國 立 交 通 大 學

管理學院（資訊管理學程）碩士班

碩 士 論 文

非正規化知識本體重構

Informal Knowledge Ontology Refactor

研 究 生：陳良彥
指導教授：李永銘 博士

中 華 民 國 103 年 7 月

非正規化知識本體重構
Informal Knowledge Ontology Refactor

研 究 生： 陳良彥　　Student： Liang-Yen Chen

指導教授： 李永銘 博士　　Advisor： Dr. Yung-Ming Li

國立交通大學

管理學院(資訊管理學程)碩士班

碩士論文

A Thesis

Submitted to Institute of Information Management

College of Management

National Chiao Tung University

In Partial Fulfillment of the Requirements

For the Degree of

Master of Science in Information Management

July 2014

Hsinchu, Taiwan, the Republic of China

中 華 民 國 103 年 7 月

非正規化知識本體重構

研究生：陳良彥　　指導教授：李永銘 博士

國立交通大學管理學院（資訊管理學程）碩士班

摘要

本研究提出一個基於半非正規化知識模型的知識漸進式演進循環架構，幫助個人與組織理解與重用所面對的大量知識，該架構有四個面向，包含知識載體、知識重構、知識模型與知識推薦。知識載體敘述知識為何需重構，知識重構說明如何透過知識重構將知識載體轉化成為知識模型，知識模型則整理了一個知識模型應該保存哪一些知識，知識推薦介紹該如何重用知識模型。實驗部分則透過平台的實做驗證知識元件的效益，包括了可閱讀性，可維護性，可延伸性，完整性與關連明確性等五個要素進行分析。最後在結論的部分提出實務上運用的說明。

關鍵字：知識本體，知識重構，知識推薦，知識模型，知識管理

Informal Knowledge Ontology Refactor System

Student：Liang-Yen Chen　　Advisor：Dr. Yung-Ming Li

Master Program of Institute of Information Management
College of Management
National Chiao Tung University

Abstract

This paper proposes a knowledge progressive evolution circulation framework, based on semi-informal knowledge model, to help individuals and organizations understand and reuse a lot of knowledge they face. The architecture has four dimensions, including knowledge carrier, knowledge refactoring, knowledge recommendations, and knowledge model. Knowledge carrier describes why we need to do refactoring. Knowledge refactoring illustrates how to transforms knowledge carrier into knowledge model. Knowledge model catches the main knowledge to be saved. Knowledge recommendation introduces the approach how to reuse the knowledge model.

Experimental verification is done by implementing a knowledge component platform. Five performance factor evaluation, including readability, maintainability, extensibility, and clarity, were analyzed. Finally, the practice of the proposed framework are described in the conclusion.

Keywords：Knowledge Ontology, Knowledge Refactor, Knowledge Recommend, knowledge Management.

誌　　謝

這個論文的產生，可以說是一波三折，歷經了兒子的出生與肝功能的異常，讓我不得不放棄在入學的第二年及時的完成。在晚了一年後，藉由許多人的幫助，最後能讓本論文得以完成，一路上真是備感艱辛與感恩。

首先感謝家人的支持，沒有家人的幫助，我根本沒有辦法把時間抽出來好好的進行論文的撰寫，每次兒子在哭鬧的時候，我一點建樹都沒有，只能夠對著電腦猛敲論文內容，感謝家人的包容，讓我可以當一個失職的爸爸，放下這些事情進行研究。

再來感謝李老師的提點，當公司的事情一多，最後就把論文的事情暫時的拋出腦海之外，還好老師有請學弟來叮嚀與關心，不然這個論文完成的時間點可能又是下一個學期的事情了。

在論文進行的過程當中，也謝謝李老師與發哥還有智華學姐的建議與指導，讓我可以看到我所不足的部分，雖然在有限的時間中沒有辦法完成所有的建議，但是還好最後能夠讓整個論文有一個完整的架構。

在實驗系統開發中，感謝彥丞學弟給予 CSS 上面的指導，特別感謝我的夥伴英蘭的耐心等候與鼓勵，我知道我在系統開發上面花了太久的時間，讓妳等超久才有辦法進行口試，這點我感到很抱歉，也很謝謝妳的鼓勵，讓我在開發到很煩躁的時候，能夠繼續下去。

在實驗初期感謝渝婷學妹的鼎力相助，如果不是妳的幫忙建立初始資料，我的實驗就沒有辦法開始進行。進行的過程當中，特別感謝我的太太珮翎，謝謝妳的幫忙，讓更多人願意來參予這個花超級多時間的實驗，學校裡面該放假的放假，出去玩的出去玩，公司的同事下班之後，看到那麼久的實驗時間，整個興致都沒有了，沒有妳的話，一個禮拜內我可能連三十個人都湊不到，真的非常的感謝妳。

一路上謝謝大家的幫忙，讓我可以在今年的暑假，順利完成口試取得碩士學位，再一次謝謝一路上幫忙的各位。

良彥 2014.7.26

目錄

表目錄

圖目錄

第一章、 緒論

1.1 研究背景

1.1.1 資訊爆炸的時代

地球村這個名詞，代表著距離這一個因素，逐漸的被各種網路所取代，包括交通網路，社交網路，網際網路…等，這樣子的環境不但會加速整個人類社會的進步，但更無法忽視的是，這樣子的環境代表著更加強烈的競爭會迎面而來，不管自己願意不願意，適者生存，不適者被淘汰，影響的不僅僅是只是個人，更包括了公司與國家。

根據(Hilbert & Lopez, 2011)的研究，在 2011 年世界上流通的資訊約為兩百億兆，分配給每一個人相當於一個人一天閱讀 174 份報紙的數量，而且這個數字只會越來越快速的增加，因此使用者可以看到諸如 BigData，Data Mining…等顯學當道，這些事實無疑的凸顯資料爆炸性的成長，整個人類社會都在適應這樣子的變化，試圖能在這樣子的環境中，找到更加有效方法來運用這些資訊，發現更有價值的資訊。

對於個人而言，當面對這麼多資訊的狀況下，如何有效的組織與內化變成個人的知識，這個議題就會變得相當重要，萬一沒有辦法有效的轉化，接受外在的知識，在面對競爭的時候就沒有辦法取得優勢。這種其況尤其出現在知識密集的行業，知識密集行業的工作者，要面臨的比一般人多上許多的各式各樣知識，而且一旦停止吸收新知識，就意味著失去了競爭力，一旦學習緩慢，就慢慢與流行的開發環境脫節。像是程式開發人員，終生需要學習新的語言以及新的開發方法，不然沒有辦法持續待在這個行業。

對於公司而言，如果沒有妥善地保管與運用組織的知識就無法空出資源進行創新，反而會一直無意義的使用不同的方法完成同樣的任務，好的方法與結果沒有被保留，會隨著人員的流動而遺失。而沒有辦法創新的公司就只會有普遍性的產品，慢慢的相同產品的公司越來越多，就會變成削價競爭的產業，面對削價競爭的窘境，總會有新的發展中國家能夠提出更低的報價，導致公司無法繼續生存

下去。

對於國家而言，當一個國家而言，個人的成功會讓整個社會進步，公司的成功會讓經濟穩定。一旦國民失去競爭力，最好的人才都沒有創新而是只有付出勞力維生，公司失去競爭力，當產業無法轉型成為知識密集的產業，永遠只有是毛三到四（毛利率 3%到 4%）的產業，沒有利潤的產業就會使得員工薪資無法調升，反而不斷的隨著削價競爭而不斷的下降，當一個國家內員工薪資不斷的降低到無法留住員工的程度，許多優秀的人員就只能去當外勞。萬一國家變成這個樣子，使用者對於這個國家還能有什麼信心與期待？因此研究如何加強對於資訊的運用，無疑是一件非常重要的課題。

1.1.2 知識搜尋與建構的學說

知識搜尋與建構的學說可以包含了下列模型，經典模型、養珠理論、動態模式、訊息覓食與常識建構，整理如下表。

表 1 知識搜尋與建構的學說

模型名稱	相關研究學者	模型說明
經典模型	(Robertson, 1977)	當有資訊需求的時候，人們會把這些資訊需求轉化成為關鍵字，並且與文件的表現形式進行配對，如果符合，該文件就是所要尋找的資訊。
養珠法	(Ramer, 2005)	就如同養殖珍珠般，根據需求先找出一篇好的文章，接著透過這一篇文章，繼續往外找尋適合的知識，例如 Wiki 的文章有很多引用，要了解一個概念可以透過其引用，往外學習到一整群的知識。

動態模型	(Bates, 1989)	又稱採野莓理論，說明當人閱讀完搜尋時候所找到的資訊，就會因對於為這一份資訊的理解，增加了新的詞彙的了解或是新的概念，因此原本的搜尋條件就會不在適用，所以每當新閱讀一份文件，就會不斷的改變自己追尋的條件，所找尋的資訊也是隨之變化。
訊息覓食理論	(Nielsen, 2003)	覓食理論最早是由 Robert Mac Arther , Eric Pianka,在 1966 年提出，其內容為解釋動物在遷徙的時候，會借由本能評估哪邊是食物最多的地方來作遷徙。後來 Pirolli ,Card,在 1999 年的時後提出訊息覓食理論，認為人在找尋資訊就如同動物的遷徙，會自動往最多 Trigger words 的內容進行找尋，Trigger words(Nielsen, 2003)則是該資訊載體中與搜尋知識群組目標符合的關鍵字。
常識建構	(Pirolli & Card, 2005)	由 brenda dervin 在 1993 年提出，描述人們如何將新知識融入就知識中，後來 Pirolli ,Card 整合了上述的理論，認為常識的建構分為六大步驟： 1. 搜尋資料：對於特定主題搜索相關聯資料。 2. 放入鞋盒：蒐集相關的知識載體。 3. 評估：在蒐集相關資料會進行理解與評估。 4. 建立架構：對於有用的資訊，建立其架構。 5. 訂定假說：對於整理的架構提出新的論點，並進行驗證。 6. 呈現：以外顯形式，再將證明過的知識進行呈現。

可以總結上述模型，常識建構前四步是知識的搜尋與整理，後兩步驟為知識的融合與創造，本論文提出問題在於知識的搜尋與整理，因此就前四步驟進行討論。

人們對於收集資料會依據所了解的詞彙與知識進行經典模型的搜尋，並透過訊息覓食也就是 Trigger words 加速找尋的過程，了解的詞彙與知識在評估的過程當中，就如同養珠一般不斷的成長，最後由於這些詞彙與知識的進步增加，使用者會如同動態模型般，改變評估的標準，再次進行搜尋，放入鞋盒，評估的循環，直到資訊需求被滿足為止。

1.1.3 知識搜尋與建構的問題

Google 這個字不僅僅是一家公司，作為搜尋的代表，這個字已經被牛津與韋伯斯特辭典所收入，其意義代表「使用 Google 搜尋引擎在網際網路上獲取資訊」（to use the Google search engine to obtain information on the Internet），這也反映了日常生活中，使用者找尋資訊的方法，通常可以分為下列幾個步驟：

