私立東海大學資訊工程研究所

碩士論文

指導教授：朱正忠　教授

以 XML 為基礎之嵌入式軟體再利用元件庫

XML-based Embedded Software

Reusable Component Repository

研究生：趙庭翊

中華民國九十九年六月

摘要

在嵌入式系統開發中，市場的需求以及系統發展趨勢會受到開發時程(Time To Market)的影響，快速的將系統實作並可靠是件不容易的事情，且在產品價格減少系統開發時程將可提高產品價值。若能減少系統開發時程，將可減少開發成本，進而提高產品價值。本研究以減少開發元件所需的時間，來達到縮短開發時程、增加可靠性及降低處理風險等好處。

在選擇軟體設計的方法時，若選擇「再利用式開發」，對嵌入式軟體開發所注重的開發時程上是十分符合的。因嵌入式軟體十分在意搶占市場的時機(Time to Market)，若能採用「再利用」的開發方式便可快速的開發系統。目前在實際執行再利用式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，取來使用以減少軟體開發時重複撰寫元件所導致的人力及時間成本，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。

本研究提出使用「再利用元件庫」來縮短開發時程，此種方法可以新增、檢索已存入元件庫之元件之「以 XML 為基礎之嵌入式軟體再利用元件庫」，將可以將軟體設計的細節以量化方式描述後儲存以備使用，可減少開發人員重新撰寫元件所需之時間，日後在開發類似應用軟體時候，可以減少元件的重複撰寫，藉由儲存再利用元件、搜尋再利用元件及使用再利用元件來減少開發時所需的時間，能大幅提升效率。

關鍵詞：嵌入式系統、再利用元件、XML、UML

Abstract

When talking to develop embedded System, the requirement of market and tendency of system development are affected by “Time to market”. It is not easy to implement a system in a short time, and reduce the price and development time of product could increase the value of product.

Choosing the method of software design, “Reusable development” is conform to development time. Because embedded software cares about time to market, use reusable development way rapidly. There are still many defects to implement reusable development method. The most important question is how to help system developer to choose component.

So, we come up with reusable component repository to shorten develop time. This way can the happening of reduce rework to increase efficient. In our search, we study about the design of reusable component repository. And come out an approach with XML and native XML database to reduce the time of store data to increase effectiveness overall. Then we get the benefit of shorten develop time, increase reliability and lower risk.

Keyword: Embedded System, Reusable Component, XML, UML

目 錄

圖 目 錄

表 目 錄

第1章　導論

1.1 前言

近年來，在嵌入式多核心軟體的風潮之下，中介軟體及應用軟體系統的需求日益增加，複雜度與異質性也越來越高。在過去嵌入式系統的儀器控制特性，產品生命週期長，但在嵌入式系統大舉進入消費性電子領域後，產品更新速度加快，因此軟硬體設計也面臨前所未有的挑戰。

根據英國電機工程師協會[1]的定義，嵌入式系統是屬於一種電腦軟體與硬體的綜合體，並且特別強調「量身訂作」，也就是客製化(Customized)的系統。雖然客製化的需求讓嵌入式系統的終端千變萬化，但一些基本的嵌入式系統特性不會改變。面對不斷發展軟/硬體技術以及越來越多樣化應用，開發者必須關切的議題比以往多，因而挑戰也比過去大的多。根據 Tech Insights[2]的嵌入式系統設計報告指出，嵌入式系統開發人員在 2008 年面臨了許多新專案開發的壓力，因此 50%以上的開發專案沒有如期完成。資策會所執行之經濟部 ITIS 計畫的分析[3]也顯示全球行動電話嵌入式應用軟體市場規模成長，2009 年已達 30.2 億美金，且 2009-2010 年間嵌入式應用軟體市場規模將持續成長，預計 2010 年經突破 40 億美元大關，達 42.9 億美元市場規模。

根據以上資訊得知，新的嵌入式應用軟體市場正如火如荼的興盛中，面對此洪流，該如何因應成為一個新的課題。

1.2 研究動機及目的

軟體開發程序中的實做階段可將系統規格轉換成一個可執行的系統，他包含軟體的設計與程式撰寫的程序。軟體設計是對欲製作的軟體結構、系統的資料、系統元件之間的界面以及所使用的演算法等做詳細的描述。在選擇軟體設計的方法時，若選擇「再利用式開發」，對嵌入式軟體開發所注重的開發時程上是十分符合的。因嵌入式軟體十分在意搶占市場的時機(Time to Market)，若能採用「再利用」的開發方式便可快速的開發系統。

在實際執行再利用式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，取來使用以減少軟體開發時重複撰寫元件所導致的人力及時間成本，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。

本研究將提出一個可以新增、檢索已存入元件庫之元件之「以 XML 為基礎之嵌入式軟體再利用元件庫」，將可以將軟體設計的細節以量化方式描述後儲存以備使用，可減少開發人員重新撰寫元件所需之時間，日後在開發類似應用軟體時候，只需抽離及加入需要更換的元件，可達到元件之再利用性，提高開發效率。

1.3 章節安排

本文將所有研究內容共分為五個章節，期以循序漸進的方式完整闡述研究方法與成果。

一、 導論

介紹本論文研究的緣起、研究動機以及研究目的，讓讀者快速瞭解背景，並做為以下章節說明的基礎。

二、 背景知識與相關研究

此章說明與本論文相關的研究，以及概述相關技術的背景知識。

三、 研究理論方法

此章說明以 XML 為基礎之嵌入式軟體再利用元件庫。

四、 案例研究

本章說明利用在以 XML 為基礎之嵌入式軟體再利用元件庫來實際操作。

五、 結論及未來方向

對目前所研究內容做出結論，並且討論此研究內容可能不足之處以及未來可以延伸發展之目標。

第2章　背景知識與相關研究

2.1 軟體工程(Software Engineering)

2.1.1　軟體工程簡介

隨著資訊科技的快速發展，軟體與硬體產業日益發達，目前全世界幾乎所有國家都必須依賴著電腦系統處理各類事務，且有越來越多的商品結合電腦的軟硬體技術。隨著科技的進步，人們對電腦科技的需求越來越大，並且越來越繁複[4]。一套完整的資訊系統只有少部分的成本花費在硬體上，軟體佔系統的總成本的比例非常高，並且持續增加中。目前軟體系統的需求量越來越大，複雜度與異質性也越來越高，若在開發軟體時未使用標準化的開發方法，或是開發流程不夠完善則會導致軟體開發時間延遲、成本比預估時超出許多、執行效率不佳與系統難以維護等問題產生。

根據 Tech Insights 的嵌入式系統設計報告(Embedded System Design Reports)所進行的「2008 嵌入式市場研究報告」(Embedded designers on tighter schedules, juggling multiple projects in 2008)(如圖 2- 1)[2]指出：嵌入式系統開發人員在 2008 年面臨了許多新專案開發的壓力，只有 46%的專案是符合進度，有高達 54%的專案延期完成，且平均延後時間為 4.4 個月。

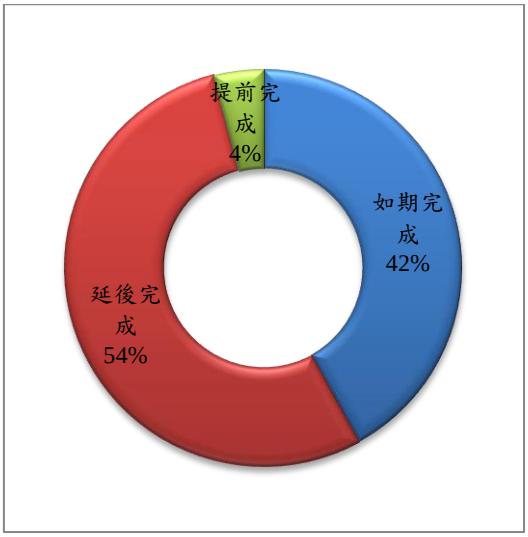

圖 2- 1 專案完成時間圖

且在同份報告也指出(如圖 2- 2)，高達 51%的開發人員最關心的是專案是否能按期完成，除錯問題暫 38%居次，解決程式碼複雜度和維持預算成本的關心比例相近，分別為 26%與 24%。

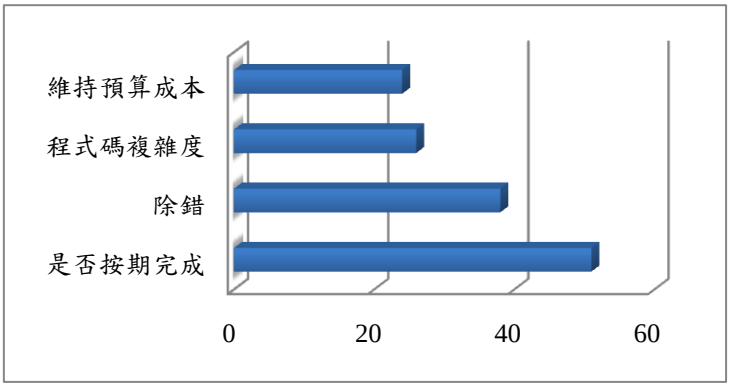

圖 2- 2 設計師在意的專案議題

為了解決上述軟體系統開發所可能會遭遇的困難，北大西洋公約組織(North Atlantic Treaty Organization, NATO)召集近 50 名的計算機科學家、程式設計師和工業界巨頭於 1968 年在德國舉辦了首次的軟體工程學術會議。此會議主要討論軟體危機的對策，並且提出以「軟體工程」來規範軟體開發時所需的專業知識，同時也建議軟體開發應該以類似工程的活動來進行[5][6]。

2.1.2 軟體工程的定義及核心價值

軟體工程主要包含軟體專案管理及軟體開發技術兩方面(如圖 2- 3)，而這兩方面的專業領域則來自於管理科學與資訊科學。隨著著眼點的不同，各個學者與專家對於軟體工程有著不同的見解，總述學者與專家的定義，軟體工程可被描述為一門研究如何系統化、正規化和量化等工程原則和方法論來達到經濟效益之軟體開發和維護的理論。

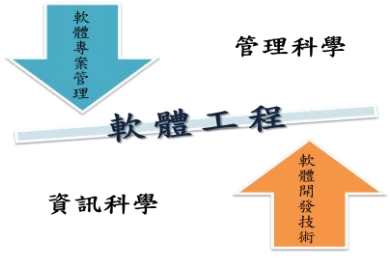

圖 2- 3 軟體工程之定義

在 1993 年美國 IEEE 的計算機協會和 ACM 組成了軟體工程協調委員會(Software Engineering Constituent Committee, SWECC)，並在 2004 年 6 月完成了軟體工程知識體(Software Engineering body of knowledge, SWEBOK)2004 版全文。SWEBOK 完整的描述出實踐軟體工程所需具備的核心知識，如圖 2-4。

圖 2-4 軟體工程的核心價值

2.1.3 軟體設計

在軟體工程核心價值中，軟體設計屬於實作的階段，對欲製作的軟體結構、系統的資料、系統元件之間的界面以及所使用的演算法等做詳細的描述，設計者不會馬上就可以產生完整的設計，而是必須不斷的開發許多不同版本的設計。設計程序可包含以不同抽象程度發展各種不同的系統模型，如圖 2-5。

