基於語意網 2.0 技術的語意 Wiki 平台

許乙清* 周伯彥

國立虎尾科技大學 資訊工程系

hsuic@nfu.edu.tw*

摘 要

本研究整合語意網與 Web 2.0 技術發展一可描述語意的 Wiki 平台，語意部份使用 OWL 來訂定抽象類別與語意解釋，相對地，法則是一種邏輯規則可建構在知識本體來執行複雜的規則推論。該語意的 Wiki 平台中，Wiki 是描述特定領域的知識平台，使用者的喜好檔中有設定該領域的類別，藉由 OWL 與 Rules 使 Wiki 系統有語意描述功能，最後使用推論引擎推論出的內文可依使用者需求顯示於 Wiki 頁面，並將 Wiki 可能產生出冗餘的文章過濾並隱藏，借此讓使用者在閱讀上更便利。為了驗證本研究所提整合方法的可行性，我們以咖啡為知識領域發展一個雛型系統，該系統執行時會依個人喜好與現有食材，推論出可替代的咖啡食材或篩選使用者的喜好分類。

關鍵字：知識本體、語意網、JSPWiki、Semantic Wiki

A Semantic Wiki Platform based on Semantic Web 2.0 Approach

I-Ching Hsu* Po-Yen Chou
Department of Computer Science and Information Engineering, National Formosa University
Huwei Township, Yunlin County 632, Taiwan
hsuic@nfu.edu.tw*

Abstract

This study has developed a semantic Wiki platform based on integrating Semantic Web and Web 2.0 technologies. We used OWL to define abstract classes and provide semantic reasoning. In contrast, the rule is a logic program, which can complement ontology to support more complex rule-based inferences. In the semantic Wiki platform, Wiki plays as a knowledge platform to adopt OWL and rules to provide sound and decidable reasoning for a specific domain. User will set their favorites based on the domain classes to create a profile. The inference engine infers new information based on user profiles. So that, when user read wiki content, wiki will automatically hide some redundancy information if this content doesn't fit user's favorites. To demonstrate the feasibility of our semantic Wiki platform, a prototype system based on coffee domain is implemented. When user login they have to set their profiles. In profile page they will select their food stuff for coffee and favorites. Then the developed system will show the common stuff and what kind of coffee they can make.

Keyword: Ontology, Semantic Web, JSPWiki, Semantic Wiki

壹、簡介

Web 2.0 的發展迄今 6 年之久，爲現在的網頁與網路服務帶來了不少助力，著名的應用如 Blog、Wiki、IGoogle 等，透過使用者互相傳達訊息與修改，使得網站資訊不斷的更新與改進，近年來，更因相機與 GIS 系統的結合，有更多的應用於網路地圖與網路相簿，但這些都只是單方面將資料釋出，而非針對個別使用者的有用資訊，Peter F. Drucker 曾經表示過，真正促使社會成長的是 "Information Technology" 裡的 "Information" [26]，科技只會帶來單方面的成長，站在使用者的立場，喜歡科技的人會不斷找出改進的空間並解決環境上的不便之處；而與科技不相關的使用者會越來越畏懼，最後會開始躲避這些科技所帶來的應用。但這些科技困難嗎？一點也不，這些系統規劃時已經爲了讓使用者可以迅速與簡易使用，把早期不易操作的指令集模式改良成圖形化介面，也因此使用上更加人性化。雖說現今的網頁可以讓使用者輕易的解讀，在網頁設計上也有許多套裝軟體讓使用者可簡易開發，但是這些單純的文字描述又有多少使用者感興趣呢？若無法讓這些科技技術與傳統產業結合，這將是未來科技進步的大問題。到底是人類與科技脫軌?還是科技帶來更多的不便利?

現在市面上有許多動態網頁開發環境，諸如：PHP、JSP、ASP 等，若想要在網頁上有更多的互動，就得靠這些動態網頁語言輔助，最常見到的是使用 PHP 透過動態描述語法作爲判斷式，當碰到特定條件時給予使用者應有的回覆，這種判斷式可用於簡易的系統架構，在設計時可自訂判別式即可，假若系統龐大且繁雜，需要許多人員開發與維護，加上系統本身有許多判斷條件，此時再把知識的判斷寫入程式碼中定義，在設計與除錯時將會是個很大的挑戰。如今使用 OWL(Web Ontology Language)語言予以表達知識，可透過當語意描述此領域的內文與之間的關聯性[31]，藉此簡化在程式中加註判別的複雜度與縮短判斷的時間；描述領域的內文與關聯性的 OWL 檔產生後，可使用市面上的 OWL 推論引擎將所撰寫的知識推論，最後可從當中獲取推論出的結果，但每套程式的介面不同，因此推論出來的結果必須先行修改成系統的需求，最後透過網頁語言結合輸出即可。

而本研究將使用 OWL 描述咖啡這塊領域的知識，有了這些知識即可使用 OWL 推論引擎呈現於 Wiki 頁面上，改良早期 Wiki 單純網頁與圖片的頁面[5]，還可透過 OWL 描述類別之間的關聯性並透過推論工具編輯，讓咖啡這個領域中的抽象描述與應用都可以呈現於網頁上。本研究最重要的是改良 Wiki 系統，早期的 Wiki 只有圖文編輯，或是藉由 Google Maps API 來表達地理情資，而本研究藉由 JSPWiki 導入 OWL 與喜好設定加以改良，結合 OWL 與推論引擎讓此系統從單一類別找出更多共同類別與關聯類別，如此可提升文章篩選的正確性與判斷的精確性，藉此找出更多與使用者息息相關的新知，若有需要介紹地理位置時候透過 Google Maps 呈現圖資訊息，使得 Wiki 在 Web 2.0 應用上更爲廣泛。

貳、文獻探討

一、標記語言

標記語言主要目的是讓不同的程式語言，甚至不同的開發平台可藉由一標準讓資訊有一共同格式可互相傳遞，藉此減少資料交換的複雜度或是藉此延伸出更多的

應用。諸如：XML、RDF、OWL、KML 等，透過這些標記語言統一固定的格式或是建立 Schema，使用者僅需依照規範建置資料並確定所建立的檔案是符合 Schema 所建置，此時系統只需使用剖析器將當中有用的資訊截取出即可，與語意網相關的標記語言是由 W3C 所制定，相關的規範包含：

•XML (Extensible Markup Language)[3].
•RDF/XML Syntax Specification (Revised) [4].
•RDF Vocabulary Description Language 1.0: RDF Schema [2].
•RDF Primer [21].
•Resource Description Framework (RDF): Concepts and Abstract Syntax [19].
•RDF Semantics [10].
•RDF Test Cases [9].
•Web Ontology Language (OWL) Use Cases and Requirements [11].
•OWL Web Ontology Language Reference [6].
•OWL Web Ontology Language Semantics and Abstract Syntax [25].
•OWL Web Ontology Language Overview [23].
•OWL Web Ontology Language Test Cases [13].
•OWL Web Ontology Language Guide [29].

