私立東海大學資訊工程與科學系研究所

碩士論文

指導教授：朱正忠 教授

以 XML 為基礎之重用嵌入式軟體元件庫系統

XML-based Embedded Software Reusable Component Library System

研究生：廖文賓

中華民國 九 十 八 年 七 月 二 號

摘要

本研究主要的目標是在嵌入式軟體系統的發展與重用中實行元件重用相關的技術，如此使其將更節省成本也具有更高的品質。雖然元件重用已經在過去展現出其優點，然而，在實際執行元件式軟體開發的過程中還有很多的不足之虞，而其中一個最重要的問題，就是如何幫助系統開發人員從先前開發過的系統中找尋符合目前系統需要的元件，取來使用以減少開發成本，而這個問題不容易解決的原因就是缺乏能夠提供系統開發人員瞭解程式元件的資訊，需要人工逐一檢視程式碼。本研究將提出一個除了可以檢索元件外，並能根據舊有系統的元件組成模式建議使用者採用某元件集合之架構，將可預期組合成特定功能，減少開發人員思索如何將元件組成的時間，提高開發效率。

關鍵詞：嵌入式元件、元件重用、元件式軟體發展

Abstract

In this research, we will talk about the component reuse and technologies to development and reusable of embedded software systems; As a result, we expect to keep the cost down and let our quality to be better than before. Although there are many advantages on component reuse and have been proven in previous researches, we still study and try to find another ways which can render the software development process more efficient. There is a question. ”Are there any ways can help software developer to find out suitable components in the system that has been ready done it? Can we reducing some of the components in ready-made system and saving our development cost?” And then, we have know it, there is much harder to retrieve suitable components. In this research, we will describe how to efficiency using component reusable, we will describe our solution to let the component reusable efficiently and give some useful suggestion. At the end of this propose, we will contain the software development experiences and related components. The software development experiences will help to develop a well-designed system and the related components may help to construct a complete structure. The technique of components reusable and components suggestible can reduce development time and decrease system cost.

Keywords: Embedded Component, Component Reuse, CBSD (Component-based Software Development)

章節目錄

圖目錄

表目錄

第一章 序論

1.1 前言

嵌入式系統在過去已經默默為人類服務 20 年，進入後 PC 時代後更是百花齊放，被譽為最有潛力的兆元產業，根據 Gartner 統計數據顯示，2005 年全球 32 位元及 64 位元的嵌入式產品，市場產值近 80 億美元，但是 2012 年產值將可過 120 億美元，年複合成長率（CAGR）達 7%。其中端點銷售系統（POS）、博奕機、資料查詢系統（Kiosk）與工業自動化控制等類別產品，CAGR 可達 11%。

結合軟體和硬體、應用、服務的嵌入式系統也快速「嵌入」我們的生活，舉凡各項可攜式產品外，便利商店等，甚至是車用電子系統、工廠生產自動控制、先進醫療器材以及智慧型辦公室等都可見各種嵌入式系統的應用。嵌入式系統實際上亦是電腦系統，在外表上看起來，假裝不是電腦，這樣卻能夠衍生出許多新價值，例如利用樂高做成的機器人，原本只是簡單的積木玩具，但加上嵌入式系統，卻能夠成為機器人、提高積木玩具的價值，降低機器人的門檻。常見的手錶亦然，加入嵌入式系統，就不止單純顯示時間功能，能更加聰明與網路世界連接。

而不同的嵌入式控制器應用環境卻可能有截然不同的需求，也許只有

「更低成本」是所有應用共通的需求。現代嵌入式軟體因為目前較少有符合標準的重用(Reuse)的流程，通常當一個嵌入式軟體需要被開發時，往往會重頭開始開發，但是因為硬體的不同，開發者的不同，而造成相當大的嵌入式軟體的開發成本，因此如何降低設備材料成本、統合系統元件、提高嵌入式軟體的重用率以及降低營運支出等等議題值得被提出，使得可有效改善高品質嵌入式產品設備的成本。

嵌入式軟體元件的重用目前遇到的問題，是不同於單純的軟體重用，必須考慮到軟硬體方面的限制、軟硬體的不同所造成的相同輸入值卻產出不同的輸出值、與設計語言必須加入較低階(Low-level)的語言參考等等問題，都是嵌入式軟體元件在重用方面所面臨的問題。

有鑑於此，為了提高嵌入式軟體元件的重用性(Reusability)與元件之間的關聯性，我們提出以 XML 為基礎的系統模式(System Pattern)[11]，並開發了一套以 XML 為基底並結合參考硬體限制的重用嵌入式軟體元件庫。本系統中，在嵌入式軟體元件被建立時，以一個系統模式的畫面呈現給元件建立者，並設定相關元件屬性來描述元件的用途，讓元件建立者能夠清楚瞭解需要輸入什麼資訊以達到元件的重用性。當元件輸入完畢後，會將資訊輸出成 XML 檔案，並連同元件並儲存到元件庫中的方法[14]，以供日後的使用者使用，若未來元件的內容有所改變，也可以經由此系統模式更新元件，並以版本控制作為解決元件重複或不一致的情況產生。本研究並根

據語言的不同提供選擇的方式，經由官方提供或已通過標準認證的 Java API(Application Programming Interface)、C (ISO/IEC9899:1999)標準程式庫和 C++ (ISO/IEC 14882:1998)標準程式庫參考進行分類[3][10][18]，將來當使用者並不瞭解自己真正需求的元件時，就可以依照一套標準的方法來引導使用者篩選出需要的元件類型，將自己所需要的元件一一的選取。未來希望藉由本研究所開發之重用嵌入式軟體元件庫可以提供嵌入式軟體元件更高的重用性，以節省系統開發的成本。

1.2 研究動機與目的

隨著嵌入式軟體大量的開發，許多嵌入式軟體工程相關技術的研究應運而生，目的即是在於如何快速開發高品質的嵌入式軟體系統。嵌入式軟體的快速演進的今日，這些已開發的軟體內存的經驗如果完全捨棄，將軟體重頭做起，往往會再度付出相當的成本，並且無法快速開發滿足使用者的需求，這些經驗可能包含了軟體的不同階段的文件、程式碼、甚至是設計師投入的開發或維護的經驗。

在系統軟體建置時的分析與設計階段中，軟體的可重覆性利用常被探討並被認為是可以促進軟體的生產力(Productivity)以及可靠性(Reliability)的技術，而這種概念與想法在軟體工程中亦為研究中的重要問題[5]，綜合上述論點，本研究中將運用軟體工程的研究論點，並探討如何使用軟體元件重用的經驗引入嵌入式系統開發時進行研究。

在元件式軟體開發的過程中，系統分析師在分析系統後，利用一套元件系統整合，能透過過去所開發過的相關元件進行搜尋，並利用元件重新利用的優勢，減少系統開發的資源成本浪費。但一個良好的元件如何被系統設計或程式開發人員能快速搜尋並可提供使用，而當前的輔助工具只有透過過去程式開發文件中找尋出可利用的資源，或是利用開發團隊的職前教育訓練教導開發人員相關的資訊。但是光這些資料是非常不足去幫助開發

人員在開發時得到較完善的解答，他們除了在元件庫中尋找可重用的元件外，還必須重新瞭解程式原始碼的意義，並找到可以組成至開發中系統的方法，而這些組成方法很可能在先前開發過的系統中已經撰寫過了，但是開發人員不一定可以記得有這些方法存在，或者這些系統是不同開發團隊的產品，他們當然也無從瞭解這些解決方法。

一個元件式一個擁有良好定義(Well-defined)的嵌入式軟體單元，最後是可以被用來與其他元件結合成較大的嵌入式軟體系統[16][21]。當前多數的重用式元件工程都是以程式碼結構分析的方式來建構，需要牽涉冗長的軟體生命週期中各項文件的產出與結合以及最後的回饋，這對開發人員單純需要某幾個元件的需要而言，太過於繁瑣。程式碼結構分析則是可以告訴使用者單一元件的用途與結構，卻不能建議元件間組成某一舊有可用功能的組合模式。

從上述複雜的技術，我們瞭解到要輕易地從自己擁有的元件庫中取出適合的元件，必須要先行作相當大的付出。因此，為了降低這個門檻，使得開發人員可以輕易地瞭解在過去所開發軟體的元件及組成模式，有賴於提供開發人員簡單並且即時的參考性的嵌入式軟體元件庫的檢索方式。這個問題也是本論文研究的動機所在與目的。

1.3 研究方法與步驟

有鑒於上述的動機與目的，本研究將提供一個比較有效地且簡單的方式來檢索嵌入式軟體元件，利用有效的檢索方式，降低元件搜尋時盲目的逐一進行不同元件瀏覽時的資源浪費，並利用介面呈現方式，協助開發者瞭解不同元件相互間關連性的聯結，以及元件間使用的限制。最後依照這些限制，確實的知道哪些元件是與目標的元件有關的，並且瞭解所選取的有哪些限制來刪除不相關的元件。

本研究中運用以下幾項方法來進行研究探討：

1. 使用目前已有的官方提供或已通過標準認證的 Java API(Application Programming Interface)、C (ISO/IEC9899:1999)標準程式庫和 C++ (ISO/IEC 14882:1998)標準程式庫參考進行分類[3][10][18]，並且描述元件。目前這部份的研究相當多，論文中將修改引用前人的研究作為基礎，免除遺漏必要描述的疑慮。
2. 提出描述已存在的系統架構模式，把舊有系統的架構保存下來。
3. 並以這些資料作為依據，透過檢索內容，分析比較程式設計師的需求與限制，提供其有用的參考資訊，也是本研究主要的貢獻所在。

1.4 章節安排

本論文內容分為五章，各章節內容的架構如下：

第一章　序論

介紹本論文研究的緣起，研究動機與目的，並解說簡略說明研究方法與步驟，讓讀者快速瞭解背景，並作為以下章節說明的基礎。

第二章　背景知識與相關研究

整理目前相關研究發展狀況，以及概述相關技術的背景知識。

第三章　XML 為基底之嵌入式元件重用

對於本研究的使用方法，做詳細的說明描述。

第四章　研究範例

以一個簡單的範例來做說明。

第五章　結論與未來工作

對於目前的研究內容作一個總結，並且概述未來的研究方向。

第二章 背景知識與相關研究

2.1 物件導向(Object-oriented)

系統設計方法主要可分為兩個大類：一類為傳統的結構化方法，另一類則是後來崛起的物件導向的方法。結構化的方式，起源於 1960 年代末期，主要強調如何應用一些概念、策略與工具，來提升系統分析與設計、程式設計與測試之效能等。此方法也重視系統可維護性，常用系統環境圖(Context Diagram)、流程圖(Data Flow Diagram)、個體關係圖(Entity-relation Model)等[27]表示系統概念，並以資料字典(Data Dictionary)說明所使用資料的定義。

與傳統結構化之軟體開發技術所不同的是物件導向方法，它是以類似於人類的思考的模式來描述以及分析系統，可用來開發元件化軟體，也易於修改和不計次數的之重複使用，在現今這種強調快速反應的時代，物件導向的技術被廣泛的應用於軟體發展的各個階段。物件導向技術目前已趨於成熟，各種開發工具幾乎都支援，許多新興的程式語言也需要物件的觀念。

現今的軟體開發觀點，許多都是透過物件導向的觀念來進行分析、設計。而所謂的物件導向應該具備下列的特性：

1. 物件(Object)以及訊息(Message)

2. 繼承(Inheritance)

3. 多型(Polymorphism)

4. 封裝(Encapsulation)

5. 動態連結(Dynamic Binding)

利用物件導向所建製的軟體系統，它們的主要構成元素，就是物件以及類別。在 Coad 與 Yourdon[4]的研究中將物件做以下的定義：「物件-問題領域中某些事物的抽象化，可以反應保留系統資訊的能力、與系統互動的能力以及兩者兼顧。」

類別，則是針對某一組共通的物件所做的一種描述。每個物件都應具有以下三點特性：

1. 識別名稱(Identity)：用以和其他的物件作區分

2. 狀態(State)：描述物件特性的屬性資料

3. 行為(Behavior)：提供給外界使用的服務或操作等三部份

而在一個使用物件導向的軟體開發過程中，主要可以分為以下三個階段：[7][19]

1. 物件導向分析(Object-oriented Analysis, OOA)：OOA 主要是在需求階段，從需求文件的字彙裡，透過物件以及類別來對問題領域(Problem Domain)進行塑模(Modeling)。

2. 物件導向設計(Object-oriented Design, OOD)：OOD 的目的是為了讓開發者能夠了解系統的架構。在 OOD 的過程中，會導致物件導向分解(Object-oriented Decomposition)。而 OOD 主要是使用邏輯(Logical)以及實體(Physical)的標記法來表示系統的靜態(Static)及動態(Dynamic)的觀點，並且會將系統之間的物件以及彼此之間的關係描述出來。
3. 物件導向程式設計(Object-oriented Programming, OOP)：OOP 是指在軟體發展過程中，將 OOA、OOD 的思想進行表達和實現。

2.2 重用導向軟體開發流程

Reuse-oriented Development 的軟體專案開發方式，亦常稱為元件式軟體開發(Component-based Software Development, CBSD)，這是一個有效地從現有元件中或是一般現成的商業軟體(Commercial-off-the-shelf, COTS)中取得需要的功能進行重用的開發流程，主要的流程包含元件分析、需求修改變更、利用現有元件的系統設計以及實作階段的開發與整合等活動，其流程概念如下圖 1[8]：

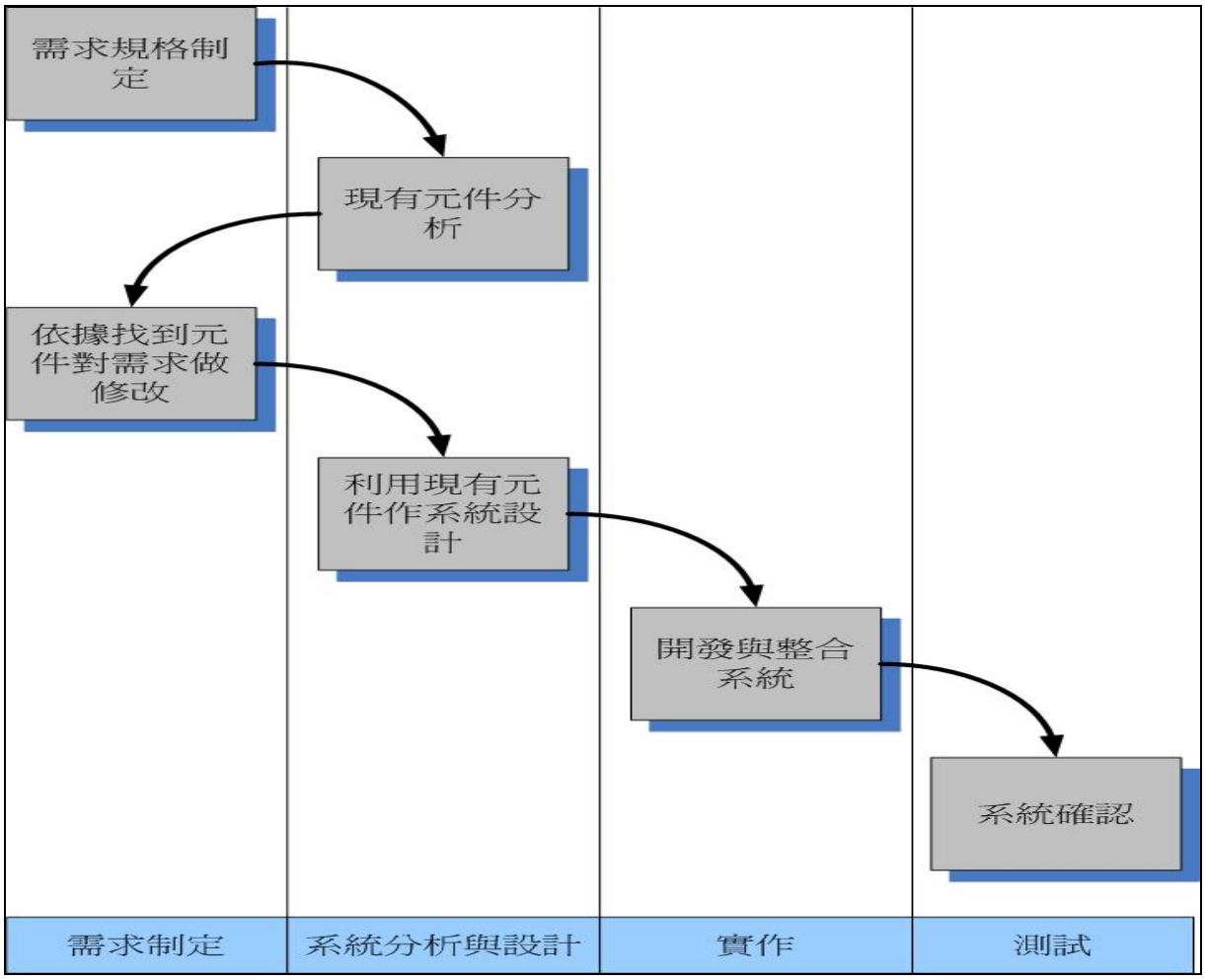


圖 1　元件式系統開發流程概念圖 [8]

