國立交通大學

資訊科學系

碩 士 論 文

本體論驅動資料探勘在生物資訊之應用

Ontology Driven Bioinformatics Data Exploration

研 究 生：劉許吉

指導教授：陳俊穎　教授

中 華 民 國　九 十 三　年 六 月

本體論驅動資料探勘在生物資訊之應用

Ontology Driven Bioinformatics Data Exploration

研 究 生：劉許吉　　Student：Chi Liu-Shu

指導教授：陳俊穎　　Advisor：Dr. Jing-Ying Chen

國 立 交 通 大 學

資 訊 科 學 系

碩 士 論 文

A Thesis

Submitted to Department of Computer and Information Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer and Information Science

June 2004

Hsinchu, Taiwan, Republic of China

中華民國九十三年六月

本體論驅動資料探勘在生物資訊之應用

學生：劉 許 吉　　指導教授：陳 俊 穎 博士

國 立 交 通 大 學

資 訊 科 學 系

摘　　要

本論文提出一個整合式生物資訊工作環境，稱為 ODEX，在這環境上允許有彈性地將隱涵著異質性且分散的生物資訊工具，透過不同層次的整合模式而整合。包括從發展階段地元件化整合、至動態地以規則為基礎的工具組合，進而藉由描述領域知識來連結工具的 ontology 輔助式資料探勘。ODEX 能夠支援不同領域技術和經驗的使用者，並輔助管理不同的分析工具和資料庫系統。此外，藉由 ontology 的輔助，ODEX 能建議適當的分析步驟並引導使用者於高複雜且相互連結的生物資料之間作有效率地資料分析及探勘。

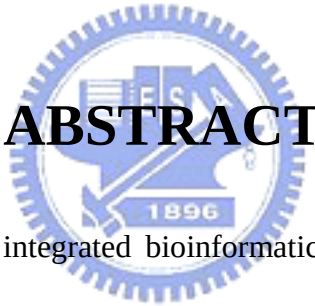
關鍵字：生物資訊、資料探勘、本體論、整合式系統

Ontology Driven Bioinformatics Data Exploration

Student：Chi Liu-Shu Advisor：Dr. Jing-Ying Chen

Department of Computer and Information Science

National Chiao Tung University



ABSTRACT

This thesis proposes an integrated bioinformatics environment, called ODEX, that allows flexible integration of potentially distributed, heterogeneous bioinformatics tools at different levels, ranging from development-time, component-based integration, to dynamic, rule-based tool composition, to ontology-assisted data exploration linking tools with domain-specific knowledge. ODEX supports users with different expertise and skills, and helps them manage different analysis tools and database systems. Furthermore, with the help of ontology, ODEX can suggest suitable analysis steps, guiding user in the exploration of highly complex, interconnected biological data.

Keywords: bioinformatics, data mining, ontology, integrated system

誌　　謝

首先要感謝我的指導教授陳俊穎教授，在碩士學業的兩年間悉心指導，往往在研究遭遇瓶頸的時候，給予我正確的方向，且在面臨口試將近的日子中，謹慎細心的修正論文的內容與概念，才使得論文能如期順利完成。同時也感謝劉興民教授、胡毓志教授百忙之中抽空來擔任我的畢業論文口試委員，並給與我許多寶貴的意見，在此特別感謝。

感謝實驗室的伙伴，舜禹、訓宏、建宏在這兩年一起打拼，不論是在生活上或是學業上皆給與我莫大的幫忙。同時也感謝許許多多在交大一起生活一起運動的朋友們，這兩年大家互相鼓勵、互相扶持，使得在研究領域之外更是受益良多。還有其他目前在不同學校就讀研究所的朋友們，常常在學術資源上相互交流，使得研究資源不予匱乏。

此外也要特別感謝我的家人，因為有他們的支持所以才能使我能在無後顧之憂的環境下學習，並且能專心完成我的學業，在此特別的感謝他們。

劉許吉　謹誌　2004年6月

於交通大學研究生室

目　　錄

摘　　要 ..... **I**

ABSTRACT ..... **II**

誌　　謝 ..... **III**

目　　錄 ..... **IV**

表 目 錄 ..... **VI**

範例目錄 ..... **VI**

圖 目 錄 ..... **VI**

第一章、緒　　論 **1**

第二章、文獻探討 **5**

第二節 API層次整合式系統 5

第三節 元件化基礎整合式系統 6

第四節 進階元件化整合式系統：ONTOLOGY-BASED系統 8

2.4.1 Ontology之簡介與探討 *8*

2.4.2 Gene Ontology之簡介 *10*

2.4.3 描述Ontology的語言 *10*

2.4.4 RDF (Resource Description Framework) 之簡介 *11*

2.4.5 OWL之簡介 *12*

2.4.6 以Ontology為基礎的生物資訊系統 *13*

第三章 問題歸納、目標與方法 **16**

第一節 問題歸納 16

第二節 目標與方法 17

第四章、服務導向之整合環境 **20**

第一節 服務導向之環境架構 20

第二節 以XML為基礎之溝通協定 22

第三節 服務容器的建構 23

第四節 服務容器的管理 24

第五節 服務註冊 26

第六節 以規則為基礎的組合模式 27

第七節 本章總結 29

第五章、ONTOLOGY驅動之資料探勘 **31**

第一節 ODEX的ONTOLOGY原型(ONTOLOGY MODEL) 31

第二節 ONTOLOGY驅動之ODEX系統細部架構.....34
第三節 ONTOLOGY驅動之ODEX系統重要元件.....35
5.3.1 Ontology服務.....*35*
5.3.2 工作集服務 (Working Set Service).....*36*
5.3.3 附加工具集.....*37*
第四節 本章總結.....38
第六章、ODEX系統實作.....39
第一節 樣版整合.....39
第二節 定義規則驅動引擎(RULE ENGINE).....40
第三節 ODEX ONTOLOGY之描述.....44
第七章 ODEX系統範例.....47
第八章 結論.....51
參考文獻.....54

表 目 錄

表格 1 目前生物資訊資料庫與資料型態對照表 1

範例目錄

範例 1 樣版整合範例 39
範例 2 ONTOLOGY XML格式 45
範例 3 系統導覽員選單設定片斷。本圖可對應至圖 20 的介面，此片斷可看出設定了一個為「ODEX」的選單，而接著將一個「DATA BROWSER」的選項加入至「ODEX」的選單中，而此選項的執行指令就描述在<ACTION/>這個標籤中，本範例是要將"ODEX/DATABROWSER"此路徑下的工具開啟。 48
範例 4 描述資料型態的ONTOLOGY 49

圖 目 錄

圖 1 ISYS之系統架構圖(截取自*A. SIEPEL, ET AL.* ISYS: A DECENTRALIZED, COMPONENT-BASED APPROACH TO THE INTEGRATION OF HETEROGENEOUS BIOINFORMTICS RESOURCES. BIOINFORMATICS, VOL. 17 NO. 1, 2001, P 83-94.) 7
圖 2 簡單的RDF概念示意圖。 12
圖 3 TAMBIS系統之架構圖 13
圖 4 分散式互助合作的生物資訊研究環境 17
圖 5 ODEX系統使用者層次示意圖 20
圖 6 以服務容器組成的ODEX系統架構圖 21
圖 7 XML為基礎之服務導向基礎架構 22
圖 8 服務建構示意圖。在HOST中的服務是以適當路徑加以管理。而每一個服務依需求會要求在建構時設定需要的服務。(A)此為兩個HOST，每個HOST都是自己所維護的服務容器。(B)使用者在建構自己的服務時，可依需求填入適當的前置服務，使得每個服務都可互相分享使用。(C)當使用者將前置服務配置完成，則服務間就間接地建立了一條無形的關係，將服務們串連起來。 24
圖 9 HOST之設計示意圖 25
圖 10 HOST控制介面。每一個服務都用一個XML設定檔，描述服務執行時所需要的其他服務，以及相關設定。 25
圖 11 DOMAIN架構與HOST之關係圖 27
圖 12 DOMAIN瀏覽員。圖中指定了ODEX / DATA / DATASERVICE 這個型態的服務，右邊顯示了此型態的服務所提供的XML介面。 27
圖 13 規則基礎組合機制流程圖 28
圖 14 ODEX ONTOLOGY模型 32
圖 15 加入ONTOLOGY驅動與規則驅動引擎之ODEX系統架構圖 34
圖 16 ODEX ONTOLOGY範例圖 36

圖 17 樣版整合示意圖。將樣版定義成左右兩個區塊，並分別將區塊所將配置的工具設定，即可將工具嵌入至樣版中。 40
圖 18 規則驅動引擎之規則比對流程圖 41
圖 19 ONTOLOGY示意圖，對應範例 1 中所描述的ONTOLOGY與之對應。 46
圖 20 系統導覽員 47
圖 21 9(A)中，顯示當游標選擇資料(YEAST/PART該筆資料型態為MATRIX)時，跳出式選單所列出可執行的動作。與 9(B)比較，當選擇的資料(KMEANS目錄下的資料型態為CLUSTERRESULT)是屬於不同型態的資料時，所列出的清單也會不同。 48
圖 22 系統導覽員執行資料分析時，執行KMEANS與階層式分群法的結果 50
圖 23 樣版式整合。最左上角與中間的視窗為分解的兩個工具，經樣版整合後可獲得右下角視窗的工具。 50


E
S
A
1896

第一章、緒　　論

生物資訊流行的風潮正在全球快速延燒，資料在生物資訊領域呈爆炸性地成長，許多生物資訊資料庫相繼建立，提供大量公開性研究資料的上傳和學術分享，以及提供資料的取得和資源的維護。從資料型態的觀點將這些資料庫做分類，每一個資料庫分別用來維護特定資料型態的的資料。現在比較廣為人知的資料庫包含，維護 nucleotide 資料型態的資料庫有 GeneBank[1]、EMBL[2]、DDBJ[3]、NSD，維護蛋白質資料型態的資料庫有 PDB[5]、SWISS_PROT。其他資料庫負責維的資料型態列在表格 1 中。

Data Type	Database
Nucleotide	GeneBank , EMBL, DDBJ, NSD
Protein	PDB, SWISS-PROT
Molecular Structure	Protein : NCBI, EBI MSD
Gene Expression	EBI ArrayExpress, NCBI GEO
Regulation Network	KEGG
Species-specific	mouse , fly ...etc

表格 1 目前生物資訊資料庫與資料型態對照表

這些資料型態不旦在生物領域中代表不同生物意義，而且相互之間往往都有特定的關係。最簡單的關係就是，DNA 轉錄成 RNA，RNA 轉譯成蛋白質。所以 nucleotide 資料庫中不僅要儲存基因序列，而且還要紀錄相對應的胺基酸結構，因為 DNA 和蛋白質之間為來源和產物的關係。其他還有蛋白質的分子結構對於該蛋白質的生物功能及其在新陣代謝中參與的反應占很重要的地位，所以蛋白質資料庫都會紀錄其分子結構與其生物功能。由此可知，資料型態之間的關係相當繁複。

此外，隨著生物資訊的技術日益進步，如：生物微陣列晶片[4]的發明，使得生物學家一夕之間可以產生數以萬計的實驗數據。生學學家為了尋找隱藏在資料中有價值的訊息，必需仰賴特定的分析工具，來計算資料的關聯性，再由生物學家從分析結果中解讀出重要的意涵。因此，為了使生物學家有效率地處理大量的資料，程式開發者開發出許多問題導向的工具，使生物學家可以操作這些分析工具處理各式各樣的數據。

因為有了這些需求，許多組織或研究單位都相繼發展資料分析工具。然而，因各自採獨立開發，造成不同單位所開發出來的工具難以相互合作，且即使一些功能相同功能的工具，它們所接受的資料型態或資料格式也不盡相同。生物學家為了產生工具可解讀的輸入，必需自己撰寫一個程式，只為了轉換資料格式；或是尋找一系列可相容的分析工具，才能正確的執行理想的結果。如此的異質性成了生物學家在數據分析上的一大阻礙。為了解決這種工具的異質性，有許多整合系統被提出，將一系列相關的工具，整合至同一個系統中，工具之間的溝通交給系統去管理，以降低系統在使用上的門檻。

簡單的整合機制透過程式的 API 將不同的工具組合成一個新的工具，這類的整合較不具備延伸性，侷限了系統功能的彈性。因此程式開發人員經由元件化的概念，將系統中重要的部分設計成可被重覆使用的元件，這些元件可以任意地加入至系統中工作，系統會依加入的元件不同而改變其功能性， *S.Fischer et al*, BioWidget 1999 [7]、*A.Siepel* et al*, ISYS 2001, [8] 都是屬於這類型的整合式系統。僅管元件式整合系統具備了延伸性與彈性，但只有熟悉資訊領域的程式設計人員才有足夠的能力增減系統的元件，並不適合於一般生物資訊研究人員所使用。

不僅是資料型態之間存在著複雜的關係，同樣的問題也存在於不同的資料分析工具之間。現今在生物資訊領域中有太多不同的資料型態和分析工具，生物學

家難以同時了解所有資料形態以及其與不同分析工具之間的關係。雖然目前的整合系統基本上須要具有延伸性，能夠自由地加入新的工具或移除不需要的元件，有彈性地建構個人化的工作環境。但是當使用者漸漸的擴大系統的規模，系統已加入了各種不同的資料和工具，由於資料具有不同的資料型態，且不同的分析工具相互又具有關聯性，漸漸系統的複雜度提高，也增加了使用上的困難。因此，一個能夠有效率地幫助使用者去管理不同的資料型態和分析工具的整合系統是當務之急。

有鑑於此，傳統的生物資訊整合系統已經不敷使用，因為傳統的整合系統雖然已具備了整合性和延伸性，只是一昧的整合這此資源，對使用者在使用系統上是一大負擔。理想的系統必須具有整合分散在各處的生物資訊資源，且可以提供一個使用者介面，讓使用者可以很有彈性地組織自己的系統，且進一步能夠引導使用者去操作不同的資料分析工具。

本論文提出一個彈性化整合平台，透過簡單的系統設定與規則可將不同的工具組合起來，成為一個新的工具，且使用者不必撰寫冗長的程式，就可以動態地組識個人化工具，讓使用者可依個人喜好或研究問題領域的特性，建構出有助於資料分析與個人操作習慣相符合的工作環境，使得系統可以適用於不同研究領域的使用者。為了更進一步輔盼使用者駕馭不同工具甚至智慧地分析不同的資料，我們使用Ontology[1]的概念來描述資料型態和分析工具之間的關係。Ontology為一種針對一個特定範疇的知識加以描述定義的規格，它將範疇中的資源或名詞概念化 (conceptual) ，並利用概念的屬性來敘述概念與概念之間的關係，稱為relation。利用Ontology將概念規格化的特性，將資料型態和分析工具之間的關係表達出來。系統可以藉由Ontology去維護生物資訊資源的相依性，進而適當地提供使用者操作不同的分析模組，處理不同型態的資料。此外，Ontology還可描述

[1] Ontology原本是屬於文學的專有名詞，中文翻譯為「本體論」，現今將Ontology引用至知識科學中，已改變了原本Ontology的功能，因此本論文將使用「Ontology」來敘述，而不使用「本體論」。