(一) XML

XML(Extensible Markup Language)為 W3C 所制定出可擴充之標記語言，XML 規範主要是讓文件符合"Well-formed"的條件。早期的 Html 格式所需的嚴謹性太低，標籤定義上的規範也十分鬆散，其原因是希望可讓使用者快速上手，而缺點是要將龐大的 Html 文件中撈出有意義的資訊，這可將是非常困難工程，原因是現在的網頁瀏覽器已經將 Html 常遇到的錯誤狀況進行自動修補的功能，假若原始的標籤使用上是不合規定，瀏覽器也會自行判讀顯示正確的訊息。想要在無規律的檔案中抓出有用資訊必須要有很精準的程式，如今規範出 XML 檔案，只要網頁遵循 XML 標準進行撰寫，最後可透過 XML 剖析器來協助驗證該檔案是否為”Well- formed”，而後可透過 XML 剖析器當中的資訊取出並加以應用[31]。

(二) RDF

RDF 原名為 Resource Description Framework，其主要架構是透過 triple 關係來描述這些資源，從字面上可知主要是描述資源的架構。所謂的 triple 關係是指 Subject、 Predicate、Object(如圖 1 所示)，Subject 可視為所描述的資源，一般都是指 URI；而 Predicate 是 Subject 與 Object 之關係，而 Object 則是要描述的物件[14]。而 RDF 只是單純描述資源的方法，因此需要一個標準定義其資源的形態與描述抽象定義。

圖 1 The triple relation of RDF

(三) OWL (Web Ontology Language)

知識本體一詞創始於哲學領域，在中文有許多翻譯方式，諸如：知識本體、本體論、知識論等，其主要用意是將抽象物件模型化，近年來拜科技所賜，知識本體在電腦領域也有了更多的應用與變化，W3C 也因此制定出通用的標準語言讓程式

之間可有更多的應用[14,15]。

OWL 是 W3C 爲了描述知識本體所制定出最新的知識本體語言，在整個架構的規範上是遵循 XML、RDF、RDF Schema 規範所撰寫的語言，藉由 XML 文件規範並遵循其標籤規則，制定了另一套用於知識本體描述上的標記語言[20,24]，OWL 也運用了 RDF 的 triple 關係使得人類有辦法傳達資訊給電腦，透過 RDFS schema 的關聯與類別的特性，自訂了兩大特性：ObjectProperty 與 DatatypeProperty，在這兩大關聯下又細分了許多常見的關聯性，如：遞移性、對稱性、反身性等。W3C 也因使用者需求分別制定了三種子語言：OWL Full、OWL DL 和 OWL Lite 這三種版本[14]，依照使用者需求來選擇合適的推論引擎使用，以達到最佳效率。

(四) RDFa

一般的 Web1.0 與 Web2.0 的網頁在撰寫時，雖有特定的規範定義，但是當中定義的欄位資訊與類別主要是讓程式設計師或是維護人員可清楚判別該欄位所對應的描述，然而這些資訊其實可以再利用。因此 W3C 爲了 XHTML 制定了 RDFa 規範，讓 XHTML 可以設置 RDFa 標籤，而本研究將其功能應用於 Profile 之中[9]，早期 XML 建置的 Profile 僅描述使用者資訊或是喜好，而本研究描述咖啡的知識本體，當中定義了許多咖啡類別，而 Profile 的喜好也是依照這些類別所建置，因此我們將單純的 Profile 串上 OWL 定義的類別，往後要修改或是維護程式可馬上看出該項目是對應到何者知識本體的類別。

二、Jena 推論引擎

Jena 主要是由 HP Lab 研發製作的語意網應用程式，Jena 是以 Java 爲開發平台的開放式軟體，此環境提供的開發語言包含有 RDF、RDFS、DMAL+OIL、OWL、RDQL、SPARQL 等[12,16]，而 Ontology 剖析器主要可區分爲三部份：1. RDF 剖析器。 2. RDQL 查詢功能。 3. OWL 剖析器。而 Jena 顯示推論結果的方式爲類似 RDF 中 triple 關聯方式將最後推論結果呈現出來，如：(Subject, Property, Object)，而 Property 主要是描述 object 與 Subject 這兩個資源間的關聯性[22,24]。另外一大特色就是 Jena 本身有自己的 Rule 推論引擎，依照本身制定的格式，即可在程式當中制定 Rule 或是透過 rules 檔案進行推論。

本研究主要是使用 OWL 描述知識本體部分，由於咖啡食材所描述的知識不複雜，大多是描述兩個資源間的關聯性，而市面上的咖啡成品大多是由 3 樣以上的咖啡食材所完成，因此若要描述複雜的關聯性必須使用 Rules 呈現。目前本研究使用 XML 定義使用者 Profile，藉由 RDF 讓電腦解讀 resource 之間的關聯資料，透過 OWL 的 ObjectProperty 和 DatatypeProperty 定義出類別間關聯性。經過多番的資料轉換後，可發現 OWL 所提供的關聯性實在會有不足之處，最後將較複雜的咖啡成品描述使用 Rules 呈現，此時即充分應用標記語言，不僅可提升資料交換的便利性，也可在語意網上有更多的應用。而本研究所用的 Rule 推論引擎是使用 HP 實驗室所開發的 Jena[13,18]。

三、Semantic Wiki

目前有幾種常見的 Semantic Wikis，其主要是沿用 Wiki 標記語言，並且可讓使用

者與管理者共同編輯頁面，知名的MediaWiki也有Semantic MediaWiki作爲延伸，主要是讓使用者可藉由語意編輯註解，常見的 Semantic Wiki 與特色大致如下：OntoWiki 提供的是語意查詢功能；IkeWiki、KauKolu、SweetWiki 提供了部份RDF、OWL、Rules 編輯與推論的功能，有些甚至可上傳RDF檔案或是使用RDF推論引擎推論[7]，但這些 Wiki 若要管理 RDF、OWL 與 Rules 都必須透過系統管理者編輯，因此如何設計一合宜頁面讓使用者藉由選擇 Fact 並搭配 Ontology 分享這些知識成爲系統關鍵。整合 Wiki 的內文主要是由 Wiki 系統內建的頁面完成，使用者僅將內文編輯與儲存，因此使用者所編輯的語法、內文、知識的整合是否有誤？這些步驟還是必須藉由對語意網充分瞭解的專家完成[17,18,27]，以現況來看只有對 RDF、OWL 較深入的使用者才有辦法對 object、subject、關聯性等進行設定，而且 Wiki 是一個公開互相編輯的平台，判別使用者設定是否正確將是無法避免的議題。