1. 想一些關鍵字，使用 Google 進行搜尋，然後對於搜尋結果進行閱讀。
2. 如果這個結果是有用的，對於一般使用者可能會使用 Evernote，Scope.it 或是 Pinterest 等軟體服務，對這些文章進行儲存，並為其加上關鍵字與摘要；如果是一個研究生，就會使用 EndNote 的軟體將搜尋結果進行儲存。
3. 最後當使用者為了某些目的要使用這些知識的時候，使用者會透過關鍵字或著是全文檢索來使用。

回頭來看知識搜尋與建構的過程，很容易會發生下列問題

- **初學者迷航與重工**

  根據(Jenkins, Corritore, & Wiedenbeck, 2003)的研究，初學者與非初學者差異在於 1.字彙的數量，2.評估的效率，3.則是探索的能力，非初學者是使用先深後廣，而初學者則是先廣後深。初學者對於所要搜

尋概念的字彙數不夠，導致使用者剛開始要搜尋與建構資訊時，動態模型與養珠法的運作，會讓評估的標準也就是正確的關鍵字與不斷的改變。找尋正確的關鍵字就有如迷路一般，因此稱之為初學者迷航。當評估標準改變，初學者對於原本已經閱讀過的知識載體變成又要重新閱讀。同一份知識載體可能會被閱讀無數次，則稱之為使用者重工。舉個例子來說像是做論文的研究，原本透過閱讀參考所找到的文獻，將其結論納入論文的結構當中，但是隨著論文研究的延伸，這時需要整理所有的研究方法，使用者就會將原本閱讀過的論文在重新拿出來做閱讀與理解。

- **評估的結果遺失**

不是每一份知識載體的品質都是好的，閱讀一份品質不良的知識載體需要花費大量時間，通常使用者都只有儲存當下評估所需的部分，而其他的部分就只存在在記憶裡面，但人的短期記憶保存時間有限，當再度評估同一份文件時，可能早就遺忘之前評估的內容，必須重新閱讀原本的知識載體。舉例以論文的撰寫來說，撰寫兩三個月是常有的事情，會因為前述的初學者迷航與重工而必須再次對原有的論文進行閱讀，對於不熟悉的內容，在經過一個月之後，對於之前所使用到的部分有整理到自己的論文當中，但是沒有整理到的部分就會忘的差不多了，如果要再進行使用，則必須進行重新閱讀。

- **知識模型的貧乏**

使用者儲存知識時，通常都只有使用關鍵字與摘要對於整個知識載體進行描述，當使用者需要再度評估這一份知識載體時，關鍵字與摘要幾乎無法直接重用，使用者還是必須看完整份文件或是在查看其他文件才有辦法對原本的文件進行了解。以論文撰寫來說，在搜尋參考資料的時候，使用者僅能對於使用其內容所提供的摘要與關鍵字進行初篩，對於細節還是必須閱讀整篇論文才有辦法理解，要整理進自己的論文當中，需要依靠的還是這些細節，關鍵字與摘要幾乎沒有辦法幫上忙。

1.2 研究動機

初學者迷航與重工、評估的結果遺失、知識模型的貧乏這三個問題是本論文所想探討的，在這邊分為兩點進行探討。

1.2.1 評估的結果遺失與知識模型的貧乏

第一點為了解決**評估的結果遺失**與**知識模型的貧乏**的問題，單純的使用關鍵字與摘要，本論文稱之為 AT(Abstract & Tag)模型，並不能解決這樣子的問題，因此使用者需要新的知識模型來描述原本的知識，而知識工程的領域，對於這一個部分已經有探討，知識工程包含了三個面向整理如下表：

表 2 知識工程面向分析

功能	方法
知識捕捉	● 非自動化捕捉 ● 萃取 ● 機械學習
知識模型	● 本體(Ontology) ● 派翠網路(Petri net) ● 物件導向(Object oriented)
知識使用	● 知識搜尋(Knowledge search) ● 知識推論(Knowledge inference)

在知識工程研究中主要的目的都是為了讓電腦能夠加有效的使用而非對人，本論文希望的是讓人能夠有效的利用這些知識模型，因此原本的模型不一定能夠直接套用。派翠網路(Petri net)，大多數用於自動化為一種離散併行系統的數學表示，後來才在知識工程上被運用，其內容為嚴謹的數學格式。而物件導向(Object oriented)與本體(Ontology)有著差不多的儲存型式，其差異在於物件導向(Object oriented)主要為提供知識樣板，本體(Ontology)著重在於知識的表達。

本體(Ontology)正規化不同主要分為可以分為四種(Uschold & Gruninger, 1996)。

- Highly informal： 高度非正規化，使用自然語言表現且結構鬆散。
- Semi-informal： 半非正規化，使用自然語言表現，有限制並採用結構化的格式，清晰且減少混淆。
- Semi-formal： 半正規化，使用人造的正規化語言表現。
- Rigorously formal： 嚴格的正規化，使用精心的定義的詞彙，正規化語意，定理與證明表現其穩健及完備性。

對於非知識工程師與領域專家的使用者，使用正規化(Formal)的本體(Ontology)來做為資料模型來表達知識，其內容過於抽象與嚴謹，而且是使用人造的正規化語言表達知識，了解這樣子的知識其花費也許比直接閱讀原本品質不佳的知識載體還來的多。

因此本研究希望透過半非正規化(Semi-Informal)的本體(Ontology)，意思就是有結構的本體，且使用自然語言來描述知識的，來做為知識模型的表達，用以解決**評估的結果遺失**與**知識模型的貧乏**的問題。

1.2.2 初學者迷航與重工

第二點為了解決**初學者迷航與重工**的問題，評估結果必須要可以重用。如果使用者能夠重用別人評估的結果，使用者可以解決初學者迷航的問題，如果使用者能夠重複使用使用者自己評估的結果，使用者可以避免初學者重工的問題，

初學者迷航代表著使用者對於該知識載體，哪一些是重要的可以使用的根本沒有頭緒，如果已經有別人所建立好的知識元件，對於知識的找尋，就可以參考別人的知識模型，加速養珠法的過程，幫助使用者快速建立評估標準與增加搜尋準確性；而對於知識的建構，就可以參考別人的東西直接重用或著經過自己的修改再進行使用。

初學者重工代表著必須重覆的閱讀同一份文件，如果已經有自己建立好的知識模型可以重用，就可以幫助自己快速的回憶之前評估的結果。

對於重用，在軟體工程裡面探討過許多，軟體開發的格言就是**不要重新發明輪子**，但是不是所有的軟體都可以重用，因為大多數的軟體都會因為許多開發中

的因素而有許多的設計債(設計不良的地方導致要修改重用都需花費大量的成本)，Fowler 在 1999 年出版了重構這一本書,這本書提出了如何對原本的軟體進行重構進而降低設計債，讓軟體能夠重用。

重構主要的目的在於改變原本已經存在程式碼的內部寫法，但是沒有改變外部的行為，透過重構可以增加軟體的可閱讀性，可維護性，可延伸性等非功能的屬性。

本研究希望透過重構的方法學來擷取知識，希望能夠增加其可重用性，也就包括了知識的可閱讀性，可維護性，可延伸性。

1.3 研究目的

本研究包括了以下目的：

- 為了解決**評估的結果遺失、知識模型的貧乏**，本論文將定義一個新知識模型，以半非正規化(Semi-Informal)的本體(Ontology)作為評估結果的儲存。
- 為了解決**初學者迷航與重工**，研究如何應用軟體重構的方法學增加知識模型可重用性。
- 研究如何加速重用的效率，重用一個重用性很高的元件多為小尺寸的元件，因此最後會產生大量的知識元件，如何找到正確的知識元件是很重要的事情,因此希望能透過推薦的方式,達到加速重用知識元件的目的。
- 使用本體(Ontology)來表現知識會有下列優點包括了，可重用性，完整性，關連明確性、使用重構的方法主要希望增加其可用性，可用性又可以拆解成為可閱讀性，可維護性，可延伸性等三項屬性。因此對於新提出的模型應該具備 1.可閱讀性 2.可維護性 3.可延伸性 4.完整性 5.關連明確性等五個要素，本研究希望透過實驗來研究此五個要素的效益。

觀上述研究目標，本研究希望能夠透過重構的方法學，對於原本琛差不齊的知識載體，進行知識元件的萃取，使用半非正規化(Semi-Informal)的本體(Ontology)作為知識模型的儲存，並透過推薦的方式加速重用的效率。讓一般使

用者可以透過該系統盡快的理解，整理，歸納，分享所需要用到的知識，這樣才能在這個資訊爆炸的時代獲取主動權與競爭優勢。

1.4 論文架構

本論文接下來討論架構包含文獻探討、系統框架、實驗設計、實驗結果、結論與建議；其關係及各章節之概述如下：

第二章、 文獻探討

包括了何謂本體(Ontology)，軟體重構與知識重構，推薦進行探討並歸納如何使用於本研究。

第三章、 系統框架

說明本研究所整合的知識框架，包含了知識載體，知識模型，知識重構與知識推薦。

第四章、 實驗設計

說明研究限制，以及 1. 可閱讀性 2. 可維護性 3. 可延伸性 4. 完整性 5. 關連明確性等五個要素如何驗證。

第五章、 實驗結果

說明本研究實驗的結果。

第六章、 結論與建議

本研究之結論與貢獻匯整，並提出實務運用與可延伸的議題。

論文架構如下圖所示。

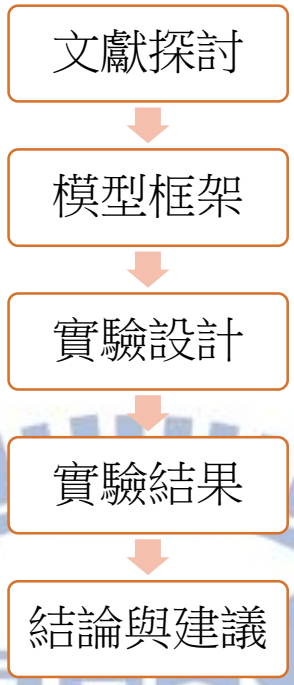


圖 1 論文架構圖

第二章、 文獻探討

本章節會就本論文所提及的重要概念包含本體，軟體重構與知識重構，推薦等主題進行探討、解釋與歸納，以利後續論文之進行。

2.1 本體 (Ontology)

2.1.1 發展史

早期為哲學家所探討，討論的內容包括了存在本身以及其基本特徵，著名的學者有亞里斯多德(Αριστοτέλης)，柏拉圖(Plato)，笛卡爾(René Descartes)…等人，後來人工智慧的學者開始將本體(Ontology)的觀念使用於知識表達上面，藉由本體(Ontology)中的概念與概念的關聯，用以描述真實世界的模型，近代本體(Ontology)的使用，則由蒂姆·伯納斯-李（Tim Berners-Lee）所倡導的語意網所引導，他主張在網頁中加入包含 Ontology 的中介資料(Meta Data)，讓電腦能夠(1)區辨詞義(sense)、(2)利用 Ontology 的知識架構判定正確詞義、以及(3)利用 Ontology 進行推理與訊息整合等能力，進而提升可用性(usability)和有效性（usefulness），實務上，有許多的公司都有使用到 Ontology，像是 Facebook Google, Yhaoo…等, Facebook 的社交圖是由 RFDa 所定義的，RFDa 可以將語意儲存在 XHTML 標籤之中；而 Google 與 Yahoo 則是使用 GoodRelations(The Web Vocabulary for E-Commerce)來加強其搜尋能力，BBC 則使用了 Programmes Ontology、Wildlife Ontology、Sport Ontology、Curriculum Ontology…等數種本體(Ontology)在其網站之中。

