I327701

公告本

第92113389號專利案99年1月修正

99年1月21日修正本

發明專利說明書

（本說明書格式、順序及粗體字，請勿任意更動，※記號部分請勿填寫）

※ 申請案號：92113389

※ 申請日期：92 年 5 月 16 日　　※IPC 分類：G06F17/30(2006.01)

一、發明名稱：（中文/英文）

在標示文件內宣告地定義與使用子類別的系統和方法

SYSTEM AND METHOD FOR DEFINING AND USING SUBCLASSES DECLARATIVELY WITHIN MARKUP

二、申 請 人：（共 1 人）

姓名或名稱：（中文/英文）

美商・微軟公司

Microsoft Corporation

代表人：（中文/英文）

史納普瑪莉

SNAPP, MARY

住居所或營業所地址：（中文/英文）

美國華盛頓州列德蒙微軟路 1 號

One Microsoft Way, Building 8, Redmond, WA 98052-6399, U.S.A.

國　籍：（中文/英文）

美國/USA

三、發明人：（共 3 人）

姓　名：（中文/英文）

1.山德瑞拉曼尼/Sundaram Ramani

2.羅伯 A. 瑞利亞/Robert A. Relyea

3.傑夫瑞 L. 博登/Jeffrey L. Bogdan

國　籍：(中文/英文)

1.印度/India

2.美國/USA

3.美國/USA

四、聲明事項：

☐ 主張專利法第二十二條第二項☐第一款或☐第二款規定之事實，其事實發生日期為：　年　月　日。

☑ 申請前已向下列國家（地區）申請專利：

【格式請依：受理國家（地區）、申請日、申請案號　順序註記】

☑ 有主張專利法第二十七條第一項國際優先權：

美國；2003 年 2 月 28 日；10/377,313

☐ 無主張專利法第二十七條第一項國際優先權：

☐ 主張專利法第二十九條第一項國內優先權：

【格式請依：申請日、申請案號　順序註記】

☐ 主張專利法第三十條生物材料：

☐ 須寄存生物材料者：

國內生物材料 【格式請依：寄存機構、日期、號碼　順序註記】

國外生物材料 【格式請依：寄存國家、機構、日期、號碼　順序註記】

☐ 不須寄存生物材料者：

所屬技術領域中具有通常知識者易於獲得時，不須寄存。

國　籍：(中文/英文)

1.印度/India

2.美國/USA

3.美國/USA

四、聲明事項：

☐ 主張專利法第二十二條第二項☐第一款或☐第二款規定之事實，其事實發生日期為：　年　月　日。

☑ 申請前已向下列國家（地區）申請專利：

【格式請依：受理國家（地區）、申請日、申請案號　順序註記】

☑ 有主張專利法第二十七條第一項國際優先權：

美國；2003 年 2 月 28 日；10/377,313

☐ 無主張專利法第二十七條第一項國際優先權：

☐ 主張專利法第二十九條第一項國內優先權：

【格式請依：申請日、申請案號　順序註記】

☐ 主張專利法第三十條生物材料：

☐ 須寄存生物材料者：

國內生物材料 【格式請依：寄存機構、日期、號碼　順序註記】

國外生物材料 【格式請依：寄存國家、機構、日期、號碼　順序註記】

☐ 不須寄存生物材料者：

所屬技術領域中具有通常知識者易於獲得時，不須寄存。

九、發明說明：

【發明所屬之技術領域】

本發明係有關於在標示文件內宣告地定義與使用子類別的系統和方法。

【先前技術】

現今的軟體開發工具允許軟體開發者使用 C, C++, C# 等一種或多種相關之程式語言，以建構可執行的組件(component)。建構可執行的組件的優點有二，其一在於該組件可以供其它軟體程式重複使用；其二為新的組件可輕易的由已建構的組件衍生而來。

一般而言，組件由子類別擴充而來，也就是新類別由已存在的類別衍生而來。這些類別和子類別是由程式語言中的一種撰寫而成，由這些程式語言所撰寫之程式碼就是所謂的原始碼。在傳統的執行環境中，軟體開發工具將這些原始碼編譯為目的碼(object code)，接著並與其它相關的目的碼連結以產生可執行程式。然而，傳統程式執行環境(runtime environment)存在之問題為每一程式語言或者程式語言的每一版本需要不同的執行環境。

因此，為了克服傳統執行環境的限制，係設計了一種新型態的的程式執行環境，該設計能夠有效的克服上述在傳統執行環境中所存在跨語言介面與語言版本的問題。在這個新型態的執行環境中，程式開發工具將原始碼編譯為中繼語言(intermediate language)，在程式執行期間，執行

環境係將中繼語言編譯為原生二元可執行碼(native binary executable code)。因此，該新型態的執行環境在程式執行期間係執行「連結型態」程序，而為了執行「連結型態」程序，該執行環境係讀取正執行中程式之關聯組件的相關資訊(如：中介資料)並存取相關之IL組合語言。該中介資料(metadata)包含型態、版本與資源等相關資源的描述。而該IL組合語言為單一或多個動態連結庫(DLL/DLLs)和其資源。

無論是傳統或新型態的程式執行環境，程式原始碼皆是使用一種程式語言所撰寫。每一種程式語言，針對特定的程式執行環境，都有其獨特的語法與應用程式介面(API)組。因之，為了撰寫程式碼，軟體開發者必須學習與該特定的執行環境相關聯的程式語言的語法與該APIs。學習程式語法與APIs對開發者而言是非常費時與充滿挑戰的工作。此外，假設程式開發者希望使用多種程式語言撰寫程式或者讓相同程式在不同執行環境中執行，該程式開發者必須能夠熟記不同執行環境的各個程式語言語法及該APIs之異同處。

有鑒於使用組件的優點，因此有必要設計一個較佳建構、擴充與使用組件的機制。

【發明內容】

本發明係關於在標示文件內宣告地定義、擴充與使用子類別的系統和方法。本發明提供一種機制讓開發者以一

種標示語言進行建構、擴充與使用組件。這些組件包括可重複使用的組件、應用程式使用者介面與文件使用者介面等相關組件。該機制讓開發者不需要熟知任一種程式語言；相反的，該機制允許開發者使用一種熟知的標示語言，如 XML(可擴充標示語言)，以建構組件。由於 XML 容易學習並且為一般電腦程式社群所熟知，本發明相較於一般程式語言提供更多的優點。第一個優點為組件可隨著其它標示文字一起定義在標示文件中用以產生一複雜的(sophisticated)電子文件。另一優點為開發者不需知道或了解任何程式語言以產生一可執行組件。

本發明提供一種使可執行組合語言能夠與撰寫於標示文件內的子類別定義所產生的子類別相關聯之系統、方法與資料結構。根據本發明，該子類別定義根據一架構而撰寫，而該架構可為以可擴充標示語言為基礎的。該架構包含用以定義該子類別之名稱的子類別標籤，當執行該可執行組合語言時，則該名稱與被樣例化物件之種類相關聯。該架構更包含一或多個指示以供，如具體指明用以編譯子類別定義的程式語言、具體指明用以衍生子類別的父類別、具體指明當物件被樣例化時應執行的動作、產生子類別之一事件定義與相關之事件處理器(EventHandler)，以及具體指明當物件被樣例化時成為物件內欄位(field)之一特性，。

【實施方式】

本發明係關於在標示文件內宣告地定義、擴充與使用

子類別的系統和方法。本發明提供一種機制讓開發者以一標示語言進行建構、擴充與使用組件。該機制讓開發者不需要知道某一種程式語言；相反的，該機制允許開發者使用某一種周知的標示語言產生組件，如可擴充標示語言(XML)。

例示的電腦環境

第1圖所示為一代表性的電腦計算裝置，用以說明本發明的實施。參照第1圖是一非常基本的組態，典型的電腦計算裝置100包含至少一處理單元102與系統記憶體104。根據明確的電腦組態與電腦計算裝置100的型態，系統記憶體104可能為揮發性記憶體，如隨機存取記憶體(RAM)、非揮發性記憶體，如唯讀記憶體(ROM)、快閃記憶體(flash memory)等、或者為兩種類型記憶體的結合。典型的系統記憶體104包含一作業系統105、一或多個程式模組106、以及可包含程式資料107。程式模組106之實例包含瀏覽器應用程式、財務管理應用程式、文書處理器等相關的程式模組。第1圖中於虛線108內所包含的組件說明此基本組態架構。

電腦計算裝置100也可能包含有其它額外的特色與功能，舉例而言，其它可攜式或非可攜式(removable and/or non-removable)資料儲存裝置，如磁片、光碟或磁帶，亦可能包含在電腦計算裝置100中。第1圖中以可攜式儲存裝置109與非可攜式儲存裝置110說明這些額外的儲存設

備。因此，電腦儲存媒體可能包含揮發性與非揮發性記憶體、可攜式與非可攜式儲存媒體，這些媒體藉由各種不同的方法或技術執行資訊的儲存，如電腦指令、資料結構與程式模組等資訊。系統記憶體 104、可攜式儲存裝置 109、非可攜式儲存裝置 110 即為電腦儲存媒體的實例。電腦儲存媒體包含但不限於隨機存取記憶體(RAM)、唯讀記憶體(ROM)、EEPROM、快閃記憶體或其它記憶體技術、CD-ROM、數位影音光碟(DVD)或其它光學儲存裝置、卡式磁帶、磁帶、磁片等其它磁性儲存設備。此外，電腦儲存媒體並不只限於以上所舉之例子，亦包含任何可以用來存放資訊與可被電腦計算裝置 100 所存取的儲存媒體，而任何電腦儲存媒體為裝置 100 的一部份。電腦計算裝置 100 亦可包含輸入裝置 112，如鍵盤、滑鼠、光學筆、語音輸入裝置、觸控式輸入裝置等等；而亦可包含輸出裝置 114，如顯示器、揚聲器、印表機等等。以上這些設備皆廣為大眾所知，因此不再贅述。

電腦計算裝置 100 亦可能包含通訊連接裝置 116，將可透過網路使裝置 100 與其它電腦計算裝置 118 通訊。通訊連接裝置 116 是一種通訊媒體。典型的通訊媒體可具體化為電腦可讀取指令、資料結構、程式模組、或是其他經調制的資料信號及任何資訊發送媒體。經調制的資料信號有載波或是其他傳輸機制。經調制的資料信號表示可以透過改變其特性將資訊編碼的信號。舉例來說，並非限制，通訊媒體包括有線媒體如有線網路或是直接連線，以及無

線媒體如聲學式媒體、無線電波、紅外線及其他無線媒體。本文中提到的電腦可讀媒體包括儲存媒體及通訊媒體兩者。

例示的實作方式

第2圖為一功能區塊圖，概括說明當執行本發明一實施例所發展之系統。該系統包含一標示編譯器202與一剖析器204，而該標示編譯器202與剖析器204可常駐在一電腦計算裝置如第1圖的電腦計算裝置100上的軟體模組(如第1圖所示之程式模組106)。該標示編譯器202輸入一標示文件206。在一實施例中，此標示文件為以可擴充標示語言(XML)為基礎之文件。簡而言之，該標示文件206包含多個標籤(未示出)用來指示子類別定義的一部份，如第4圖至第6圖所示，其細節並將於以下描述之。之後亦將詳述標籤指示子類別的存在與其相關之元件。一旦標示編譯器202讀到這些標籤便會與剖析器204通訊以建構子類別。

