第 4 章 系統設計

4.1 領域物件模型

藉由第 3 章的系統分析，我們為供稿系統建立了如圖 4-1 的初步領域物件模型（Domain Object Model），並以此模型作為後續發展的基礎，開發完整的系統。圖中可分為三個主要概念：與 NewsML 相容之新聞內容架構、支援多種類型新聞稿件，以及新聞中心模式的組織架構。

由於 NewsML 已經成為國際上各個重要的新聞組織如 Reuters、NSK 等所採用並且大力推廣，已經成為傳輸新聞內容上的標記語言標準。在本研究中的外部供稿系統部份，也將採用 NewsML 作為封裝新聞內容的標記語言，因此在物件設計上，即以 NewsML 的架構作為參考，使將來的格式轉換能更為簡便。每則完整的新聞可能需要不同類型的新聞稿件來作為描述上的輔助，因此我們將一則完整的新聞視為多個稿件的集合，即圖中之 NewsSet 物件。NewsSet 包含不同類型之稿件，而這些稿件皆繼承自 News 物件。

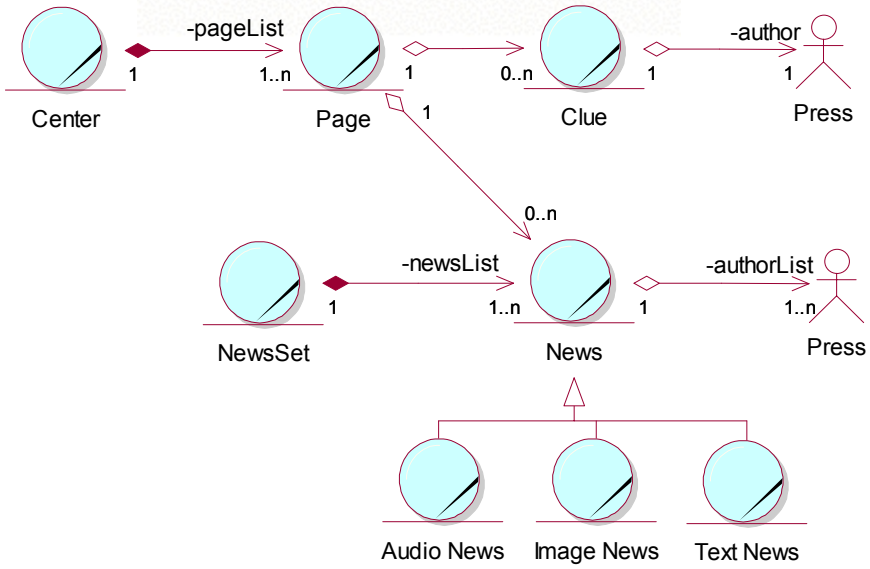

圖 4-1、供稿系統領域物件模型

在本研究中，我們實做了文字、影像與語音等三種不同類型的新聞稿件。由於一個 NewsSet 中不同類型的稿件大多由不同記者負責，且這些稿件也可能分屬不同新聞中心。因此在系統中，仍然以稿件作為新聞管理的基本單位。

對於記者與採訪主任而言，NewsSet 是系統中的隱含物件，在平常的編採流程中並不會看到 NewsSet 的存在。只有在 3.6 節所提到採訪主任指定相關稿件時，才會將兩個稿件併入同一個 NewsSet 物件中。而閱讀新聞時，也會將同一個 NewsSet 中的其他稿件，一併列出供使用者參考。

領域模型中的另一項基本概念為組織架構的規劃，本研究採用新聞中心模式的雙層式組織架構，每個新聞中心物件可包含多個新聞版面。雖然線索與稿件建立時是供稿給新聞中心，但最終將會被歸到各個新聞版面之下。我們以大學報編採組織為例，系統所預設的編採組織規劃如圖 4-2（a）。當編採團隊將組織架構改為採用新聞版面模式時，則如圖 4-2（b）所示，我們替每個新聞版面設立對應的新聞中心，以此例而言，即為設立八個新聞中心，每個中心僅包含一個新聞版面。記者供稿時，如果該新聞中心只有單一版面時，系統及會自動將該稿件分派至此版，藉此支援新聞版面模式的組織架構。

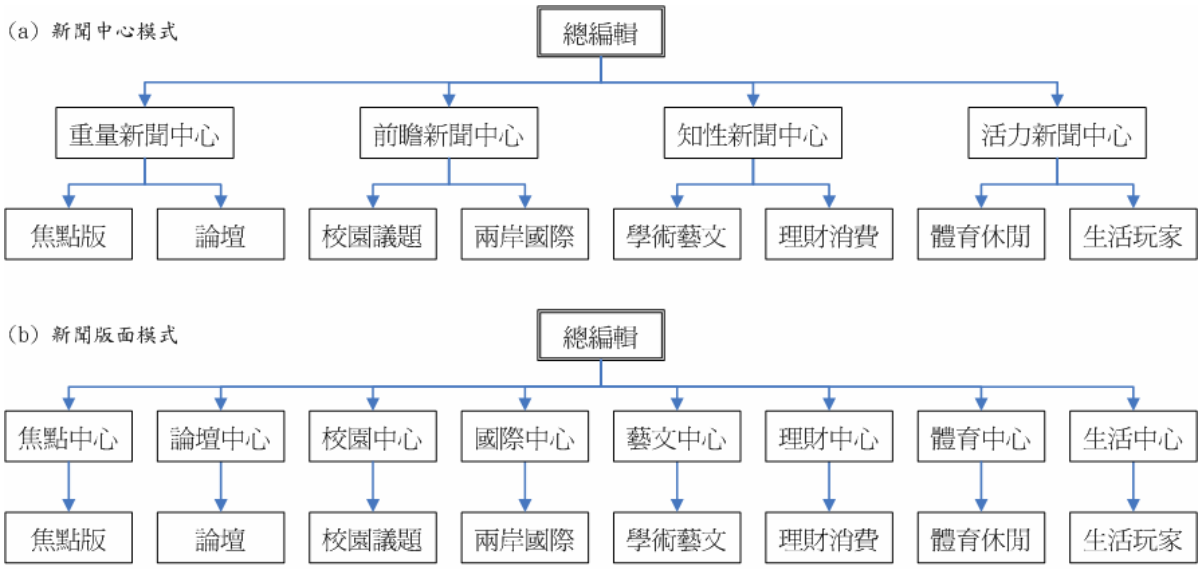

圖 4-2、供稿系統組織範例 (a) 新聞中心模式 (b)新聞版面模式

4.2 權限架構

4.2.1 存取控制策略

為了建立一套能彈性適應不同組織架構的供稿系統，我們需要一套完善的權限管理機制來幫助我們達到這個目的。在討論存取控制策略前，我們先將權限簡單地定義為「主體（Subject）對於物件（Object）執行某種操作（Operation）的權力」。主體可能是某位系統上的使用者，或是另一個應用程式，物件則包含文件，檔案等。常見的存取控制策略，可分為強制存取控制、自由裁量存取控制與以角色為基礎之存取控制三種[7]。

強制存取控制（Mandatory Access Control，MAC）中只有系統管理者能對權限的設定作控管，其主要概念為將系統中的物件分類並給予不同的安全層級。設定權限時，同樣賦予主體不同的安全層級，每個主體只能操作安全層級與自己權限相同或較本身的安全層級低的物件。而針對我們希望對於資訊的保護方式，強制存取控制也可分為 Bell-LaPadula（BLP）與 Biba 兩種模式。

BLP 模式的特性為上讀／下寫，如圖 4-3（a）所示，上層的主體只能對下層物件做讀取的動作，而下層的主體亦僅能寫入上層的物件。在 BLP 模式下，可以避免層級較高的主體將機密資訊寫入層級較低的物件，而將資訊外洩，進而確保資訊的隱密性。

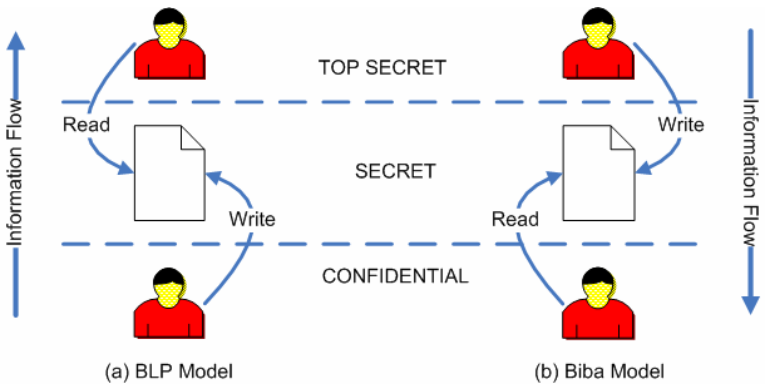

圖 4-3、強制存取控制模式 (a) BLP 模式 (b) Biba 模式

MAC 的另一種常見模式為 Biba 模式，此模式與 BLP 的特性正好相反，為下讀／上寫。從圖 4-3（b）可看出，上層主體只能對下層的物件作寫入的動作，而下層主體也僅能讀取上層物件。相對於 BLP 模式保障資料的隱密性，Biba 模式則是限制下層主體無法寫入上層物件，來保障資料的完整性，在此模式下的資訊流，為由上往下傳遞。

相較於 MAC 是由系統管理者來進行權限控管，自由裁量存取控制（Discretionary Access Control，DAC）則是由物件的擁有者來自行設定該物件之權限。在自由存取控制，每個物件都擁有自己存取控制串列（Access Control Lists，ACL）。物件擁有者可透過設定存取控制串列，來指定其他主體對於此物件的存取限制。當其他主體要存取物件時，系統即會檢查此存取控制串列，確認該主體是否擁有足夠的權限。DAC 的優點在於對權限的設定較有彈性且易於擴展，而實做上也常可見到 MAC 與 DAC 互相搭配的應用。

由於現今資訊的分享程度與企業間交易型態複雜化，強調分工合作與分權制度[24]，以及傳統的存取控制策略在維護上的成本很高，因此許多不同的權限管理模式被提出來。以角色為基礎的存取控制即為其中之一，其主要的功能在於減少管理存取控制的成本與複雜度，同時能適用於大型組織[19]。

