國立交通大學

資訊科學系

碩 士 論 文

生物資訊分析自動化流程管理系統

Toward a Distributed Bioinformatics Environment
A Workflow approach

研 究 生：林建宏

指導教授：陳俊穎　教授

中 華 民 國　九 十 三　年 六 月

生物資訊分析自動化流程管理系統

學生：林建宏　　指導教授：陳 俊 穎　博士

國 立 交 通 大 學

資 訊 科 學 系

摘　　要

整合分散且異質的生物資訊資料庫及軟體資源持續的吸引研究人員的投入，而最近幾年的網路服務(Web Service)開啟了生物資訊整合系統的新浪潮。由於網路服務的標準都還在制定中，對於生物資訊複雜的環境能帶來多少裨益還是個未知的問題。在這篇論文中我們以網路服務協調為核心，提出一個生物資訊服務協調平台-BioOrch,去描繪出以網路服務為基礎的整合系統應該要包含哪些重要的設計考量。BioOrch 將流程邏輯中的活動分離出來，讓它可以更簡單且更容易延伸。 此外我們將活動定義為自我滿足的執行單元，使的活動可以在不同的工作流程管理系統之間交換執行。我們展示了這樣的模式可以大大的簡化流程管理系統的設計與實作，並讓不同組織的人可以互相合作。

關鍵字：生物資訊、工作流程、整合式系統、網路服務

Toward a Distributed Bioinformatics Environment – A Workflow approach

Student：Chien-Hung Lin Advisor：Dr. Jing-Ying Chen

Institute of Computer and Information Science

National Chiao Tung University

ABSTRACT

Integration of distributed, heterogeneous biological databases and software resources continues to attract research interests, and the recent Web services movement has also generated a new wave of development of integrated bioinformatics systems. As Web service standards are still evolving, to what degree bioinformatics integration can be achieved using Web services remains a question to be answered. In this thesis, we focus on Web service orchestration aspect, and present a service orchestration framework for bioinformatics, called BioOrch, to illustrate important design considerations when building Web services-based integrated systems. In particular, BioOrch separates flow logic from activities in a workflow, making each activity simpler and more extensible. Furthermore, as activities are self-describing, they can be exchanged among different workflow managers. We show that this model can greatly simplify the design and implementation of cross-organizational workflow systems involving people from multiple sites.

keyword：bioinformatics、workflow、integration system、web service

誌謝

本論文承蒙執導教授 陳俊穎老師不辭辛勞的指導及反覆審視，始得以順利完成。於研究所求學過程中，老師不僅給予我自由的思考空間以及許多的建議與指導，尤其是老師在系統設計與程式撰寫上的豐富經驗與技巧，讓我在短短的兩年之間，可說是獲益匪淺，在此特別表達感謝之意。

同時，也要特別感謝交通大學 胡毓志老師及中正大學資訊工程系劉興民老師兩位口試委員，所提出的寶貴審查意見及建議，使得本論文得以修正疏漏及不足之處。

再者，所要感謝的是CCL實驗室的同學阿吉、訊宏、舜禹於研究過程中所提供的相關協助。在兩年的研究生生涯中，由於有各位同學在學業上、生活上以及各方面的相互扶持，使我得以順利的完成研究所學業。

最後，所要感謝的就是我的父親、母親在求學過程中所給予我的鼎力支持，使我得以免除後顧之憂，全心專注於學業上，僅將此論文獻給我摯愛的家人、親友，願將此榮耀與你們共享。

林建宏 謹識 2004年6月

於交通大學資訊科學所碩士班

目錄索引

圖索引

表索引

第一章 緒論

近年來生物資訊已成為眾所矚目的新興產業。隨著生物資訊多年來的研究，累積了大量的資料與知識。為了要提高研究的效率與降低實驗的成本，許多不同的分析工具先後被發展出來。而隨著網際網路的快速發展，生物科學研究人員可以自由的透過網路存取這些工具或資料，譬如：GenBank[1]、EMBL[2]、DDBJ[3]等公開的資料庫或是GCG[5]，Emboss[4]，SRS[6]等分析程式，舉例來說：單單EBI (European Bioinformatics Institute)[7]一個組織就提供了 50 多種工具和 40 多種資料庫供研究人員下載或線上使用。對於生物科學研究人員來說使用這些網路上的資源能大大的降低研究成本。因此，使用這些網路上的分析工具與資料庫，變成現今生物科學研究一種重要的技術選擇。這些工具和資料庫因為是由不同的組織所發展，所以可能各有其操作的方法、需要的環境、使用的格式等等，而且彼此之間往往又沒有良好的溝通介面。生物科學研究人員在使用這些工具時，常常需要深入的了解各個工具與資料庫，才能在眾多異質的資源之間完成工作。雖然這些資源促進了生物科學的發展，但是使用這些異質的資源也造成生物科學研究人員的困擾。舉例來說：生物科學研究人員可能必須在各個工具與資料庫之間穿梭，並手動的轉換彼此需要的格式、手動的填入資料..等。

為了緊密整合網際網路上的異質資源，許多組織早已開始著手進行相關的研究，如 Web Service、CORBA、J2EE…等。以 Web Service 技術來說，它透過公開標準和溝通協定，使的網際網路上不同組織所提供的服務能互相溝通與合作。使用網路服務這種元件化的技術能讓系統彈性的組裝，使的系統的開發和維護更容易。對生物資訊來說使用網路服務提供了不同平台間溝通與合作的方法，使的異質的資源得以整合。越來越多生物資訊的相關資源都以這種型態出現在網際網路上，譬如：DDBJ、EBI，都有提供這樣的服務。

除了整合異質的資源外，生物科學經常需要與不同領域的人一起合作，像是一些大型的藥廠或學校的研究團體。由於參與者的相關經驗、知識和能力都不相

同，所以合作時往往都要先了解對方領域的相關知識和處理的流程步驟，不但花費時間、人力且暴露太多組織內部的資訊，因此大型醫藥公司和其它生物科學組織必須要考慮如何系統地削減費用、增加效率、提高安全，使其能專注於科學的發展。

為了要使組織之間的流程能讓很容易的為外部系統使用，網路服務的相關研究如 BPEL4WS[8]，WSCI[9]等等，嘗試將組織內部的網路服務以流程的方式整合起來，並將流程以網路服務的方式提供外部系統使用。對使用者來說使用流程就像使用網路服務一樣。如此可以輕易的整合不同組織裡的服務流程，而且也能輕易的隱藏內部流程的資訊。使用流程除了易於跨組織之外，對於組織內部能提供更有效率的監控與管理。在實際流程裡每個人都有負責的工作，一般來說其內容與規範大多是靠制度與人員的訓練，所以流程的進行可能因為人爲失誤而產生錯亂。使用自動化流程的好處就是能把每個人的工作在適當的時候交付，而需要完成的工作則可以有適當的說明引導以利工作的進行。在流程進行中，相關的資料會被記錄下來以達到實驗所需要的完整性，並可透過流程管理介面來管理和監控，如此可以使生物科學研究人員專注於生物科學的研究。

綜合上述，我們認為以工作流程為核心的生物科學研究系統將會是未來發展的趨勢，而這樣的系統應該要能擴充分散且異質的生物資訊資源、支援多個相關人員參與工作流程、能與不同的組織合作。現在的系統不是無法支援上述的需求，就是必需要額外的使用一些其他的系統。另外現在的技術如之前所提的 BPEL4WS、WSC 等等。在整合網路服務上還是有些許的不足，不是對於流程之間的互動並沒有詳細的規範，就是沒有實際定義可執行的流程。

有鑑於此，本論文的目的為發展一個理想的生物研究整合系統，能有效管理分散且異質的生物資訊資源，並透過流程定義可以很輕易的達到跨組織合作的目的。組織之間的合作不再需要冗長的事前準備，只需要了解流程定義需滿足的條件即可。我們希望藉由此系統的輔助，能大量降低生物科學研究的成本、提高研究的效率。在後續的章節中，我們將先對生物資訊的領域及現狀，Web Services 的

整合趨勢，及流程管理的進展作一概括說明， 接著我們說明本論文探討的問題，我們的目標，及提出的方法。其後我們描述實際系統的內部設計及實作，並使用跨組織生物晶片資料分析的流程和 BPEL4WS 的貸款流程為例子介紹系統的用法。最後我們分析探討已知的相關系統作分析比較，並對本論文總結。

第二章 背景介紹

隨著生物科學技術的進步，累積了大量的資料，使的生物科學人員不能再使用傳統的方法儲存與分析這樣龐大的資料。為了解決這個問題，生物科學研究人員開始使用電腦來處理這些資料，透過電腦高速計算能力與龐大儲存空間，資料能被妥善的使用與管理，進而進行適當的分析。由於生物科學對電腦的依賴，使的一門新的學問被獨立出來稱之為“生物資訊”。生物資訊是一門跨領域的科學，結合多種不同的科學如生物學、電腦科學及資訊科技等等。其主要可分為三個方向：

1. 研究新的演算法和統計法，以發現資料彼此之間的關係。
2. 分析並闡釋不同類型的資料，包括核酸與胺基酸序列、蛋白區塊、以及蛋白結構。
3. 發展與補強應用工具，使能對不同形式資訊的取得與管理上，增進工作效率效率。

2.1 生物資訊資源

由於技術的進步，生物資訊產生了許多不同的資料，包含基因的資料、蛋白質的資料、生物晶片的資料等等，以及一些其他的相關資料。如下表所整理，一般常見的資料庫有核酸資料庫、蛋白質資料庫、生物晶片資料庫、文獻資料庫和一些整合型的資料庫。核酸資料庫主要存放許多合序列的資料，蛋白質資料庫主要儲存蛋白質相關的資料如 2D/3D 結構資料、蛋白質功能等，生物晶片資料庫主要存放生物晶片實驗後的結果，包含實驗的目的、實驗的數據等等。文獻資料庫主要可以查詢生物相關的文獻紀錄，而整合型資料庫則是整合多個與多種不同的資料庫，於同一個介面中使用，方便使用者不用在不同的資料庫中往返。

種類	常用資料庫
核酸資料庫	GenBank[1]、EMBL[2]、DDBJ[3]、FlyBase、MGI、INE
蛋白質資料庫	PDB[23]、CATH[24]、SCOP[25]、PIR[26]、TrEMBL、SWISS-PROT、PFAM、PathDB、PDBsurn
生物晶片資料庫	Gene Expression Omnibus [27]、Stanford Microarray Database [28]、ArrayExpress[29]
文獻資料庫	PubMed[30]
整合型資料庫	SRS[6]

表格 1、常用生物資料庫

目前生物資訊資料庫的數量相當多，DBcat[22]是一個很完整的生物資料庫目錄，目前總共整理出 511 個生物資料庫。這些生物資料庫通常都是架構在網路上，並且透過網頁的形式來顯示與搜尋。除了資料搜尋外，生物資訊資料庫通常還會提供一些分析的程式，方便使用者找到需要的資訊。並且有些資料庫彼此之間可能也互有關聯，讓使用者能快速的得到所有相關的資訊。

除了大量的資料與各種資料庫之外，生物資訊還發展出許多不同的工具來分析與處理這些資料，如表所示列出了一些常用的工具列表，總共分為 5 大類，分別是：

1. Homology & Similarity：主要是用來找尋或比較彼此之間的相似程度
2. Protein Function. Analysis：這個類別主要提供各種工具來判斷蛋白質的功能
3. Sequence Analysis：序列分析提供了各種生物的方法來判斷基因和蛋白質的生物功能與結構
4. Structural Analysis：用來預測蛋白質的 2D/3D 結構
5. Tools Miscellaneous：其他有用的生物資訊工具