元件亦為物件導向概念的延伸，在 OOP 中常會將程式分割成比較小的組合單元(Building Block)，通常就是物件或物件的集合，這兩者皆強調軟體的重用性(Reusability)。因此，這也是物件導向程式設計成為後起之秀的因素之一，兩者可以相輔相成幫助實現快速開發並兼顧高品質軟體的需求。

軟體元件的開發應包含整個結構(Architecture)、領域模組化(Domain Modeling)、元件規格(Component Specification)等[12]。整體結構就是在探討軟體架構(Software Architecture)，從中才能知道哪些部份可以引用現成的元

件達成目標。領域模組化是針對特定問題或是某個應用領域上的問題建立較為完整的元件模組，達到高度元件重用的目標。元件規格則在討論如何運用語言文字或圖形方式對元件內外部加以描述，提供元件的使用者快速瞭解元件的概觀及功能。另一個也是在幫助軟體重用的技術是 Design Pattern，它提供的是在某一個領域情況下可採用的組織關係，避免掉入泥沼，甚至重蹈前人的覆轍，後面 2.7 節將介紹 Design Patterns 作簡介。

2.3 COMDES

建置一個系統根據 COMDES(Component-based Design of Software for Distributed Embedded Systems)[1]的工作流程可以分為三個部份：

1. Function Unit
2. Function Activities
3. Function Blocks

COMDES 技術是將一個嵌入式系統想像在一個軟體的功能元件(Function Unit)中辦扮演了自動子系統的代理(Agent)角色，而它的功能就是為了安裝特殊的應用程式。概觀系統可由圖 2 來解釋訊號流程的特殊功能單元和它們之間的相互影響。

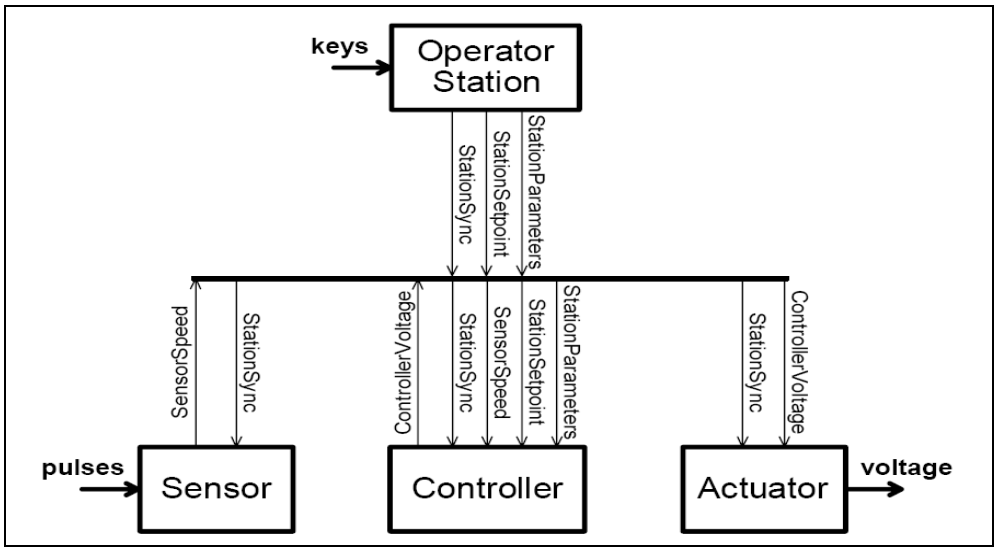


圖 2　Function Unit Diagram [1]

Function Unit 之間是由交換訊號來影響，它們之間的關係就如同生產者與消費者或是客戶端與服務端，此種關係是一對多的(One-to-many)。

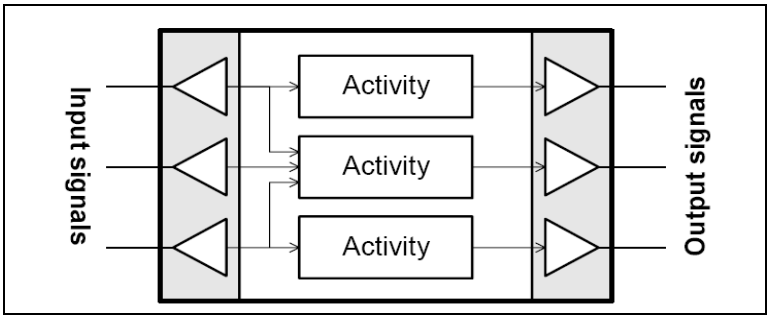


圖 3　Function Unit Structure [1]

關於 Function Units 之壓縮輸出/輸入領導訊號，和控制一個或多個的執行緒(如圖 3)，引起這些特殊應用程式之間的相互影響的低階(Low-level)元件稱為 Function Block。Function Block 為實作特殊訊號流程與控制功能的存取(如圖 4)。

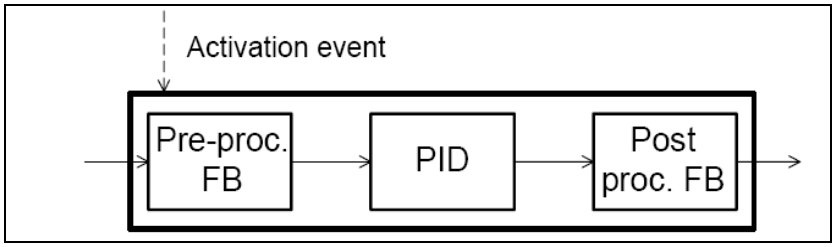


圖 4　Function Unit Activity [1]

本研究主要根據此技術之概念如下圖 5，將元件輸入與輸出也一併加入到參考之中，嵌入式系統往往因為輸入輸出的不同，而導致所需求的元件也不相同，所以將元件之輸入值與輸出值加入到元件之中，此方法則可以更增加嵌入式軟體元件的重用性。

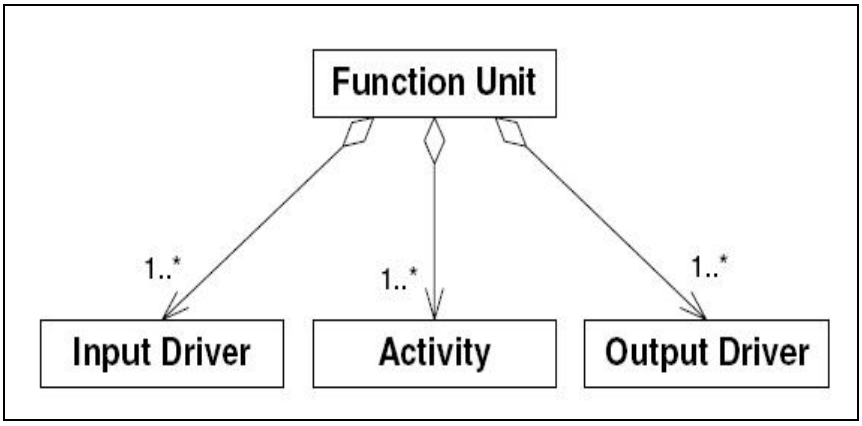


圖 5　Function Unit [1]

2.4 相關研究

對於目前為止在重用式元件系統上的研究根據其所採用的方法，可以被粗略的分為三類：文字上的分類的資訊檢索(Information Retrieval)[20]、以

知識為基礎(Knowledge-based)方式的人工智慧的決策系統以及最近 Web Services 的幾項標準(例如:SOAP、WSDL 和 UDDI)也被用來發展企業的元件庫。實際上後面兩者都是以知識為基礎所發展出的不同方法的架構。

2.4.1 資訊檢索

資訊檢索的方式[20][21]，是由文件中的架構所萃取出資訊來描述元件，它沒有語義上的知識涉入，也不是在翻譯文件，更不是在瞭解文件，而只是單純地把文件特徵化[13]，使得可以取得文字上的特徵。這種方式有些會把元件的程式碼整個讀入取出註解(Comments)的部份，然後分析這註解的文字取出有用的部份，再把這些資料儲存起來以供檢索使用。

2.4.2 以知識為基礎(Knowledge-based)方式

相對地，在以知識為基礎的方式所開發的重用式系統[25]，它的目的在於瞭解查詢的內容並提供功能上的解答，所以通常會建立較為完善的知識庫，可能包含整個軟體生命流程中的各種資料(例如:軟體需求、設計模式、詞彙庫、企業流程等)，因此通常會比較資料檢索方式有智慧，事實上也有常與人工智慧(例如:Rule-based的人工智慧就需要這些領域知識為基礎)以相輔相成的方式作為技術。

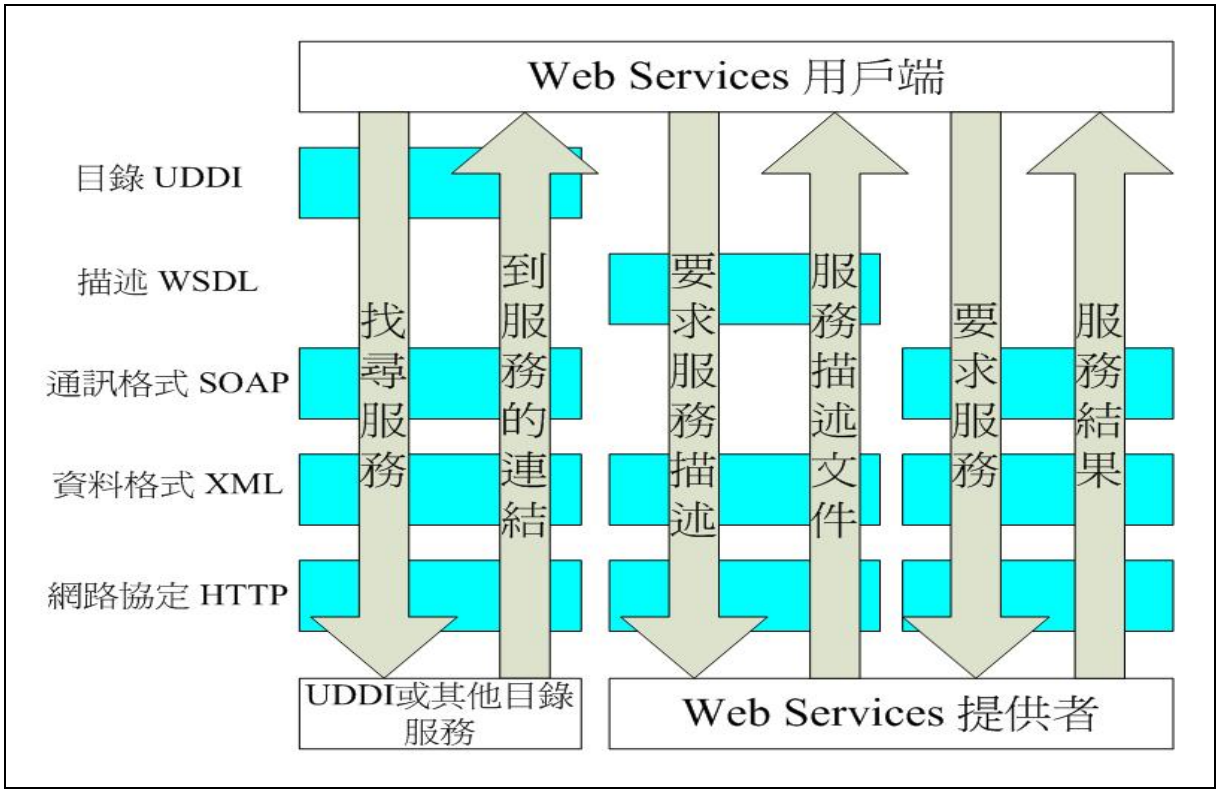
人工智慧的一個比較流行的定義，也是該領域較早的定義，是由John

McCarthy在1956年的達特矛斯會議(Dartmouth Conference)上提出的:人工智慧就是要讓機器的行為看起來就像是人所表現出的智能行為一樣[26]。由此可知，這個方式是交由機器主動分析並依據現有知識提供決策方式。

2.4.3 Web Services

隨著XML技術的逐漸地成熟，Web Services的名詞也隨之炒熱，Web Services的中文名稱可以稱為「網路服務」或是「網路資訊服務計算技術」[22]，它的出現是為了提供除了個人外，擴展到軟體作為服務對象，也就是提供軟體間互相溝通的管道，而基礎的資料格式就是XML。由於這種軟體溝通的好處，使得資料交換變的簡易許多，所以最近幾年也有元件重用的研究透過Web Services來實現。

Web Services的基礎技術包括：XML、WSDL、SOAP、UDDI等[22]，它們之間運作的方式如下圖 6所示：



資料來源；資策會2002 年12 月

圖 6　各項技術在Web Services運作中所扮演的角色

XML(eXtensible Markup Language)：

將在2.5節中說明。

WSDL(Web Services Description Language)：

WSDL只是一種簡單的定義語言，雖然簡單但XML-Based的WSDL利用XML原生的延展性、電腦可讀性，完善的定義出一個Web Service的服務內容與使用方式。

透過Web Services的使用，使用者不必考慮擔心這些服務是建構在什麼平台之後、是用什麼技術來實作，而將來如果有更好的服務或服務

提供者時，也可以輕易的將服務更換或更新。對資料收集平台來講，將Web Service當作標準化的元件，可以取用現有的元件，快速的將系統建構完成；而將心思專注在規劃更好、更完善的系統上。

WSDL 語法分為兩個部分：

1. 描述服務介面（Service Interface Definition）：

   這個部分主要是說明，WSDL 檔案資料的格式、傳遞溝通的訊息、如何進行作業等簡單規範性的說明，這個部分是可以重複使用的。

2. 實作服務定義（Service Implementation Definition）：

   這個部分主要是說明，由如何進行由服務介面所提供的服務。

WSDL是用來描述Web Services應用程式能力，也是使用XML文件來描述。

SOAP(Simple Object Access Protocol)：

獨立式安全管理。將安全的控管與Web Services的開發分離，降低系統開發的複雜度及提高系統的安全政策變動彈性。安全控管與系統開發的分離，不需擔心網路的攻擊導致資料洩漏，可以專心執行其所提供之服務，讓後端的系統開發更單純化，提高在系統安全政策管理上的彈性。例如當想要開放一個Web Services給新加盟的組織或客戶，只需要

在Gateway端更改Policy的設定，不需要修改任何的程式碼。舊的安全規格的修改或新的安全規格的制訂，也只需要修改及提升安全防禦系統的功能，而不需要修正已經建置的Web Services。如此可以大幅降低系統開發和維護的複雜度及成本。

SOAP提供在網路上交換結構化和型別資訊的一種簡易通訊協定，讓應用程式之間互相溝通，但不需要知道彼此的作業平台是什麼，也不需要知道對方時做的細節資訊。

UDDI(Universal Description, Discovery and Integration)：

本部份為執行服務時，使用者必須輸入的想查詢的項目。當Matchmaker找到相關的服務後，此時WSAF(Web Services Automatic Framework)便會將這些服務列出來給使用者，使用者可以從中選出一個想要查詢的項目，WSAF會透過UDDI找到選定服務的連結資訊，依照查詢的項目需要的輸入參數類型和數量，進行輸入介面的自動設計，以供使用者查詢想要的項目。最後，我們透過服務執行者去連結和執行服務，便將執行的結果傳回給使用者。

UDDI也是以XML作為基礎描述的文件，目的是讓服務提供註冊Web Services，也讓服務的使用者搜尋這些Web Services。

2.5 XML (eXtensible Markup Language)

XML(eXtensible Markup Language)[2]是由World Wide Web Consortium (W3C)所制定的標準，是SGML(Standard Generalized Markup Language)的一部份，SGML是用來描述電子文件的結構及內文的一項國際標準(ISO8879)。XML已經成為廣泛用來表示複雜文件的標準，具有擴展性(Extensibility)、結構性(Structure)、描述性(Description)、確認性(Validation)等特性，這些特性提供有效的結構化文件的表示方法，並可以驗證內容(例如:DTD、XML Schema)。