以角色為基礎的存取控制（Role-Based Access Control，RBAC）在 1970 年代被提出來。RBAC 的主要概念為不直接將權限賦予主體，而是在主體與權限中間創造一層「角色」。每個角色可被賦予一到多個權限，而主體可對應一或多個角色，系統根據主體所對應的角色是否具備相關權限來做存取控制。

RBAC 透過角色的設定，將低階的存取限制描述，轉為實際工作流程中高階的角色描述，使權限控制更為清楚明瞭。對於新聞編採組織而言，當組織中的工作有異動時，也不需要逐一設定使用者的各項權限，只需要修改其對應的角色即可。尤其如大學報或政大之聲等實習單位，每學期都會有組織上的更動，採用 RBAC 更能降低管理成本。因此在供稿系統的設計上，我們也採用 RBAC 來作為存取控制的策略。

4.2.2 以角色為基礎的存取控制模式

RBAC 的概念被提出之後，由於其適合用於大型組織與管理較為簡便的特性，而被廣泛地討論與應用。但是卻沒有一套統一的標準，因此美國國家標準局（National Institute of Standards and Technology，NIST）提出一套標準，將 RBAC 歸納為單層式角色存取控制（Flat RBAC）、階層式角色存取控制（Hierarchical RBAC）、條件限制式角色存取控制（Constrained RBAC）與對稱式角色存取控制（Symmetric RBAC）四種模型[42][43]：

Flat RBAC 定義了 RBAC 的基本概念，如圖 4-4 所示，使用者並不被直接賦予權限，而是透過其所扮演的角色來取得相關權限。所有實作 Flat RBAC 的系統都必須要能支援使用者與角色的多對多關係，亦即可以多個人扮演同個角色，而一個人也能同時兼具不同角色；角色與權限間是多對多的關係，每個角色可被賦予多項權限，而不同角色間也可同時擁有同一項權限。

使用者可同時使用其扮演的不同角色之權限；以供稿系統而言，我們為每個新聞中心的記者定義一個角色，只有該新聞中心記者可供稿至所屬的新聞中心。如果有一位使用者同時具備兩個新聞中心記者的權限時，則該使用者應該可同時對兩個新聞中心供稿，而不需要重新切換不同角色。

在管理上，Flat RBAC 要能夠方便的檢視使用者與角色間的關係，讓系統管理員可容易了解一位使用者被賦予哪些角色，以及每個角色已被哪些使用者所扮演，而角色與權限間的關係也應該有相同的功能。

圖 4-4、單層式角色存取控制[43]

Flat RBAC 為 RBAC 中最基本的模型，NIST RBAC 中另外三個模型皆以此模型為基礎發展。

Hierarchical RBAC，除了必須滿足 Flat RBAC 的條件之外，另外實作了角色的階層架構（Role Hierarchy），使權限架構能更符合企業組織的運作方式。

Hierarchical RBAC 可提供兩種優點，從下而上的觀點來看，透過階層架構可提供基本權限的設定。以供稿系統為例，我們可先定義基本使用者的角色，其權限只能看見一般公告。而各新聞中心也分別定義其基本使用者的角色，除了包含系統使用者的權限外，也可參與各新聞中心的討論。在往上一層，則為各新聞中心的記者與編輯，除了包含新聞中心使用者的角色外，也分別擁有對其新聞中心供稿與編輯的權限；而由上而下的觀點來看，角色的階層架構則符合了組織中部門主管通常包含其部門員工之權限。例如新聞中心主任可能包含了該新聞中心記者與編輯的權限，而總編輯則包含了所有新聞中心主任的權限。

有時我們並不希望部門主管繼承太多的權限，此時可透過定義虛擬角色（Private Role）的方式來避免主管取得過多的權限[42]。圖 4-5 為 Hierarchical RBAC 的角色階層圖範例，我們分別定義了虛擬記者與虛擬編輯兩個虛擬角色，賦予其較一般且可被新聞中心主任繼承的權限。而記者與編輯則分別在繼承自己的虛擬角色，另外新增不希望被新聞中心主任所繼承的權限部分。

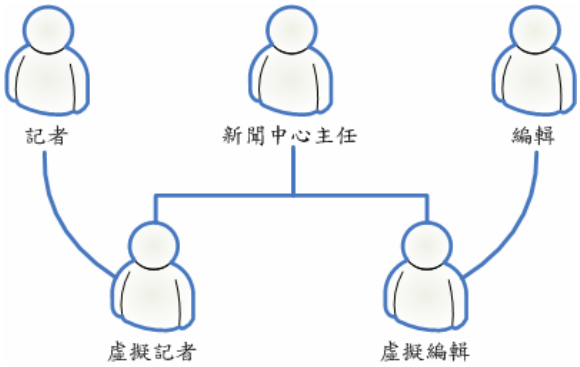

圖 4-5、Hierarchical RBAC 階層圖

有些系統會為了效能或其他考量，對於角色的階層架構有所限制，例如其階層必須為樹狀結構等。我們將對角色的階層有設定限制的角色存取控制稱為 Limited Hierarchical RBAC。相反地，對於角色階層沒有設定任何限制的系統，則稱為 General Hierarchical RBAC。

在企業組織上，我們希望能做到權責分明（Separation of Duties，SOD），例如會計稽核制度中有項基本原則，及管錢不管帳，而管帳的人不管錢。因此在存取控制策略上，也應該避免此類弊端發生，這就是 Constrained RBAC 的主要概念。除了滿足 Hierarchical RBAC 的需求外，還要能針對角色的賦予動作加上條件限制（Constraints），以符合組織運作上的實際情形。圖 4-6 為 Constrained RBAC 的基本模型，其中從系統監控的範圍不同，可分為 Static SOD 與 Dynamic SOD 兩種模式。兩者的主要差異，在於是否對於工作階段（Sessions）中的角色做條件限制檢查的動作。

工作階段的概念是 RBAC 為了避免使用者擁有過多的權限，並希望使用者在每個工作都僅使用最適當或最少的權限（Least Privilege）來作業。因此允許使用者在每個工作階段中，使用不同的角色來執行相關操作。

在 Static SOD 中，並不會對工作階段中的使用者角色作檢查，因此在系統管理員進行角色賦予動作時，採用較嚴格的檢查機制，使用者無法被賦予兩個互斥的角色。另一方面，Dynamic SOD 在角色賦予時，可允許使用者被賦予互斥的角色，但是每個工作階段上，無法使用兩個互斥的角色同時作業。

NIST RBAC 模型中的最後一個模型，為 Symmetric RBAC。雖然 Constrained RBAC 會針對角色的賦予動作做檢查，但是在角色權限變更之後，仍然可能會產生疏失而造成權責區分不明的情形發生。為了解決此一問題，如圖 4-7 所示，Symmetric RBAC 除了檢查對角色賦予的動作之外，也會針對權限的賦予動作做檢查。而與 Constrained RBAC 相同，Symmetric RBAC 也由監控範圍不同而分為 Static SOD 與 Dynamic SOD 兩種模式。

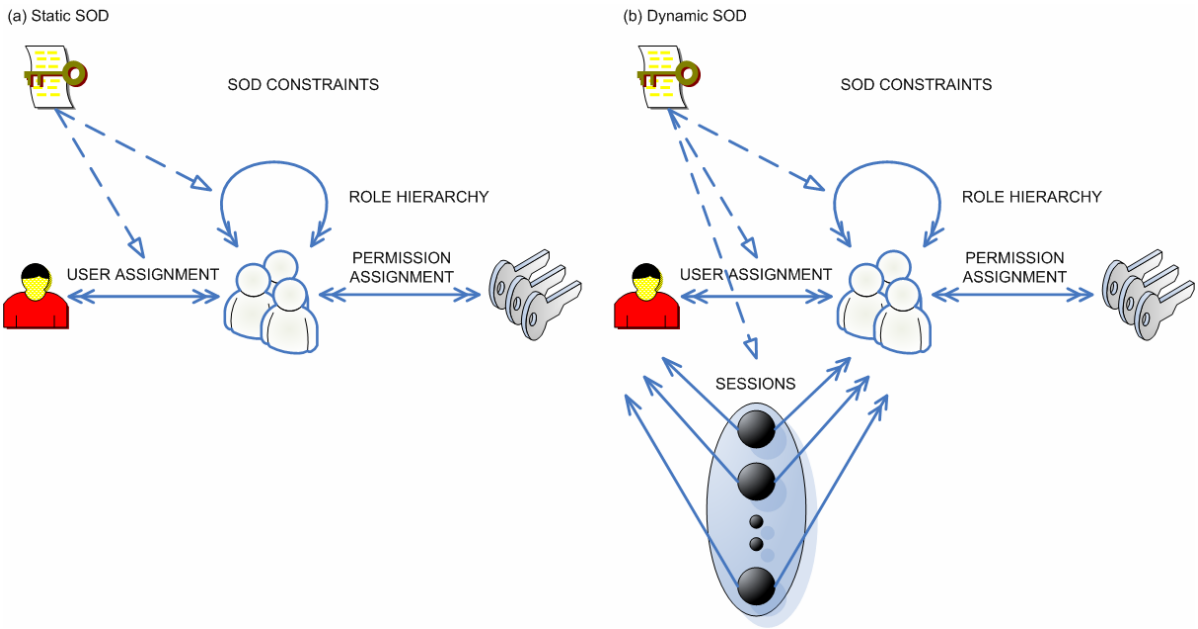

圖 4-6、Constrained RBAC Model (a) Static SOD，(b) Dynamic SOD[43]