在一實施例中，由剖析器204所提供之功能，可能在標示編譯器202內提供；在另一個實施例中，由剖析器204所提供之功能，可能藉由從標示編譯器202內已存在的剖析類別中衍生一剖析類別而提供。該衍生的剖析類別，可包含根據本發明而定義之每一個子類別符記(token)(如：標籤)之功能撤銷。簡而言之，功能撤銷可作為一系列的回叫功能(callback function)的部分，該等回叫功能係用來傳訊

(signal)和子類別相關聯之元件的定義的開始與結束，如第10圖所示並將於往後詳述之。

剖析器 204 用來剖析在標示文件 206 內的子類別定義。簡而言之，標示編譯器 202 編譯標示文件 206 的內容。在一實施例中，標示編譯器 202 將該內容轉換為符記二元資料流(tokenized binary stream)並將之儲存在符記二元資料檔案(tokenized binary file)208。熟習該項技術者將瞭解該符記二元資料檔案 208 可為許多種形式中的一種。符記二元資料檔案 208 以樹狀結構表示在標示文件 206 內所指定之組件。然而，標示編譯器 202 可能無法直接轉換某些內容，此內容可被傳送至剖析器 204 來轉換。根據本發明之在標示文件 206 內定義之子類別定義即為此種型態內容之實例。剖析器 204 辨識在子類別定義中之特性、事件等...，並且將這些有關項目之相關資訊傳遞給標示編譯器 202。

一旦收到這些相關之資訊，標示編譯器 202 將增加與標示文件 206 內的子類別定義相關聯之符記(token)。標示編譯器 202 亦將產生代表原始碼 212，而由該代表原始碼 212 產生 IL 組合語言 210。IL 組合語言 210 包含定義在標示文件 206 內之子類別(如組件)的電腦指令。過去，係使用一軟體開發工具來產生這些 IL 組合語言，該軟體開發工具係利用一程式語言進行原始程式碼的編譯與連結。在另一實施例中，熟習該項技術者亦將瞭解標示編譯器 202 可在不產生符記二元資料檔案 208 的情況下產生 IL 組合

語言 210。

由標示編譯器 202 所產生之 IL 組合語言 210 可以被傳統之程式開發工具重複使用；除此之外，亦可重複使用於其它標示語言。第 7 圖及其相關敘述說明如何在標示文件內重複使用 IL 組合語言 210。因此，本發明讓組件開發者使用標示語言輕易的建構與擴充組件。一旦根據本發明來建構新組件，此新組件就像是使用傳統程式語言所開發出來的，因此，程式開發者在無需學習一種或多種程式語言的語法或了解語言間細微之差異的情況下，將可使用本發明之機制與方法建構組件。

第 3 圖所示為一功能區塊圖，概括說明為執行本發明一實施所須之執行環境。該執行環境包括一執行引擎(runtime engine)302、一符記二元讀取器(tokenized binary reader)304 以及一符記二元載入器(tokenized binary loader)306。當執行引擎 302 收到載入一特殊資源(如第 2 圖所示的標示文件 206)的請求時，該執行引擎 302 將存取分頁映射表 308。該分頁映射表 308 將辨識該標示文件 206 是否具有一已編譯版本(如符記二元檔案 208)，如果一已編譯版本存在，該執行引擎 302 將與符記二元載入器 306 通訊以載入符記二元檔案 208。在一實施例中，符記二元檔案 208 將辨識與它有關的 IL 組合語言（如 IL 組合語言 210)，隨後符記二元載入器 306 開始下載被辨識的 IL 組合語言 210。一旦部份或全部的符記二元檔案 208 及其相關的 IL 組合語言 210 載入完成，該符記二元讀取器 304 開始

讀取符記二元檔案208及IL組合語言210以產生在處理器(如第1圖所示之處理單元102)上可執行的原生指令。此外，該符記二元讀取器304可存取中介資料(metadata)310以判斷型態、方法與事件之資訊。一般而言，中介資料310包含關於方法、欄位、特性與事件的資訊，而這些項目可能都有屬於自己的中介資料以供讀取更詳細的資訊。因此，使用中介資料310，符記二元讀取器304在執行期間使用反射(reflection)用以程序性的判別在符記二元檔案208內相關元件的資訊。除此之外，原本在標示文件206所定義之子類別，可使用根據本發明所產生之IL組合語言210直接執行，往後將詳述細節。

第4圖到第6圖說明一系列在標示文件206內重要的標示部分，說明根據本發明之實施例用以宣告定義子類別之示範性語法。第10圖係示範性原始碼列表，以供圖示說明該標示編譯器202可根據標示重要部分(第4圖到第6圖所示)而產生的代表性原始碼。在實施例中，當開發者設定偵錯旗幟，標示編譯器202可實際產生一含有代表性原始碼的檔案。該檔案內有代表性原始碼，因此讓開發者可以判斷在標示文件206內文或標示編譯器202內可能之問題。另一個實施例中，該代表性原始碼也許並不存放在檔案中。在這個實施例中，標示編譯器202可在有或沒有事先產生的代表性原始碼的情況下，產生符記二元檔案208與IL組合語言210。

綜觀而言，第4圖到第6圖逐步說明如何根據本發明

在標示文件內宣告地定義一子類別之不同態樣。第4圖所示為一定義子類別與子類別階層的示範性語法；第5圖所示為定義子類別之辨識器、程式碼與建構者的示範性語法；而第6圖所示為定義子類別特性與事件的示範性語法。以下將詳述各圖示的細節。

第4圖所示為一定義子類別與子類別階層的示範性語法。標示400重要的部分(如子類別定義)包括一父類別標籤402(如"Button")。以下所述每個標籤(如父類別標籤402)都有相對應的結束標籤；不過，為了方便，結束標籤不一定明確地以書寫描述，但會在相關圖式中說明。標示400，包括名稱空間屬性404、定義名稱空間的宣告406、以及數個編譯器指示(如指示408和410)。名稱空間屬性404，辨識名稱空間(如"System.Control")及父類別(如"Button")所在之組合語言。定義名稱空間的宣告406，指示任何在有"def:"字首的標示400內的屬性係代表編譯器需執行哪些動作的指示。

舉例而言，指示408(以下稱語言指示408)，指定編譯器去使用指定給語言指示408的程式語言(如C#)來產生IL組合語言，而語言指示408可具有被指定的不同數量的程式語言之任一種語言，如C、C++等相關的程式語言。指示410(以下為def:Class指示410)指定編譯器去使用指派給def:Class指示410的名稱414(如MyButton)來定義子類別。def:Class hint 410可能包含一子類別名稱空間412以用來辨識其中新的子類別關聯的名稱空間(如

“MyControlLib”)。因此，在第 4 圖中，程式開發者宣告定義一個名為“MyButton”的新子類別，其擴充位於“System.Control”名稱空間的“Button”類別；而該新的子類別將與“MyControlLib”的名稱空間相關聯。

第 4 圖所示的標示 400 中，定義一元件宣告標籤 420(如“Image”)。由於在元件宣告標籤 420 中沒有定義特定之名稱空間，因此該名稱空間屬性 404 亦定義了與元件宣告標籤 420 相關聯的元件（如“Image”）的位置。該元件宣告標籤 420 包含一元件 421 與一來源屬性 422；而來源屬性 422 包含一特性 424 (如“Source”)及一數值 426 (如“HappyFace.jpg”)。由於元件宣告標籤 420 係在定義為“MyButton”的子類別之子類別定義中，因此當子類別被樣例化時與元件宣告標籤 420 相關的元件亦將被樣例化。除此之外，與元件宣告標籤 420 相關的元件包含在新的子類別的子收集之中；亦即該子類別為和元件宣告標籤 420 相關元件的母體(parent)。因此，熟知此技術的人將知道標示 400 允許開發者以一階層方式表現而使編譯器產生元件樹狀圖，其係根源於已定義的子類別(如“MyBotton”)。

第 5 圖所示為定義子類別之辨識器、程式碼與建構者的示範性語法。標示 500 重要的部分除包含上述標示 400 外，亦包含定義子類別之辨識器、程式碼與建構者的標示。為了閱讀上的清晰，定義在第 4 圖的參考符號，除非有助於解釋新的概念，其參考符號將不在第 5 圖中顯示。在第 5 圖中，元件宣告標籤 420 更包含其它屬性，如 id 屬性 520

與事件屬性 526。id 屬性 520 包含 id 特性 521(如“ID”)及 id 數值 523(如“img1”)。Id 屬性 520 說明一根據本發明而定義子類別之辨識器的示範性機制。

該事件屬性 526 包括一事件觸發器 527(如“DataLoaded”)與一事件數值 529 (如“OnLoaded”)。事件觸發器 527 說明其受監控之事件，而事件數值 529 說明當事件觸發器 527 發生時所執行之方法。該事件數值 529 與一方法 530(如“OnLoaded”功能)相關聯，而方法 530 可使用一程式語言撰寫。方法 530 可參照定義在標示 500 的類別或子類別之實例。因之，當方法 530 在標示文件內使用一程式語言撰寫，方法 530 將與一程式碼指示 540 相關聯；而該程式碼指示 540 允許該開發者增加片段的程式碼於子類別定義之中。在一實施例中，該程式碼跟隨程式碼指示 540 且為一可呼叫之方法或事件處理器。舉例而言，在標示 500 中，Onloaded 功能 530 視為由 Image 控制所啟動之 DataLoaded 事件的事件處理器。子類別定義主體可再加入其它程式碼片斷，如第 5 圖所示之 CustomInit 功能 550。

標示 500 亦包括建構者指示 542，該建構者指示 542 允許開發者撰寫補充的建構者，用以補充在子類別所產生的預設建構者。在一實施例中，預設的建構者最後才執行補充的建構者，而該補充建構者所包含的程式碼將影響子類別進行建構時的行為。開發者可呼叫在補充建構者內的父類別的建構者。第 5 圖所示補充建構者名之為 CustomInit 功能 550 為使用者所定義的專用方法。

第 6 圖所示為定義子類別特性與事件的示範性語法。標示 600 內重要的部分除包括上述標示 400 與標示 500 外，亦包含定義特性與類別的標示。為了閱讀上的清晰，定義在第 4 圖與第 5 圖的組件符號，除非有助於解釋新的概念，其組件符號將不在第 6 圖中顯示。

標示 600 包含特性指示 610 可於子類別中定義特性(property)，而該特性可作為子類別的構件變數。該特性指示 610 可包含一或多個屬性，如名稱屬性 612、型態屬性 614、預設數值屬性 616 與旗標屬性 618。該名稱屬性 612 具體指明特性的名稱，在一實施例中，名稱有大小寫之分，且執行引擎並不限定使用名稱已使用的字元，但該名稱對於該名稱的擁有者為獨有的。該型態屬性 614 具體指明該特性數值的型態，此型態包含內含的(intrinsic)、使用者定義的(user-defined)、結構(struct)、(class)、介面(interface)、數字(enum)，等等...。預設數值屬性 616 具體指明當特性被樣例化時所指定的數值。該旗標屬性 618 具體指明當特性被樣例化時，包裝(wrapper)所產生的方法型態。該旗標具有控制特性特質之能力，如 ReadOnly、Inheritable to child elements、Private，等等...。

標示 600 亦包括事件指示 620，該事件指示 620 允許在子類別中定義事件。該事件指示 620 包括多個屬性，如名稱屬性 622、路由屬性 624、旗幟屬性 626，等等...。名稱屬性 622 具體指明關聯該事件的字串，當一 xml 檔案使用該子類別，該 xml 檔案將使用此字串以附加適當的應用