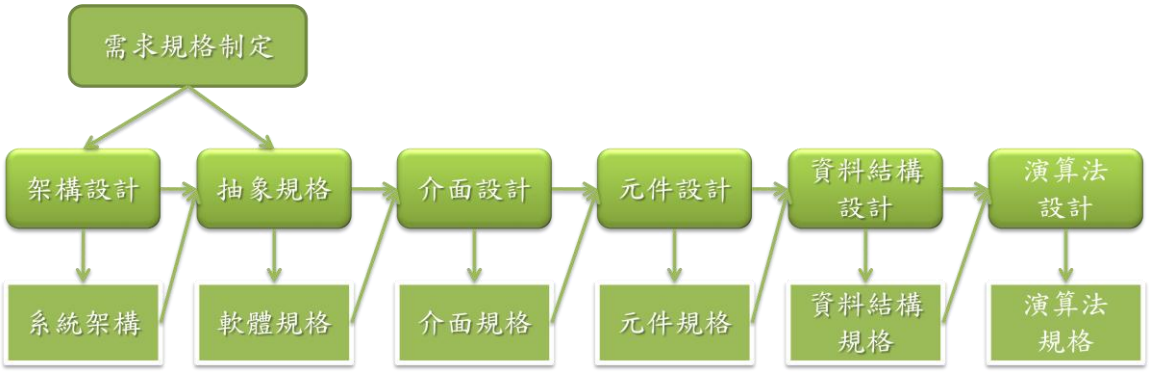

圖 2-5 設計程序的通用模型

每一個設計活動的輸出可以作為下一個階段的規格。這些規格可以是抽象的、用來澄清需求的正規劃規格或是指定如何實現系統某個部分功能的規格。上圖這些設計程序是一般模型常見的活動，時計的程序則必須是使用的不同開發方法而定。

在許多開發專案中，軟體設計仍採用特殊的程序，以自然語言的方式列出一些需求，再以一些非標準的方式來設計。這有可能會導致程式碼已經開始撰寫，但設計仍然還在修改的情況發生。為改善此情況，應使用「結構化」的軟體設計，利用一些軟體設計的表示方式及指引來設計軟體。結構化方法包括設計程序模型、用來表示設計的標計方式、報表格式、規則及設計指引等，可支援以下的系統模型：

- 資料流模型(Data-flow model)
- 實體關係模型(Entity-relation model)
- 結構化模型(Structural model)
- 物件導向方法(Object-oriented methods)

在本論文中，採用物件導向方法的軟體設計架構，物件導向設計是一種設計的方法，執行系統時不是傳統的以操作或功能面來思考，而是以「事物」的角度來思考。一個物件導向設計程序包括物件類別的設計及這些類別之間相互關係的設計。物件導向設計是物件導向開發程序的其中一個部分，開發軟體系統的物件導向模型以符合確認的需求，物件與待解決問題的解決方案具有相關性。

2.1.4 再利用的設計

軟體開發需要一個比較好的方式，軟體應視為一項資產，而重複利用這些資產可以減少開發的成本。若能廣泛且有系統的重複利用，將可以達到降低軟體製造與維護費用的需求。

為達到軟體再利用的需求，在軟體設計階段就應將此項需求列入考慮。有系統的重複使用元件需要在設計程序中考慮如何對現存設計再利用，並明確的將設計組織成可再利用的軟體元件。

重複利用的軟體單元可以大小分為以下三種(如圖 2- 6)：

1. 應用系統的再利用：

   整個應用程式系統皆可再利用，不做任何改變就可以納入其他系統，或是開發跨平台、針對特定客戶需求的應用程式系列產品時的再利用。

2. 元件再利用：

   重複的使用不含詳細實作細節的設計，只重複利用抽象的設計。本研究目的是對設計元件提出一個再利用的工具，將使用元件再利用作為實踐的方法，詳細內容將於下一章介紹。

3. 功能再利用：

   被製作成單一功能的軟體元件可以拿來再利用，如數學函式。此種再利用方法在過去四十年一直普遍使用的標準程式庫所用的方法。

軟體再利用的明顯好處是可以降低系統整體的開發成本，因為需要重新指定、設計、實作以及確認的軟體元件數量減少了。然而，節省成本只是軟體資產再利用的潛在利益之一，其他還有許多好處，如表 2- 1 所示。

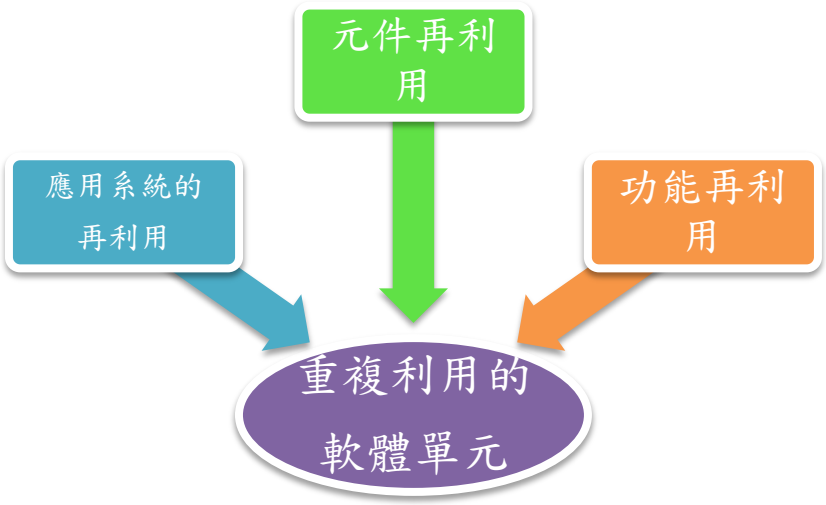

圖 2- 6 重複利用的軟體元件

表 2- 1 軟體再利用的好處

好處	說明
增加可靠性	重複利用在實際運作系統中使用過的元件應該比新元件更可靠，因為這些元件已經在各 種不同環境中受過測試。這些元件在剛開始被使用時，在設計和時做上的缺失都已經被找出來並且修正過，因此再利用這些元件時，他們的缺失就應該減少許多了。
降低處理風險	如果元件已經存在，該元件的再利用成本會比重新開發的成本確定。這對專案管理而言是一個重要的考慮因素，因為他可以降低專案成本預估時的不確定因素。如果重複使用非常大型的元件，例如子系統，這項優點會更加的明確。
有效的利用專家	與其讓某修應用領域的專家對不同專案做相同的工作，不如讓這些專家專注開發一些可再利用的元件，將他們的知識全部投入在這些元件上。
符合標準	有些標準可以時做一組標準的元件，例如使用者介面標準。舉例來說，使用者介面中的功能表單就可以製作成可再利用的元件，讓所有的應用程式都呈現相同功能表單。使用標準的使用者介面可以增進可靠性，因為使用者對熟悉的介面比較不會出錯。
加速開發	讓產品盡早問世通常比降低整體開發成本更重要，重複的使用元件將可加快系統開發的速度，因為開發與確認的時間都可以減少。

2.1.5 元件的種類

元件有大有小，可以視為一個整體，可以獨立存在。元件與元件間是使用介面(Interface)作為溝通的管道，以供其他的元件呼叫。至於元件的內部實作則被封裝起來，當元件內部改變時，只要介面不改變，則仍可與其他元件溝通。

元件可以分門別類，如使用者(User Interface Component)元件、企業(Business Component)元件或是功能性元件(Functional Component)等，端看用戶的需求，可

以開發不同種類的元件。

目前業界的標準規格常見如下幾種，主要是由 Microsoft, Sun Java Group 及 OMG 所提供的元件：

- Microsoft 的 COM+

  COM+, 微軟元件服務(Microsoft Component Services)是微軟在 Windows 2000 開始，針對 Microsoft Transaction Server 所強化更新的 COM 服務實作，作為 Windows 平台上的應用程式伺服器服務，是利用微軟平台開發分散式應用程式不可或缺的一個服務。

- Sun Java Group 的 EJB

  EJB, 企業級 JavaBean(Enterprise JavaBean)是一個用來構築企業級應用的伺服器端可被管理元件。EJB 是一個封裝有某個應用程式之業務邏輯伺服器端元件。EJB 最早於 1997 年由 IBM 提出，旋即被 Sun 採用並形成標準（EJB 1.0 和 EJB 1.1）。其後在 Java 社群行程（Java Community Process）支援下陸續定義新的 EJB 標準。EJB 規範的目的在於為企業及應用開發人員實作後台業務提供一個標準方式，從而解決一些此前總是在作業過程中總是重複發生的問題。EJB 以一個標準方式自動處理了諸如資料持久化，事務整合，安全對策等不同應用的共有問題，使得軟體開發人員可以專注於程式的特定需求而不再飽受那些非業務元素的困擾。

- OMG 的 CORBA

  CORBA(Common Object Request Broker Architecture)通用物件請求代理架構是軟體構建的一個標準，CORBA 定義了一系列 API，通訊協議，和物件/服務資訊模型用於使得異質應用程式能夠互相操作，這些應用程式用不同的程式語言編寫，執行在不同的平台上。CORBA 因此為定義明確的物件提供了平台和位置的透明性，這些物件是分散式計算平台的基礎。CORBA 使用一種介面定義語言用於刻畫物件將呈現出來的介面。CORBA 又規定了從 IDL 到特定程式語言，如 C++或 Java，實現的對映。這個對映精確的描述了 CORBA 資料類型是如何被用戶端和伺服器端實現的。標準對映的有 Ada、

C、C++、Smalltalk、Java、以及 Python。 還有一些非標準的對映，為 Perl 和 Tcl 的對映由這些語言寫的 ORB 實現。

2.1.6 使用者介面設計

使用者與系統的第一接觸就是在使用者介面[7]，良好的使用者介面是系統成功的一個重要關鍵。若介面很難使用，好的情況只是會讓使用者經常操作錯誤；壞的情況則可能是使用者拒絕使用這個軟體系統，完全不考慮可能會有十分優異的功能。如果介面呈現資訊的方式會讓人產生混淆或誤導，使用者可能會曲解資訊表達的意義，而做出一連串損毀資料的動作，甚至導致嚴重的系統故障。

圖形化的使用者介面(GUI)必須根據人類的能力來設計，本論文將會遵從表 2- 2 之議題設計圖形化的使用者介面。

表 2- 2 使用者介面設計原則

原則	說明
使用者的熟悉度	介面的設計應該以最常使用該系統的使用者為主，以他們經驗中所熟悉的名詞和觀念來設計界面。
一致性	介面應該盡可能的達成一致，相類似的操作應該用相同的方式啟動。
最少驚喜	使用者不太喜歡系統行為所帶來的驚喜。
可復原性	介面應該包含可讓使用者從錯誤中復原的機制。
使用者指引	在錯誤發生時介面應該提供有意義的回饋機制，並且提供即時線上的使用者說明工具。
使用者多樣性	介面應該為不同類型的系統使用者提供適當的互動功能。