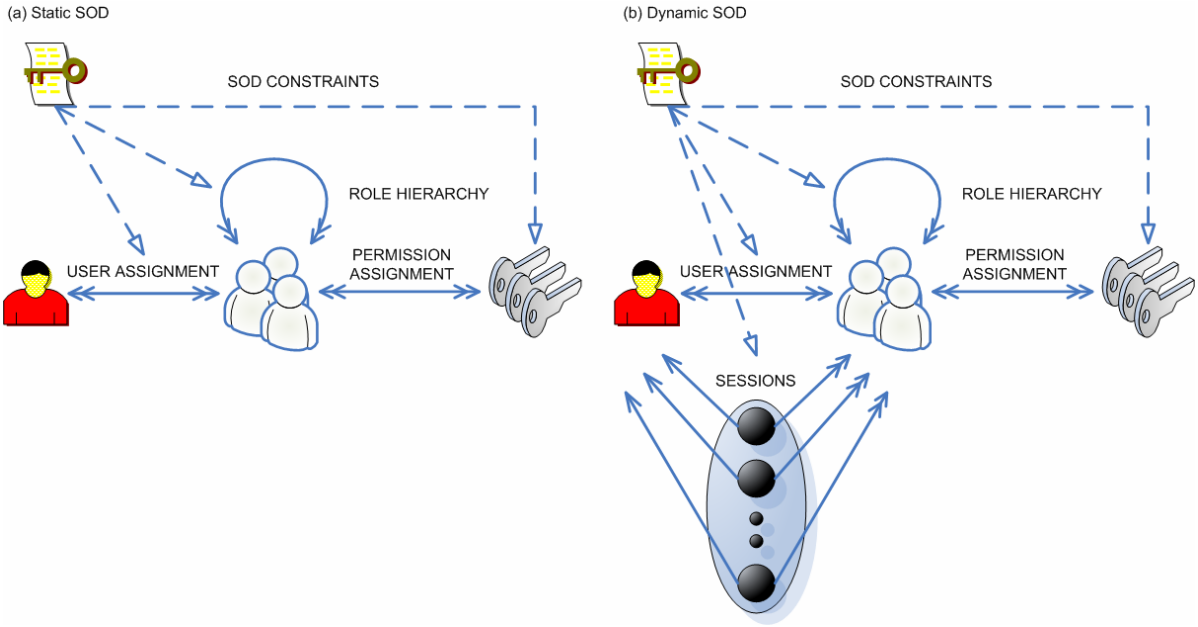

圖 4-7、Symmetric RBAC Model (a) Static SOD，(b) Dynamic SOD[43]

4.2.3 供稿系統中之存取控制模式

雖然 NIST RBAC 模型可以符合大部分企業在組織架構上的需求，但卻沒有提供對於需要針對事件流程或物件狀態而做的存取控制機制與標準。因此許多文獻開始針對此議題作探討，而較常被提出來的做法，皆為藉由擴充 RBAC 模型中的 Constraint，由原本僅針對角色或權限間互斥的靜態條件，延伸為可針對系統流程、物件狀態或時間等資訊來做動態的監控[28][34][38]。

在新聞編採流程中，有許多權限會隨著時間、物件的資訊以及狀態而不同。以稿件為例，採訪主任會因為稿件所屬的新聞中心不同，而決定是否擁有管理的權限。在 3.6 節也曾提到，記者、採訪主任與編輯也會因為稿件的狀態不同，而擁有編輯的權限；因此對新聞供稿系統而言，雖然採用 RBAC 模型作為存取控制策略可以符合新聞編採組織的架構，但仍無法完全滿足系統的完整需求。

由於上述需求，本研究採用實做 NIST RBAC 模型中的 Symmetric RBAC 模型，同時針對 RBAC 中的 Constraint 作擴充，成為可針對物件與其他系統資訊做監控的動態條件限制，以此作為新聞供稿系統中的存取控制模型。

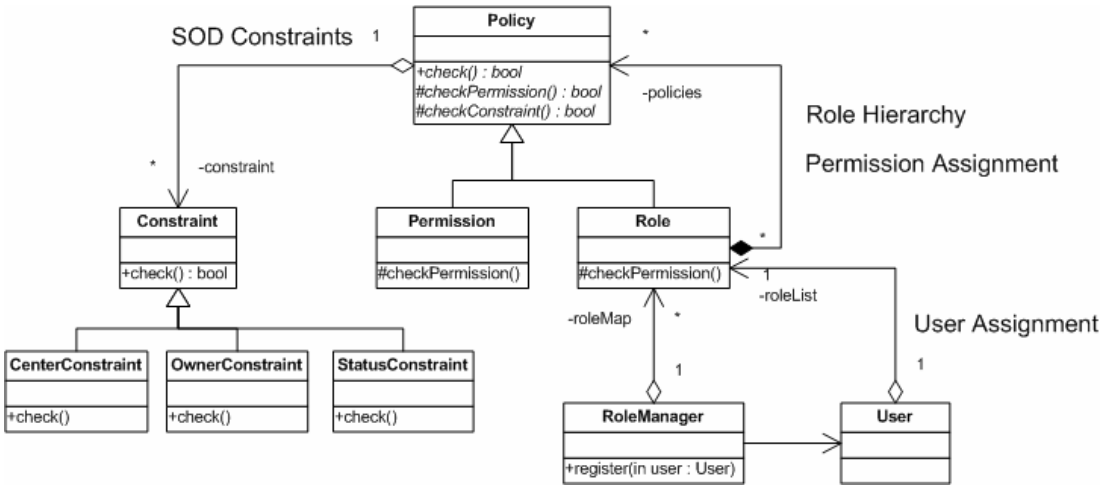

圖 4-8、權限架構類別圖

```
CHECK(user, object, operation)
  RESULT←false;
  // 檢查使用者是否具備操作此物件之權限
  if CHECK_PERMISSION(object, operation) then
    // 檢查是否被其他條件所限制
    if CHECK_CONSTRAINT(user, object) then
      RESULT←true;
  return RESULT;
```

圖 4-9、Policy check 方法

圖 4-8 為新聞供稿系統之存取控制架構類別圖，我們定義存取策略（Policy）類別作為檢查存取控制的基本形別，而角色（Role）與權限（Permission）皆是繼承自 Policy 類別。

在角色的階層架構上，我們使用合成樣式（Composite Pattern）來實作 Hierarchical RBAC 模型。Policy 為 Role 與 Permission 共同的父類別。Permission 為合成樣式中之 Leaf 參與者，不再放入其他的類別。Role 類別則為合成樣式中之 Composite 參與者，我們可在 Role 類別中再加入多個 Policy 類別，這些 Policy 類別可能是 Permission 或是其他的 Role 類別，藉此實做出角色的階層架構。

另一方面，我們採用 Template Method 樣式來設計權限驗證的機制，Policy 為 Template Method 樣式中之抽象類別，包含了 Template Method「check」(如圖 4-9 所示)，以及 check 方法中所使用到的 checkPermission 抽象方法。Role 與 Permission 則為 Policy 之具象類別（Concrete Class），分別覆寫 checkPermission 方法來實做不同的權限檢查機制。

我們在 Policy 類別加上了條件限制類別（Constraint）用以實做 Symmetric RBAC，所有的存取除了必須擁有適當權限之外，還必須要不被其他條件所限制，才可通過權限驗證。Constraint 為一抽象類別，我們可以視系統所需加入不同的條件限制。這些條件限制除了可為 NIST RBAC 模型中所提針對角色與權限間的互斥做檢查的靜態限制外，也可以根據物件的相關資訊來做動態的限制。

舉例而言，我們想規範記者只能在稿件提交之前，或是處於「駁回修正」的狀態時才能修改稿件。便可實做出狀態限制（Stat Constraint）來限定記者對於稿件的編輯權限，只有在特定狀態才有效；除了狀態限制之外，本研究中另外實作了新聞中心限制、討論區限制與擁有者限制三項動態限制，分別限制使用者的權限只有物件屬於特定的新聞中心、討論區，甚至必須是物件的擁有者才能執行相關權限。供稿系統中的大部分設計，也都以提供動態權限的方式發展。

以下我們將從使用者登入開始介紹供稿系統中的權限驗證流程，圖 4-10 說明了使用者登入系統流程之概念圖，由於角色及權限等資料並不會常常變動，因此我們在供稿系統啟始時即將角色、權限等資訊放入角色管理員（Role Manager）中管理。

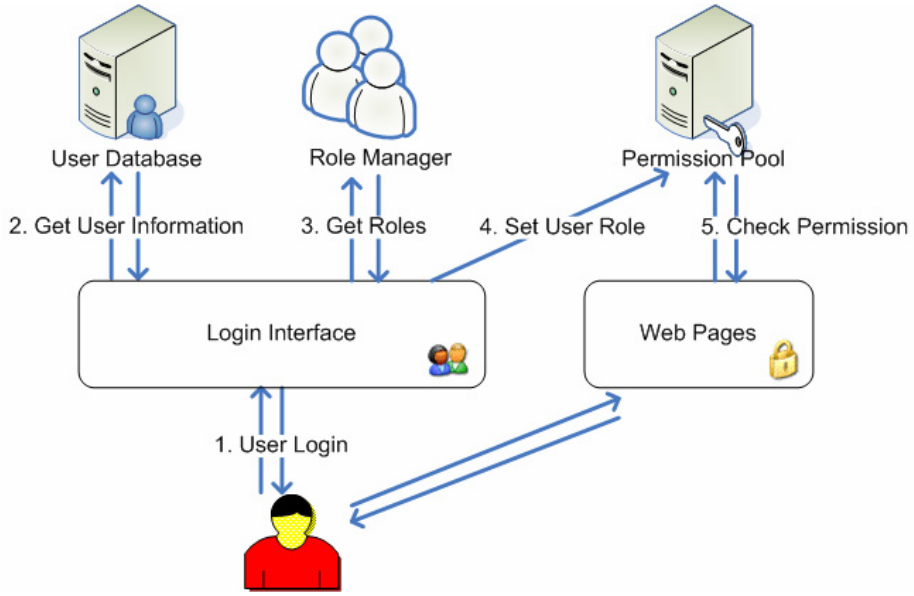

圖 4-10、系統登入流程

使用者登入時，系統會先到資料庫取得使用者相關資訊，接著向角色管理員註冊此使用者。角色管理員的工作則是建立一個空的虛擬角色（User Role），依據使用者所擁有的角色取得相關權限，將使用者所擁有的權限加至 User Role，最後將 User Role 存入權限池（Permission Pool）中。當使用者進入受保護的頁面時，系統會直接到權限池中取得 User Role 物件，並以此判斷使用者是否擁有適當的權限。登入流程關於權限設定部分，在系統中實際的運作方式如圖 4-11 所示。

當使用者執行需要權限檢查的頁面時的流程如圖 4-12，系統會先取得使用者的虛擬角色，以迭代的方式輪流驗證虛擬角色中所包含的角色，如果該角色不具備相關權限時，則會繼續尋訪下一個角色，直到其中一個角色通過驗證，或所有角色皆無相關權限為止。