種類	工具
Homology & Similarity	Blast – Parasite、Blast2 – EVEC、Blast2 – NCBI、Blast2 – WU、Fasta、Fasta - Geno./Proteo.、MPsrch
Protein Function. Analysis	CluSTr Updated、GeneQuiz、InterProScan
Sequence Analysis	Align、ClustalW Updated、GeneWise、PromoterWise
Structural Analysis	DALI、Maxsprout、MSD Services、MSDfold
Tools Miscellaneous	EMBL Computational Services、Expression Profiler、NEWT、QuickGO、Readseq

表格 2、常用工具列表[31]

由於生物資訊的內容豐富，衍生出許多不同的工具來解決各種問題。大量的工具對使用者來說雖然可以解決所遇到的問題，但卻提高了使用者的使用門檻。除了要去學習各種不同工具的使用方式外，工具彼此之間如果沒有良好的介面也會造成使用者的負擔。

2.2 生物資訊的整合問題

生物資訊的研究單位遍佈全球，雖然有一些重要的組織如 NCBI、EMBL、DDBJ 會進行資料的整合和交換，但仍然沒有一個專門的組織如 W3C 一樣來統一生物資訊資源的格式和標準。所以造成了生物研究組織各自為政的情形，各個組織依自己的需求發展適合的工具與標準。雖然保持多樣性是適合生物科學等新興科學的發展方式，但是卻造成了生物資訊資源的分散性、異質性。

就分散性而言，舉例來說，生物科技研究人員若希望能找出某段基因的資訊，可能會透過 Blast[34]找出與基因序列相似的其他基因序列。利用這些基因序列的 gi number 去 Entrez[35]去找出完整的序列資料。從這些資料中找此序列相關的 protein，再用 protein 去找組成這個 protein 的 domain。在此過程中，可能需使用至少三種不同功能的工具，或者是造訪三個不同種類的生物資訊資料庫來進行處理。處理過程當中，面對不同的生物資訊資源，使用不同的操作方式，並不是

非常方便。雖然生物資訊資源經常都以網頁的形式呈現可提供較友善的使用者介面，但在不同功能用途的生物資訊資源之間傳遞序列資料、gi number 等資料仍有可能因操作不當而發生錯誤。除了生物資源的分散外，跨組織的合作也是生物資訊的一大考驗。生物資訊實驗經常需要與不同領域的人一起合作，來解決生物資訊豐富的內容，像是一些藥廠或一些學校團體，就經常互相合作，由於使用者的相關知識，經驗和能力都不相同，所以合作時往往要多花時間了解不同領域的知識，並且要和不同團體的人緊密聯繫以協調流程的進行。因而降低了實驗的效率而且同樣可能因為流程控制不當導致實驗的錯誤。

除了分散性，各個生物資訊資源所使用的資料格式不一致所造成的異質性問題，也是生物科技研究人員的一大隱憂。舉例來說，生物科技研究人員透過網際網路取得生物資訊資料庫上的資料，由於提供來源的資料庫所採用的資料格式可能跟分析工具的格式不一致(以 NCBI 為例，一筆核酸的資料就有可能 13 種不同格式的表示方式)使得生物科技研究人員需先將取回的資料加以轉換後再予以分析，造成處理步驟的繁雜，同時也降低分析處理的效率。

如之前所述，生物資訊資料庫已有超過 500 多個。EBI 就包含了 50 種工具和 40 種資料庫。在此同時，各式各樣的生物資訊資源仍在大幅的成長中，儘管有某些相關研究機構嘗試對這些資源進行整理與分類，但要如何從這麼龐大的生物資源中，找到自己所需要的資源，對生物科學研究人員來說也是一大挑戰。綜合上述，目前生物資訊的主要問題可歸納為以下幾點：

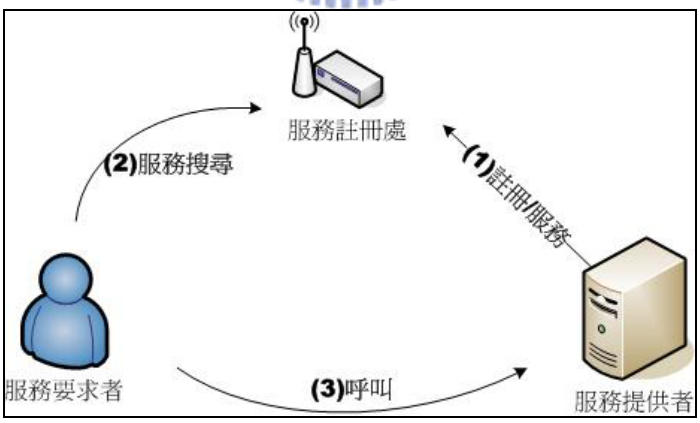
1. 由於生物資訊資源的異質性與分散性，使的使用者在研究的過程中分析步驟繁雜且效率不佳。這個問題主要是整合不易，我們可以嘗試使用一些分散式的技術來達到整合的目標。如 Web Service、CORBA …等等。
2. 生物資訊資源的種類繁多，使的生物科學研究人員在尋找與使用上會產生負擔。所以我們可以使用工作流程的概念，將一些可自動化的工作交由電腦處理，在需要的時候再由電腦通知該執行的工作和相關資訊如使用的工具、使用網路服務..等等。

3. 生物資訊常需要跨組織的合作與交流，有時實驗流程在進行時需要緊密的聯繫以達到流程的控制與管理，但卻耗費太多人力與資源造成效率的降低。從另一方面來說，流程管理可能因人為的因素而產生混亂或無法追蹤。如上所述，工作流程可以由電腦自動化管理減少人為的失誤，並且加上跨組織的合作機制，以達到生物資訊的特殊需求。

2.3 網路服務的興起

為了克服異質性與分散性的問題，許多研究都在努力嘗試解決這個問題，其中以 Web Service[12]最受到注目，許多企業已經使用 Web Service 這樣的技術將商業服務元件化，而許多生物資訊資源如 EBI 的 SoapLab[10] ， DDBJ， PDBJ 等等，也使用 Web Service 這種技術提供免費的服務。透過 Web Service 可以很有彈性的調整組織內部的工作流程，也可以輕易建構和外部系統間的橋樑。

服務導向架構（SOA）是一種由 W3C 所提出來的應用開發的概念，主要是讓整個系統能用一些可重複使用的服務元件組合起來，以降低企業的成本，並讓開發人員能很容易的擴充新的功能，以因應實際的需求。

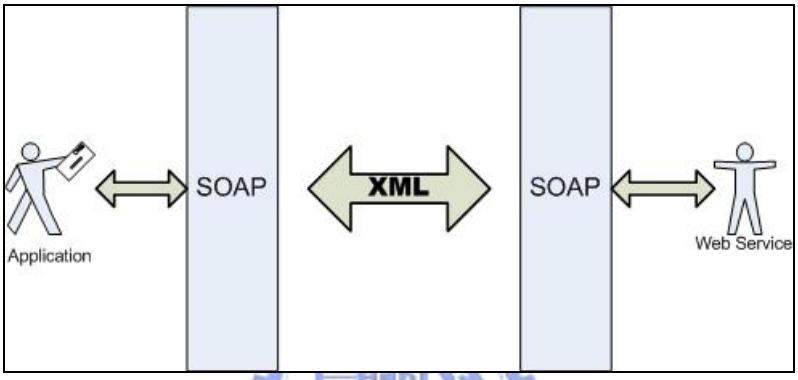


圖表 1、SOA 架構圖

Web Service 便是在 SOA 的基本架構下發展出來的技術，使用 Web Service 的公司或組織能輕易的將組織流程、功能與資料包裝成網路上的服務，供其他組織或個人存取。由於使用了完全標準化的架構，讓過去使用不同系統而難以互動

的情況有了改變。在 Web Services 架構下，能為組織與組織、組織與個人間的互動帶來最直接的效益，並促使生物科學能以更動態的方式進行。標準的 Web Services 包含了下列功能及相關標準：

- SOAP：SOAP(Simple Object Access Protocol)是 Web Service 底層用來傳遞消息的協定，他透過網際網路的標準協定如 HTTP， FTP， SMTP..等，傳遞用 Xml 表示的訊息，如此可以將訊息有組織的傳送到另一端。而接收端可以依照 Xml 的內容將訊息還原。

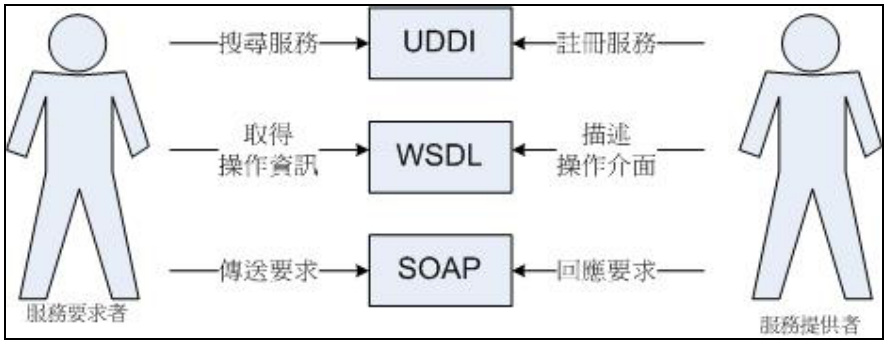


圖表 2、soap 示意圖

- WSDL：Web Service 使用 WSDL 來描述其所提供的服務與支援的介面，及其實體的位置。
- UDDI：UDDI (Universal Description， Discovery and Integration) 的用途，是能將提供者所提供的網路服務相關資訊紀錄下來，並提供服務要求者以電子化的方式搜尋適合的網路服務。

由於 Web Service 是依據 SOA 的架構下發展的，所以也有與 SOA 相似的使用步驟，如下所示：

1. 服務提供者透過 UDDI 的 API，將其提供的服務註冊，以供他人查詢。
2. 服務要求者透過 UDDI API 尋找適合的 Web Service，並取得其 WSDL 的描述。
3. 透過 WSDL 之描述，與服務提供者建立 SOAP 連線，並執行需要的服務。



圖表 3、Web Service 輪廓圖

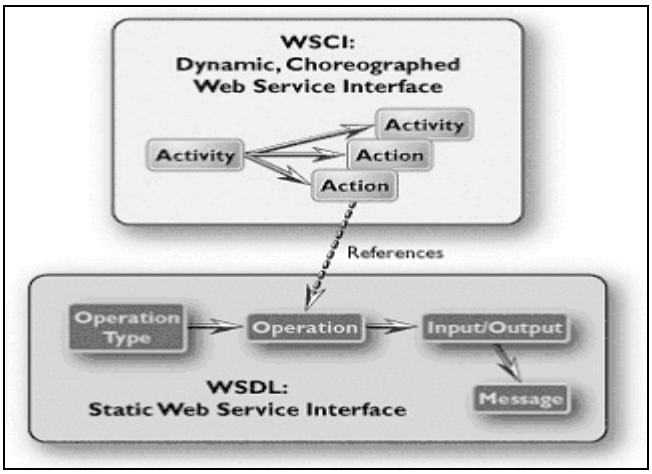
由上圖我們可以清楚的了解，SOAP、WSDL、UDDI 與服務提供者和服務要求者之間的關係。SOAP、WSDL、UDDI 是目前 Web Service 公認的標準。使用 SOAP 協定來處理底層傳輸與訊息的序列化、反序列化，讓應用程式只需要處理高階的訊息內容即可。當底層的傳輸改變時並不影響應用程式處理訊息的內容，讓不同平台的系統之間能更為容易的溝通與合作。而使用 WSDL 與 UDDI，使的 Web Service 有了低耦合的優點，只要支援相同的 WSDL 即可互相合作。