XML在寫法上十分類似HTML，在功能上可以補足HTML標籤不足，擁有更多的擴充性。不過XML不是用來編排內容，而是用來描述資料，它並沒有如同HTML一般的預設標籤，使用者需要自己定義描述資料的各種標籤。

XML相關技術相當的多[23]，各種的技術關聯大致如下圖 7所示:

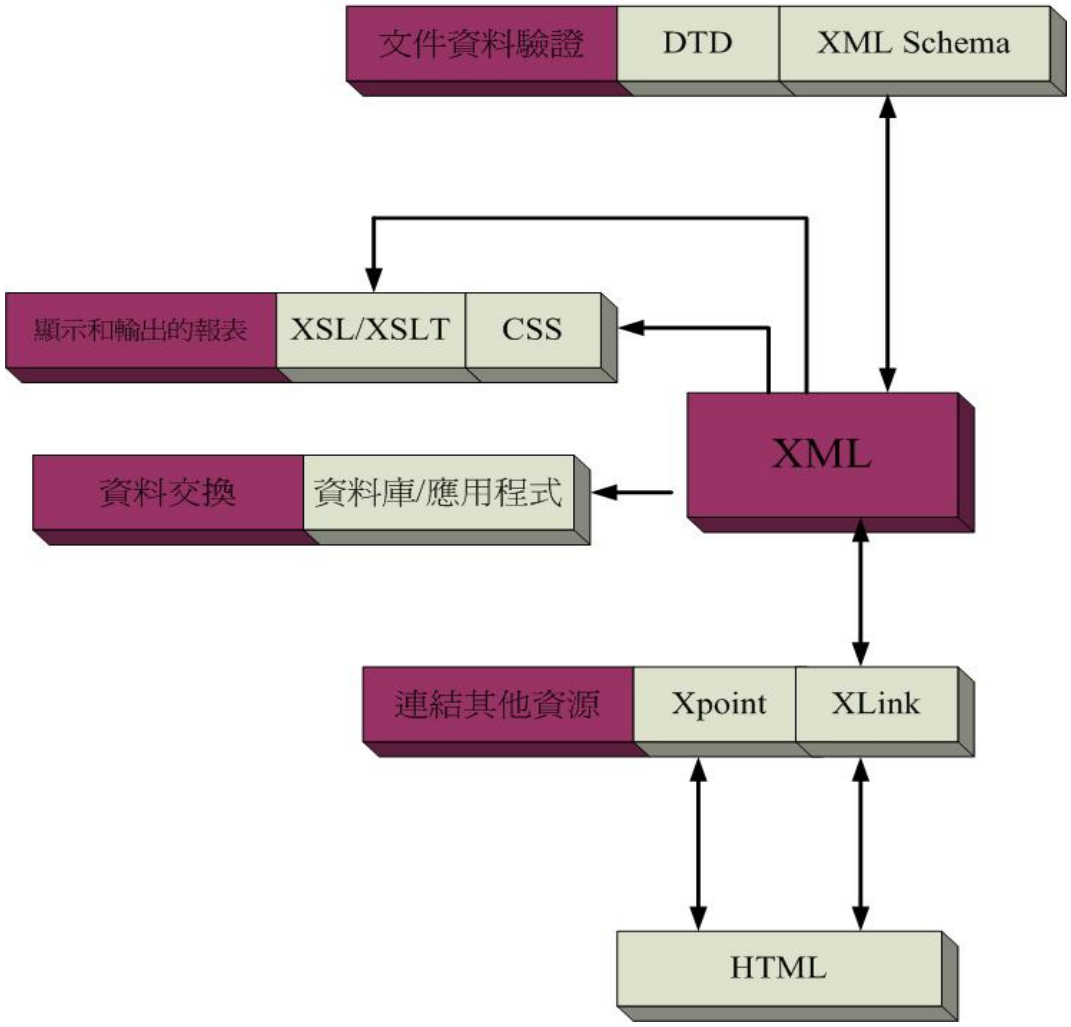


圖 7　XML相關技術概念圖 [23]

上圖 7所示，可以知道關於語法檢查的是DTD和XML Schema，顯示報表輸出的是和XSL/XSLT，而可以連結其他資源的是Xpoint和XLink。

2.5.1 XML文件的資料驗證

XML 資料驗證的相關技術就是 DTD 和 XML Schema，所謂驗證就是使用 DTD 或是 XML Schema 定義一組規則，這套規則可以檢查 XML 文件的架構和標籤內容是否符合原本定義的規則。

DTD(Document Type Definition)[2][23]：

DTD 就是 SGML 的語法檢查，DTD 可以幫助 XML 剖析器來解釋 XML 文件的內容，例如：DTD 可以告訴食品清單每一個食品需要有的編號、名稱、價格、成份、製造商和生產日期等資料，成份清單可以擁有一種或多種原料成份。

XML Schema Definition Language (XSD)[2][23]：

XML Schema 是由 W3C 組織所提出的檢查方式。XML Schema 定義語法的方式是使用 XML 標籤，因此它的架構其實就是一份 XML 文件，所以在使用方式上和撰寫一份 XML 文件相同，不太需要像 DTD 學習另一種語法。

2.5.2 XML連結其他資源的方法

有別於 HTML 使用超連結快速連結其他網頁或網路上的資源，W3C 組織提出了 Xpoint 和 XLink，提供比 HTML 更好的技術支援，但可惜大部分的剖析器不支援這兩種技術。

2.5.3 XML輸出和文件轉換

CSS(Cascading Style Sheets)[24]：

HTML 文件常使用 CSS 來重新定義標籤的樣式，相同地，XML 也可以使用 CSS 來定義 XML 標籤顯示樣式。

XSL/XSLT(eXtensible Style sheet Language)[23]：

XSL 亦屬於樣式程式語言,但是它比 CSS 提供更強大的顯示功能來輸出顯示 XML 文件，它能夠使用程式碼取出 XML 文件所需的資料，並顯示的樣式。除了顯示 XML 文件內容外，XSL 還有另一項功能就是文件轉換，也就是 XSLT(eXtensible Style sheet Language Transformation),可以轉換 XML 文件,它使用 XPath(XML Path Language)在 XML 文件中找尋資料，可以指出文件架構和資料的位置。

2.6 JAVA API and C/C++ Standard Library

2.6.1 JAVA API(Application Programming Interface)[10]

Java 平台標準版(Java Platform, Standard Edition；Java SE，舊稱為 J2SE)，提供單機電腦平台上的 Java 應用系統、Applet 等程式的開發工具與執行環境，並包含一組好用的標準 API 類別，讓 Java 開發人員能在個人電腦和伺服器上快速地開發或部署 Java 應用系統。開發人員可使用 JAVA 現有完整並因應不同資訊環境所不斷開發的 API(Application Programming Interface)。這些 API 封存於所謂的套件(Package)中並以類別(Class)或介面(Interface)的形式存在，這些類別或介面可以被延伸

(extend)或實作(implement)來實現物件導向程式設計對於封裝(Encapsulation)，繼承(Inheritance)，多型(Polymorphism)等特色。

JAVA API則是由昇陽(Sun)公司所制定的一套標準的類別程式庫，主要的目的是提供開發者快速且有系統的開發系統，JAVA的API就如同C/C++的函式程式庫相同，提供一套參考標準讓開發者只需將程式套用，不須自行撰寫，提供開發者節省撰寫程式所花費的時間成本。

2.6.2 ANSI ISO/IEC 9899:1999 and ISO/IEC 14882:1998 [3][18]

ISO/IEC 9899:1999主要是由ISO/IEC 9899:1990增訂而來的，ANSI也規定一套了標準函式程式庫。國際標準化組織(ANSI ISO)成立ISO/IEC JTC1/SC22/WG14工作組，來規定國際標準的 C 語言。通過對ANSI 標準的少量修改，最終通過了 ISO/IEC 9899:1990。

傳統 C 語言 到 ANSI/ISO 標準 C 語言 的改進包括：

1. 增加了真正的標準庫
2. 新的預先處理的命令與特性
3. 函數原型允許在函數申明中指定參數類型
4. 一些新的關鍵字，包括 const、volatile 與 signed
5. 字元、字串與位元組多字元
6. 對約定規則、聲明和類型檢查的許多變動與說明

在 ANSI 的標準確定後，C 語言的規範在一段時間內沒有大的變動，然而 C++在自己的標準化建立過程中繼續發展壯大。不過，這個標準引出了 1999 年 ISO/IEC 9899:1999 的發表。它通常被稱為 C99。

至於 C++的標準函式程式庫 ISO/IEC 14882:1998。在 1998 年國際標準組織 (ISO) 頒布 C++程式設計語言的國際標準 ISO/IEC 14882:1998。C++的標準化是由 ANSI 和 ISO 共同進行的，兩個組織在 1998 年正式發布了編號為 ISO/IEC 14882:1998 的 C++標準，簡稱為 ISO C++98 或 C++98。標準程式庫是 C++標準規格的一部份，使用支援 I/O、字串、容器(資料結構)、演算法、數值計算、國別(不同字元)等內容。

在本研究中，主要依據 ISO/IEC 9899:1999 與 ISO/IEC 14882:1998 所制定的 C 與 C++標準函式程式庫，來幫助元件進行分類，當元件使用 C 與 C++語言為開發工具時，可經由標準來選取元件所使用的函式，並經由這些函式來定義元件所屬的分類。

2.7 Design Patterns

Design Patterns(中文譯為設計樣式、設計樣板或設計模式)[6]，以 Gamma、Helm、Johnson 以及 Vlissides 四人影響最大，他們四人也被稱為四人幫(Gang of Four, Gof)，設計模式之所以受到更大的重視，他們也是主

要的功臣，他們為設計模式做了定義：一個模式(Pattern)是由三個部份所組成的規則，分別是陳述某些背景環境(Context)間的關聯性、一個問題以及一個解決方案。下表 1 中說明設計模式的四個不可或缺的元素。

表 1　Design Patterns 的四個重要基本特質

項目	描述
Pattern Name(名字)	定義一個設計模式，從而增加我們的字彙，讓我們可以在高度抽象化階段做設計。
Problem(問題)	描述何時可以應用這個設計模式。
Solution(解決)	不應只是描述特定的實作，應該以一般化的方式來描述模式，才可以應用在不同情境(Context)之下。
Consequences(結果)	應用某個設計模式的結果，觀察在設計模式中的互相作用，提供評估替代方案與制定決策的關鍵要素。

在大部分的設計上就可以依據需要重複地使用這些 Patterns，替設計者省下許多功夫或者避免一些不必要的錯誤，並減少維護上的次數和困難。對於給軟體設計問題一個明確且精細的答案，一般而言，通常會在較大的軟體系統裡廣泛地使用 Design Patterns。

然而實際上，設計師可能會採取某個 Design Pattern 來當作這個系統的設計依據，但是實作或維護程式的工程師有時會因為並不瞭解這個設計樣

板，或者為了某種方便而不嚴謹的遵照設計規格撰寫程式，這將會使架構失去原本的功能或者在結構方面受損。因此，如何有效地使 Design Patterns 落實在實作上,是需要去做到 Design Patterns 的程式碼的自動化產生並驗證撰寫的程式碼是否符合設計樣板，目前 Integrated Development Environment(IDE)工具尚無提供這類完整的功能，以致一般沒有 Patterns 經驗的工程師沒有足夠的能力依循設計規格上所制定的格式來撰寫程式碼，即使撰寫出來也無法驗證是否符合設計規格。

第三章 XML 為基底之嵌入式元件重用

本研究中，我們運用重用資料庫(Reuse Library)進行元件重用，以便有效率的整合及使用。重用嵌入式元件庫是用來存放所有處理中或處理後的重用嵌入式元件。由於可預期的嵌入式元件指定術語與推論述語累積後，元件與元件之間的異質性擴大，必然造成資料庫因龐大而變得相當複雜，而連帶影響後半階段元件檢索時的困難度問題。因此，一個以物件導向技術為基礎的嵌入式元件程式資料庫之設計與運作的成功與否，實是整個系統運作效率良莠的關鍵。

在本研究中，我們將探討包含 XML 的儲存方式研究以及如何透過使用者介面(User Interface)呈現給使用者。在過去，元件式重用的研究常是重整工程中一環，把原有系統中可能存在的元件重新產生新的程式碼，其中的範疇大致上包含可重用的元件辨識(Identification)、取得(Extraction)、呈現方法(Representation)、檢索(Retrieval)。而在本研究中，將依循這樣的流程來設計本系統的開發流程。

3.1 系統說明

3.1.1 系統概念

本研究主要概念是以 Three-tier(三階層)的概念來研究元件的重用性

(Reusability)，如下圖 8 所示，概念主要依據 BOTS(Building Operate Transfer System)[15]將嵌入式軟體系統分為三個階層，而它們上下之間為繼承的關係：

1. Interface(介面)：介面主要目的是記錄元件之間的關聯性，並且將其記錄下來，將來使用者在重用時瞭解與元件與其他元件的關聯性，對於同時要尋找其他相關的元件是有相當大的幫助。
2. Component(元件)：元件主要目的是將元件分類，本研究依據官方提供或已通過標準認證的 Java API(Application Programming Interface)、C (ISO/IEC9899:1999)標準程式庫和 C++ (ISO/IEC 14882:1998)標準程式庫參考[3][10][18]進行分類。當元件建立時，以元件所使用的參考分類，當元件重用時，依照這些分類檢索出需要的元件。在過去，常常因為無法找尋到真正需要的元件而造成時間上與成本上的浪費，依照本研究的方法可降低找尋元件所花費的時間與風險成本。
3. Attribute(屬性)&Constraint(限制)：屬性主要為元件使用的參數，當元件被使用時，需要哪些輸入參數，會產生哪些輸出參數，對於嵌入式軟體而言，是相當重要的資訊。當元件重用時，記錄此項資訊使元件的重用性大大的提高，使用者不會因為元件無法產出自己所需的參數而導致無法找尋到使用者真正想要的元件。而限制部份以 3.1.3 節說明為主。

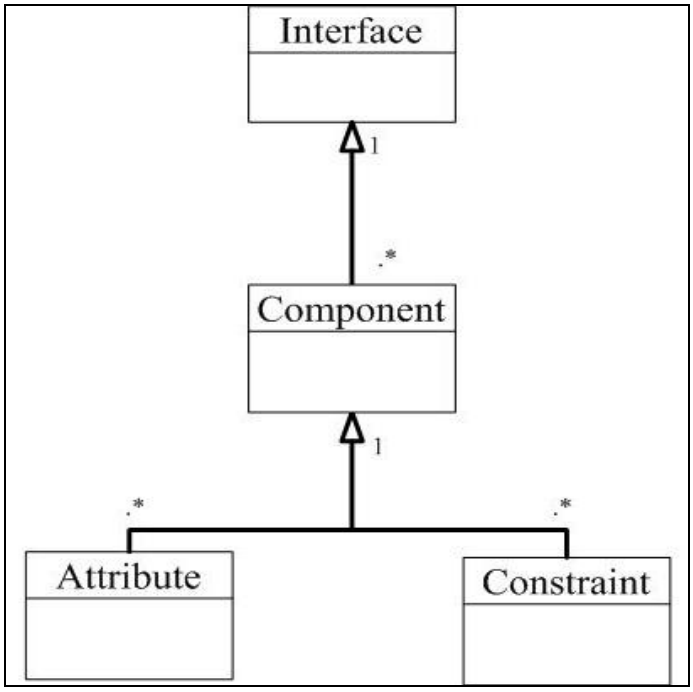


圖 8　系統階層概念圖

3.1.2 系統架構

本研究主要架構內容如下圖 9，主要依據上述有關物件導向繼承概念開發重用嵌入式軟體元件庫系統，經由本研究的系統產生出下列四種不同類型的文件，分別代表了各自的意義，以下會分別說明，這些文件經由本系統檢查是否符合格式，進而經由 XML Generator 產生 XML 文件記錄相關資訊，最後將元件與 XML 文件一起存入元件庫中，將來檢索元件時也依據這些 XML 記錄資訊來進行條件檢索。