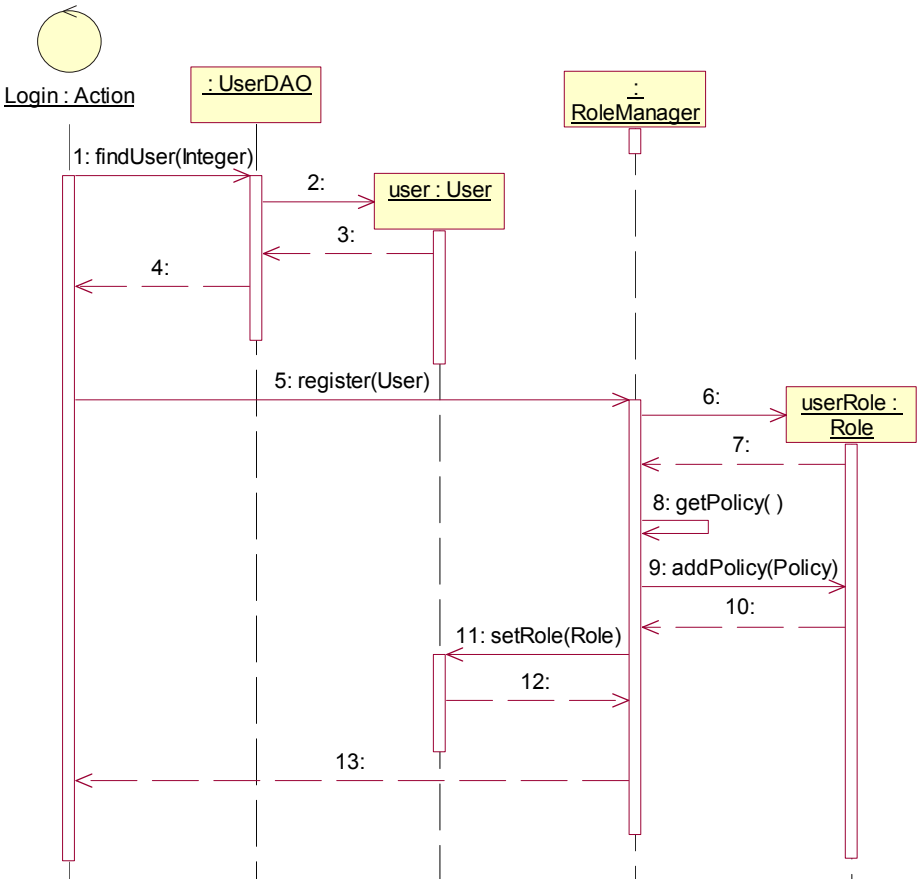

圖 4-11、使用者登入循序圖

角色對於權限驗證的流程同樣以迭代的方式輪流驗證角色中的所有 Policy（可能為 Permission 或另一個角色物件）。而最底層的 Permission 物件的權限驗證則如前所述，系統會先判斷使用者是否具備相關權限，通過第一道驗證之後，還需要檢驗此權限是否能通過此 Permission 的條件限制，才算完成了完整的權限驗證流程。

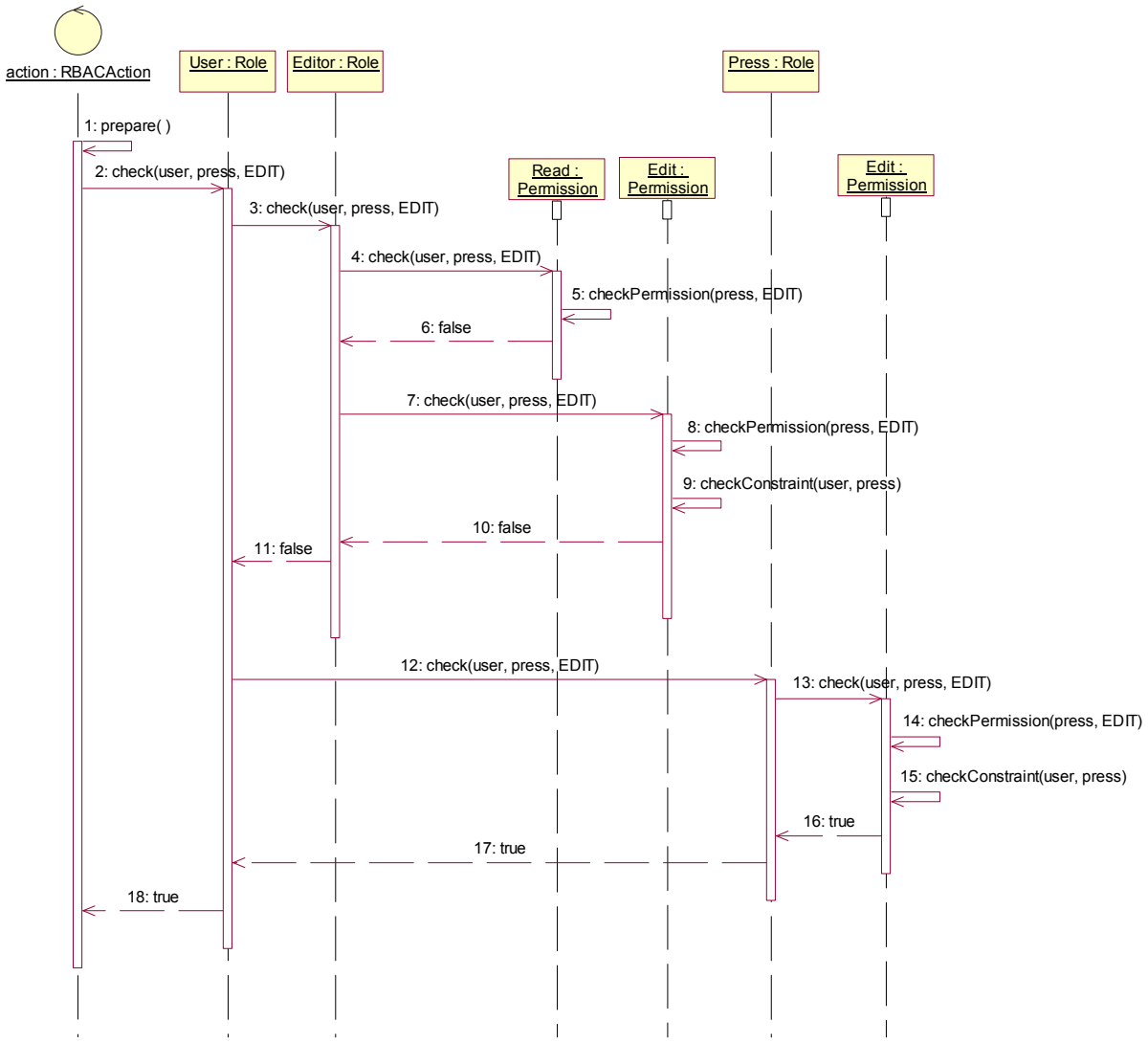

圖 4-12、權限驗證流程循序圖

4.3 新聞管理

4.3.1 個人新聞管理

大部分供稿系統中的稿件是以新聞中心當索引作為分類的依據，當記者的稿件被轉至其他新聞中心後，如果該記者不具備被轉入之新聞中心的記者權限時，就會發生記者無法繼續編採自己的稿件之情形；而沒有特別作權限管理的系統，又不符合平面新聞較為嚴謹的守門程序。

在本研究中，我們採用「個人新聞管理」的觀念來解決此問題。以稿件編採流程為例，除了以新聞中心做分類外，另外獨立出「個人稿件管理」功能來做為記者的工作區域，在個人稿件管理中，記者可以查閱並修改、管理所有由自己所新增的稿件，並且追蹤由自己所提交的稿件目前之狀態為何。因此不論記者的稿件被轉至任何新聞中心，仍然可以在個人新聞管理中繼續進行新聞編採工作，而不會被權限所限制。

要解決個人新聞管理中的權限問題，我們在供稿系統中先定義「記者」的基本角色。此角色包含了記者所需的基本權限，可以閱讀與修改所有線索與稿件，這些基本權限的限制為擁有者必須是自己。而各新聞中心的記者，皆繼承「記者」角色之權限。

4.3.2 自由記者

多數的記者皆屬於單一新聞中心，也僅對單一新聞中心供稿。但編採組織中可能會出現所謂的自由記者，意指該記者並不屬於特定的新聞中心，可同時對多個新聞中心供稿。在供稿系統中，便可藉由 Hierarchical RBAC 的支援，讓自由記者同時繼承多個新聞中心記者的角色權限，來達成此一目的。

4.4 外部供稿系統

在外部供稿方面，我們希望由內部供稿系統所產製出來的新聞內容，能夠方便地傳遞給外界，圖 4-13 說明了不同的外部供稿服務模式：

第一類為供稿系統將新聞內容傳遞給後製平台。我們希望供稿系統要能針對不同需求的後製平台，提供適合的新聞內容。另一方面，後製平台除了跟單一供稿系統取得新聞內容之外，也可能會需要從不同的供稿系統，取得不同題材或範圍的新聞內容。

第二類為供稿系統間的資訊交換。除了可將供稿系統內的新聞傳遞給不同的後製平台外，我們仍希望在不同的供稿系統間也能夠互相交換彼此的新聞內容，以提高新聞內容的重用性。

由於不同的供稿系統以及後製平台，可能使用不同的作業系統及程式語言來實作，要能在這些不同系統與平台間做資訊交換，找出一個共通的標準是非常重要的。本研究中，採用 Web Services 來作為資訊交換的協定；而新聞內容的封裝上，則採用 NewsML 來作為新聞內容的標示語言。