使用者在執行資料探勘時所必需使用的一些動作或指令，使得當使用者在使資料分析的時候，系統將會針對使用者執行特定動作或指令表現特定反應，輔助使用者導覽資料。本論文將這個彈性化整合平台概念，實作一個系統雛型，稱作ODEX (Ontology Directed data EXploration)。

下一章將先對過去生物資訊在整合性系統的發展和現況做背景的簡介，了解目前整合性系統的趨勢和特性，之後第三章再針對過去系統的分析，提出新的概念和想法，在第四章我們描述服務導向整合平台的設計的概念和架構，進而第五章提出 ontology 驅動之資料探勘模式，並在第六章描述比較詳細的設計和實作的方法，第七章提供一些範例描述使用者如何建構個人化的系統，而最後一章對本篇論文做個總結以及日後還需要努力的方向。

第二章、文獻探討

第一節 整合式系統

由於生物資訊的技術快速發展，科學家可在短時間內產生大量的實驗數據，這些初始數據，通常必須經由一定的程序分析才可以獲得實驗的結果，以生物晶片為例，早期的生物學家在處理生物晶片實驗後，透過掃描機分別掃描晶片上綠色和紅色螢光劑顯影的強弱，就會獲得一張張的陣列影像，影像中每一個點(Spot)就是一個樣本，而每個點的顏色就相對地表示此樣本雜交反應的強弱。當生物學家獲得一堆影像之後，還需要經過濾雜訊、量化、正規化、修正錯誤值、或填滿遺漏值…等等一些前處理的步驟，經過這些處理後的資料才具有正確性，才能透過資料探勘的方法找出有利於研究的結果才會有公信力，倘若缺乏這些前處理的動作，直接將資料拿來分析，如此分析出來的結論並不足以讓人信服。資料前處理就可以視每一個步驟都是交給一個工具來計算，計算後的結果再交給下一個工具或程式繼續處理，直到最後得到分析的結果，就交給圖像具現化(visualization)的工具，將結果呈現給使用者。簡言之，生物學家面臨的問題包含：1、不同的工具都有特定目的。2、且不同工具之間往往具有輸入輸出關係，或是合作關係。基於這兩點理由，整合式系統概念的提出，目的就是將一堆合作性的工具集合起來，讓研究人員可以便利地操作適當的工具。

第二節 API 層次整合式系統

傳統的整合性系統是將一整套特定用途的工具，利用程式的 API 將原本各自獨立的工具組合起來，使得工具和工具之間透過特定的 API 相互溝通，傳遞適當的資料，我們統稱這類的整合為 API 層次的整合 (API-level Integration) ，此類整合系統通常是為了解決某個流程的問題。由於是在 API 層次的整合，工具之間

的相依性會因 API 的設計而不同，若 API 設計過於特殊，會照成相依性增加，使得系統不易變動，造成系統的延展性不足，系統無法因不同使用者的需求擴充系統的功能，大大的侷限了系統功能的彈性，使得此類系統大多僅用來處理特定方向的問題。為了提高系統的擴充性，增加系統功能的彈性，系統開發人員提出了元件化基礎整合式系統的概念。

第三節 元件化基礎整合式系統

將有用的工具或程式包裝成一個個的元件，每一個元件有如積木一般透過共同的介面與其他元件組合成為一個新的元件，而系統就是藉由這種組裝的方式建構而成，我們稱這類的系統為以”元件化為基礎” (Component-based) 的整合式系統。使用元件化的方式使系統的功能更有彈性，使用者可依需要增加新的元件至系統中，間接的增加了系統的功能。每一個元件都具有重覆使用性，減少不必要的資洲重覆。由於這些特性，因此元件化成為整合性系統一個必要條件。在生物資訊中，已經許多現存的系統應用了元件化基礎整合的概念，例如：*S.Fischer et al* BioWidget 1999 [7]、*A.Siepel* et al* ISYS 2001 [8]。

BioWidget 是提供一堆可被重覆利用的使用者介面元件並且提供一個架構讓使用者可以利用這些元件件組裝成一個應用程式。強調視象化在生物資訊中是相當重要的，若能顯示出適當的資訊，有助在研究人員研發的速度和正確性，所以 BioWidget 發展了一些在基因比對中常用的視象化元件，使用者可依個人需求組合，使呈現的資訊更有意義。

BioWidget 利用 JavaBeans 的架構為基礎，所以使用者可以用物件封裝來組合不同的元件，而形成全新的客戶端應用程式。BioWidget 解決資源異質性的方法是利用模組封裝的概念，所以資料可以經由封裝的程式介面來互相溝通。但是由於整個架構中，元件之間的關係都是以事件驅動的模式，所以當系統完成後就

很難再加以更改元件之間的關係，所以使用者僅限於在程式設計專精的生物學家或在生物資訊方面有經驗的研究人員。

ISYS 之系統架構如下圖 1，主要的特色是利用一個 Event Channel，類似匯流排的機制，所有的元件將事件丟到 Event Channel 上，而其他元件將會監聽這個 channel 上的事件，若是要交給自己處理的，就會將事件抓下來處理，並執行適當的反應和結果，或引發另一個事件再將事件丟回 channel。

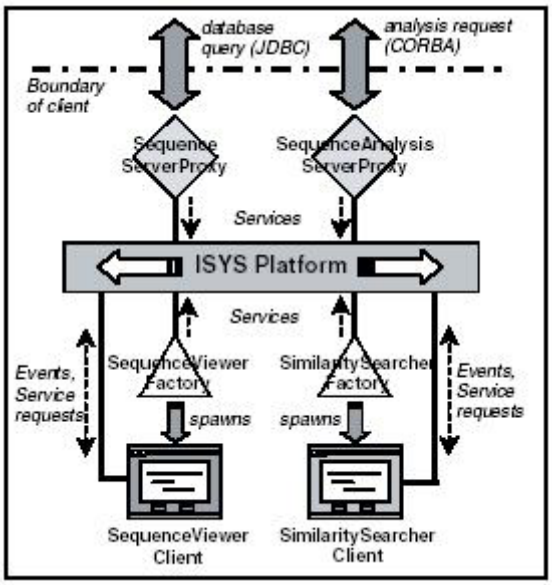


圖 1　ISYS 之系統架構圖(截取自 *A. Siepel, et al*. ISYS: a decentralized, component-based approach to the integration of heterogeneous bioinformtics resources. Bioinformatics, Vol. 17 no. 1, 2001, p 83-94.)

所有元件都是透過 Event Channel 這個介面去交換訊息，所以將各個元件之間的相依性降低。系統還提供了一個 Dynamic Discovery 的機制，可以用來判斷某個服務是否可以處理目前執行服務的輸出結果，如果可以，系統將會提供此服務給使用者，反之，則不會將此服務列入選單中。所以當一個服務要加入到系統時，必需要 Dynamic Discovery 註冊，定義與其他服務之間的關係，如此系統才可以針對使用者的要求做出適當的回應。

上述的例子，利用元件化的方法把相關的工具集中，使用目前已開發的整合性平台，如：CORBA、JAVA-Bean…等，或是依系統需求開發自己的平台，如：ISYS。其目的都是要讓不同的元件之間有個溝通的管道，透過這個管道，各個元件之間可以交換訊息，解決不同元件的異質性。但是隨著生物資訊的問題難度增加，需要的分析工具和資料型態也相對成長，整合進系統的元件愈來愈多，系統提供的功能也愈來愈複雜。系統變成僅適用於系統開發者，或是熟悉系統架構的人，一般生物學家難以同時了解所有資料形態以及其與不同分析工具之間的關係，使得過度的整合反而帶給使用者操作上的困擾。

第四節 進階元件化整合式系統：Ontology-Based 系統

為了提昇系統的控制機制以及使用上的便利性，以解決前述有關系統繁雜度的問題，一種方法是將本體論 (Ontology) 概念引入到知識科學的領域中，使用 Ontology 來幫助維護生物資訊範疇的知識，並利用 Ontology 所維護的知識，輔助使用者在操作系統時，能讓使用者直覺地以生物學家的角度操作每一個元件，本節將會先簡單的介紹本 Ontology 的概念，並討論幾個目前應用 Ontology 的系統。

2.4.1 Ontology 之簡介與探討

史丹佛大學知識系統實驗室[9]表示一個 Ontology 是針對某個範疇所明確定義的規格形式。也就是說 Ontology 會依應用的領域不同而有所改變。Ontology 明確定義出一個特定範疇中所可能出現的概念及其相互之間的關係。因此一個 Ontology 是由某個特定範疇中的概念所組合而成，概念中定義了屬性，屬性可能是此概念所具有的特性，或是與其他概念之間的關係，經由這些關係，Ontology 構成了一個有系統的網狀或樹狀結構，且概念的連結具有方向性，用來明確定義出關係的主體和受體(相當於英文中的主詞和受詞)。

組成一個 ontology 的主要成員包含概念 (Concept) 、關係 (Relation) 、及實體 (Instance) 。概念表示在描述域中的一個集合或一類的實體或事物。例如：蛋白質即為分子生物學這個領域中的一個概念。關係表示概念與概念之間的互動情形，也可以透過另一個概念來描述某個概念的特性。在 Ontology 中所描述的關係大致可以分成兩類[9]，第一類是用來描述分類的關係，當需要清楚的敘述階層式的架構，就可以使用這類的關係，通常使用 is_kinds_of 來表示繼承或子概念的關係，使用 is_part_of 表示子部分的關係，套用這兩個關係就可以把描述域中的概念加以分類，讓使用者更容易了解整個描述域的架構；第二類是用來描述聯集的關係，這類的關係很廣泛，可以描述概念的名稱、特徵…等獨特的屬性，有別於第一類的關係是用來敘述階層式架構，第二類聯集的關係超越了階層式的描述，提供一個自由的空間讓設計者可以依實際描述域的需求，適當的加入聯集的關係，有助於 ontology 的完整性。關係除了具有功能性的不同，還有不同的關係與概念之間也有強弱程度的不同，有些關係在概念中是必要不可缺的，稱為必要關係，舉例來說一個蛋白質在資料庫中都必須要有一個存取碼，所以蛋白質必需要有”具有存取碼”的關係，且此關係連結的概念就是”存取碼”；有的關係是可有可無的，舉例來說酵素具有輔因子(cofactor)，所有酵素有”具有輔因子”的關係，但是並不表示所有的酵素都具有輔因子；有時侯需要對關係要連結的概念加以限制，才可以確定關係可以正確地描述，當描述蛋白質具有受體的功能(Protein hasFunction Receptor)，表示蛋白質的“具有…的功能”關係僅接受“受體”這類的生物機能，但若僅描述蛋白質具有某功能(Protein has_Function)，此時並沒有指定該功能是什麼類別，表示任何概念都可以當成蛋白質的功能，並不適用於正確的概念。實體表示利用特定概念所描述的具現化物體，在 ontology 中通常是不包含實體，因為 ontology 通常只是用來定義描述域中的概念，所以當 ontology 中描述了實體，就變成了一般所認定的知識庫(knowledge base)，然而要決定一個東西是不是屬於一個概念的實體並不容易，這與 ontology 所應用的領域有關。

在生物資訊領域中，利用 Ontology 來描述生物領域的資訊，有助於生物學家在闡述生物現象或分子功能時，有個共通的概念或術語，以便於研究人員在閱讀相關文獻或分享資訊時，可以有相同的認知和共同的概念。比較常被提到的生物資訊 Ontology，如：Gene Ontology [10]，底下將會對 Gene Ontology 做簡單的介紹。

2.4.2 Gene Ontology 之簡介

Gene Ontology試圖利用ontology記錄生物組織中基因產物在生物機能中所扮演的角色之相關資訊，將基因功能加以分類，再針對不同的分類繼續細分，儘可能地清楚的描述這方面的問題領域。一開始Gene Ontology的建構只利用Flybase的資料庫來描述果繩的基因功能，但是目前已經延伸到包含老鼠 (MGD Mouse Genome Database) 、酵母菌 (Saccharomyces Genome Database) 和基因表現資料庫，日後也會繼續擴展Gene Ontology所囊括的領域，以增加Gene Ontology的功能。Gene Ontology主要發展出三個重要的結構，用來敘述分別相對於在生物程序 (biological process) 、細胞組織 (cellular component) 、和分子功能 (molecular functions) 的基因生成物[2] (gene product) 。Ontology善用了ontology的基本特性，針對上述的範疇清楚的描繪出所有可能出現的概念，並且加以解釋或用屬性來強調概念中的特色，所以Gene Ontology可以當成是一個專門敘述註釋基因產物、生物程序、分子功能這些方面的控制詞彙來使用。Gene Ontology使用了多重繼承的概念，利用了“is a kind of”的關係描述了概念之間的階層架構，同樣的也使用了“is part of”的關係描述概念與概念之間的包含關係，當然還有其他特殊的關係同樣的定義在特定的概念中。

2.4.3 描述 Ontology 的語言

[2] 基因生成物(gene product)：在Gene Ontology中，基因生成物可能具有一個或一個以上的分子功能，基因生成物所描述範圍相當廣泛，簡單的可以像是紅血球具有α血紅蛋白和β血紅蛋白這兩個基因生成物，以及小分子血紅素；複雜的可為很多基因生成物的合成物，例如：核醣小體。

由於 Ontology 的概念日益被大家所重視，也漸漸被利用在不同的研究領域上，為了提供一個標準讓使用者可以正確的描述 Ontology，所以 W3C [15]也提出了一些規格，可以用來描述 Ontology 這種特殊的資料結構。透過這個統一的標準，使用者可以提供自己定義的 Ontology 給其他人使用，也可以把別人分享的 Ontology 合併到自己的 Ontology 中。Ontology 描述的語言從一開始的 RDF，再來為了提供更強的功能，提出了 DAML，再來是之後的 DAML+OIL，一直到最新提出的 OWL (Web Ontology Language) ，以下將會簡單地介紹 RDF 以及 OWL。

2.4.4 RDF (Resource Description Framework) 之簡介

目的是要描述所有在 www 上的資源的資訊，它是特別強針對網頁資源的描述資料的資料 (Metadata) ，例如：網頁的標題、作者、修改日期、等等。RDF 不僅僅是要把資料讓使用者易於閱讀，還強調可以讓不同的應用程式之間利用 RDF 交換訊息，因為 RDF 是使用一個共同的架構所建構出來的，所以可以在應該程式之間溝通而不會有訊息的遺失。RDF 的基本架構有三個重要的部份，subject、predicate、object，由這三個部份組成一個完整的敘述句，例如：

