由後設資料驅動的動態著錄系統
Toward A Metadata-Driven Dynamic Cataloging System

林宗德、陳宏儒、施學琦
Tsung-Te Lin、Hung-Ru Chen、Hsueh-Chi Shih

國立雲林科技大學資管所
Department of MIS, National Yunlin University of Science and Technology

gmi102@mis4k.mis.yuntech.edu.tw

摘要

頻繁的使用者需求變更，經常導致資訊系統的開發和維護成本高於當初所預期。為解決此問題，本研究先由著錄系統著手，嘗試在需求變動頻繁的情況下，仍然能夠以極低的成本快速地更改著錄系統，以期符合使用者需求的變化。本研究所提出的系統架構，以後設資料為核心，當使用者透過親和的維護介面自行修改後設資料後，著錄介面就會自動跟著改變，不需要系統發展者以人工方式修改任何程式。這個系統架構已成功地應用於運作中的系統，達到快速回應使用者需求和降低系統變動成本的目的。

關鍵字：需求變更、動態系統、著錄系統、後設資料、XML Schema

Abstract

Frequent user requirement changes often lead to unexpectedly high software development and maintenance costs. Our attempt in attacking the problem started with the cataloging systems. The aim is to enable the rapid change of the catalog system in a cost effective manner so that user requirements may be fulfilled in spite of the frequentness of these changes. The proposed architecture centers on the metadata. End users can change the metadata through a user-friendly maintenance interface. The catalog system will change accordingly automatically, without any manual intervention in code modification. The architecture has been implemented in an online cataloging system and achieved its goals successfully.

Keywords: requirements changes, dynamic system, cataloging system, metadata, XML schema

一、簡介

1. 研究背景與動機

在環境快速變遷的今日，資訊系統需求的變動亦愈益頻繁，這些頻繁的需求變動，使得資訊系統必須經常隨之修改，導致資訊系統的開發及維護成本高出原來所預期。如何能快速地改變資訊系統以因應變動的需求，並降低資訊系統變動的成本，已成為資訊系統發展者的重要議題(Highsmith 2001)。

本研究先由著錄系統著手，在需求變動頻繁的情況下，能夠以極低的成本快速地更改系統，以符合使用者的需求。

在著錄系統的建置之初，通常會事先建立後設資料。所謂後設資料(metadata)，是指有關資料的資料(data about data)(Hillmann 2001)。建立後設資料的目的，在讓資料管理者和使用者可透過它來了解資料的結構，進而使用並管理資料。有了後設資料，著錄系統可以決定那一些欄位必須被記錄在資料庫中。接著，資訊系統開發者，即可依據後設資料去設計資料庫與使用者的著錄介面。

後設資料可以根據一些標準來建立。公眾檢索導向的後設資料標準以都柏林核心集(Dublin Core)為主要代表，所強調的是描述資料的通用性和簡單性。而另一類的後設資料標準則是以學科為導向的，著重於特定領域資訊的共同需求與著錄標準，如國際博物館協會文件委員會的 CIDOC、英國博物館文件管理委員會的 MDA、保羅・蓋提信託基金的 CDWA、AAT、ULAN…等後設資料標準(陳亞寧 2001)。但是目前沒有一種後設資料標準可以完全符合所有的需求，縱使有一致的後設資料標準，其客製化仍有其必要。而客製化後設資料的建立往往需要經歷許多的嘗試與摸索。換言之，在著錄系統建置的過程中，後設資料常常會不斷地變動，以符合"資料"上的需求。

每次後設資料有變動，著錄系統的程式亦必須跟著變動，尤其當後設資料非常複雜時，系統開發者必須花費大量的精力在使用者需求變動上。

2. 研究目標

本研究的目標是發展出一個方法，來解決由於後設資料的經常變動，使得著錄系統要隨之而經常變動，所造成的系統開發和維護成本增加的問題。在後設資料需要變動時，這個方法可以讓使用者自行修改其後設資料，而系統發展者不用以人工修改任何的程式碼，即可以讓著錄介面跟著改變，以符合變動的需求。因此，不管後設資料變動多少次，使用者都可以自行完成這個任務，而不需要系統發展者的介入。