2.2 開發技術的選用

2.2.1 XML 的優點

XML 是一個目前被廣泛使用的標準，XML Core Working Group 根據十項設計目標發展出 XML，在此提出較為重要的五項[11]：

1. XML 能直接在網路上使用：

   在網路環境瀏覽 SGML 文件必須搭配文件資料型別(Document Type Definition, DTD)與樣式表(Style Sheet)。但寫好的 XML 文件只要是具有良好結構，就可以直接在網路上閱讀。目前市面上主流的瀏覽器如 Mozilla Firefox、Internet Explorer、Chrome 與 Netscape 都能支援 XML 文件。

2. 使用者能容易開發處理 XML 文件的程式：

   因為 SGML 過於複雜不易開發程式,為了改善,所以 XML 的語法簡單、易於使用，方便程式人員開發應用程式。讓 XML 能廣泛地被接受，目前的 XML 標準已達到這項目標。

3. XML 文件應該是清晰且讓人易讀的：

   XML 的語法應嚴謹且能讓人讀懂，幫助程式開發人員快速撰寫程式，並使應用程式能精確的處理 XML 格式的資料。

4. XML 的設計應該有規律的格式且簡單：

   讓 XML 有規律的格式且簡單明瞭，可讓文件更容易被應用程式讀入並解譯。XML 的語法有嚴謹、簡單與規律的特性，可表是成 Extended Backus-Naur Form(EBNF)的格式，EBNF 是一個可容易被電腦程式解譯的表示法。

5. 使用者能很容易建立 XML 文件：

   在建立 SGML 文件時，必須定義對應的 DTD，用來驗證(validation)。建立 XML 文件不需定義 DTD，使用者只要注意文件是否有良好結構。建立 XML 文件不需要特定的文字編輯器，幾乎所有的文件編輯程式都可以建立 XML 文件。

XML Core Working Group 達成了上述五項目標，讓 XML 具有能直接在網路上使用、容易開發處理 XML 文件的程式、清晰且讓人易讀懂、有規律且簡單的格式與讓使用者容易建立 XML 文件的優點。XML 保留 SGML 的優點，如擴充性、具有結構、資訊再利用(information reusability)等，但沒有 SGML 的過於複雜與不易在網路環境使用等缺點。

XML 具有良好的擴充性，允許使用者定義標記，因此能容易地修改和增加資料，故 XML 可用來描述在各種領域中結構複雜的資料。此外，XML 具有自我描述(self-describing)的特性，當建立 XML 文件時，描述文建中資料的結構也會被建立，XML 標記依照 XML 建議標準定義的規則夾住被標記描述的資料，元素與屬性的宣告方式亦依照定義好的規則。這個特性讓程式不需經過困難的剖析就能檢索文件中特定的資料。上述優點讓 XML 廣泛地被使用在資料交換與各種不同的領域。

2.2.2 XML 介紹

XML（eXtensible Markup Language）是由 World Wide Web Consortium (W3C)所制定的標準[10]，是一種延伸式的標記語言，其表示方式如表 2- 3 所示。原本是 Standard Generalized Markup Language(SGML)的一個子集，但由於 SGML 過於複雜不易開發程式，故無法廣泛使用，因此 XML Core Working Group 開發了 XML 來解決這些問題。XML 具有擴展性(Extensibility)、結構性(Structure)、描述性(Description)、確認性(Validation)等特性。同時 XML 具有跨平臺的功能，對於不同的作業系統、硬體設備、應用軟體、多元的輸入模式，開發者可以自行制定符合己身需求的標記(tag)，做結構性的描述，促使相同的一份文件呈現不同的規格，適用於不同的軟體，符合不同的設備、滿足多重的輸入方式。

表 2- 3　XML 文件範例

XML Document Example
<?xml version="1.0"?> **<note>**

```
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

2.2.3 資料庫的選用

我們可以使用關聯式資料庫或原生型(Native)XML 資料庫來儲存 XML 檔案作為再利用元件庫之儲存文件，本節會說明它們的優點及缺點，並選出適合本研究適合的資料庫。在本節中說明使用關聯式資料庫儲存 XML 文件的方法及其優缺點、原生型 XML 資料庫並說明為何選擇原生型 XML 資料庫當作再利用元件庫的原因。

- **關聯式資料庫**

關聯式資料庫是以一組有結構與規則的表格來儲存資料，以關聯式資料庫儲存 XML 文件的方法一般可分為以下兩種：

1. 以 Large Object(LOB)的資料型態儲存：

   LOB 是一種資料型態，專門用來儲存內容龐大的資料，像是多媒體檔案等。此方法是把整份 XML 文件存入表格中一個資料型態為 LOB 的欄位中。

2. 拆解 XML 文件的內容以儲存於表格中：

   此方法是把 XML 文件依照 DTD 或 XML Schema 拆解成多筆資料後，接著把資料分別存到多個表格中。要取出或是使用 XML 文件時，利用 SQL 取出這些表格的內容並組合成 XML 文件。

其優缺點比較如表 2- 4 所示：

表 2-4 關聯式資料庫優缺表

	優點	缺點
以 Large Object(LOB)的資料型態儲存	不需要設計 XML Schema 即可完整保留 XML 文件的結構，並可快速的取出整份 XML 文件	無法有效率的存取 XML 文件特定部分的資料，因為 XML 文件是以二進位或文字的形態存在 LOB 欄位中，要存取其中某元素時必須先剖析整份文件才能找到。
拆解 XML 文件的內容以儲存於表格中	速的處理結構簡單且固定的 XML 文件，並可使用 SQL 有效率的存取指定的元素，對 XML 文件作部分的更新	i. 可能會遺失 XML 文件的結構，因為使用 SQL 取出資料時，資料出現的順序是 SQL 語法搜尋到的順序，可能會與原本的 XML 文件不同，因而遺失原來的結構。 ii. 當 XML 文件結構很複雜的時候，必須使用大量的表格儲存 XML 元素與對應的關係，而把這些表格組合成 XML 文件也會花費大量的時間。

●原生型 XML 資料庫

由於使用關聯式資料庫儲存 XML 文件會衍生出許多不必要的問題，近年來開發出許多專門儲存 XML 文件的原生型 XML 資料庫[15]，能直接存取 XML 文件，不需搭配 LOB 欄位或拆解文件。此外也有許多配合原生型 XML 資料庫使用的 API，以提供查詢、檢索、更新資料庫等功能。原生型 XML 資料庫具有以下的特性：

1. 原生型 XML 資料庫以 XML 文件為基本儲存單位。
2. 原生型 XML 資料庫具有 Round-trip 的特性，Round-trip 是把 XML 文件存入資料庫後再從資料庫中取出完全相同的 XML 文件。
3. 原生型 XML 資料庫大部分都支援 XQuery、XPath 等查詢檢索 XML 文

件的語言，能快速的更新或存取文件的部分內容。

使用原生型 XML 資料庫儲存 XML 文件不會有使用關聯式資料庫遇到的眾多缺點，因為它以 XML 文件為儲存單位，不需定義轉換的規則即可以直接存取整份文件，並可儲存結構不同與結構複雜的 XML 文件。而使用原生型 XML 資料庫主要的缺點是以 XML 文件為基本儲存單位，若我們需要結合其他關聯式資料庫系統的資料則需要視情況將 XML 文件轉換成關聯式資料庫的表格，或是將表格轉換成 XML 文件，在資料整合較不便。

●選擇的原因

XML 文件從文件結構的角度可以分為以資料為中心(data-centric)與以文件為中心(document-centric)兩種。以資料為中心的 XML 文件結構具有規則，其資料出現的順序不重要，適合以關聯式資料庫系統來處理；以文件為中心的 XML 文件資料結構不太規則或是不規則，資料出現的順序是重要的，適合以原生型 XML 資料庫來處理。

在製作再利用元件時，元件出現的順序通常是有特定的用意，若順序錯亂會造成使用的誤解。我們可選用原生型 XML 資料庫或是關聯性資料庫搭配 LOB 的方式，但後者在存取 XML 文件特定部分的資料時效率較差。基於以上原因，本論文選擇原生型 XML 資料庫作為再利用元件庫之儲藏庫。

2.2.4 XQuery

XQuery 是專門用來查詢 XML 文件的語言，它是從 XPath 所延伸而來。XPath 是一種對 XML 文件內容定址(addressing)的運算式(expression)語言，其用處是可有效率的找出 XML 文件的特殊元素或屬性。XQuery 使用 XPath 語法描述 XML 文件中特定資料的位址或查詢條件用以取得資料，取出後再進行比較或排序等運算，得到查詢結果後輸出給使用者，就某個角度而言 XQuery 與 XML 文件的關係就如同 SQL 與關聯式資料庫的關係。

W3C 為 XQuery 定義 11 種運算式供使用者撰寫需要的 XQuery 語法來查詢資料，以下介紹較常用的幾種：

- 基本運算式(Primary expression)

  定義資料型態與變數的表示法，變數的表示法式$加上變數名稱，如$ex1就是一個變數。

- 路徑運算式(Path expression)

  即前述的 XPath，用來描述 XML 文件中特定資料的位置或條件。

- 比較運算式(Comparison expression)

  用來作值的比較，如=、!=、<、<=、>與>=等。

- FLWOR 運算式(FLWOR expression)

  FLWOR 是由 For、Let、Where、Order by 以及 Return 五種子句取締一個英文字母組合而成的，它們的用途分別如述：

For 及 Let 子句用來宣告變數並連結(bind)路徑運算式描述的資料，每個 XQuery 至少須包含一個 For 或 Let 子句。For 與 Let 的差異在於 For 類似程式語言的迴圈，每遇到一個符合的資料就輸出一次；Let 則是將所有符合的資料集中，輸出一次。Where 子句用來定義篩選的條件，測試 For 及 Let 子句找到的資料。Order by 子句定義對變數連結到的資料排序方法。Return 子句定義隊滿足條件之資料的運算與傳回結果的格式。

在此使用一個範例來說明 XQuery 的使用方式，表 2- 5 是一份儲存三項物品的編號、價格及進貨日期資訊的 XML 文件。XML 文件的名稱為 itemlist.xml，可使用的 XQuery 來查詢，找初價格大於 300 元的物品，並將找到的物品依照物品名稱排序，再列出名稱及價格。

表 2- 5 一個 XML 文件範例-itemlist.xml

```
<?xml version=”1.0”
encoding=”Big5”?>

<itemlist>

      <item number=”001”>
```

```
        <title> teddy
              bear</title>
        <price>106</price>
    </item>
    <item number="002">
        <title> barble </title>
        <price>323</price>
    </item>
    <item number="0013">
        <title> toy motor car
            </title>
        <price>471</price>
    </item>
<itemlist>
```

表 2- 6 查詢 itemlist.xml 文件的 XQuery

```
for $item in doc("itemlist.xml")/itemlist/item
where $item/price>300
order by $item/title
return
<items>
{$item//title}
{$item//price}
</items>
```

表 2- 6 第一行的 for 子句宣告一個名稱為$item 的變數，它會指向 itemlist.xml 文件中，itemlist 元素下名稱為 item 的子元素。第二行的 where 子句檢查這些 item 子元素的價格，選出滿足大於 300 元的子元素。接著第三行的 order by 子句對這

些選出的子元素依照名稱排序。最後在第四行的 return 子句依照表 2- 6 定義的結構輸出查詢結果如表 2- 7 所示。

表 2- 7 以表 2- 6 的 XQuery 查詢表 2- 5 文件的結果