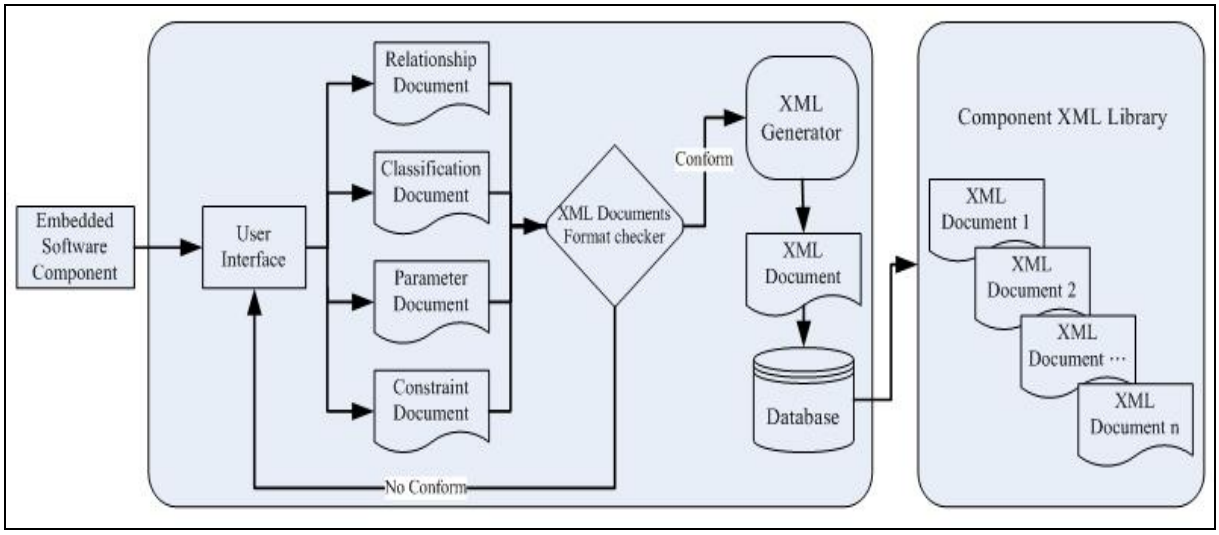


圖 9　系統架構圖

嵌入式元件系統，本研究主要依據上述的系統概念，將嵌入式軟體元件分為四個主要的部份：

1. Relationship(關聯性)：如同 3.1.1 節中所提到的介面(Interface)，內部可能包含一個至數個元件，讓使用者能夠經由此元件所屬之介面，進而瞭解此元件之間的使用與關聯，在本系統藉由此介面增加元件之間之關聯性，並且能夠記錄良好使用元件的方式。
2. Classification(分類)：此部分如同 3.1.1 節中所提到的元件(Component)，主要依據官方提供或已通過標準認證的 Java API(Application Programming Interface)、C (ISO/IEC9899:1999)標準程式庫和 C++ (ISO/IEC 14882:1998)標準程式庫參考[3][10][18]為基礎將元件進行分類，此技術主要依照所屬的語言之 API 與標準參考，元件依照所選取的參考進行分類。
3. Parameter(參數)：此部份如同 3.1.1 節中所提到的屬性(Attribute)，

主要分類為輸入參數、輸出參數。輸入參數主要是使使用者瞭解元件所需要之參數，提供使用者選取元件，輸出元件是為提供使用者參考參數輸出是否符合使用者需求。

4. Constraint(限制)：限制主要因為環境限制，由於嵌入式系統軟體開發受到環境上的限制可能無法隨心所欲的設計軟體，所以在本研究加入了環境限制的考量，讓軟體開發流程符合嵌入式系統的領域。以下 3.1.3 節將對於環境限制作細部的說明。

本系統主要融合 JAVA API、C 和 C++標準程式庫參考技術，提供一個元件重用空間給使用者使用，進而開發一套重用嵌入式軟體元件庫系統。在軟體的重用上，使用者常常因為不完全瞭解其屬性如何設定而增加元件重用的困難性，此系統提供良好使用的樣版(元件)屬性設定，讓使用者更容易且快速瞭解元件屬性設定以及使用方式，以及利用這些屬性讓使用者更快找尋到需要的軟體元件，此為本研究之最主要之貢獻。

3.1.3 環境限制

本系統可讓使用者藉由環境限制(硬體限制、軟體限制)更瞭解元件建立時的限制，而這些限制也是使用者在使用本系統檢索嵌入式元件時條件其中之一部份，一般嵌入式系統，通常都會因為軟硬體方面的限制而對於選取的元件會有所不同，舉例來說，當我們在 Windows 平台進行軟體封裝時，

使用 Winrar 自動解壓縮技術來協助安裝相關套件，但不同系統平台中所能判讀的軟體會有所不同，因此將此安裝元件移動至 Linux 平台時，會因檔案讀取格式的不同而無法順利運作。所以，在找尋元件時加入軟硬體限制是相當重要的，本節會對於軟硬體限制進行詳細說明。

本研究主要依據 BOTS[15]，嵌入式元件在建立的時候必須記錄相關元件的軟硬體限制，而之後重用時，能夠依據這些條件讓使用者更精確的找尋到正確的元件，以下分別對於軟體與硬體限制進行詳細說明。

1. 硬體限制：在硬體限制方面，記錄並說明嵌入式軟體開發時必要的硬體設備，重用嵌入式軟體元件庫系統在尋找元件時，常常因為硬體無法達到軟體元件的要求等問題，而使得元件的重用性(Reusability)變的相當低，為了解決這個問題，本研究以開發嵌入式元件時，所需要的硬體限制記錄下來，將來重用元件時，便不會因為上述的問題而導致嵌入式軟體元件的重用性(Reusability)降低。舉個例子：當嵌入式軟體元件其中一項的硬體限制為記憶體必須大於等於 32KB 時，利用本系統所找到的元件就會符合這一項條件。
2. 軟體限制：在本研究中軟體限制，主要參考 Software Architecture and Embedded System[17]除了硬體限制之外，在軟體限制方面主要分為三項：

   I. Operation System (作業系統)：此為軟體限制最主要參考，因為

O.S 不同所開發的軟體元件，亦不相同，所以使用者在檢索元件時，也會因為 O.S 的不相容而降低了元件的重用性(Reusability)，所以針對嵌入式軟體元件開發時所使用的 O.S 加入考量。

II. Language(程式語言)：元件在建立時，使用不同的語言、呼叫函式甚至到使用的軟體工具與資料庫都會有所不同，為了解決此問題，所以在本研究也記錄元件建立時使用的語言的軟體限制以解決此一問題。

III. Development Software Tools(開發軟體工具)：元件在開發過程中，可能會需要一些軟體開發工具，但是一般使用者找到元件時並不知道使用什麼軟體在協助開發，而將此一限制加入可以幫助使用者解決此問題。

3.2 系統流程

3.2.1 儲存系統流程

在這個嵌入式元件庫系統裡，我們主要著眼於下圖 10 中儲存階段的嵌入式元件介面的呈現，當新的元件或系統在專案中被產出後，把該專案中被特別定義且撰寫出來的元件和該專案整個組成的模組儲存起來，用以當作我們的其中一個重點，也就是呈現方式的基礎，日後檢索元件的時候就以此為依據和參考。

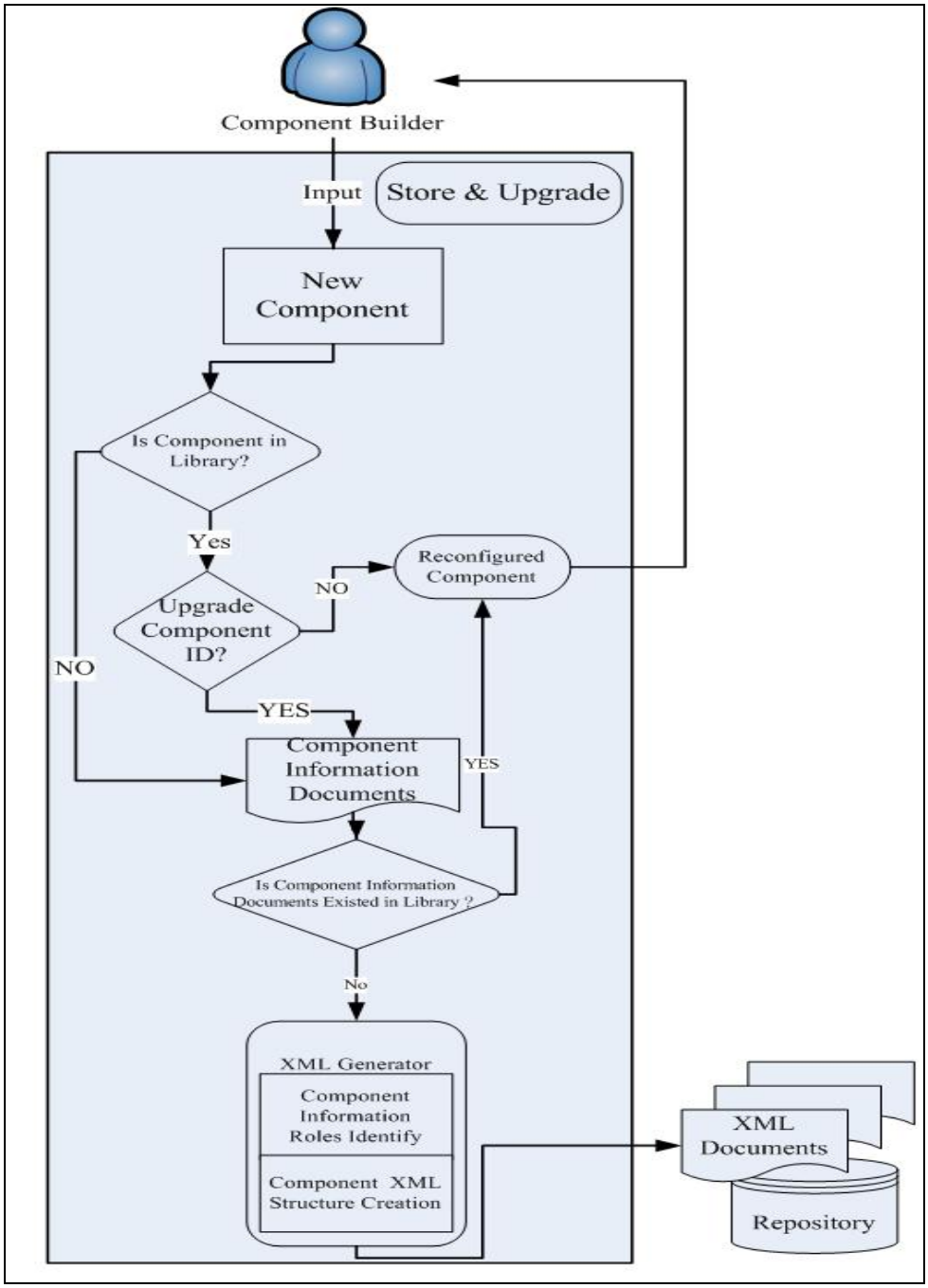


圖 10　儲存系統流程圖

儲存系統流程如下：

Step 1. Is Component Existed in Library？

檢查是否有此元件存在？主要依照元件名稱在應用在檢查元件是否已存在於元件庫中，當使用者輸入不相同名稱，但相關資訊已經存在於資料庫時，會在步驟四時被系統檢查出來。

Step 2. Upgrade Component ID？

當元件確定已存在於 Library 中時，詢問使用者是否以目前的元件與 XML 檔案，對於目前已有之元件進行更新，並詢問相關元件是否修改。

Step 3. Manual Input New Component or System and Component Information Roles Identify

元件建立者根據系統設計文件中所定義的資料輸入描述內容。

Step 4.Is Component Information Documents Existed in Library？

從第一步驟中瞭解，當元件資訊已存在於資料庫中時，在此步驟會被檢查出來，以防止元件資訊重複登錄。

Step 5. Component Information XML Structure Creation

自動產生 XML 檔案，以記錄相關資訊。

Step 6. Store

儲存 XML 文件與軟體元件到 Library 中。

3.2.2 檢索系統流程

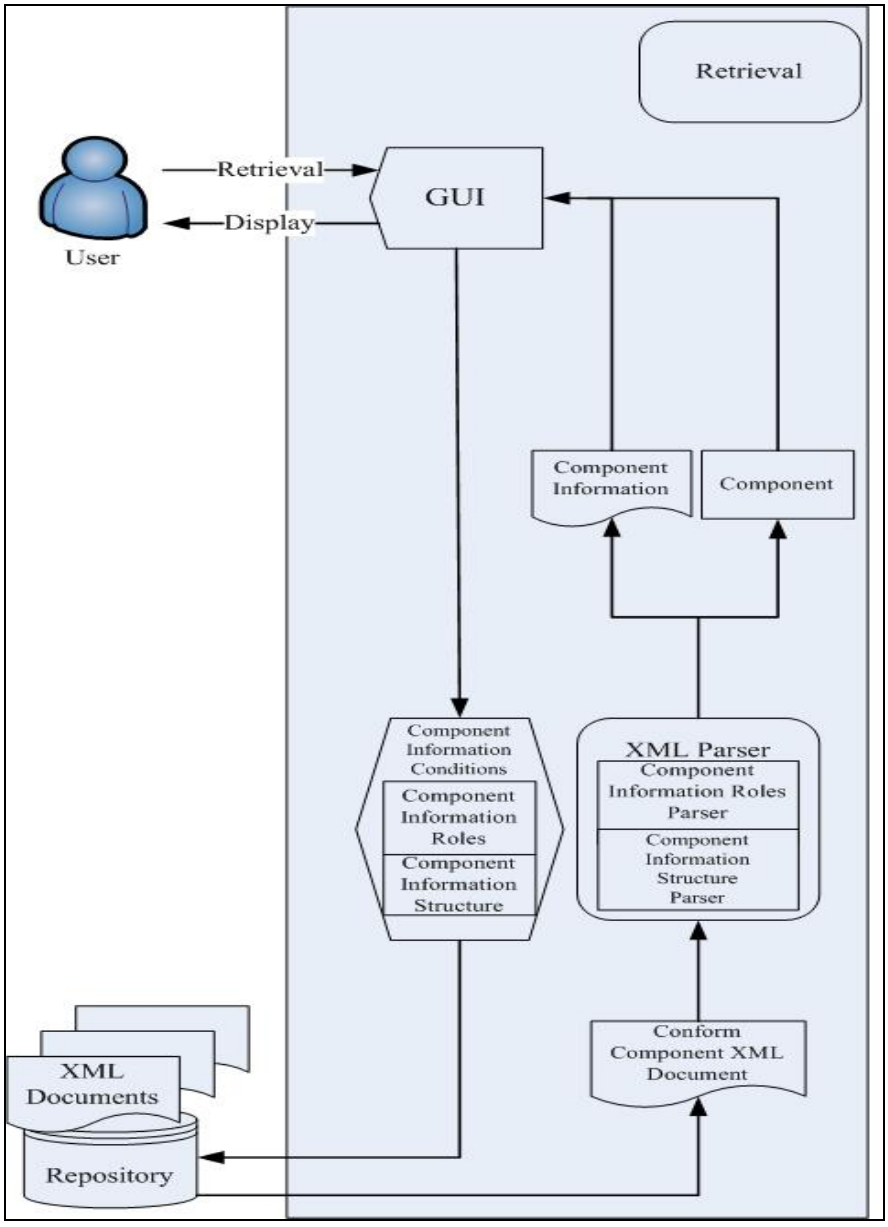


圖 11　檢索系統流程圖

檢索系統流程如下：

Step 1. Input Component Information Roles and Component Information XML Structure

根據圖 11，使用者依據需求條件尋找相關元件時，透過本系統可以找尋到與目標元件相關元件與 XML 檔案，藉由使用者介面(User Interface)輸入使用者所需要的元件描述條件，經由這些描述條件，檢索出符合使用者描述條件的元件。

Step 2. List Conformed Components and Component Information

使用者介面將元件所記錄的資訊呈現，讓使用者更清楚創造元件時所記錄的所有相關元件的描述，除了能夠幫助使用者更容易找到真正需要的元件，也能夠提供資訊讓使用者更快瞭解元件。

支援軟體系統開發部份，我們運用了重用嵌入式元件庫，利用物件導向技術，發展一套將針對嵌入式系統元件儲存工具，透過工具將可利用操作介面快速載入相關程式系統元件，支援軟體中並截取載入該元件描述，讓程式設計者透過元件描述中所提供的資訊，確認是否載入該元件，並透過新增與修改所儲存嵌入式元件，做為前後端各工具間存取資訊之中間存取區。

3.3 XML 標籤對應

在系統軟體生命週期中的所有階段，都會製作相關的文件來記錄各個階段所做的事，或是提供下個階段做為依據；同樣地，即使是採用 CBSD(Component-based Software Development)的開發方式，在元件分析、

需求的修改、系統設計等等活動中也會製作各種文件，其中，元件分析的依據就是參考過去開發元件之時所製作的文件，才有資料輔助使用者透過分析找尋適合當前需要的功能元件。

Component Model 與 XML tag 對應如表 2，本研究依據 ORES(On-line Repository for Embedded System)[9]與 BOTS[15]定義應用領域分類所需依循的標籤及限制和參數問題等描述標籤，並包含元件間溝通關係使用經驗儲存的標籤。