程式開發者所定義的程式碼。路由屬性 624 具體指明判別於樹狀結構中啟動事件之組件的方法。旗幟屬性 626 具體說明其它與事件有關的特性，舉例而言，在標示 600 中，標示編譯器將產生啟動一事件呼叫 DblClick 的程式碼以及包含與該事件相關的資訊，如支援路由、旗標、...等等有關資訊。熟知此技術的開發者將可在不脫離本發明的範疇下，進行屬性相關名稱與數值的修改。

標示 600 亦包含一事件處理器宣告 630。事件處理器宣告 630 包含相關的事件處理器修改(event handler modifier) 632，如公共的/專用的、代表、返回型態等等...。在第 6 圖中，事件處理器宣告 630 宣告一事件處理器(亦即方法)(如“DblClickEventHandler”)，當相關的事件發生時，該事件處理器將被呼叫。

標示文件內亦定義其它的標籤。舉例而言，根標籤(如“def:Library”)可用於通知編譯器及/或剖析器於另外的檔案之中定義子類別、在同一檔案中定義以其它子類別定義該子類別等等。一名稱空間可被宣告來定義一個訂製的組件、所有在根標籤下定義的類別,等等...。

第 7 圖所示為在標示文件內重要的部分，並說明使用於標示文件內的子類別之示範性語法。標示 700 包含一根元件 702(如 FlowPanel 元件)、一預設名稱空間宣告 704、定義名稱空間前置字串 706 與一元件 708。元件 708 可參照根據以上語法所建構之客制化元件。該預設名稱空間宣告 704 說明標籤之預設名稱空間而不需要前置字串。在舉

例的標示 700 之中，預設名稱空間宣告 704 指的是“System.Controls”名稱空間。該定義名稱空間前置字串 706 具體指明客制化元件與元件所存在的位置。在舉例的標示 700 之中，定義名稱空間前置字串 706 指的是“MyControlLib”。該元件 708 包含一類別名稱 710（如“MyControlLib”）、一辨識器 712 與文字字串 714。該類別名稱 710 指得是針對該元件而樣例化的類別，而該辨識器 712 指得是該元件且包含一名稱(如“ID”)與一數值(如“button1”)。在程式執行期間，該數值成為類別名稱 710 實例的名稱(如“button1”)。

例示的實作方式之一般操作流程

第 8 圖為一邏輯流程圖，主要根據本發明之實施例，概括說明進行編譯宣告定義子類別之處理流程。第 8 圖所示之流程 800 將根據第 4 圖至第 6 圖所示之示範性之標示並配合第 10 圖所示之代表原始碼以進行解說。當比較第 10 圖與標示 600 後會立即清楚的發現，本發明可以宣告性的方式產生複雜的子類別，開發者無須了解其使用之程式語言，如第 10 圖中之代表原始碼為 C#之原始碼。然而，根據本發明所指之標示編譯器將可根據任何一種程式語言之語法產生代表性程式碼。

區塊 801 為流程 800 的起始區塊，在此區塊中，標示編譯器正編譯一標示文件並遭遇具有子類別定義之標示。該標示編譯器將透過以下機制來判斷是否讀到一子類別定

義，例如，該標示有無法辨識的格式或者是辨識到一子類別標籤(如 def:Class)等等相關方式。舉例而言，在標示 400 中，標示編譯器在處理過數個敘述句後，才遇到用來辨識子類別定義之子類別標籤(如 def:Class 指示 410)。在其它實施例中，子類別可能在許多敘述句之前就出現，例如子類別沒有具體指明一基礎類別時(如 button)。一般而言，子類別標籤可能出現在標示內組件標籤會出現的任何地方。一旦標示被辨識為具有子類別定義，處理程序將進入區塊 802 內。

在區塊 802 中，將開始處理子類別標籤。根據第 4 圖所描述，該子類別標籤(也就是 def:Class 指示 410)指定一名稱 414，並且可包含一子類別名稱空間 412。根據此資訊，第 10 圖第一行所示之代表程式碼可能為組合語言而產生。除此之外，擁有指定名稱的類別亦將被產生，如第 10 圖第 5 行所示。當其它額外敘述(如名稱空間與父類別)已被處理，第 5 行的將以其他產生的程式碼擴充。在處理子類別標籤部分，該流程將辨識該子類別是否擴充自任何類別(即父類別)並得到該關聯父類別的相關資訊。在第 4 圖中，“MyBotton”子類別的產生擴充自父類別標籤 402 所定義“Button”。因之，第 10 圖第 5 行所示為 MyBotton 擴充自 Botton 的代表性程式碼。

除此之外，子類別所預設之建構者也將被產生。再次重申，一旦處理額外的敘述，該預設建構者可以額外代表性程式碼擴充，如第 10 圖第 23 行至第 24 行與第 26 行所

示對應所產生的代表性程式碼。然而，"this._Initialize_This()"函式將在其它以下將介紹之敘述被處理完後加入。子類別標籤一旦被處理，處理程序將進入區塊 804。

在區塊 804 中，將擷取出下一個在標示中的敘述。下一敘述，可能出現在子類別標籤之前或之後的敘述，其出現的位置端看子類別標籤敘述在子類別定義內出現的位置。一旦下一個敘述被擷取，處理程序將進入決策區塊 806。

在決策區塊 806 中，主要判斷該敘述是否定義一類別。就如之前曾提到，本發明允許宣告性的表示階層架構。在一實施例中，定義該子類別之子的另一類別(如元件)的敘述將在子類別標籤之後發生。在此，假設目前的敘述並沒有定義一類別，處理程序將進入區塊 808。

在區塊 808 中，該敘述將被處理。在子類別定義中，將會有各種型態的敘述，它們並不依照一定的順序出現。每種型態敘述的處理方式將於以下詳述之。一般而言，當每一個敘述被處理後，將在子類別定義中產生額外的代表性程式碼。一旦這些敘述之一在區塊 808 被處理，處理程序將進入決策區塊 810。

在決策區塊 810 中，主要判斷在子類別中是否還有任何敘述要被處理。假設還有其他必須處理的敘述，處理迴路將回到 804 與 808 之間的區塊處理該敘述。一旦在子類別定義內的所有敘述處理完畢，處理程序繼續由決策區塊

810 進入區塊 812。

區塊 812 中，由以上程序所產生的代表性程式碼將被用來產生子類別定義的 IL 組合語言。程式開發者可能當產生代表性程式碼時已經在某一編譯指示敘述中（區塊 820）指定程式語言，可能指定哪一個組合語言檔案用來儲存子類別定義等等...。一旦一或多個 IL 組合語言產生後，處理流程便完成。如同之前曾經提及的，上述產生的組合語言將可使用傳統的程式語言來執行或使用根據本發明所示之標示敘述等等。而該組合語言就像是用一程式語言撰寫而成。

回到決策區塊 806，假設該敘述開始一新類別的定義(如組件)，處理程序將進入區塊 814。在區塊 814 中，所有屬於新類別的敘述將使用在區塊 808 內的方式進行處理，直到所有屬於新類別的敘述被處理完畢。舉例而言，在第 5 圖中，於新類別“Image”的敘述 520、526 與 422 將被處理，之後，Image 成為子類別“MyButton”的一子。程序區塊 804 繼續執行與該子類別有關的處理敘述。

接著，再回到區塊 808，描述處理每一種可能出現在子類別的敘述之個別型態（如區塊 820-832 所示）。在區塊 820 中，編譯指示的敘述將被處理。一般而言，這些敘述為編譯指示用以提供標示編譯器關於如何產生組合語言的相關資訊。舉例而言，在第 4 圖中，語言指示 408 為編譯指示敘述，用來通知當產生代表性程式碼時，標示編譯器應該使用之程式語言。在第 4 圖中，語言指示 408 指定語

言為 C#，因此，第 10 圖所示代表性程式碼為 C#原始程式碼。

在區塊 822 中，將處理有關定義名稱空間的敘述。這些名稱空間將包含於代表性程式碼中，如第 10 圖第 2 行與第 3 行所示。

在區塊 824 中，將處理定義一類別之辨識器(id)的敘述。對於任何在標示文件內的 id 屬性，該標示編譯器產生相關類別的欄位構件。所產生的欄位構件有其型態，該型態與定義 id 屬性的類別相同，並且該欄位構件的名稱即是指派給 id 屬性的 id 數值。舉例而言，在第 5 圖中，當開始處理 Image 類別的敘述時，將遇到 id 屬性 520；因此，如第 10 圖第 7 行所示，標示編譯器將產生 MyBotton 類別的代表性程式碼，而其具有一欄位構件定義為：

```
private System.Control.Image img1;
```

在程式執行期間，該欄位構件將被起始為 InitComponent()方法產生的相對應型態(如 Image)的實例。因此，任何在 MyBotton 類別的程式碼可以透過類別 id 數值存取在階層架構中的元件實例。起始的欄位構件如第 10 圖第 37 行所示。

在區塊 826 中，有關定義類別特性的敘述將被處理。標示編譯器將產生所有定義在標示文件內特性的代表原始碼，並且註冊該特性。舉例而言，第 6 圖中，標示 600 包

含特性指示 610，該特性指示 610 有 612-618 所示多種不同的屬性。該特性指示 610 用以通知標示編譯器其特性指示 610 之後的敘述為關於特性的敘述，接著 612-618 屬性將被讀取為特性敘述。該標示編譯器將產生"Label"特性的代表程式碼以註冊為 MyBotton 類別的屬性，如第 10 圖第 9 行至第 12 行所示。該標示編譯器亦將產生定義該特性的代表性程式碼，如第 10 圖第 28 行至第 30 行所示。第 28 行至第 30 行說明名稱屬性 612 如何成為於代表性程式碼中特性之名稱，並說明型態屬性 614 如何成為於代表性程式碼中特性之型態。

在一實施例中，該流程藉由呼叫特性型態 TypeConverter 產生特性數值的程式碼，其利用反射(reflection)執行程式碼並且轉換字串成為該特性類別的物件實體。於另一實施例中，為了得到實際的物件數值流程 800 將呼叫特性的型態轉換器並且將數值轉換為 InstanceDescriptor 物件。該 InstanceDescriptor 物件包含充分的資訊，使得標示編譯器能夠反射該物件以產生代表性程式碼，而該代表性程式碼將產生與特性指示相關屬性所指定的數值型態的新實例。在第 10 圖第 28 行至 30 行中說明在程式執行期間，當 Label 特性有指派數值情況下，LabelProperty 如何在設定於 MyBotton 實例之中。該指定數值可能以宣告形式於標示中指定（如第 7 圖所示）或者使用任何程式語言程式指定。

在區塊 828 中，將處理定義類別事件的敘述。標示編

譯器將根據事件敘述產生該事件的代表程式碼。事件可能使用不同機制於標示中定義，如第5圖所示之事件屬性526與第6圖所示之事件指示620。首先，該機制使用的事件屬性如下所描述，事件屬性包含事件觸發器與事件值。當事件觸發器發生時，事件相關的事件觸發器及與方法相關的事件值將開始啟動執行。當定義"this.OnLoad"為新的System.Controls.DataLoadedEventHandler時，第5圖中的事件值529("OnLoaded")將以事件處理器增加至代表程式碼，如第10圖第38行至40行所示。藉由定義AddHandler的第一個參數為System.Controls.Image.DataLoadedEvent，事件觸發器527("DataLoaded")將增加為第10圖第38行代表程式碼的事件。AddHandler的其它參數可能使用預設值或已經在標示文件內具體指明。