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
             xmlns:exterms="http://www.example.org/terms/">
   <rdf:Description rdf:about="http://bioserv.cis.nctu.edu.tw/index.html">
       <exterms:author>jimmy</exterms:author>
   </rdf:Description>
</rdf:RDF>
```

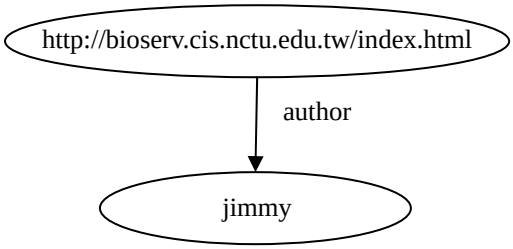


圖 2　簡單的 RDF 概念示意圖。

上面的例子是在描述 bioserv.cis.nctu.edu.tw/index.html 這個網頁的作者是 jimmy，其中 "http://bioserv.cis.nctu.edu.tw/index.html" 就是為 subject，而 <exterms:author>為 predicate，最後 jimmy 則為 object。所以由這簡單的例子可以看出 RDF 已經可以提供描述各個資料之間關係的架構，但是在網路資源之間的關係並不完全是那麼簡單，需要其他特別的結構才可正確的描述繼承、聯集、交集.....等等，於是便提出新的 Web Ontology Language (OWL) [17]

2.4.5　OWL 之簡介

網路實體語言用於表示網路上文件中術語的意義和術語間的關係，可供應用程式處理文件資訊之用，為語意網路技術的一種。OWL 藉由提供額外的字彙與正規的語義來提供網路內容更具相容性。OWL 包含三種子語言，OWL Lite、OWL DL、OWL Full。OWL Lite 提供了使用者所需要地主要分類階層以及簡單的限制條件，舉例來說，當此提供了一個基數的限制，此時僅允許基數的值為 0 或 1。OWL DL 提供了使用者最大的表達空間但是仍保持計算邏輯的可完成性以及可決定性(表示所有的計算都可在有限的時間內完成)，OWL DL 包含 OWL 的語言中所有建構子，但是它們僅可在特定限制下被使用，例如，一個類別可以為許多類別的子類別，但是一個類別不可為另一個類別的實體。OWL FULL 有別於 OWL DL，它一樣提供了使用者可用的最大表達空間，此外增加了不具可計算確認的語義自由 RDF，舉例來設，一個在 OWL FULL 的類別，可以被視為一個獨立的集合，也可以是類別本身。

2.4.6 以 Ontology 為基礎的生物資訊系統

- *Carole A.Goble* , TAMBIS(The Transparent Access to Multiple Bioinformatics Information Source), 1999 [11] [12] [13]

  TAMBIS 是英國曼徹斯特大學的生物科技、資訊管理以及資訊科學三個系所的聯合研究計畫,是利用 ontology 來幫助生物學家在對其他資料庫執行查詢指令時,有一個共同的查尋介面,使用者在使用系統時,就感覺是針對一個資料庫系統做查尋,事實上是存取了不同資源的資料庫系統。TAMBIS ontology (TaO) 廣泛地敘述生物資訊領域的工作和資源 ,在 TAMBIS 中占非常重要的角色。使用者可藉由 ontology 的 concept 之間的關係,組成一個複雜、多個資源的詢問,例如:要查尋一個 protein 與某個個 protein 相似,而且具有特定的 accession number。經由如此與資料來源無關的查尋語言,使用者可以不用知道真正存放資料的到底是一個 oracle 的資料庫,還是一個檔案系統,TAMBIS 會將這種與真正資料庫系統無關的查尋語言,轉換成在資料庫系統真正可以執行的程式或在此資料庫系統可以使用的查尋語法,再交給真正的資料庫系統處理。這時使用者就好像很直接的存取到不同的生物資訊資料庫系統,給使用者有一種透明化存取多個生物資料庫的假象,只要用同一種 TaO 定義的查尋語法,透過 TaO 描述的生物資訊領域的工作和資源,可產生各式各樣不同的查尋結果。下頁圖 3 為 TAMBIS 的系統架構:

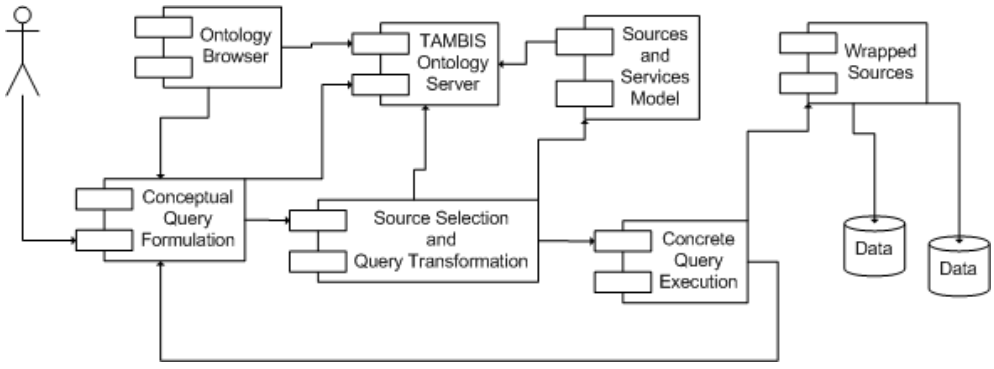


圖 3 TAMBIS 系統之架構圖

TAMBIS 強調出透過 ontology 的標準化，提供一個共同的介面，也就是 ontology，經由這個介面與不同的資料庫作溝通。對於使用者而言，只需要學會和 TAMBIS Ontology Server 溝通的語言，也就是 Conceptual Query Formulation，而系統會將藉由查尋找出適當的 Service Model，將 Ontology 的查尋語法轉換成真正資料庫可以執行的查尋語法，再交給真的資源提供者去執行，如此使用者面對的只是 TaO Server，不需要知道真正要如何去存取不同的資料庫。TAMBIS 具有一個中央集中化的 Ontology Server，所以用同一個 Ontology 去描述不同的資料庫所提供的資料，不會有因為由不同的資料提供者都具有自己的 Ontology，而造成語義的異質性的問題。

- *Russ B. et al*, RiboWeb, 1999, [14]

  RiboWeb 是一個以網頁介面操作的核醣體三度空間模型分析系統，主要用於幫助使用者易於建構核醣體三度空間模型並且與現存的研究結果作比對，利用已經被發表的研究成果來比對使用者輸入的資料，用一些比對的模組來計算之間的相似度，來推論未知核醣體結構的生物功能。要完成這個目標所需要的知識，RiboWeb 利用四個 ontology 來紀錄，physical-thing ontology 、data ontology 、reference ontology 、methods ontology。 Physical-thing ontology 是用來描述核醣體的構成要素和其他輔因子(cofactor)。 Data ontology 明確說明從生物實驗中所收集到的資料型態。Reference ontology 敘述已發表研究的型態。 Methods ontology 紀錄在 RiboWeb 中可執行的一些功能的型態，與需要執行這個功能所需要的主要參數。研究人員可將自己發表的核醣體 3D 結構模型加入至系統的知識庫中，當研究人員要了解自己發表的這個 3D 結構與現存已發佈的實驗結果的一致性，研究人員需要先從 method ontology 中選取適當 interpret 模組，它藉由使用者設成的參數，將資料解讀成可以被相對比較的單位，此時研究人員只要選擇自己有興趣的資料，設定一些執行此模組所需要的參數(可從 method ontology 中獲得哪些參數需要設定)，執行

interpret 模組，即可獲得資料在這個 interpret 模組的單位距離。研究人員再利用同樣的方法，對自己發表的 3D 結構模型執行相同的 interpret 模組，再使用 compare 模組比對剛剛有興趣的資料與自己發佈的模型之間的一致性，以幫助了解新發表的這個核醣體 3D 結構模型的可性度和正確性。

由此可看出 RiboWeb 是一種 Method-Oriented 的生物資訊整合系統。研究人員自行判斷分析資料的方法，然後指定系統執行特定的資料分析程序，系統會依照知識庫中描述分析程序所需要的參數或資料，在資料庫中選取出來，再由使用者判斷自己需要的部分來執行分析動作，本論文將之歸類為模組導向資料分析系統 (Method-Oriented Data Analysis System) 。反之，若是研究人員希望針對某個型態的資料做分析，但是卻不知道有什麼樣的分析方法可以使用，則系統的知識庫中將會描述分析工具的輸入資料型態，系統就可以判斷研究人員指定的資料型態，並提供適當的模組給研究人員參考和執行，本論文將之歸類為資料型態導向資料分析系統 (Data Type-Oriented Data Analysis System) 。

由上述之討論可歸納出在生物資訊的領域中，資料型態和分析方法之間的關係相當繁雜，研究人員往往具備豐富生物資訊方面的知識，和大量的實驗數據，資料中隱藏了人眼看不出來的有趣資訊，為了能夠把這些隱藏的資訊找出來，資料型態導向資料分析系統就比模組導向資料分析系統較滿足生物資訊研究人員的需求，研究人員僅需要將自己的實驗數據做正確的判斷，判斷出資料的型態和特性，就可以直覺地針對這些資料執行分析動作。雖然僅僅是系統著眼點的不同，但是對於使用者來說，從資料型態的角度判斷適合的分析方法，輔助使用者更直覺的操作流程。

第三章 問題歸納、目標與方法

第一節 問題歸納

不同的資料型態以及資料之間的關係繁雜，一直是生物資訊系統需要考量的問題，僅管目前已有許多描述生物資訊領域的知識庫被提出，但就使用者操作系統的角度而言，不同的工具需要什麼樣的資料型態，才會最為重要的。然而工具隨著不同的組織各自研發，所需要的資料型態也不盡相同，研究人員難以同時了解所有資料形態以及其與不同分析工具之間的關係，如此工具之間的異質性，往往造成研究上的阻礙。

傳統解決工具的異質性是仰賴整合式系統的開發，將相關性高的工具集中至同一個系統中，且提供既定的控制介面，使用者僅能透過有限的操作介面，以及顯示的資訊，找尋適合需求的操作模式。然而不同領域的研究人員對於系統的需求也不同，因此要順應不同需求的系統，就必需具備延伸性，而元件化整合式系統提供了延伸性，但也僅侷限於程式層次的整合。

以系統整合的角度來說，從元件化整合至加入知識領域的整合系統，一直希望能夠提供有彈性的整合機制，使得系統可以任意的建構適合個人的工作環境。從使用者的角度，能透過簡單的設定，就可以完成元件的變動是最恰當的。然而目前生物資訊整合系統仍是在程式層次的組合，僅適用於資訊領域的研究人員，使得若要變更系統中的元件，就需要仰賴資訊域域的研究人員來代為建置。如此缺乏即時性的操作，帶給使用者相當的困擾。

此外，使用者在操作系統時，往往是透過使用者介面的資訊來判斷下一個該執行的動作。若能將這些動作描述在系統中，且提供某種程式客製化的介面，如此使用者就可以設計屬於個人風格的操作摸式，也可以適用於研究領域。藉由簡單的設定，有助於往後的操作都可以循著既定的模式完成工作。

第二節 目標與方法

本論文將提出一個以 Ontology 驅動之彈性化整合系統，稱為 ODEX 系統 (Ontology-Driven Data Exploration)，為了讓系統具備合作與資源分享的特性，本系統將建立在一個以服務組合的基礎架構上，目的是將各式各樣的資源都可以用服務的形式分享給不同的客戶端。

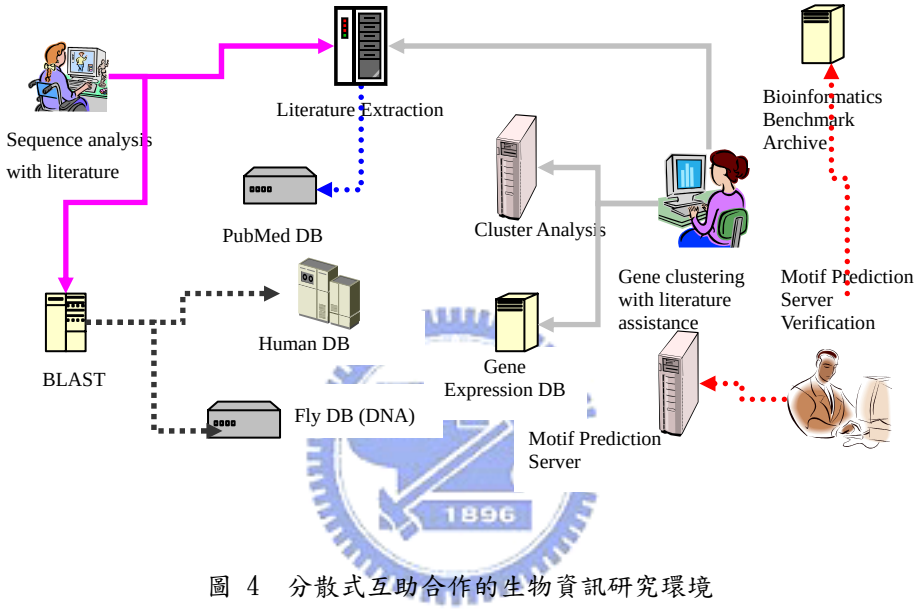


圖 4 分散式互助合作的生物資訊研究環境

透過 ODEX 的架構，希望能提供生物資訊研究一個分散式地、以網路為基礎、共同合作、並能多領域相輔相呈的作業環境。如上圖 4 所示，不同的資料型態和不同的工具類別分別在各自的機器上運作，當這些資料和工具加入到系統環境中，就可以提供出特別的服務分享給其他加入環境的使用者，使用者也可以依個人的需要組合適合研究方向或問題領域的服務，有助於研究人員在操作工具時的便利性，和分析研究數據的效率。

為了能夠滿足不同層級或不同領域的使用者，ODEX 提供了不同的整合機制，從最基本的加入元件來豐富系統的功能，至較進階的經由簡易的設定使得相關的元件共同合作。目的是為了能讓使用者依個人的狀況及需要選擇適合的整合

機制，對於系統架構和設計比較了解的使用者，可自行撰寫程式將不同的元件組合，反之，則可以透過較簡單的系統設定，達到整合的功能。

最基本的整合，僅將新的元件加入至系統中。新的元件被包裝成服務放入服務容器中，此時服務容器有如服務管理員，當有要求 (Request) 發生時，服務管理員就會將要求導向給正確服務處理。若要將多個元件組合成新的元件，就必需使用者自行撰寫程式將元件組合，這個方法讓使用者有最大的彈性去建構個人化的服務，但它適用於對於程式開發，或元件本身相當熟悉的使用者，一般的使用者無法透過這類的步驟來完成元件的整合。

若使用者已經了解完整的操作流程，對於分析工具的使用也有經驗，使用者可以利用系統中已預設的基本可執行的動作，例如：複製、貼上、執行、開啟.....等，一步步的將資料在各個服務或工具之間流動。系統僅僅提供一些基本的操作，使用者藉由一連串的基本操作，將不同的工具的服務整合在一起互相合作。使用者有很大的自由度去設計並執行個人化的程序，以符合個人研究的需求。

除了系統預設的一些基本的操作之外，使用者也可以透過設定規則，把個人喜好的流程，或是經驗上常常需要使用到的操作程序，用規則描述出來，當系統執行時就會依設定的規則反應適當的動作。因為使用者可以依個人研究領域的不同，定義出相對應有趣的資訊，所以這不僅僅提供了一個較簡單的整合機制，還容許一定程度的客製化，讓系統不會因研究領域的束縛，而造成不必要的限制。

此外，針對相關性比較密切的使用者介面工具，ODEX 提供樣版式整合 (Template Integration)，是將一些常用的排版版面製成樣版，使用者可以選擇適當的樣版，將工具一一配置到樣版中，就可以組合成新的工具。使用者也可以利用系統本身提供的基本樣版，組成較複雜的排版版面。樣版整合不僅僅可以將排版規格化，使用者可以定義屬於基本樣版的導覽規則，只要套用這個樣版，新的工

具就會依此規則處理特定的操作，使得系統的所有工具都具有特定的排版和操作風格，有助於使用者在不同工具之間轉換運用時更容易上手。

具備了上述的整合環境，以及規則客製化的機制，進而加入 ontology 知識庫的概念。藉由規則與 ontology 配合，由於 ontology 中維護了豐富的知識庫，使得系統可藉由知識庫的存取和查尋，提供更強大的規則以達到更靈活的系統行為模式。

第四章、服務導向之整合環境

第一節 服務導向之環境架構

ODEX 為本論文所要提出的系統架構，下圖(圖 5)為 ODEX 系統使用層次示意圖，基本上系統是透過一個系統導覽員(Explorer)的使用者介面做為工作平台，將圖形化介面工具整合起來。而在圖形化介面的後端存在著實際運算的邏輯服務，負責提供圖形化介面工具在顯示時所需要的資料。

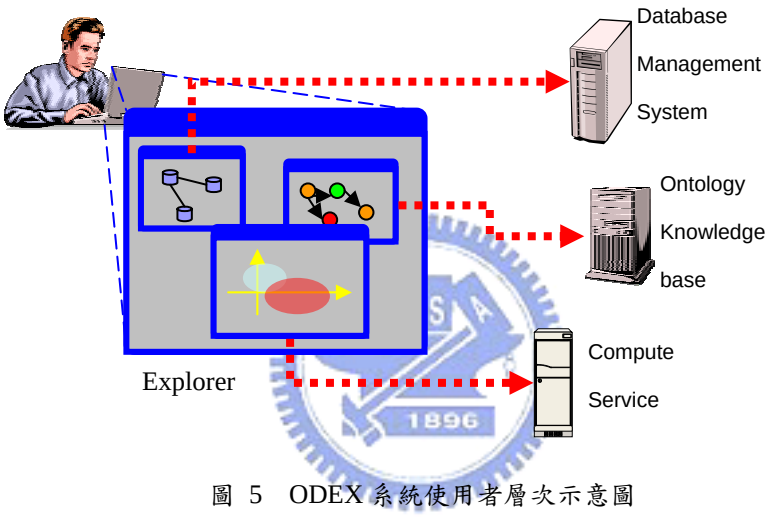


圖 5 ODEX 系統使用者層次示意圖

在系統中視每一個圖形化介面工具為一個服務，因此系統導覽員也是系統中的一個服務工具。ODEX 系統將服務放置服務容器 (Service Container) 中，交給主機 (Host) 控制以及維護 (如下頁圖 6)，本機的服務容器中可以管理屬於圖形化介面的服務工具，並透過服務整合的機制，將不同的工具利用系統導覽員整合起來。其他的後端邏輯運算的服務，也交由服務容器管理。服務可透過服務容器提供的介面，與其他服務交換訊息，讓本機的圖形化介面工具可以與遠端的邏輯服務建立連結，使得位在不同位址機器上的服務，都可以經由服務組成的機制互相的共享服務資源。

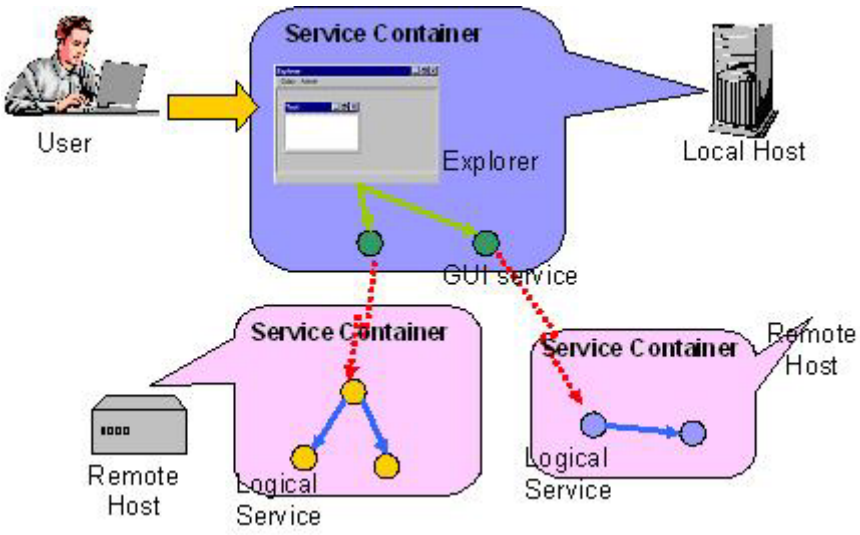


圖 6　以服務容器組成的 ODEX 系統架構圖

為了讓各個服務容器所提供的服務資源可以被有效的利用，ODEX 系統將服務的資訊與服務的型態交給 Domain 集中管理，Domain 利用服務的型態將各個服務有效的分類，提供適當的查尋介面，讓使用者可以經由型態獲得適當的服務，如此就可以加入規則至系統中，定義搜尋服務型態的規則，透過彈性化的規則將服務組合，使系統具備動態的整合功能。

為了進一步提昇系統的靈活性，將相同功能的規則集中成為規則集合，且透過規則驅動引擎服務將不同功能的規則集合集中維護，再藉由參考 ontology 知識庫中的資訊，驅動規則依不同情況執行不同的判斷與反應，有助於使用者可建構符合個人化的工作環境和風格。

圖 7 描述的是一個服務導向基礎架構，而架構中相互合作的服務之間所溝通用的通訊協定是透過 XML 為基礎的訊息。服務(Service)是架構中主要的成員，而 Domain 是一個服務註冊員，用來紀錄和維護服務的型態和服務實體的相關資訊。這個基礎架構與現存由 SOAP、WSDL、和 UDDI (Universal Description Discovery and Integration)所組成的 Web Service 架構標準相似，其中的差異就在於本論本所提出的架構中具有 Host，這裡的 Host 與過去認知的中央集中式網路

架構中扮演主機角色的 Host 有所不同，這裡的 Host 被當成是一些服務的服務容器(service container)，目的是希望在網路的環境中不會有一個中央集中管理的地方，且 Host 僅是一個可以被存放並且管理各別服務的地方。因此這個 Host 可以是任何一台主機，也可以是本機上的一個硬碟空間，使用者可以透過 Host 提供的介面，取得需要的服務。

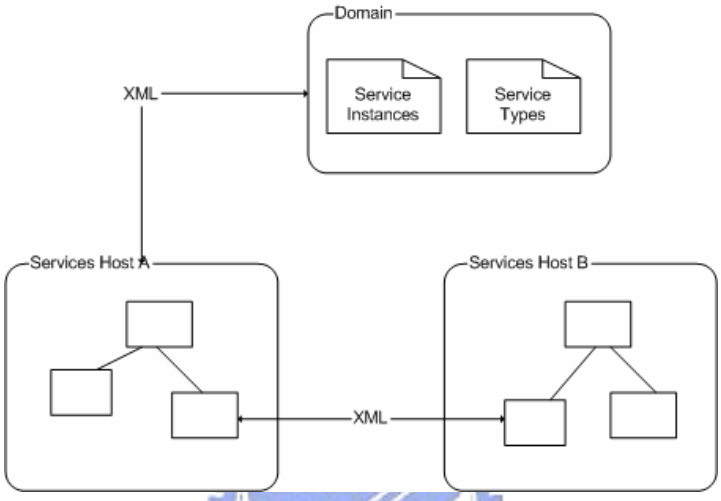


圖 7　XML 為基礎之服務導向基礎架構

為了達到上述的服務導向環境，底下將針對溝通協定、服務容器的建構、服務的管理、服務註冊等重要的概念加以描述，進一步提出以規則為驅動的系統架構。

第二節　以 XML 為基礎之溝通協定

本架構使用 XML 做為基礎的溝通語言，將所有的系統設定、服務設定、以及服務之間的溝通介面，都是利用 XML 標準。因為有了共同的溝通協定，使得系統在建構時可以定義規則，透過規則的比對，來控制系統的運作。如此系統不僅僅是單純的將服務整合起來，而且可以藉由靈活性的規則設定，使得服務之間的溝通更加有彈性，進而還可以提供具有搜尋功能的規則，讓系統可以尋找特定條件的服務，有助於減少服務之間不必要的相依性，以及增加系統的延展性。

第三節 服務容器的建構

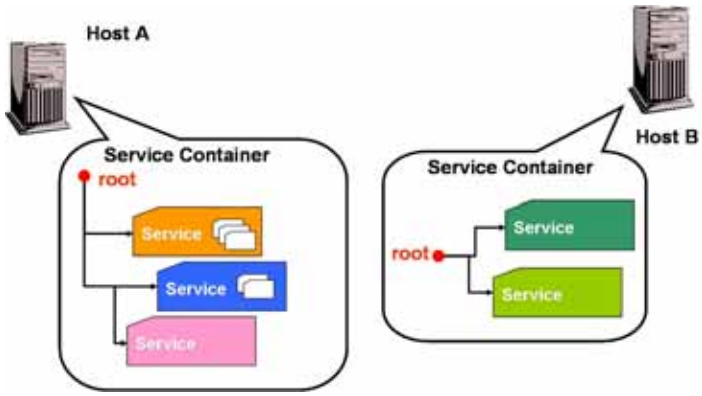
分散在網路上的主機具有服務容器的功能，我們通常將有使用者介面的元件稱為工具，僅負責計算和分析的元件稱為服務，而本機的工具容器也不一定只存放負責使用者介面的元件，有些計算的元件也可以放置在本機的工具容器中，因此在這裡將把這兩個名詞統一，在此之後所以系統的元件都將稱為「服務」，而不再另外解釋。這一節中將大致描述服務容器的建構方式，與每一個服務的設定。

在服務容器中，會將服務分類放置在不同的路徑下 (如下頁圖 8(a))，當然每一個服務必需具有唯一路徑，而要如何分類，就可以依使用者個人需求而設定。基本上服務容器中具有預設的系統服務，負責系統相關設定與修改，再者就是使用者自己所需要的其他服務。而每一個服務將在服務容器被建構起來時被紀錄在服務容器中，當第一個要求被傳送至服務容器時，才會將服務實體化。

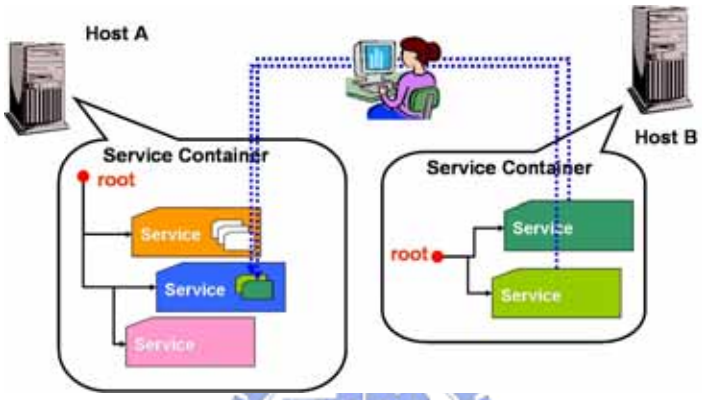
有些服務必需在建構前使用其他的服務，此時就要設定「脈絡服務」 (如下頁圖 8(b))，所謂的「脈絡服務」就是當服務在運作時，必需仰賴其他服務的幫助，這時就要在此服務被使用之前，將所有的需要仰賴的服務配置完備，才可以讓該服務正常運作，否則此服務將無法被使用，所被仰賴的服務我們統稱為「脈絡服務」。底下為設定服務的範例，