2.4 網路服務協調

許多企業或組織將其服務以網路服務呈現，以提高企業內部架構的彈性。但是客戶在使用這些單一的網路服務時，通常都會伴隨著一些使用流程。使的客戶在使用上產生不便，也使的企業在提供網路服務時變的複雜且難以追蹤，所以如何有效的使用這些個別的網路服務以提供客戶或合作對象更有效率的使用方法，對內能更有效的管理與追蹤，變成企業下一步思考的問題。最近許多組織正在積極制定有關網路服務協調的標準，如 BPEL4WS、WSCI 等。

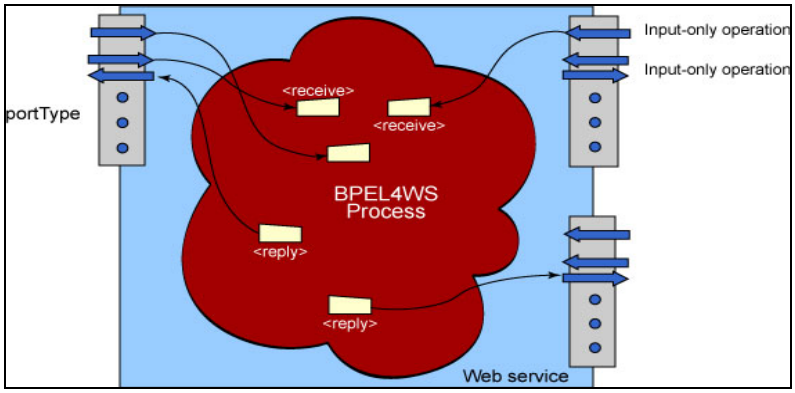
網路服務協調分為兩大類：一種是中央編排式，另一種是可執行流程式。中央編排式的網路服務協調，主要是專注於將訊息在不同的網路服務之間傳遞，以達到網路服務協調的目的。以 WSCI 為例，他會對 Service 定義一個介面，這個介面描述了 Service 所提供 Operation 的使用順序，包含一些 Activity 與 Action。每個 Activity 可能包含一個或多個 Action，每個 Action 對應 WSDL 的一個 Operator 如下圖所示，在 WSCI 中 Global Model 可以將原本介面裡定義的 Action 連接起

來，完成網路服務之間的協調。但是 WSCI 並沒有定義實際可執行的流程而只是定義了使用網路服務的流程，使的實際在執行流程時還需要額外的幫助，如使用 BPML 來當作實際執行的語言。



圖表 4 WSCI 示意圖[9]

另一種為可執行流程式的網路服務協調，主要是專注於建立可執行的流程。以 BPEL4WS 為例，其需要定義一個流程，這個流程主要包含 receive、invoke、reply。一開始由 receive 接收客戶端的訊息起始流程，透過 invoke 去呼叫 pattern 的網路服務，最後由 reply 回傳客戶端資訊。



圖表 5、BPEL4WS 概念圖[8]

BPEL4WS 將流程當成是網路服務一樣，並透過 WSDL 來描述，所以對使用者來說就像使用一般的網路服務。不同於 WSCI 有 Global Model 可以整合各個 WSCI 介面，在 BPEL4WS 則可以透過定義 abstract process 來結合各個 BPEL4WS，只

需要將各個 BPEL4WS 當成是一般的網路服務 invoke。由於這種使用方式使的流程之間的互動並沒詳細的規範，不像 WSCI 將網路服務的使用方式也定義清楚。所以流程在整合時需要特別注意彼此的互動使的流程之間能互相合作。

2.5 總結

在生物研究組織中成本的降低與效率的提高是必須的，所以除了要整合生物環境中異質的生物資訊資源，還有能有效率的使用它們。對組織外部來說，要能輕易的提供流程的服務。對於組織內部來說，要能輕易的掌握流程的狀態。而對於實驗的參與者來說，要能協助其完成工作。綜合上述，我們將目前生物資訊整合的方向可歸納為以下幾點：

1. 使用網路服務來整合異質的生物資訊資源
2. 以流程形式支援跨組織的合作
3. 內部使用流程自動化的機制來監控與管理，並協助實驗的參與者完成工作。

我們的系統便是以此為核心，發展適合生物資訊的研究整合環境。以解決生物科學研究人員所遇到的瓶頸。

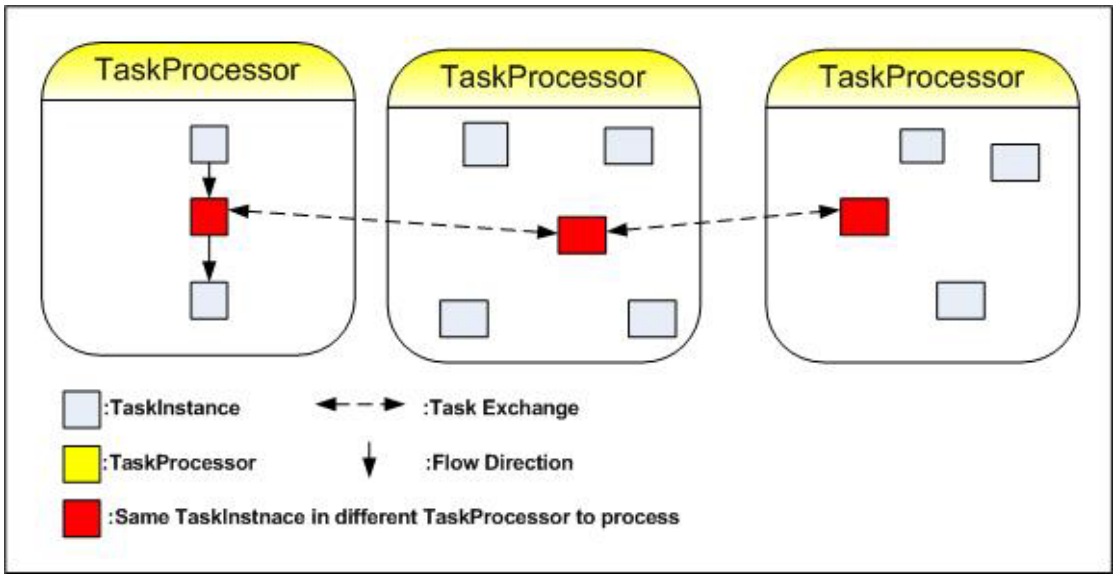
第三章 問題、目標與方法

我們希望發展一個以工作導向為流程核心的生物資訊系統，整合使用者與網路服務為流程的參與者，並提供不同組織的流程可以彼此合作與資源分享，且很容易的擴充新的資源。更進一步說明， 此系統必須以自動化流程為核心、能擴充分散且異質的生物資訊資源、支援多個相關人員參與工作流程、與不同的組織合作等。然而現在的流程管理系統並不完全符合生物資訊的需求[16] [19]。譬如之前所提的 Web Services 標準技術如 BPEL4WS、WSCI 等等， 在整合網路服務上還是有些許的不足[17] [18]，不是對於流程之間的互動並沒有詳細的規範，就是沒有實際定義可執行的流程。由於在 Web Services 上建立整合型生物資訊系統將成為未來幾年的趨勢，在本篇論文中我們嘗試發展一個以工作導向為流程核心的生物資訊系統 - BioOrch。 為了滿足生物資訊系統複雜多變的需求， BioOrch 的設計包含了下列的特色：彈性的整合方式、工作導向、跨組織的合作、組織人員參與等。

BioOrch 能以抽象的形式整合不同的網路服務，只需要透過抽象概念裡定義好的介面去存取網路服務，而不需要知道網路服務的執行細節或使用哪一個網路服務，降低了系統與網路服務之間的耦合程度。除了抽象形式的整合外，BioOrch 還提供了動態整合的方式，包含執行與發佈時。執行時，BioOrch 可以透過抽象的概念使用網路服務，而這些抽象概念與網路的實體資訊都能透過一個負責註冊的網路服務得到，所以系統在執行時能動態的到該服務找尋適合的網路服務。而在發布時的動態整合，是透過管理者對系統作設定使其知道某些服務的位置。對於系統作設定有兩種使用情況：(1)使用名稱與網路服務位置的連結，在設計時只需指定名稱即可，流程引擎會依照名稱使用相對應的網路服務。(2)指定抽象概念的網路服務，BioOrch 會用抽象概念來描述網路服務，並於執行時去尋找該網路服務，當系統已經設定抽象概念所要使用的網路服務時，流程引擎將直接使用設定的資訊。除了整合的方式外，為了要彈性的使用網路服務，BioOrch 提供

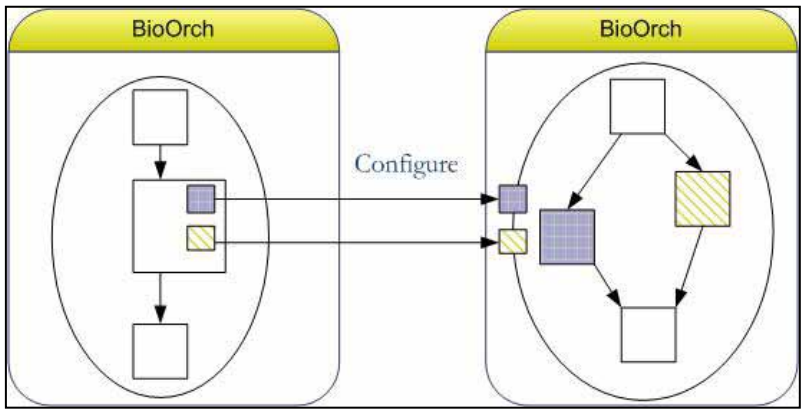
了許多控制流程的方式包括：Fork、Join、Parallel、While、Switch 等，還支援 Label 和 Link 的使用方式，讓生物科學研究人員能彈性的定義其需要的整合方式。

BioOrch 的流程(Process)是由一些工作(Task)所組合起來的，不同於 Activity[13]，工作(Task)並不定義流程進行的順序，在 BioOrch 中另外有控制單元來控制流程的進行。工作也不像是 Action[13]，他還包含了工作的來源，工作的目的，與工作的內容，使的工作變成是一個可以移交的單位。如圖 6 所示，對 BioOrch 來說，核心流程的驅動都是透過工作的分配與接收，由於 BioOrch 的工作可以知道自己是屬於哪一個流程，所以流程引擎可以找到其流程實體並繼續分配新的工作給負責的工作處理單元(Task Processor)。流程處理單元會視工作的內容執行。執行時可能是呼叫其他的網路服務或是交由另一個工作處理單元執行。



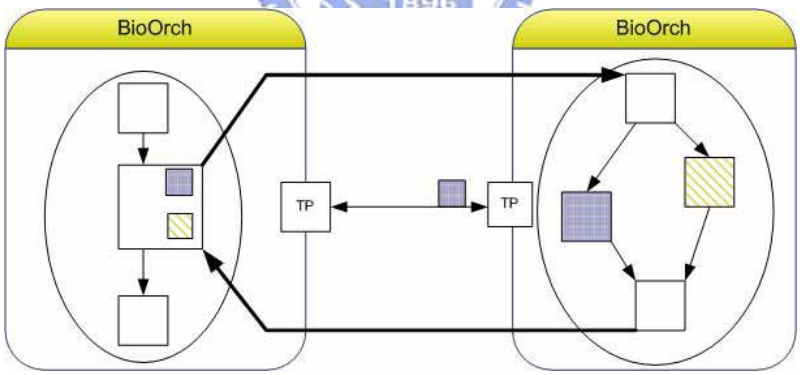
圖表 6、Task Processor 示意圖

在跨組織的合作中流程被視為是一種網路服務，也可以用抽象的形式來表達並註冊。所以在組織之間互相使用對方的流程時，要求者的一方可以像是使用一般網路服務一樣去呼叫，執行完畢後再回傳執行結果。和一般技術不同的是每個流程在使用之前都必須要填入相關資訊，而有一些資訊是用來協調彼此之間的合作。