以下將描述使用事件指示620的機制。事件指示620包含一事件名稱屬性與其它屬性，而指派給事件名稱屬性的名稱被註冊為該事件。舉例而言，參考第6圖所示，"DblClick"為指派給事件名稱屬性的名稱；因此，"DblClick"為RegisterEvent方法的第一個參數，如第10圖第14行至第16行所示。而屬於方法"DblClickEventHdlr"之事件處理器宣告630的型態亦將包含為RegisterEvent方法的一個參數，如第10圖所示。再者，RegisterEvent方法的其它參數可能使用預設的參數值，亦可能為在標示文件中具體指明的其它屬性。根據實施例所示，在程式執行期間，該事件將被連結使用反射以辨識

DblClickEventHandler 型態並配置該方法。

於區塊 830 中，定義程式碼的敘述將被處理。定義程式碼的敘述跟隨一編譯器指示敘述(區塊 820)，如程式碼指示。這些敘述以一程式語言撰寫並以代表性程式碼顯示。舉例而言，第 5 圖說明方法 530 與 CustomInite 功能 550。方法 530 與 CustomInite 功能 550 分別顯示在第 10 圖第 18 行至第 19 行與第 21 行。

在區塊 832 中，定義輔助性建構者的敘述將被處理。重複之前所言，定義輔助性建構者的敘述將跟隨編譯器指示敘述(區塊 820)，如建構者指示。標示編譯器將產生一輔助性建構者("this._Initialize_This()"於第 10 圖中)並且在輔助性建構者內增加敘述。舉例而言，第 5 圖中在建構者指示 542 之後的敘述稱之為"CustomInit()"。因此，標示編譯器增加"CustomInit()"於代表性原始碼之輔助性建構者中，如第 10 圖第 33 行所示。

在不脫離本發明的範疇之下，熟知此技術的人將知道，當執行區塊 804 至 810 期間，區塊 812 可能以漸增方式運作執行。除此之外，根據剖析器 204 與標示編譯器 202 的功能性實施，處理流程 800 期間也許有回叫在剖析器 204 與標示編譯器 202 之間。

第 9 圖為一邏輯流程圖，概括說明根據本發明之實施例使用在標示文件中的子類別宣告之一執行程序 900。第 7 圖所示的示範性標示將與第 9 圖搭配描述流程 900。區塊 901 為流程 900 的起始區塊；在區塊 901 中，執行引擎將

接收特殊資源(如標示文件)的請求並且判定該特殊資源存在一編譯版本。舉例而言，假設第7圖中標示700事先並沒有被編譯為符記二元檔案，因此，當標示編譯器讀到元件708，該標示編譯器將判斷該MyBotton類別具有相關的符記二元檔案與IL組合語言。包含MyBotton類別的符記屬性值之相關符記二元檔案將於流程900中被處理。相關的符記二元檔案與IL組合語言將根據本發明產生，而處理程序將進入區塊902。

在區塊902中，符記二元檔案被載入，該符記二元檔案可一次完整載入也可能以漸增方式被載入。符記二元檔案可辨識與需要被載入的符記二元檔案相關的IL組合語言。參考第7圖所示，例如與Mybotton類別相關的IL組合語言被載入，而處理程序進入區塊904。

在區塊904中，由符記二元檔案中擷取一符記(token)。如前所述，在新的執行環境中，所產生的符記二元檔案與用於產生該符記二元檔案之程式語言的關係是獨立的。因此，執行引擎將可在不需要知道原來產生該符記二元檔案的程式語言情況下處理該符記二元檔案。處理程序繼續進入決策區塊906。

在決策區塊906中，將判別所擷取的符記(token)是否為一事件，若該符記不為一事件，處理程序進入區塊908。

在區塊908中，該符記(token)將被處理。如同之前所述，符記(token)的處理並不需依賴符記二元檔案的產生方式。換句話說，不論由標示文件宣告所產生的符記二元檔

案之符記或使用程式語言所產生的符記，該符記處理皆使用相同方式。由於對於熟知此技術的開發者皆知道的符記二元檔案內處理符記(token)方式，在此不再贅述處理方式。處理程序進入決策區塊 910。

在區塊 910 中，將判斷在符記二元檔案中是否還有更多符記(token)。假設還有其它符記(token)，處理迴路將回到 904 並且依照之前所說的方式處理；反之，如果沒有其它的符記(token)，則處理流程完成並進入結束區塊。

回到決策區塊 906，若該符記(token)為一事件，處理程序繼續進入區塊 912。在區塊 912 中，與符記二元檔案關聯的中介資料(metadata)被載入。該中介資料包含型態、資源、方法等相關描述；接著，處理程序繼續進入區塊 914。

在區塊 914 中，處理程序使用中介資料與反射發現事件之目標元件型態。該程序包括存取中介資料並且檢視每一欄位以判別型態；接著，處理程序進入區塊 916。

在區塊 916 中，目標元件型態的事件被驗證；如此可確保該事件為一有效事件。假設對目標元件型態而言該事件不為一有效的事件，則發生錯誤。處理程序進入區塊 918。

在區塊 918 中，反射被用於目標元件型態用來得到事件方法，然後執行該事件方法，而執行該事件方法包括在關聯的 IL 組合語言內的執行程式碼。因之，參照第 7 圖所示，一旦 MyBotton 被樣例化後，將找到該事件，並將附加該事件方法(“DblClickEvent”)。亦即一事件處理器(如

DblClickEventHandler)被附加於該事件(如 DblClick)。處理程序將進入決策區塊 910 並且依照之前所述之方式處理。

因此，本發明提供一種在標示文件中宣告地定義子類別與使用該子類別的機制。該機制讓開發者專注在更多組件的使用方式而不是擔心如何在任何一種程式語言中執行組件。

根據上述的說明書，我們以實例與資料提供完整製造與使用本發明的描述。本發明之諸多實施例可在不脫離本發明精神與範圍下完成，本發明之範圍依隨附之申請專利範圍而定。

【圖式簡單說明】

第 1 圖所示為一示範計算裝置，用以說明本發明的實施。

第 2 圖所示為一功能區塊圖，概括說明執行本發明一實施例所須之組件。

第 3 圖所示為一功能區塊圖，概括說明為執行本發明一實施所須之執行環境。

第 4 圖到第 6 圖根據本發明說明一系列在標示文件內重要的部分，該標示文件以示範性語法說明宣告地定義子類別。

第 7 圖所示為在標示文件內重要的部分，並以示範性語法說明如何使用於標示文件內的可執行組件。

第 8 圖為一邏輯流程圖，主要根據本發明之實施例，概括說明進行編譯宣告定義子類別所需之處理流程。

第 9 圖為一邏輯流程圖，根據本發明之實施例概括說明一執行期間的程序，使用在標示文件中宣告的子類別。

第 10 圖所列為示範性原始碼，為標示編譯器(如第 2 圖所示)根據標示文件(第 4 圖到第 6 圖所示)而產生代表性原始碼。

【元件代表符號簡單說明】

100	電腦計算裝置	102	處理單元
104	系統記憶體	105	作業系統
106	程式模組	107	程式資料
108	構成電腦基本組態之組件		
109	可攜式儲存裝置		
110	非可攜式儲存裝置	112	輸入裝置
114	輸出裝置	116	通訊連接裝置
118	其它電腦計算裝置	202	標示編譯器
204	剖析器	206	標示文件
208	符記二元資料檔案	210	IL 組合語言
212	代表原始碼	302	執行引擎
304	符記二元讀取器	306	符記二元載入器
308	分頁映射表	310	中介資料
400	標示	402	父類別標籤
404	名稱空間屬性	406	定義名稱空間的宣告

408 編譯器指示　410 編譯器指示

412 子類別名稱空間　414 名稱

420 元件宣告標籤　421 元件

422 來源屬性　424 屬性

426 屬性數值　520 id 屬性

521 id 特性　523 id 數值

526 事件屬性　527 事件觸發器

529 事件數值　530 方法

540 程式碼指示　542 建構者指示

550 輔助性建構者

610 特性指示　612 名稱屬性

614 型態屬性　616 預設屬性

618 旗標屬性　620 事件指示

622 名稱屬性　624 路由屬性

626 旗幟屬性　630 事件處理器宣告

632 事件處理器修改　702 根元件

704 預設名稱空間宣告　706 名稱空間前置字串

708 元件　710 類別名稱

712 辨識器　714 文字字串

801-917 流程區塊

1000 代表性程式原始碼

五、中文發明摘要：

本發明揭露一種使可執行組合語言(executable assembly)能夠與標示文件(markup document)內的子類別(subclass)定義所產生的子類別相關聯之系統、方法與資料結構。根據本發明，該子類別定義係根據一架構(schema)而撰寫，而該架構可為以可擴充標示語言為基礎的(XML-based)。該架構包含用於定義子類別名稱的子類別標籤。當執行該可執行組合語言時，則該名稱與被樣例化(instantiated)物件之種類相關聯。該架構更包含一或多個指示(hint)以供，如具體指明用以編譯子類別定義的程式語言，具體指明用以衍生出子類別的父類別(superclass)，具體指明當物件被樣例化時應該執行的動作，產生子類別之事件定義與事件處理器，以及具體指明當物件被樣例化時成為物件內構件(member)之一特性(property)。

六、英文發明摘要：

Described is a system, method and data structure that enables an executable assembly associated with a subclass to be generated from a subclass definition within a markup document. In accordance with the invention, the subclass definition is written based on a schema. The schema may be XML-based. The schema includes a subclass tag for defining a name for the subclass. The name is associated with a type for an object that is instantiated when the executable assembly executes. The schema further includes one or more hints, such as for specifying a program language to compile the subclass definition, for specifying a superclass from which the subclass derives, for specifying actions to perform when the object becomes instantiated, for creating an event definition and event handler for the subclass, and for specifying a property that becomes a member within the object when the object is instantiated.