表 2　Component Model 與 XML tag 對應表

Models	Model Elements		Model Attribute	XML Element
Component	Name			<name>
	Identification			<id>
	Relationship Documents	Relationship	Name	<relationship name="">
			Necessity	<relationship necessity="">
			From System	<relationship fromsystem="">
			Annotation	<relationship annotation="">
	Classification Documents	Classification	Language	<classification language="">
			Reference	<classification reference="">
			Annotation	<classification annotation="">
	Parameter Documents	Parameter	Name	<parameter name="">
			Attribute	<parameter attribute="">
			Content	<parameter content="">
			Annotation	<parameter annotation="">
	Constraint Documents	Constraint	Name	<constraint name="">
			Attribute	<constraint attribute="">
			Annotation	<constraint annotation="">

以下為敘述每一個標籤與屬性的內容：

Component：記錄元件完整的資訊。

Name：建立輸入元件名稱。

Identification：辨識號碼(提供編號及辨識功能)，主要在應用在檢查元件

是否已存在於元件庫中。範例說明：1-1，前碼為元件輸入時所編的號碼，後碼為版本的編號，如果需要更新元件，則將編碼加一號為 1-2。

Relationship Documents：包含 Relationship 內容

Relationship：一份關聯文件，屬性包含以下。

Name：與輸入元件相關聯元件名稱。

Necessity：相關聯之元件存在於系統之必要性。

From System：元件所屬的系統名稱。

Annotation：敘述相關聯元件與輸入元件之關聯備註。

Classification Documents：包含 Classification 內容

Classification：一份分類文件，屬性包含以下。

Language：選取程式語言所參考的程式庫。

Reference：以此方法將元件以 Java API(Application Programming Interface)、C (ISO/IEC9899:1999)標準程式庫和 C++ (ISO/IEC 14882:1998)標準程式庫參考為基礎[3][10][18]，以比對分類的資訊。

Annotation：分類註解。

Parameter Documents：包含 Parameter 內容

Parameter：一份參數文件，屬性包含以下。

Name：參數名稱。

Attribute：輸入/輸出參數。

Content：參數內容。

Annotation：參數註解。

Constraint Documents：包含 Constraint 內容

Constraint：一份限制文件，屬性包含以下。

Name：限制名稱。

Attribute：軟體/硬體限制

Software：開發時軟體限制(如：開發應用到的 O.S、開發使用的程式語言、必要使用的軟體工具)。

Hardware：開發時的硬體限制(當開發程式時，必要的硬體有哪些或是在硬體上的容量限制等等...，例如：如在硬體限制上 SDRAM 必須大於等於 32KB，則需輸入此一項目)。

Annotation：限制註解。

第四章 研究範例

4.1 系統開發環境

本研究所開發的以 XML 為基底之重用嵌入式軟體元件庫系統，利用 C# 進行系統開發，運用 Visual Studio 2008 開發工具協助編輯本系統，並運用 SQL Server 2008 作為後端資料庫進行本研究中所需的系統開發環境，如下圖 12。

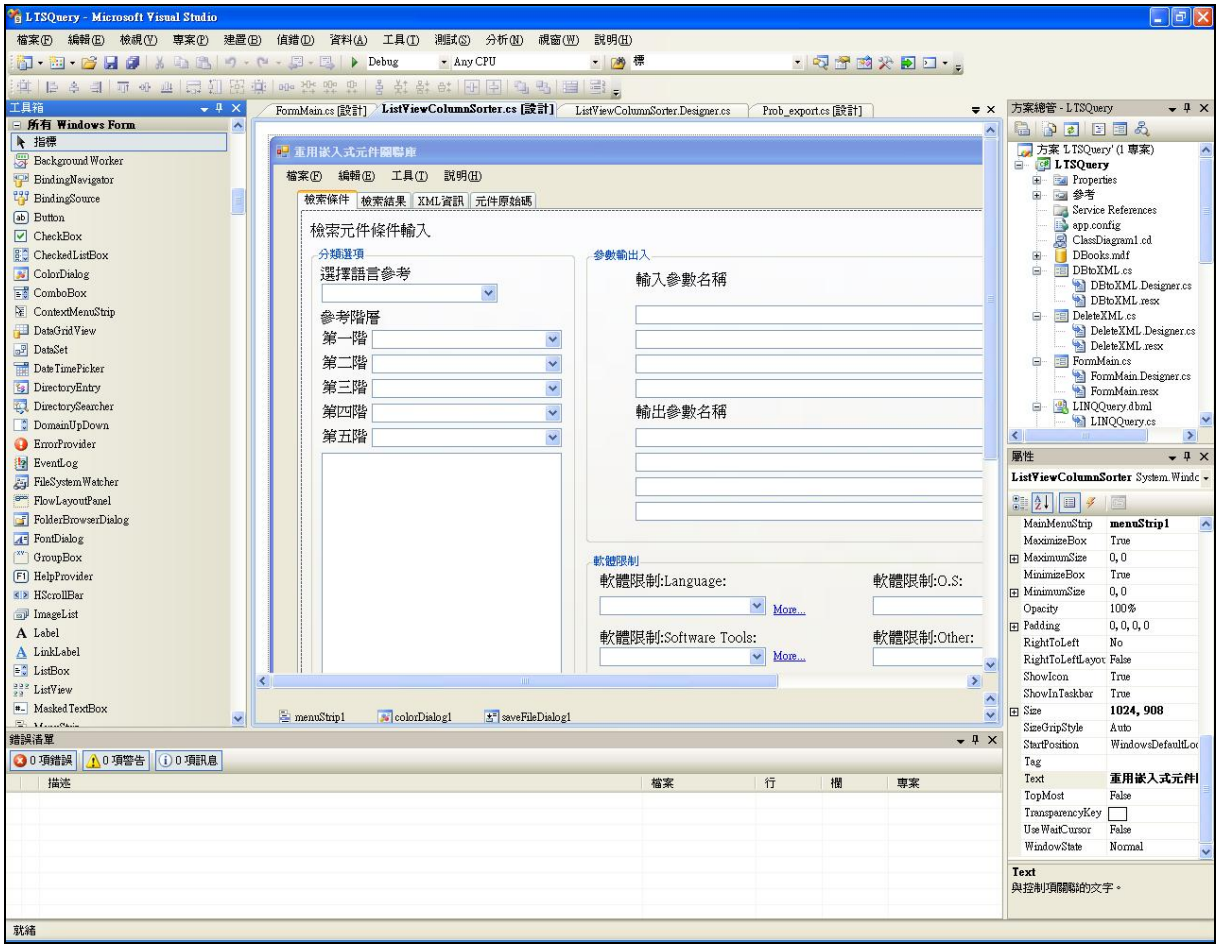


圖 12　研究平台開發環境畫面

4.2 系統文件儲存

本研究範例是使用符合樂高(LEGO)公司所開發的標準教學模型 LEGO MINDSTORMS NXT 機器人開發套件(公司產品編號:9797)軟體系統。本範例以 LEGO MINDSTORMS NXT 設計一套圖形化介面遙控程式，其系統名稱為 RobotControl，主要控制機器人功能：計算前方障礙物距離、前進、後退、左轉、右轉、停止、開始偵測、停止偵測、速度設定、結束程式，其中每一項功能都為一個嵌入式軟體元件，以下根據 3.2.1 節儲存系統流程之步驟來說明本研究方法。

Step1：Is Component Existed in Library？

檢查元件是否已在資料庫中存在，並免重複輸入資料，同時載入元件原始程式依同輸入到元件資料庫之中，如下圖 13。當元件已存在資料庫中時，系統會詢問是否更新元件？當選擇不更新時，則必須重新定義文件。

圖 13　系統介面圖(登入畫面)

Step2：Upgrade Component ID？

當按下查詢鍵後，會直接對資料庫進行檢查，是否已有符合的元件存在，當元件存在時，系統會詢問元件建立者是否對於元件進行更新的動作？將元件編號進行更新，如下圖 14。

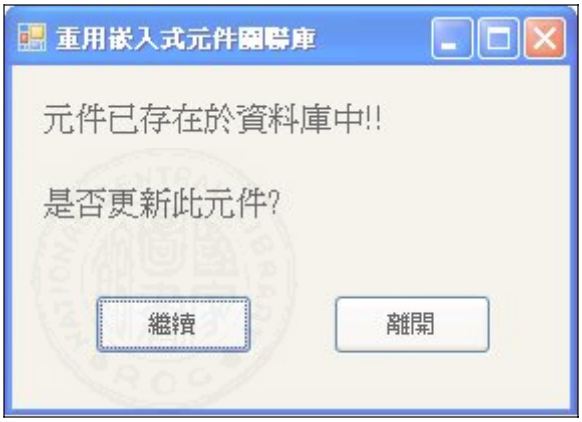


圖 14　系統介面圖(更新檢查)

Step 3. Manual Input New Component or System and Component Information Roles Identify

輸入 Relationship Documents：

本研究範例主要全部都是來自 RobotControl 系統的元件，所以其主要的關聯也是來自 RobotControl 系統。參考下表 3，本研究範例以其中一項 jBtn_Left 元件表示整個重用嵌入式軟體元件庫系統的操作流程，並將上述的 RobotControl 系統元件的關聯性記錄下來，如下圖 15。

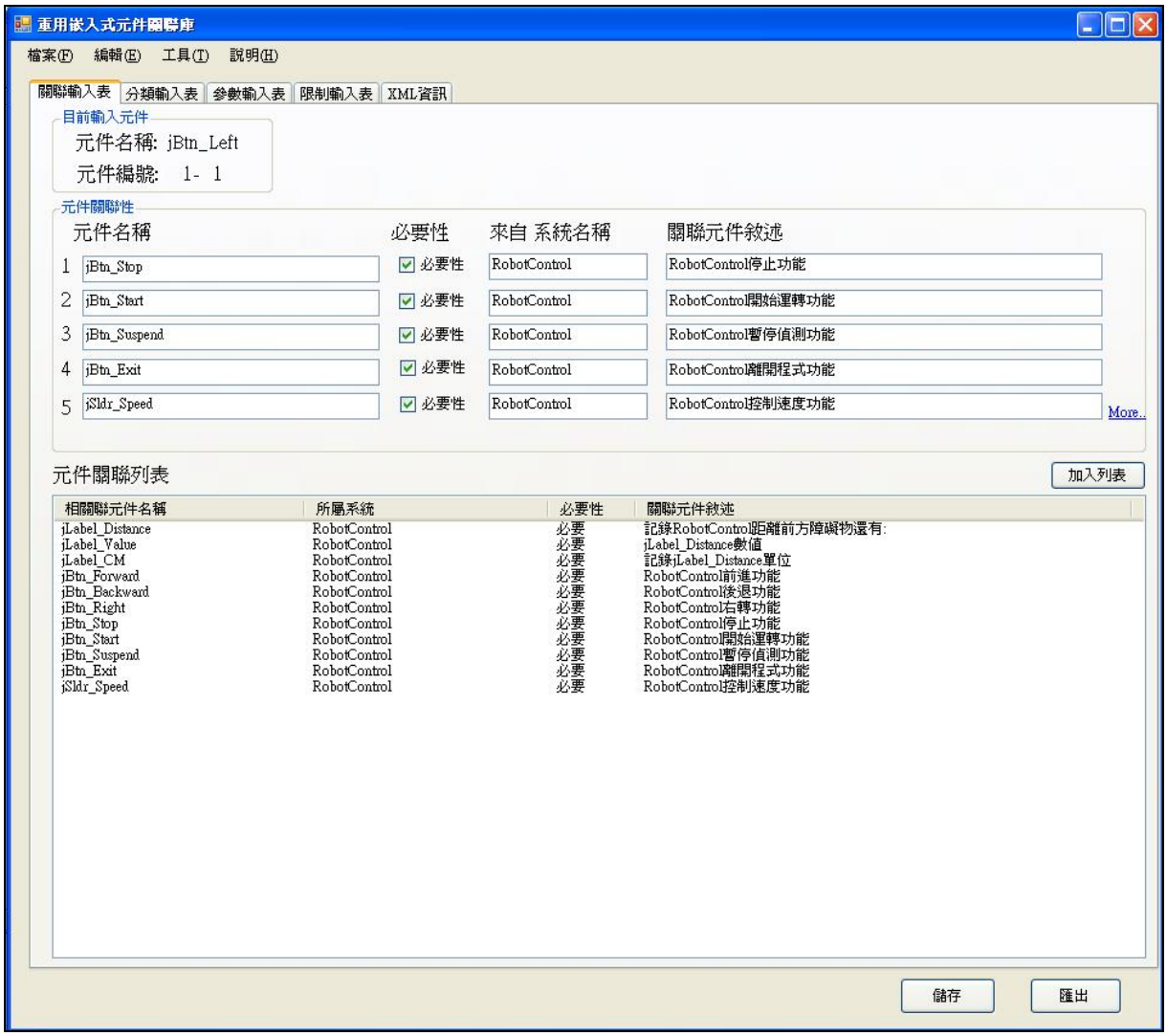


圖 15　系統介面圖(Relationship)

輸入 Classification Documents：

如下表 3，是圖形化介面遙控程式的所有的元件的分類表，經由本

研究記錄元件所使用的 JAVA API、C 和 C++標準程式庫參考如下圖16，由本系統選取語言的 JAVA API、C 和 C++標準程式庫參考進而將元件進行分類，當元件加入了哪些 JAVA API、C 和 C++標準程式庫參考就會加入所選取的類別，讓使用者將來檢索元件時，依據此分類來找尋真正需要的元件。

表 3　研究範例系統元件分類表

元件名稱	元件分類別	元件說明
jLabel_Distance	javax.swing.JLabel	前方障礙物距離
jLabel_Value	javax.swing.JLabel	數值
jLabel_CM	javax.swing.JLabel	公分 (單位)
jBtn_Forward	java.awt.event.ActionEvent	前進
jBtn_Backward	java.awt.event.ActionEvent	後退
jBtn_Left	java.awt.event.ActionEvent	左轉
jBtn_Right	java.awt.event.ActionEvent	右轉
jBtn_Stop	java.awt.event.ActionEvent	停止
jBtn_Start	java.awt.event.ActionEvent	開始偵測
jBtn_Suspend	java.awt.event.ActionEvent	停止偵測
jBtn_Exit	java.awt.event.ActionEvent	離開程式
jSldr_Speed	javax.swing.JSlider java.swing.event.ChangeEvent	速度設定

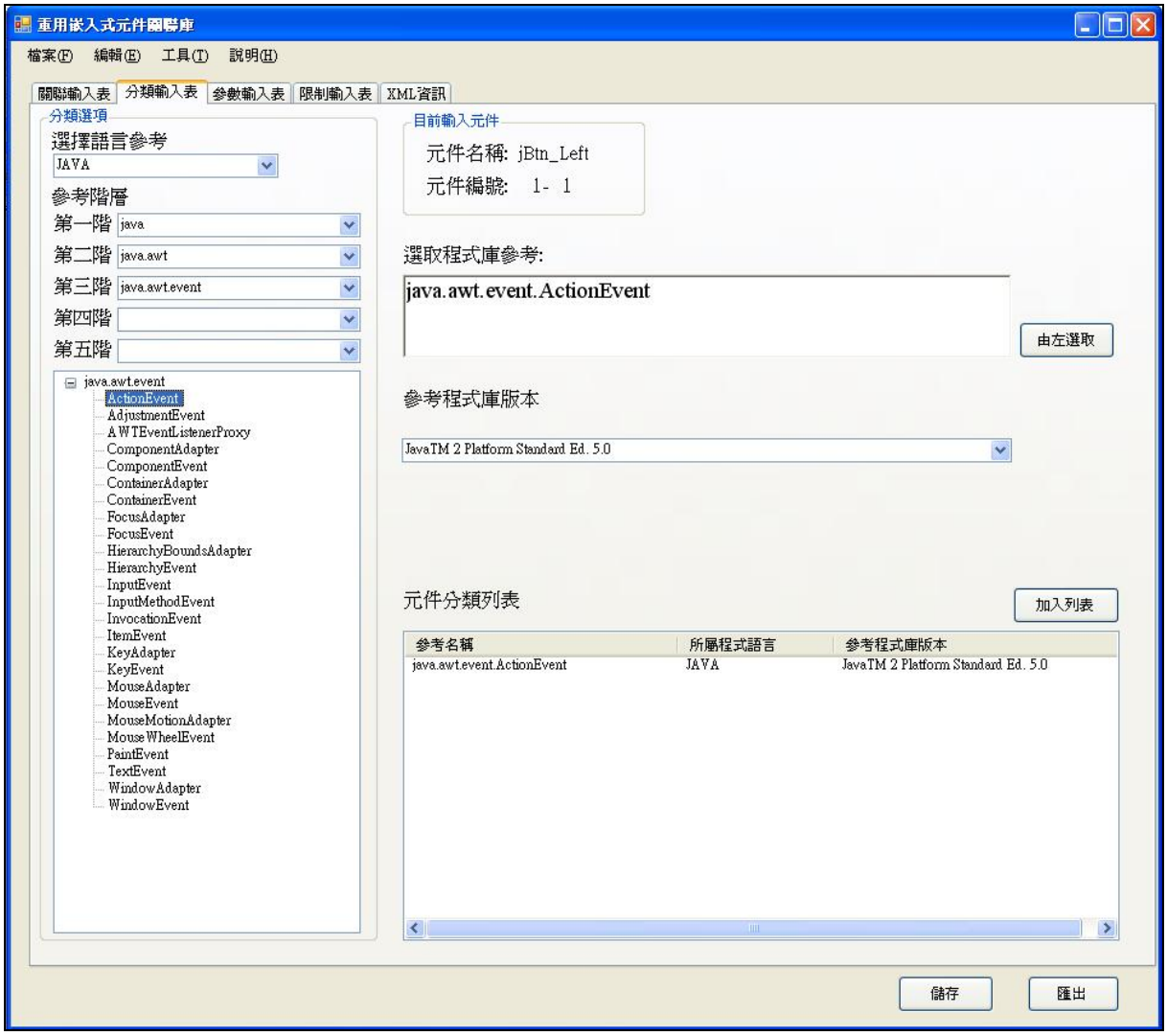


圖 16　系統介面圖(Classification)