圖表 7、流程設定示意圖

如圖 7 所示，流程提供者可能需要或者是選擇性的讓使用者能定義部分的工作要如何執行，以利流程之間的合作。例如，在外包的工作流程中可能要使用到要求者內部的網路服務，或者需要要求者內部的人員來執行，這時要求者的流程就會填入相關的資訊，如服務的位置、人員的資訊等等。而工作處理單元在收到這樣的工作時就會將他傳回到要求者的工作處理單元，處理完後在回傳到原本的工作處理單元以繼續未完的流程。



圖表 8、BioOrch 跨組織合作示意圖

我們使用了這樣的方法，定義流程之間合作的機制，有別於 BPEL4WS 的方法，我們較接近於 WSCI 將服務之間的操作連結起來，以工作來定義連結的網路服務，但又不是以 WSCI 中的“Global Model”形式，而是以“abstract process”的方式，延續了可執行的優點。

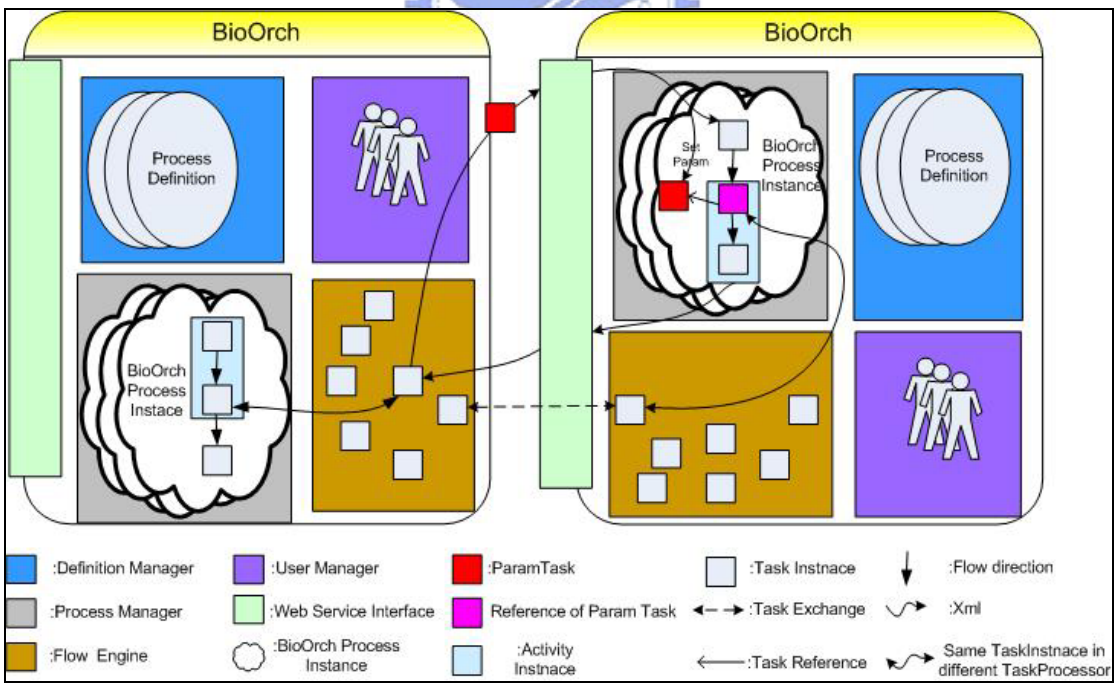
對於生物研究組織來說許多流程除了需要自動化的呼叫網路服務外，許多部份還是需要組織人員的參與，在 BioOrch 的中，包含組織人員管理的部分並交給一個網路服務負責管理稱為 User Manager，這個網路服務是屬於 BioOrch 內部的網路服務，而且也是一個工作處理單元。組織內的人員可以透過 User Manager 得到現在有哪些工作是他所需要執行的，當他執行完成時再透過 User Manager 將工作回傳到流程之中。如此流程除了網路服務外還有許多研究人員可以參與其中，達到生物資訊需要的特性。

第四章 系統介紹

BioOrch 是一個架構在 Web Service 上結合流程管理的生物資訊整合系統。除了使用 Web Service 的技術來整合異質與分散的資源外，主要是提供自動化流程管理來提高生物科學實驗的效率，降低生物科學研究人員參與實驗的門檻，提供工作相關資源的整合，減少人為因素的錯亂。在 BioOrch 中所有可用的服務都必須使用 Web Service 當介面。這包括 BioOrch 本身。它提供一組流程定義語言，能將各別的 Web Service 串聯起來提供流程服務。BioOrch 將流程以網路服務的形式提供外部系統使用。每個 BioOrch 都有自己的人員管理，能將組織內的人員整合進入流程之中，當作執行工作的單位。

4.1 BioOrch 的架構

在我們的系統裡包含了幾個部分，如圖表 9 所示，包括了流程定義與流程、定義管理者、流程管理者、工作與工作處理者、流程引擎、人員管理等等。



圖表 9、BioOrch 架構圖

流程定義紀錄了流程執行的相關資訊。如流程需要的輸入、流程包含的工作、流程進行的順序..等等。而流程則會依照流程定義的內容執行，並且在執行時會紀錄流程的相關資訊，如流程的識別碼，流程的狀態、流程所包含的屬性等等。除了紀錄之外，流程也會負責產生下一個要執行的工作交由流程引擎處理。

所有的流程定義都會交由定義管理者管理，並提供一些管理的介面，包括新增，修改與刪除的功能。在上傳新的流程定義時，管理者必須要先檢查定義的內容是否合法，才允許其上傳。已上傳到系統的流程定義即可被使用者使用。而關於流程的管理都是交由流程管理者負責。當流程的實體建立後，流程管理者就必須一直維護流程的狀態。即使流程結束了，也要將流程執行的過程紀錄下來，透過流程管理者我們可以查詢流程目前的狀態或工作執行完的結果，也可以暫停和恢復一個正在執行的流程或直接將流程刪除。

在流程中包含了許多的工作，工作是 BioOrch 中處理的最小單位，也是一個自我滿足的單元，包含了工作的來源，工作所屬的流程與工作的內容..等等，讓工作可以交由不同的工作處理者處理。工作處理者負責協助流程的工作執行，並將某些工作依照其定義的內容轉交給其他工作處理者執行。如之前提到的流程規定了工作執行的順序，而每個工作為自我滿足的單元，所以使用工作處理者來執行工作，使的 Task 的狀態能簡單的得到控制，並降低流程和工作之間的耦合，工作處理者是系統核心的部份，而且並不只限定只能執行某些特定的工作，而是只要是符合規定的工作都可以執行，這樣的訴求跟我們稍後會提到的可延伸性有關。

在整個系統中最主要的工作處理者便是流程引擎，流程引擎是以工作驅動(Task Driven)來運行的。流程管理者將流程交付流程引擎執行後，流程會產生工作交付流程引擎處理，在工作完成後流程引擎會從回傳的工作找到其所屬的流程，並產生下一個要執行的工作繼續執行。另一個主要的工作處理者是人員管理者(User Manager)，其負責管理使用者資訊並紀錄組織內部人員的工作列表。本身也是一個工作處理單元，會將使用者完成的工作回傳。使的組織內部包含許多

的人員，可以加入流程中當成是工作處理的單位。一個參與者可能是一個人或是一個群組。

4.2 流程定義語言

我們定義的流程包含一組輸入與輸出，並使用 Xml 的格式。當新增流程時，只要輸入符合格式可開始執行流程，並於結束時傳回流程的結果。流程裡包含的工作(Task)也是以相同的概念定義，同樣包含一組輸入與輸出。我們將工作(Task)定義為自我滿足的執行單元，也就是說 Task 包含了一切執行時需要的資訊，流程定義包含一個名字用來當作定義的識別碼，其下包含“description”用來描述流程定義的內容。body 則是整個流程的主體，紀錄了流程的輸入輸出、所有的工

```
<processdef name="bio/microsrray/analysis">
      < description>this is description for bio/microarray/analysis</description>
        <in><say words="@words"/></in>
        <out><hello words="@words"/></out>
      <body>…</body>
</porcessdef>
```

作和執行的先後順序。流程的輸入輸出是用 Xml 的格式定義，並可以指定變數來記錄重要的部份。一旦變數被指定過，在整個流程運行間都可以用變數來代表紀錄的值，如上圖所示：當輸入是<say words=“world”/>時，輸出就會是<hello words=“world”/>，因為在輸入時會將“words”這個變數的值設為“world”。除了 attribute 可以指定變數外，Xml 也可以用變數代替，其使用的方式是在輸入的 pattern 使用<rw:var name=“變數”/>這樣的格式來連結變數與 Xml。當流程開始執行時，會依照 Body 裡工作的定義順序來執行除非遇到特殊的控制單元。所以一般來說 body 裡的第一個工作就是起始的工作。

4.2.1 工作定義(Task Definition)與工作(Task)

工作定義紀錄了工作執行需要的相關資訊，如工作的輸入輸出、工作要交由誰去執行，工作需要的參數等。輸入輸出的格式主要是以執行對象的需要制定並以 Xml 來表示，使用變數來代表重要的值，以利格式的轉換，如之前在說明流

程定義的輸入輸出時一般。除了輸入與輸出的資訊外，工作定義還可以紀錄網路服務的資訊，我們可以只指定<service>的 Type，也就是網路服務的抽象名稱，讓系統幫我們透過註冊者尋找符合這種 Type 的網路服務，也可以直接指定此 Service 的位置。如果此工作是要交給一個使用者去執行的話，我們可以使用<user>指定是哪一個使用者或群組要來執行這項工作。當需要起始另一個流程時，我們可以使用<process>來定義相關的資訊如流程定義的名稱、流程輸入的參數等等。

```
<taskdef name="ccl/bioorch/taskdef/serviceTask" type="default">
   <in><say words="@words"/></in>
   <out><hello words="@words"/></out>
   <service type="ccl\test\echo">
         <host>localhost</host>
         <port>1119</port>
         <path>echo</path>
   </service>
   <user name="eros"/>
   <process name="eros">
      <paramTask def="ccl/bioorch/taskdef/serviceTask" name="task1">
            <user name="eros"/>
      </paramTask>
   </process>
</taskdef>
```

工作為自我滿足的單元，也是系統中的最小單位，工作引用一個工作定義，並可以加入或改寫需要的部分，如輸入輸出等。工作紀錄了工作的來源，工作的目的，與工作的內容，使的工作變成是一個可以移交的單位。工作會交由 Task Processor 處理，而 Task Processor 會依據引用工作定義的 Type 來決定如何處理，使用者可以發展自己的 Type 以符合實際的需求。在跨組織合作時，流程有時需要資源的共享。這些資源可能是一些內部的服務或是組織內部的人員。所以我們可以在流程中定義需填入的< paramTask >，而使用者再起始流程時必須填入相關資訊。

4.2.2 控制單元(Control Unit)

為了使流程的定義能更有彈性，所以加入了許多控制單元來幫助使用者定義特殊情況的執行，包含 Fork、Join、Par、Switch、While 等等。當程式執行到

```
<fork>
      <task name="task1" def="task/analysis/kmeandef">…<task>
      <task name="task2" def="task/analysis/som"> …</task>
</fork>

<join>
      <task name=" task1"/>
      <task name=" task2"/>
</join>

<par>
      <task>……</task>
      <task>……</task>
</par>
```

<fork>時會同時執行 fork 裡的工作，並繼續執行下面的其他工作。所以一個流程就會有多個正在執行的地方，而流程會等到所有正在執行的工作都完成才會結束。為了要管理 fork 出去的工作，可以使用<join>確保 fork 出去的工作都已經完成才繼續，使的流程能順利的執行。在<join>中定義了要合併工作的名稱，也就是在<fork>裡所定義的工作名稱,這些名稱會隨著工作的執行而傳遞下去,當 join 接收到所有符合條件的工作時才會繼續流程。<par>是簡化版的 fork 與 join，在<par>裡的工作都會被同時執行，並且在所有的工作都要被完成時才會離開。

