元件會送出 Over 的控制訊號，來控制 Buffer 3 將 Counter 最後的值輸出，這也就是 Row 的總數 (TotalRow)。因此，目前設計的這個流程，若是使用者不指定要哪一列的話，會依序輸出每個列，並且最後輸出目前這一列的序號(RowCount)。

四、 VEIP 系統架構

以「階層式結構」的觀念所設計出來的 VEIP，其系統架構如下圖所示：

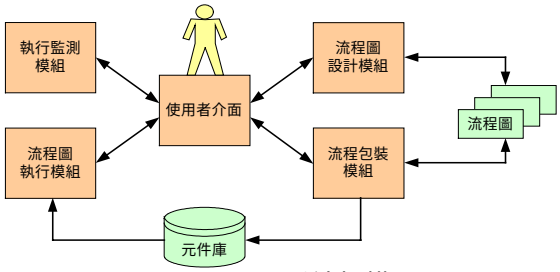


圖 7：VEIP 系統架構圖

架構圖中每個部分的內容與功能說明如下：

1. 「流程圖設計模組」：此模組可以讓使用者利用各種元件來設計流程圖，負責處理所有的操作動作，包含流程圖的繪製，拖拉，連接線的繪製，以及流程圖的儲存與載入等動作。
2. 「流程圖」：以 XML 的格式來存放流程圖中每個元件與連接線的屬性以及相關設定。
3. 「流程包裝模組」：此模組提供給使用者來將某個流程圖包裝成一個元件（稱為「流程元件」）來供上層使用。
4. 「元件庫」：元件庫用來存放繪製流程圖的各種元件，除了「基本元件」之外，還包含流程圖經過流程包裝模組所得到的「流程元件」。
5. 「流程圖執行模組」：此模組為負責流程圖的執行，由於流程圖中會包含流程元件，因此必須先將流程圖展開到最底層的基本元件，接著便是依照流程圖的動作依序來執行。
6. 「執行監測模組」：由於流程執行的過程中，使用者必須能夠監測某個元件的輸入與輸出，以便瞭解流程圖的設計是否正確。因此，此模組可以列出執行過程中，有設定 Dump 控制輸入點為”On”之元件的輸入與輸出內容，以利流程的除錯。

在 VEIP 中利用這些模組的協同合作，透過「使用者介面」來提供一個良好的資訊處理環境。這裡用例子來說明一個流程包裝動作，如圖 8 所示：

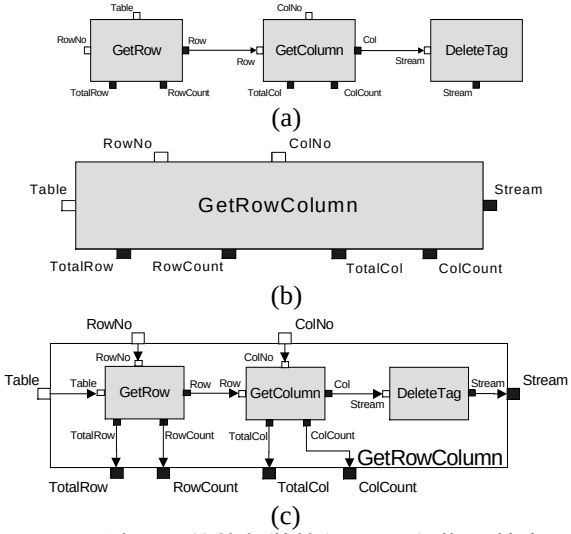


圖 8：流程元件的包裝範例-(a)一個擷取特定列與欄的流程圖 (b)一個包裝後之流程元件 (c)流程元件之內部示意圖

圖 8(a)為欲包裝之流程圖，經過此模組的包裝後的流程元件如圖 8(b)所示，圖 8(c)表示流程元件與原先流程之間的關係。利用流程包裝模組，使用者可將某個流程圖包裝後供日後重複使用。