爲了讓使用者可簡易上手爲前提，本研究將新增偏好設定的頁面，讓使用者可以依照 OWL 現有的類別選擇現有的項目，在此以咖啡食材爲例，當中內文與 OWL 屬於不同檔案但有相互關係，使用者在新增頁面時同時會建立 OWL 當中的類別，有了 OWL 區分類別的不同與關聯，即可使用 Jena 先行推論，此時再把推論出的新知識與 rules 檔送給 rule 引擎推論，透過先前 OWL 檔可推論出些許替代食材，藉此可找出更多種可製作的咖啡成品，當中有些食材需要到特殊的地方去採買，此時可透過使用者的分享，有了基本介紹與店家地址後，可使用 Google Maps API 顯示地理資訊。因此在整體知識編輯上就必須透過管理者編寫 OWL 與 Rule 檔，讓使用者只需選擇現有食材，最後再結合推論引擎推論出更多的資訊。

四、GIS(geographic information system)

科技的發達，取代了早期紙本文件，就連地圖都可以電子化，常見的電子 GIS 系統有：Google Maps、Yahoo Maps、URMaps，而本研究所選擇的是 Google Maps。Google Maps 不僅在圖資系統上提供較完整資訊，還提供輸入所在地址或是使用經緯度來查詢地理情資，但是使用者只能看到圖片或是店家、景點並整合 Google Maps 的資訊[30]，論使用僅談的上便利。然而 Google 也提供了 Google Maps API 讓使用者可以在網頁上嵌入 Google Maps[8]，讓使用者在網頁上，或是 Web 環境上可以一邊講解訊息，一邊呈現地理情資，因此應用上更爲廣泛。

而本研究是透過 JSPWiki 呈現咖啡的知識，若需要特殊商品、耗材或是特殊器具時，早期的方法就是透過 Google 來搜尋資訊，如今透過 Wiki 的結合，本研究可以透過 Wiki 編輯店家資訊，並依照內文編輯的所在地使用 Google Maps API 呈現地理情資，並開放權限給所有註冊會員一起編輯與修改，此時不但可以增加 Web 2.0 資料互享與互相編輯的特色，還可以讓使用者更確切知道地理位置，也不需額外去搜尋。

參、系統結構

本研究之 Wiki 系統大致可區分爲三大塊，1. 推論引擎。 2. JSPWiki。 3. Google Maps(如圖 2 所示)。推論引擎主要是將 Profile 所產生的 Fact 與咖啡知識本體讀入後，先使用 OWL 推論引擎推論出可替換的食材與最多可能篩選的文章段落，最後再

將咖啡成品規則檔案讀入後，使用 Jena Rules 推論引擎將可製作的咖啡成品推論出來。

JSPWiki 是本研究主要呈現資料的頁面，無論是讓使用者可自行登入 JSPWiki 編輯頁面資訊或是勾選 Profile 給予適當的建議，若頁面資訊要呈現地理情資則可使用 Google Maps 呈現之。

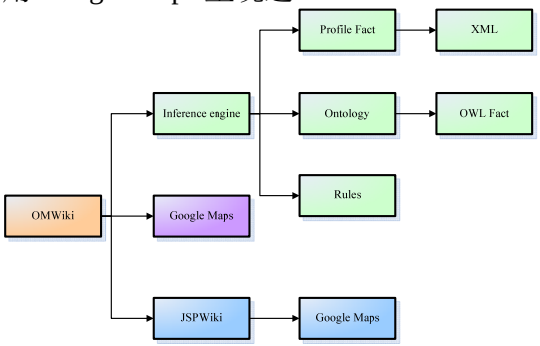


圖 2 OMWiki 系統結構圖

一、推論引擎

市面上常見的 OWL 推論引擎有：Jena、Pellet、RacerPro、FaCT，其中屬 Jena 在本研究使用上最爲便利，因著有自行的推論介面可將推論結果以 triple 格式輸出，本身也有 Rules 推論引擎可使用，因此不需外掛 Jess 或是其他推論引擎。因此在推論引擎部份大致可分爲三大項目：Profile fact、知識本體與 Rules。

(一) Profile

由於不同使用者會有不同的喜好與個人的現有資源，然而，這些資訊不只是某特定網站才可使用，若透過 XML 的定義與 RDFa 的銜接，也可將這些 Fact 對應到 OWL 中，如此 Fact 即可與 OWL 做串接，唯一要注意的是 Namespace 設置必須相同，否則推論出的資訊將會是 Fact 與 OWL 不同區塊的知識，因此本研究使用 RDF 與 RDFa 呈現 OWL 之 Fact 部分。

(二) Jena-OWL 與 Jena-Rules

Jena 推論引擎本身爲 Ontology 推論引擎，支援的 OWL 的語法，因此知識本體部分使用 OWL 呈現即可，Jena 支援 OWL DL 的推論引擎，在語法上的描述均無任何問題，支援 OWL 中所有的 Properties。以網頁階層來說，大致上可區分爲七層(如圖 3 所示)，由下往上來看最底層主要是描述一般網頁、內文或是網址的 Unicode 或是 URI，也就是早期 HTML，然而，單純描述網頁中的資訊可直接輸入內文，若是要將部分資訊相互交換或是最更多的應用，則必須制定出一標準讓使用者之間透過共同的格式編輯資料，也就是圖 3 中第二層的 XML、Namespace 與 XML Schema。到目前爲止都是人類可理解的電腦語言，以電腦的角度來看只是單純的 0 與 1，資料的交換與描述上可正確的運行，但是要透過電腦找出當中的特性與可利用之處還是尚缺規範，因此 W3C 制定出了 RDF 與 RDF Schema，藉由 Triple 關聯式將人類理解的語言包成 Object、Property 與 Subject 讓電腦可區分其差異，也就是圖 3 中的第三層。

當電腦有了自行的溝通能力後，使用者可藉此創造知識本體語言讓電腦協助推論，電腦的好處是所處理的資料量越大成效越明顯，因此使用者只須將資料依照電腦理解的格式傳給電腦，電腦可藉由推論引擎將大量的資料推論出來有用的資訊，也就是圖 3 中的第四層：Ontology Vocabular。然而，知識本體描述的資源有限，若要將較複雜的資訊描述必須使用 Rules 呈現， Rules 使用上可依照使用者的需求自訂 and、or、not 等邏輯敘述，也就

是圖 3 的第五層。有了這文件與知識後，需要一個平台來印證，本論文選擇 Wiki 系統，最後將推論引擎推論出的結果顯示於 JSPWiki 中即可，也就是圖 3 的第六、第七層。