```
<items>
   <title> barble </title>
   <price>323</price>
</items>
<items>
   <title> toy motor car
        </title>
   <price>471</price>
</items>
```

2.2.5 Sedna 介紹

Senda[16]是一套開放原碼的原生型 XML 資料庫，具有自動索引(automatic indexing)、全文搜尋(full-text search)及支援 XQuery 查詢等功能。

Sedna 儲存 XML 文件的機制有以下幾項特點：

1. 以 XML 文件為最小的儲存單位。
2. 使用者不需預先匯入 XML 規範就可以存入 Sedna。
3. XML 文件在存進資料庫前，Sedna 會自動剖析文件確認語法正確與否，若有錯誤則無法進行儲存。
4. Sedna 管理 XML 文件的方式與檔案總管相似，文件被存在類似資料夾的 Collection 內，其中可儲存多份文件，亦可建立多個階層。

Sedna 支援 Java、C 及 C++，在本論文中以 Java 實現，可使用一些已經開發好的 Java API[17]，如 XML:DB 及 XQJ。程式設計人員可使用 XML:DB 及 XQJ

提供的方法從 Sedna 取得特定的 XML 文件及其內容，並對資料進行查詢、新增、修改或刪除等動作。

2.2.6 VERTAF/Multi-Core (VMC)

VERTAF/Multi-Core (VMC) 為一針對多核心嵌入式架構之整合性軟體開發環境，開發者可藉由本環境以 SysML 描述其系統需求，以 UML 標準符號塑模其系統設計，並將該設計套用樣式結構自動轉換為更高品質的設計，透過優良的設計模型自動產生程式碼，最後並可對程式碼加以測試。因此，本系統具下列優點：(1) 支援多核架構嵌入式軟體之設計與驗證；(2) 提供一個整合性的開發環境；(3) 因本環境乃基於 UML 標準所建構，所以易與目前既有之 UML 開發工具整合；(4) 支援既有之多核架構開發函式庫，如：OpenMP 以及 Intel Threading Building Blocks 。

本整合計畫一共區分為七個子計畫，含括系統開發生命週期之需求、設計、實作、測試與維護階段，各子計畫間的關係如圖 2-7 所示，總計畫包含七項子計畫：子計畫一以 SysML 為基礎之多核心嵌入式系統需求塑模環境，子計畫二支援多核心嵌入式軟體設計之多重觀點整合模型與可再用元件庫，子計畫三多核心嵌入式軟體設計之設計支援系統，子計畫四多核心嵌入式軟體之合成與程式碼生成，子計畫五多核心嵌入式軟體設計工具系統之架構支援，子計畫六多核心嵌入式軟體設計工具系統之平行程式優化支援，以及子計畫七多核心嵌入式軟體設計之測試支援系統。

本論文是由朱正忠教授所領導的 Integration and Maintenance 階段「VMC_MUM」的 RCR Editor & Management 及 Reusable Design Component Repository 兩個部份，其研究方法及研究案例會在第三、四章中詳述之。

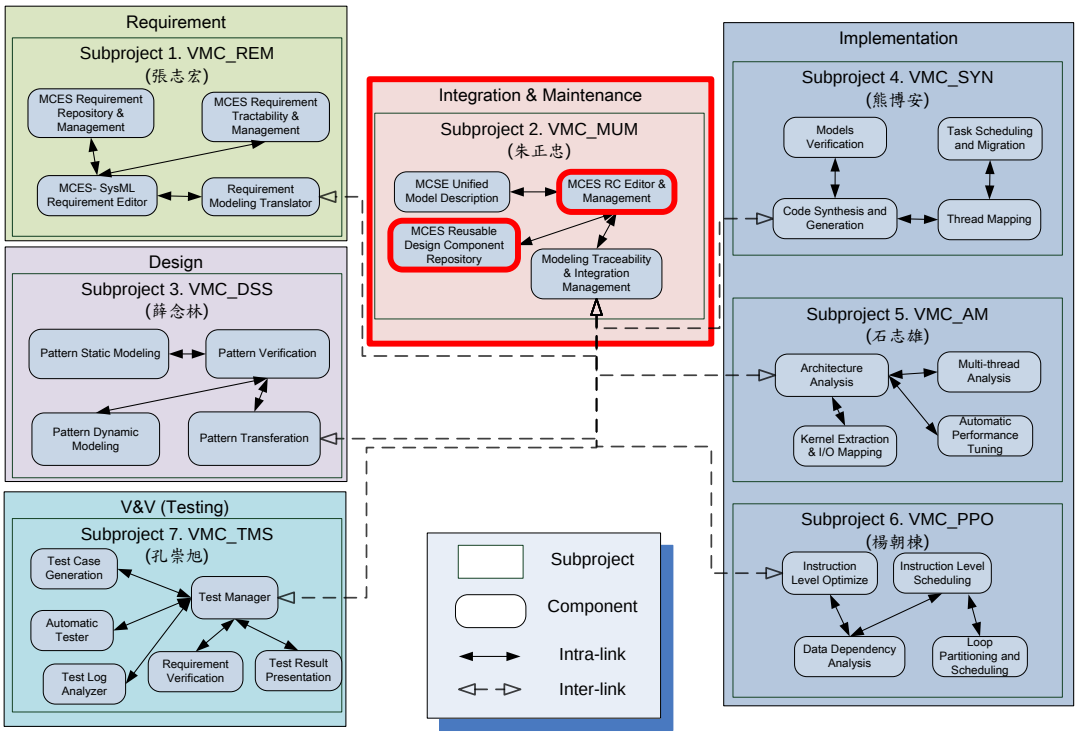

圖 2- 7 VMC 各子計畫之關係圖

第3章　研究方法

在 2.2.6 VERTAF/Multi-Core (VMC)中提到，本論文為 VMC 的 MUM 子計畫的一部分，VMC_MUM 的系統示意圖如圖 3- 1 所示，其中紅色的部分是本篇論文所實作的部分，RCR Editor Management 及 Reusable Component Repository，在接續的章節中會依照本計畫所實作部分之系統架構、系統流程、使用者案例、及再利用元件的定義、再利用元件的查詢及作廢等，一一的詳盡介紹之。

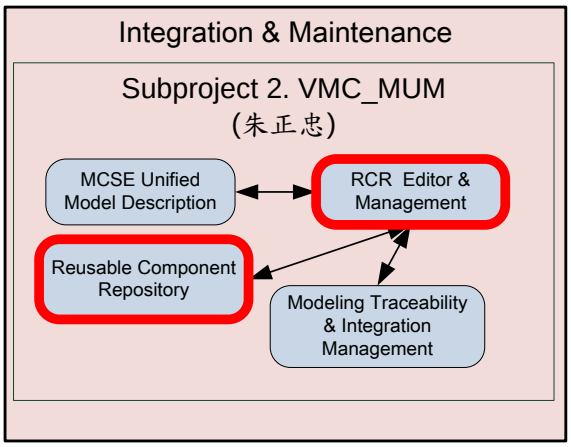

圖 3- 1 VMC_MUM 系統示意圖

3.1　系統架構

在本論文中提出再利用元件庫來幫助我們快速選擇已存在的元件減少元件重製的時間，圖 3- 2 為本論文研究之架構圖，分為以下若干步驟：

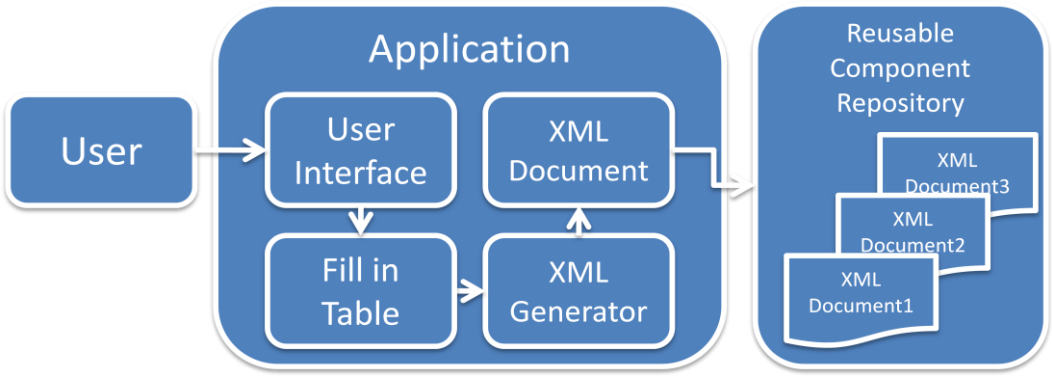

圖 3- 2　系統架構圖

第一，使用者透過使用者介面填寫表格，在此步驟我們提供使用者熟悉、友善並含有指引的使用者介面，讓使用者得以快速上手。第二，使用者將再利用元件的資料填入表格中，藉由填入的過程，可得知該元件的類型、結構圖、UML 圖、程式語言、作業系統及硬體的限制等，此動作有助於再利用元件的分類、擷取及理解。在此步驟也包含版本控管的功能，使用者可在已有的元件上加上新的描述及功能。第三步驟則透過 XML 轉換器作文件的轉換，將使用者所輸入的再利用元件內容轉換成 XML 文件的格式以便儲存。第四步驟是透過 XML 產生器得到 XML 文件，透過 XML 文件的擴展、結構、描述及確認等特性制定符合系統的 Tag 做結構性的描述。最後一個步驟是將產生出的 XML 文件儲存至再利用元件庫，在本系統所使用的元件庫為原生型 XML 資料庫，透過原生型 XML 資料庫的方便直接存取及配合許多 API，以提供查詢、檢索、更新資料庫等功能，在整合時比較方便。

3.2 系統流程

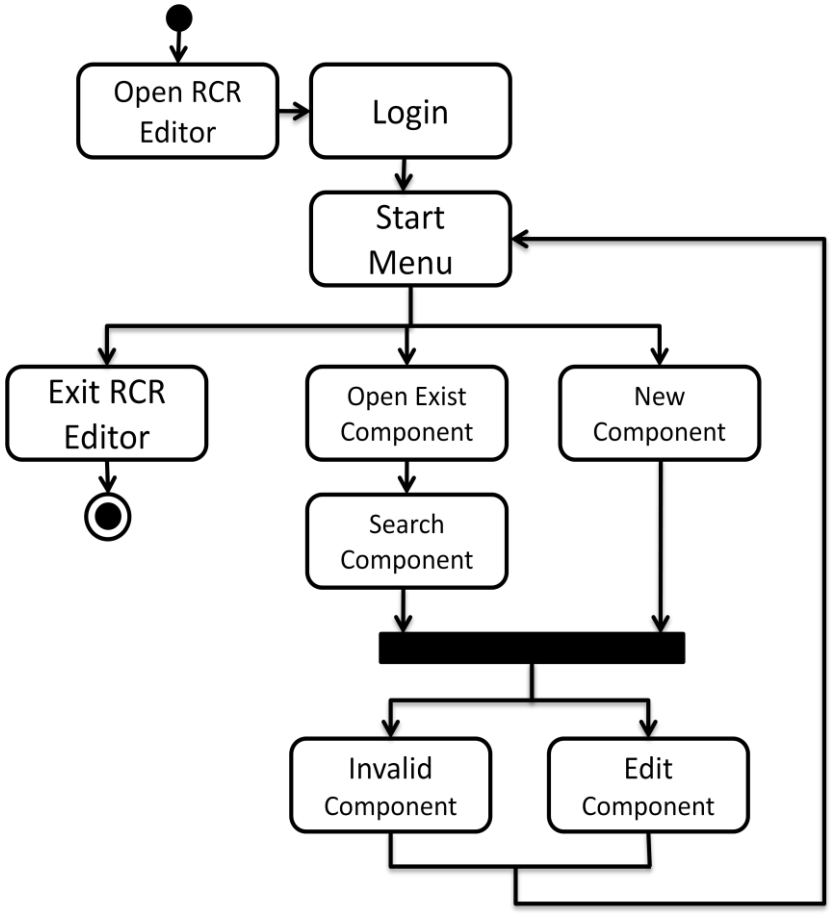

圖 3- 3 系統流程圖

本系統的流程如圖 3- 3 所示，其系統流程如下：

Step 1: Open RCR Editor

開啟本系統(RCR Editor)。

Step 2: Login

輸入使用者資料，如姓名、e-mail 及密碼，用以管理使用者資訊。

Step 3: Start Menu

登入之後的選單，可選 **Step 4: Open Exist Component, Step 5: New Component** 或是 **Step 9: Exit RCR Editor。**

Step 4: Open Exist Component

開啟已存在的再利用元件，並在 **Step 5: Search Component** 可以查詢想看的元件。

Step 5: New Component

新增再利用元件至元件庫中，在 **Step 7: Edit Component** 中可編輯。

Step 6: Search Component

將現有的再利用元件展示在一個表格中供使用者做 **Step 7: Invalid Component** 或是 **Step 8: Edit Component** 的後續動作。

Step 7: Invalid Component

使用者覺得該元件再也沒有利用價值，則可透過此步驟作廢元件。

Step 8: Edit Component

對新的元件或是修改的元件做編輯的動作。

Step 9: Exit RCR Editor

若使用者結束 RCR Editor 的使用，可透過此步驟結束動作。

3.3 使用者案例

本系統提供之 Use Case Diagram 圖如圖 3- 4 所示：

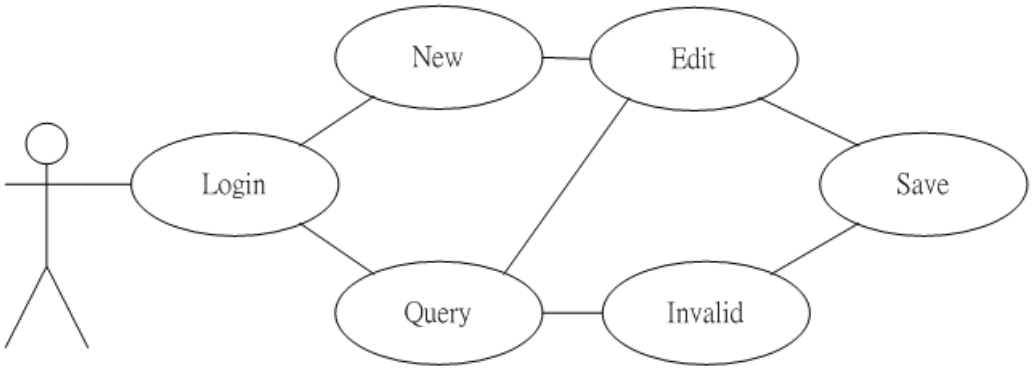

圖 3- 4 Use Case Diagram

Use Case Scenario 1:

Name: Login

Actors: User

Goals: User feed in personal information

Step:

Actor actions	System responses
1.**TUCBW** User fill out personal information	2. System receive data 3. Save user data in database 4. **TUCEW** Notify user save complete

Use Case Scenario 2:

Name: New

Actors: User

Goals: User add new reuse component

Step:

Actor actions	System responses
1. **TUCBW** User fill out component information	2. System receive data 3. Save user data in database 4. **TUCEW** Notify user adding complete

Use Case Scenario 3:

Name: Query

Actors: User

Goals: User open exist reuse component

Step:

Actor actions	System responses
1. **TUCBW** User choose component	2. System receive data 3. **TUCEW** Show that component information

Use Case Scenario 4:

Name: Edit

Actors: User

Goals: User edit component

Step:

Actor actions	System responses
1. **TUCBW** User fill out component information	2. System receive data 3. Save in database 4. **TUCEW** Notify user adding complete

Use Case Scenario 5:

Name: Invalid

Actors: User

Goals: User modify exist component

Step:

Actor actions	System responses
1. **TUCBW** User choose the invalid component	2. System receive data 3. Save change in database 4. **TUCEW** Notify user adding complete

Use Case Scenario 6:

Name: Save

Actors: User

Goals: User save component

Step:

Actor actions	System responses
1. **TUCBW** User save component	2. System receive data 3 Save in database 4. **TUCEW** Notify user saving complete

3.4 再利用元件的定義

3.4.1再利用元件之 XML 結構

再利用元件之 XML 文件格式如圖 3- 5 所示，其 XML Element 及 Attribute 對應之意涵在 3.4.2 及 3.4.3 會一一解釋之。