本研究的應用範圍，適用於任何後設資料複雜，但工作流程單純，且時常會更動的的著錄系統。所謂複雜的後設資料，是指後設資料有很多欄位、或很多階層、或很多種分類。而單純的工作流程是指，資料在輸入完畢後可以直接存入資料庫，而不用再經過許多的程序。

二、文獻回顧

半自動資料資源出版(semiautomatic data resource publishing)被用來輔助快速開發網站應用程式(Fraternali 1999)，後設資料表格(metadata table) 被用來產生動態的網站介面(Weiner et al. 2001)，網站模型語言(WebML) 被用來

輔助設計者了解網站資料的呈現關係(Ceri et al. 2002)。但這些方法在後設資料變動時，雖然可以節省時間，卻仍然需要系統開發者的介入，以人工的方式來修改程式。

三、系統架構

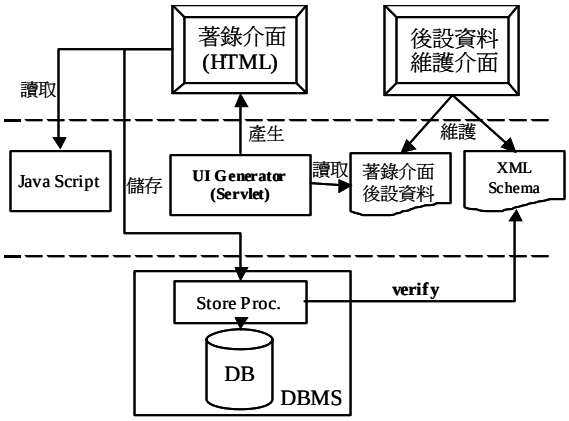


圖 1 系統架構

本研究所提出之系統架構，如圖 1 所示。

首先，使用者在建立後設資料時，本研究採用 XML Schema (Fallside 2001)來描述後設資料。這個 ***XML Schema*** 文件(圖 1 最右邊)決定了使用者需要輸入那些欄位，以及欄位間的階層關係。而***著錄介面後設***資料文件則是以 XML Schema 文件為基礎，在每個欄位加上產生著錄介面所需要資訊的描述。例如後設資料中若有一個名為「中文品名」的欄位， 在 ***XML Schema*** 文件中便會有一個其 name 屬性之值為「中文品名」的節點，而在***著錄介面後設資料***文件中，也同樣會有一個其 name 屬性之值為「中文品名」的節點，而且緊接在這個節點的後面，會有一串文字在描述「中文品名」這個欄位會用什麼樣的方式來讓使用者輸入。例如，該欄位應以文字輸入的欄位來顯示，且輸入方塊的寬度是 680 像素等。因為使用者可能不懂 ***XML Schema*** 文件和***著錄介面後設資料***文件的描述規則，所以本研究設計了一個***後設資料維護介面***，來幫助使用者更容易地建立這兩個文件。

本研究接著設計了一個***使用者介面產生器(UI Generator)***。當使用者從客戶端送出請求給使用者介面產生器時，從傳入的參數中，***使用者介面產生器***可以得知應該讀取那個***著錄介面後設資料***文件，並由此文件中取得產生***著錄介面***所需的資訊，以產生***著錄介面***，傳回給客戶端。客戶端瀏覽器(著錄介面)會讀取事先寫好的 ***Java Script*** 模組，來控制使用者和系統間的互動。當使用者輸入資料完畢之後，即送出結果給後端的***資料庫***，資料庫的內儲程序(***Stored Procedure***)會讀取 ***XML Schema*** 文件來檢核要存入資料庫的資料是不是一個有效(valid)的資料，如果是一個有效的資料就將之存入資料庫中。

1. XML Schema

本研究選擇 XML Schema 來描述後設資料，是因為它有足夠的能力來描述很複雜的資料關係。而且 XML Schema 是一個文字型態的檔案，符合 XML 文件標準，方便資料的交換和維護。