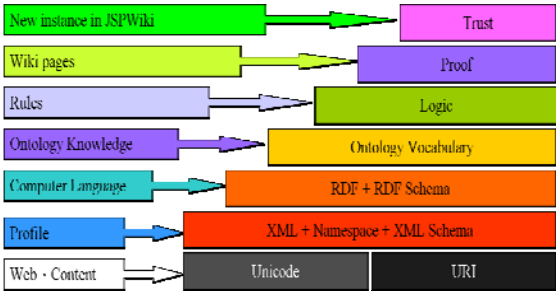


圖 3 系統結構階層圖

肆、系統實做

一、系統環境介紹

本研究環境是建構於 Windows XP 平台上，爲了讓系統可跨平台使用，因此本研究使用開放式平台 Java J2SE 5.0 版本，Java 伺服器則使用 Apache Tomcat 6.0 版本。開發工具主要是使用 Eclipse 3.3 版，Eclipse 不僅是 Java 的編輯器，也可除錯、設定 Tomcat 伺服器，也提供 JSP 頁面編輯功能。Ontology 描述語言是使用 OWL，因 OWL 架構於 RDF 上，因此每次判讀的類別只可爲兩個，同時判別的類別與規則有限，因此使用 Rule 引擎協助一次判別多個類別之推論，使用套件爲 Java 環境下的 Jena 2.5.7 進行 OWL 與 Rule 的推論，呈現的 Web 平台是 JSPWiki，透過 JSPWiki 讓使用者可以互相分享資料，假若有使用到地理情資部份則透過 Google Maps 呈現之。表 1 爲本研究所使用的環境介紹表。

本研究主要是以 Client-Server 方式進行運作，伺服器端必須具備 Tomcat server 來運行，依照 JSPWiki 所需必須使用 Tomcat 6.X 版本才有辦法進行系統開發的載入與

表 1 系統環境介紹表

	項目	使用
開發環境	作業系統	Windows XP
	Java	JDK 1.5.0_17
	Java IDE	Eclipse 3.3.1.1
系統環境	Semantic Web toolkit	Jena 2.5.7
	OWL Inference Engine	Jena 2.5.7
	Rules Engine	Jena 2.5.7
	JSP Server	Apache-Tomcat-6.0.18
	Wiki Engine	JSPWiki 2.8.1
	GIS	Google Maps API

設定，雖說 Tomcat 主導了伺服器環境，但整個系統主要還是透過 JSPWiki 當中的 Wiki 標記語言來呈現，當 JSPWiki-src 檔案下載並解壓縮後，可以看到有許多類別庫與 JSP 檔案，若針對 JSPWiki 系統頁面的 JSP 檔案來看，可發現有許多類似 XML 語法的 Wiki 標籤，這些都屬於 Wiki 本身的語法，執行時 Wiki 會有自己的剖析器先行分析與判別，最後使用 JSP 語法將資料撈出並呈現頁面資訊於瀏覽器上。

而本研究所呈現的 Wiki 在使用時異於一般的 Wiki 系統；使用前，使用者必須先到個人偏好設定頁面設定現有食材與預篩選的文章段落，當編輯完成會依照使用者名稱存成 Fact 檔；由於推論引擎推論出來的結果會依照 Fact 檔改變，若使用者沒有再更改 profile 則 Fact 不會隨意變動，因此只要把推論結果包成新 Fact 檔並於適當的頁面將新知識顯示出即可。因此當 Fact 檔產生後，系統會先行將該使用者的 Fact 檔

與 OWL、Rules 檔先行推論，並區分舊有的選項與推論出新找出的知識。而到每個頁面時會顯示該頁面的資訊，系統會去讀取使用者要過濾的段落資訊，此時系統會把使用者設定的選項與推論出的項目區分，若是有關咖啡食材才顯示於首頁頁面，若是顯示文章內容時，系統會先去判斷個人偏好，並判斷內文段落是否有需要被篩減，內文篩減後若有描述到地理位置則使用 Google Maps API 呈現，最後透過 Wiki 語法與 JSP 將結果回傳到使用者瀏覽器上顯示輸出結果。步驟如圖 4 所示，詳細說明如下：

1. 使用者連結至 Wiki 頁面，若是第一次使用則到偏好設定編輯個人資料。
2. 若使用者有更改個人資料則回存系統資料庫，否則準備個人 Fact 檔以便推論。
3. 取出 Fact、OWL 與 Rule 檔案。
4. 將 Fact、咖啡 OWL、咖啡 Rules 輸入至推論引擎先行推論，並把推論的新知識另外存成新的 Fact 檔以便區分。
5. 將新知識存起來，並判斷是何者頁面所需。
6. 於適當頁面顯示有關咖啡食材的推論結果，若有地理情資資訊則包成 Google Maps API 格式。
7. 透過 Wiki 頁面顯示 Google Map。
8. 將 Wiki 最後結果回傳至使用者瀏覽器。

二、Domain Ontology

本研究主要是呈現咖啡這塊領域的知識，透過 Wiki 資料共同編輯的方式，讓喜歡咖啡的使用者可以透過網頁學到更多的知識，有經驗能力的先進可以透過 Wiki 讓想踏入這塊領域的新手們分享許多豐富的經驗或讓試用者可以判別現有的材料可以讓咖啡有更多的變化，因此本研究主要建

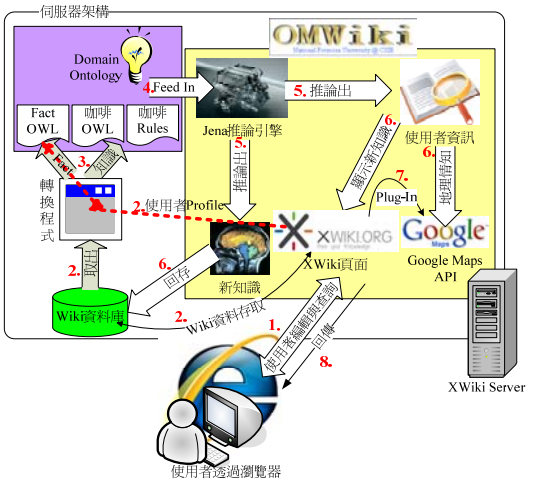


圖 4 系統架構流程圖

立了一個咖啡領域的知識本體。然而，咖啡這塊領域用到的食材、器具也較廣的，小至橘子皮、大至咖啡機都可以帶來許多變化，因此，在這塊領域上本研究定義了兩大類別：食品、器具，這兩類，食品這類知識本體主要描述：1. 咖啡成品，諸如：摩卡咖啡、義式咖啡、美式咖啡等這方面的成品。2. 咖啡食材，如咖啡豆、牛奶、糖水等調製咖啡所用到食品。器具這個類別描述泡咖啡會用到的機具或是耗材，依照個人喜好會有濾泡式咖啡機、摩卡壺、法式咖啡壺等機具，這些咖啡機具當中，最常見的耗材爲濾紙，濾紙有分大小、機具的種類、材質，適用的機具或是大小都可透過知識本體描述並加以區別。