2.1.2 定義

Neches 最早提出關於本體(Ontology)的敘述如下：

一個本體(Ontology)對於一個特定的主題定義了基礎的詞彙與關聯以及這些詞彙與關聯組合的規則用以延伸這些詞彙(Neches et al., 1991)。

而後 Gruber 在 1994 年提出了下面的定義。

本體(Ontology)是對於分享的概念的一種約定，包括了對於概念模型化的領

域知識的框架，以及如何進行溝通的協定。(Gruber & Olsen, 1994)。

綜合上述敘述，本體(Ontology)基本上包含了詞彙，關聯，規則，框架，協定…等，用以表現知識，其優點包括了讓機器也可以使用這些知識，知識之間的關聯性更加明確，知識分析的結果更佳的完整以及知識可以重用。

2.1.3 模型研究

看了許多目前對於本體(Ontology)的研究，整個研究的方向可以概分為下述四個：

1. 如何更有效的捕捉知識，例如使用人工智慧進行捕捉。
2. 提出具備更好運算能力的正規化的模型。
3. 如何驗證知識模型內容是否正確無誤。
4. 如何讓電腦更好的使用本體(Ontology)。

其中對於模型的研究，學說上發展的模型有許多種，尤其是各種針對特定領域的所進行的探討，本論文著重於通用性，因此只有整理較為通用的模型，依照年分順序整理如下表

表 3 Ontology 模型整理

學者	概念
(Chaudhri, Farquhar, Fikes, Karp, & Rice, 1998)	Open Knowledge Base Connectivity (OKBC)包含了下列知識元件 ● Classes and Instances：類別與實體 ● Slots：類別與實體的屬性。 ● Facets：指的是 constraint，Slot 所能允許的值。 Frame：為資料結構，包含了 Classes, Instances, Slots 與 Facets.

(Devedžić, 1999)	敘述了在人工智慧領域，知識分為四個種類，分別為 ● 領域知識：代表著事實，理論。 ● 控制知識：描述了系統解決問題的能力。 ● 解釋知識：定義了推理過程及解釋語說明的內容。 ● 系統知識：描述知識庫的內容與結構 並且認為知識儲存的基本單元為OVA(Object-Attribute-Value)值組，框架，規則，邏輯運算與程序，複雜的邏輯則為基本單元的組合。
(潘旭偉, 顧新建, 仇元福, & 程耀東, 2003)	提出KM=(KC,KI,KS)物件模型，將顯性與隱性的知識用統一的物件方式進行建模。 ● KM(Knowledge Management)：知識管理，包含了知識名稱，以及KC,KI與KS ● KC：表示知識載體，可能是文件，資料庫，人…等。 ● KI：表示知識內容，包括了該知識的簡單描述與關鍵字。 ● KS：知識情境，使用不同的KD(Knowledge Dimension)來表示該知識使用的元素，再加上關聯構成網路圖，不同的知識情境可以透過路徑來計算相識度，以便判別兩者知識情境是否易於整合。
(Bobillo, Delgado, & Gómez-Romero, 2008)	首先定義了KR，KT 與 KA三種知識元件 ● KR：規則的集合 ● KT：術語的集合 ● KA：物件的集合 ● 接著定義了OD(Domain Ontology)=(KR,KT,KA)與OC(Content Ontology)=(KR,KT,KA)，最後定義CDR(Context-Domain Relevance) OP=(OD,OC)

可以發現晚期的本體(Ontology)，除了原本所包含的類別，關聯，實體，限制與規則外，另外增加了情境的概念，代表著知識運用會在特定的場景內，如果

要重新使用這些知識，就必須考慮這一些知識的情境，進而避免誤用。

2.1.4 正規化程度

本體(Ontology)正規化不同主要分為可以分為四種(Uschold & Gruninger, 1996)。

- Highly informal： 高度非正規化，使用自然語言表現且結構鬆散。
- Semi-informal： 半非正規化，使用自然語言表現，有限制並採用結構化的格式，清晰且減少混淆。
- Semi-formal： 半正規化，使用人造的正規化語言表現。
- Rigorously formal： 嚴格的正規化，使用精心的定義的詞彙，正規化語意，定理與證明表現其穩健及完備性。

半正規化還有嚴格的正規化對於一般使用者而言不常見，該類知識多半需要仰賴知識工程師與領與專家共同發展與定義，而且所定義出來的知識表現形式隨著正規化程度越來越嚴謹與抽象,這代表著對於一般使用者而言也越難以被理解；而一般使用者所接觸到的知識多半以高度非正規化以及半非正規化表示，高度非正規化，舉例來說，就像是開會時的交談紀錄、Blog 的內容，半正規化則像是論文、圖書等，有經過結構化整理的知識。

2.2 其他知識模型

2.2.1 SCROM

參考其官方網站 http：//www.adlnet.org/scorm/，SCROM 為共享內容物件參考模型，由 ADL (Advanced Distribution Learning Initiative)整合各個學習標準制定而成。對數位內容教材的製作、內容開發提供一套共通的規範。SCORM 的實際內容包括內容包裝模型（CAM），執行時環境支援（RTE）和排序與導航（SN）三大部分。

可以發現作為一個數位內容教材的主流的標準格式，除了教材內容與如何解譯執行的環境外，排序與導航單獨成立了一個大的部分，這是因為排序與導航在知識的學習上有很重要的意義。

2.2.2 Knowledge model

Rech 等人(Rech, Decker, Ras, Jedlitschka, & Feldmann, 2007)對於知識模型則定義了兩個部分，包含了知識元素與知識組件。

- 知識元素：基本的知識元件，具備有 AID 的特性：
  - 原子性(Atomic)：不可以在被切分成更小的知識項目。
  - 獨立性(Independent)：單獨一個知識項目也應該可以被使用，包括閱讀與組合。
  - 持續性(Durable)：知識項目是有時效性的。
- 知識組件：能夠完整且自給自足的描述知識，包含了至少一個知識項目，這些知識元件則應該有具備 4C 的條件：
  - 正確性(Correct)：描述正確沒有混淆。
  - 完整性(Complete)：具備所有相關的知識沒有遺漏。
  - 一致性(Consistent)：所有的知識組件應該具備一致性。
  - 簡潔性(Concise)：其知識內容應該為簡單、精準。

2.3 軟體重構(Software Refactor)

2.3.1 重構的發展史

重構在物件導向軟體開發中，不可或缺的一部分，最早由(Opdyke, 1992)提出，其定義為： 對於已經存在的程式碼進行重新建構的程序，但不改變任何的外在行為，並增進非功能的軟體屬性。

重用在 Biggersta,Charles(Biggerstaff & Richter, 1989)等的人研究中，分為四個步驟，包括了搜尋用的元件，了解搜尋到的元件，修改這些元件，組裝這些元件。

目前幾乎主流的 IDE 都支援這樣子的程序，在重構－改善既有程式的設計(Fowler & Beck, 1999)中提到，軟體的重構包含了四個目的，1.重構可以改善軟體的設計，2.重構使得軟體更容易被理解，3.重構可以幫助找到 Bug，4.重構可以

幫助提高編程速度。

而在 2004 年 Kerievsky 撰寫了重構－向模式前進 (Refactoring to Patterns)(Kerievsky, 2005)則明確的說明了，重構的最後目標就是模式。

模式(Patterns) 設計模式這個術語是由 Erich Gamma 等四人(Gamma, Helm, Johnson, & Vlissides, 1994)從建築設計領域引入到計算機科學的，此四人也稱四人幫 Gang of Four。模式是對於軟體設計中普遍存在（反覆出現）的各種問題，所提出的經過驗證有效的解決方案。

2.3.2 重構的核心概念

重構的核心概念(Fowler & Beck, 1999)，包括了三個部分 1.演進式設計 2.許多的小步驟 3.測試驅動。

1. **演進式設計**： 軟體在開發的過程當中，會有很多的限制跟條件，例如時程壓力、對於領域知識的了解、這一些會讓軟體在開發初期時，對於一些功能與設計會沒有辦法達到最佳；反過來說軟體開發如果一開始就定義與時做了許多用不到的功能與設計，只是讓開發的時程延長，而且需求總是一變再變，做了過多的設計，最後可能都只是一場空根本不會用到，因此對於軟體開發，重構的核心概念就是不多不少做到剛剛好，一旦有新增的需求時，再透過重構的方式變更原本的結構以迎合新的需求。

2. **許多的小步驟**： 在重構的過程當中，應該小步驟小步驟的施行變更，小步驟變更時，可以將影響範圍縮小，幫助使用者更容易的找到問題點。如果一次做太多步驟，當有問題發生時，必須不斷的還原之前的修改以確定是哪一個步驟發生問題，這樣會很花費許多成本，由其是一次修改多個程式碼又進行部分還原，最後只會是一團亂。

3. **測試驅動**：Beck 在 2003 年寫了關於測試驅動的書籍，Test-driven development： by example(Beck, 2003)，其開發步驟如下述：

   A. 在軟體開發前先行依照需求撰寫測試。

B. 確認該測試失敗。

C. 進行軟體開發。

D. 進行測試如果失敗就進行修正。

E. 確保所有的測試都是成功的。

2.3.3 重構的步驟

重構的步驟(Fowler & Beck, 1999)包括了察覺壞氣味(bad smell)，使用重構的方法，達成模式(Pattern)(Kerievsky, 2005)