當流程執行時，有時候我們會讓流程重複做某件工作直到某些條件滿足。這時我們可以使用<while>來達成，在<while>裡可以使用 condition 來判斷情況，當情況為真的時候會執行<while>裡定義的工作，否則就跳出<while>的迴圈。這裡的 condition 使用<if>來測試輸入的格式是否符合 pattern，如果輸入的 Xml 符合 pattern 則回傳 true，否則回傳 false。

```
<while>
      <if> <pattern/> <if>
      <task>……<task>
</ while >

<switch>
   <case><if><pattern> <if>
      <task>……</task>
   </case>
   <default>
      <task>……</task>
   </default>
</switch>
```

另一個包含 condition 的是<switch>，<switch>能讓流程在不同的情況下能決定執行不同的工作，如一般的 switch 一樣有不同的 case，每一個 case 可以定義自己的 condition。當第一個情況滿足時即執行其下的工作，並結束 switch。如果沒有滿足的情況則執行 default。

除了之前所提掉的控制單元之外，有時在流程執行時還需要類似程式的“goto”或“jump”功能，因此在 BioOrch 中提供<activity>可以使用<next>定義下一個要執行的地方，除了<activity>之外包含<fork>、<join>、<par>、<while>和<switch>也都能使用<next>，雖然 Task 不能使用<next>，但是可以定義 Task 的 name，讓其他的<next>可以指定要執行哪一個 Task。

```
<activity name="a1">
      ……
      <next name="a2"/>
</task>
```

4.2.3 錯誤處理與狀態回覆

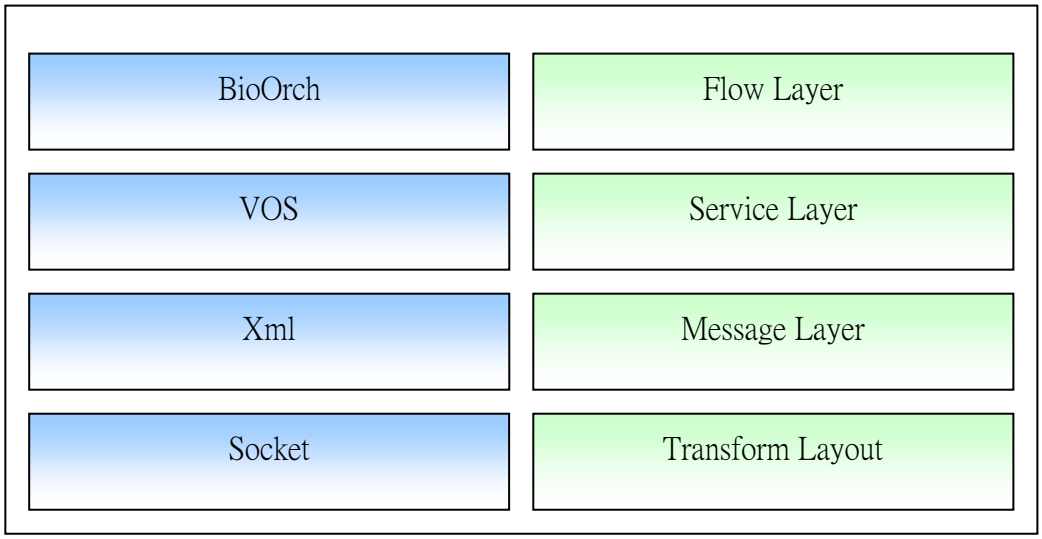
當發生錯誤時，整個流程會停止下來會並將錯誤的訊息傳回流程管理者。如果在流程定義裡有定義 Exception Handler 時，會先交由 handler 處理。Handler 無法處理時，在通知使用者。

```
<processdef name="bio/microsrray/analysis">
    <exceptionhandler handler="ccl.bioorch.exception.DefaultExceptionHandler">
        ……
    <exceptionhandler/>
    <body>
        <activity>
            <exception>SayService not complete，  say is fail <exception>
            ……
        </activity>
    </body>
</processdef>
```

由於所有的工作都是自我滿足的，且完整的紀錄下來。所以可以透過流程管理介面將流程恢復到某個狀態並將其後所產生的工作通通刪除掉。一般來說是不適合做這樣的操作，但是遇到特殊的情況如錯誤產生...等，還是需要有這樣的機制存在。

4.3 系統實作

在 BioOrch 中為了達到軟體元件化的整合，在架構上我們採用 Web Service 的概念，但是省略了複雜繁瑣的部分以利我們發展與測試。Web Service 透過 Socket 收到 Xml 即可以依 Xml 的內容去執行相關的動作，並將結果以 Xml 回傳給呼叫的對象。為了達到抽象的整合方式 BioOrch 對網路服務定義不同的 Type，對 BioOrch 來說擁有相同 Type 的網路服務就會支援相同的操作，而關於 Type 的資訊與服務服務的發布我們使用 Domain 來記錄，使用者可以透過 Domain 來找到符合需求的網路服務。雖然這樣的架構不是標準的 Web Service，但是也可以很容易的與 Web Service 整合起來，因為流程的核心並沒有牽涉到底層傳輸的部分，如下圖所示。

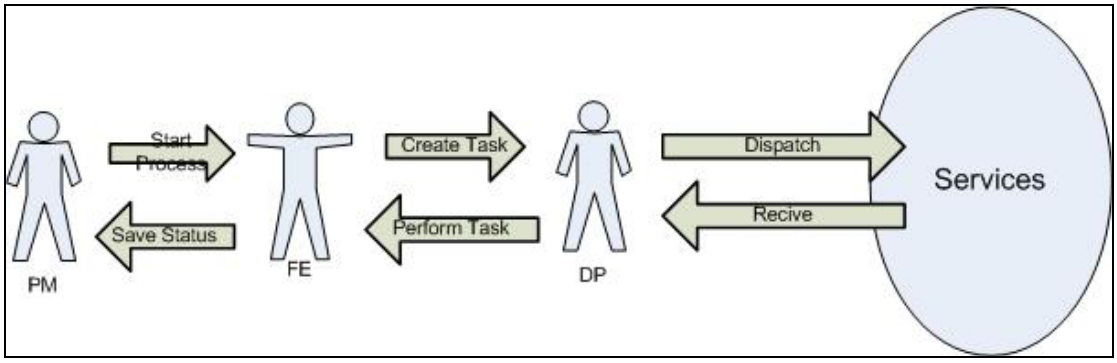


圖表 10、BioOrch Stack

在流程核心中主要包含了幾個部份：定義管理者、流程管理者、流程引擎、人員管理者等。由於流程除了一開始使用到流程定義外，執行時都不需要流程定義的參與，因為相關的定義都已經紀錄在流程裡了，而管理定義的網路服務我們稱之為 Definition Manager(DM)。Definition Manager 使的流程管理者能專注在流程本身。而流程定義的部分也有專責的網路服務處理。在流程管理的部份都交由流程管理者負責，流程管理者對外提供流程服務，對內則提供流程的管理以及維護流程的狀態。流程管理者收到流程起始的訊息後，會使用流程定義起始一個新的流程，此流程會包含所有流程定義的內容，並加入一些流程需要紀錄的資訊如流程的識別碼、流程的狀態與流程的屬性等，之後交由流程引擎處理。流程管理者除了會將流程的資訊保留下來外，對於流程所包含的工作也會另外保存下來。由於我們定義的流程與工作可以使用 Xml 序列化，所以我們可以透過將流程與工作序列化後存在檔案系統內，並於流程更新時，一併更新檔案系統。來達到流程狀態的保存。

系統核心部份的流程引擎(Flow Engine)也是一個 Task Processor，負責協助 Task 執行其內容。在我們的實作中使用另一個網路服務我們稱為 Dispatcher(DP)來負責實際工作的執行。 Dispatcher 本身也是一個 Task Processor，所以也會分

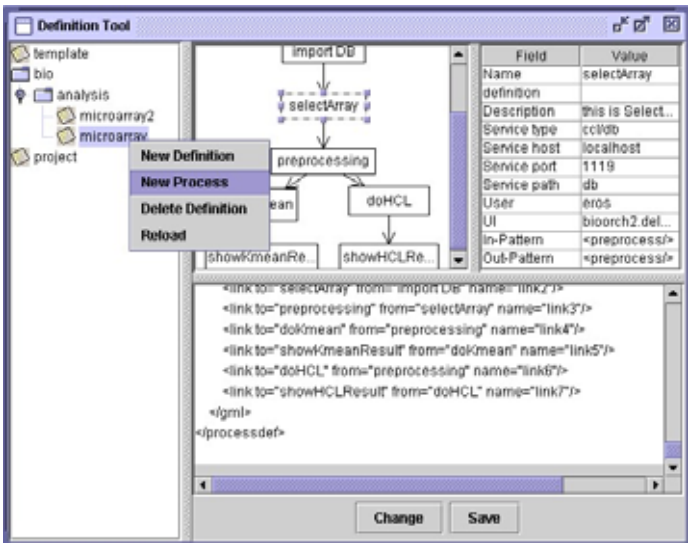
派與回傳。使用如此的設計可將 workflow 的執行和實際的工作執行抽離，避免底層的動作影響到正常環境的運行。下圖說明工作如何在流程管理者、流程引擎與分配者之間傳遞。



圖表 11 工作傳遞示意圖

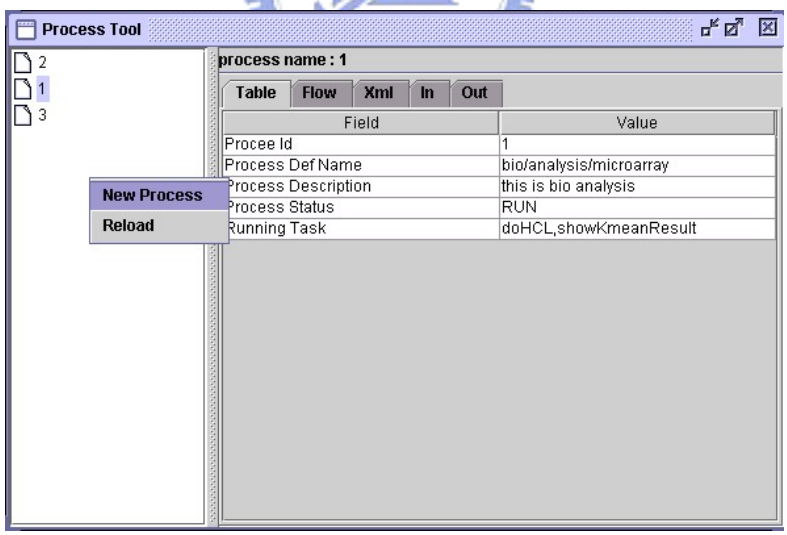
另一個主要的 Task Processor 是人員管理者，負責管理人員資訊 ，記錄人員相關 Task 的資訊 ，協助參與者執行工作..等。如流程與工作一般，人員(User)也可以用 Xml 序列化，將及存於檔案系統中。User 除了紀錄使用者的資訊外還包含使用者需要執行的工作紀錄，使用者管理收到工作時會判斷此工作屬於哪一個使用者或群組。在將工作交由該使用者執行