使用者在建立後設資料時，使用 XML Schema 來描述有那些欄位需要被記錄在資料庫中，並且以階層的方式來表現欄位(節點)間的關係。圖 2 顯示了一個 XML Schema 文件的片段。每一個節點(node)的內容包括：name(欄位的名稱)、maxOccrus(資料的最大出現次數)、minOccurs(資料的最小出現次數)、type(資料的型態)等屬性(attribute)。其中 minOccurs 若是 0，表示這個欄位可以不用輸入；若是 1，表示這個欄位至少要輸入一個值。

```
<element name="品名" minOccurs="1" maxOccurs="1">
   <complexType>
      <sequence>
         <element name="中文品名" type="string" minOccurs="1" maxOccurs="1"/>
         <element name="英文品名" type="string" minOccurs="0" maxOccurs="unbounded"/>
         <element name="功能" type="string" minOccurs="0" maxOccurs="1"/>
         <element name="歷史文化背景" type="string" minOccurs="0" maxOccurs="1"/>
      </sequence>
   </complexType>
</element>
<element name="紋飾" minOccurs="1" maxOccurs="unbounded">
   <complexType>
      <sequence>
         <element name="紋飾位置" type="string" minOccurs="0" maxOccurs="1"/>
         <element name="紋飾類別" type="string" minOccurs="0" maxOccurs="1"/>
      </sequence>
   </complexType>
</element>
```

圖 2 XML Schema 文件的片段

maxOccurs 若是 1，表示這個欄位最多只有一個值；若是 unbounded，表示這是一個多值的欄位，可以允許輸入多個值。舉個例子來說，某一項產品的英文品名若有多種不同的翻譯，那麼在「英文品名」這個節點的 maxOccurs 的值就會是 unbounded，可以輸入多個英文品名。多值的情況也有可能跨越多個節點。例如，某個物品可能有多個紋飾，而每個紋飾的資料都包含紋飾位置和紋飾類別兩個欄位。所以使用者會輸入多組包含這兩個欄位的紋飾資料，這種情況就是複雜的多值。

XML Schema 文件的主要功用，是當使用者輸入的資料要存入資料庫之前，可以用它來驗證所輸入的資料的有效性。

2. 著錄介面後設資料

圖 3 顯示了本研究所使用的另一類文件 — 著錄介面後設資料文件。此文件除了包含 XML Schema 文件的內容之外，緊接在每個節點的後面，另外增加了一些用來顯示著錄介面所需的資訊。由於本研究採用 Web-based 的用戶端程式，所以這些用來產生著錄介面的相關資訊，大部份是用來產生 HTML 裡的使用者輸入欄位。使用者輸入的欄位型態，大致上可分為：文字型態

```
<element name="品名" minOccurs="1" maxOccurs="1">
   <complexType>
      <sequence>
         <element name="中文品名" type="string" minOccurs="1" maxOccurs="1"/>
         <input type="text" name="chinese_item_name" id="中文品名" size="80" style="width: 680px"/>
         <element name="英文品名" type="string" minOccurs="0" maxOccurs="unbounded"/>
         <input type="text" name="english_item_name" id="英文品名" size="80" style="width: 680px"/>
         <element name="功能" type="string" minOccurs="0" maxOccurs="1"/>
         <select option1="食器" option2="酒器" option3="樂器"/>
         <element name="歷史文化背景" type="string" minOccurs="0" maxOccurs="1"/>
         <textarea name="background" id="歷史文化背景" cols="80" rows="20" wrap="off" style="width: 680px"/>
      </sequence>
   </complexType>
</element>
<element name="紋飾" minOccurs="1" maxOccurs="unbounded">
   <complexType>
      <sequence>
         <element name="紋飾位置" type="string" minOccurs="0" maxOccurs="1"/>
         <input type="text" name="line_position" id="紋飾位置" size="80" style="width: 680px"/>
         <element name="紋飾類別" type="string" minOccurs="0" maxOccurs="1"/>
         <input type="text" name="line_category" id="紋飾類別" size="80" style="width: 680px"/>
      </sequence>
   </complexType>
</element>
```