```
<tool>
    <create class="odex.tool.kmeans.gui.KmeansTool"/>
    <cntx path="../host/kmeans" name="kmeansService"/>
</tool
```

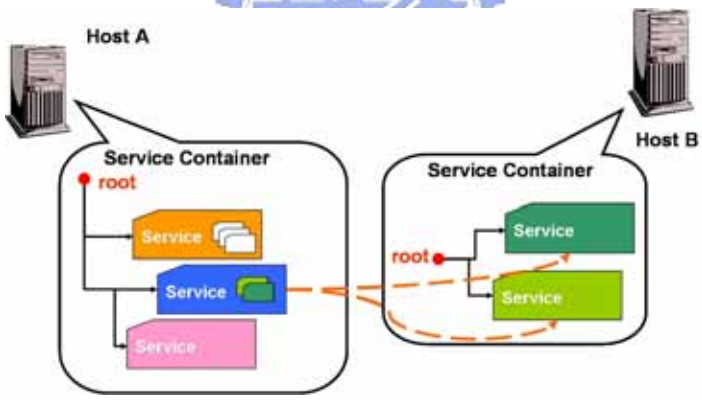
由上述的範例描述一個 Kmeans 的結果顯示服務，需要脈絡服務為一個 Kmeans 計算服務，此服務是位於服務容器中”../host/kmeans”這個路徑下，服務容器就會找到些服務，並將之設定給 Kmeans 的顯示務。



(a)



(b)



(c)

圖 8　服務建構示意圖。在 Host 中的服務是以適當路徑加以管理。而每一個服務依需求會要求在建構時設定需要的服務。(a)此為兩個 Host，每個 Host 都是自己所維護的服務容器。(b)使用者在建構自己的服務時，可依需求填入適當的前置服務，使得每個服務都可互相分享使用。(c)當使用者將前置服務配置完成，則服務間就間接地建立了一條無形的關係，將服務們串連起來。

第四節 服務容器的管理

在架構中，最基本的單位就是「服務」，任何人都可以提供服務，把服務放在網際網路上分享給其他人使用。因此，架構中利用一個服務容器管理員，稱為 Host (圖 9) ，將這些服務分類管理，當有需求的時候，特定的服務將會被啟動。

此 Host 並不是將分散在網路上所有的服務集中起來管理，它僅僅是扮演一個服務容器的角色，將本機所分享給他人使用的服務，同階層式結構的方式，適當的管理，且每一個服務都有一個唯一的路徑與之對應，此時 Host 才可以透過正確的路徑存取服務。 當服務要被提供給其他人使用時，必須將設定檔加入 Host 的適當路徑中，建構檔中必需描述當服務被產生時所需要的設定，Host 就可以利用這個建構檔將服務實體化並且發佈在網路上供其他服務使用。下圖 (圖 10) 為 Host 的控制介面，圖中可看出每一個服務都有其建構檔，且可以透過這個控制介面修改並重新發佈。

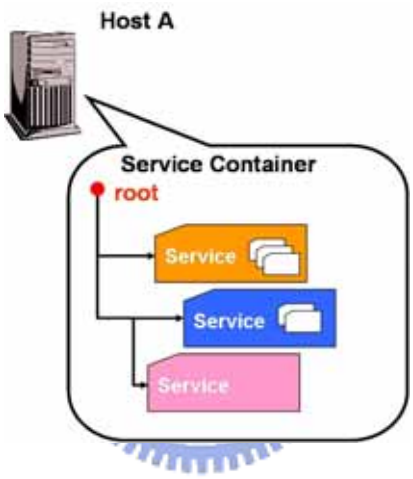


圖 9　Host 之設計示意圖

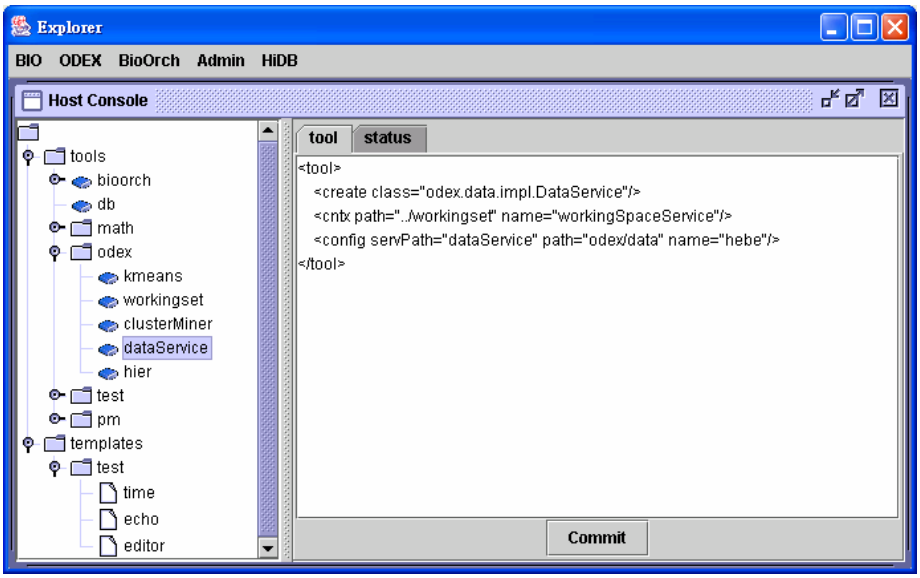


圖 10　Host 控制介面。每一個服務都用一個 XML 設定檔，描述服務執行時所需要的其他服務，以及相關設定。

第五節 服務註冊

為了讓分散網路的服務能夠被更有效地利用，我們將具有一個集中管理服務資訊的地方，稱為Domain。本架構下的Domain與目前標準的Web Service架構中的UDDI具有類似的功能，希望提供一組標準的規範用於描述和發現服務，使得不同的使用者或組織單位可以經由一套適應性強的服務描述，描述服務的相關訊息。本架構是以XML為標準，將所有的資訊都以XML的格式加以描述，特定的資訊必需符合系統規格，而Domain是將各個服務藉由服務實體 (Service Instance) 與服務型態 (Service Type) 這兩項資訊加以管理。服務實體中描述了服務的名稱、位址、以及相關設定，當使用端[3]與Domain要求服務時，Domain就會將服務實體透過XML溝通協定傳送給使用端，使用端即可利用XML中所描述的資訊直接與該服務溝通，不需再以Domain做為媒介。服務型態Domain用來將不同的服務做適當的分類，使得使用端可以依型態選擇適當的服務，此外，不同型態的服務具有不一樣的輸入和輸出介面，這些資訊將會被描述在說明文件中 (圖 11)，讓使用者可以依照設明文件定義的XML介面，正確的使用服務。透過服務型態的分類以及服務型態介面的定義與描述，使用端可以定義彈性化的規則，例如：尋找特定型態的服務，使得使用端所使用的服務不侷限於特定服務實體，且因為型態相同，所以可接受的輸入和輸出介面也相同，因此使用端不用更改程式就可以因改變使用不同的服務，而提供不一樣的效能或結果。

[3] 使用端：因為使用服務的不一定為客戶端，有時為另一個服務，所以統稱為使用端。

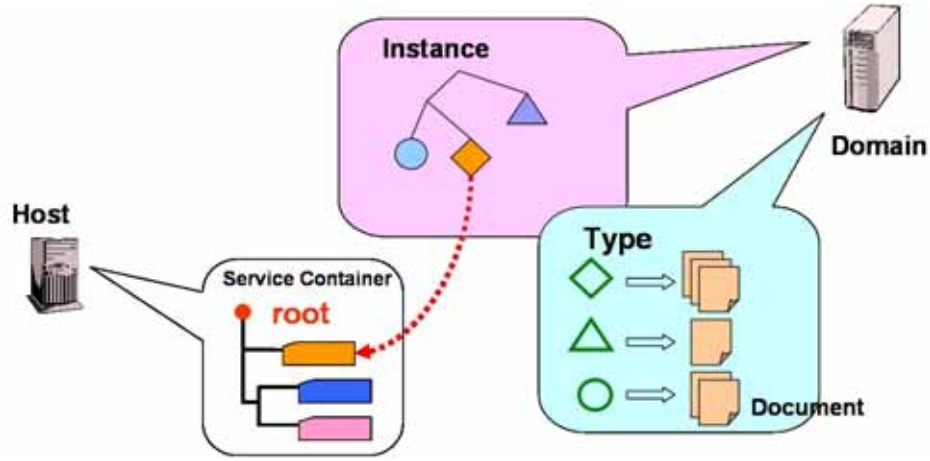


圖 11　Domain 架構與 Host 之關係圖

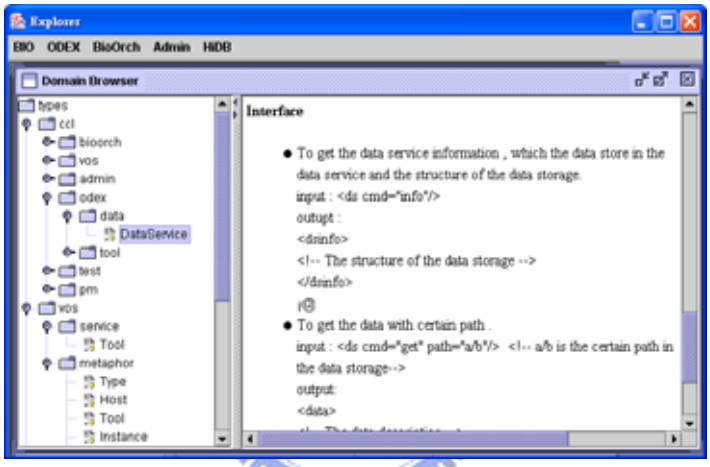


圖 12　Domain 瀏覽員。圖中指定了 odex / data / DataService 這個型態的服務，右邊顯示了此型態的服務所提供的 XML 介面。

當服務要跟 Domain 註冊時，必須將提供服務實體給 Domain 管理，服務實體中會描述服務型態，若 Domain 不具有該型態，則必需新增一個服務型態，且最好適當的撰寫說明文件，以便讓以後使用者可以清楚的了解服務的功能以及與其使用的方式。

第六節　以規則為基礎的組合模式

除了經由使用者設定達到服務整合的功能，透過進階的規則設定，則可以提供更有彈性的整合。藉由向規則驅動引擎提出要求，規則驅動引擎會依使用者定義的規則，做出適當的回應，如此就可以獲得其他服務的資訊，進而互相溝通傳遞訊息。

規則驅動引擎為 ODEX 重要的元件之一，因為 ODEX 的操作模式都將視規則驅動引擎判斷輸出，使用者可依個人的需求，撰寫符合個人需求的規則，當然若不需要 ODEX 來幫助導覽資料時，也可以完全不使用規則驅動引擎的功能，系統依然可以正常運作。所以規則驅動引擎主要的功能就是比對輸入的指令，交給適當的 Rule Set 處理，找出符合的規則並處理輸入然後回傳結果，若沒有符合的規則，則不做任何事。所以規則驅動引擎做的事情非常簡單，它完全是依照使用者撰寫的 Rule Sets 反應，也不知道規則本身代表什麼意思，僅僅只是輸入/輸出的機制。

舉例來說，如下頁圖 13，一開始服務會向規則驅動引擎要求另一個脈絡服務，而此脈絡服務必需符合特定的型態，此時規則驅動引擎的工作就是找出適當的服務並回應給需求端。此時規則驅動引擎會藉由詢問 Domain 的方式尋找符合條件的服務，當然 Domain 的基本功能就是可以藉由型態來查尋服務實體。此時 Domain 就會將服務實體的資訊傳回給規則驅動引擎，規則驅動引擎再將此資訊經過簡單的轉譯，再回應給需求端的服務，該服務就可以利用所獲得的服務實體資訊直接使用另一個服務。

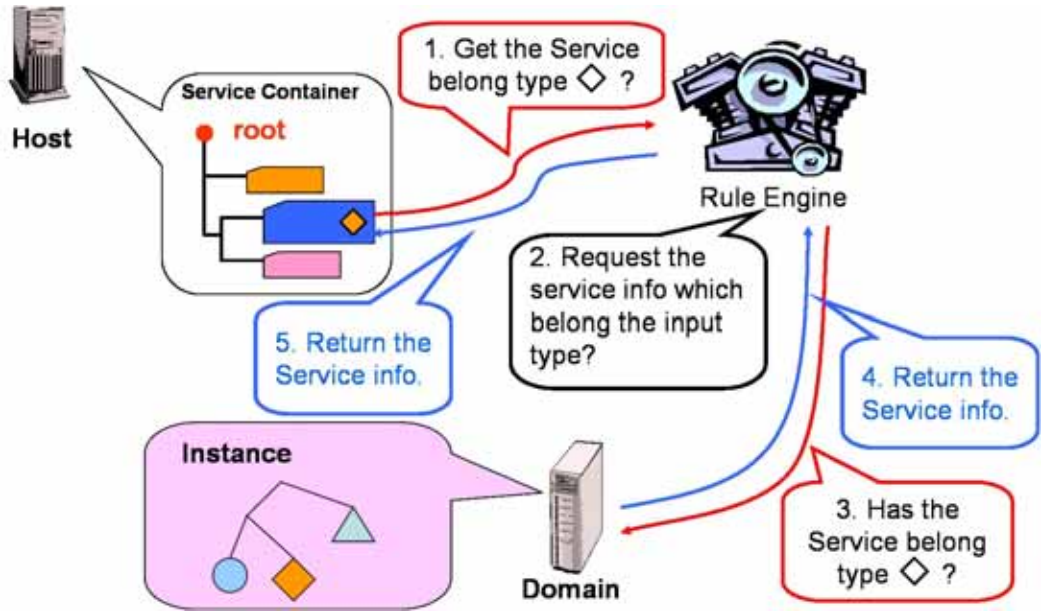


圖 13　規則基礎組合機制流程圖

第七節 本章總結

歸納本章所描述的系統架構，可得知 ODEX 主要的設計重點必需包含下列幾項特色：

1. ODEX 必需提出一個機制，而不是策略，以避免系統被侷限在特定的領域和方法。
2. ODEX 必需是元件式系統，所以需具備延伸性，當有新的資料型態和工具類別要被整合到系統中一起工作時，並不會因為這樣的擴充功能而導致過多的困擾。
3. 當工具被整合到 ODEX 的環境中時，必需很有彈性和具備高度的客製化，使得工具在操作上更適用於特定問題領域。換句話說，ODEX 必需提供一些排版的 scripting 機制，讓使用者可以將多個工具組合成一個新的工具，提供更適用的操作介面。
4. ODEX 必需是一個以使用者為中心的機制，幫助使用者分析並解讀生物資訊的各種現象，且提供一個彈性的環境讓不同領域或是不同經驗的使用者都可以依個人情況，調整適當的操作模式。

此外由於規則驅動引擎的加入，使得「規則集合客製化」成為系統重要的特性。研究人員在解決不一樣的問題時，有時需要不一樣的瀏覽資料的方式，針對問題的特性顯示適當的資訊，才可能有效率的解決問題。為了使 ODEX 的介面能依不同的領域、不同工具、不同的問題而有所改變。所以本論文提出用規則導向來驅動導覽員介面的方法，研究人員可以將自己理想的操作方法設定在規則集合中，則導覽員介面就會依使用者設定的動作正確的反應。規則導向引擎成了 ODEX 系統非常重要的角色，不過其實這個元件的功能很簡單，僅僅只是接收要求，然後處理後回傳回應而已。規則導向引擎需要有一個存放規則的地方，稱作為規則倉儲，規則倉儲會存放很多的規則集合，每一個規則集合通當都是負責相

關的要求，而規則倉儲會提供一個結構讓規則集合可以儲存起來，等到有要求進來時，規則導向引擎會將輸入的要求做初步的判斷，再把要求導向適當的規則集合處理。所以使用者可以依個人研究的需要，自行加入正確的規則至規則集合中，也可以自行增加具有特定功能的規則集合，如此可以改變 ODEX 在導覽資料或執行工具的行為。當然預設下，引擎本身有一些已經設定好的集合，這些集合關係到 ODEX 系統本身的運作，使用者最好不要去更改，但是如果使用者已經熟知規則的設定，當然就可以做適當的修改，以達到客製化的功能。

第五章、Ontology 驅動之資料探勘

由於生物資訊的廣泛繁雜，已有不少研究希望藉 Ontology 描述資料的特性來幫助管理系統內特殊的背景知識與系統控制，以上一章的服務導向環境架構為基礎下，ODEX 可進一步藉由 Ontology 的特性來輔助規則驅動引擎能提供進階的規則比對來強化系統的動態效能。

在使用 Ontology 之前，首先要建構 Ontology，有不同的方法去呈現或建構 ontology。依複雜的程度來區分，ontology 可以被分類從較簡單的資料庫綱要 (metadata)，如 TAMBIS [11] [12] [13]、描述特定領域中的詞彙，如 Gene Ontology [10]、至物件式結構和邏輯運算，如 DAML [17]、OWL [18]。ODEX 系統設計的目標之一就是要盡可能的避免不必要的限制。我們的 ontology 知識庫中允許不同的 ontology，每一個 ontology 都對應到一張圖 (Graph) 。各 ontology 中，node 就是一個概念，而每一個連結就代表概念之間的關係，且關係集合中需要一些預設的內建式關係，如"is_a"和"part_of"，有助於在 ontology 的建構與描述。

第一節 ODEX 的 Ontology 原型(Ontology Model)

在 ODEX 系統中將每一個 ontology 視為一個服務，所以建構在本章第一節所描述的服務組合基礎架構上的 ontology 服務，可以讓不同的 ontology 服務互相合作，使得 ontology 之間可以互相參考，達到知識分享的特性。ODEX 對於 ontology 的描述，將概念之間的連結，簡化至利用簡單的「關係」與「屬性」兩種敘述加以定義(圖 14)，不同領域都可以透過這個標準的資料表示方式來定義 ontology。

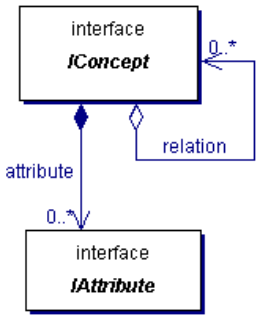
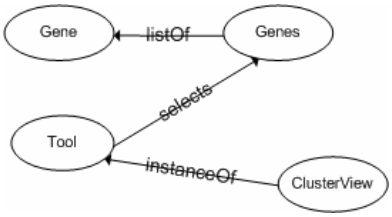


圖 14　ODEX Ontology 模型

雖然提供標準的資料表示法，但是實際上因為與每一個 ontology 溝通的介面是透過服務的介面，所以經由這個標準介面容許不同系統的 ontology 加入到 ODEX 的環境中。每個 ontology 的解譯和比對正確性的工作交給個別的解譯元件處理。藉由這個方法，使系統有辦法提供不同形式的 ontology。

下圖是一個簡單的範例，圖中有四個概念和三個關係，這些都可由使用者任意定義。然而，這個 ontology 在概念上正確與否則是交給其他驗証者來驗証(舉例來說，可以比對”isA”這個關係是否具有迴圈)。



ODEX 並不限制 ontology 所描述的領域，任何可被 ontology 資料結構所敘述出來的範疇都可以加入到 ODEX 的系統中。此外 ODEX 提出用 ontology 來描述資料導覽或資料探勘的一些操作，也可以用來描述各個工具的資訊，例如：輸入/輸出、參數、型態....等相關資訊。此時不再需要靠程式本身去撰寫工具和工具之間的關係，ODEX 把存在於各個工具之間的關係，交給描述工具的 ontology 紀錄，間接的大大減少了各個工具之間的相依性，增加了工具合作的彈性。所有的工具被適當的描述在 ontology 中，將容許系統定義更靈活的規則。因此，我們