五、 實作展示

目前 VEIP 的系統是以 Microsoft.NET 的 C#語言所開發出來的，這裡先說明 VEIP 中用來協助使用者找出流程圖錯誤的「執行監測模組」，前面流程圖表示法中談到每個元件都具有一個 Dump 的控制輸入，若是給予 “On” 的輸入值時（預設是 “Off”），就表示當這個元件有資料輸入或輸出時，需要將此元件各個連接點的內容列印出來。因此，這個執行監測模組的運作就是當發現資料輸出或輸入時，會檢查對應元件的 Dump 輸入值是否為 “On” ，若是就會將這個元件所有的連接點分為控制輸入(Ctrl In)、資料輸入(Data In)、控制輸出(Ctrl Out)與資料輸出(Data Out)四類分別列出，下圖所顯示的畫面是將圖 6 流程圖中元件的 Dump 都設定為 “On”之後，執行時所列印出來的結果。

圖 9：監測模組的輸出畫面

所以使用者可以從上面列印出來的內容，來檢視元件的輸入與輸出結果是否是原先所預期的，並藉此修正流程圖上的錯誤。

接著使用兩個實例來說明 VEIP 的實用性，第一個實例是從網路上收集資訊並整理成文字格式後存放在檔案中。第二個實例是監視某一項物品的價格，一旦發現價格低於某個設定值的時候，直接將網頁內容以 E-mail 通知使用者。

在第一個實例中，使用者主要處理的資訊是一份價格的清單，如下圖所示：

圖 10：價格清單的網頁畫面

由於使用者想要知道 8 月份中每一週所有物品的價格資料，並且將這些物品的廠牌、品名、價格等資料以逗號隔開後存放在文字檔中，針對這樣的需求，使用 VEIP 所描繪出來的流程圖如圖 11 所示。

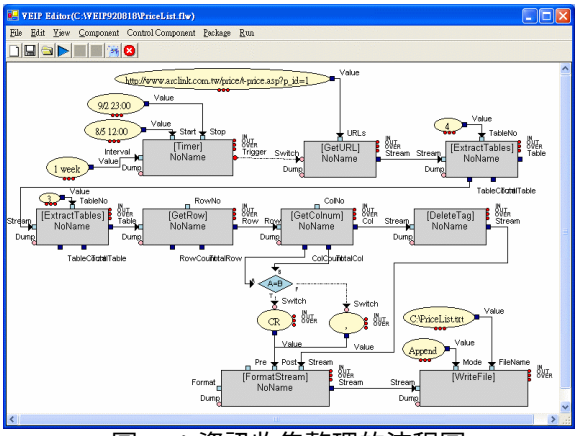


圖 11：資訊收集整理的流程圖

在圖 11 的左上角，首先有一個 “Timer”元件，其設定的週期為 “1 week”(一週)，並且設定啟動的時間，因此時間到達後每一週都會送出一個 Trigger 的訊號，來啟動 “GetURL”元件，接著，這個元件就會到使用者所指定的 URLs 來取得網頁內容並輸出到 Stream，當 “ExtractTable” 元件收到 “GetURL” 所送來的 “Stream”之後，就會將第 4 個 Table 取出來（因為流程上設定所要取出的表格編號 TableNo 為 4），由於取出來的內容還包含好幾個 Tables，因此再透過另一個“ExtractTable” 元件，取出所要的第 3 個表格。然後送給下面的元件，當 “GetRow” 得到 Table 的資料後，由於沒有指定 “RowNo”，因此會將每一列依序輸出，然後再送給 “GetColumn”，這個元件會先算出總共有多少個 Column，並且輸出 “TotalCol”，然後再將每個 Column 的內容送到下一個“DeleteTag”元件，用來刪除標籤，當 HTML 的標籤被刪除之後，就會送給 “FormatString” 進行格式化的動作，將每個 Column 的資料後面加上一個 “,”或是 CR（跳行），這是根據菱形的判斷元件，來判斷目前送出來的 Column 資料是否等於 TotalCol（最後一欄），若是的話，則加上一個 CR 字元，否則加上“,”。接著 Stream 會送到 “WriteFile” 元件，並依照使用者所訂定的寫入格式（Append 表示附加在檔案後），儲存到“C:\PriceList.txt” 檔案中。