99年1月21日修正本

十、申請專利範圍：

1. 一種具有由一電腦可讀取資料結構所編碼的一明確組件(tangible component)之電腦可讀取媒體，該資料結構包含：

一架構(schema)，用於在一標示文件內宣告性地定義一子類別定義；該架構能夠被編譯至一可執行組合語言(executable assembly)內，該可執行組合語言係依據該組合語言的執行來樣例化(instantiate)一與定義於該子類定義內的子類別相關聯的物件；其中該可執行組合語言係獨立於原本被用以產生該組合語言之程式語言，使得該可執行組合語言係無須考慮該程式語言即可被編譯。

2. 如申請專利範圍第1項所述之電腦可讀取媒體，其中該架構為一種以可擴充標示語言(XML)為基礎的架構。

3. 如申請專利範圍第1項所述之電腦可讀取媒體，其中該架構包含用於定義一名稱的子類別標籤，該名稱與該成為樣例化之物件的型態相關聯。

4. 如申請專利範圍第1項所述之電腦可讀取媒體，其中該架構包含一語言指示器，以供具體指明當編譯該子類別定義時所使用的程式語言。

5. 如申請專利範圍第4項所述之電腦可讀取媒體，其中該

架構進一步包含一程式碼指示器，以供描述該標示文件內之程式碼，且該程式碼係直接被編譯至該可執行組合語言內。

6. 如申請專利範圍第 5 項所述之電腦可讀取媒體，其中該程式碼，係直接根據該語言指示器所具體指明的程式語言的語法而被編譯。

7. 如申請專利範圍第 1 項所述之電腦可讀取媒體，其中該架構包含一父類別標籤，以供具體指明衍生該子類別的父類別。

8. 如申請專利範圍第 7 項所述之電腦可讀取媒體，其中該架構包含一父類別名稱空間(name space)屬性，以供具體指明該父類別所存在之名稱空間。

9. 如申請專利範圍第 1 項所述之電腦可讀取媒體，其中該架構包含一子類別標籤，以供具體指明該子類別。

10. 如申請專利範圍第 1 項所述之電腦可讀取媒體，其中該架構包含一建構者指示器，以供具體指明當樣例化該物件時，應執行之一或多種輔助性動作。

11. 如申請專利範圍第 1 項所述之電腦可讀取媒體，其中

該架構包含一事件指示器，以供具體指明一事件與一對應事件處理器，當樣例化時其係與該物件相關聯。

12. 如申請專利範圍第 1 項所述之電腦可讀取媒體，其中該架構包含一特性指示器，以供具體指明一特性，當該物件被樣例化時該特性係成為該物件內的構件。

13. 如申請專利範圍第 1 項所述之電腦可讀取媒體，其中該架構包含一元件(element)宣告，以供具體指明該子類別的子元件。

14. 一種具有一明確組件之電腦可讀取媒體，該明確組件包含用以產生子類別之可執行組合語言的指令，該等指令包含：

辨識一標示文件(markup document)內之一子類別定義，該子類別定義係定義一子類別：及

根據該子類別定義來產生一組合語言，該組合語言係可執行以產生與該子類別相關聯之一物件之實例(instance)；其中該產生的組合語言係獨立於原本被用以產生該組合語言之程式語言，使得該產生的組合語言係無須相關於該程式語言即可被執行。

15. 如申請專利範圍第 14 項所述之電腦可讀取媒體，其中辨識該子類別定義，係包含剖析一子類別標籤，該子類別

標籤包含用以定義一名稱之子類別名稱屬性，該名稱與成為該樣例化之物件的型態相關聯。

16. 如申請專利範圍第 14 項所述之電腦可讀取媒體，其中根據該子類別定義來產生一組合語言，係包含將該子類別定義加以編譯至代表性原始碼內，該代表性原始碼係被編譯至該組合語言內。

17. 如申請專利範圍第 16 項所述之電腦可讀取媒體，其中該代表性原始碼係根據一程式語言。

18. 如申請專利範圍第 14 項所述之電腦可讀取媒體，其中根據該子類別定義來產生一組合語言，係包含將在該子類別定義內的程式碼編譯至該組合語言內，而該程式碼係藉由一程式碼指示器在子類別定義內加以描述。

19. 如申請專利範圍第 14 項所述之電腦可讀取媒體，其中根據該子類別定義來產生一組合語言，係包含：剖析一用以定義一父類別之父類別標籤，該子類別係由該父類別所衍生；利用與一程式語言相關聯的語法來具體指明一繼承；以及，將該語法編譯為該組合語言的一部份。

20. 如申請專利範圍第 14 項所述之電腦可讀取媒體，其中根據該子類別定義來產生一組合語言，係包含剖析一用以

定義至少一輔助性動作之建構者指示器，該輔助性動作係在該物件被樣例化時被執行，該輔助性動作係該組合語言的一部份。

21. 如申請專利範圍第 14 項所述之電腦可讀取媒體，其中根據該子類別定義來產生一組合語言，包含剖析一事件指示器，該事件指示器係產生該子類別之一事件定義與一對應事件處理器。

22. 如申請專利範圍第 14 項所述之電腦可讀取媒體，其中根據該子類別定義來產生一組合語言，係包含剖析一用以定義一特性之特性指示器；當該物件被樣例化時，該特性係成為該物件內的構件。

23. 如申請專利範圍第 14 項所述之電腦可讀取媒體，其中根據該子類別定義來產生一組合語言，係包含剖析一元件宣告標籤，該元件宣告標籤係辨識該子類別之一子元件。

24. 一種產生一可執行組合語言的電腦實施方法，該方法係至少部分藉由一計算裝置來加以實行，該方法包含：

辨識於一標示文件內之一子類別定義，該子類別定義係定意一子類別；及

根據該子類別定義來產生一組合語言，該組合語言係可執行以產生與該子類別相關聯之一物件之實例，其中該產

生的組合語言係獨立於原本被用以產生該組合語言之程式語言，使得該產生的組合語言係無須考慮該程式語言即可被執行。

25. 如申請專利範圍第 24 項所述之電腦實施方法，其中辨識該子類別定義，係包含剖析一子類別標籤，該子類別標籤包含用以定義一名稱之子類別名稱屬性，該名稱與成為樣例化之物件的型態相關聯。

26. 如申請專利範圍第 24 項所述之電腦實施方法，其中根據該子類別定義來產生一組合語言，係包含：剖析一用以定義一父類別之父類別標籤，該子類別係由該父類別所衍生；使用依據一程式語言的語法來具體指明一繼承；以及，將該語法編譯為該組合語言的一部份。

27. 如申請專利範圍第 24 項所述之電腦實施方法，其中根據該子類別定義來產生一組合語言，係包含剖析一用以定義至少一輔助性動作之建構者指示器，該輔助性動作係在該物件被樣例化時被執行，該輔助性動作係該組合語言的一部份。

28. 如申請專利範圍第 24 項所述之電腦實施方法，其中根據該子類別來定義產生一組合語言，係包含剖析一事件指示器，該事件指示器係用來產生該子類別之一事件定義與

一對應事件處理器。

29. 如申請專利範圍第 24 項所述之電腦實施方法，其中上述之根據該子類別定義來產生一組合語言，係包含剖析一用來定義一特性之特性指示器；當該物件被樣例化時，該特性係成為物件內的構件。

30. 一種用於由一標示文件之子類別定義而產生一組合語言之電腦系統，該電腦系統包含：

一處理器；及

一記憶體，該記憶體係配置複數個被載入該記憶體中的電腦可執行指令，

其中，該處理器係執行該等電腦可執行指令，以供：

辨識一標示文件內之一子類別定義，該子類別定義係定義一子類別：及

根據該子類別定義來產生一組合語言，該組合語言係可執行以產生與該子類別相關聯之一物件之實例，其中該產生的組合語言係獨立於原本被用以產生該組合語言之程式語言，使得該產生的組合語言係無須考慮該程式語言即可被執行。

31. 如申請專利範圍第 30 項所述之電腦系統，其中辨識該子類別定義，係包含剖析一子類別標籤，該子類別標籤包含用以定義一名稱之子類別名稱屬性，該名稱與成為樣例

化之該物件的型態相關聯。

32. 申請專利範圍第30項所述之電腦系統，其中根據該子類別定義來產生一組合語言，係包含剖析一用以定義至少一輔助性動作之建構者指示器，該輔助性動作係在該物件被樣例化時被執行，該輔助性動作係該組合語言的一部份。

33. 如申請專利範圍第30項所述之電腦系統，其中根據該子類別定義來產生一組合語言，係包含剖析一事件指示器，該事件指示器係用來產生該子類別之一事件定義與一對應事件處理器。

34. 如申請專利範圍第30項所述之電腦系統，其中根據該子類別定義來產生一組合語言，係包含剖析一用來定義一特性之特性指示；當該物件被樣例化時，該特性係成為物件內的構件。

35. 一種產生一可執行組合語言的系統，該系統包含：

一用以辨識一標示文件內之一子類別定義的機構，該子類別定義係定義一子類別；及

一用以根據該子類別定義來產生一組合語言的機構，該組合語言係可執行以產生與該子類別相關聯之一物件之實例，其中該產生的組合語言係獨立於原本被用以產生該組合語言之程式語言，使得該產生的組合語言係無須考慮該

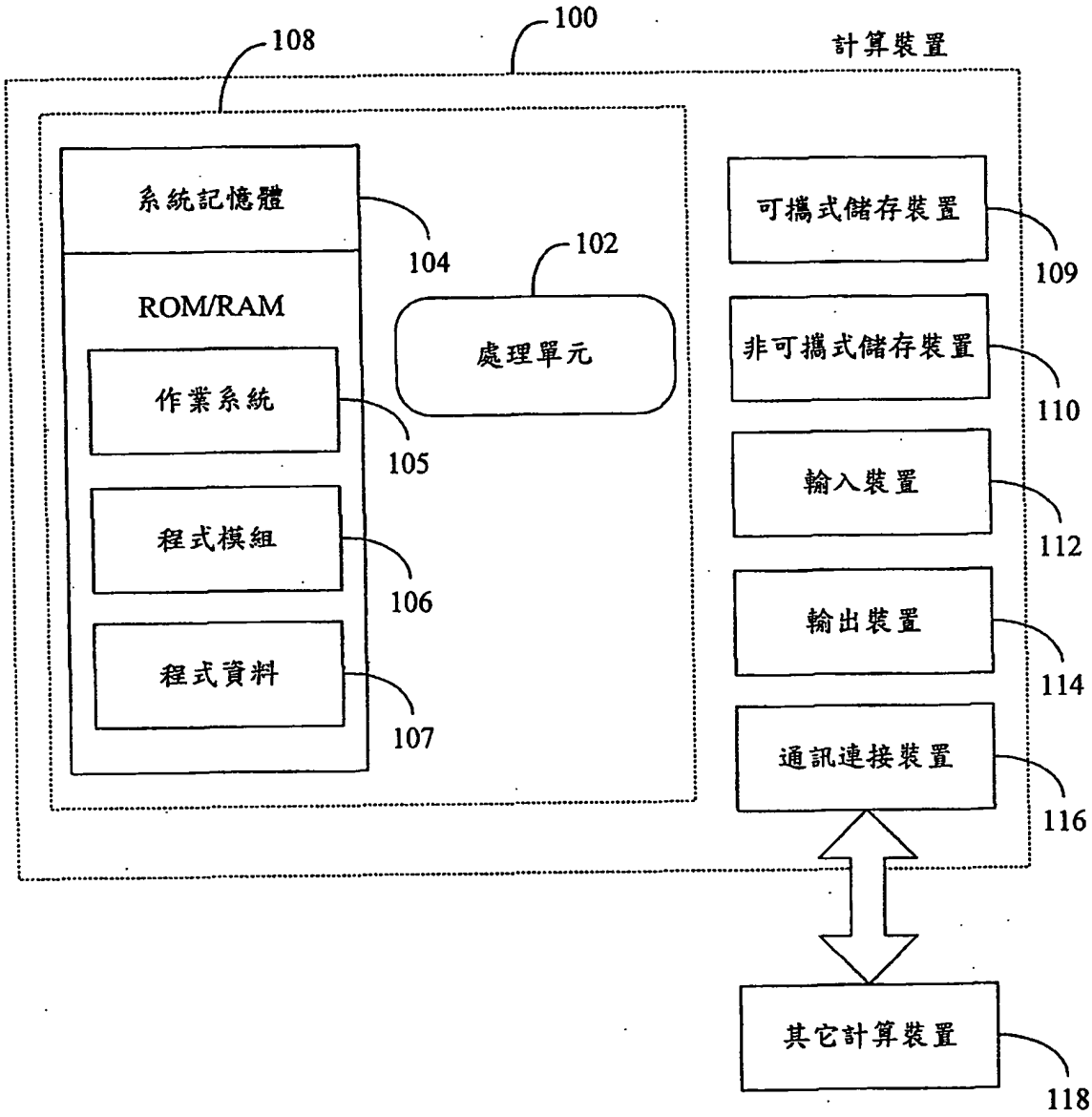
程式語言即可被執行。

36. 如申請專利範圍第35項所述之系統，其中用以辨識該子類別定義的機構，係包含剖析一子類別標籤的機構，該子類別標籤包含用以定義一名稱之子類別名稱屬性，該名稱與成為樣例化之該物件的型態相關聯。