引入 ontology 知識庫至系統中所希望達到的優點有以下三點，

1. 有助於分析問題領域的效率。

   當問題空間被描述清楚後，表示整個知識空間已經被定義好了，就可以針對某個概念的知識搜尋，也可以尋找特定條件的概念，就好像把整個 Ontology 當成一個知識庫來使用。使用者也可以經由一個已知得概念開始，針對此概念來推論，利用 Ontology 定義好的關係，推論出其他的資訊，讓使用者可以從不同的角度去探討問題的解答。而生物資訊往往需要從一群已知的理論中，推倒出隱藏的資訊，或是將未知的序列跟一群已知的序列比對推論其未知序列的功能，這些問題都需要推論的功能，而 Ontology 正好符合需求。

2. 將問題範疇定義的更明確。

   若說要提供一個生物資訊的整合系統，這樣子系統要包含的問題範疇太廣泛了，無法給使用者一個明確的觀念，了解真正問題的範圍。若透過 Ontology 的概念化特性，將問題空間中的概念一一描述清楚，再利用屬性定義它們之間的關係，將問題的空間用一個 Ontology 來表示，系統若使用了這個 Ontology，自然就將問題的範圍集中在這個 Ontology 所描述的空間。若使用者覺得問題的範圍太小或太大，也可以藉由修改 Ontology 來達到調整的效果。

3. 將概念與實體分開描述。

   在 Ontology 中的每一個節點，都代表的是一個概念，一個在被描述領域中的重要概念，而 Ontology 會利用屬性，對此概念加以敘述，所以概念僅代表是一個想法，而真正此概念的實作，就稱為實體，又稱為屬於此概念的實體，實體中具有真正實作這個概念的方法，所以同一個概念可以有很多個實體，每個實體真正在實物不一定相同，而生物資訊中往往因為認知的不同，而同一個說法，但是在兩個地方表示的東西不同，一個基本的例子就是”gene”這個名詞，在某個資料庫中定義為”DNA 的密碼區段”，另一個資料

庫可能定義為”可以被轉錄或轉譯成蛋白質的 DNA 片段”。若能將概念清楚的描述，則就不會出現上述的情況，因為在屬性中已經明確的說明該概念所代表的意義。

第二節 Ontology 驅動之 ODEX 系統細部架構

本節將針對加入 Ontology 驅動之 ODEX 的細部設計架構描述重要的設計概念，下圖為加入 Ontology 驅動與規則驅動引擎之 ODEX 的系統架構(下頁圖 15)，

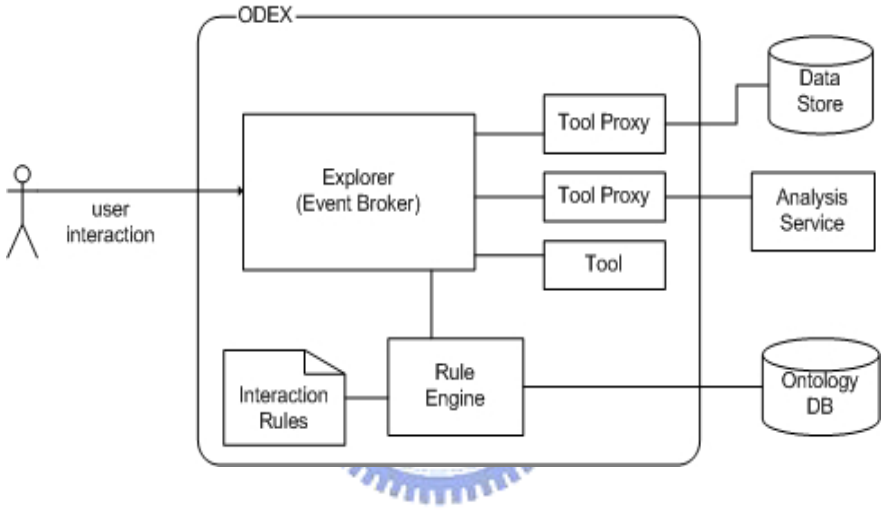


圖 15 加入 Ontology 驅動與規則驅動引擎之 ODEX 系統架構圖

ODEX 基本上是一個經由許使圖形化使用者介面(GUI)工具所集合而成的圖形化使用者介面架構，其中一些工具所提供的功能是經由服務代理人(service proxy)來存取遠端的資料或是計算的服務。這些工具是透過一個整合性介面來管理，這裡稱為系統導覽員(Explorer)。此系統導覽員的功能類似於一般廣為人用的視窗作業系統中的視窗管理員的角色，但是比較簡化。明確來說，當使用者操作工具的同時，就好像是操作系統導覽員，而相對應的事件(event)也會被產生，而系統導覽員本身也被當成是一個事件中繼者(event broker)的角色，負責決定什麼樣的反應將會被驅動。系統導覽員將判斷事件相對應動作的決定權委任給不同的規則驅動引擎(Rule Engine)，規則驅動引擎是以互動性規則中描述的反應為基

礎回應給系統導覽員相對應的動作，而互動性規則事實上是由使用者自定的規則集合所構成。因為互動性規則允許使用者以 Ontology 知識庫為基礎，定義出適當的反應動作，所以可以有彈性的組合或驅動新的工具，並且 Ontology 變成相當的具有客製性。同時將 ODEX 系統中所有的元件之間交換的訊息定義標準化格式(以 XML 為標準)。因此，規則驅動引擎的工作就簡化為樣版比對(pattern-matching)和訊息轉譯(message-interpretation)，讓規則驅動引擎的功能更加明確，且不失其客製化的彈性。下一節將詳述幾個系統中重要元件及其系統在所扮演的角色和功能。

第三節 Ontology 驅動之 ODEX 系統重要元件

5.3.1 Ontology 服務

在 ODEX 中每一個 ontology 就相對應到一個圖的資料結構，Ontology 中的每一個 concept 就是圖中的一個節點，而 concept 和 concept 之間的關係就是節點和節點之間的連結，而圖中完整的描述系統所有可執行的指令以及它們之間的關係，其中包含一些固定的，而且專門給系統使用的關係，如：is_a 和 instance_of ，這兩個是專門分別描述繼承和包含這兩種特別的關係。對於 ODEX 系統來說，ODEX 的 ontology 可視為一個地圖，和其他的地圖的功能一樣，是用來查尋下一步該往哪裡走。只是 ODEX ontology 這個地圖專門用來描述資料瀏覽的動作或不同的工具和資料型態。使用者可以利用這個地圖來操作系統，也可以交給 ODEX 處理，由使用者的操作，並且查尋 ODEX ontology 地圖，找出下一個可執行的工具或是顯示的動作，再反應給導覽員視窗介面，幫助使用者在操作上更輕鬆，更容易找出隱藏在大量資料中的訊息。藉由 ontology 知識庫的加入，使得在執行 Pattern 的比對時，也可以加入 Ontology 的查尋，如此可以強化 Rule Engine 的功能，而且增加導覽資料的靈活性。考慮一個簡單的例子，

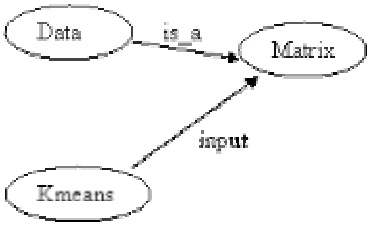


圖 16　ODEX ontology 範例圖

上圖 (圖 16) 有三個 Concept，其中 Matrix 有一個 is_a 關係指向 Data，表示 Matrix 是一種 Data，而 Kmeans 有一個 input 的關係指向 Matrix，表示當執行 Kmeans 時，它的輸入值必須是一個 Matrix 型態的資料。此時，當使用者選取到 Matrix 型態的資料時，可以寫一個簡單的規則，找出所有輸入是 Matrix 的 Concept，則 Kmeans 就是其中一個，則使用者就可以針對選取的資料執行 Kmeans 的分析。

5.3.2　工作集服務 (Working Set Service)

為了在分散式的架構下提供一個整合的工作環境，我們加入了一個虛擬的資料管理服務稱為 Working Set，Working Set 是表示目前正在工作的資料集合，就有如在開發一個程式的時候，會在一個目錄下新增一個 project，而同一個目錄下也有其他的 project 被開啟，所有的工作都在這一個目錄下執行，則這個目錄就被稱為 Working Space。同理，當在執行資料分析或是資料處理的時候，都需要準備一些尚未處理的資料，稱為 raw data。而這些 raw data 就要被集中放在一個共用的地方，本論文稱這個地方叫做 Working Space，當資料處理後，處理完的結果也會放回 Working Space 中，只是不同於之前的 raw data，這些處理完的結果必需集中起來，用一個新的類別把資料加以分類，而集中起來的集合就稱作是 Working Set，所以一個 Working Space 下會有很多個 Working Set，每一個 Working Set 都會屬於某一種型態，特定型態的 Working Set 只會收集特定型態的資料，例如 Gene Set 這個型態的 Working Set，只會收集 Gene 這個型態的資料。而每一個 Working Set 都有名字。所以若要對一個 Working Space 存取某一個 Working Set，

就必須指定要哪一個型態的 Working Set，而且叫什麼名字，才可以取得想要得到的 Working Set。

Working Set 的目的就是提供 ODEX 的工作環境一個可以暫存資料的地方，所有在不同資料庫或是檔案伺服器的資料，都可以用暫存的方式存放在 Working Set，所謂暫存的方式就是資料的實體事實上是放在資料庫或檔案伺服器中，但是在 Working Space 裡只存放該資料儲存的位置，不存放真正的資料。因為在生物資訊的資料往往非常的大，要將資料在不同的元件之間互相傳遞是相當不經濟的。因此，當有一個分析程式需要對一個 Working Set 下的所有資料做分析，Working Space 只要將它指定的 Working Set 下所有資料的位置告訴該分析程式，分析程式自己再到位置中紀錄的資料庫去存取該筆資料即可。使用指標的方法還有另一個好處，就是不會照成資料不一致的問題，因為資料永遠都只有一份，當資料庫中的資料更改了之後，其他應用程式再來存取，一定可以拿到最新的資料。

5.3.3 附加工具集

除了上述的各個特殊的元件之外，在系統中也可以加入其他使用者介面元件，提供特定的目的和特定的分析功能，本論文稱作是工具(Tool)。因為這類的元件都是為特定問題而開發出來，而 ODEX 的基本目標就是將不同任務的工具整合起來，讓彼此之間可以互相有關係，相輔相承。每一個工具再被加入到 ODEX 的環境中，就被視為一個元件，而元件和元件之間的溝通都要透過 Explorer 的轉遞才可以達成。每一個工具在被使用者操作的同時，都會產生出事件，這時就必須去判斷，哪些事件是 local-event (僅需要交給工具的事件處理者處理即可)，哪些事件是 explore-event (需通知 Explorer 或是其他工具，本身的況態改變或是發出事件)，這項工作是交由程式開發者自行決定，其中系統有預設提供幾個比較常使用的 explore-event 讓工具的開發者可方便使用，包含 right-click、mouse-click…等等。當工具接收到這些事件時，不一定要交給 Explore 處理，若事件只影響工具本身，或是工具本身相當獨立，則工具可以自行處理。

第四節 本章總結

ODEX 以服務導向環境為基礎，進階地提供「Ontology 驅動化」的特性。ODEX 的基本功能是整合不同的生物資訊工具和資料，輔助使用者循序地操作不同的工具，瀏覽不同的資料。而 Ontology 的功能就是描述網路上的資源與資源之間的關係，並且用 metadata 的方式對網路上的資源加以描述，使得其他應該程式可以透過搜尋的方式找到想要的資訊。因此系統使用 Ontology 來幫助使用者瀏覽資料，操作使用者介面。在系統中，Ontology 中定義了使用者在操作使用者介面時所可能使用到的動作，或是工具，並清楚的定義動作的屬性、型態、與其他元件之間的關係，以便供規則導向引擎查尋，查尋結果會獲得該動作的特性、型態、所需要的輸入、和輸出，規則導向引擎會根據查尋結果，使出適當的回應，如：要求使用者輸入特定的參數、呼叫其他工具來完成工作...等等。因此當 Ontology 中描述了不同的關係，系統就可以在 Ontology 中找出一條路徑，系統會利用此路徑來引導使用者完成某件特殊的工作，如此不僅僅提高了資料導覽的效率，還可以提供一個簡易的操作流程，協助對 Ontology 所描述的領域不善長的使用者輕鬆地操作系統中的工具。

第六章、ODEX 系統實作

第一節 樣版整合

樣版整合是為了提供一個類似 Java 中排版的方式，將關係較為密切的工具組合，而 ODEX 目前提供了基本的樣版，可以讓使用者自行組合成較複雜的樣版。而事實上在 ODEX 的樣版整合是提供一個樣版整合的工具，這個工具和之前的本章第一節的工具一樣必需設定脈絡工具，而這個樣版工具所允許的脈絡工具形態必需是具備使用者介面的工具，如此才可以將脈絡工具放置到適當的排版位置。底下為樣版整合的範例：