除了網路服務的部份，我們還開發了一些圖形化的工具，協助使用者完成工作。BioOrch 的使用者介面是架構於一個整合性的工作平台，我們稱之為 Explorer。Explorer 類似一個 Container 可以依照實際的需要加入不同的工具。在 Explorer 中，工具可以定義彼此之間要如何的互動。在這裡主要包含三種工具：流程定義工具、流程管理工具與使用者執行工具。流程定義工具主要是提供使用者友善的視窗介面，並將流程以圖形化的方式呈現。使用者可以透過滑鼠的拖曳或是填入一些參數來完成流程定義的設計。整個工具包含四個部分，如下圖所示。位於左邊的流程定義瀏覽器列出了所有可執行的流程定義。透過流程定義瀏覽器使用者可以新增流程定義、起始流程、刪除流程定義與重新整理瀏覽器。位於中間的流程定義圖將流程以圖型的方式呈現，配合右邊的工作屬性表與下方的文字定義區塊，讓使用者可以很容易且快速的設計流程。



圖表 12 流程定義工具

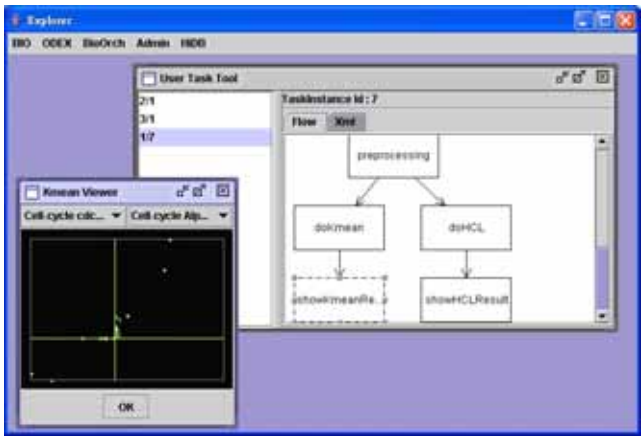
流程管理工具提供使用者監控與管理系統內執行的流程。如下圖所示，位於左邊的流程瀏覽器列出了所有的流程。使用者可以透過流程瀏覽器新增流程、暫停流程、恢復流程與刪除流程。位於右邊的區塊則將流程資訊顯示出來，包含流程的屬性表、流程的狀態圖、流程 Xml 顯示，與流程的輸入輸出。



圖表 13、流程管理工具

使用者執行工具可以協助使用者完成工作的執行，如下圖所示。使用者執行工具從 User Manager 得到使用者需執行的所有工作，並將所有工作列於使用者工作列表。在執行工具的右邊則包含了工作的內容與工作的流程圖。使用者透過工作

列表可以開啟相關工具來完成工作，而這些工具與工作的連結是事先定義在使用者執行工具裡的。當工作完成後，會將工作結果傳會去 User Manager，並更新使用者的工作列表。

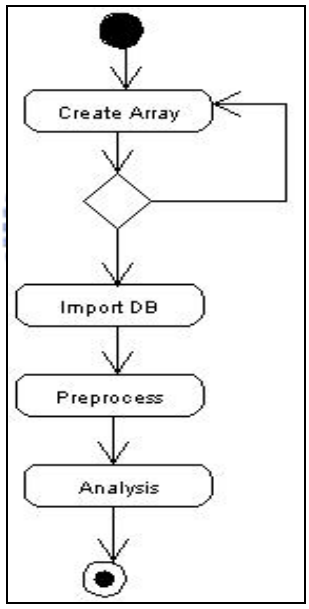


圖表 14、使用者執行工具

第五章 實作範例

5.1 生物晶片分析流程

由於生物晶片技術的問世,使的生物科學研究人員能得到大量的實驗數據以進行分析。本章將以生物晶片資料分析為例,說明我們的系統如何能完成整個處理的流程。如下圖所示,是某單位實際的分析流程,包含建立陣列的資料、儲存至資料庫、資料前置處理與進行分析。該單元所使用的是一套封閉型整合平台,所有的工具與資料都已設置在系統裡。在這個範例中,我們將以 BioOrch 來執行這個流程。過程中我們會示範如何跨組織的合作、如何使用控制單元、如何呼叫其他流程等等。



圖表 15、Mcroarray 流程式意圖

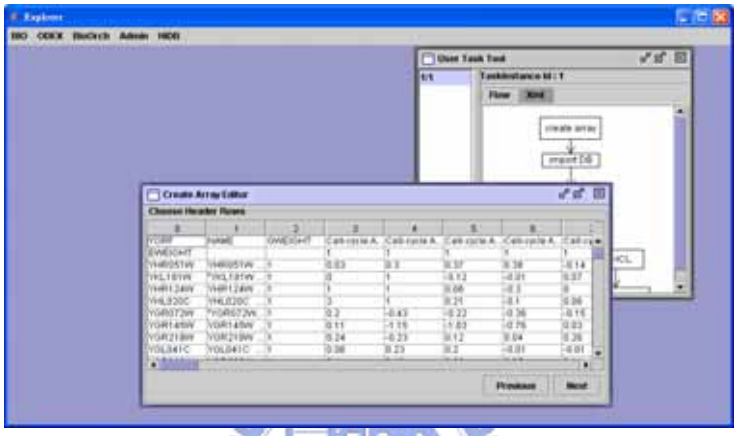
首先會建立 array 的資料,當流程執行到此的時候使用者需完成下列的工作,主要是將陣列的值輸出,但必須做一些需要的前置處理如掃描陣列、量化、去除背景、正規化,最後將陣列數值輸出。

```
<processdef name="bio/analysis/microarray"/>
   <taskdef type="CreateArray">
        <in> <folder path="@path"/></in>
        <out> … </out>
   </taskdef>
   <in><folder path="@path"/></in>
   <out><rw:var name="result"/><out>
   <body>
      <while>
            <ifnot><done/></ifnot>
             <task def="CreateArray"><user name="eros"/><task/>
      </while>
      <task>
            <service name="DBService"/>
            <in>…</in>
            <out>…<out/>
      <task>
      <task>
          <process name="preprocess">
               <paramTask    name="inputData" def="task1">
                    <user name= “…” />
               </ paramTask >
          <process>
          <service name="preprocessServ"/>
          <in>…</in>
          <out>…<out/>
      <task>
      <task def ="kmean"/>
      <task def="showKmean"/>
   <body>
</processdef>
```

表格 3、Microarray 分析流程定義

如上所示，我們定義了一個 while 的 block。在這個 Block 之中我們定義了一個 Task。此 Task 是交給使用者“eros”執行 create array 的動作，並回傳<succ/>繼續執行 create array，或<done/>結束 while。在 while 裡還包含了一個條件測試<infot>

是使用 pattern match 的方式來比對輸入的 xml，如果輸入的 xml 格式符合的話，則回傳 false，反之則回傳 true。第一次執行輸入的 xml 是從外面輸入的，之後是使用 while 裡最後一個 Task 的輸出來比較。在工具方面，我們是使用 ArrayBuilder 來產生 Array 的檔案。ArrayBuilder 實作了系統工具的介面，所以可以整合進我們的使用者操作環境。當使用者要執行這項工作時，系統會通知使用者有這工具可以使用，並可以在使用者操作環境開啟這個工具。



圖表 16、ArrayBuilder 工具

當陣列都建立好之後，必須匯入資料庫儲存。這裡我們使用一個網路服務負責將陣列上傳至資料庫之中以便稍後由資料庫進行處理。如表格 3 所示，在上傳之前我們透過 bind 的動作將 array 的內容從檔案連結到 arraycontent 的變數。接著便執行上傳的動作，由 Flow Engine 連接到一個叫 db 的網路服務。將 arraycontent 上傳，並回傳 array 的 id。

將陣列匯入資料庫後，流程就會交由另一個組織去做分析前置處理，對另一個組織來說，會產生一個前置處理的流程。此流程的輸入需要指定使用哪一個資料庫的哪些資料，如下表所示：再流程定義中使用者必須要填入 task1 的 processor 資訊。而對原本的流程來說必須要將合作流程所需要的資訊填滿，其定義的方法如表格 3 所示。

```
<processdef name="preprocess">
      <paramsTask name=” task1” >
            <in>…<in>
            <out>…</out>
      </pramsTask>
      <in><preprocess id="@id"></in>
      <out><data><rw:var name="content"/></date></out>
      <body>
            <paramTask name="task1"/>
      </body>
</processdef>
```

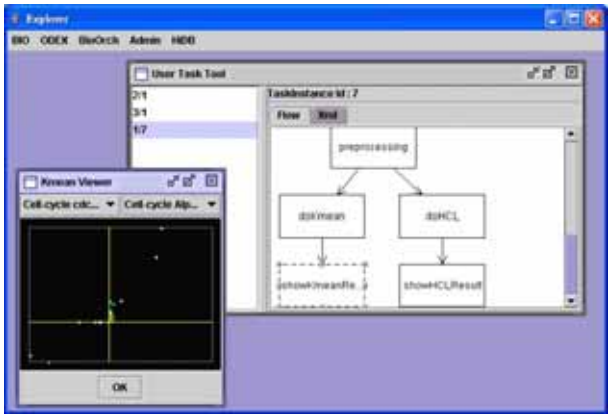
表格 4、Microarray Preprocess 流程定義

當結束後，原來流程的工作就像是呼叫 Web Service 一般接收到回傳值，並繼續下面的工作。資料分析的目的是在找出特殊的基因表現型或是特殊的模式(pattern)。透過不同的演算法來尋找有趣的部份，並加以研究、解釋。常用的演算法如 Kmean、Hierarchical Clustering、SOM 等等。

```
<taskdef name="kmean">
      <service name="analysisServ"/>
      <in><dokmean><rw:var name="predate"></dokmean></in>
      <out><result><rw:var name="kmeandate"></result><out/>
<taskdef>

<taskdef name="showKmean">
      <user name="user"/>
      <in><rw:var name="kmeandate"></in>
      <out><ok/></out>
</taskdef>
```

我們在這個步驟使用的演算法是 Kmean，我們還多加入一個工作使的使用者可以直接看到 Kmean 的結果。

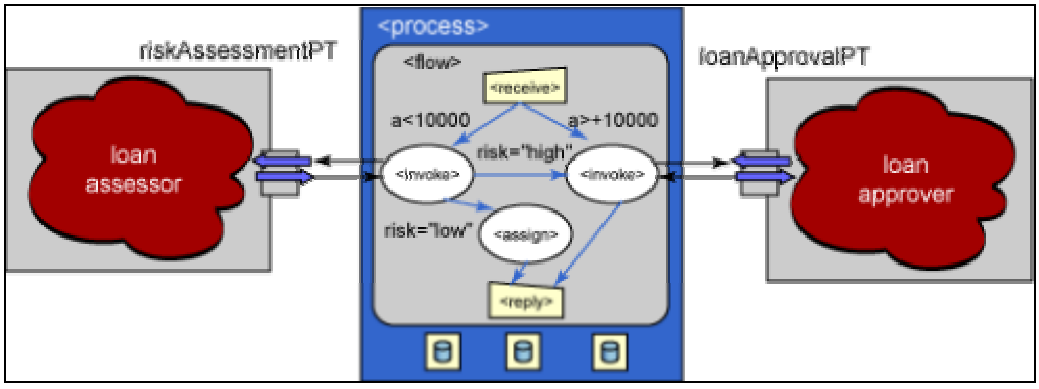


圖表 17、分析工具

使用 BioOrch 來執行生物晶片分析流程，對流程的參與者來說，可以明確的知道現在流程進行的情況，也可以知道工作的內容。在跨組織的部分，我們也展示了如何使用不同組織的流程。這樣的系統同樣可以執行生物科學研究人員的分析流程另外還提供了流程的管理與跨組織合作的優點，提高了實驗的效率並降低研究成本。

5.2 貸款流程

以下重新實作 BPEL4WS 貸款的例子[8]。如圖表 18 所示，當貸款的要求進入流程系統後，同時需要輸入使用者的姓名和貸款金額。當金額大於 10000 時會交由 Loan Approver 決定是否核可。如果金額小於 10000 則會交由 Loan Assessor 先作風險評估，評估結果風險高則交回 Loan Approver 處理，風險低則直接核可貸款申請。