當知識本體建置完成後則開始定義 Fact 與知識本體之間的資源定義，由於 Fact 是由 JSPWiki preference 頁面所產生的 Instance，這些實例與本研究知識本體中的類別有關係，但是在這邊的類別只能視爲對應到知識本體的物件，因此系統建置時必須考慮到所建置的實例選項必須正確對應到知識本體，否則無法推論出正確的結果。

```
<濃縮咖啡 rdf:about="&coffee;現有食材1"/>
<鮮奶 rdf:about="&coffee;現有食材2"/>
<奶泡 rdf:about="&coffee;現有食材3"/>
<發泡鮮奶油 rdf:about="&coffee;現有食材4"/>
<軟式發泡鮮奶油 rdf:about="&coffee;現有食材5"/>
<檸檬皮絲 rdf:about="&coffee;現有食材6"/>
<烤椰子絲 rdf:about="&coffee;現有食材7"/>
<柳橙皮絲 rdf:about="&coffee;現有食材8"/>
<糖水 rdf:about="&coffee;現有食材9"/>
<太妃糖粒 rdf:about="&coffee;現有食材10"/>
<砂糖 rdf:about="&coffee;對應食材1" />
<糖包 rdf:about="&coffee;現有食材11" >
  <砂糖替代 rdf:resource="&coffee;對應食材1" />
</糖包>
<冰糖 rdf:about="&coffee;對應食材2" />
<糖包 rdf:about="&coffee;現有食材12" >
  <冰糖替代 rdf:resource="&coffee;對應食材2" />
</糖包>
```

圖 5 使用者 Fact 程式碼圖

當實例定義時主要是以 RDF 格式進行建置，以簡易的方式來看就是<OWL 類別 rdf:about=”編號”/>，將所勾選的類別轉換成此種格式，最後再透過計數器將勾選的結果編號，而使用者所勾選的項目會變成如圖 5 所呈現的程式碼。

三、Rules

(一) Fact Rules

在使用 Rules 推論引擎之前必須先將知識本體先行推論，否則無法找出最多的新知識，透過推論引擎找出隱含的知識後，一般會先將新推論出有用的知識另行儲存，而本系統爲了減少推論的次數，將推論出有用的結果存至暫存檔，當使用者到首頁時才將有用的資訊顯示出來，然而，若要將這些有用的資訊包回原先 Fact 再行推論非常不便，因此本系統直接將 OWL 推論後的 model 與 Rules 推論引擎直接整合並直接推論出可製作的咖啡成品。

Fact 爲使用者勾選的現有食材與對應的食材進行編碼，然而這些編碼主要是定義使用者現有食材與對應食材的編號，使用 Jena 或 Pellet 推論引擎推論出最後的結果只是單純描述現有食材與對應食材之間的關係，如：(現有食材 1 替代對應食材 1)，這些資訊不僅對使用者毫無意義，對 Jena Rules 推論引擎也無幫助，因爲推論引擎不會去判別標號正確與否，因此定義 Rules 時只會比照現有食材去推論，省略了當中的編號，因此要先將推論出的新知中的編號移除後，再使用 Rules 推論引擎推論，而使用 OWL 推論後的結果不僅可以找出現有食材的替代品，推論引擎也把推論的對應食材給回覆了真實類別。因此，本研究將 OWL 推論引擎推論出的結果先使用下列兩條 Rules 先行推論：

[Rule1: (?a ?b ?c), (?a rdf:type ?d) -> (現有食材 rdf:type ?d)]

[Rule2: (?a rdf:type ?c) -> (現有食材 rdf:type ?c)]

首先先將 OWL 推論出的結果以 Rule1 先行推論，Rule1 中的 (?a?b?c), (?a rdf:type ?d)是對應到使用 OWL 推論引擎的 (現有食材 1 砂糖替代 對應食材 1), (現有食材 1 rdf:type 糖包)，藉此找出所有可能性；概觀來看此 ruls 其最大的用意是將 OWL 推論出可替換的食材，當中有包含一些編號，透過此 rule 可將對應食材以現有食材做替代，如此一來可將替換的現有食材直接納入 Rule 推論引擎推論。然而並非所有的選項都有對應到的食材或是與其他類別對應的關聯性，爲此，本研究另增加 Rule2 輔助當僅有描述單一 instance 也可將其結果推論出來，而此 rule 將會判別許多不可能的資訊爲現有食材，這些不必要的資訊不會與本研究定義的類別有直接關係，因此不會造成最後將現有食材進行推論的規則誤判。

(二) Domain rules

由於 OWL 主要是描述兩個類別間的簡易關聯，若要描述多個類別與之間的關聯性必須使用 Rules 方式來描述，因咖啡成

品的發源地、特色與食材而異，製作一杯有特色的咖啡將會用到數種不同的食材，也因此若要描述與判斷現今的食材可否作出合宜的咖啡成品則使用 Rules 描述，假設要泡製皇家火焰咖啡，準備的食材有：熱咖啡、方糖、白蘭地酒、奶油球，若用 OWL 則無法描述，因此本研究使用 Jena 內有的 Rules 來呈現。

由於 Rules 只會判定是否符合條件，而不會自將篩選或判斷編號，若未先使用前面兩條 Rules 推論則會找不到對應的關係，因爲無法確定當下是第幾個現有食材與替代食材，當使用者在 Preference 項目中若有勾選熱咖啡、奶油球、方糖、白蘭地酒這四項食材時，將 OWL 檔、Fact 檔與 Rules 檔案送到 Jena 推論引擎，即可依照現有食材推論出可製作出皇家火焰咖啡。本論文也摘錄了市面上 80 種咖啡成品[1]，當中的 80 種咖啡與相對應的食材都透過 Jena Rules 來描述。全部程式碼如附件所示。

四、Jena 推論引擎

本研究在推論時分別會使用 Jena 的 OWL 與 Rules 推論引擎，由於 Jena 是在一般的 Java 上使用，若用在 JSP 上必須使用 Java bean 方式呼叫，因此無論要使用 Jena OWL 推論引擎或是 Jena Rules 推論引擎都必須透過 Java Bean 方式呼叫才可。

(一) OWL Inference

本研究在推論的時候會先讀取原先設計的 OWL 檔，而後透過 Jena 推論，Jena 推論流程如圖 6 所示，步驟如下：

1. 使用 Jena 的 Model 設置一 Schema 並將 OWL 讀取進來。
2. 使用 Jena 的 Model 設置一 Data 並將 Fact 讀取進來。

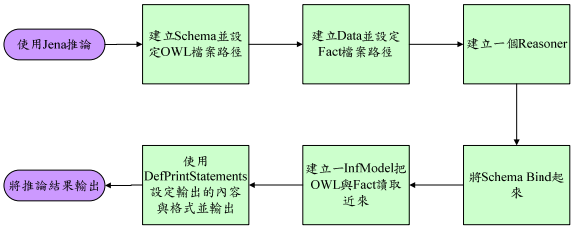


圖 6 Jean 推論引擎實做流程圖

3. 建立一 Reasoner 並設定一暫存器存取 OWLReasoner。
4. 將 Schema 匯入 Reasoner。
5. 建立一 InfModel，並餵入 Reasoner 與 data。
6. 使用 DefPrintStatements 將所產生的結果輸出，在此可設定要輸出的 resource 或是 properties，在此設定全部輸出。