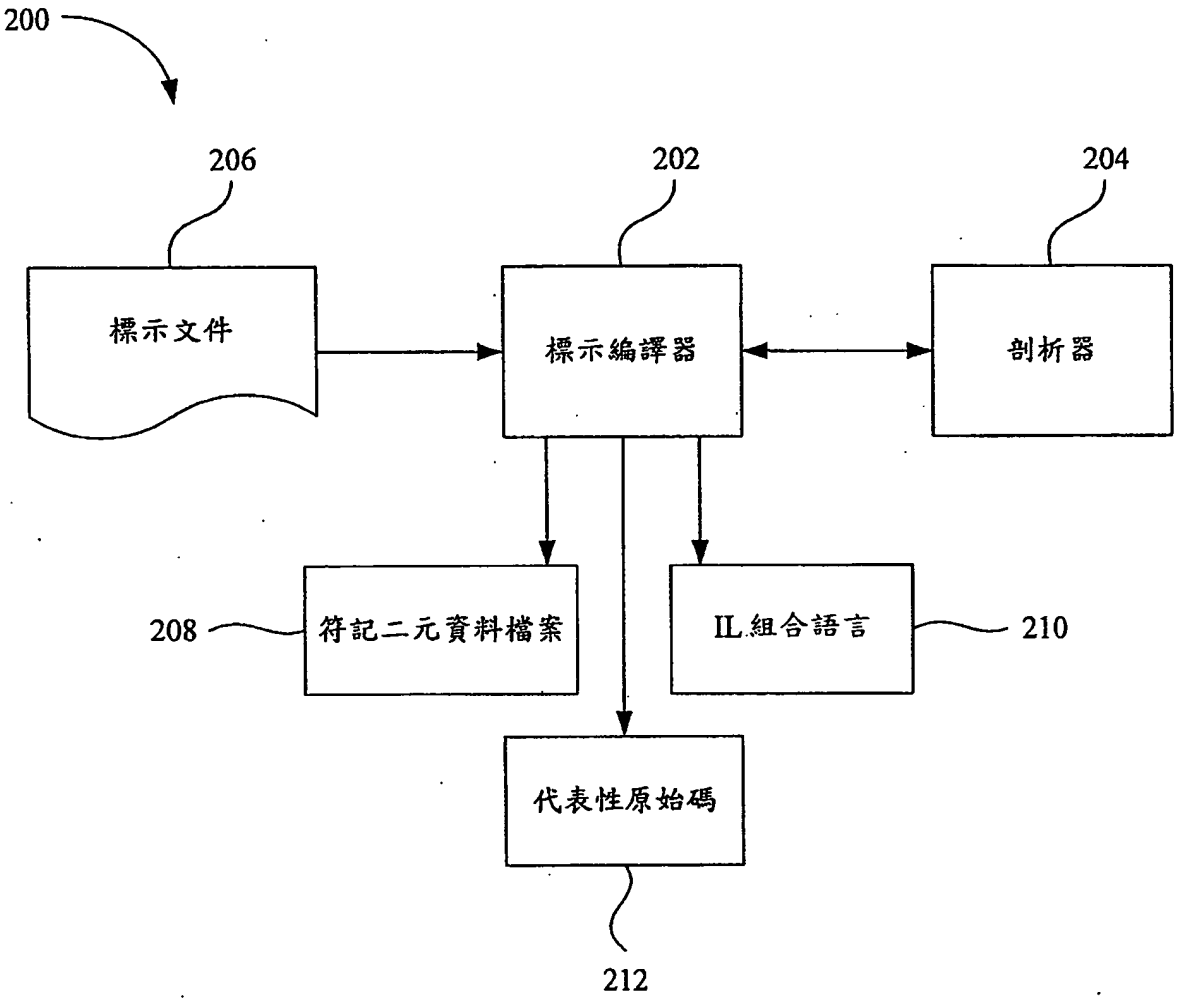
37. 如申請專利範圍第35項所述之系統，其中用以根據該子類別定義來產生一組合語言的機構，係包含剖析一用以定義一父類別之父類別標籤的機構，該子類別係由該父類別所衍生；使用依據一程式語言的語法來具體指明一繼承的機構；以及，將該語法編譯為該組合語言的一部份的機構。

38. 如申請專利範圍第35項所述之系統，其中根據該子類別定義來產生一組合語言的機構，係包含一剖析用以定義至少一輔助性動作之一建構者指示器的機構，該輔助性動作係在物件被樣例化時被執行，該輔助性動係該組合語言的一部份。