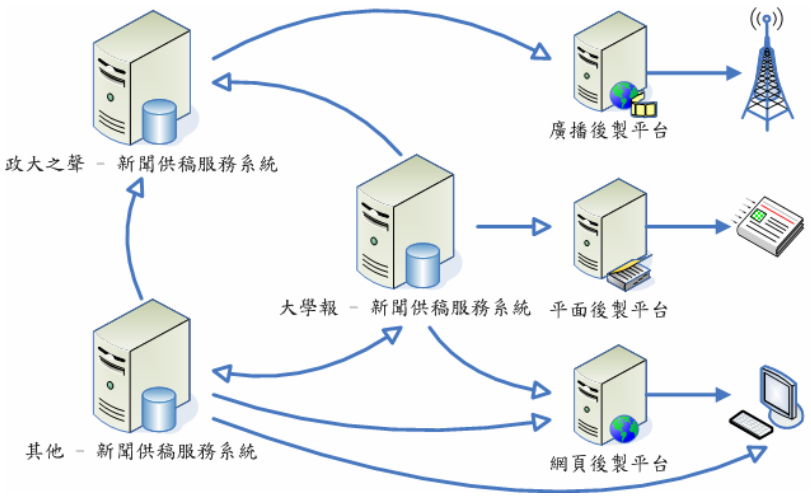

圖 4-13、外部供稿服務概念圖

4.4.1 NewsML

雖然在華人地區尚未有統一的新聞標示語言，但美歐及日本等地的主要通訊社、報社與其他新聞編採組織皆採用 IPTC 所提出的 NewsML 作為新聞內容交換時的標示語言標準。因此在外部供稿系統的部分，我們選擇 NewsML 作為系統發展時的新聞標示語言。

NewsML 的主要目的在於提供一個獨立於新聞類型，以有系統的結構來進行多媒體新聞的交換[44]。在結構上，NewsML 從「內容（ContentItem）」、「結構（NewsComponent）」、「管理（NewsItem）」、「互換及組合」四個層次來描述不同組之間交換新聞內容時的資訊。

NewsML 目前最新的版本為 2.0 版，但本文付梓之際仍在草稿階段。因此以下將採用 IPTC 最後釋出的正式版本 1.2 版作為討論。從圖 4-14 可看出，一份 NewsML 文件的根元素（root node）為「NewsML」元素。此元素必須包含一個 NewsEnvelope 及一到多個 NewsItem。

NewsEnvelope 包含了關於傳遞這份文件的相關資訊，包含發送者、收件者等資訊，這些資訊皆可被省略，唯有傳遞時間（DateAndTime 元素）是必須包含的。NewsItem 則可被視為在某個時間點中，對於一個新聞事件所紀錄的訊息，其中包含了關於新聞的內容與各項管理上的描述。

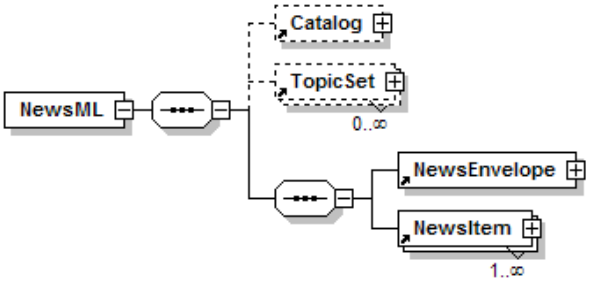

圖 4-14、NewsML 結構圖

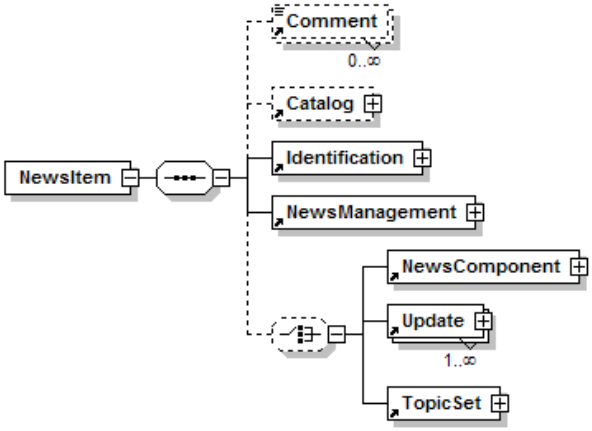

圖 4-15、NewsItem 結構圖

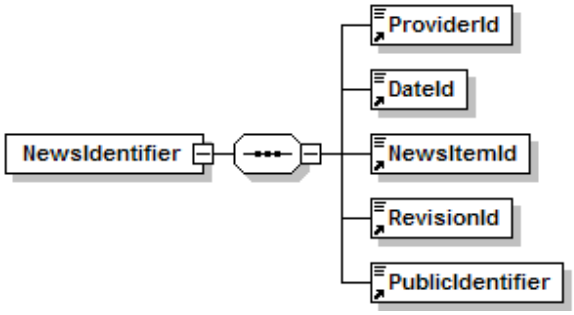

圖 4-16、NewsIdentifier 結構圖

NewsItem 的結構如圖 4-15，可分為識別資訊（Identification）、管理資訊（NewsManagement）以及新聞內容描述三個部分。Identification 包含許多讓外界可以識別此則新聞的新關資訊，其中最重要的是 NewsIdentifier 元素。其所紀錄的資訊如圖 4-16，包含新聞提供者、日期、新聞編號、版本編號以及 PublicIdentifier 元素，PublicIdentifier 代表此 NewsItem 的 URN（Uniform Resource Name）。

NewsManagement 則紀錄有關管理上的相關資訊，包含 NewsItem 型態、第一次被建立的日期、此版本 NewsItem 被建立的日期，以及此 NewsItem 的狀態等資訊。通常 NewsItem 的狀態可依不同編採組織的需求自行定義，在 IPTC 所提供的規格中，可分為可用（Usable）、禁止（Embargoed）、保留（Withheld）與取消（Canceled）四種。

NewsItem 中關於新聞內容的描述，可能為一個 NewsComponent，或是同則 NewsItem 之前版本的更新資訊，與 TopicSet 資訊三者其中之一。

NewsComponent 的主要功能為描述新聞內容的結構，為了說明方便，以下我們將 NewsComponent 元素稱為「新聞元件」。通常一個 NewsItem 元素中，會包含一個主要的新聞元件，以及為了補足此新聞元件的其他次要新聞元件。圖 4-17 為新聞元件之結構圖，我們可將新聞元件視為一種具有遞迴結構容器（Container）。

依據不同情形，新聞元件代表的涵義也有所不同，最上層的新聞元件所代表的是新聞內容間的結構與關係；另外也可能代表著同一種媒材的內容集合，例如同一個畫面不同解析度的影像檔案；或是以相同目的的內容集合，例如提供給廣播平台，或是提供給網頁的新聞內容。最底層的新聞元件，代表的才是真正的新聞內容，此時新聞元件會包含一個 ContentItem 來儲存真正的新聞內容，其結構如圖 4-18 所示。

為了描述不同新聞元件間的關係，除了最上層的新聞元件外，其餘的新聞元件會藉由 Role 元素來描述此元件所代表的涵義。當同一層中存在多個新聞元件時，也透過指定 EquivalentsList 與 Essential 屬性來說明彼此間的關係。EquivalentsList 說明這些新聞元件是否需要同時存在，或是僅由其中一個元件即可解釋完整的新聞內容。Essential 則是說明此新聞元件是否為說明此則新聞事件的必要元素。

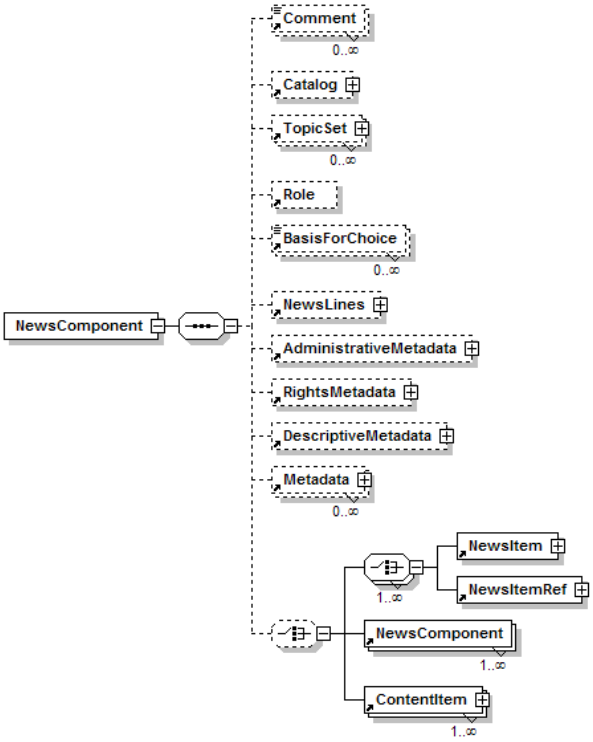

圖 4-17、NewsComponent 結構圖

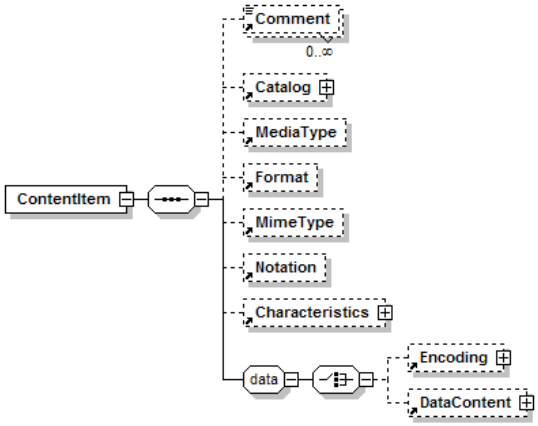

圖 4-18、ContentItem 結構圖

我們除了希望能將供稿系統中的新聞內容傳遞給其他平台之外，也希望能接收外面的新聞資源，來幫助組織本身的新聞編採作業。因此除了需要將我們所設計的物件轉為 NewsML 文件外，也需要有能力將 NewsML 文件轉為供稿系統中物件儲存。

圖 4-19 中之 NewsMLBuilder 類別負責由系統中的新聞物件建立提供外界的 NewsML 文件。我們提供了二種建立 NewsML 文件的模式，傳入單一 NewsItem 物件建立針對單一新聞事件的 NewsML，可在客戶端程式只想瀏覽單一新聞時採用。第二種模式為一次傳入多個 NewsItem 物件，在客戶端依據某個搜尋條件取回多筆新聞內容，例如一次傳回當期所有新聞時，即可採用此模式。