圖表 18、BPEL4WS 貸款流程圖[8]

首先輸入貸款的資訊，如表格 5 所示：包括 first name、 name、 amount。我們使用 pattern binding 和 Task 輸入輸出，來完成資料的儲存與資料的流動。由於輸入的格式和“比較 amount 大小”的 Task 不相同，所以我們使用“util/bindTranslator”這個工作定義來幫我們轉換格示。將輸入的變數與實際的值 binding 起來，並再輸出時使用 binding 的值轉換成需要的格式輸出。下一個比較大小的工作則使用“util/math”這個工作定義，當<compare>裡的“value”大於“to”時會回傳 value=“1”，反之則回傳“0”。

```
<processdef name="loan" package="test">
    <in><credit firstname="@fn" name="@nmae" amount="@amount"/></in>
    <out><approval accept="@accept"/></out>
    <body>
        <task def="util/bindTranslator">
            <in><credit firstname="@fn" name="@nmae"
amount="@amount"/></in>
            <out><compare value="@amount" to="10000"></out>
        </task>
        <task def="util/math">
            <in><compare value="@amount" to="10000"></in>
            <out><result value="@compareResult"></out>
        </task>
```

表格 5、貸款流程(1)

比較完成後，我們使用 Switch 來比較輸出的結果與決定執行的部分。如果結果是“0”，則交由 Loan Assessor。我們將風險評估的部份交由內部組織的人員 loan-assessor 群組處理。當風險評估的結果是"low"的時候，則直接接受使用者的申請。當結果是"high"的時候，則回到一般的申請流程，在這裡使用<next>來指定下一個執行的部分。在 BioOrch 可以在各個 Activity 中使用<next>來指定下一個要執行的 Activity 或 Task，讓使用者能更有彈性的定義自己的流程。

```
<switch>
<case><if><result value="0"></if>
      <task def="loan/riskAssessment">
            <in><credit firstname="@fn" name="@nmae" amount="@amount"/></in>
            <out><result risk="@risk"></out>
            <user group="loan-assessor"/>
      </task>
      <switch>
            <case><if><result risk="low"></if>
                  <task def="loan/riskAssessment">
                        <in><result risk="low"></in>
                        <out><approval accept="yes"/></out>
                  </task>
                  <next name="end"/>
            </case>
            <default>
                  <next name="loanApproval"/>
            </default>
      </switch>
</case>
<case>
      <if><result value="1"></if>
            <next name="loanApproval"/>
      </case>
</switch>
```

表格 6、貸款流程(2)

貸款核准的部分則是交由另一個組織的流程處理，有別於原來的例子，在核准流程中加入"貸款者在該組織存款”的加權，使的貸款的核准與否和其存款有關。由貸款者的存款金額應該是組織內私有的資訊，所以我們將這個工作交回原組織執行。在原本的流程中定義了要使用“test/LoanApprover”這個流程，並將<paramTask>填入，指定由 bankSavingsGroup 來計算加權的部份。當對方流程執行到存款加權的部份，則會將工作交回執行。BioOrch 使用了<paramTask>定義

流程間互動的機制，而且因為 Task 是可以傳遞的執行單元，使的跨組織的合作能更緊密。

```
        <activity name="loanApproval">
                <task def="loan/apporver">
                    <in><credit firstname="@fn" name="@nmae"
amount="@amount"/></in>
                    <out><approval accept="@accept"/></out>
                    <service type="loan/approver">
                    <process name="test/LoanApprover">
                        <paramTask    name="bankSavingsWeight"
def="loan/bankSavingsWeight">
                                <user group="bankSavingsGroup"/>
                        </paramTask >
                    <process>
                </task>
        </activity>
        <activity name="end"/>
    </body>
</processdef>
```

表格 7、貸款流程(3)

以下是 test/LoanApprover 流程定義，總共有三個步驟：(1)初步加權 (2)銀行存款加權 (3) 核准結果。並在這個流程定義值中指定了需要輸入的<paramTask> - “bankSavingsWeight”，在這裡只定義了輸出輸入的格式，其他的定義則需要由要求者輸入。當要求者起始流程並填入<paramTask>時流程會將填入的部分紀錄下來，直到執行到引用<paramTask>部分，則會將此 Task 實體化並填入上一個工作的輸出，傳回要求者執行。當要求者完成後會回傳執行完的 Task，流程會將此 Task 的輸出當成下一個 Task 的輸入，並繼續流程的運行。

```
<processdef name="LoanApprover" package="test">
    <in><credit firstname="@fn" name="@nmae" amount="@amount"/></in>
    <out><approval accept="@accept"/></out>
    <paramsTask name="bankSavingsWeight">
        <in><weight value="@initWeight"><in>
        <out><weight value="@newWeight"></out>
    </pramsTask>
    <body>
        <task def="loan/weighter">
            <in><credit firstname="@fn" name="@nmae"
amount="@amount"/></in>
            <out><weight value="@initWeight"><out>
        </task>
        <paramsTask name="bankSavingsWeight"/>
        <task def="loan/approver">
            <in><weight value="@newWeight"></in>
            <out><approval accept="@accept"/></out>
        </task>
    </body>
</processdef>
```

表格 8、貸款核可流程

以 BioOrch 來重新實作 BPEL4WS 的貸款流程，能更為明確的達到組織之間資訊隱藏與資源的分享，並且能將工作分配給組織內部的人員，使的內部人員能直接參於流程的執行與接受流程的管理。最後列出 BPEL4WS 和 BioOrch 的比較:

- *資料的處理*。BPEL4WS 可以定義傳輸<message>的格式，<message>會被放到<container>裡，並使用<assign>來做資料的操作與搬移，而 BioOrch 則是使用輸入輸出的 Xml Pattern 來定義資料傳輸的格式，並用變數 Binding 來保存重要的資料，一個工作的輸出是下一個工作的輸入，透過輸入輸出的傳遞讓資料流動。
- *跨組織的合作*。BPEL4WS 將不同組織之間的流程都當成一般的 Web Service 處理，所以對流程的合作並沒有特別的定義。而 BioOrch 則是

在流程定義中使用<paramTask>定義要交回執行的工作內容。使的 BioOrch 能像使用 Web Service 一樣使用其他組織的流程，還可以在流程進行間交互影響與分享資源。

- *組織人員需求*。BPEL4WS 並沒有將人員的概念加入流程之中，而 BioOrch 則可以在流程中指定參與者執行工作，包括群組和使用者。使的參與者也能透過流程管理系統管理
- *可延伸的流程定義*。BioOrch 允許使用者發展自己的工作定義，使的系統能讓使用者依照實際的需求作更改，提供使用者更大的彈性。

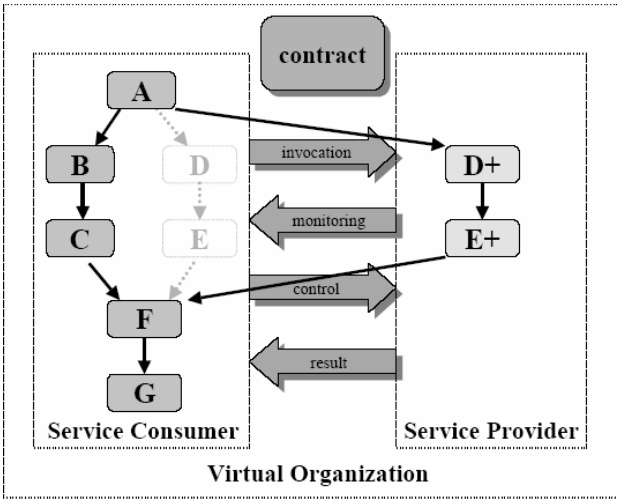
第六章 相關研究

本章將介紹一些流程管理的相關系統與標準，包含工作流程管理聯盟的 XPDL、myGrid 的流程系統 Taverna、contract-base 的 CrossFlow project 與 OpenSource 的 jBpm project。

工作流程管理聯盟(WFMC)[20]，目的在透過專有名詞的界定，建立與其他工作流程的溝通橋樑，來推廣與發展工作流程，以因應企業對於流程自動化的需求且解決工作流程之標準化問題。WPDL[21]是由 WFMC 所提出來的流程定義語言，其目的在定義流程管理系統間資料的交換，包含許多 Activity，每一個 Activity 由 Transitions 連結起來，Transitions 可以使用 split 和 join，另外還定義了許多資料的格式在流程進行中使用。雖然他並沒有特別定義組織之間流程互動的機制，但是許多 WPDL 的概念也同樣適用於跨組織流程系統。

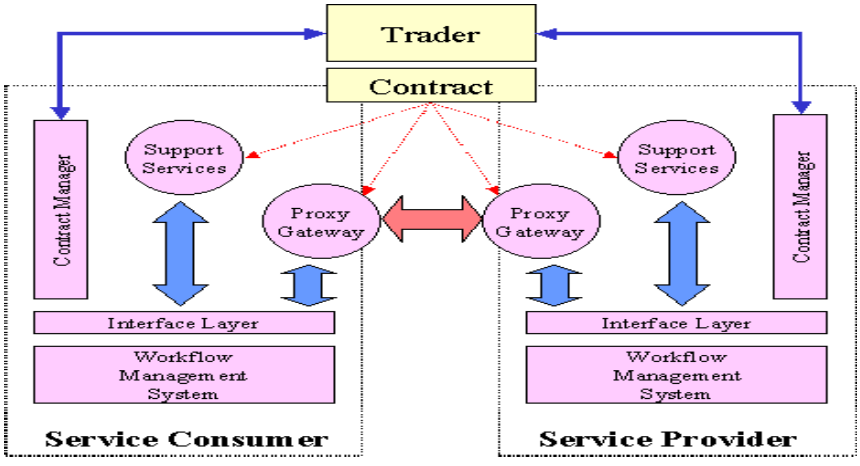
Taverna[11]也是一個生物資訊分析流程系統，是 MyGrid Project 的一個部分，目的是提供生物科學研究人員一個流程語言和相關工具，以 Workflow 有效的完成實驗步驟[14]。其自定發展的語言 XScufl，支援的操作元素包含：WSDL 模式的 Web Service、SoapLab、Talisman 和其他 scufl 流程等。透過資料流的概念，輸入資料會從 source 透過 link 流向不同的 processor，最後流到 sink 停止，將使用到的操作元素都當成 processor 定義在語言裡，而流程需要的參數在一開始的時後就必須要填滿。Taverna 還提供了一個工作環境，將 workflow 需要的工具都整合成一個介面，包括 Advance model explorer、Available Services List、Workflow diagram。跟 BioOrch 不同的是，他並不支援跨組織的合作，它的目的是針對生物科學研究人員使用的，主要專注在整合 Web Service 完成一項特定的目的，而過程中並沒有參與者的概念在流程中。而 BioOrch 可以在流程中指定某些事情要由相關人員完成

以 Contract-Base 為導向的 CrossFlow[32]，目的是要提供動態的跨組織的工作流程支援。他是以實際需求發展的系統，就如現實生活中的情況一樣，企業之間的合作是透過合約來規範，所有的需求與資源都明訂在合約裡。



圖表 19、ClossFlow 示意圖[36]

而為了要動態的促使組織之間的合作，所以使用了 Trader 來尋找或提供服務。依照發布在市場裡的消息，Trader 會去幫彼此配對，如果吻合了便會通知雙方進行合作。透過內部的溝通機制如 Cooperation Support Services (CSS)、Proxy-Gateways(PG)、Coordinators (Coord)，來完成資源分享與監控管理。