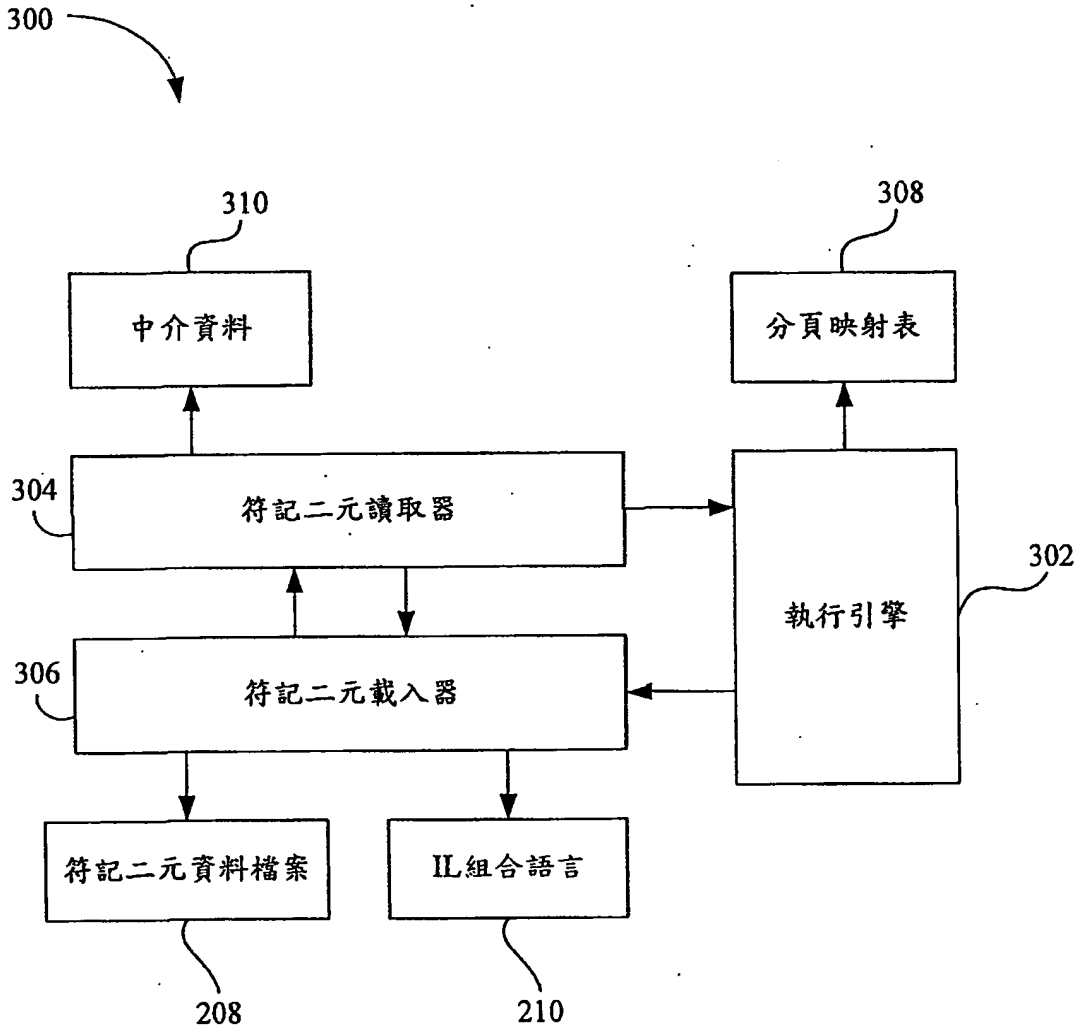
99年1月21日修正本



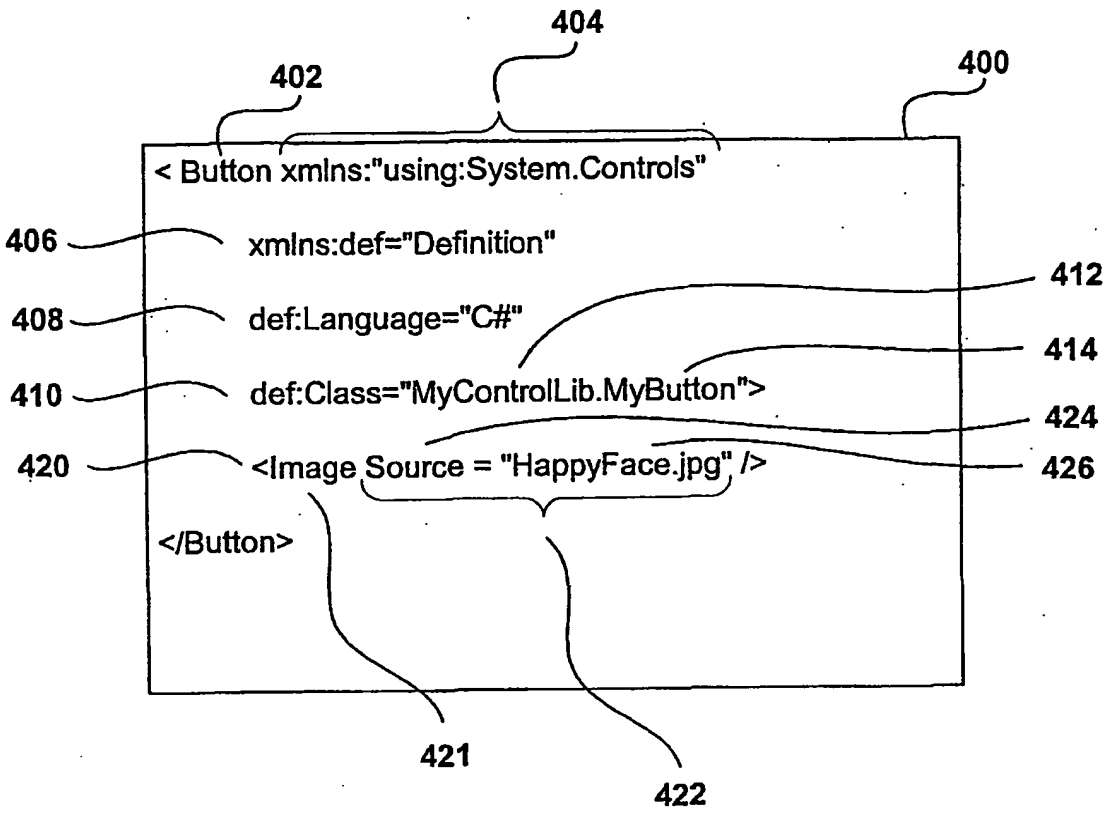
第 1 圖


200
206
202
204
標示文件
標示編譯器
剖析器
208
符記二元資料檔案
IL組合語言
210
代表性原始碼
212

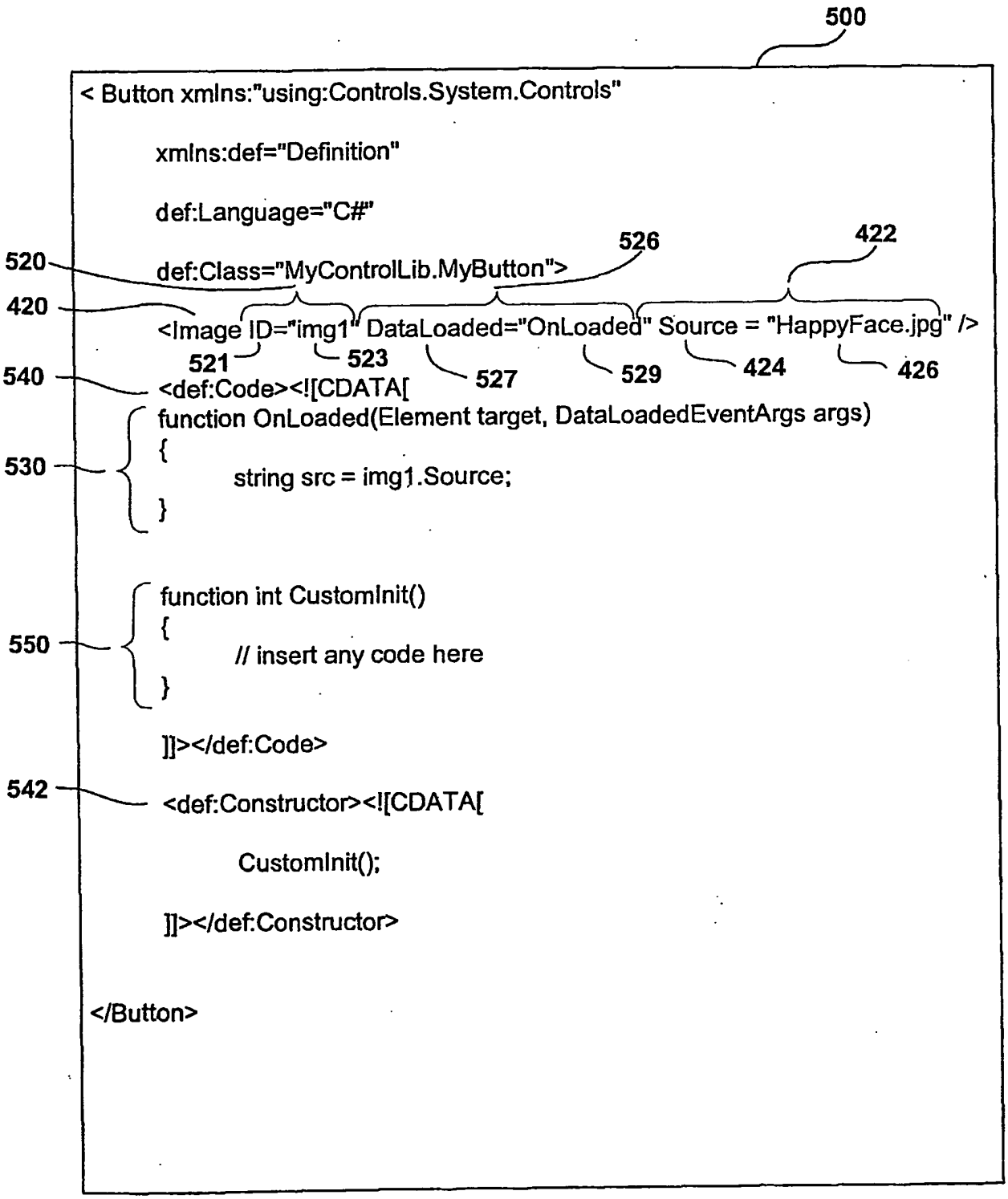
第 2 圖


300
310
中介資料
308
分頁映射表
304
符記二元讀取器
302
執行引擎
306
符記二元載入器
符記二元資料檔案
IL組合語言
208
210

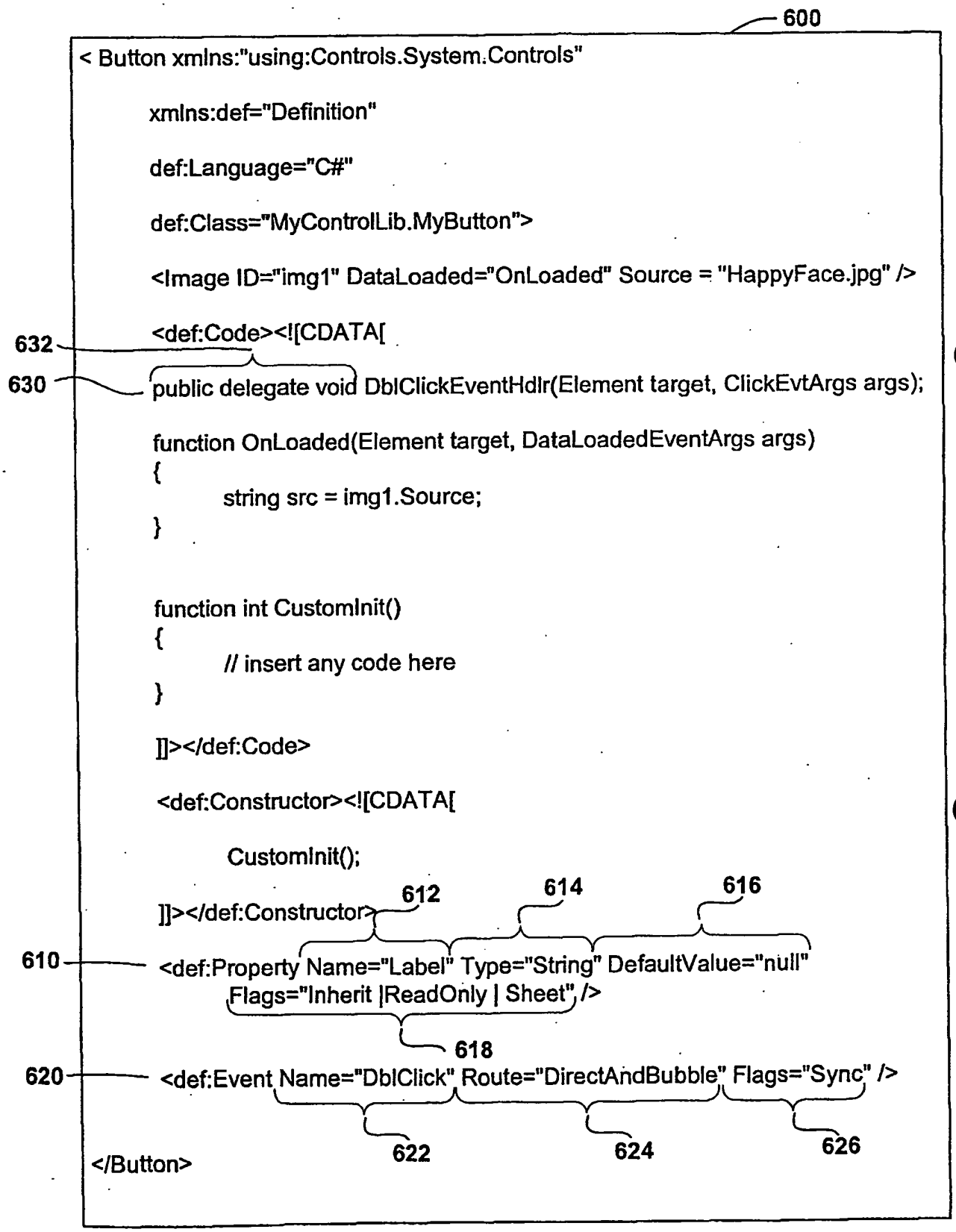
第 3 圖


404
402
400
< Button xmlns:"using:System.Controls"
406
xmlns:def="Definition"
412
408
def:Language="C#"
414
410
def:Class="MyControlLib.MyButton">
424
420
<Image Source = "HappyFace.jpg" />
426
</Button>
421
422

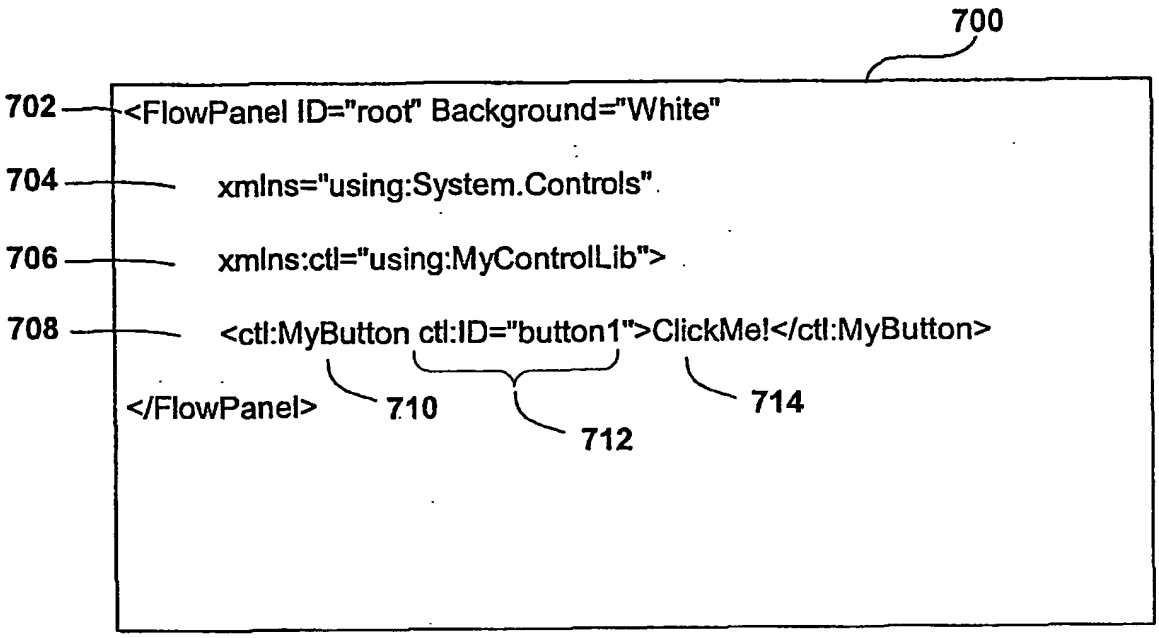
第 4 圖


500
< Button xmlns:"using:Controls.System.Controls"
xmlns:def="Definition"
def:Language="C#"
def:Class="MyControlLib.MyButton">
<Image ID="img1" DataLoaded="OnLoaded" Source = "HappyFace.jpg" />
<def:Code><![CDATA[
function OnLoaded(Element target, DataLoadedEventArgs args)
{
string src = img1.Source;
}
function int CustomInit()
{
// insert any code here
}
]]></def:Code>
<def:Constructor><![CDATA[
CustomInit();
]]></def:Constructor>
</Button>
520
420
521
523
526
527
529
422
424
426
540
530
550
542