```
<tool>
   <create template="gui/panel"/>
   <cntx path="./toolTree" name="left"/>
   <cntx path="./toolEditor" name="right"/>
   <config style="hsplit">
      <left tool="left"/>
      <right tool="right"/>
   </config>
</tool>
```

範例 1　樣版整合範例

上述範例將兩個工具放入水平分割的樣版中，有如 JAVA 程式中的 JSplitPane 類別，我們將原本需要撰寫程式的功能，用簡單的設定達成。下頁(圖 17)為樣版整合示意圖，是以範例 1 為例子簡單的畫出樣版整合的概念。

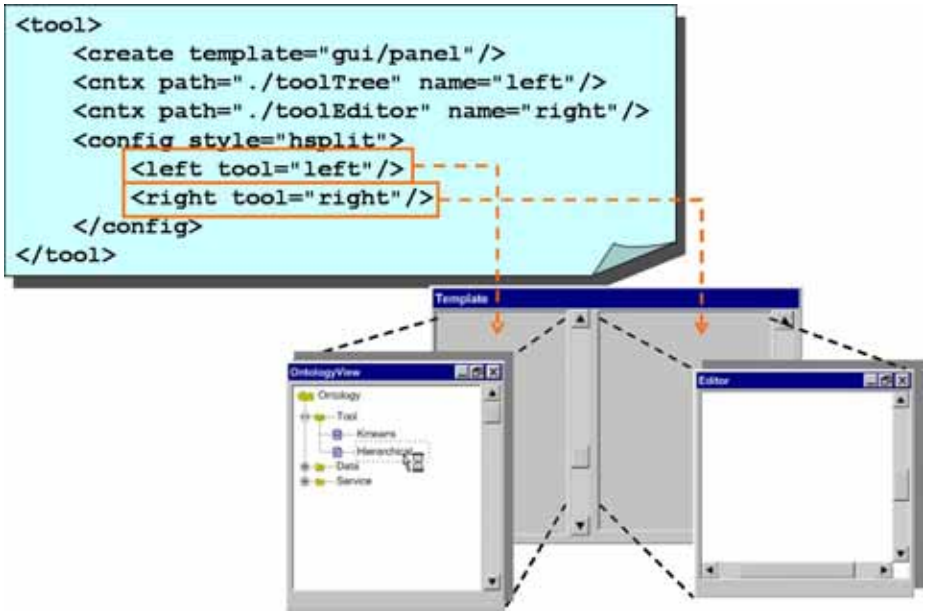


圖 17　樣版整合示意圖。將樣版定義成左右兩個區塊，並分別將區塊所將配置的工具設定，即可將工具嵌入至樣版中。

第二節　定義規則驅動引擎(Rule Engine)

規則驅動引擎主要是要增加系統的彈性，讓系統能依不同的狀況，透過規則的比對做出適當的反應。而規則驅動引擎是依據規則倉儲中的規則集合來判斷回應，因此規則集合成了規則驅動引擎的主要構成元素。規則驅動引擎就是由許多的規則集合則組合而成，在規則集合中往往都是功能性相似的規則，才會被集合在同一個規則集合中。將規則有效的分類，在比對時也可以只針對特定集合來運算，不需把整個引擎的規則都比對一次，如此有助於規則驅動引擎的效率。

每一個規則是由兩個部分所組成，第一個是 if 判斷式，if 判斷式中又分成兩個小部分，第一，先會直接與輸入的格式做 pattern 的比對，若不符合，則再比對下一條 Rule，若符合則進入第二個小部分，Condition 比對，這裡允許多個 condition，若所有的 condition 都通過，則就會跳出 if 判斷式，執行第二部份的敘述式，也就是程式語言當 if 條件成立後，要執行的動作就是敘述式，而規則驅動引擎最後會將敘述式的結果回傳給呼叫規則驅動引擎的元件。下圖 18 為規則

驅動引擎在比對規則的流程圖：

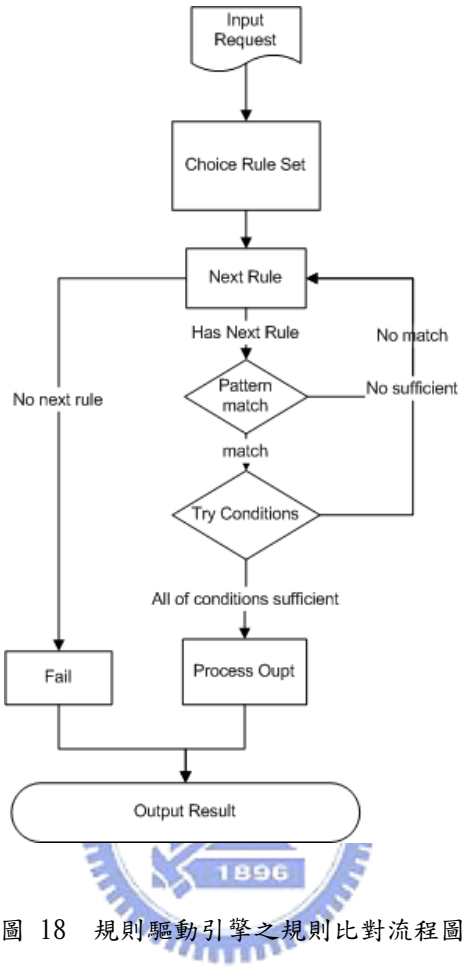


圖 18　規則驅動引擎之規則比對流程圖

底下是一個簡單的規則，