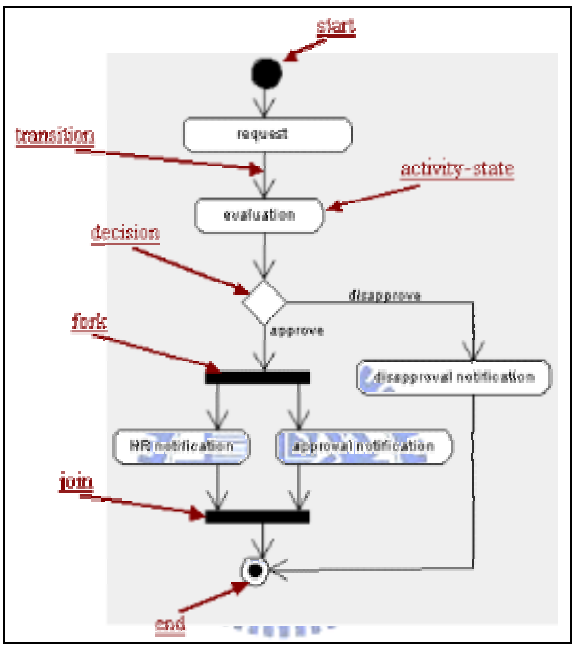


圖表 20、CrossFlow 執行架構圖[36]

不同於 CrossFlow 將合作部份與工作流程獨立。BioOrch 本身即支援跨組織的合作，從資料的互動與資源的共享，透過工作的輸入輸出與 Task Processor 的概念，

使的跨組織的合作能輕易的達成，不需要透過複雜的定義就能保有了公開與私有的彈性。

Opensource 的 jBpm[33]提供了管理者和開發者之間的共同語言，jBpm Process Definition Language (jPdl)，一種以 UML 為延伸的語言。再 jPdl 裡大部分的關鍵字都是延續 UML 的使用方式，並且在定義語言時能以圖形流程的思考方式制定。



圖表 21、jBpm 流程語言元素組成圖[33]

另外他還包含了webinterface.xml來定義外觀該如何顯示。它的整個系統是架構在J2EE上(jBpm2嘗試將整個架構改以J2SE)，包含前端網頁jsp的部分、中間的流程邏輯EJB、和後端的資料庫。jBpm與其他語言最大的不同就是能延伸定義語言，由使用者自行擴充需要的定義。幾乎所有流程中主要部分都可以延伸擴充。 我們的系統延續了jBpm的延伸性，允許使用者擴充自己的流程語言，並加到流程中。與其不同的是我們將工作變成是可延伸的，透過發展各種Type的工作定義，讓工作的內容能視使用者的需要調整。

第七章 結論

本論文中我們發展了 BioOrch 生物資訊流程管理系統，使的不同組織之間能輕易的使用彼此的流程，達到跨組織合作的目的。有別於一般工作流程整合系統，我們使用工作導向的流程管理方式，使的我們再跨組織合作時能更有效的整合彼此的資源。 我們將使用者當作是流程中的一個重要元素，因為在生物實驗流程中有許多部份是需要使用者參與的。 綜合言之， 我們設計的生物科學研究環境成功的達到了以下的結果

- **跨組織的合作** 流程的互動。不同組織可以使用彼此的工作流程，將其他組織的工作流程當作自己流程的一部分。資源共享。組織之間可以透過流程而達到資源共享
- **易整合的系統環境** 易於加入新的網路資源。我們的系統可以很有彈性的加入及使用新的網路資源，透過 Web Service，不管是生物資料庫、演算法分析工具、高速運算服務...等，都可以很輕易的整合進我的的系統裡。易於不同格式的資料流傳。我們系統允許有不同格式的資料在流程中傳遞、轉換。使用 Xml 為統一的表達方式，透過簡單的 Binding 動作將資料輕易的從一種格式，轉換到另一種格式。
- **自動化的工作流程** 避免人為因素導致錯誤。流程管理都交由電腦執行，電腦會依照定義的內容將實驗步驟在適當的時後交由負責的人去處理。並且將實驗中所產生的資訊一一紀錄下來，以便將來查詢及檢視實驗的過程。 協助生物科學研究人員完成工作。當研究人員要執行一項工作時，系統可以告知工作的內容或開啟適合的工具...等，來協助使用者完成工作。 易於監控與管理。透過流程管理工具，可以很輕易的查詢現在有多少實驗正在進行、實驗進行到什麼步驟、該步驟是由誰負責的或是查詢以完成的實驗結果...等等。除了查詢流程的相關資訊，管理工具還能將流程刪除或是將流程回復到某個狀態。 提高效率、降低成本。自動化管理除了可以減少不必要的人力

資出，也可以檢視流程的過程是否有可以改進的地方，以節省成本、提高效率。

- **友善的使用者介面** 使用圖形化的使用者介面。依據不同的角色開發適合的使用者工具，如針對流程定義所設計的工具，使用圖形化的設計工具讓使用者能輕易的制定流程，針對流程管理所設計的工具，能使用流程圖的方式顯示流程的進度..等等。 協助使用者完成工作。當使用者登入之後， 使用者可以看到使用者工作列表，並選擇工作去執行它。在工作列表上可以看到工作的相關說明包含工作在流程中的什麼地方、工作執行的目的、工作執行的內容、相關的工具...等等。 易於發展網頁介面。對使用者來說，有時使用網頁介面會較為方便，由於我們擁有定義良好的溝通介面，而且使用簡單的 Xml 格式當作訊息傳遞，所以可以很容易的發展相關的網頁介面。不管是流程定義，流程管理還是使用者介面，都可以使用網頁形式來表達。

我們認為本研究針對生物資訊整合的問題提供一個很好的解法，透過工作流程來整合生物資訊，我們已經達到能有彈性且可擴充性的設計去解決生物資訊特別的需求，且持續保持友善的使用方式。未來我們可以朝基於我們現在建立的架構上，與更多不同領域的生物學家，合作研究，建立更多不同的 Web Service 以增強儲存與計算的能力 。但是還是有許多關鍵的部份我們需要去加強。然而，就研究生物資訊而言，我們在工作流程的部份已經以適當的方法，來勾勒出我們認為的生物研究環境的藍圖。例如，我們並不試著去影響使用者如何去創造它的流程，但是我們執著於如何有效降低成本，提高效率，並符合使用者直覺。我們也使用許多方法來增進使用的效率如自動化的 Service 驗證、容易建構不同的使用者介面..等，更重要的是整合了生物資訊所需要的資料儲存、計算服務，也整合了異質與分散的問題。

參考文獻

[1] Genbank, National Center for Biotechnology Information (NCBI), National Institute of Health, USA. http:/www.ncbi.nlm.nih.gov/Genbank

[2] European Molecular Biology Laboratory (EMBL), European Bioinformatics Institute (EBI), European. http://www.ebi.ac.uk/embl/index.html

[3] DNA Data Bank of Japan (DDBJ), National Institute of Genetics (NIG), Japan. http://www.ddbj.nig.ac.jp

[4] European Molecular Biology Open Software Suite (EMBOSS), EMBnet, European. http://www.embnet.org/index.html

[5] Genetics Computer Group (GCG), Accelrys Corp. http://www.gcg.com/

[6] Sequence Retrieval System (SRS), EBI, European. http://srs.ebi.ac.uk

[7] European Bioinformatics Institute, European, http://www.ebi.ac.uk

[8] *BPEL4WS Specification: Business Process Execution Language for Web Services Version 1.1*, BEA, IBM, Microsoft, SAP AG and Siebel Systems, May 2003. http://www-106.ibm.com/developerworks/library/ws-bpel/

[9] *Web Service Choreography Interface (WSCI) 1.0. Standards*, World Wide Web Consortium (W3C), August 2002. http://www.w3.org/TR/wsci/

[10] Martin Senger, Peter Rice, Tom Oinn, *Soaplab - a unified Sesame door to analysis tools*, Proceedings UK e-Science All Hands Meeting 2003, September 2003, pp. 509-513.

[11] Taverna project, The Engineering and Physical Sciences Research Council (EPSRC), UK. http://sourceforge.net/projects/taverna

[12] *Web Services Architecture Working Draft*, World Wide Web Consortium (W3C), February 2004. http://www.w3.org/TR/ws-arch/

[13] *Process Definition Interchange Process Model*, Workflow Management Coalition (WFMC), Document Number TC-1016-P, October 1999. http://www.wfmc.org/standards/docs/TC-1016-P_v11_IF1_Process_definition_Interchange.pdf

[14] Matthew Addis, Justin Ferris, Mark Greenwood, Peter Li, Darren Marvin, Tom Oinn, Anil Wipat, *Experiences with e-Science workflow specification and enactment in bioinformatics*, Proceedings UK e-Science All Hands Meeting 2003, pp. 459-466, 2003.

[15] M. Greenwood, C. J. Wroe, R. D. Stevens, C. A. Goble and M. Addis, *Are bioinformaticians doing e-Business?*, Proceedings Euroweb 2002: The Web and the GRID - from e-science to e-business, pp. 3-26, 2002.

[16] Martin BERNAUER, Gerti KAPPEL, Gerhard KRAMLER, Werner RETSCHITZEGGER, *Specification of Interorganizational Workflows - A Comparison of Approaches*. Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2003), pp. 30-36, July 2003.

[17] Rania Khalaf, Nirmal Mukhi, and Sanjiva Weerawarana, *Service-oriented composition in BPEL4WS*, presented at The Twelfth International World Wide Web Conference, 2003.

[18] Prasad Yendluri, Principal Architect, webMethods, Inc. *Web Services Choreography*. WebServices.org, September 2003. http://www.webservices.org/index.php/article/articleview/1178/

[19] W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. *Web Service Composition Languages: Old Wine in New Bottles?* , Proceeding of the 29th EUROMICRO Conference: New Waves in System Architecture, pp. 298-305 , 2003.

[20] The Workflow Management Coalition, http://www.wfmc.org/

[21] Michael zur Muehlen, Jörg Becker. *Workflow Process Definition Language –Development and Directions of a Meta-Language for Workflow Processes.* KnowTech Forum, September 1999.

[22] DBCAT, INFOBIOGEN, France. http://www.infobiogen.fr/services/dbcat

[23] Protein Data Bank (PDB), Rutgers, The State University of New Jersey, the San Diego Supercomputer Center at the University of California. http://www.rcsb.org/pdb

[24] CATH, BSM group, University College London, UK. http://www.biochem.ucl.ac.uk/bsm/cath/

[25] Structural Classification of Proteins (SCOP), MRC Laboratory, University of Cambridge, UK, http://scop.mrc-lmb.cam.ac.uk/scop/

[26] Protein Information Resource (PIR), Georgetown University Medical Center (GUMC), USA. http://pir.georgetown.edu/

[27] The Gene Expression Omnibus(GEO), NCBI, USA. http://www.ncbi.nlm.nih.gov/geo/

[28] Stanford MicroArray Database (SMD), Stanford University School of Medicine, USA. http://genome-www.stanford.edu/microarray

[29] ArrayExpress ,EBI, European. http://www.ebi.ac.uk/arrayexpress/

[30] PubMed , NCBI. http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=PubMed

[31] European Bioinformatics Institute (EBI) toolbox, http://www.ebi.ac.uk/Tools/index.html

[32] Y. Hoffner, H. Ludwig, P. Grefen, K. Aberer. *CrossFlow: Integrating Workflow Management and Electronic Commerce.* ACM SIGecom Exchanges, Vol. 2, No. 1, pp. 1-10, 2001.

[33] Java Business Process Management (jBpm) ,Tom Baeyens, Sourceforge . http://www.jbpm.org

[34] Basic Local Alignment Search Tool (BLAST) , NCBI, USA

http://www.ncbi.nlm.nih.gov/BLAST/

[35] Entrez , NCBI, USA. http://www.ncbi.nlm.nih.gov/gquery/gquery.fcgi

[36] CrossFlow, Virtual Enterprises ESPRIT Project, European

http://www.crossflow.org/