輸入 Parameter Documents：

本研究中將元件輸入參數與輸出參數分別記錄起來，記錄元件處理的參數。如下圖 17 中，在範例中所需要的輸出參數分別為 setIcon：圖形路徑/icons/Left.gif、setBounds：矩形圖形規格 Rectangle (16, 99, 63, 66) 和 addActionListener：顯示字串(左轉) & 函式 Motor.Right.forward()，將這些資訊記錄到文件當中，在後面檢索元件時也是檢索的條件之一，並讓使用者瞭解元件是否能夠符合解決使用者問題的需求。

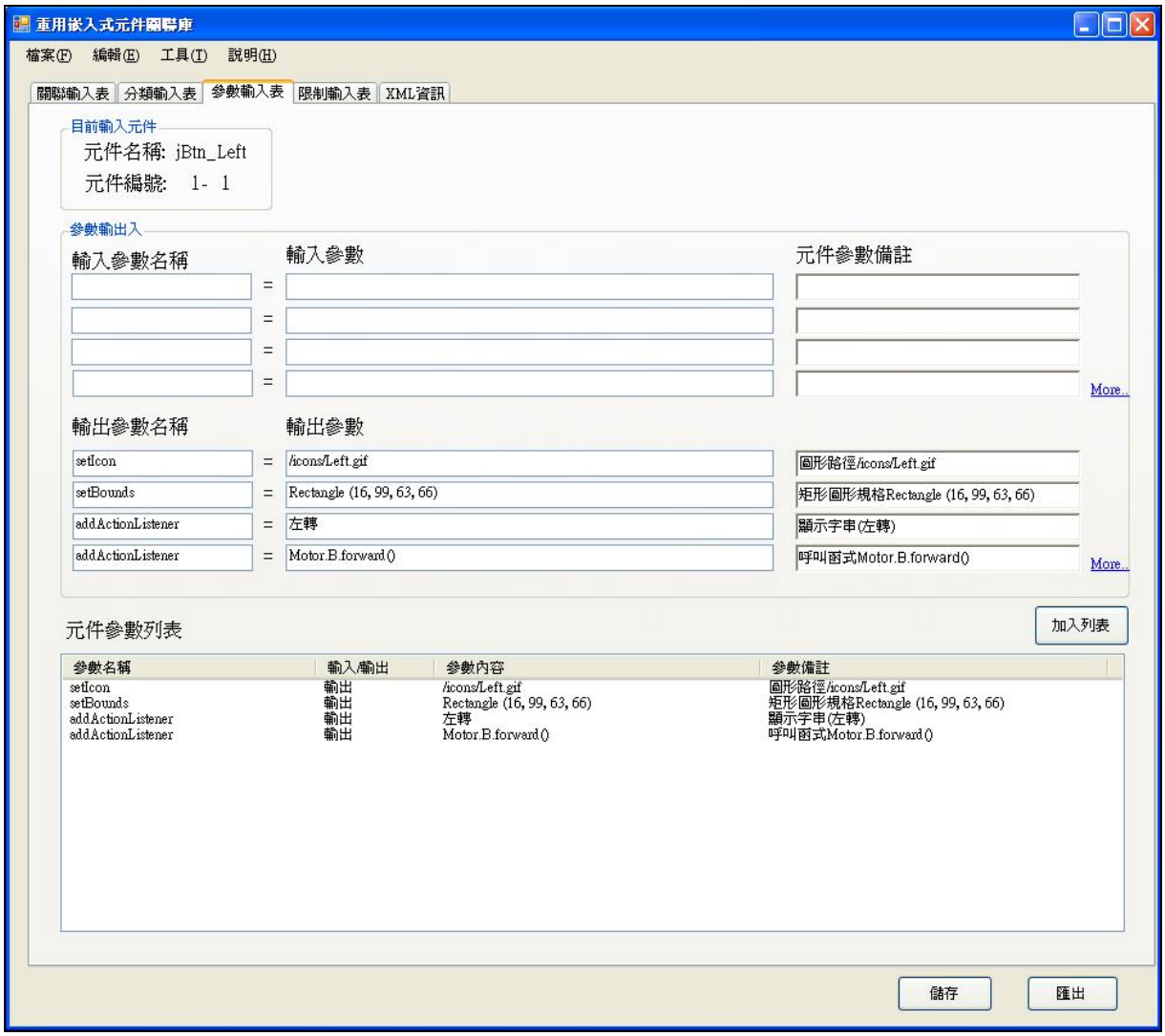


圖 17　系統介面圖(Parameter)

輸入 Constraint Documents：

本研究範例是使用標準教學模型 LEGO MINDSTORMS NXT 機器人開發套件(公司產品編號:9797)、CPU: Intel Pentium 至少 800MHz 和 Memory:至少 256MB 為硬體限制。軟體限制方面，本研究範例所使用之作業系統(O.S)為 Microsoft Windows，語言選擇為 JAVA，如下圖 18 為本研究範例之環境限制文件。

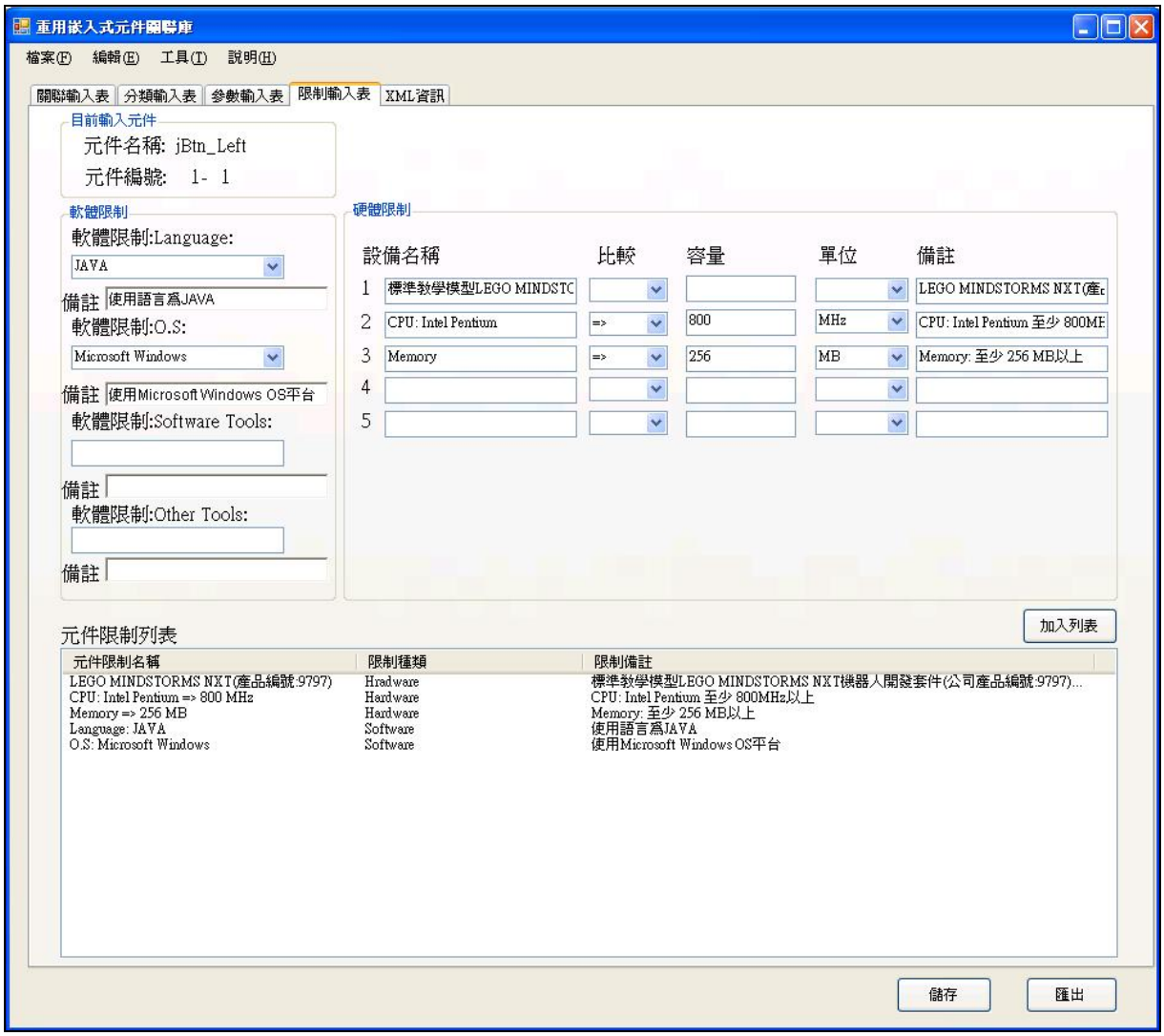


圖 18　系統介面圖(Constraint)

Step 4. Is Component Information Documents Existed in Library？

當完成文件的建立，按下匯出鍵時，系統會先對元件建立者所輸入的資料與資料庫中的資料進行比對，當重複輸入時，會在匯出時被檢查出來，如下圖 19。

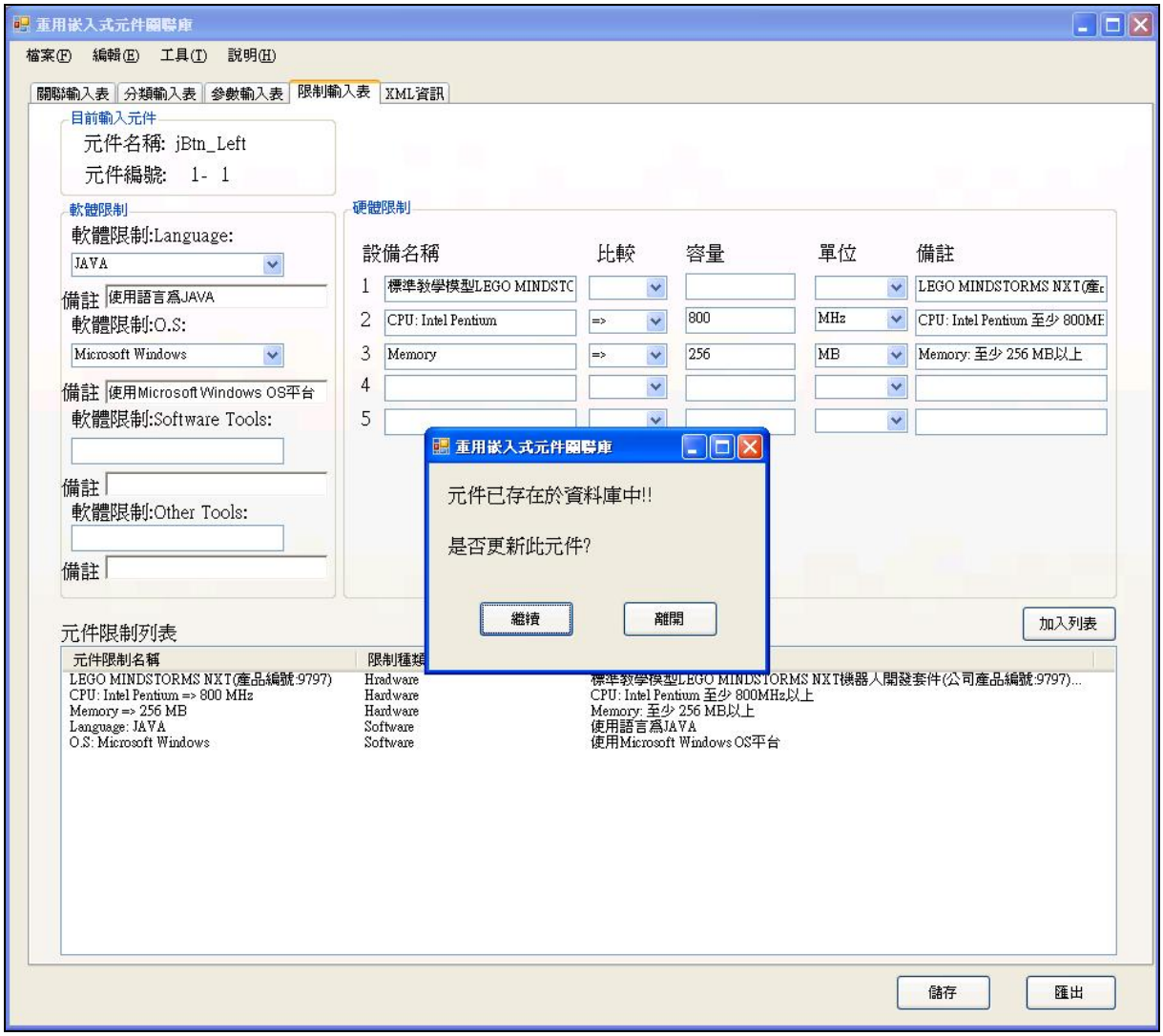


圖 19　系統介面圖(元件檢查)

Step 5. Component Information XML Structure Creation

本範例中的 LEGO MINDSTORMS NXT 軟體的系統在開發完成後將會被記錄下來，透過我們的系統建立一份 XML 文件，如下圖 20、圖 21 所示，可以清楚的把系統資訊記錄下來，當我們要檢索的時候就可以根據標籤中的項目去尋找，電子化地操作減少了不少人工的作業，以前的元件庫系統並提供較少這類的服務，所以當找到元件時，卻不知道該去什麼地方找系統文件的參考，在本論文中提出這個問題描述系統

的方法，希望可以透過輸入資料的方式，快速找尋到需要的軟體元件，並提高嵌入式軟體系統的開發效率。

相同地，基本的嵌入式元件問題描述，也可以透過我們的系統來描述，並把該元件的文件打包在一起，產生如下圖 20、圖 21 中的 XML 文件，這是以一個 LEGO MINDSTORMS NXT 機器人開發套件(公司產品編號:9797)的 RobotControl 系統中的 jBtn_Left 元件開發作表示，說明以上四個部份的文件資訊。這些都可以作為檢索的依據，增加檢索的正確性，減少檢視程式碼的需要。

圖 20　XML 範例檢視(網頁檢視)