圖 3 著錄介面後設資料文件的片段

(text)、文字區塊(textarea)、下拉式選單(select)等。並可指定其欄位的顯示寬度、文字大小、字型、內定值等資訊。

其語法和 HTML 很像。

3. **使用者介面產生器**

使用者介面產生器是一支 Java

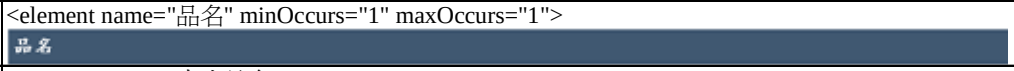
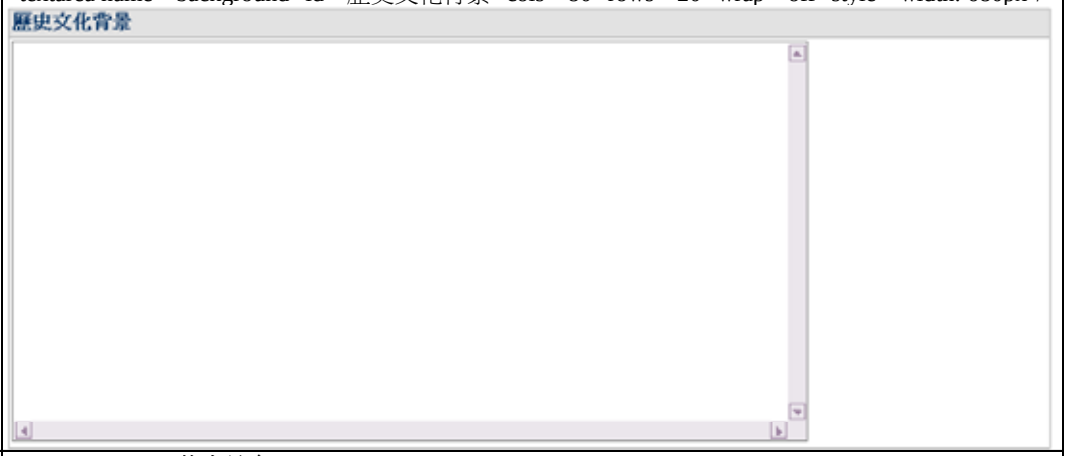
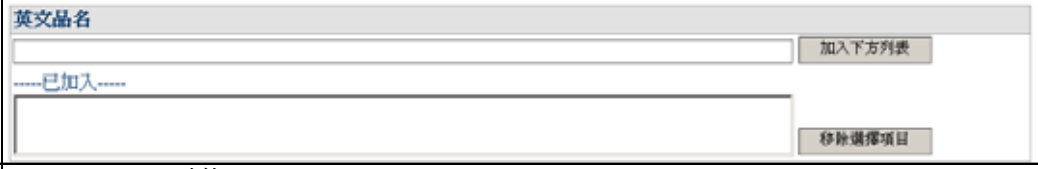
項	著錄介面後設資料及產生之使用者介面
1	<element name="品名" minOccurs="1" maxOccurs="1"> 
2	<element name="中文品名" type="string" minOccurs="1" maxOccurs="1"/> <input type="text" name="chinese_item_name" id="中文品名" size="80" style="width: 680px"/> 
3	<element name="歷史文化背景" type="string" minOccurs="0" maxOccurs="1"/> <textarea name="background" id="歷史文化背景" cols="80" rows="20" wrap="off" style="width: 680px"/> 
4	<element name="英文品名" type="string" minOccurs="0" maxOccurs="unbounded"/> <input type="text" name="english_item_name" id="英文品名" size="80" style="width: 680px"/> 
5	<element name="功能" type="string" minOccurs="0" maxOccurs="1"/> <select name=”item_function” id=”功能” option1=”食器” option2=”酒器” option3=”樂器”/> 
6	<element name="紋飾" minOccurs="1" maxOccurs="unbounded"> <complexType> <sequence> <element name="紋飾位置" type="string" minOccurs="0" maxOccurs="1"/> <input type="text" name="line_position" id="紋飾位置" size="80" style="width: 680px"/> <element name="紋飾類別" type="string" minOccurs="0" maxOccurs="1"/> <input type="text" name="line_category" id="紋飾類別" size="80" style="width: 680px"/> </sequence> </complexType> </element> 