- **察覺壞氣味(bad smell)**：壞氣味這一個詞彙是由 Martin Flower 書中所提出，意指程式的不好的設計。其包含項目如下圖所示：

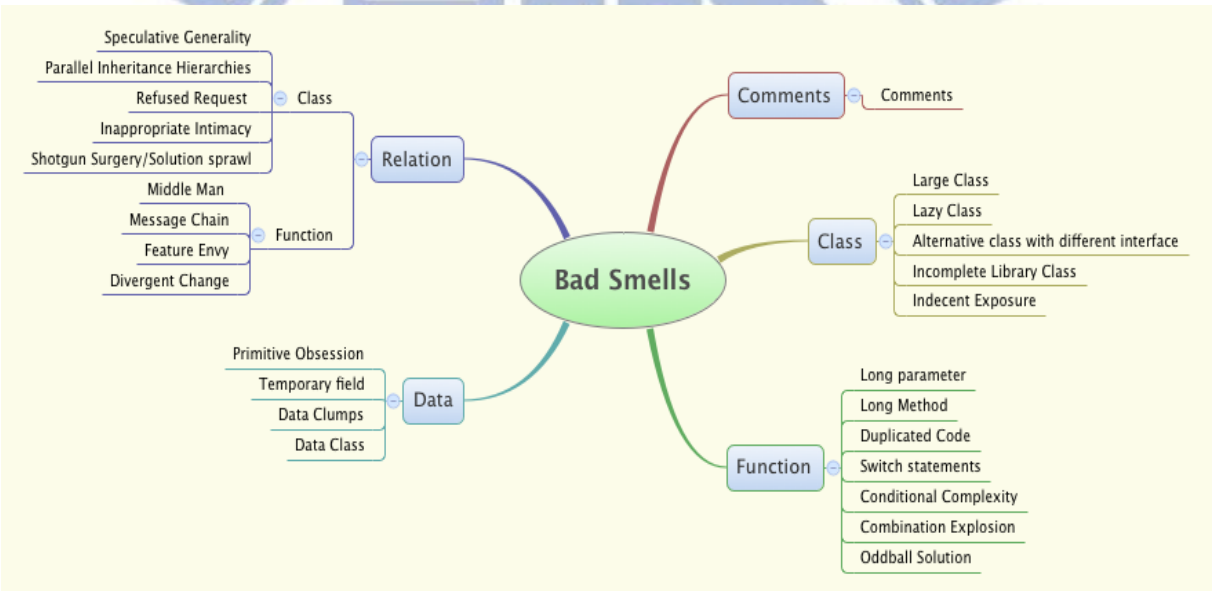


圖 2 軟體重構的 Bad Smells

以 Commnets 來說，過多的註解代表著程式本身無法良好的表達該程式的用意，因該透過重構的方式讓程式碼具備有良好的可閱讀性。

- **重構的方法**：一系列的步驟與方法，用來執行重構的步驟。

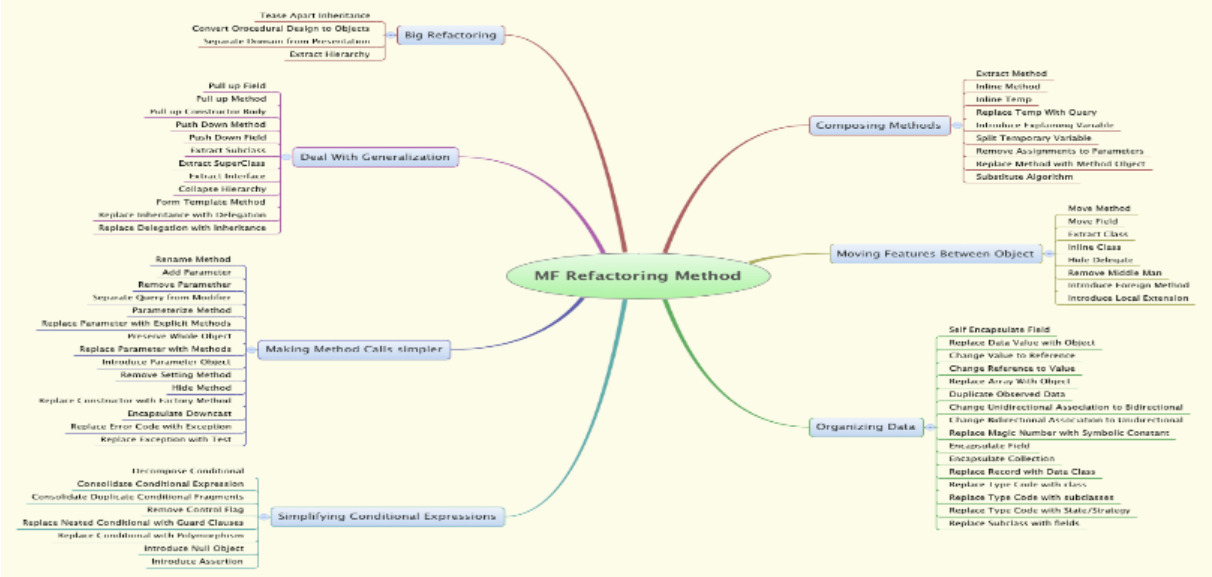


圖 3 軟體重構的方法

以 Decompose Conditional 為例，其動機為程式中會有許多複雜的 If-Else 語句來表達領域規則，如果每一個 If-Else 條件式中，都含有許多程式碼，則會讓整個程式的可閱讀性變得很低。其方法為將每一個 If-Else 條件式中程式碼用函式進行包裝，這樣子可以更清楚的表達其程式的意義與條件式的規則。

- **達成模式(Pattern)**： 如同前內容所述，模式是重構最終的目標，對軟體開發而言，模式的類型可以分為下述四種：
  - Creational： 提供產生物件的 Pattern，像是工廠模式，單例模式。
  - Structural： 透過一定的構造來達成資料轉換或是功能修飾，例如組合模式，橋接模式。
  - Behavioral： 資料與控制流的模式，包括了命令模式，迭代器模式。
  - Concurrency: 併發相關的模式，例如雙重檢查鎖定，執行緒池模式。

2.4 知識重構(Knowledge Refactor)

如果重構能夠應用到知識領域，也應該可以達到下列四個目的： (1)重構可以改善知識結構的設計，(2)重構使得知識更容易被理解，(3)重構可以幫助知識

的釐清與找到知識的謬誤，(4)重構可以使知識更快速的被運用。

對於知識重構的步驟也比照軟體重構步驟進行探討，包含了知識的壞氣味，知識重構的方法，知識模式。

2.4.1 知識壞氣味

知識的壞氣味也就是知識的設計與表達上有所問題，Rech 在 2007 年(Rech et al., 2007)提出了下列知識反模式(Knowledge Anti-Pattern)，代表知識不好的設計，在本論文中也就是知識的壞氣味。

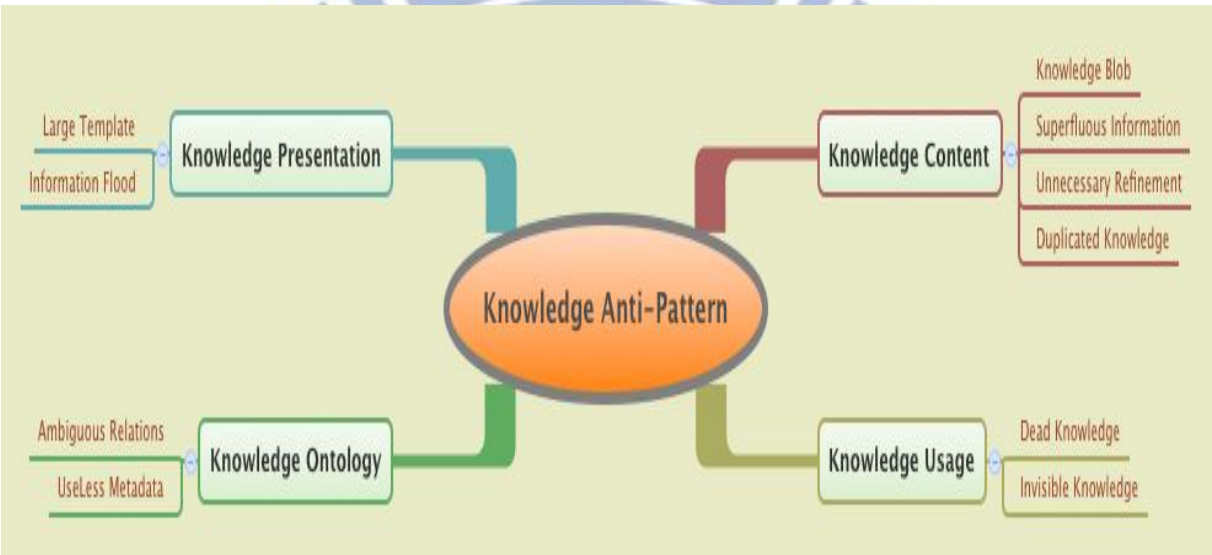


圖 4 知識重構的壞氣味

模式的表達上面通常使用目錄的形式來進行呈現，其格式包含了，模式的名稱、問題說明、問題原因與解決方式，整理這些反模式如下:

- 知識內容反模式(Knowledge Content Anti-Pattern)

模式名稱	知識泡泡(Knowledge Blob)
問題	一個知識元件隨著時間越來越大，承載了過多的資訊，在開放式的資訊元件中比較常發生。
問題原因	知識元件太容易可以修改，而且修改者沒有常識來維護這些知識元件，沒有遵守標準化的方式建立知識元件。
解法	壓縮知識(Compact Knowledge) 抽取知識(Extract Elements) 抽取共通(Extract Commonalities)

模式名稱	冗余知識(Superfluous Knowledge)
問題	一個知識元件有太多不必要的資訊，例如無關於該主題，已經在其他地方描述過，或是過時的資訊，導致使用者的學習被打斷。
問題原因	當知識建立者不知道該如何正確清楚簡短的描述知識時，或是在建立結構化資訊時沒有指導原則(Guidelines)
解法	壓縮知識(Compact Knowledge) 提供模板(Offer Templates)

模式名稱	不需要的精緻化(Unnecessary Refinement)
問題	過多頁數來描述同一個主題，而且不能被其他知識所重用，使用者需要打斷學習，直到所有的頁面閱讀結束。
問題原因	寫太多的細節，但是不每一個人都需要這樣子的資訊。
解法	壓縮知識(Compact Knowledge)

模式名稱	重複的知識(Duplicated Knowledge)
問題	同一個知識多個版本的資訊儲存在各種不同的位置，如果要修改該知識，就要修改許多不同的地方。
問題原因	使用者不在乎是不是已經有相似的知識存在。
解法	壓縮知識(Compact Knowledge) 抽取共通(Extract Commonalities)

- 知識使用反模式(Knowledge Usage Anti-Pattern)

模式名稱	死亡知識(Dead Knowledge)
問題	一個知識元件沒有人使用，浪費資源，例如增加搜尋所花費的時間。
問題原因	在各種情況下所產生沒有質量的內容
解法	融合知識(Fuse Knowledge) 忘卻知識(Forget Knowledge)

模式名稱	不可拜訪的知識(Invisible Knowledge)
問題	有用的知識元件，但是使用者沒有辦法找到，浪費空間或系統資源。
問題原因	因為重構或知識系統本身的問題所導致。
解法	重新整合知識(Reintegrate Knowledge) 融合知識(Fuse Knowledge) 忘卻知識(Forget Knowledge)

- 知識本體反模式(Knowledge Ontology Anti-Pattern)

模式名稱	混淆的關聯(Ambiguous Relations)
問題	關聯沒有被清楚地描述，如果有多個相似或者太多的關聯會讓作者混淆。
問題原因	沒有作者指導原則、模板或是工具來協助創造知識。
解法	最小化關係集合(Minimize Set of Relations) 提供知識創作指導原則(Provide Authoring Guidelines) 提供模板(Offer Templates)

模式名稱	無用的中介資料(Useless Metadata)
問題	一個 Ontology 含有太多特性描述沒有用的觀點，而這些資料對於使用者來說沒有帶來任何利益。
問題原因	知識創造者對於未來的特性做了許多不必要的設計。
解法	只有需要才維護這些特性 刪除中介資訊(Remove Metadata) 融合中介資訊(Fuse Metadata)

- 知識表現反模式(Knowledge Presentation Anti-Pattern)