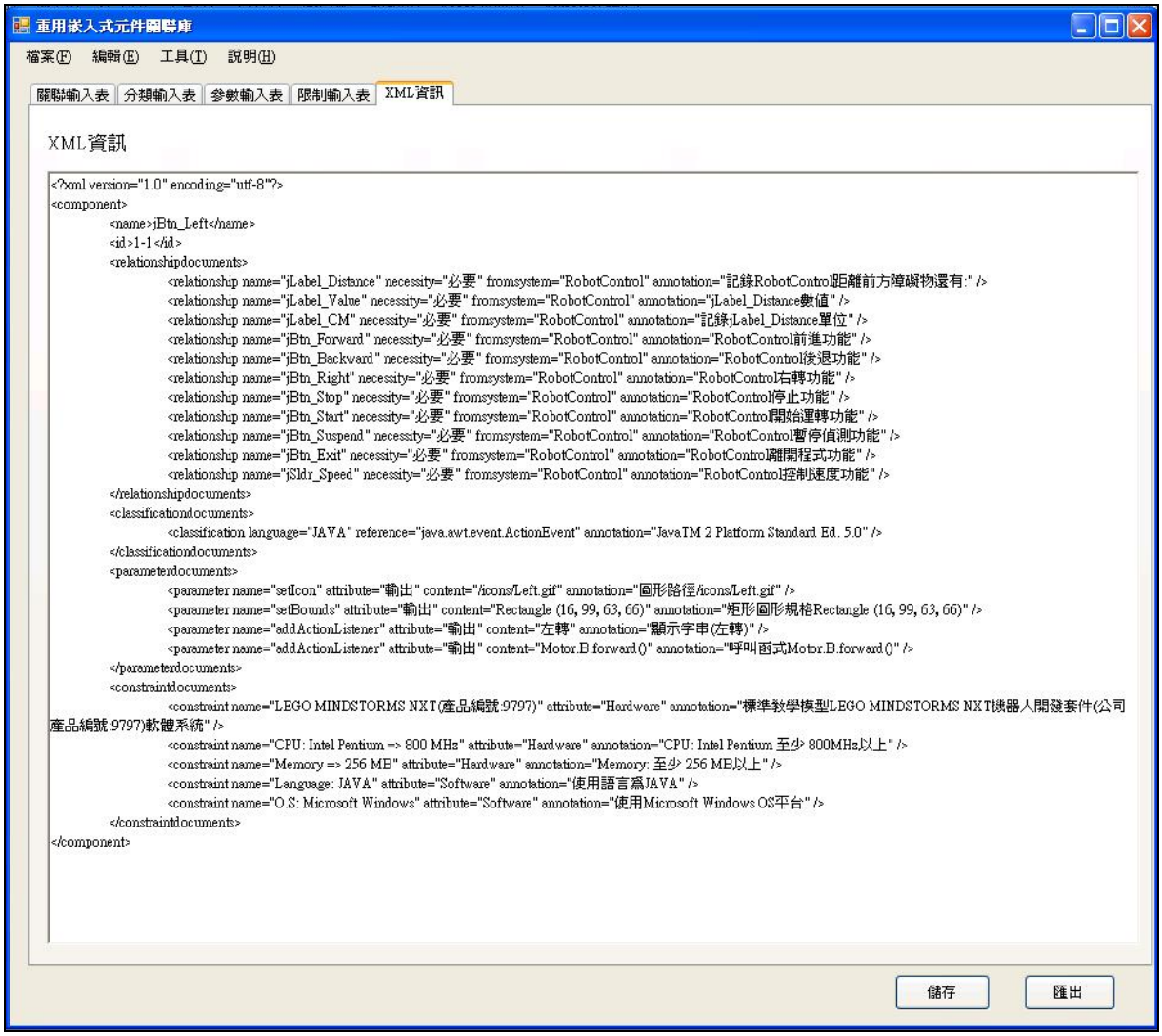


圖 21　XML 範例檢視(系統檢視)

Step 6. Store

當元件相關資訊尚未被建立在資料庫中時，系統會將使用者輸入之相關文件儲存到資料庫中，如下圖 22。

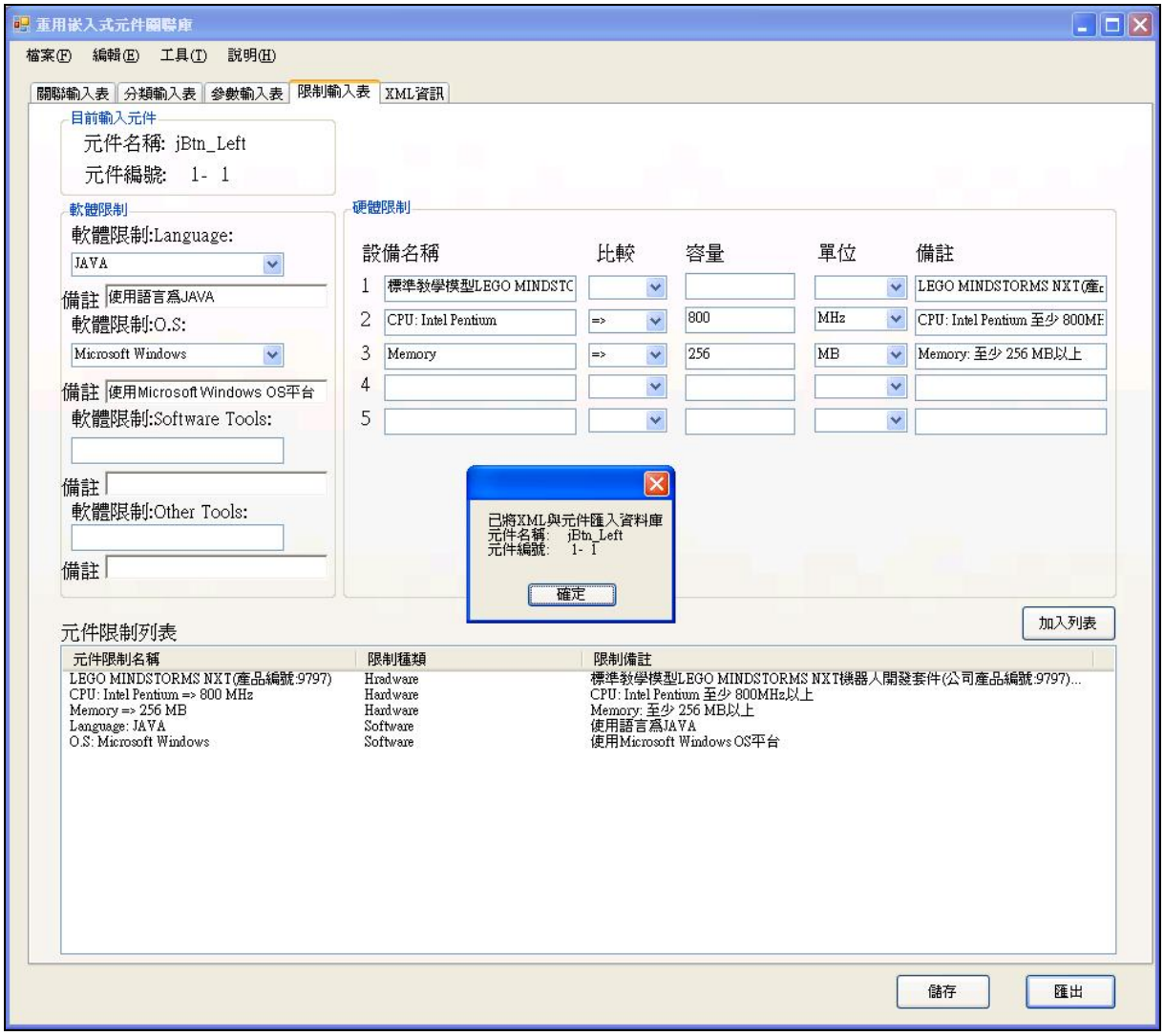


圖 22　系統介面圖(儲存元件)

4.3 系統元件檢索

檢索系統根據 3.2.2 節中，檢索系統流程的步驟來進行資料的檢索，其步驟如下。

Step 1. Input Component Information Roles and Component Information XML Structure

系統元件檢索主要依據系統文件描述來進行條件檢索，利用本研究

提供的使用者介面輸入範例檢索條件，如下圖 23，利用上述制定的四項文件提供元件檢索所需要的條件，來檢索 jBtn_Left 元件。

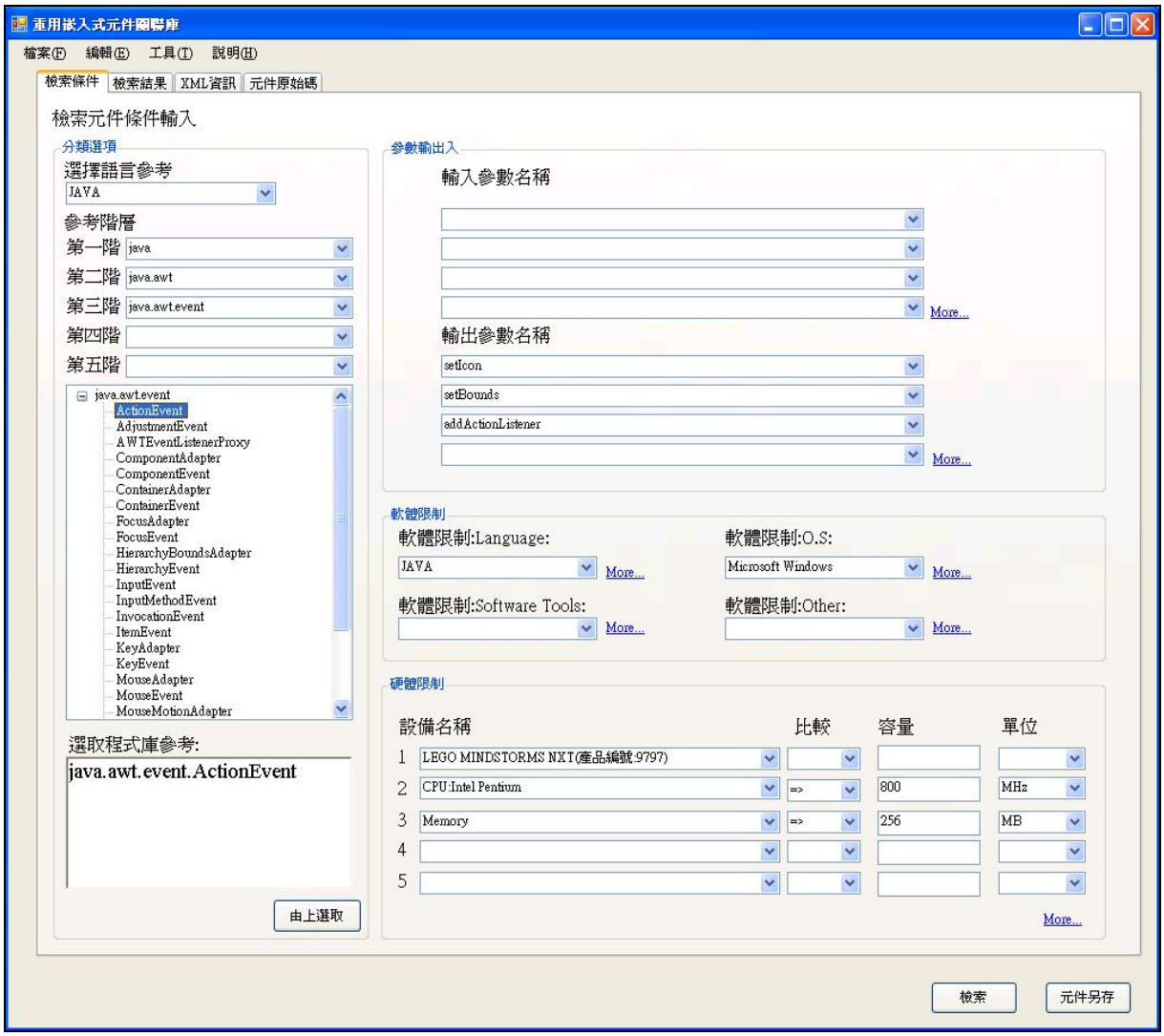


圖 23　系統檢索圖(條件輸入)

Step 2. List Conformed Components and Component Information

依據下表 4 中的檢索條件輸入到系統檢索畫面中，並且按下檢索鍵，即可檢索到所有符合輸入條件限制的元件，當檢索元件出現多個以上時，可以由系統檢視畫面中查詢元件的相關資訊，快速找到符合使用

者需求的元件，並且快速瞭解如何運用元件，如下圖 24、圖 25、圖 26。

表 4　jBtn_Left 元件檢索條件

檢索條件選項	檢索條件內容
API 分類	java.awt.event.ActionEvent
硬體限制	LEGO MINDSTORMS NXT(產品編號:9797) CPU: Intel Pentium 至少 800MHz 以上 Memory: 至少 256 MB 以上
軟體限制	Language: JAVA O.S: Microsoft Windows
參數	輸入參數: setIcon、setBounds 和 addActionListener 輸出參數: setIcon：圖形路徑/icons/Left.gif setBounds：矩形圖形規格 Rectangle (16, 99, 63, 66) addActionListener：顯示字串(左轉) & 函式 Motor.Right.forward()

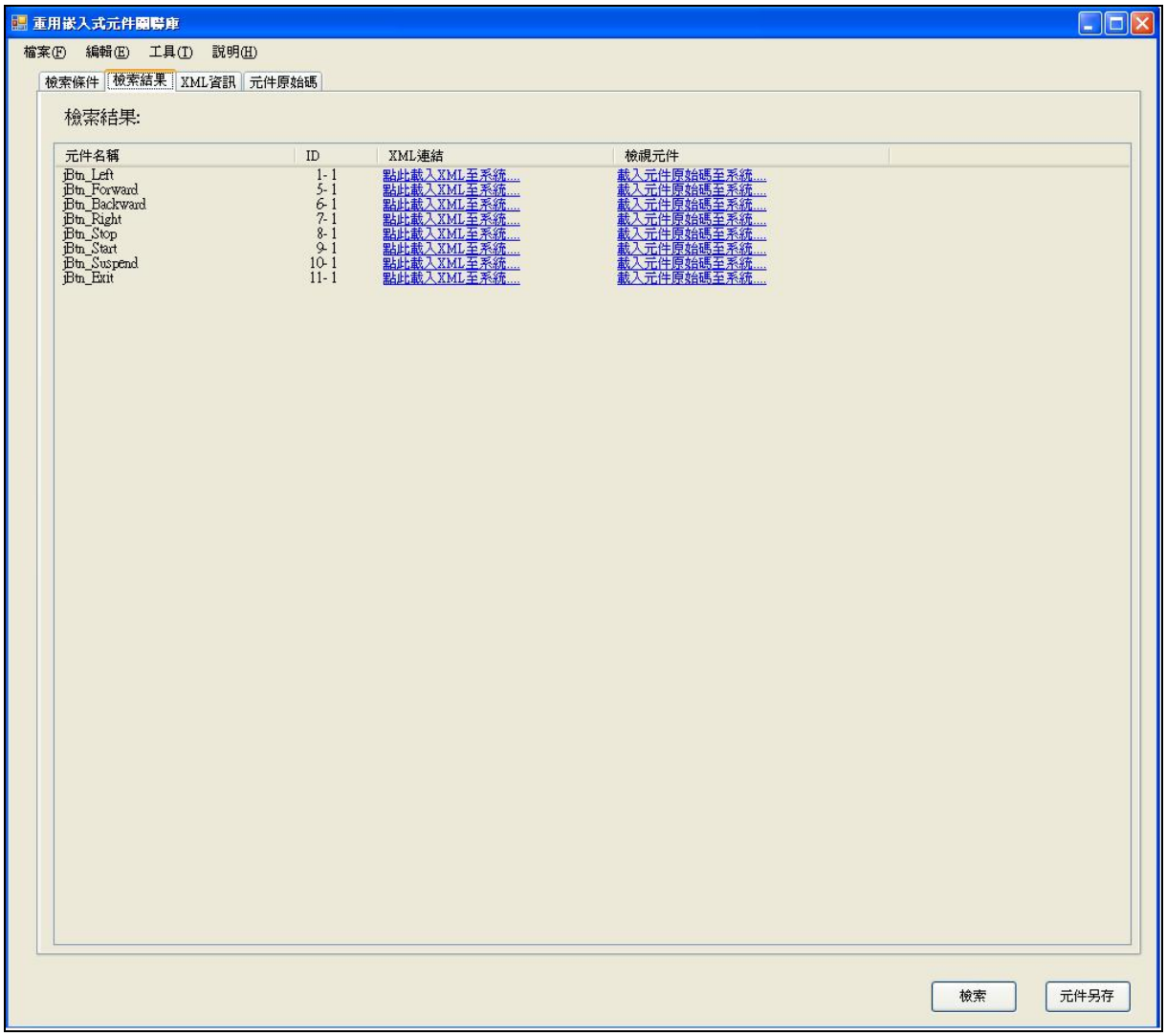
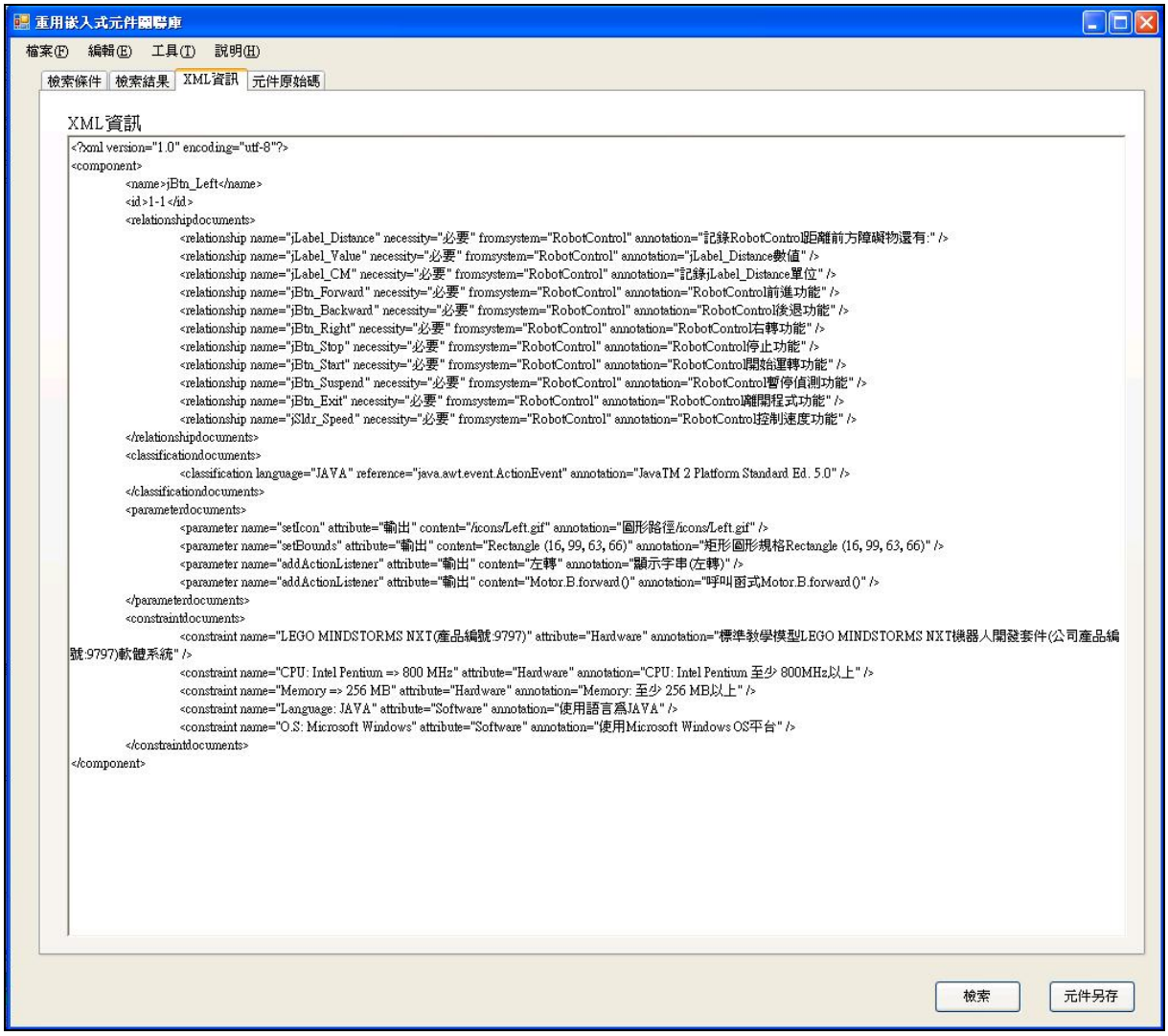