```
<?xml version="1.0" encoding="Big-5"?>
<component type="Android" id="Android-001">
  <management author="Tingyi Chao" email="babyiar@gmail.com" version="1.2" date="2010/5/20" status="normal" referenceCount="0">
    <name>PrintPhoto</name>
    <intent>This component is for draw cycle</intent>
    <structure>C:\User\Public\Files\printphoto_structure.jpg </structure>
    <uml>C:\User\Public\Files\printphoto_use_case.jpg</uml>
    <code>C:\User\Public\Files\printphoto.java</code>
    <programming language="java" os="MS Windows7"></programming>
    <hardware_constraint power="110V" cpu="QualcommR MSM7227, 600 MHz" memory="512MB" screenSize="3.2inch" other=""></
hardware_constraint>
    <note>Draw cycle by yourself!</note>
    <invalid_note> </invalid_note>
  </management>
</component>
```

圖 3- 5 XML 文件格式

3.4.2再利用元件表格

本論文中的再利用元件主要考量到類型、管理控制、系統架構圖、UML 圖、程式碼、程式語言、使用的系統及硬體的限制等，其欄位、內容及備註如表 3- 1 中所示：

表 3- 1 元件內容表

Field		Content	Remarks
Type		The type of embedded device	Windows CE, Android, ARM, Other
ID		ID from Type and the turn in database	Ex: Android-001
Name		The name defined by user	Ex: PrintPhoto
Management	**Author**	Creator 's name	Ex: Tingyi Chao
	e-mail	Creator 's e-mail	Ex: babyiar@gmail.com
	Version	Version of this component	Ex: 1.2
	Date	Entry Date	Ex: 2010/5/20

Field		Content	Remarks
	Status	The status of this component	Normal or Invalided
	Reference Count	The number of refer time	Default: 0
Intent		Intent of this component	Ex: This component is for draw cycle
Structure		System architecture diagram of this component	Save system architecture diagram as refer for future user
UML		UML diagram of this component, there are 13 options. Can press "Add more UML" to add more UML diagram	Save UML diagram as refer for future user
Code		Code of this component	Save Code as refer for future user
Programming Language		Programming Language of this component	Java, C++, C or Other
Programming OS		Programming OS of this component	MS Windows7, MS XP, Linux or Other
Hardware Constraint	**Power**	The power deployment of this component on embedded device	110V, 220V, 230V or other
	Core	The core of this component on embedded device	1, 2, 4, 8 or other
	CPU	The CPU of this component on embedded device	Creator can feed in CPU of the embedded device

Field		Content	Remarks
	Memory	The memory of this component on embedded device	512MB, 1G, 2G, 4G or other
	Screen size	The screen size of this component on embedded device	Creator can feed in screen size of the embedded device
	Other	Other constraint feed in by creator	Creator can feed in other constraint of the embedded device
Note		Creator can give a note for future user to describe more contents	By nature language
Invalid Note		When the component be invalided, user should fill out the reason	By nature language

3.4.3對應之 XML Tag

3.4.2 中對本論文中的再利用元件做了定義，在本節中將這些欄位對 XML 產生對應的 Element 及 Tag，如表 3- 2 所示：

表 3- 2 元件對應 XML 表

XML Element	XML Tag
Type 及 ID	<component type="" id="">
Name	<name>
Management	<management>
Author	< author>
e-mail	< e-mail>
Version	< version>
Date	< date>
Status	< status>
Reference Count	<reference_count>
Intent	<intent>

Structure	<structure>
UML	<uml>
Code	<code>
Programming Language	<programming_language>
Programming OS	<programming_os>
Hardware Constraint	<hardware_constraint>
Power	<power>
Core	<core>
CPU	< cpu>
Memory	< memory>
Screen size	< screen_size>
Other	< other>
Note	<note>
Invalid Note	<invalid_note>

3.5 再利用元件查詢

在再利用元件庫中，查詢已存在資料庫中的元件是很重要的一個步驟，如何提供使用者一個有效率、及時且搜尋結果有幫助的查詢方法是很重要的。若資料庫中有一萬筆資料，使用者查詢後得到一千筆資料也是無用的。系統必須提供使用者有鑑別度的查詢關鍵字，在 3.5.1 及 3.5.2 中我們將探討該如何實作再利用元件庫查詢的部分。

3.5.1 對再利用元件庫做查詢

誠如以上章節所描述，本系統對再利用元件的分類如表 3- 3 所示，在此節我們將適合做為查詢條件的欄位列出，並搭配 XQuery 語法。

表 3- 3 XQuery 欄位

Field	Content	XQuery Syntax
Type	Query by Type	for $doc in collection("/RCR_Editor/") return $doc//component/@type

Field	Content	XQuery Syntax
Name	Query by Component Name	for $doc in collection("/RCR_Editor/") return $doc//component/@name
Author	Query by Author Name	for $doc in collection("/RCR_Editor/") return $doc//management//author
Date	Query by Entry Date	for $doc in collection("/RCR_Editor/") return $doc//management//date
Reference Count	Query by Reference Count	for $doc in collection("/RCR_Editor/") return $doc//management//reference_count
Programming Language	Query by Programming Language	for $doc in collection("/RCR_Editor/") return $doc//programming_language
Programming OS	Query by Programming OS	for $doc in collection("/RCR_Editor/") return $doc//programming_language
Hardware Constraint Power	Query by Power	for $doc in collection("/RCR_Editor/") return $doc// hardware_constraint//power

Field	Content	XQuery Syntax
Core	Query by the number of Core	for $doc in collection("/RCR_Editor/") return $doc// hardware_constraint//core
CPU	Query by the number of CPU	for $doc in collection("/RCR_Editor/") return $doc// hardware_constraint//cpu
Memory	Query by the size of Memory	for $doc in collection("/RCR_Editor/") return $doc// hardware_constraint//memory
Screen size	Query by Screen size	for $doc in collection("/RCR_Editor/") return $doc// hardware_constraint//screen_size
Other	Query by Other key of Hardware Constraints	for $doc in collection("/RCR_Editor/") return $doc// hardware_constraint//other

3.5.2 按照不同的排列法查詢

在系統回應使用者的查詢後，使用者可以根據日期、作者名字第一字或是被重用的次數做不同的排序方法。其 XQuery 語法如表 3- 4 所示。

表 3- 4 Different Sort

Sort by	XQuery Syntax
Entry Date	Order by $date
Author Name	Order by $author
Reference Count	Order by $reference_count

3.6 再利用元件的作廢

在本系統中的再利用元件若是不復使用，則可以做作廢的動作，在本節介紹作廢的時機及方法。當元件已經不適合再做再利用的動作時，使用者可以標示該元件，並填入為何認為該元件不適合再使用，系統會將元件標示為「作廢」，日後其他使用者在查詢時會將被標示為作廢的元件置於查詢結果的最後面，使用者點選該元件則會顯示元件作廢的原因及作廢日期，如表 3- 5 所示。

表 3- 5 An invalided component example