由於 Jena 會把推論出的結果以他自己的格式顯示，並且會把所有有關 RDF 跟 OWL 的資訊一併推論出，由於當中有過多的冗餘資訊，故在顯示時可以先行過濾，最後在依照使用者需求設定所需的格式或是介面，因此本研究會將推論出的結果口語化，並使用 JSP 語法呈現在 JSPWiki 上。

(二) Rule Inference

使用 Rules 推論引擎前必須要先定義 rules 檔，以 Jena 推論引擎來說有兩種方式可載入，最簡單的是建立一字串定義規則，或是先將 rules 使用純文字檔儲存，推論前先使用 filereader 將檔案讀進來即可。由於本研究主要是在公開系統上進行推論，因此規則檔必需隨時可以撰寫，因此本系統自行寫了 fileread()函式，推論前先行讀取即可，隨後建立一 rules 剖析器讀取

rules，此時會先行驗證當中的 rules 格式是否正確或不正當定義，若無問題則建立 schema，並設定待推論的 ontology 路徑，此時必須設定 setDerivationLogging 爲 true，否則無將結果推論出，隨件建立一 OWLReasoner，並使用指令將 Reasoner 與知識本體、fact 使用 bind 整合，此時建立一 InfModel 將 OWL 與 fact 讀取，再建立一 InfModel 把 OWL 推論引擎或是前一 Rules 推論引擎與 Rules 匯入，最後即可將推論的結果輸出(流程圖如圖 7 所示)。

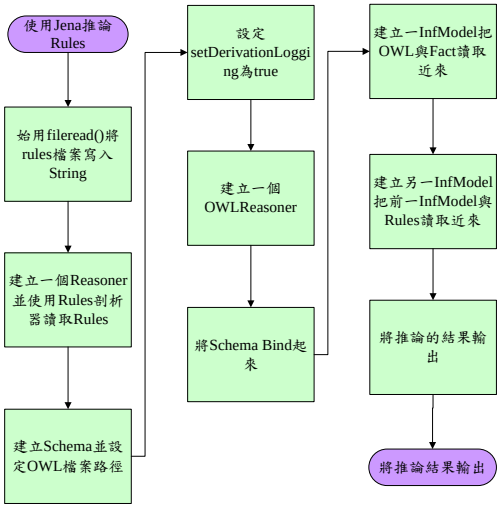


圖 7 Jena rules 推論引擎實做流程圖

本研究撰寫 JSPWiki 整合推論引擎爲例，首先先把 Rules 檔案使用 FileReader 讀取進來後，新增一個 reasoner2 整合 rules 剖析器並匯入 rules，使用 Model 匯入知識本體與 Fact，最後將 reasoner 的 setDerivationLogging 參數設定爲 true，此時將建立 reasoner 暫存器並將 reasoner 與 fact 讀入，最後將 reasoner2 與 infmodel 匯入新的 InfModel 中，此時即可將最後推論出的結果顯示出來。

五、Google Maps API

若要使用 Google Maps API 時，必須先到 Google Maps API 的頁面登入申請驗證的密碼，而後依照 JavaScrip 或 HTTP 等方式將密碼與參數以網頁的語法進行編輯，此時即可呈現在所需的頁面上。而本研究所使用的 Java Base 的環境，因此在開發的時候使用 JavaScrip 的方式進行編輯。如果是在 JSPWiki 上可以透過 Plugin 的方式將 Google Map 一同呈現。

本研究在撰寫時，必須使用特殊符號作爲 Google Maps 語法，因此必須在內文判斷出是否有碰到特殊符號，若有則擷取當中的文字做爲關鍵點輸出，而 JSPWiki 內文的判別是在 InsertPageTag.class 檔中進行修改。

六、系統呈現

本研究爲了不失 Wiki 的特色，讓使用者可使用原先的編輯與搜尋功能，因此本研究頁面如圖 8 所示，頁面框架還是以 JSPWiki 呈現。使用者在使用時，可以先到偏好設定選擇現有的食材與喜好設定，若儲存後會包成 Fact 檔，系統在執行一般頁面時會先讀取 Fact 與 OWL 先行推論，此時會先把現有食材與 OWL 當中的替代食材推論出，找出更多食材後，再使用 rule 引擎依照現有食材推論出更多種可製作的咖啡。依照推論結果可點藍色超鏈結部分，即可看到 Wiki 內文，由於文章經過眾多使用者撰寫，有些訊息並非使用者所喜好，此時藉由剛剛推論出的 OWL 檔判別出所有不需要的文章類別，若有碰到相同的標題則忽略顯示。

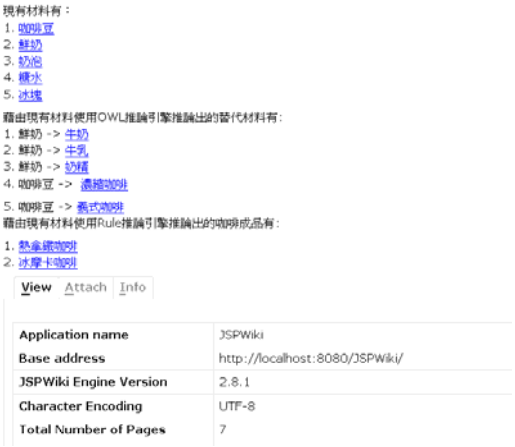


圖 8 本研究系統頁面圖

伍、系統評估

一、系統特色

本研究主要是藉由 JSPWiki 平台將 OWL、Rules 與推論引擎建置在系統中，改良早期單純的只有文字描述的功能，使得文章標題可視爲類別，透過 OWL 定義類別與之間的特性，藉此建立文章間的特性橋樑，甚至可透過 OWL 推論引擎推論出對使用者有意義的資訊。

因而原先系統在使用者資訊上大多是與使用者直接相關的訊息，諸如使用者名稱、帳戶資訊、聯絡方式，但是在使用者喜好上毫無任何描述，因此本研究藉由 JSPWiki 的 preference 頁面新增與該領域相關的資訊，以本研究爲例是描述咖啡食材，使用者可勾選現有的食材與預篩選的頁面字詞，藉由 OWL 描述類別間的關係。透過 OWL 推論引擎，可將使用者現有的食材找出所有可替換的可能性，並將使用者預篩選的字詞透過 OWL 的等價描述找出語言描述間可能替換的字詞，這些使用者資訊其實可重複使用，但是大多的 Profile 都是使用資料庫作存取，爲此，本研究於該領域制定的 Profile 儲存成 XML 檔案，使用者若有需要可重複使用。

此時，已經有了咖啡食材，可透過 Rules 描述非單品的咖啡，藉由 Rules 邏輯描述複雜的關聯性，而在 Rules 描述咖啡成品的時候可以依照最簡易或理想的食材進行描述，最後將使用者勾選的現有食材與推論引擎推論出的現有食材餵給 Rules 推論引擎進行推論，即可得知最可製作最多種類的咖啡成品。