模式名稱	大樣板(Large Template)
問題	知識創作者填寫的太多的中介資訊來描述這一些資料，時常讓使用者喪失對於閱讀知識的興趣。
問題原因	Ontology 過大，提供過多的功能給使用者，並且支援太多的應用程式。
解法	使用自動化的方式減少需要維護的地方，例如更新日期。 提供模板預設值(Create Template Defaults) 縮短模板(Shorten Template)

模式名稱	資訊洪流(Information Flood)
問題	一次有太多的知識元件呈現給使用者，例如一次呈現全部的收尋結果。
問題原因	知識庫成長到大數量的時候。會依照進行分割與收斂，如果忘記做這些動作就會導致知識庫太大。
解法	彙總知識(Aggregate Knowledge) 區塊呈現(Chunk Presentation) 結果分群(Cluster Results) 精煉分類(Refine Classification)

2.4.2 知識重構的方法

對於知識的重構(Rech et al., 2007)定義了下列方法，來進行知識的重構：

- 壓縮知識(Compact Knowledge)：總結與重寫原有的知識，讓知識以較短的方

式呈現，並且不改變原有知識的含義。

- 抽取知識(Extract Elements)：抽取原本知識的一部份形成新的元素，遞迴的執行分割與征服(divide & conquer)的策略，使得原本的知識成為一個互斥的知識元素集合。
- 抽取共通(Extract Commonalities)： 將兩筆知識元素共通的部分，截取出來成為新的知識元素。
- 融合知識(Fuse Knowledge)：將知識中有用的部分抽取出來，加入相似的知識之中，並把原本沒有用的部分刪除。
- 忘卻知識(Forget Knowledge)：將要廢棄的知識元素透過標記的方式，宣告這個知識已死，從各種結構中移除該知識元素，這樣子的動作需要經過實做或專家驗證。
- 描述上下文(Describe Context)：清楚地描述該知識的情節，並說明誰創造了這個知識元素。
- 重新整合知識(Reintegrate Knowledge)：將原有的知識重新整合成為新的導航或索引之中。
- 連接起始頁(Link To Start Page)：將每一個知識元素或元件加入起始點。
- 彙總知識(Aggregate Knowledge)：將多個類似的知識元素進行彙整，並建立新的知識元素體系，舊有的元素不會進行刪除。
- 區塊呈現(Chunk Presentation)：將所有知識元素細分成幾個區塊，每一個區塊顯示於單頁。
- 結果分群(Cluster Results)：將搜尋結果依照共通性顯示。
- 列出知識(List Knowledge)：依照主題將相關知識列表表示。
- 鏈接知識(Link Knowledge)：自動將知識本體敘述內的知識進行鏈接。
- 知識序列化(Serialize Knowledge)：序列化串接連貫的知識塊。
- 精煉分類(Refine Classification)：將擁有過多元素的分類進行整理，產生

更多有意義的子類別。

- 創造共通中介類別(Create Common Metadata Class)：創造共通中介屬性的類別，例如版本，作者名稱...等維護性資訊。
- 刪除中介資訊(Remove Metadata)：將沒有人使用的中介資料刪除。
- 融合中介資訊(Fuse Metadata)：找尋類似的中介資訊，可以直接使用或是定義新的中介資訊給所有的人使用。
- 最小化關係集合(Minimize Set of Relations)：類似的關係使用相同的關係名稱，或著是引入一個更一般化的關係。
- 提供知識創作指導原則(Provide Authoring Guidelines)：教導使用者如何在遵循規範下創建知識，例如樣板如何使用。
- 提供模板(Offer Templates)：提供整個主題相同的知識模板，包括相同的知識架構，還有所需要用到的知識。
- 提供模板預設值(Create Template Defaults)：提供有意義的預設值，如果沒有意義，這個模板需要改進。
- 縮短模板(Shorten Template)：將模板中不需要的部分移除。
- 分離焦點(Separate Concerns)：不同的人關注的焦點不同。
- 安排會議空間(Arrange Meeting Space)：提供一個舒適的地方討論知識。
- 討論知識(Talk About Knowledge)：定期談論知識，例如部門會議。
- 顯性知識會議(Explicit Knowledge Meetings)：關注特定主題與傳播知識，並找出知識差距。

2.4.3 知識模式

對於知識良好的設計 Rech 在 2007 年(Rech et al., 2007)提出了下列知識模式(Knowledge Pattern)：

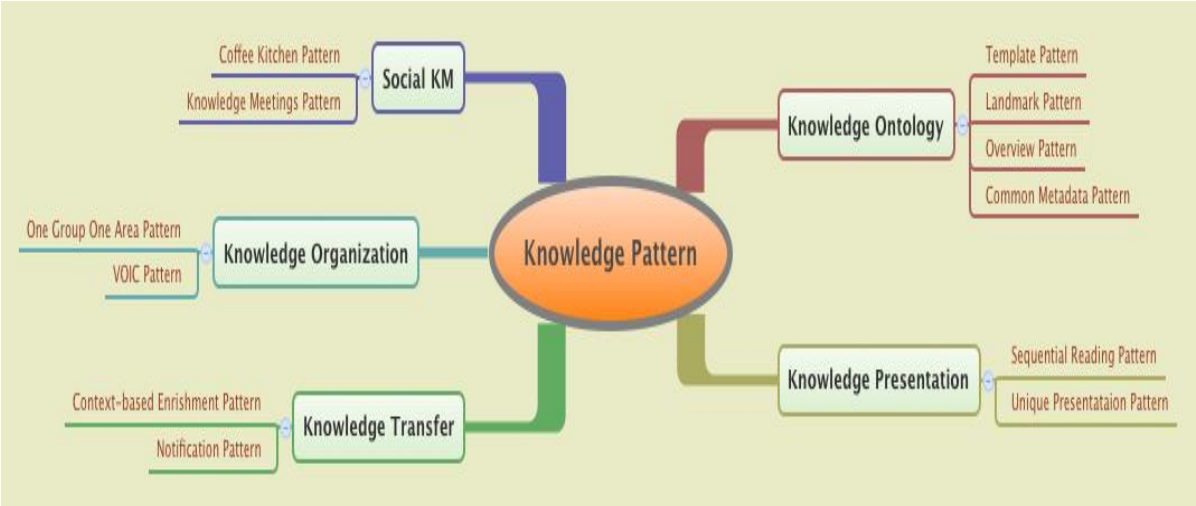


圖 5 知識重構的模式

對於知識模式一樣使用目錄進行描述，包含了問題，問題原因，解法。

- 知識本體模式(Knowledge Ontology Pattern)

模式名稱	範本(Template)
問題	特定類型知識元素有不同的知識結構是很難被理解的，因為使用者除了瞭解知識本身，還需要了解整個架構。
問題原因	知識撰寫者通常使用自己的架構描述知識，，也有可能因為不同的專案有不同的寫法。
解法	提供模板(Offer Templates)

模式名稱	指引(Landmark)
問題	知識元件沒有連結到首頁，導致使用者在結果頁遺失前一頁的一些知識。
問題原因	搜尋使用整個頁面作為回傳頁面。
解法	連接起始頁(Link To Start Page)

模式名稱	大綱(Overview)
問題	語意類似的知識如果沒有大綱只有條列的話，會變得難以理解及找尋。
問題原因	知識創作者沒有被通知相關聯的知識必須整合到原有的架構之中。
解法	列出知識(List Knowledge)

模式名稱	共通的中介資料(Common metadata)
問題	一個知識系統中總是有不同類型的知識元件，但是對於索引，分類這一些共通性的中介資料是每一個類別都需要的。
問題原因	如果沒有規則建立這些共通資料，知識創作者時常會忘記編寫。
解法	創造共通中介類別(Create Common Metadata Class) 提供知識創作指導原則(Provide Authoring Guidelines) 提供模板(Offer Templates)

- 知識表現模式(Knowledge Presentation Pattern)

模式名稱	序列化閱讀(Sequential Reading)
問題	知識如果沒有被妥善地安排順序提供給閱讀者，閱讀者時常無法正確的了解與運用該知識。
問題原因	同時有多個知識創作者為了不同的目的在維護這些資訊，導致最後的結果不是線性的描述，例如解決問題方式。
解法	知識應該依照使用者的需求顯示不同的順序 知識序列化(Serialize Knowledge)

模式名稱	唯一呈現模式(Unique Presentation)
問題	如果一個知識元件有多種呈現會方式會混淆讀者，每一個群組的使用者應該都有唯一的知識呈現方式。
問題原因	知識創作者，可能依照不同的專案有不同的知識呈現方式。
解法	每一種知識都應該有一種制式格式，像是顏色字體。 提供模板(Offer Templates)

- 知識轉移模式(Knowledge Transfer Pattern)

模式名稱	豐富上下文(Context-based Enrichment)
問題	知識創作者通常以某一種角度或觀點來描述知識，應該建立內容性的知識來幫助初學者。
問題原因	知識創作者不知道如何使用，過於簡短的描述這些知識元件。
解法	描述上下文(Describe Context) 鏈接知識(Link Knowledge)

模式名稱	通知(Notification)
問題	在靜態知識系統中，如果對於知識元件有修改並不會通知相關的使用者進行更新。
問題原因	沒有觀察者自動的通知使用者更新知識元件，尤其對於正在開發中的狀況。
解法	監控知識(Monitor Knowledge) 監控本體(Monitor Ontology)

- 知識組織模式(Knowledge Orgination Pattern)

模式名稱	一群組一區域(One Group One Area)
問題	每一個群組都有關注的焦點，幾乎沒有或小部分重疊，例如 Wiki 的開發者，管理者跟使用者。
問題原因	不同群知識組知識創作者在描述知識時非常類似導致混淆。
解法	分離焦點(Separate Concerns)

模式名稱	VOIC
問題	如果知識系統沒有區分呈現(View), 結構(Ontology), 規則(Inference)，知識(Content)，維護會非常困難。
問題原因	因為時間因素或著設計限制，知識並沒有外顯的分離。
解法	分割 VOIC 區分出 View,Ontology,Inference 與 Content。

- 社會化知識管理模式(Social Knowledge Management Pattern)

模式名稱	咖啡廚房(Coffee Kitchen Pattern)
問題	交換知識需要空間，像是咖啡廚房，抽煙角落或閱讀室，讓人員可以彼此認識，這樣子資訊才會流通。
問題原因	管理者因為某些原因，例如浪費時間，而不讓人員交談。
解法	安排會議空間(Arrange Meeting Space)

模式名稱	知識會議(Knowledge Meetings)
問題	知識內容因為時間壓力或是工作負債而沒有被使用，但這些知識是有幫助的。
問題原因	使用者因為時間壓力或限制，沒有意願使用KM系統，或著與同事溝通，交流知識。
解法	討論知識(Talk About Knowledge) 顯性知識會議(Explicit Knowledge Meetings)

2.5 推薦(Recommend)

Schafer 等人在 1999 年對於電子商務提出了推薦的概念(Schafer, Konstan, & Riedl, 1999)，透過資料的分析，推測或建議使用者應該有興趣的資料。以下就推薦方法進行介紹。