```
<management author="Tingyi Chao"
      email="babyiar@gmail.com" version="1.2"
      date="2010/5/20" status="invalided"
      referenceCount="0">
</management>
<invalid_note>This component is no longer to use.</invalid_note>
```

3.7 XML 文件範例

表 3- 6 為一個填妥的再利用元件的 XML 範例：

表 3- 6 An XML document example

```
<?xml version="1.0" encoding="Big-5"?>
<component type="Android" id="Android-001">
    <management author="Tingyi Chao" email=babyiar@gmail.com
     version="1.2" date="2010/5/20" status="normal"
     referenceCount="0">
        <name>PrintPhoto</name>
        <intent>This component is for draw cycle</intent>
        <structure>
        C:\User\Public\Files\printphoto_structure.jpg
        </structure>
        <uml>C:\User\Public\Files\printphoto_use_case.jpg</uml>
        <code>C:\User\Public\Files\printphoto.java</code>
        <programming language="java" os="MS Windows7">
        </programming>
        <hardware_constraint power="110V"
        cpu="QualcommR MSM7227, 600 MHz"
        memory="512MB" screenSize="3.2inch" other="">
        </hardware_constraint>
        <note>Draw cycle by yourself!</note>
        <invalid_note>    </invalid_note>
    </management>
</component>
```

第4章　案例研究

4.1 案例介紹

在本案例研究中，我們針對 DVR(Digital Video Recording)系統中的 Remote Monitor Client (RMC)進行再利用元件的再利用的流程作為本篇研究的研究案例，來說明以 XML 為基礎的再利用元件庫在新增、儲存、搜尋、刪除再利用元件等功能的流程，DVR(Digital video recording)[19][20]是一個即時錄影的多媒體系統，可以提供遠端使用者即時觀看或者是收看已錄影的存檔影片，是由多部攝影機與資料庫結合建構而成的系統，DVR 系統主要分為三大部份，包括：Remote Monitor Client (RMC)、Video Streaming Server(VSS)及 Parallel Video Encoding (PVE)，其系統架構圖如圖 4- 1 所示。

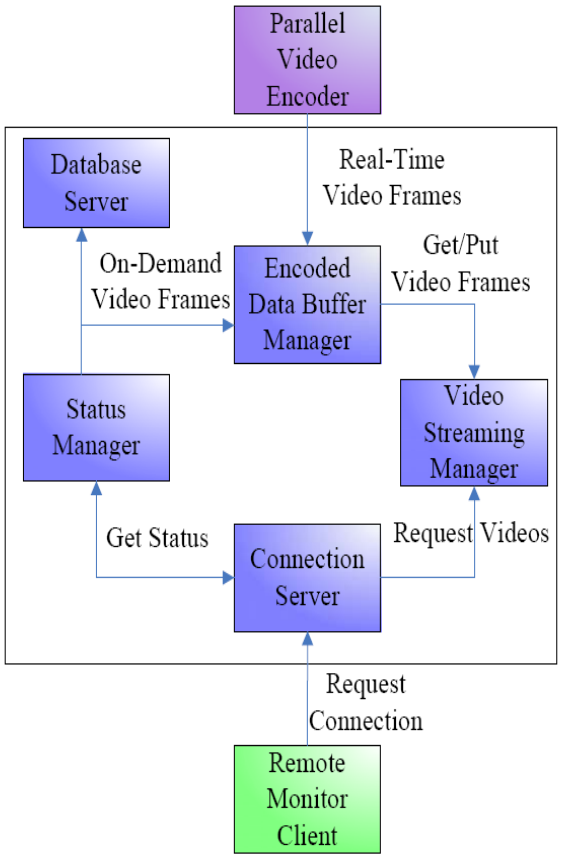

圖 4- 1 DVR 系統架構圖

4.2 案例實作

在此節將展現本系統在操作方面的功能，分為登入及啟始畫面及新增、儲存、搜尋再利用元件等功能的實現。

4.2.1 登入及啟始畫面

在使用 RCR Editor 的一開始必須先登入系統，如圖 4- 2 所示：填入 User Name, e-mail 與 password，在系統中識別使用者的身分，可以作人員管理的動作，在新增、修改再利用元件時皆會註明作者及聯絡方式。

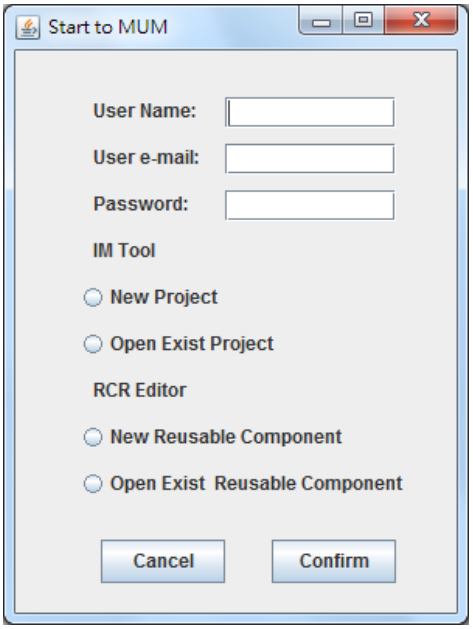

圖 4- 2 Start VMC_MUM

在輸入使用者資料後，可選擇使用 IM Tool 或是 RCR Editor 的功能，IM Tool 為 VMC_MUM 計畫中的另外一部分，在此不贅述其內容。在 RCR Editor 有兩個選項，可以選擇 New Reusable Component 或是 Open Exist Reusable Component，選擇後按下 Confirm 便可開始使用 RCR Editor，如圖 4- 3 所示。

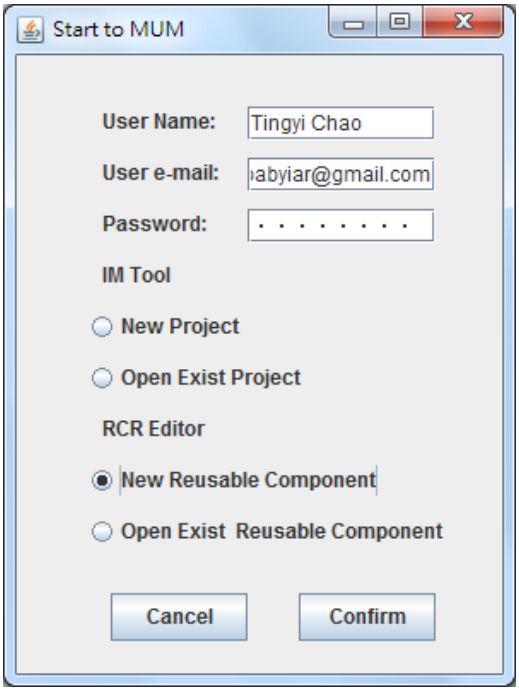

圖 4- 3 Start to RCR Editor

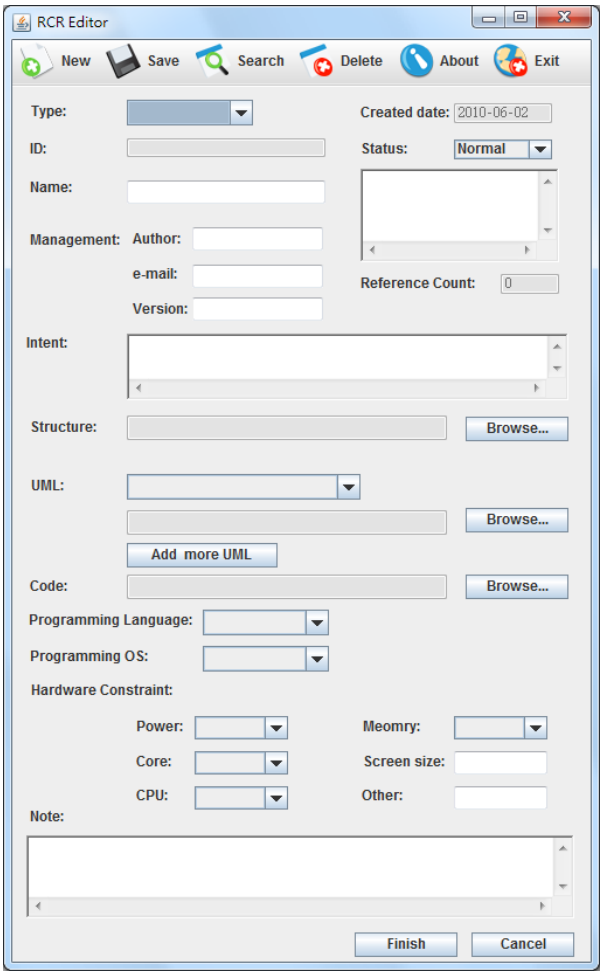

圖 4- 4 新增一個再利用元件

4.2.2 新增再利用元件

在輸入完使用者資料並選擇使用 RCR Editor，可分為 New Reusable Component 或是 Open Exist Reusable Component，若選擇 New Reusable Component 可得到圖 4-4 的畫面，在本系統中新增一個再利用元件時，可藉由此畫面的動作新增一個再利用元件，其中各欄位的限制及範例在 3.4.2 有深入的探討過。將表格陸續填滿，及將該再利用元件會用到之架構圖、UML 圖及程式碼之路徑也填入，如圖 4-5 及圖 4-6 所示填妥，即可新增一個可再利用的元件。

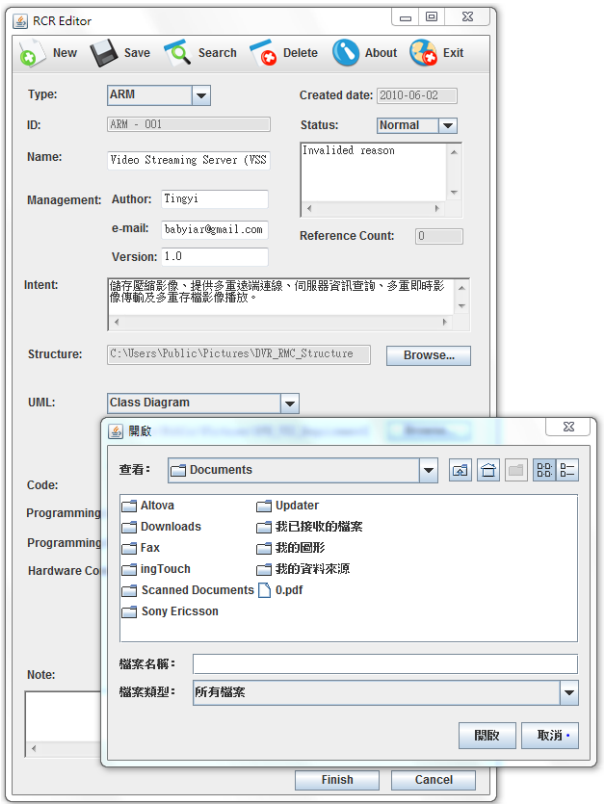

圖 4-5 選取再利用元件之 UML 圖型

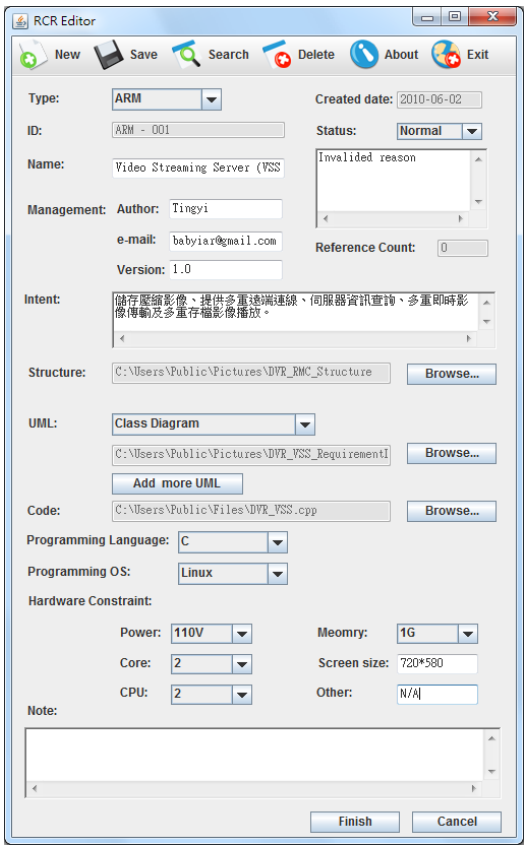

圖 4- 6 填寫再利用元件資訊

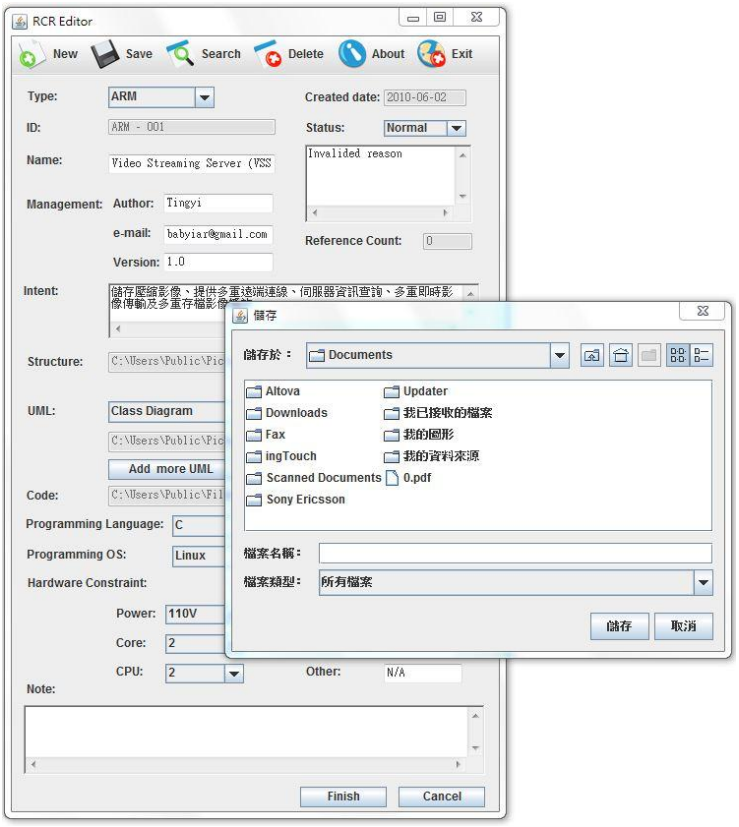

圖 4- 7 儲存 RMC 再利用元件

4.2.3 儲存再利用元件

透過按下 Save 鈕後，可得到圖 4- 7 的畫面，即可將 RMC 這個元件儲存至再利用元件庫。至於如何觀看已經存在資料庫中的再利用元件及對這些元件作刪除的動作，在 4.2.4 與 4.2.5 將詳述之。

4.2.4 搜尋可用之再利用元件

使用者可透過 Search 按鈕開啟目前現有的再利用元件，如圖 4- 8 所示，這些可選的條件是由 3.5.1 所制定的規範而來。在填妥圖 4- 8 的表格後，按下查詢的鈕後，系統就會列出此類型中可再利用的元件，如圖 4- 9 所示。若想查看完整的元件內容，則可點選項目，系統會跳出完整的元件內容視窗，方便使用者觀看。圖 4- 10 為一個完整的再利用元件表格，並已點選架構圖放大該圖片。若使用者要修改此元件，則可按下 Edit 鍵即可進入編輯的畫面，如圖 4- 4。

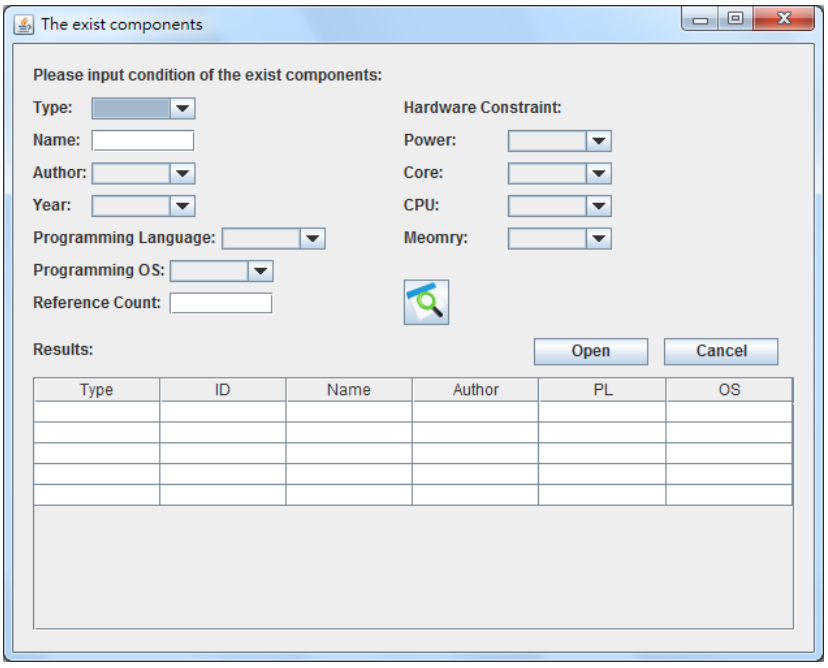

圖 4- 8 查詢介面

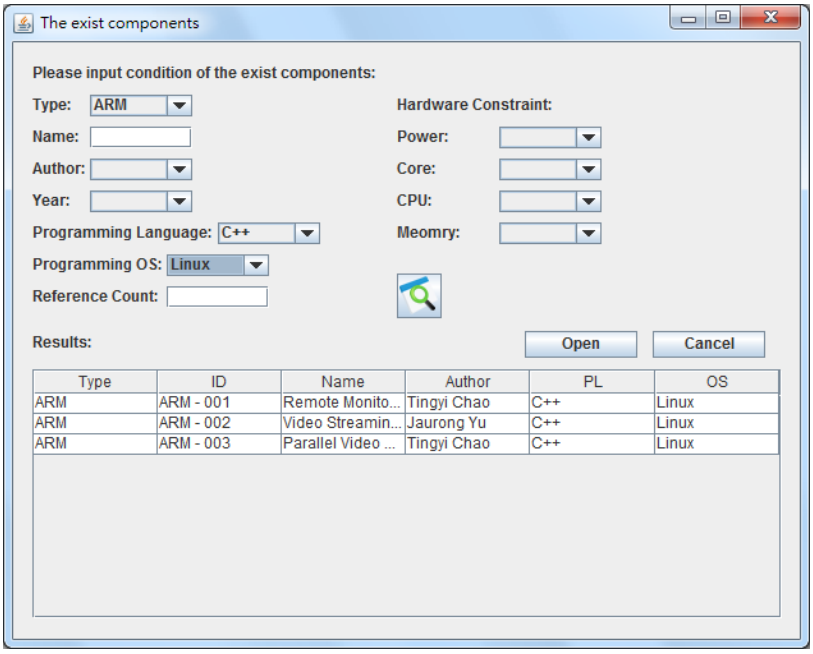

圖 4- 9 查詢結果

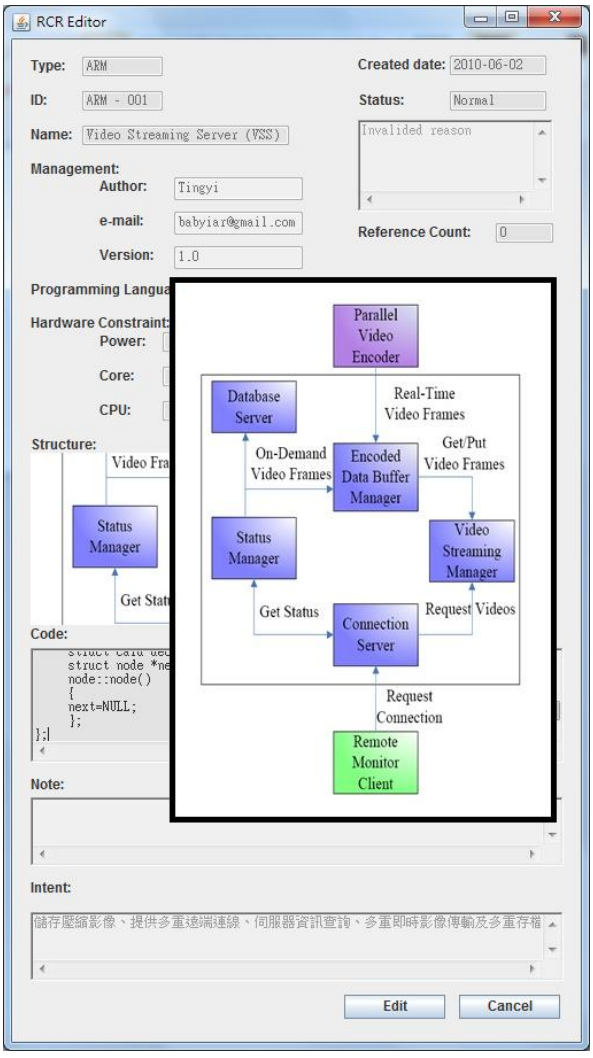

圖 4- 10 展開再利用元件

4.2.5 作廢再利用元件

若使用者要將再利用元件作廢，可在編輯再利用元件的過程中，將 Status 選擇成 Invalided，並填入作廢的原因，如圖 4- 11 為正常狀態的元件，若不建議他人再使用則選擇 Status 成 Invalided，並填入原因，如圖 4- 12 所示。

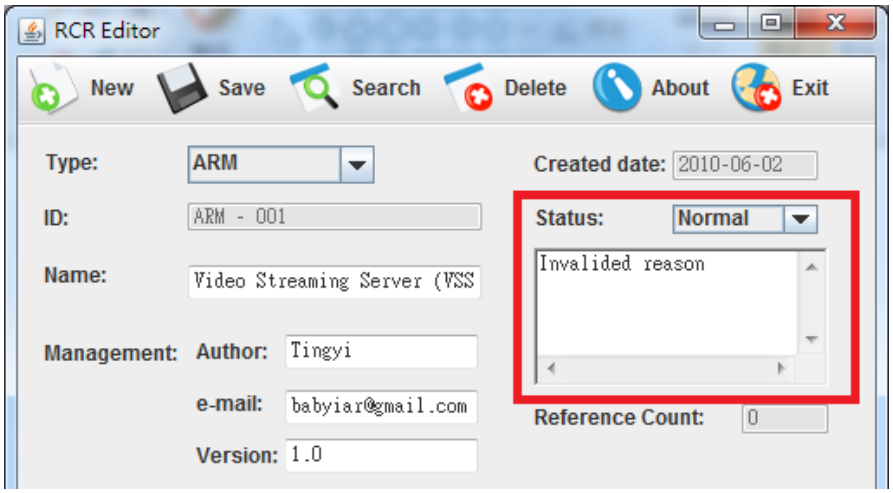

圖 4- 11 正常狀態的元件

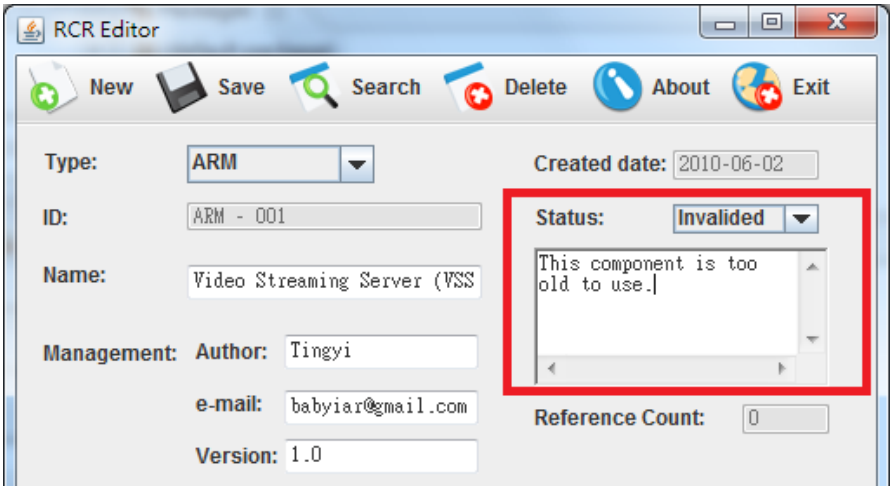

圖 4- 12 作廢狀態的元件

第5章　結論與未來方向

由於開發時程與系統發展趨勢會受到市場需求的影響，因此若能減少系統開發時程，同時減少開發成本，則能提高產品價值(低成本、高效率)。在本研究中使用了再利用元件資料庫來降低開發的時程，此種設計方式可以減少重複撰寫元件的發生，雖然並非適用於所有的元件，但可以大幅提升效率。本研究中進行再利用元件庫的設計探討，提出以 XML 為基礎並搭配原生型 XML 資料庫，減少儲存所需的時間，提升系統整體效能。進而達到縮短開發時程、增加可靠性及降低處理風險等好處。