因此，上面的網頁內容經過處理之後，所得到的結果如下面圖 12 的文件檔所示。

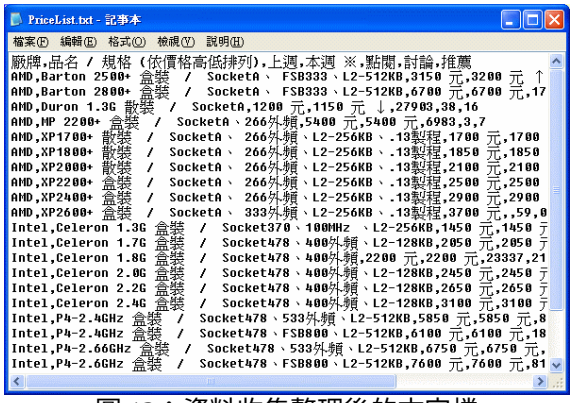


圖 12：資料收集整理後的文字檔

從上面例子的說明，藉由流程圖中的元件將資訊一步一步地處理成使用者想要的結果，讓使用者可以容易的清楚整個處理的流程，也更容易地設計出整個處理的動作。

接著，在第二個實例中，假設使用者想要監視某個購物網站的網頁中某一台數位相機的價格(假設是 KODAK DX4330 數位相機)，如圖 13 所示：

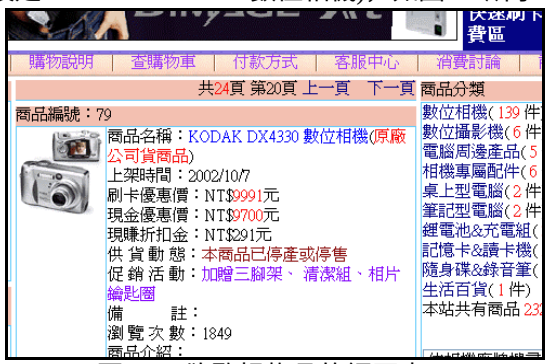


圖 13：欲監視物品的網頁畫面

一旦這台相機的價格低於 10000 元以下時，就將當時的網頁內容寄到使用者的電子郵件信箱中。根據上述的需求而言，基本的處理流程會每天檢查價格資訊，一旦發現符合使用者的要求時，就寄出電子郵件。因此，使用 VEIP 所描述出來的流程圖如圖 14 所示。