```
<rules>
     <rule>
          <if> <a/> </if>
          <b>
   </rule>
</rules>
```

這個規則集合中定義了一個簡單的規則，這個規則所做的事就是，若輸入為<a/>則輸出就是<b/>。

為了能達到可以像迴圈一樣處理輸入中有重覆出現的樣版，規則驅動引擎還

提供了遞回處理的功能，輸入的部分內容可以再被同一個規則集合比對一次，以達到用 if 判斷式模擬迴圈的功能。

```
<rules>
   <rule>
   <if><a><rw:list name="list"/></a>
   </if>
   <a>
   <rw:rw><bList><rw:list name="list"/></bList></rw:rw>
   </a>
</rule>
<rule>
   <if><bList><b><rw:list name="tail></bList></if>
   <c>
   <rw:rw><bList><rw:list name="tail></bList></rw:rw>
</rule>
<rule>
    <if><bList></bList></if>
</rule>
</rules>
```

上圖的規則集合是由三個規則所組合而成，第一個規則描述若是由<a/>的標籤內有其他子標籤時，就滿足第一條規則，則輸出結果會將<a/>標籤保留，把中間的子標籤用<bList/>這個標籤包起來，再重覆對這個規則集合做比對，而<rw:rw/>就是告訴Rule Engine要把<rw:rw/>中的內容再重新對同一個rule集合做比對，若<bList/>的子標籤的內容滿足第二條規則，也就是第一個標籤是<b/>，其他則 don't care 時，就會將第一個標籤改成 <c/>，其他子標籤再用<bList/>包起來重覆對同一個集合比對，直到<bList/>中不包含其他子標籤時，就滿足第三條規則，則不做任何的回應。由此可以看出這是一個類似遞迴的撰寫方式，會一直重覆執行比對，直到內容滿足停止條件為止。所以當輸入為，

```
<a>
    <b/>
    <b/>
    <b/>
</a>
```

經過上述的 rule 集合，則輸出是，

```
<a>
    <c/>
    <c/>
    <c/>
</a>
```

規則驅動引擎還提供了委任的功能，能將部份解讀的工作，委托給其他的規則集合完成，可以將不同功能和特性的規則集合起來，所以同一個規則集合所負責的任務都差不多，若是其他規則集合遇到特定的 pattern，就可以將這個 pattern 交給專門處理的規則集合解讀，再把解讀結果回傳即可。清楚依功能來劃分規則，可以讓這些規則集合被重覆使用，也可以間接的增加規則驅動引擎在執行比對的效率，因為每一個規則集合中的規則數量減少，所需要判斷比對的次數也相對減少，而且在規則的維護上也比較方便，當系統需要修正或改變功能時，只需要修改負責這個功能的規則集合即可，不需要影響到其他功能的運作。

```
<rules>
<rule>
<if> <init/>
    <rw:delegate name="init">
    <initMenu/>
    </rw:delegate>
</if>
</rule>
</rules>
```

上述的規則就是描述當輸入為<init/>時，就將工作委任給 init 這個規則集合去執行，並且新的輸入值為<initMenu/>。

第三節 ODEX Ontology 之描述

在 ODEX 中的 Ontology 最主要的功能是用來幫助使用者導覽資料和操作工具，所以 Ontology 被視為一個地圖，導覽資料的地圖。因此在 ODEX 的系統中，不需要太強大的資料結構，僅需要能夠將各種資料型態和工具的屬性描述清楚，而且可以表示出他們之間的關係就可以達到地圖的功能，因此這裡就將目前的 Ontology Language 簡化，Ontology 的 Concept 用<entity/>來描述，Concept 與 Concept 之間的關係是用<rel/>放在<entity/>之間，<rel/>可以使用 name 這個屬性定義關係的名字，因為這個 Ontology 是描述工具的型態，所以<rel/>還有另一個屬性是 type，表示是和哪一個型態有關係，若要和其他的 entity 有關係，就必須 rel 這個標籤加入 entity 這個屬性，其值為另一個 entity 的 id，例如：

```
<entity id="Tool">
<rel name="isA" entity="OdexOntology"/>
</entity>
```

上述表示 Tool 這個型態和 OdexOntology 這個 entity 有 isA 的關係，也等於 Tool is_ A OdexOntology 的意思。<entity/>還可以用<attr/>來定義 entity 具有的屬性，name 是屬性的名字，value 為屬性的值。

```
<Ontology>
<entity id="OdexOntology"/>
<entity id="Tool">
   <rel name="isA" entity="OdexOntology"/>
</entity>
<entity id="InterestingNessTool">
   <rel name="isA" entity="Tool"/>
   <rel name="input" entity="GeneSet"/>
</entity>
<instance id="ClusterMiner">
   <rel name="instanceOf" type="InterestingNessTool"/>
   <attr name="path" value="tool/cmTool"/>
</instance>
<entity id="KmeansTool">
   <rel name="isA" entity="Tool"/>
   <rel name="input" entity="Matrix"/>
   <rel name="output" entity="ClusterResult"/>
</entity>
<entity id="HierTool" >
   <rel name="isA" entity="Tool"/>
   <rel name="input" entity="Matrix"/>
   <rel name="output" entity="ClusterResult"/>
</entity>
<entity id="MyKmeansTool">
  <rel name="instanceOf" entity="KmeansTool"/>
  <attr name="path" value="odex/kmeansTool"/>
</entity>
<entity id="MyHierTool">
  <rel name="instanceOf" entity="HierTool"/>
  <attr name="path" value="odex/hierTool"/>
</entity>
</Ontology>
```

範例 2　Ontology XML 格式

範例 2 是 Odex Ontology 的部分片斷，其功能是用來描述 ODEX 維護的工具型態，在文件中每一個 entity 都是視為一個型態，在上述 Ontology 中有幾個

"built-in"的關係，分別是 isA 和 instanceOf， isA 是用來描述屬於的關係，因為上述的 entity 都起始於 Tool，所以上述的 entity 都屬於 Tool，表示都可以加到 Explorer 中被顯示出來。其相對應的示意圖如下：圖 19 的虛線部份表示 instance。

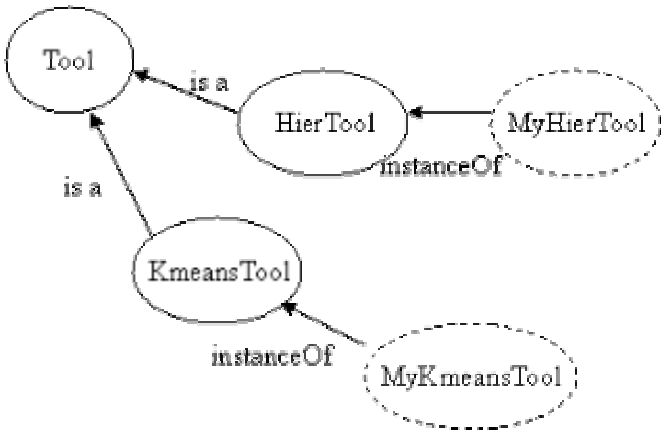


圖 19　Ontology 示意圖，對應範例 1 中所描述的 ontology 與之對應。

第七章 ODEX 系統範例

本章將利用範例說明本論文所提出的概念，並提供一些設定片斷和系統剪影的對應，再藉由文字的說明，有助於了解前面章節所描述的概念。

下圖(圖 20)為系統導覽員介面，如前面章節所描述的一樣，系統導覽員在設計架構中主要是一個中繼者的角色，提供工具容器（Tool Container）和系統事件的處理的功能，本身並不提供任何計算或資料的傳遞，基本上若未加入任何服務或工具至系統中，系統導覽員就是一個空的工具容器，不具備其他額外的功能。而系統導覽員可以利用簡單的規則設定去建構初始化的選單，如圖中所示，已加入了預設的工具，範例 3 為設定初始化選單的規則片斷。

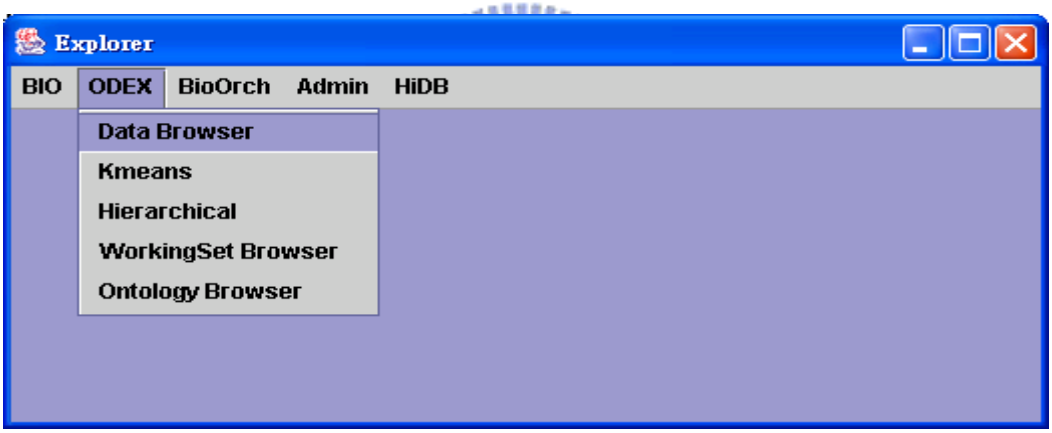
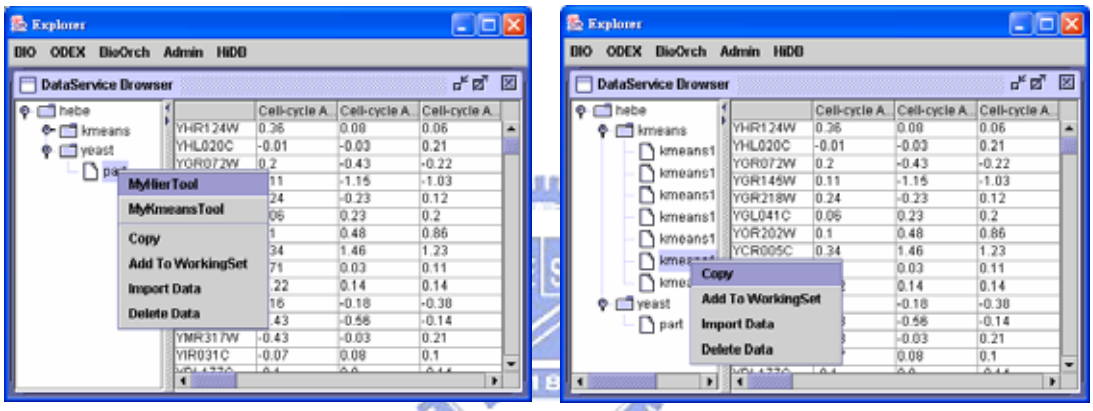


圖 20　系統導覽員