圖 24　系統檢索圖(檢索結果)




```
<?xml version="1.0" encoding="utf-8"?>
<component>
        <name>jBtn_Left</name>
        <id>1-1</id>
        <relationshipdocuments>
                <relationship name="jLabel_Distance" necessity="必要" fromsystem="RobotControl" annotation="記錄RobotControl距離前方障礙物還有:" />
                <relationship name="jLabel_Value" necessity="必要" fromsystem="RobotControl" annotation="jLabel_Distance數值" />
                <relationship name="jLabel_CM" necessity="必要" fromsystem="RobotControl" annotation="記錄jLabel_Distance單位" />
                <relationship name="jBtn_Forward" necessity="必要" fromsystem="RobotControl" annotation="RobotControl前進功能" />
                <relationship name="jBtn_Backward" necessity="必要" fromsystem="RobotControl" annotation="RobotControl後退功能" />
                <relationship name="jBtn_Right" necessity="必要" fromsystem="RobotControl" annotation="RobotControl右轉功能" />
                <relationship name="jBtn_Stop" necessity="必要" fromsystem="RobotControl" annotation="RobotControl停止功能" />
                <relationship name="jBtn_Start" necessity="必要" fromsystem="RobotControl" annotation="RobotControl開始運轉功能" />
                <relationship name="jBtn_Suspend" necessity="必要" fromsystem="RobotControl" annotation="RobotControl暫停偵測功能" />
                <relationship name="jBtn_Exit" necessity="必要" fromsystem="RobotControl" annotation="RobotControl離開程式功能" />
                <relationship name="jSldr_Speed" necessity="必要" fromsystem="RobotControl" annotation="RobotControl控制速度功能" />
        </relationshipdocuments>
        <classificationdocuments>
                <classification language="JAVA" reference="java.awt.event.ActionEvent" annotation="JavaTM 2 Platform Standard Ed. 5.0" />
        </classificationdocuments>
        <parameterdocuments>
                <parameter name="setIcon" attribute="輸出" content="/icons/Left.gif" annotation="圖形路徑/icons/Left.gif" />
                <parameter name="setBounds" attribute="輸出" content="Rectangle (16, 99, 63, 66)" annotation="矩形圖形規格Rectangle (16, 99, 63, 66)" />
                <parameter name="addActionListener" attribute="輸出" content="左轉" annotation="顯示字串(左轉)" />
                <parameter name="addActionListener" attribute="輸出" content="Motor.B.forward()" annotation="呼叫函式Motor.B.forward()" />
        </parameterdocuments>
        <constraintdocuments>
                <constraint name="LEGO MINDSTORMS NXT(產品編號:9797)" attribute="Hardware" annotation="標準教學模型LEGO MINDSTORMS NXT機器人開發套件(公司產品編
號:9797)軟體系統" />
                <constraint name="CPU: Intel Pentium => 800 MHz" attribute="Hardware" annotation="CPU: Intel Pentium 至少 800MHz以上" />
                <constraint name="Memory => 256 MB" attribute="Hardware" annotation="Memory: 至少 256 MB以上" />
                <constraint name="Language: JAVA" attribute="Software" annotation="使用語言為JAVA" />
                <constraint name="O.S: Microsoft Windows" attribute="Software" annotation="使用Microsoft Windows OS平台" />
        </constraintdocuments>
</component>
```

圖 25　系統檢索圖(XML 資訊)

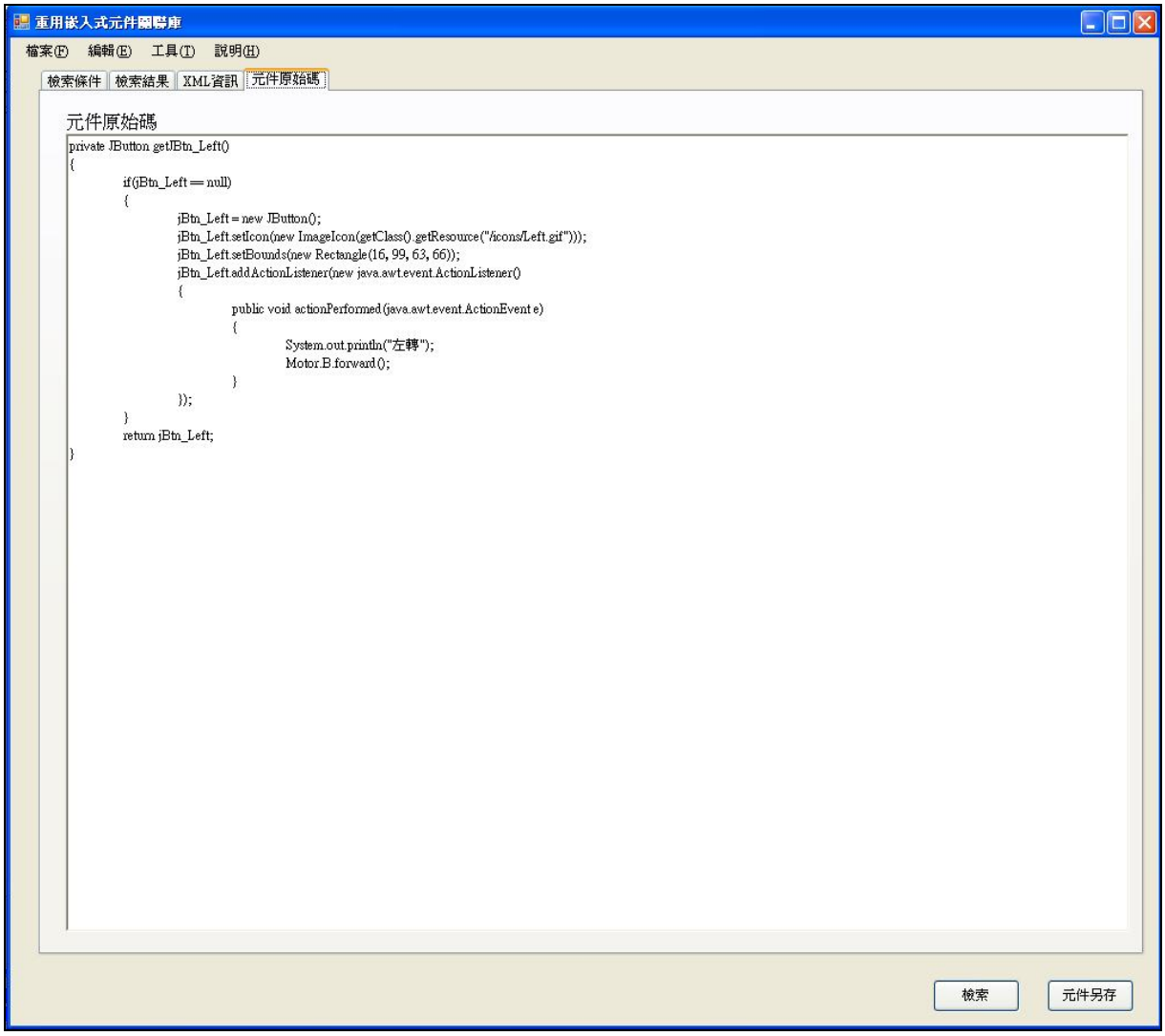


圖 26　系統檢索圖(原始程式碼)

第五章 結論與未來工作

本研究依據標準來產生的 XML 檔案不但記錄與元件有相關的資訊，也能夠利用這些資訊來檢索元件，能夠讓使用者找到更符合使用者需求的元件。當檢索到元件時，經由 XML 文件提供關聯資訊，同時也可以找到與找尋的元件相關聯的其他元件，這樣不僅減少使用者找尋其他元件的時間，亦可達到整個嵌入式軟體的快速開發的目標。

本研究最大貢獻之處是讓系統使用者不僅利用 XML 文件所呈現的畫面來檢索合適的元件，並且利用這些文件所呈現的畫面提供使用者當元件建立時的相關資訊，讓使用者對元件瞭解時，能夠簡單且快速的呈現資訊。讓當初建立元件時的經驗與實作能夠被重用，這樣不但可以讓嵌入式軟體開發者更快開發軟體系統，而且能夠達到嵌入式軟體「更低成本」的需求，這也正是一開始本研究最主要的研究目的。

目前在語言參考程式資料庫，只加入了一般嵌入式經常使用的 C、C++ 和 JAVA 所屬的標準程式資料庫。在未來的工作上，在各個程式語言建立所屬的標準程式資料庫是目前的趨勢，如目前的 C#、.NET 等語言都有建立屬於自己的標準程式資料庫，將來可以將其他所屬的語言標準程式資料庫加入系統參考。

參考文獻

[1] Christo Angelov and Krzysztof Sierszecki, “A Software Framework for Component-based Embedded Applications,” *Software Engineering Conference,* 2004, 11th Asia-Pacific, Nov 2004, pp. 655-662.

[2] Tim Bray, Jean Paoli, Michael Sperberg-McQueen, François Yergeau and Eve Maler, *Extensible Markup Language (XML) 1.0(Fifth Edition)*, W3C, 2008.

[3] Peter Prinz and Tony Crawford, *C In Nutshell: A Desktop Quick Reference*, O’Reilly, Dec 2005.

[4] Peter Coad and Edward Yourdon, *OOA – Object-oriented Analysis (Second Edition)*, Prentice Hall, 1990.

[5] William B. Frakes and Paul B. Gandel “Classification, Storage and Retrieval of Reusable Components,” *in Proc. SIGIR ’89*, 1989, pp.251-254.

[6] Erich Gamma, Richard Helm, Ralph Johnson and John M. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[7] Hermann Kaindl, “Difficulties in the Transition from OO Analysis to Design,” *IEEE Software*, Vol. 16, No.5, Sep 1999, pp. 94-102.

[8] Ian Sommerville, *Software Engineering (Seventh Edition)*, Addison Wesley, 2004.

[9] I-Ling Yen, Jayabharath Goluguri, Farokh Bastani and Latifur Khan, “A Component-based Approach for Embedded Software Development,” *Proc. IEEE ISORC*, 2002, pp. 402-412.

[10] Elliotte Rusty Harold, *JAVA Networking Programming (Third Edition)*, O’Reilly, Oct 2004.

[11] Ralph Johnson, “Framework = (Components + Patterns),” *Communications*

of the ACM (CACM), Vol. 40, No. 10, Oct 1997, pp. 39-42.

[12]Wojtek V. Kozaczynski and Jim Q. Ning, "Component-Based Software Engineering (CBSE)," *fifth Intl'Conf. On Software Reuse*, April 23-26, 1996, pp. 236-241.

[13]Yoelle S. Maarek, Daniel M. Berry and Gail E. Kaiser, "An Information Retrieval Approach For Automatically Constructing Software Libraries," *IEEE Transactions on Software Engineering*, Vol. 17, No. 8, 1991, pp. 800-813.

[14]Maninder Singh and Shivani Goel, "Identifying Asset Type for Various Reusable Component Storage/Retrieval Methods," *Proceedings of National Conference on Challenges & Opportunities in Information Technology*, RIMT-IET, Mar 23, 2007, pp. 38-41.

[15]Raju Pandey and Jeffrey Wu, "BOTS: A Constraint-based Component System for Synthesizing Scalable Software Systems," *Proceedings of the International Conference on Languages, Compilers and Tools for Embedded Systems (LCTES)*, ACM Press, 2006, pp. 189-198.

[16]Thomas Genßler and Christian Zeidler, "Rule-Driven Component Composition for Embedded Systems," *Proceedings of the Foul1h ICSE Workshop on Component-Based Software Engineering*, May 2001, pp. 4-14.

[17]Nenad Medvidovic, "Software Architectures and Embedded Systems: A Match Made in Heaven?," *IEEE Software*, Sept/Oct 2005, pp. 83-86.

[18]Ray Lischner, *C++ In A Nutshell*, O'Reilly, May 2003.

[19]Mitchell Lubars, Colin Potts and Charles Richter, "Developing Initial OOA Models," *In Proceedings of the Fifteenth International Conference on Software Engineering (ICSE-15)*, IEEE Computer Society Press, 1993, pp255-264.

[20]Ricardo Baeza-Yates and Berthier Ribeiro-Neto, *Modern Information*

Retrieval, Addison Wesley, 1999.

[21]Vijayan Sugumaran and Veda C. Storey, "A Semantic-based Approach to Component Retrieval," *The DATA BASE for Advances in Information System*, Vol. 34, No. 3, 2003, pp. 8-24.

[22]Cal Henderson, *Building Scalable Web Sites*, O'Reilly, May 2006.

[23]Erik T. Ray, *Learning XML*, O'Reilly, May 2001.

[24]Eric A. Meyer, *CSS: The Definitive Guide (Third Edition)*, O'Reilly, November 2006.

[25]Eduardo Ostertag, James Hendler, Ruben Prieto-Diaz and Christine Braun, "Computing Similarity in a Reuse Library System: An AI-based Approach," *ACM Transactions on Software Engineering and Methodology*, Vol. 1, No. 3, 1992, pp. 205-228.

[26]Patrick J. Hayes and Leora Morgenstern, "On John McCarthy's 80th Birthday, in Honor of His Contributions," *American Association for Artificial Intelligence*, Vol. 28, No. 4, 2007, pp. 93-102.

[27]James W. Hooper and Rowena O. Chester, *Software Reuse Guidelines and Methods*, Springer, May 1991.