由於許多咖啡食材須經由特殊管道才可所取，因此當使用者要描述地理情資資訊時，可透過 Google Maps 呈現，由於一般的 Google Maps API Plug-in 僅支援經緯度的描述，若要描述地址則必需先行轉換。爲此，本研究直接修改 Wiki 語法，只需透過簡易的指令描述即可藉由 JSPWiki 輸入地址或是特殊標的物顯示 Google Maps。

二、推論引擎比較

市面上有許多免費與付費的推論引擎，以 Java 來說最常見的就是 Jena、Pellet、RacerPro 推論引擎，而這些推論引擎有各自的優缺點，Jena 本身有內建的推論引擎、Pellet 在做一致性的驗證最快，RacerPro 是以 client-server 方式進行連線與推論，因研發單位的性質不同所開發的軟體各有優缺點。而本研究是比較 Jena 與 Pellet 這兩款推論引擎讀取 OWL、Fact 檔並推論之，爲了公平起見，比較時以推論 10 次後的平均值計算。

而所推論的輸入範例文件有三份，分別爲描述 Learn Ojbect Metadata 的 LOM 知識本體、W3C 撰寫的 Food 知識本體與本研究所撰寫的 coffee 知識本體，而下表示

表 2 知識本體使用 OWL、RDF 比較表

	LOM	Food	Coffee
owl:Class	64	94	180
rdfs:subClassOf	19	105	121
owl:Restriction	0	152	0
owl:onProperty	0	152	0
owl:minCardinality	0	4	0
owl:intersectionOf	0	24	0
owl:hasValue	0	59	0
owl:allValuesFrom	0	89	0
owl:oneOf	0	2	0
owl:Thing	0	16	0
owl:ObjectProperty	18	4	9
rdfs:domain	18	4	21
rdfs:range	45	4	18
owl:unionOf	0	1	0
owl:sameIndividualAs	0	12	0
owl:disjointWith	0	39	0
owl:SymmetricProperty	0	0	8
owl:inverseOf	0	0	5
Fact	17	33	25
總計	181	794	387

統計各知識本體所使用的 OWL、RDF 之項目統計(項目比較如表 2 所示)。

推論時使用最簡易的方式將所有推論結果撈出來，並統計最後執行時間，以下將一一探討之。

(一) 使用 Jena 推論引擎推論 OWL

使用 Jena 推論引擎後，推出的訊息無論是關於 OWL、RDF、所定義的知識都會推論出來，然而這些資訊大多是與使用者無關的訊息，因此使用者必須自行過濾其結果，當中資訊量非常大，因此可發現第一次推論時間與之後的推論時間無多大差異，屬於是比較平均的將結果推論出來(詳如表 3 所示)。

這三份知識本體中屬 LOM 最簡單，主要是定義類別間的關聯性，因此推論的時間較短，平均時間爲 985 毫秒。其次是

表 3 Jena 推論咖啡知識本體時間統計表(單位：毫秒)

知識本體	LOM	Food	Coffee
Jena 第一次推論時間	2203	95766	10125
Jena 第二次推論時間	937	93906	8453
Jena 第三次推論時間	829	96765	8578
Jena 第四次推論時間	843	92079	8531
Jena 第五次推論時間	828	90046	8360
Jena 第六次推論時間	922	92485	8719
Jena 第七次推論時間	844	94547	8328
Jena 第八次推論時間	813	92640	8328
Jena 第九次推論時間	828	90891	8328
Jena 第十次推論時間	812	92172	9109
Jena 十次平均推論時間	985	93129	8685

Coffee 知識本體，描述了些許類別與子類別的關聯性，可發現 Jena 推論類別量較大的知識本體非常吃力。Food 知識本體描述的知識量較適中，以 16KB 檔案大小的知識本體來說，JVM 記憶體要設定成 1024MB 上下才可以推論，平均時間爲 93129 毫秒，十分不理想。以 Jena 來說，類別與關聯描述的多寡將會影響推論時間，描述越複雜速度越慢。

(二)使用 Pellet 推論引擎推論 OWL

由於 Pellet 主要是使用可使用 DIG Reasoner 或是 Jena Reasoner 的介面，而本範例是使用 Jena 作爲輸出的介面，Pellet 主要使用 OntModel 建立推論模組，最後使用 model.read()將 OWL 與 Fact 匯入後，即可將當中 OntModel 推論的結果推論出來。可發現當中的資訊都是與咖啡知識定義的類別有直接關係的知識。

推論時間比較，可發現第一次讀取推論的時間最久，之後的時間都非常短，其原因是 Pellet 會將推論後的資訊存放於記憶體中，若沒有更改則直接將推論結果輸出。此時可發現，Pellet 推論時間較不受類別數量大小影響，增加推論時間主要是受到描述類別間的關聯性所影響(詳如表 4 所示)。

(三) 推論引擎綜合比較

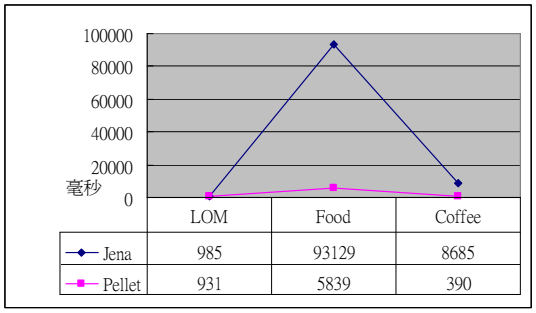
因系統的需求不同，推論引擎的選擇也是非常重要的，以本研究爲例，主要是塑造一整合 OWL 與 Rules 推論引擎之 Wiki 平台，在效能上無太大要求，主要是要支援 Rules 功能與可正常顯示即可，因而選擇開發較迅速的 Jena。本研究也藉此比較這兩款推論引擎，其比較圖如下，可發現 Pellet 第一次與第二次推論時間差異甚大，其主要原因是 Pellet 推論前會先進行 Classification 動作，藉此將知識本體的類別與子類別的關聯性使用階層的方式存至記憶體中，因此，只要記憶體沒被釋放這些資料還是存於記憶體中，因此第二次以後的推論時間快上許多，Pellet 還可藉 Classification 動作回覆使用者的詢問，因此效能會更加顯著，加上數據的顯示(如圖 9 所示)，以推論引擎支援 OWL 程度來看，使用 Pellet 推論的效能將比 Jena 來的好。

陸、結論

社會的演進在於資訊而非科技，資訊的成長是有許多人力支出在提升新的知識呈現，傳統產業並非與科技無關，而是無法很直接與現今的科技結合，本研究希望藉由可分享新知的 Web 2.0 系統提升國人在傳統領域上可以有更多的突破，藉由 Wiki 與知識本體描述前人累積的經驗，往後就不需要再去花費資源去做不必要的測試，甚至可以將不足的部份改良。因此希由此系統，藉由咖啡領域進行示範如何將成品與半成品予以描述，最後透過先進的

表 4 Pellet 推論咖啡知識本體時間統計表 (單位：毫秒)