NewsMLParser 類別負責將由外界供稿系統傳遞過來的 NewsML 文件轉為新聞物件。在內部供稿系統中，我們支援了文字、語音及影像三種類型的新聞稿件，因此我們針對這三種類型的稿件，分別設計了不同的 NewsComponent Builder 與 NewsComponent Parser 來處理不同類型的稿件。

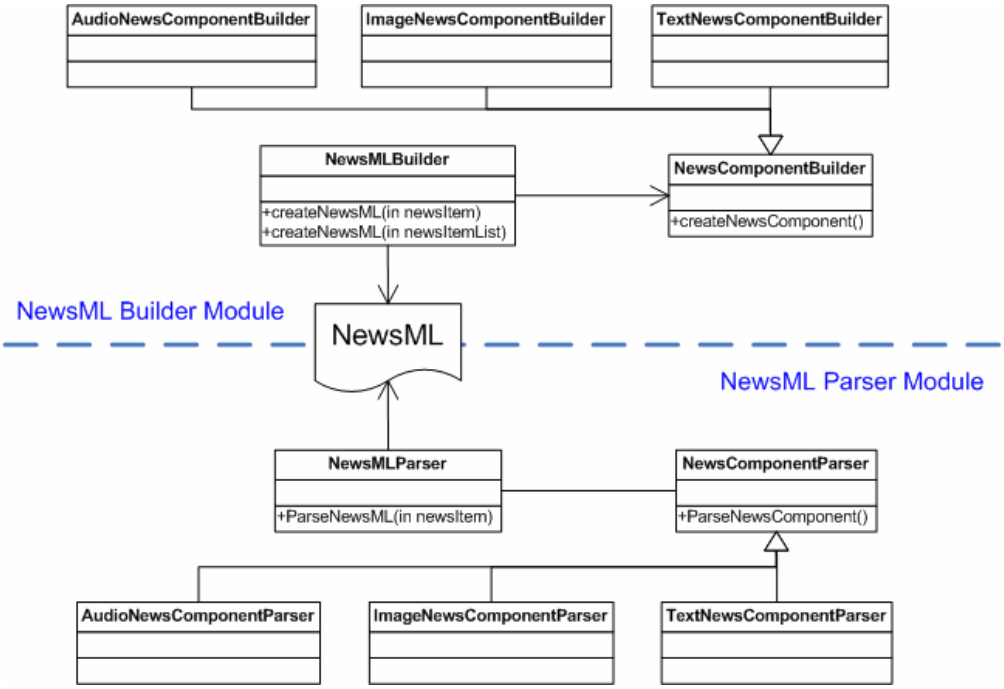

圖 4-19、NewsMLBuilder 與 NewsMLParser 類別圖

由於 NewsML 中的新聞元件為遞迴結構，要正確解析不同類型的新聞元件有其困難度。因此在供稿系統中，我們利用新聞元件中的 Role 元素，來輔助 NewsMLParser 類別判斷新聞元件的型態。

圖 4-20 為一則包含文字、影像與聲音稿件的新聞所產生的 NewsML 文件。除了最上層的新聞元件之外，第二層的新聞元件以此新聞元件所提供的平台做區分。提供給廣播電台使用的新聞元件，Role 為「RADIO」。提供給平面媒體與網站所使用的新聞元件，其 Role 元素內容皆為「WEB」。

供廣播電台使用的新聞元件中，語音檔案使用「AUDIO」。而語音檔案的文字內容部分，則使用「AUDIO CONTENT」；供平面與網站使用的新聞元件可分為兩個部分，第一部分為文字新聞，其產生的新聞元件，Role 元素內容為「MAIN TEXT」。而影像新聞則使用「IMAGE」，其中包含兩個子新聞元件。由圖說所產生的新聞元件使用「LITERARY」，由影像檔產生的新聞元件使用「PHOTO」。

平面新聞與網站新聞的差異在於對影像檔解析度的要求不同，供稿系統也分別針對平面、網站與手機分別產生適合的影像檔。對於不同解析度的影像檔，可由檔案大小來判別。或是藉由新聞元件中，Format 元素之不同屬性值來做識別。

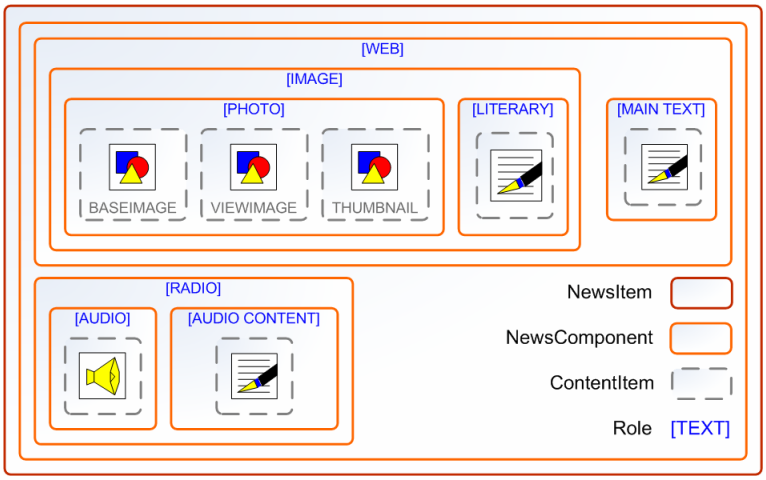

圖 4-20、NewsML 文件圖例

圖 4-21 為供稿系統所產生的 NewsML 範例。範例中之新聞包含兩則不同類型之稿件。Role 元素內容為「MAIN TEXT」者，其內容為此新聞中的文字稿件部分。而 Role 元素內容為「PHOTO」者，其內容為此新聞的影像檔案。我們亦為不同的瀏覽介面提供三種解析度的影像檔案，可由 Format 元素之屬性質來判斷。


```
  <?xml version="1.0" ?>
  <!DOCTYPE NewsML (View Source for full doctype...)>
- <NewsML>
  + <NewsEnvelope>
  - <NewsItem>
    - <Identification>
      - <NewsIdentifier>
          <ProviderId>news.cmits.nccu.edu.tw</ProviderId>
          <DateId>20040624</DateId>
          <NewsItemId>1</NewsItemId>
          <RevisionId PreviousRevision="0" Update="N">1</RevisionId>
          <PublicIdentifier>urn:newsml:news.cmits.nccu.edu.tw:20040624:1:1</PublicIdentifier>
        </NewsIdentifier>
      </Identification>
    + <NewsManagement>
    - <NewsComponent EquivalentsList="yes" Essential="no">
      - <NewsComponent EquivalentsList="no" Essential="no">
          <Role FormalName="WEB" />
        - <NewsComponent Essential="no" EquivalentsList="no">
            <Role FormalName="MAIN TEXT" />
          - <NewsLines>
              <HeadLine>政大校慶特別報導</HeadLine>
              <ByLine>何昱哲</ByLine>
              <CopyrightLine>Copyright (c) 2004 by NCCU</CopyrightLine>
            </NewsLines>
          + <ContentItem>
          </NewsComponent>
        - <NewsComponent EquivalentsList="no" Essential="no">
            <Role FormalName="IMAGE" />
          - <NewsComponent Essential="no" EquivalentsList="no">
              <Role FormalName="LITERARY" />
            + <NewsLines>
            + <ContentItem>
            </NewsComponent>
          - <NewsComponent EquivalentsList="yes" Essential="no">
              <Role FormalName="PHOTO" />
            - <ContentItem Href="http://gemini.cs.nccu.edu.tw:8080/NIS/pressFile/28/P1010011.jpg">
                <MediaType FormalName="Picture" />
                <Format FormalName="BASEIMAGE" />
                <MimeType FormalName="image/jpeg" />
              </ContentItem>
            - <ContentItem Href="http://gemini.cs.nccu.edu.tw:8080/NIS/pressFile/28/m_P1010011.jpg">
                <MediaType FormalName="Picture" />
                <Format FormalName="VIEWIMAGE" />
                <MimeType FormalName="image/jpeg" />
              </ContentItem>
            - <ContentItem Href="http://gemini.cs.nccu.edu.tw:8080/NIS/pressFile/28/s_P1010011.jpg">
                <MediaType FormalName="Picture" />
                <Format FormalName="THUMBNAIL" />
                <MimeType FormalName="image/jpeg" />
              </ContentItem>
            </NewsComponent>
          </NewsComponent>
        </NewsComponent>
      </NewsComponent>
    </NewsItem>
  </NewsML>                                          '>
```


圖 4-21、NewsML 文件範例

4.4.2 Web Services

Web Services 是以 web 的開放標準作為基礎，提供服務給使用者的一項標準。由於其採用 web 的開放標準，所以大多數的平台與程式語言皆能支援 Web Services 標準。因此在外部供稿系統上，我們也以 Web Services 來提供新聞內容的服務。為了說明方便，以下我們將稱 Web Services 為「網路服務」。

由圖 4-22 中可看出，一個 Web Services 服務包含了三個參與者，服務提供者、服務登錄與服務要求者。當服務提供者準備好要提供網路服務時，會先向「服務登錄」註冊所提供給外界的服務，其中包含服務提供者本身的資訊、所提供的服務資訊，以及提供服務的相關技術細節等。服務要求者需要某個網路服務時，會先向「服務登錄」搜尋適合的網路服務，當找到合適的網路服務後，即自行連結服務提供者要求相關網路服務。

圖 4-23 為網路服務的架構圖，整個網路服務的標準，是由數個不同的技術所堆疊而成。在查詢服務上，採用 UDDI 作為服務登錄與搜尋的標準。每項網路服務，也需要有一份對應的 WSDL 來描述該服務的詳細內容。而要求服務時，雖然可採不同網路協定，但通訊的內容則採用 SOAP 包裝。上述三項網路服務的主要標準，皆以 XML 為基礎，因此可被多數平台所支援。