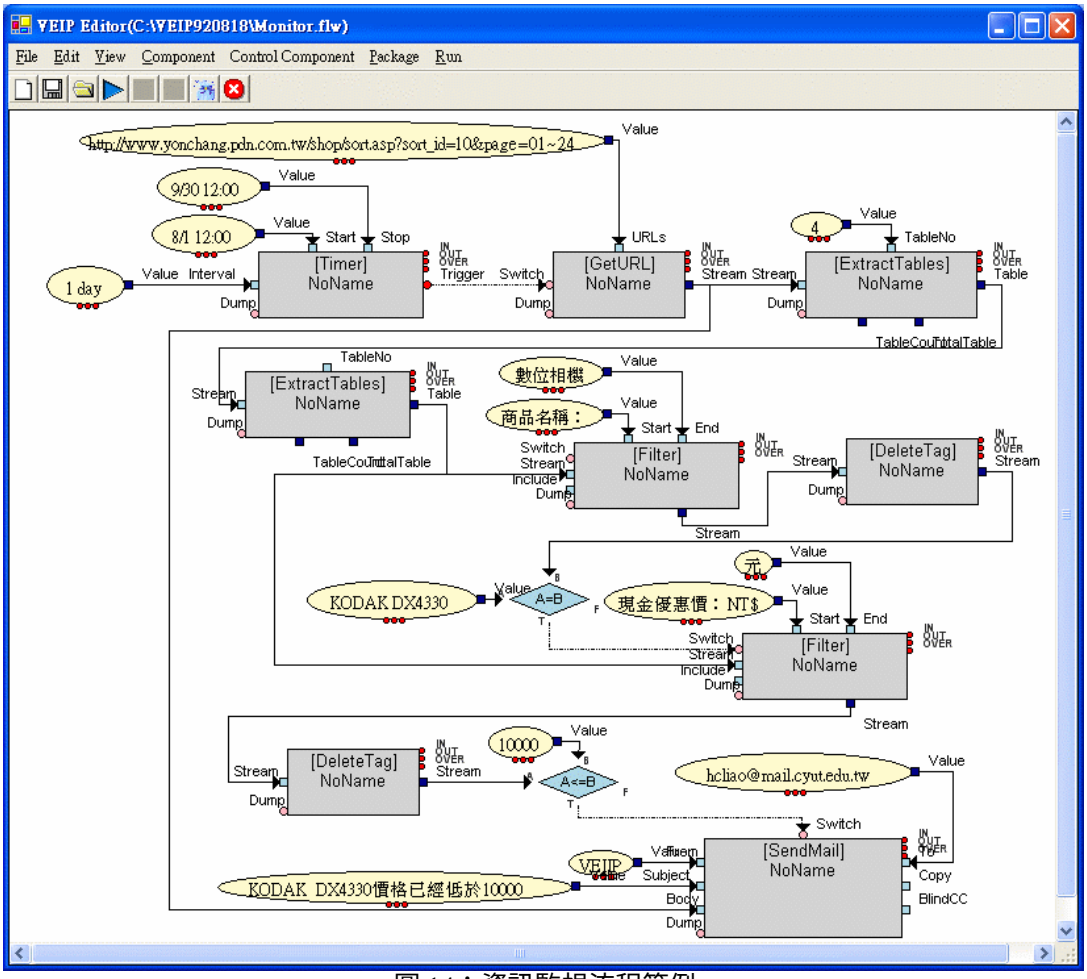


圖 14：資訊監視流程範例

在圖中左上角，首先有一個 “Timer” 元件，其設定的週期為 “1 day” (一天)，當啟動時間開始後，每天送出一個 Trigger 的訊號來啟動 “GetURL”元件， “GetURL” 元件會將指定的 URLs 網頁內容由 Stream 送出，這裡所指定的 URLs 一共有 24 個(01~24)，要一次監視這麼多網頁，主要原因是由於物品清單會更動的緣故，導致這台相機的資訊可能出現在其中任何一頁。當 “ExtractTable” 元件收到這個網頁內容，就會取出使用者所指定的第 4 個表格 (“TableNo”= 4)，而由於這個表格內還包含好幾個表格依序存放不同物品的價格資訊，因此，這裡又透過另一個 “ExtractTable” 元件，來擷取出每一個物品的表格，接著直接利用“Filter”元件來取出表格中從 “商品名稱：”到 “數位相機” 之間的資訊，這個部分的資訊就是要檢查的物品名稱，這個名稱的部分透過“DeleteTag”元件將 HTML 的標籤刪除後，送給一個判斷元件看看是否等於“KODAK DX4330”，若是的話，則送一個 Trigger 訊號啟動 “Filter” 元件，並將價格的部分取出來 (價格的資訊是從 “現金優惠價：NT$” 到 “元” 的內容)，接著，價格資訊經過“DeleteTag”元件來刪除標籤之後，判斷是否小於等於 10000，若是的話，則送一個 Trigger 訊號來啟動 “SendMail” 元件，這個元件就會將原來的網頁內容以電子郵件的方式傳送到設定的信箱。下圖顯示使用者所收到郵件內容的畫面：

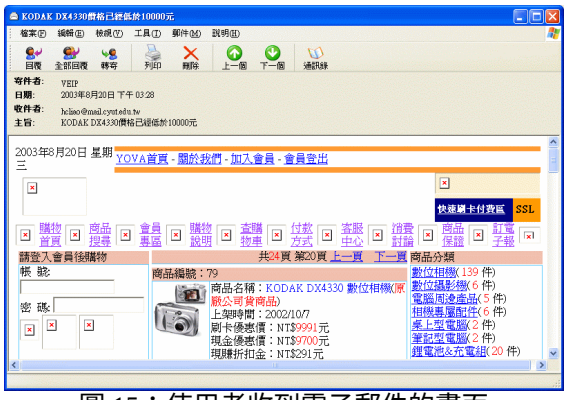


圖 15：使用者收到電子郵件的畫面

從上面這兩個實例的說明，可以瞭解到 VEIP 提供了一個便利的環境來讓使用者自行設計資訊處理流程的能力，克服了前面所提的「不易定義與調整」以及「流程無法包裝重複利用」的問題。雖然目前這兩個實例的流程圖似乎有些複雜，但是配合「階層式結構」的設計，使用者可自行包裝出各種不同的流程元件，並將整個動作包含在好幾層的流程圖中，如此可以簡化最上層流程圖的複雜度。