2.5.1 推薦輸入

推薦系統仰賴資料的分析，當系統要對使用者進行推薦分析時，會抓取下列資料做為輸入，推薦可以以個人做為輸入推薦，也可以以群組作為輸入推薦。

- 個人輸入

  包括導覽，關鍵字，物品屬性，評分，購買歷史…等。

- 群組輸入

  物品屬性，物品受歡迎的程度，群組購買歷史，評分與文字評論…等。

2.5.2 推薦方法與相關問題

推薦的方法可以分為三種，以內容為基底的推薦，協同過濾推薦與混合推薦。

- 內容為基底的推薦

將輸入與對應的物品進行相似度或著是其他計算模型進行計算，將分數高的物品推薦給使用者。由於其比對內容只有文字敘述，所以經常會遇到下列問題 1. 對於未知物品無法進行推薦。2. 當遇到一詞多意或是多詞一意時無法判定是否具備相同屬性。

- 協同過濾推薦：

藉由相同興趣群組內的其他人推測使用者對於目標物品是否有興趣，所以計算時，先將使用者進行分群，在透過同一群的其他人的輸入進行計算。常見的問題包括了 1. 稀疏矩陣，即使是相同興趣的群組也不一定對於所有物品都有輸入，因此對於沒有輸入的物品預測能力較差。2. 冷啟動，在營運的初期缺乏資料，所以無法進行有效的分群與協同推薦，必須等到使用者輸入資料累積到一段時間時，其推薦準確率才會上升。

- 混合推薦

結合內容基底推薦與協同過濾推薦。

第三章、 系統框架

在本章節中將對整個架構進行介紹，首先先對系統框架進行簡介，接著在對架構的每一個部分進行探討。

3.1 系統框架簡介

回顧一下本研究目的：

- 為了解決評估的結果遺失、知識模型的貧乏，本論文將定義一個新知識模型，以半非正規化(Semi-Informal)的本體(Ontology)作為評估結果的儲存。
- 為了解決初學者迷航與重工，研究如何應用軟體重構的方法學增加知識模型可重用性。
- 研究如何加速重用的效率，重用一個重用性很高的元件多為小尺寸的元件，因此最後會產生大量的知識元件，如何找到正確的知識元件是很重要的事情，因此希望能透過推薦的方式，達到加速重用知識元件的目的。
- 使用本體(Ontology)來表現知識會有下列優點包括了，可重用性，完整性，關連明確性、使用重構的方法主要希望增加其可用性，可用性又可以拆解成為可閱讀性，可維護性，可延伸性等三項屬性。因此對於新提出的模型應該具備 1.可閱讀性 2.可維護性 3.可延伸性 4.完整性 5.關連明確性等五個要素，本研究希望透過實驗來研究此五個要素的效益。

因此本研究的系統框架包含了知識載體，知識重構，知識模型與知識推薦，以下圖來做表示：

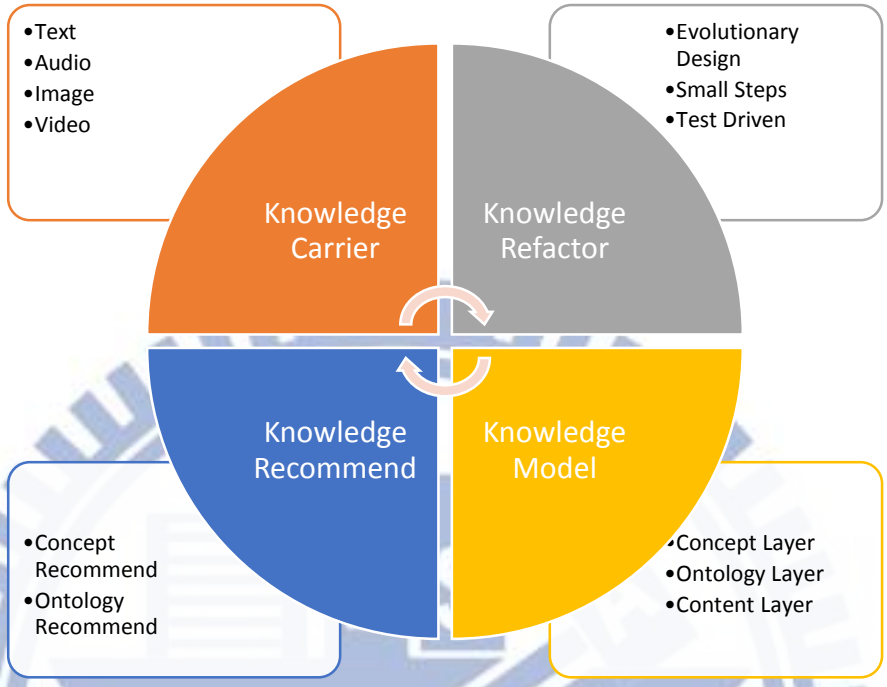


圖 6 知識系統框架

接著對於以上四個大項進行說明。

3.2 知識載體

知識的來源，包括了各式各樣可能的載體，像是 Word，PPT，網頁，書籍…等，其形式表達不外乎文字，聲音，圖片與影像等四種，本研究只著重於文字的表現形式進行探討。

這些知識的作者寫作的風格與組織知識的方式往往是南轅北轍，對於使用者而言無疑加深了要了解這些知識的成本。在公司內如果沒有做好知識管理，這種品質不良的知識載體會頻繁出現，導致組織內的知識學習與運用成效不彰，這些知識載體經常出現下面的知識反模式：

- 知識泡泡(Knowledge Blob)

  隨著公司的不斷成長，知識也會隨之累積，例如軟體開發中的使用案例，剛開始只有數十頁的規模，可是到後來可能會變成數百頁的規模，對於這麼大一份文件，系統分析人員可能無暇也無力全部進行通盤的了

解，在面對時程壓力下，通常只有修改與新增該次專案所要進行的部分，這會變成整份文件前後可能彼此矛盾，所用的格式上下不同，其結構可能會出現頭重腳輕的狀況。

- 冗余知識(Superfluous Knowledge)

  該知識已經過時卻沒人整理，隨著公司人員的改朝換代，往往新人無法理解為什麼有這些知識，而且也不敢對這些知識進行處理，導致每一個新人都要理解一遍這些無用的知識，浪費時間成本。

- 重複的知識(Duplicated Knowledge)

  公司人員的流動是一個公司很正常的事件，因此交接就會是一件很重要的事情，如果交接沒有做好就會造成知識斷層，對於同一個職務而言，當人員流動頻繁，就會造成同一個知識同時存在多個知識載體，一旦這些知識發生改變，就必須同時修改多份知識載體，萬一有遺漏，就會造成知識的矛盾。

因此這些知識載體需要進行知識的重構，透過後面所介紹的知識重構，增加知識的可閱讀性，消除上述的知識反模式。

3.3 知識模型

在知識重構介紹前，先行定義知識模型來了解重構最後達建立的結果，這樣子在閱讀知識重構時，才更能理解知識重構所要表達的內容。

由於只有關鍵字與摘要的方式並不能幫助使用者快速的理解這些知識，因此本研究參考其他正規化(formal)知識模型，提出一個基於半非正規化(Semi-Informal)的知識模型如下圖所示：

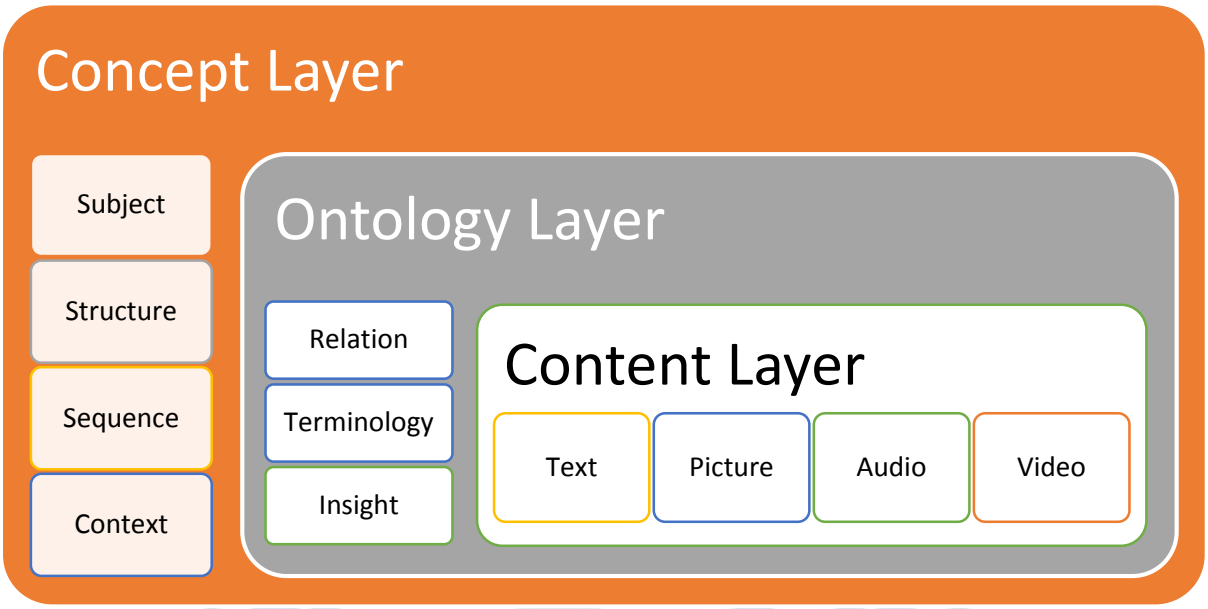


圖 7 半非正規化(Semi-Informal)知識模型

模型分層三層包括了內容層(Content Layer)，本體層(Ontology Layer)與概念層(Concept Layer)，其內容由低層級到高層級說明如下：

3.3.1 內容層(Content Layer)

將原始的知識載體進行分解而成，可以依據 Word 中的段落，PPT 中的重點或 HTML 中的<P>標計進行分解，目的是將原有的大段文章拆解成為小塊，這樣子在建立知識模型時，可以有更明確的關聯。除了可以在閱讀時說明該關鍵字或是觀點指向的是哪一個段落，當需要重用這些內容時，也較容易修改。

定義： KC(Knowledge Content)：原始知識載體的切片，除了內容外還包括含了在原始知識載體中的位置，一個知識載體包含多個 KC。

3.3.2 本體層(Ontology Layer)

用以表現知識元件，如果知識載體是血肉的話，那本體層所要描述的即是知識的骨架，其類別主要可以分成術語(Terminology)、觀點(Insight)與關連(Relation)。

- 知識術語(Knowledge Terminology)

  描述特定的知識觀念下的關鍵字為客觀的表示，與原本單純的關鍵字差

別在於多了關鍵字的敘述，對於該概念的初學者而言，在建構知識的過程當中，關鍵字的理解會影響評估的結果，其搜尋的方向也會有所差異。因此不能只有給使用者例如 EMH 這樣子的縮寫，還必須讓使用者知道 EMH 是效率市場假說的縮寫，其定義為如果在一個證券市場中，價格完全反映了所有可以獲得的信息，那麼就稱這樣的市場為有效市場。