未來可再加強再利用元件的規範及搜尋的強度，讓使用者可以更精確的搜尋到想要的元件。還可進一步的整合不同的軟體開發階段，使各開發階段都可以做到重用的動作，降低開發成本，然後更為完善。且可延伸此方法之精神，結合 XML 文件對資料存取的優勢，專注於一嵌入式平台上的再利用元件方法，能提供更準確的元件給使用者。

參考文獻

[1] The Institution of Engineering and Technology-the IET, http://www.theiet.org/ (2010.4.12 browsed)

[2] TechInsights, "*Embedded designers on tighter schedules, juggling multiple projects in 2008*" http://www.ubmtechinsights.com/ (2010.4.19 browsed)

[3] 洪聖敏(2009)，《*行動裝置之熱門嵌入式應用軟體趨勢與商機*》。經濟部產業管理委託研究計畫，ISBN:9789575813963。

[4] IEEE computer society, "Guide to the Software Engineering Body of Knowledge (SWEBOK)". http://www.computer.org/portal/web/swebok (2010.4.12 browsed)

[5] Wikipedia, "Software engineering". http://en.wikipedia.org/wiki/Software_engineering (2010.4.12 browsed)

[6] I. Sommerville, *Software Engineering 6e*, 基峯資訊股份有限公司, 2004

[7] B. Shneiderman, "*Designing the User Interface: Strategies for Effective Human-Computer Interaction, 3rd edition*", 1997

[8] H. Deitel, et.al., XML How to Program, Prentice Hall, Upper Saddle River, NJ, Prentice Hall, 2001

[9] w3school, "XML Tutorial".

http://www.w3schools.com/xml/default.asp (2010.4.12 browsed)

[10] W3C, "XML Core Working Group Public Page" http://www.w3.org/XML/Core/ (2010.4.12 browsed)

[11] W3C, "XSL Transformations (XSLT)" http://www.w3.org/TR/xslt (2010.4.12 browsed)

[12] W3C, "XML Path Language (XPath)" http://www.w3.org/TR/xpath/ (2010.4.12 browsed)

[13] W3C, "Extensible Stylesheet Language (XSL)" http://www.w3.org/TR/xsl/ (2010.4.12 browsed)

[14] W3C, "The Extensible Stylesheet Language Family"

http://www.w3.org/Style/XSL/ (2010.4.12 browsed)

[15] Wikipedia, "Native XML database". http://en.wikipedia.org/wiki/XML_database (2010.4.20 browsed)

[16] Sedna, "About Sedna" http://modis.ispras.ru/sedna/index.html (2010.4.20 browsed)

[17] Sedna, "Documentation" http://modis.ispras.ru/sedna/documentation.html (2010.4.20 browsed)

[18] 吳俊融," *XML 文件再利用軟體工具*",國立台灣科技大學資訊工程系, 2007

[19] P.A. Hsiung , C.S. Lin , S.W. Lin , Y.R. Chen , C.H. Lu , S.Y. Tong ,W.T. Su , C.H. Shih , C.S. Koong , N.L. Hsueh , C.H. Chang ,“VERTAF/Multi-Core: A SysML-based Application Framework for Multi-Core Embedded Software Development,” *Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing* 2009 (ICA3PP09), pp. 303-314, 8-11 (2009.06).

[20] P.A. Hsiung, S.W. Lin, C.H. Tseng, T.Y. Lee, J.M. Fu, and W.B. See, “VERTAF: An Application Framework for the Design and Verification of Embedded Real-Time Software” *IEEE Transactions on Software Engineering*, 30(10), Octber 2004.

[21] XML:DB Application Programming Interface fir XML Database, http://xmldb-org.sourceforge.net/xapi (2010.4.20 browsed)

[22] W3C, XQuery 1.0: An XML Query Language, http://www.w3c.org/TR/xquery(2010.4.20 browsed)

[23] R. Porkodi, V. Bhuvaneswari, R. Rajesh and Amudha, T., ”An Improved Association Rule Mining Technique for Xml Data Using Xquery and Apriori Algorithm," *Conference on IACC 2009 IEEE International Advance Computing*, 2009, pp. 1510-1514.

[24] R.Jonathan, " XML Processing and Data Integration with XQuery," *IEEE Internet Computing*, 2007, vol.11, Is. 4, pp. 62-67.

[25] P. Reveliotis and M. Carey, " Your Enterprise on XQuery and XML Schema: XML-based Data and Metadata Integration," *22nd International Conference on Data Engineering Workshops*, Proceedings, 2006, pp. 80-80.

[26] H. Wang, X. Wang and W. Zeng, "A Research on Automaticity Optimization of KeyX Index in Native XML Database," *2008 International Conference on Computer Science and Software Engineering*, 2008, Vol. 4, pp. 700-703.

[27] H. Liu, J. Zhao, L. Wang, G. Lv and Y. Ji, "Native XML database technology of digital distribution network," *SUPERGEN '09 International Conference on Sustainable Power Generation and Supply*, 2009, pp. 1-5.

附件一、XML Schema