```
<init>
    ...
    <menu label="ODEX" path="" name="odex"/>
    <menuItem label="Data Browser" path="odex" name="dataBrowser">
        <action>
            <open path="odex/dataBrowser"/>
        </action>
    </menuItem>
    ...
</init>
```

範例 3　系統導覽員選單設定片斷。本圖可對應至圖 20 的介面，此片斷可看出設定了一個為「ODEX」的選單，而接著將一個「Data Browser」的選項加入至「ODEX」的選單中，而此選項的執行指令就描述在<action/>這個標籤中，本範例是要將”odex/dataBrowser”此路徑下的工具開啟。

系統提供了基本的操作動作，例如：按右鍵 (Right Click) 、複製 (Copy) 、貼上 (Paste) 、選取 (Select) …等等，可以在規則集合中加入有關基本操作規則，使得這些基本操作可以依不同的工具，執行適當的動作。當使用者針對資料操作時，可定義一旦執行了滑鼠右鍵的動作，可利用彈出式選單將可執行的操作列出來。



9(a)　　9(b)

圖 21　9(a)中，顯示當游標選擇資料(yeast/part 該筆資料型態為 Matrix)時，跳出式選單所列出可執行的動作。與 9(b)比較，當選擇的資料(Kmeans 目錄下的資料型態為 ClusterResult)是屬於不同型態的資料時，所列出的清單也會不同。

除了基本的操作可以藉由規則來設定其功能，當加入了 Ontology 知識庫至系統中，就可以設定有與 Ontology 中描述的概念相關的規則。本系統利用 Ontology 來描述目前被提供的工具和服務之間的關係。參考前章範例 2 所描述的 Ontology，該 Ontology 僅定義出工具之間的關係和分類階層，當系統再加入描述資料型態的 Ontology，針對這兩個 Ontology 撰寫更強大的規則。圖 21 中的資料瀏覽器中顯示了兩種不同型態的資料，在 yeast 目錄下的為 Matrix 型態，在 kmeans 目錄下的為 ClusterResult 型態，此時設定規則為「列出所有可以接受輸

入為選取資料的型態的工具。」，就如圖 21 中顯示，當選取的資料型態為 Matrix 時，選單中會多出兩個選項 MyKmeansTool 和 MyHierTool，表示在 Ontology 中描述出這兩個工具可以接受 Matrix 資料。

```
<Ontology>
<entity id="OdexOntology"/>
<entity id="Data">
    <rel name="isA" entity="OdexOntology"/>
</entity>
<entity id="Matrix">
    <rel name="isA" entity="Data"/>
</entity>
<entity id="ClusterResult">
    <rel name="isA" entity="Data"/>
</entity>
</Ontology>
```

範例 4　描述資料型態的 Ontology

當分別選取 MyKmeansTool 和 MyHierTool 這兩個選項，相對應的工具將會被開啟。可透過基本的操作將資料導向這兩個分析工具中，分別是執行 Kmeans 分群法和階層式分群法，計算的服務是一個位在另一台主機上，而本機僅是表現層介面，將計算的結果顯示給使用者。所以當建構這兩個工具是必需設定其實體服務的位置。底下分別是這兩個工具的設定檔。而圖 22 為執行的結果。

```
<tool>
   <create class="odex.tool.hier.gui.HCLTool"/>
   <cntx path="../host/hier" name="hierService"/>
</tool>
```

```
<tool>
   <create class="odex.tool.kmeans.gui.KmeansTool"/>
   <cntx path="../host/kmeans" name="kmeansService"/>
</tool>
```

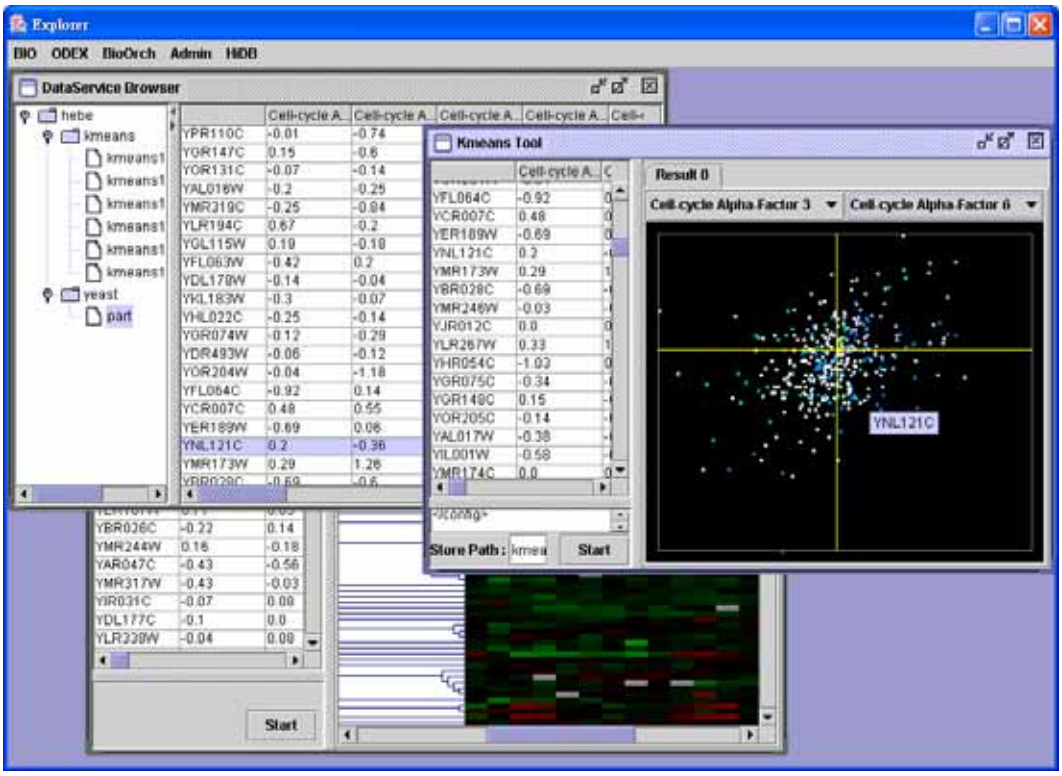


圖 22　系統導覽員執行資料分析時，執行 Kmeans 與階層式分群法的結果

系統還提供基本的樣版式整合，其工具之間的溝通與前提的相同，但可以藉由設定樣版將使用者介面工具組合成新的工具，方便使用者操作與瀏覽。下圖(圖23)為樣版整合的實作結果。

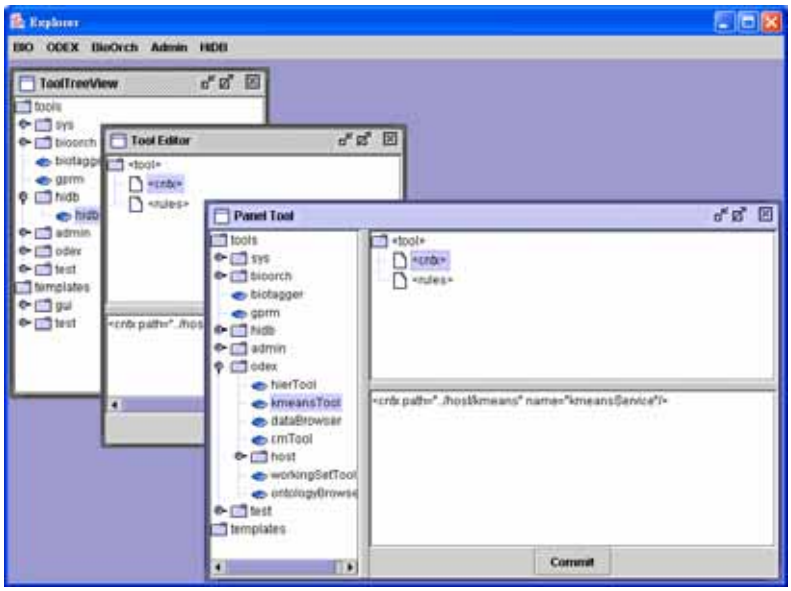


圖 23　樣版式整合。最左上角與中間的視窗為分解的兩個工具，經樣版整合後可獲得右下角視窗的工具。

第八章 結論

本論文以 Ontology 描述資料型態與不同資料處理工具之間的關係以及一些資料探勘的操作，再加上透過以 Ontology 為知識庫而驅動使用者介面的進階規則，使系統能輔助使用者操作和導覽不同的生物資訊資源，且經由規則的設定，研究人員可建構符合自己研究領域的操作風格，除此之外，利用基礎的樣版，研究人員可以組合相關性較高的分析工具。如此系統不僅僅具備了簡單的介面整合，且提供了智慧型的引導，讓研究人員可更便利的組識屬於個人化的工作平台，有助於在研究上的發展和分析實驗數據的效率。

整體的系統架構都是以 XML 為基礎，利用 XML 多元化的格式，將所有服務和工具之間的溝通協定透過 XML 格式來傳輸，且規則的設定以及 Ontology 的建構，也是使用 XML 格式表示，使得系統有辦法提供進階的搜尋和比對，幫助系統在引導使用者操作時，可以提供更靈活的使用者介面和更有意思的數據資訊，改善目前單調的分析模式。

系統架構主要是 Ontology 所驅動，而本論文將 Ontology 概念簡化至圖的資料結構，能夠描述大多數資源與資源之間的關係所構成的關係圖，因此對於 Ontology 所描述的領域給與高度的彈性，所以系統也具高度的彈性，讓不同領域的研究人員都可以建製適用於研究領域的 Ontology，如此也便於整合不同領域的知識至系統中，使得不同領域的資料和服務可以互相合作。

由於套用分散式服務驅動基礎架構，使得不論是本機或是遠端的服務，因為都是透過服務代理人來使用，所以都像是本機的工具一般被整合至系統中，且加入了主機 (Host) 的概念，提供了非中央集中式的架構，使系統具備高度的延伸性和彈性。

系統導覽員為單純的使用者介面，負責基本的系統事件處理，和工具管理與維護，由於分散式的架構，將計算模組與表現層分開，真正的資料或計算皆可存在於遠端的機器上，使得系統導覽員成為一個非常輕的應用程式，對於本機電腦資源的需求也相對的降低。客戶端僅需載入需要的使用者介面工具，再經由系統設定，就可以獲得位在遠端的資料並執行資料分析的工作。

雖然 ISYS (*A.Siepel* et al*, ISYS 2001) 亦提出了非中央集中式的概念，同樣讓系統具有靈活性、可建構性、適合將已開發完成的軟體整合至系統中互相合作、和組裝元件容易等特性。但其不同點在於 ISYS 是利用匯流排的機制，所有元件透過代理人與 ISYS 平台中的匯流排溝通，當然資料也會傳遞至匯流排上，再交給目標元件處理。而本論文將服務交由提供服務容器功能的主機 (Host) 共同管理，而主機僅是一個窗口，而每一台機器都可以為主機，使得服務之間的溝通是透過主機這個窗口，如此把客戶端系統平台的負擔分散給不同的主機，資料直接經由各個主機的窗口相互傳遞，客戶端僅傳送簡單的指令，直到計算完成，再把結果一次傳送回來，顯示給使用者。這種機制不僅減少了客戶端的負擔，也減少不必要的資料傳遞，有助於系統執行效能。

在元件之間的耦合度，ISYS 利用動態發掘(Dynamic Discovery)的機制，管理各個元件之間的關係，元件在加入至系統中，需要先跟動態發掘員註冊，表示建立了一個連結，當特定的物件被選取時，動態發掘員就會動態的顯示有關係的模式。ODEX 系統將元件之間的耦合性從系統層轉移至 Ontology 的描述中，透過 Ontology 的建構同時將元件之間的關係連結起來，而元件和元件之間僅用代理人溝通，因此只要更改 Ontology 的結構，相對的也改變了元件和元件之間的關連性。ODEX 同樣的也提供較鬆散的耦合度，使得系統可以更直覺的將元件組合，且因為 ODEX 是利用 Ontology 來驅動，所以提供了簡易的設定，讓使用者可以更輕易的修改系統元件的關連性。

ODEX 系統採資料與模組複合式導向分析，與 RiboWeb (*Russ B. et al*, RiboWeb, 1999)採用模組導向分析不同，結合了模組導向與資料導向。模組導向分析適用於已對問題有一定程度的了解。但本系統針對不同程度的研究人員提供不一樣的導覽方向，增加了以資料型態為出發的導覽方式，讓本身系統提供工具不熟悉的研究人員使用。未來，希望能更進一步針對問題導向分析，將問題定義於 Ontology 中，若下次遇到相同的問題領域，可以套用先前已建構的流程，讓「經驗」也可以加入至系統成為 Ontology 的重要概念。

僅管目前 ODEX 系統的雛型雖僅提供基本的分析工具，隨著系統的發展，希望能提供更人性化的控制介面，將系統的設定規格化，有助於使用者在建構個人工作環境時，可以更直覺的設定系統的架構。日後期望能獲得生物相關領域的研究室幫助，提供實例和資料，將此概念實現在真正的問題上，讓研究人員可以在一個分散且互相合作的環境下工作。

參考文獻

[1]

GeneBank

National Center for Biotechnology Information (NCBI), National Library of Medicine, National Institutes of Health, USA.

http://www.ncbi.nih.gov/Genbank/index.html

[2]

European Bioinformatics Institute (EBI)

Wellcome Thrust Genome Campus, UK. http://www.ebi.ac.uk/Databases/

[3]

DNA Data Bank of Japan (DDBJ),

Center for Information Biology, National Institute of Genetics, Japan.

http://www.ddbj.nig.ac.jp/

[4]

Microarray

DeRisi Lab, Dept. of Biochemistry & Biophysics, Univ. of California at San Francisco.

http://www.microarray.org

[5]

The Protein Data Bank (PDB),

http://www.rcsb.org/pdb/

[6]

David J. Lockhart and Elizabeth A. Winzeler.

Genomics, gene expression and DNA arrays.

Nature, 405, 827 – 836 (June 2000).

[7]

S.Fischer et al,

BioWidget: data interaction components for genomics

Bioinformatics, Vol. 15 no. 10, 1999, p837-846

[8]

A. Siepel, et al.

ISYS: a decentralized, component-based approach to the integration of heterogeneous bioinformtics resources.

Bioinformatics, Vol. 17 no. 1, 2001, p 83-94.

[9]

Natalya F. Noy and Deborah L. McGuinness , Ontology Development 101 : A Guide to Creating Your First Ontology , Stanford University, Stanford, CA, 94305

[10]
The Gene Ontology Consortium.
Gene Ontology: tool for the unification of biology. Nature America Inc. 2000.
http://genetics.nature.com.
http://www.geneontology.org/
[11]
P.G. Baker, A. Brass, S. Bechhofer, C. Goble, N. Paton, and R. Stevens.
TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources.
An Overview.
In Proceeding of the Sixth International Conference on Intelligent System for Molecular Biology (ISMB' 98), pages 25-34, Menlow Park, Califonia, June 28-July 1 1998. AAAI Press.
[12]
Carole A.Goble and Rober D. Stevens
Managing Biological Information Using Biological Knowledge.
http://imgproj.cs.man.ac.uk/tambis/details.html
[13]
TAMBIS
University of Manchester
http://imgproj.cs.man.ac.uk/tambis/
[14]
Russ B. Altman, et al
RiboWeb : An Ontology-Based System for Collaborative Molecular Biology
IEEE Intelligent System, 1999, p68-76
[15]
Robert Stevens et al
Ontology-based Knowledge Representation for Bioinformatics
September 26, 2000 p1-19
[16]
W3C Web Services Activities,
http://www.w3.org/2002/ws/
[17]
DMAL
DARPA Agent Markup Language
http://www.daml.org/index.html
[18]
OWL
Web Ontology Language

W3C Recommendation 10 February 2004
http://www.w3.org/TR/owl-ref/

[19]
The UDDI Project,
http://www.uddi.org/

[20]
Microarray Gene Expression Data (MGED) Society,
http://www.mged.org/

[21]
The race to computerise biology,
Economist.com, Technology Quarterly, Dec. 12, 2002
http://www.economist.com/science/tq/displayStory.cfm?story_id=1476685

[22]
Michelle Kessler, USA TODAY, Jan. 8, 2003.
www.usatoday.com/money/industries/technology/2003-01-08-shared-computing_x.htm