對於一個知識密集的產業，其專有名詞與縮寫是非常多的，對於資深的使用者而言，這些專有名詞與縮寫已經內化成為一種本能，但是對於初學者而言，頻繁出現的的術語卻大幅增加了其學習的難度，因此知識在呈現上，除了知識載體本身外，應該透過這樣子的知識元件，幫助初學者使用者理解知識內容。

定義： 知識術語(Knowledge Terminology)： 客觀的對於**關鍵字**以及**關鍵字**在特定領域下的敘述。

- 知識觀點(Knowledge Insight)

以人的主觀對於知識載體的理解，包括了重點與心得兩個類型。使用者在建立觀點知識元件時,必須先對既有知識載體進行閱讀並且進行知識內化，再透過知識外化，將所學的知識表現出來，如此對於個人而言即完成了知識的學習。

每一個人對於整個知識的理解程度不同所表達的重點與心得也就不同，因此如果能夠透過別人的重點與心得,除了可以幫助自己加速理解這些知識，也可以進行知識理解的自我測試，避免抓錯重點與理解錯誤。

定義： 知識觀點(Knowledge Insight): 對於特定知識內容**主觀**的理解，包括**重點**與**心得**兩種類型。

- 知識關連(Knowledge Relation)

不同知識元件中的關聯，例如一個知識內容可以有零到多個知識術語，每一個知識內容可以有零到多個知識觀點，知識術語與知識術語之間也可以有零到多個關聯，當這些關聯定義的越明確，不但能幫助使用者進行搜尋關

注的移動，也可以透過關聯進行相似度的計算做為推薦之用。

3.3.3 概念層(Concept Layer)

用以描述特定主題的中介資料，包括了主題，結構，順序與情境。

- 主題(Subject)

  包括了主題名稱與摘要，讓幫助使用者能對該概念有初步的了解。

- 結構(Structure)

  包含了內容層與本體層的結構，對於知識內容，知識術語與知識觀點進一步的進行歸納與整理，讓相關聯的知識元件能有良好的分群與架構。

  良好的結構可以達成 Knowledge Ontology Pattern 中的大綱(Overview)與指引(Landmark)。

- 順序(Sequence)

  對於同樣的結構能夠提供不同的順序，例如以 CSS 的學習來說，對於美工背景的使用者可以先提供其語法的表現形式，再提供語法的格式，而對於程式背景的使用者，則先提供語法的格式，再提供語法表現的形式，來幫助使用者理解知識。

  良好的閱讀順序可以達到 Knowledge Presentation Pattern 中的序列化閱讀(Sequential Reading)。

- 情境(Context)

  特定主題的概念有其使用的情境，為了幫助使用者進行搜尋，因此必須提供主題的屬性進行配對，例如：主題名稱，作者，版本名稱，建立時間，來源，出版社，ISBN，主題類別、關鍵字集，關鍵字關聯路徑…等。

  描述清楚明確的情境可以達成 Knowledge Transfer Pattern 中的豐富上下文(Context-based Enrichment)。

3.4 知識重構

重構的核心概念包括了演進式設計，許多的小步驟、測試驅動，以下這些概

念進行分析與探討。

3.4.1 演進式設計

使用者在建構知識時，會不斷的新增詞彙與理解，因此會不斷的修改已經存在的知識元件，因此 Knowledge Transfer Pattern 中的通知(Notification)就會很重要的議題。對於共有模型(多人維護同一份本體，例如公司內共有的知識模型)或是私有模型(每一個人都有自己的一份本體)，應有不同的處理方式。

對於共有模型應該顯示最新版本，只要有修改的事件出現，就應該通知所有相關聯的使用者對修改進行確認。

而對於私有模型而言，應該明確的紀錄使用者所選的版本是哪一個，只要使用者進行修改，就應該產生一個全新的版本，而不影響原有模型的其他使用者。

同一個版本的知識元件不應有多個，應避免知識內容反模式(Knowledge Content Anti-Pattern)中的重複的知識(Duplicated Knowledge)。

3.4.2 步驟

對於知識的重構，依照 Bottom up 的原則，根據下列小步驟進行處理

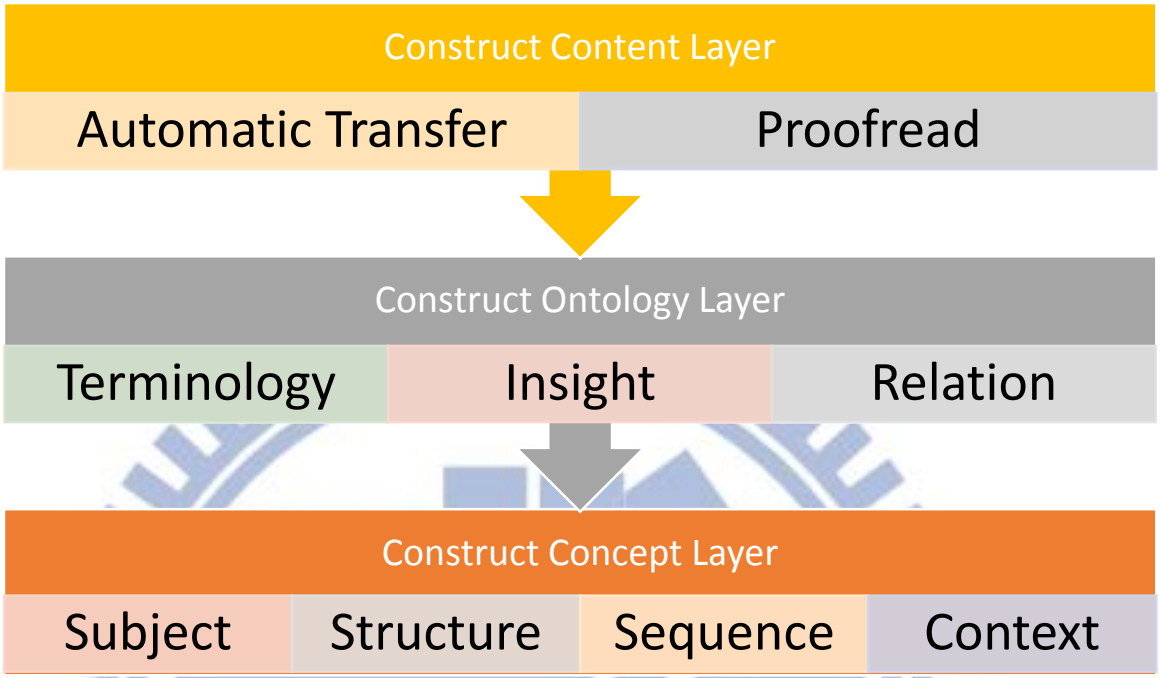


圖 8 知識重構流程

- 建構內容層

  將原本的知識載體做為輸入，經由解析規則將其切分為多個知識內容，由於每一個知識創作者對於知識的表現都不同，解析規則對於不同的知識載體解析的結果可能會發生誤差，因此必須透過校正，確認知識內容的大小與完整性。

- 建構本體層

  對於每一個知識內容都進行知識術語與知識觀點的萃取，可以選擇重用別人所建立的知識元件，也可以重用自己的知識元件，如果都沒有可以使用的知識元件，則透過抽取知識(Extract Elements)、抽取共通(Extract Commonalities)、融合知識(Fuse Knowledge)等手法，建立新的知識元件，最後進行檢覈表測試，確保所建立的知識元件都符合一定的規範。

- 建構概念層

一旦本體層建構完成，使用者對於整個知識載體就有了基礎的了解。首先需要建立架構，其中包含內容層與本體層。再來要建立的是閱讀的順序，要確保對於使用者能夠提供序列化閱讀(Sequential Reading)。接著要建立的是主題，最後建立的是情境。

3.4.3 測試驅動

重構的最後目的是達成模式與避免反模式，所以本論文中以模式做為測試的內容，提出對於新增知識元件的檢核表，在使用者建立知識模型之前，就先確定測試項目，使用者所建立的知識必須通過所有的測試。

對於新增知識元素必須考慮下述條件：

- 確認正確性：確定所輸入的內容正確無誤。
- 確認獨立性：確認所輸入的內容即使單獨閱讀也可以使用。
- 避免冗余知識： 確定所輸入的內容都與術語或知識內容有關的主題，。
- 避免不需要的精緻化： 確定所輸入的內容大小是適合的，避免太多的內容造成不需要的精緻化。
- 避免重複的知識： 確認所新增的知識模型並沒有重複。

對於一個知識內容要確認下述條件：

- 確認完整性：新增的知識模型是否涵蓋所有的知識內容。
- 避免矛盾：要確認新增的知識模型是否有矛盾的地方。
- 避免重複的知識： 確認所新增的知識模型並沒有重複。

3.5 知識推薦

知識推薦分為兩個部分一個是知識概念的推薦用於知識的搜尋與閱讀，一個則是知識元件的推薦用於知識模型的創造。

3.5.1 知識概念推薦

對於想了解特定知識的初學者而言，其字彙數是不足的，因此透過已經建好

的知識模型，讓使用者進行選擇所需知識術語以及其關聯是必要的，不然使用者會不斷的在動態模型上進行循環，所造成的時間成本花費就會大量的增加。

首先使用者輸入想了解的知識名稱,系統將符合該知識名稱的知識術語列出，使用者再進行知識術語選擇，系統接著將有關聯的知識術語進行拓展，讓使用者在進行選擇，重複此步驟直到使用者選擇完畢所想了解的知識術語以及其關聯。

接著透過計算知識術語與其關聯與系統中所有的知識概念情境進行相似度計算，最後依照相似度高低呈現給使用者。

3.5.2 知識元件推薦

知識元件全部都由單一使用者建立的話是很花費時間的事情，在時間有限的狀況下，重用別人的知識元件是必要的，但有一個很重要的課題，就是在大量知識元件的狀況下如何找到最符合的知識元件來進行重用，在本論文中知識元件推薦包含兩個部分，知識元件被重用的次數與其他使用者重用知識元件的相似度。

知識元件如果被重用的次數越多，代表具有一定的重要性，因此對於同一個知識內容，如果某一個知識元件被重用的次數很多，則優先顯示於使用者重用的頁面中，提供使用者進行選擇。

另外一方面則是與其他使用者的相似度，使用者所選用的知識元件與某一個使用者越相近，則代表該其他使用者與使用者對於事情的看法更趨近一致，因此該其他使用者所重用的知識元件應當優先顯示於使用者重用的頁面中，提供使用者進行選擇。

第四章、 實驗設計

在本章節中會描述實驗的限制，實驗目的、實驗設計上的考量與問題，最後介紹系統架構。

4.1 實驗限制

由於是新的架構，沒有現存的資料庫或平台可以使用，必須自行實做所有的內容與框架，在時間有限的狀況下，實做的範圍與實驗的規模受到影響，因此本次實驗有下列限制。

- 在框架中知識載體包含了文字、聲音、圖片、影像，本論文與實驗都僅探討文字的部分，其他的部分留待後人進行研究。
- 在模型建構上面，僅提供內容層(Content Layer)與本體層(Ontology Layer)建置，概念層(Concept)僅有提供摘要，其他部分並無實做。
- 在推薦實現上，知識概念的推薦並沒有包含在本次的實驗當中。