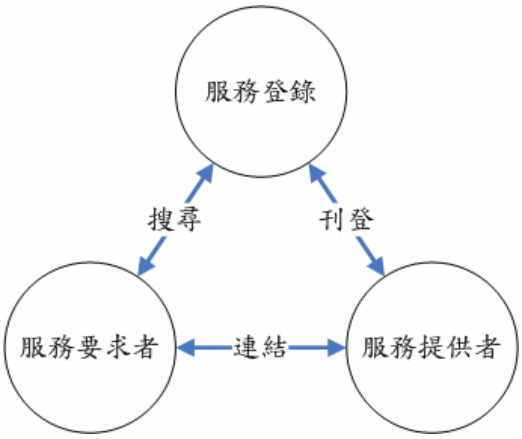

圖 4-22、Web Services 服務導向架構

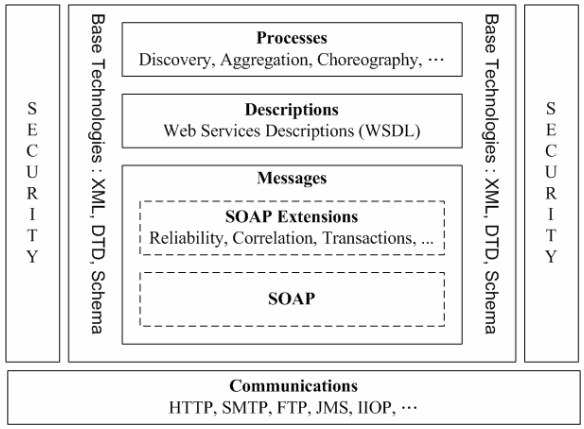

圖 4-23、Web Services 架構圖[29]

劉育嘉將 Web Services 訊息傳遞的流程整理為圖 4-24，Web Services 的訊息傳遞是以 SOAP（Simple Object Access Protocol）作為傳輸的協定[23]。每一筆資訊的傳輸，都是由發送端的 SOAP 處理器根據輸入的資訊以及網路服務的 WSDL 描述檔來產生 SOAP 訊息，傳送給接收端。而接收端根據相同的 WSDL 描述檔來解析出發送端所傳遞的訊息，進行後續的處理。

SOAP 訊息是以 XML 為基礎，由三個主要元素所構成：Envelope、Header 與 Body。Envelope 為 SOAP 訊息的根元素，包含 Header 與 Body 兩個子元素。Header 紀錄需要額外提供給接收端的訊息，當我們有需要擴充 SOAP 的訊息時，就可放在 Header 中。如果沒有額外的訊息時，可不使用 Header 元素。Body 訊息則紀錄了 SOAP 的核心資訊，可能是一個服務的要求、服務提供者的回傳訊息或是錯誤訊息等。

圖 4-25 為外部供稿系統中，客戶端程式要求取得指定報刊的所有期數資訊之 SOAP 訊息。此例中，客戶端程式向外部供稿系統要求呼叫 getPeriodList 方法，此方法包含了一個變數，名稱是 periodicalId，且變數型態為整數。由於此例中並沒有用到需額外提供的資訊，因此沒有包含 Header 元素。

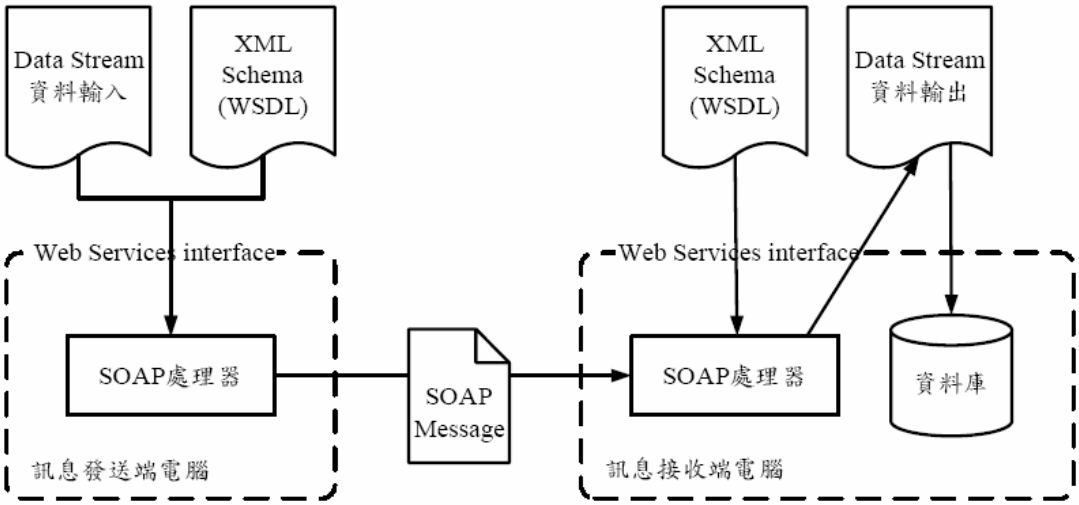

圖 4-24、Web Services Interface 訊息處理流程圖（引用自[23]）。

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <ns1:getPeriodList
        soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:ns1="http://service.nis.nccu.edu.tw">
      <periodicalId xsi:type="xsd:int">1</periodicalId>
    </ns1:getPeriodList>
  </soap:Body>
</soap:Envelope>
```

圖 4-25、SOAP 訊息範例

為了讓服務要求者或其客戶端程式能了解所提供服務的相關細節，例如此服務提供哪些方法、如何與之溝通等資訊，我們採用 WSDL（Web Services Description Language）來為服務提供一個詳盡的描述。

如圖 4-26 所示，WSDL 的六種主要元素依據性質可分為兩類。第一類屬於「服務介面定義」，此類元素的意義在於定義網路服務的介面，讓服務請求者了解此服務所提供的內容，以及相關的訊息型態等，主要有四項元素。Type 元素描述此服務中所使用到的資料型態。Message 元素描述此服務中的各項訊息，包含輸出、輸入與錯誤訊息等。PortType 元素描述的是此服務的介面，定義此服務提供哪些方法可供呼叫。Binding 則描述客戶端程式應該如何繫結此服務，以及應該使用的訊息格式。

第二類則偏向於網路服務在實作上的描述，主要有兩項元素。Service 元素定義 WSDL 文件中所要描述的網路服務集合，包含多個 Port 元素。而 Port 則描述此服務所提供的連結方式。

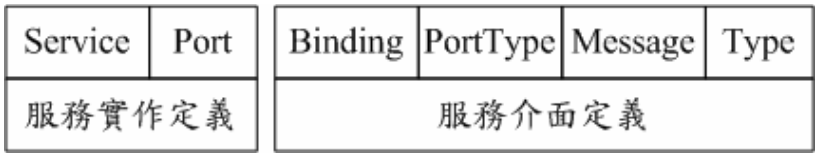

圖 4-26、服務實作定義、服務介面定義和 WSDL 元素[9]

圖 4-27 為外部供稿系統中，提供查詢期數列表的網路服務 WSDL 文件範例。此文件分為服務介面定義與服務實作定義兩部分。

在服務介面定義上，我們首先定義了 ArrayOf_xsd_anyType 與 Period 兩種資料型態，分別表示任何型態的陣列與與期數資訊。接著定義此服務中會用到的訊息，分別是型態為 ArrarOf_xsd_anyType 的 getPeriodListResponse 訊息，以及整數型態的 getPeriodListRequest 訊息。在 PortType 中，我們定義一個可供呼叫的方法 getPeriodList，呼叫此方法時需傳入的參數為 getPeriodListRequest 訊息，而結果則會傳回 getPeriodListResponse 訊息。最後定義繫結此服務的方式 NewsServiceBinding，採用 HTTP 傳輸協定，且此提供 getPeriodList 方法可供叫用。

在服務實作定義上，我們定義了一個 NewsService 服務，實做 NewsServiceBinding 所定義的繫結方式與方法同時指定此服務的連結點位置。

當我們實作好一個網路服務，也提供詳盡的 WSDL 描述檔之後，最後的問題便是如何讓其他使用者知道有這個網路服務的存在？

發展 UDDI（Universal Discovery Description and Integration）之目的，即在於讓網路服務的提供者，可以登入所提供的網路服務資訊，供其他使用者查詢。在 UDDI 登錄的資訊當中，我們可依資料的型態分為三種層次[20]：白頁、黃頁與綠頁。白頁包含提供網路服務的企業相關資訊，如聯絡方式。黃頁藉由支援不同的產業分類方法，可將企業歸於每一產業類別，以方便使用者搜尋之用。綠頁則屬於實作上包含如何使用此服務等細節資訊。

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="http://service.nis.nccu.edu.tw"
    xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:apachesoap="http://xml.apache.org/xml-soap"
    xmlns:impl="http://service.nis.nccu.edu.tw" xmlns:intf="http://service.nis.nccu.edu.tw"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tns1="http://periodical.nis.nccu.edu.tw" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <wsdl:types>
    - <schema targetNamespace="http://service.nis.nccu.edu.tw"
        xmlns="http://www.w3.org/2001/XMLSchema">
        <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      + <complexType name="ArrayOf_xsd_anyType">
      </schema>
    - <schema targetNamespace="http://periodical.nis.nccu.edu.tw"
        xmlns="http://www.w3.org/2001/XMLSchema">
        <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      - <complexType name="Period">
        - <sequence>
            <element name="clueBegin" nillable="true" type="xsd:dateTime" />
            <element name="clueDelay" nillable="true" type="xsd:dateTime" />
            <element name="clueEnd" nillable="true" type="xsd:dateTime" />
            <element name="id" type="xsd:int" />
            <element name="name" nillable="true" type="soapenc:string" />
            <element name="periodicalId" type="xsd:int" />
            <element name="pressBegin" nillable="true" type="xsd:dateTime" />
            <element name="pressDelay" nillable="true" type="xsd:dateTime" />
            <element name="pressEnd" nillable="true" type="xsd:dateTime" />
          </sequence>
        </complexType>
      </schema>
    </wsdl:types>
  - <wsdl:message name="getPeriodListResponse">
      <wsdl:part name="getPeriodListReturn" type="impl:ArrayOf_xsd_anyType" />
    </wsdl:message>
  - <wsdl:message name="getPeriodListRequest">
      <wsdl:part name="periodicalId" type="xsd:int" />
    </wsdl:message>
  - <wsdl:portType name="NewsService">
    - <wsdl:operation name="getPeriodList" parameterOrder="periodicalId">
        <wsdl:input message="impl:getPeriodListRequest" name="getPeriodListRequest" />
        <wsdl:output message="impl:getPeriodListResponse" name="getPeriodListResponse" />
      </wsdl:operation>
    </wsdl:portType>
  - <wsdl:binding name="NewsServiceSoapBinding" type="impl:NewsService">
      <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
    - <wsdl:operation name="getPeriodList">
        <wsdlsoap:operation soapAction="" />
      - <wsdl:input name="getPeriodListRequest">
          <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="http://service.nis.nccu.edu.tw" use="encoded" />
        </wsdl:input>
      - <wsdl:output name="getPeriodListResponse">
          <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="http://service.nis.nccu.edu.tw" use="encoded" />
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
  - <wsdl:service name="NewsService">
    - <wsdl:port binding="impl:NewsServiceSoapBinding" name="NewsService">
        <wsdlsoap:address location="http://localhost:8080/NWS/services/NewsService" />
      </wsdl:port>
    </wsdl:service>
  </wsdl:definitions>
```