圖 4 六種不同的剖析情況

Servlet 程式，用來產生著錄介面。當客戶端送出請求給使用者介面產生器，由傳過來的參數可以知道應該讀取那一個著錄介面後設資料文件。接著利用 XML 剖析器剖析這個文件，依序產生欄位名稱或輸入的欄位。

WebML(Ceri et al. 2002)將網頁的內容單位(content unit)分成五種型式。資料(data)、多資料(multidata)、索引(index)、卷軸(scroller)以及資料單(data entry)。

本研究則依照資料結構的不同列出了六種不同的剖析情況，如圖 4 所示。第一項，節點中沒有 type 屬性，表示這個節點只是一個抬頭，沒有使用者輸入的部份，所以只要將 name 屬性的值顯示出來就可以了。

第二項，上面的節點中有 type 屬性，表示這個欄位包含使用者輸入的部份，那麼緊跟在後的節點就是使用者輸入部份的描述。上面節點的 name 屬性顯示成欄位名稱，下面的節點經過剖析之後，發現這是一個文字型態的輸入欄位，而且是單值的欄位(max Occurs= “1”)。所以只要把各個屬性重新組合起來就是一個 HTML 的節點了。

第三項，與第二項的情況一樣。但是下面的節點經過剖析之後，發現是一個文字區塊，所以重組之後的輸入欄位是一個文字區塊。

第四項，是一個多值的欄位(maxOccurs= “unbounded”)。所以除了產生與第二項相同的欄位名稱和輸入的欄位之外，還另外產生一個列示欄位、一個新增按鈕和一個刪除按鈕。使用者在輸入欄位中輸入資料後，按新增按鈕把輸入的資料放入列示欄位中，故可以在列示欄位中存放多個值。也可以透過刪除按鈕移除列示欄位中的某一個值。這些使用者和系統的互動控制是透過 Java Script 來執行的。這些 Java Script 是經過一般化的程式，可以適用於任何多值的欄位。

第五項，是一個單值的欄位。但是下面的節點經過剖析之後，發現是一個下拉式選單。所以後面的 option1~option(n)就是選單的項目。

第六項，是一個複雜多值。第一個節點的 name= “紋飾”，其 maxOccurs = “unbounded”，而且緊接在這個節點之後的是一個 complexType 節點。被包含在這個節點裡面的有兩個欄位，分別是紋飾位置和紋飾類別。換言之，紋飾節點可以有多組包含紋飾位置和紋飾類別兩個欄位的資料。這裡所舉的例子是一個非常簡單的複雜多值的狀況。複雜多值可以包含以上五種的任何狀況，包括多值的欄位。更複雜的狀況是，複雜多值中可以包含複雜多值，形成巢狀的複雜多值。在複雜多值中除了新增和刪除按鈕之外，還有上一筆按鈕、下一筆按鈕和一個下拉式選單，以便查看有多少組資料在裡面。這些控制也是由經過一般化的 Java Script 來執行的。

當需要輸入的欄位很多時，如果全部都顯示出來會使畫面拉得很長。為了

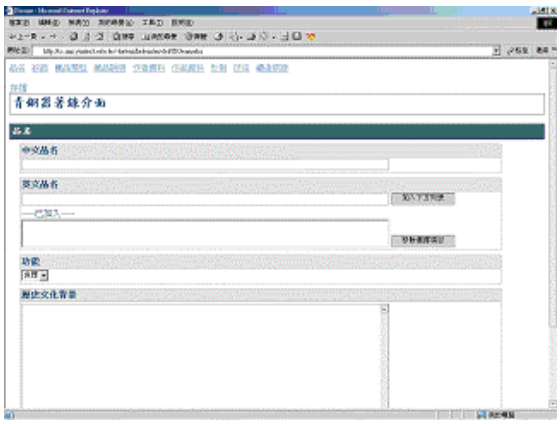


圖 5 自動產生的著錄介面

讓畫面看起來簡潔一點，使用者介面產生器會先把第一層的節點顯示在最上面。當使用者點選了某一個第一層的節點時，畫面就只顯示該第一層節點那一段的欄位，而把其他段的欄位隱藏起來。如圖 5 所示。