六、結論

資訊處理的動作幾乎是現代人每天不可避免的工作，許多較為例行性的工作若是可以使用 VEIP 中的流程圖來描述，相信可以減少許多不必要的負擔，並且藉由 VEIP 中的階層式結構，提升了 VEIP 本身的應用範圍，讓使用者可以設計出更強大的流程以處理更複雜的資訊。

目前針對資訊的輸入（網頁、檔案等），處理（過濾、擷取、判斷等），以及資訊的輸出（檔案、電子郵件等），VEIP 都實作了這些基本元件供使用者自行組裝出新的流程元件，但是並無法容易的加入新的基本元件，因此，未來努力的方向之一是提供一個容易而且快速的管道可以讓 VEIP 新增這些基本元件。

另外，由於使用者設計資訊處理流程時必須熟悉資訊輸入的格式，以目前最常接觸到的網頁而言，大部分使用者並不瞭解這些網頁的 HTML 格式，導致設計所需的流程上有所阻礙。因此，在未來另一個努力的方向是提供一個「流程產生模組」，用來將一些基本的流程自動產生出來，加快使用者定義流程上的速度。

在資訊技術不斷地提升其功能與應用領域的同時，一般使用者有相當多的需求卻尚未被有效的滿足，起因於使用者大都無法自行擴充來超越系統原先實作出來的功能範疇。本論文中所提出來的 VEIP，針對資訊處理的需求，利用階層式的結構來讓使用者建立出更高階的流程元件，未來會持續將類似的設計概念應用到更多的使用者需求上。

七、參考文獻

[1] C. Ahlberg and B. Shneiderman, “Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays,” *Proc. on Human Factors in Computing Systems (CHI’94)*, April 24-28, 1994, pp. 313-480.

[2] S. Cranefield, E. Moreale, B. McKinlay, and M. Purvis, “Automating the Interoperation of Information Processing Tools,” *Proc. of the 32nd Annual Hawaii International Conference on System Sciences*, Jan. 1999.

[3] B. Heckel and B. Hamann, “EmVis-A Visual E-mail Analysis Tool,” *Proc. of the Workshop on New Paradigms in Information Visualization and Manipulation*, 1998, pp. 36-38.

[4] H. W. Hsieh and F. M. Shipman III, “VITE: A Visual Interface Supporting the Direct Manipulation of Structured Data Using Two-Way Mappings,” *Proc. of the 2000 International Conf. On Intelligent User Interfaces*, 2000, pp. 141-148.

[5] R. Idini, M. Mosconi, and M. Porta, “Programming Web-Based Application within a Data-Flow VL,” *Proc. of IEEE Symposium on Visual Languages*, 1998.

[6] G. Karsal, “A Configurable Visual Programming Environment,” *IEEE Computer*, March 1995, pp. 36-44.

[7] C. Lee and Y. T. Chen, “An Embedded Visual Programming Interface for Intelligent Information Retrieval on the Web”, *Proc. of 1997 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX '97)*, Nov 4, 1997, pp. 46-53.

[8] D. Rus and D. Subramanian, “Customizing Information Capture and Access,” *ACM Trans. on Information Systems, Vol. 15, No. 1, Jan. 1997*, pp. 67-101.

[9] D. H. Sonnenwald, “Developing a Theory to Guide the Process of Designing Information Retrieval Systems,” *Proc. of the Fifteenth Annual International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR’92)*, June 21-24, 1992, pp. 310-317.

[10] A. Spoerri, “InfoCrystal: A Visual Tool for Information Retrieval & Management,” *Proc. of the third international conference on Information and knowledge management (CIKM’94)*, Nov. 29-Dec. 2, 1994, pp. 11-20.