```
<?xml version="1.0" encoding="Big-5"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="version">
        <xs:complexType/>
    </xs:element>
    <xs:element name="uml">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value=" "/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    <xs:element name="structure">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value=" "/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    <xs:element name="status">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value=" "/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    <xs:element name="screen_size">
        <xs:complexType/>
    </xs:element>
    <xs:element name="reference_count">
        <xs:complexType/>
    </xs:element>
    <xs:element name="programming_os">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="&#9;"/>
```

```
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    <xs:element name="programming_language">
        <xs:complexType/>
    </xs:element>
    <xs:element name="power">
        <xs:complexType/>
    </xs:element>
    <xs:element name="other">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value=" "/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    <xs:element name="note">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value=" "/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    <xs:element name="name">
        <xs:complexType/>
    </xs:element>
    <xs:element name="memory">
        <xs:complexType/>
    </xs:element>
    <xs:element name="management">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="author"/>
                <xs:element ref="email"/>
                <xs:element ref="version"/>
                <xs:element ref="date"/>
                <xs:element ref="status"/>
                <xs:element ref="reference_count"/>
```

```
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="invalid_note">
        <xs:complexType/>
    </xs:element>
    <xs:element name="intent">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value=" "/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    <xs:element name="hardware_constraint">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="power"/>
                <xs:element ref="core"/>
                <xs:element ref="cpu"/>
                <xs:element ref="memory"/>
                <xs:element ref="screen_size"/>
                <xs:element ref="other"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="email">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value=" "/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    <xs:element name="date">
        <xs:complexType/>
    </xs:element>
    <xs:element name="cpu">
        <xs:complexType/>
    </xs:element>
```

```
<xs:element name="core">
    <xs:complexType/>
</xs:element>
<xs:element name="component">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="name"/>
            <xs:element ref="management"/>
            <xs:element ref="intent"/>
            <xs:element ref="structure"/>
            <xs:element ref="uml"/>
            <xs:element ref="code"/>
            <xs:element ref="programming_language"/>
            <xs:element ref="programming_os"/>
            <xs:element ref="hardware_constraint"/>
            <xs:element ref="note"/>
            <xs:element ref="invalid_note"/>
        </xs:sequence>
        <xs:attribute name="type" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value=""/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="id" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value=""/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
</xs:element>
<xs:element name="code">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value=" "/>
```

```
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="author">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value=" "/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
</xs:schema>
```