圖 4-27、供稿系統 WSDL 範例

UDDI 中另一項相當重要的元素為 tModel。網路服務除了由服務提供者各自提供自己的服務之外，也可能會有同一個組織提供同一類服務的不同實作，例如同一集團中的不同報業組織所提供的新聞服務，背後可能使用同一種技術模型，只是提供的內容不同。或者是藉由產業結合的方式，而有多家企業提供同一類型的服務。tModel 元素即為提供這些背後使用同一種技術模型的網路服務抽象化的功能，只要網路服務中所繫結的是同一個 tModel，即代表我們可由同一種方式來連結這些服務。

藉由 SOAP、WSDL、UDDI 三種標準，得以使網路服務發揮其最大的功能。一個完整的網路服務要求如圖 4-28 所示，會先由服務要求者向 UDDI 探索。尋得適當的服務後，服務要求者會先向服務提供者要求 WSDL 文件，了解服務的相關實作細節。利用 WSDL 文件建立相對應的客戶端程式，正式進行服務的呼叫，完成整個網路服務的流程。

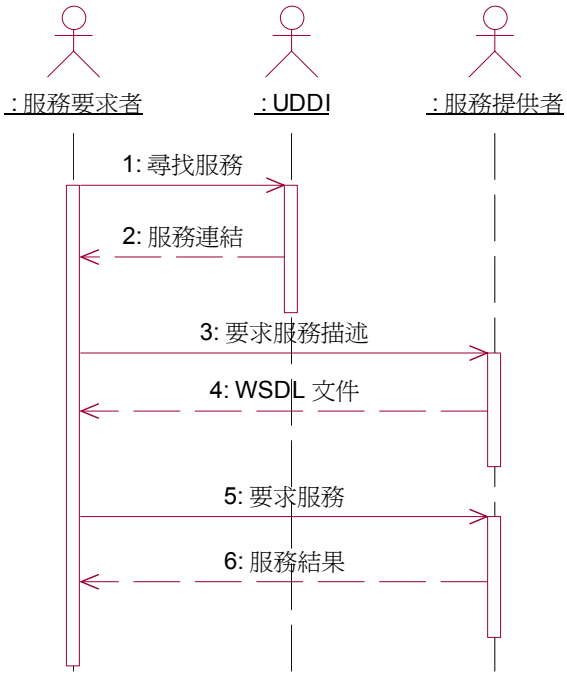

圖 4-28、網路服務要求循序圖

4.4.3 外部供稿系統設計

如圖 4-29 所示，我們分別針對服務提供者與服務要求者兩種不同角色，設計不同模組以滿足外部供稿系統在新聞傳遞上的需求。在服務提供者方面可分為兩個子模組：News Service 與 NewsML Builder。News Service 模組負責處理客戶端程式的要求，目前提供四項基本操作，分別服務要求者查詢此服務所提供的新聞類型、報刊、期數資訊，並可取回某一期的所有新聞內容。當 News Service 依據客戶端程式的要求取出新聞內容後，會先交由 NewsML Builder 模組將新聞內容轉換為 NewsML 文件，再傳遞出去。

在服務要求者方面，同樣依據功能區分為三個子模組：News Finder、News Client 與 NewsML Parser。News Finder 的主要功能為透過 UDDI 尋找網路上有提供新聞內容服務的供應者，而在搜尋時的依據，則是依照對方是否提供預先設定的 tModel 服務。當 News Finder 找到有提供相關服務的供應者後，即交給 News Client 與服務提供者的 News Service 模組溝通，進行新聞取得的步驟。NewsML Parser 則負責將最後所取得的新聞內容，解析為內部供稿系統所能識別的新聞元件，存回本地端的新聞資料庫中。

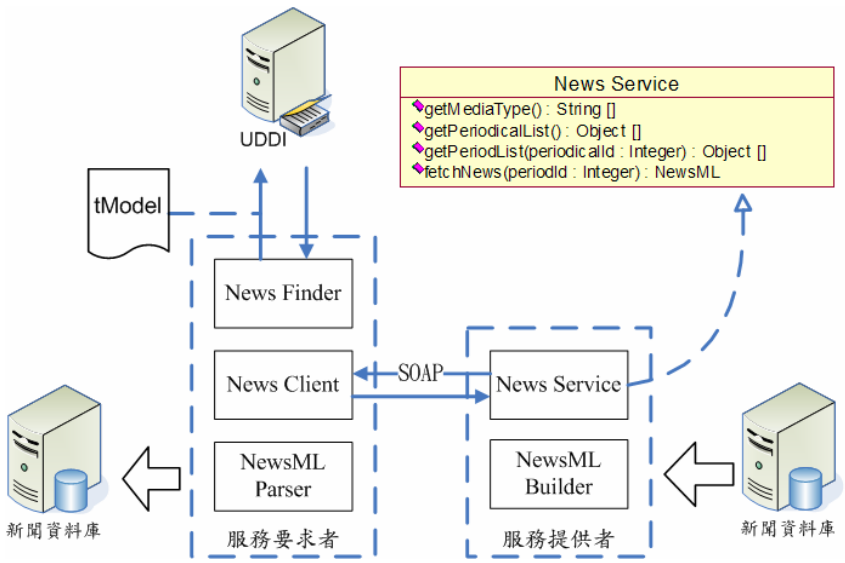

圖 4-29、外部供稿系統架構圖

圖 4-30 為 CAR 子系統向其他供稿系統的外部供稿服務取得新聞內容之流程。使用者可先建立預先篩選條件，如僅接受特定類型之新聞等。News Finder 會先從 UDDI 上搜尋新聞服務提供者，以及其所提供服務之內容。而後由 News Client 主動連向提供者之 News Service 模組詢問相關細節，判斷此服務是否符合篩選的條件，傳回符合使用者篩選條件的服務清單。第二步驟則由 News Client 模組依據使用者所建立之客製化需求，從 News Service 取回新聞內容。

其中 News Finder 與 UDDI，以及 News Client 與 News Service 間皆以 Web Services 方式來進行溝通。傳遞過程中，我們使用 NewsML 格式來封裝新聞內容，服務提供者與要求者分別以 NewsML Builder 與 NewsML Parser 來做新聞內容的封裝與解析處理。

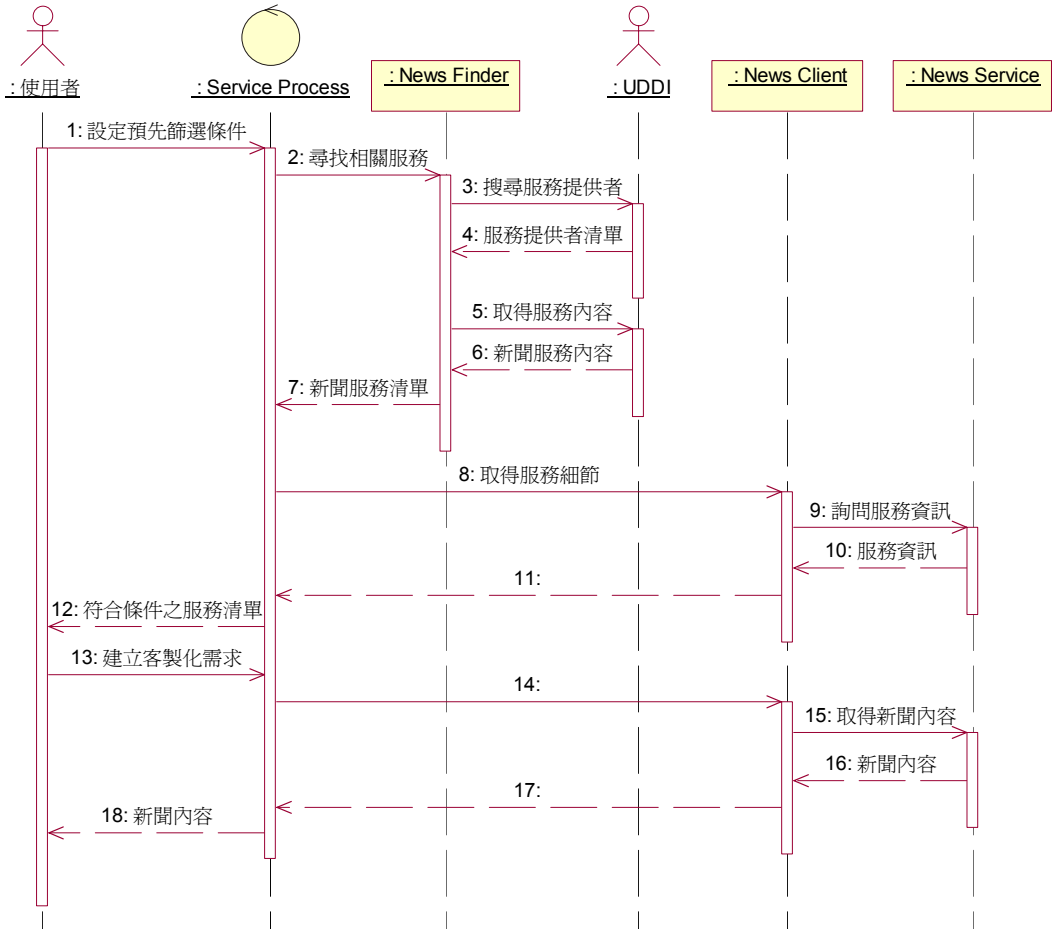

圖 4-30、外部供稿服務流程