知識本體	LOM	Food	Coffee
Pellet 第一次推論時間	1956	7703	1828
Pellet 第二次推論時間	328	5937	297
Pellet 第三次推論時間	297	5782	265
Pellet 第四次推論時間	390	5750	219
Pellet 第五次推論時間	313	5406	297
Pellet 第六次推論時間	250	5234	219
Pellet 第七次推論時間	312	5735	171
Pellet 第八次推論時間	344	5390	204
Pellet 第九次推論時間	266	6031	218
Pellet 第十次推論時間	297	5422	188
Pellet 十次平均推論時間	475	5839	390



	LOM	Food	Coffee
Jena	985	93129	8685
Pellet	931	5839	390

圖 9 Pellet 與 Jena 推論平均時間表

科技與電腦的運算得知有用的資訊。

本研究使用 JSPWiki 結合語意網、推論引擎與 Web 2.0 的工具，這些工具是隨手可得的開放式軟體，建構系統時主要在於 Wiki 是否可以呈現使用者所需要的資訊，由市面上正紅的維基百科評估，是套讓使用者可以輕易將資訊分享的系統，但是若要描述知識會有力不從心的感覺，甚至在文章中常常會發現有許多冗餘的資訊，因此本研究結合推論引擎加以修改，由於主要的推論引擎是由 Java 開發，因此本研究大多使用 Java 的開放式軟體進行開發。經過多番測試與比較後，JSPWiki 在開發與使用上較穩定；以知識方面的呈現 OWL 是缺一不可的，但是所能描述的關聯性卻不多，若要判別較複雜的規則必須搭配 Rule 呈現，而 Jena 同時支援 OWL 與 Rules 的推論，因此本研究選定了 Jena 最爲推論核心，在系統整合上僅需使用 JavaBean 即可完成；最後再透過 Google Maps API 呈現地理情資，如此即可滿足大部分使用者在特殊領域上的呈現與找尋有用的資訊。

本研究的 OWL 均是事先撰寫好，但實際上使用者也會有新的知識要去編輯與修改，雖然是以 Wiki 系統，在內文的新增修改 Wiki 本身會去把關，但 Wiki 如何去編輯 OWL 是門很重要的課題，如何修改程式讓使用者編輯 OWL 檔時不會破壞原先的內容並在適當的地方增加新知識與描述，並適時做 OWL 與 Rules 檔案版本控管，這將會影響整個系統的穩定性。未來可針對如何透過網頁讓使用者編輯新的知識與 Rule 檔案，讓系統更加人性化。

參考文獻

[1] 邱偉晃，「咖啡教科書」，大境文化，台北，民國 97 年 6 月。

[2] Beckett, D.,“RDF/XML Syntax Specification (Revised),” http://www.w3.org/TR/rdf-syntax-grammar/.

[3] Bray T., Paoli J., and Sperberg-McQueen C. M., “Extensible Markup Language (XML) 1.0,” http://www.w3.org/TR/REC-xml.

[4] Brickley D. and Guha R. V., “RDF Vocabulary Description Language 1.0: RDF Schema,” http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

[5] Carroll J., De Roo J., “OWL Web Ontology Language Test Cases,” http://www.w3.org/TR/owl-test/.

[6] Dean M., Schreiber G., F. van Harmelen, Hendler J., Horrocks I., McGuinness D. L., P. F. Patel-Schneider, L. A. Stein, “OWL Web Ontology Language Reference,” http://www.w3.org/TR/owl-ref/.

[7] Ebert, C., Consulting, V., “Semantic Wikis,” *IEEE internet computing*, 2008, pp. 8-11.

[8] “Google Maps API,” http://code.google.com/intl/zh-TW/apis/

maps/.
[9] Grant J., Beckett D., “RDF Test Cases,” http://www.w3.org/TR/rdf-testcases/.
[10] Hayes, P., “RDF Semantics,” http://www.w3.org/TR/rdf-mt/.
[11] Heflin, J., “Web Ontology Language (OWL) use cases and requirements,” http://www.w3.org/TR/webont-req/.
[12] Heo, S., Kim, E., “A study on the Improvement of Query Processing Performance of OWL Data based on Jena2,” *IEEE COMPUTER SOCUETY*, 2008, pp. 678-681.
[13] Herman, I., “Web Ontology Language (OWL),” http://www.w3.org/2004/OWL/.
[14] Hsu, I., ”Yuan Kwei Tzeng; Der-Cheng Huang, OWL-L: An OWL-Based Language for Web Resources Links,” *Computer Standards & Interfaces*, 2008.09.29.
[15] Hsu, I., Chi, L., and Bor, S., “A platform for transcoding heterogeneous markup documents using ontology-based metadata,” *Journal of Network and Computer Applications*, 2008.07.6.
[16] “Jena,” http://jena.sourceforge.net.
[17] Jing, H., Fan, Y “Usability of Wiki for Knowledge Management of Knowledge-Base Enterprises,” *Knowledge Acquisition and Modeling*, 2008, pp. 201-205.
[18] Kawamoto, K., Mase, M., Kitamura, Y., Tijerino, Y., “Semantic Wiki Where Human and Agent Collaborate,” *Web Intelligence and Intelligent Agent Technology*, 2008, pp. 147-151.
[19] Klyne G., Carroll J., “Resource Description Framework (RDF): Concepts and Abstract Syntax,” http://www.w3.org/TR/rdf-concepts/.
[20] Liebig, T., “Reasoning with OWL System Support and Insights,” *Ulmer Informatik-Berichte*, 2006.
[21] Manola F., Miller E., “RDF Primer,” http://www.w3.org/TR/rdf-primer/.
[22] McBride, B., “Jena: A Semantic Web Toolkit,”*IEEE Internet Computing*, vol. 6, no. 6, November/December, 2002, pp. 55-59.
[23] McGuinness D. L., Harmelen, F., “OWL Web Ontology Language Overview,”
[24] Min, W., Jianping, D., Yang, X., Henxing, X., “The Research on the Jena-based Web Page Ontology Extracting and Processing,” *SKG*, 2005.
[25] Patel P. F., Hayes P., Horrocks I., “OWL Web Ontology Language Semantics and Abstract Syntax,” http://www.w3.org/TR/owl-absyn/.
[26] Peter F. D., “Management Challenges for the 21st Century,” Harpercollins, 1999.
[27] Priedhorsky, R., Terveen, L., “The Computational Geowiki: What, Why, and How,” *CSCW*, 2008, pp. 93-98.
[28] “RDfa,” http://www.w3.org/TR/xhtml-rdfa-primer/.
[29] Smith M. K., McGuinness D., Volz R., Welty C., “OWL Web Ontology Language Guide,”

http://www.w3.org/TR/owl-guide/.

[30] Ullman, A., Kay J., “WikiNavMap: A Visualisation to Supplement Team-Based Wikis,” CHI 2007, pp. 2711-2716.

[31] Yan, Y., Zhang, J., Yan, M., “Ontology Modeling for Contract: Using OWL to Express Semantic Relations,” *EDOC*, 2006.