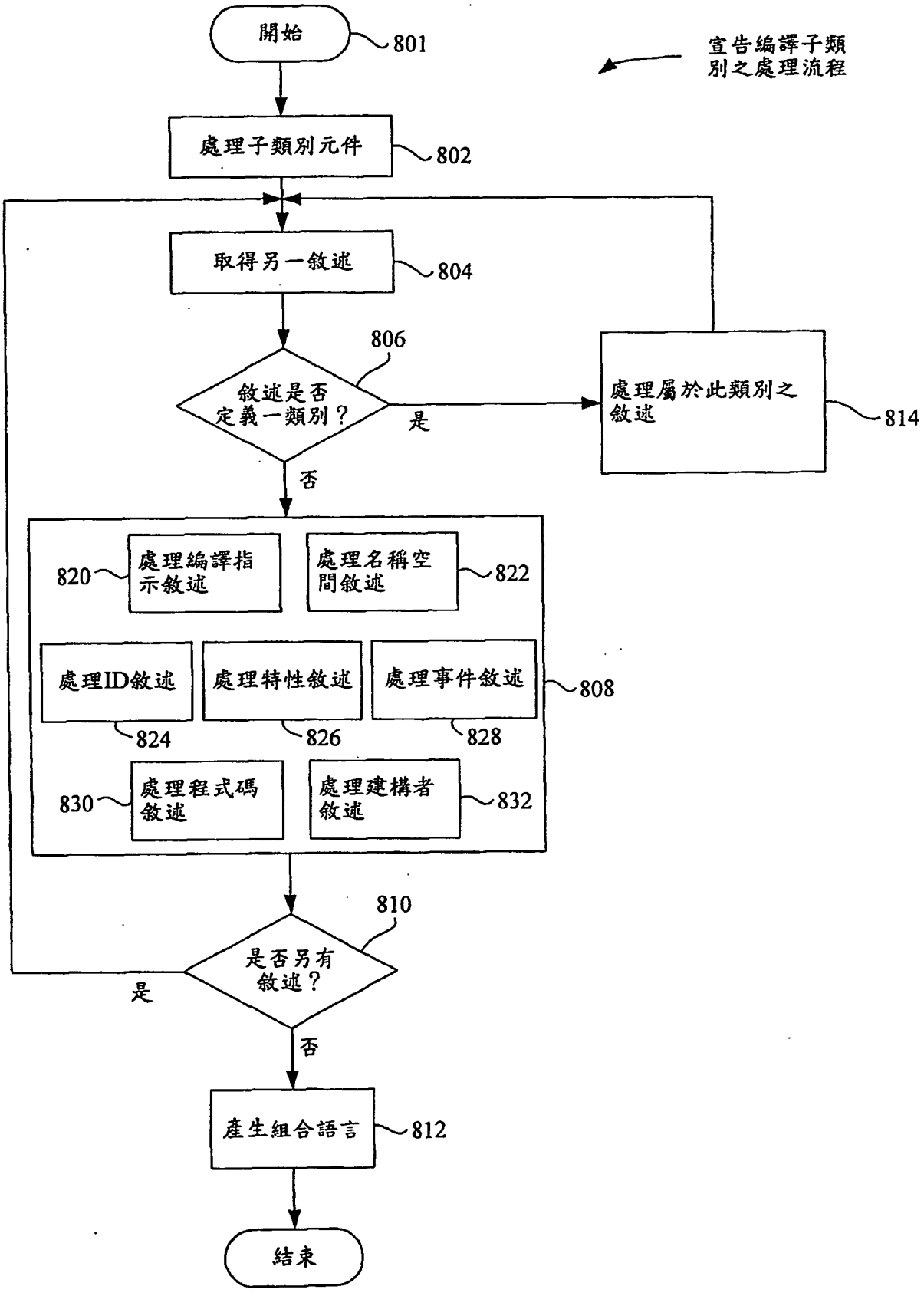
第 5 圖


600
< Button xmlns:"using:Controls.System.Controls"
xmlns:def="Definition"
def:Language="C#"
def:Class="MyControlLib.MyButton">
<Image ID="img1" DataLoaded="OnLoaded" Source = "HappyFace.jpg" />
<def:Code><![CDATA[
632
630
public delegate void DblClickEventHdlr(Element target, ClickEvtArgs args);
function OnLoaded(Element target, DataLoadedEventArgs args)
{
string src = img1.Source;
}
function int CustomInit()
{
// insert any code here
}
]]></def:Code>
<def:Constructor><![CDATA[
CustomInit();
]]></def:Constructor>
612
614
616
610
<def:Property Name="Label" Type="String" DefaultValue="null"
Flags="Inherit |ReadOnly | Sheet" />
618
620
<def:Event Name="DblClick" Route="DirectAndBubble" Flags="Sync" />
622
624
626
</Button>

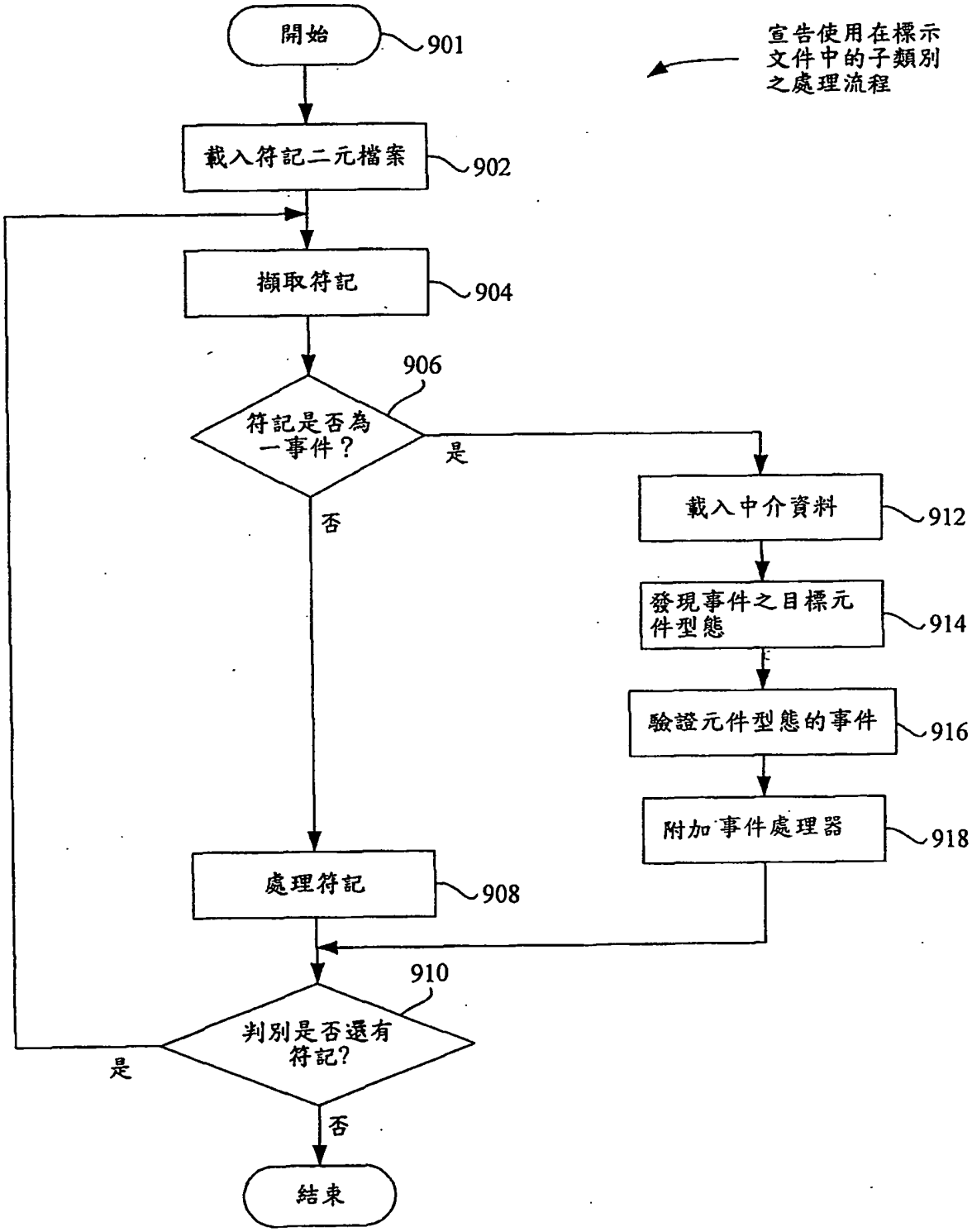
第 6 圖


700
702
<FlowPanel ID="root" Background="White"
704
xmlns="using:System.Controls"
706
xmlns:ctl="using:MyControlLib">
708
<ctl:MyButton ctl:ID="button1">ClickMe!</ctl:MyButton>
</FlowPanel>
710
712
714

第 7 圖


開始
801
宣告編譯子類別之處理流程
處理子類別元件
802
取得另一敘述
804
806
敘述是否定義一類別？
是
處理屬於此類別之敘述
814
否
820
處理編譯指示敘述
處理名稱空間敘述
822
處理ID敘述
處理特性敘述
處理事件敘述
808
824
826
828
830
處理程式碼敘述
處理建構者敘述
832
810
是否另有敘述？
是
否
產生組合語言
812
結束

第 8 圖


開始
901
宣告使用在標示文件中的子類別之處理流程
載入符記二元檔案
902
擷取符記
904
906
符記是否為一事件？
是
否
載入中介資料
912
發現事件之目標元件型態
914
驗證元件型態的事件
916
附加事件處理器
918
處理符記
908
910
判別是否還有符記?
是
否
結束

第 9 圖

1000

```
1  namespace MyControlLib {
2    using System.Controls;
3    using System;
4
5    public class MyButton : System.Controls.Button {
6
7    private System.Controls.Image img1;
8
9    public static System.DynamicProperty LabelProperty =
10     System.ComponentModel.Properties.PropertyManager.RegisterProperty(
11       "Label", ((PropertyFlags.Inherit | PropertyFlags.ReadOnly) | PropertyFlags.Sheet),
12       typeof(String), null, typeof(MyButton), null, typeof(MyButton), 0);
13
14   public static System.DynamicEvent DblClickEvent =
15     System.EventManager.RegisterEvent("DblClick", EventRoute.DirectAndBubble,
16     EventFlags.Sync, typeof(DblClickEventHandler), typeof(MyButton));
17
18     function OnLoaded(Element target, DataLoadedEventArgs args)
19     {  string src = img1.Source;              }
20
21     function int CustomInit()                 { }
22
23     public MyButton(System.Windows.Element parent) : base(parent)
24     {  this.__Initialize_This();              }
25
26     public MyButton() :  this(null)           { }
27
28     public String Label {
29        get { return ((String)(this.GetValue(MyButton.LabelProperty)));      }
30      }
31
32      private void __Initialize_This() {
33        CustomInit();
34        System.Controls.Button _Button_1_ = this;
35        System.Controls.Image _Image_2_ = new System.Controls.Image();
36        this.img1 = _Image_2_;
37        _Image_2_.SetValue(System.Element.IDProperty, "img1");
38        _Image_2_.AddHandler(System.Controls.Image.DataLoadedEvent, new
39                    System.Controls.DataLoadedEventHandler(this.OnLoaded),
40                    EventStage.Bubble, EventHandled.Unhandled, true);
41        _Image_2_.SetValue(System.Controls.Image.SourceProperty, "HappyFace.jpg");
42        _Button_1_.Elements.InsertBefore(null, _Image_2_);
43        }
44 } }
```

第 10 圖

七、指定代表圖：

(一)、本案指定代表圖為：第 8 圖。

(二)、本代表圖之元件代表符號簡單說明：

801-812 流程步驟

八、本案若有化學式時，請揭示最能顯示發明特徵的化學式：

無化學式