4. 後設資料的變動

由本研究提出的架構得知，如果使用者要求變動後設資料，我們只要變動 XML Schema 文件和著錄介面後設資料文件就可以了。使用者介面產生器在讀取變動後的著錄介面後設資料文件後，就會自動產生變動後的著錄介面。而在存檔時,也可以透過變動後的XML Schema 文件，來判斷送過來的資料是不是符合新的 XML Schema 文件的規定。

本研究的目標是，當後設資料有所變動時，可以由使用者自行完成變動的工作，而不用系統開發者進行人工修改。但是使用者通常不懂 XML Schema 和著錄介面後設資料的描述規則，所以必須建立一個後設資料維護介面讓使用者操作(如圖六)。透過這個介面，使用者可以不用了解這兩個文件複雜的描述規則，也能夠對這兩個文件進行變動維護的動作。

圖六 後設資料管理介面

後設資料維護介面，主要是針對著錄介面後設資料文件進行維護。而只要把其中不屬於 XML Schema 的部份拿掉，另外存成一個檔案，就是 XML Schema 文件了。

5. **擴充性**

本研究的系統架構有很好的擴充性。除了前面六種剖析狀況所提到的元件外，其他如 check box、radio button…等元件也都可以任意地加入剖析的程式中。

如果希望限制使用者輸入的值的範圍(domain)時，也可以在後設資料中加入限制輸入值的範圍，以得到更嚴格更精確的值。

另外，有許多的共通的資訊，可以引用自其他資源，就是所謂的權威檔。例如，人名、地名等，已經有其他單位整理出共通的後設資料，也可以加上連結，直接找到符合的資料引用進來。

四、 結論

本研究的系統統架構已成功地應用在數位典藏國家型科技計畫的國立歷史博物館數位典藏系統的後設資料建置子系統中。該系統之目的是將歷史博物館的典藏品(如青銅器、國畫、瓷器、版畫等)相關基本資料儲存於資料庫中，以便查詢。歷史博物館數位典藏計畫的後設資料非常複雜，但工作流程單純，因此很適合使用本研究的方法來發展。面對這樣複雜的後設資料，本研究證明這個架構可以減少因為後設資料的時常變動而產生的維護成本。並且可以讓使用者有更多的自由來進行後設資料變動。

對於未來的研究，有幾點值得我們再投入開發的：

- 在這個研究中我們假設有一個資料庫可以存放 XML 資料。這個資料庫可以用原生 XML 資料庫 (Native XML database) 或是關聯式資料庫 (Relational database) 來實作。我們將針對幾種不同的資料庫型態及架構進行效能的比較，包括大量資料檢索、存取速度、以及允許後設資料變動的彈性…等。
- 利用 XSL/XSLT 的技術，來做文件的轉換，讓使用者可以自行進行網頁排版的功能，讓網頁的呈現方式更自由。
- 後設資料的版本轉換時，如何比對新舊版本的不同，以很有彈性的方式自動更新資料庫綱要(schema)和資料。

參考文獻

1. 陳亞寧、陳淑君，2001『Metadata 在數位博物館之發展與分析』，圖書館學與資訊科學，第 27 卷‧第 2 期：52~71 頁
2. Ceri, S., Fraternali, P. and Matera, M. “Conceptual Modeling of Data-Intensive Web Applications,” IEEE Internet Computing, vol. 6, no. 4, 2002, pp.20-30
3. Fallside, D. C. “XML Schema Part 0: Primer W3C Recommendation,” http://www.w3.org/TR/xmlschema-0/, May 2001
4. Fraternali, P. “Tools and Approaches for Developing Data-Intensive Web Applications: A Survey,” ACM Computing Survey, vol. 31, no.3, Sept. 1999, pp.227-263
5. Highsmith, J. and Cockburn, A. “Agile Software Development: The Business of Innovation,” IEEE Computer, Sept. 2001, pp.120-122
6. Hillmann, D. “Using Dublin Core,” http://dublincore.org/documents/usageguide/, April 2001
7. Weiner, M., Sherr, M. and Cohen, A. “Metadata Tables to Enable Dynamic Data Modeling and Web Interface Design: The SEER Example,” International Journal of Medical Informatics, vol. 65, issue 1, 2002, pp.51-58