4.2 實驗目的

主要的實驗目的有下列主題，包含知識元件的效益，先備程度對於效益的影響，分別介紹如下。

4.2.1 知識元件的效益

使用 Ontology 來表現知識會有下列優點包括了，可重用性，完整性，關連明確性、使用重構的方法主要希望增加其可用性，可用性又可以拆解成為可閱讀性，可維護性，可延伸性等三項屬性。因此對於新提出的模型應該具備 1.可閱讀性 2.可維護性 3.可延伸性 4.完整性 5.關連明確性等五個要素。

- 可閱讀性：

目的在於了解知識架構對於可閱讀性的影響，測量讀者閱讀此知識模型所花費的時間做為量測標的。但有閱讀不一定代表有了解與記憶，所以加入後測的機制，確認使用者是否真的了解這些知識，如果有錯誤代表使用者應該花費更多的

時間進行閱讀才有辦法完全理解知識內容，因此可閱讀性的表現上以(閱讀時間/後測答題正確率)做為評量指標。

- 可維護性

可維護性代表的是否容易將原本的知識元件修改成為自己想要的版本，因此透過量測修改時間做為評量指標。

- 可延伸性

可延伸性代表的是否能夠能被重複使用，因此透過不同模型的重用次數做為評量指標。

- 完整性

完整性代表知識模型對知識載體表現的程度，因此透過計算平均使用者對於知識元件(包含術語與觀點)的建構數量來做為評量指標。

- 關連明確性

關連明確性代表的是知識模型對於知識載體關聯的狀況，因此透過計算平均使用者所建構知識模型對於知識模型的關聯數做為評量指標。

實驗的對象模型分為三項分別為 AT(Abstract&Tag)、SIO(Semi-Informal Ontology)與 SIOR(Semi-informal Ontology with Refactor)

- AT 為傳統的關鍵字與摘要。
- SIO 為使用半非正規化(Semi-Informal)的本體(Ontology)做為知識模型，但在知識建構上沒有提供重構的指示。
- SIOR 為使用半非正規化(Semi-Informal) 的本體(Ontology)做為知識模型並提供重構的指示。

由於 AT 模型並不考慮可維護性與可延伸性，因此在 AT 模型上不會計算編輯時間與重用次數。

整理實驗的目的與標的如下表：

表 4 實驗模型與模型要素

	Abstract & Tag	Semi Informal Ontology	
Factor\Model	AT	SIO	SIOR
Readability	Reading time/Correct rate		
Maintainability	Can’t compare	Editing time	
Extensibility	Can’t compare	Reuse times	
Completeness	User average keywords number		
Clear Causality	User average relations number		

4.2.2 先備程度對於效益的影響

考慮到不同人的先備程度不一致，為了解先備程度對於效益的影響，因此加入前測，分析使用者對於實驗知識概念的了解程度，再透過可閱讀性做為效益的比較標的。

4.3 實驗設計

為消除單一文章所造成的影響，實驗使用三個主題總共六篇文章進行測試，文章來源為 Wiki 上隨機找尋的結果，符合字數沒有太短，主題文章字數差異不大的條件，分別為兩家公司 Apple 與 Google，兩個都市包括東京與柏林，兩個學術知識分別為技術分析與知識管理，對於三個模型的知識模型初始化建構，則請三個使用者分別對六篇文章進行知識模型建構，讓使用者得以進行閱讀的實驗。

系統在受試者進入時，平均分派三個主題對應三個模型，所以對於受試者而言，所做的模型與主題的配對是隨機的。每一個主題都包含兩篇文章，一篇作為閱讀的測驗，一篇作為建立知識模型的測驗，並且在實驗步驟操作前，提供相關的說明。

閱讀的測驗都包含前測與後測,其測驗正確答案皆平均分佈至四個內容選項。使用者會依序進行前測，閱讀知識模型，其閱讀時間會紀錄在系統之中，最後進行後測。

建立知識模型的測驗,系統提供知識內容給與使用者進行術語與觀點的編寫，系統會紀錄知識元件重用的次數，以及知識元件修改所花費的時間。

整個實驗流程圖如下:

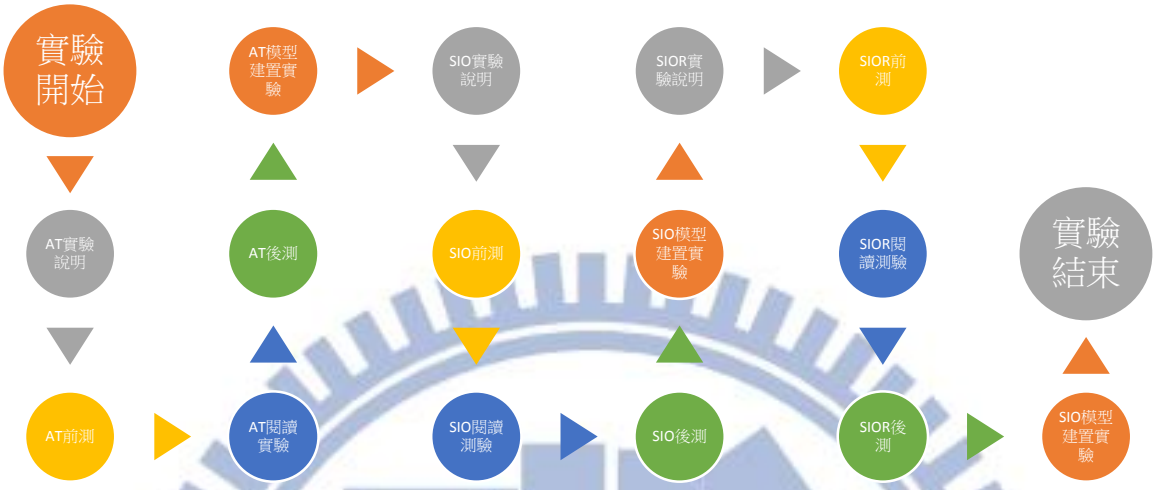


圖 9 實驗流程圖

4.4 實驗系統

4.4.1 系統架構

本實驗系統為兩階層式架構,所有程式皆使用 AngularJS 進行編寫,AngularJS 為 Google 所維護的 JavaScript Framework,網頁排版則使用 bootstrap 進行,伺服器端僅提供 HTML 與 JavaScript 文件到瀏覽器的資料傳輸,資料庫使用的是 Neo4j,資料構通則使用 Restful Service。

4.4.2 Neo4j 圖形資料庫

Neo4j 為一個圖形化資料庫,使用 Java 進行開發、完全相容 ACID。資料以一種針對圖形網絡進行過優化的格式保存在儲存媒介上。 Neo4j 的核心是一種極快的圖形引擎,具有數據庫產品期望的所有特性,如恢復、兩階段提交、符合 XA 等。

對於以關聯路徑相似度做為推薦計算的系統的實現,Neo4j 比傳統的關聯式資料庫快速許多,其原因在於對於關連式資料庫而言,要實現不同的資料節點之間的關聯查詢,代表著需要做 SQL Join,當路徑很長的時候,就必須 Join 大量的 Table,這也代表著查詢速度的減緩,以及大量的記憶體負擔。而對於 Neo4j 而言,

其儲存格式與核心均對圖形搜尋做過最佳化，因此很適合作為關聯路徑計算為基礎的系統使用。

Neo4j 提供類似 SQL 的語法進行資料的查詢，其語法稱之為 Cypher，Cypher 使用 ASCII 的圖形模式描述結點與關聯，例如：

```
match (u：User)-->(i：Insight)
return u.email,count(i)
```

match 語法之後 (u：User)-->(i：Insight)就是圖形模式。

(i:Insight)中的( ) 代表節點,i 為名稱簡寫,Insight 代表該節點有 Insight 這個標籤。

(u：User)-->(i：Insight)中的-->代表 User 的節點對於 Insight 的節點有單方向的關聯性。

整個語法意思則是對每一個 User 找尋有關聯的 Insight，尋找完畢之後，以使用者 Email 為主軸，統計 Insight 的數量並進行回傳。

Neo4j 同時也提供一個圖形化的查詢介面，輸入所要查詢的 Cypher 語法，其結果就如下圖所示，其內容為一個 Knowledge Concept 包含了許多的 Knowledge Content。

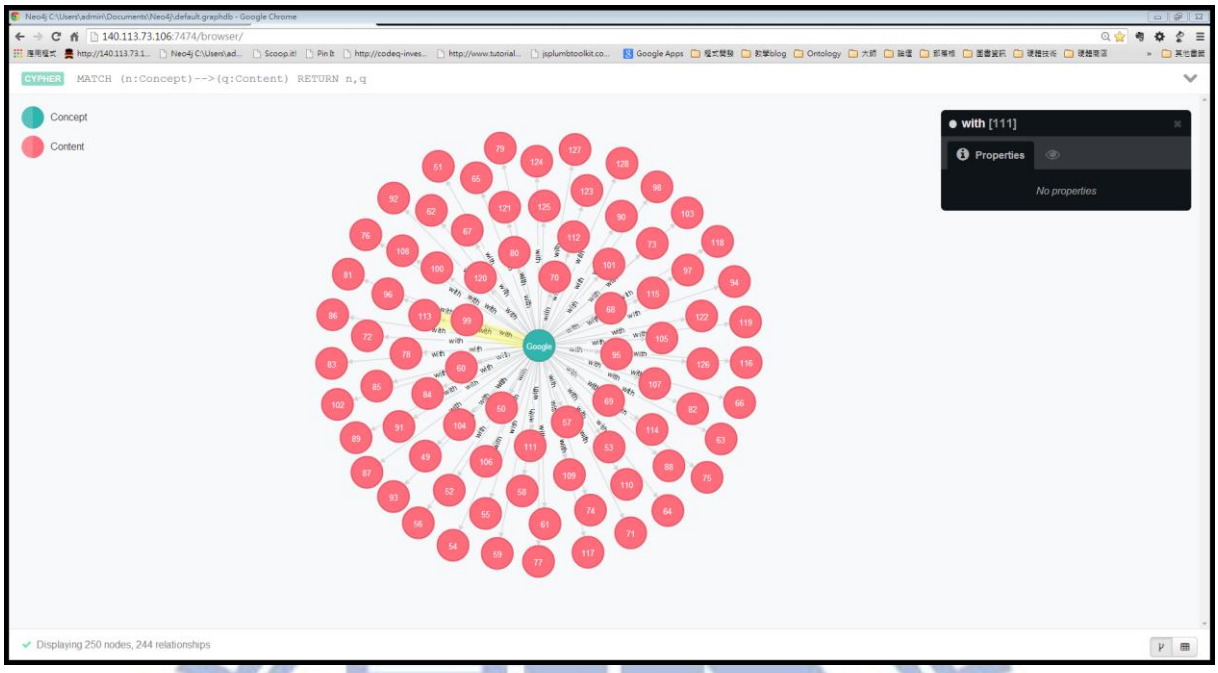


圖 10 Neo4j 主控台查詢結果

4.4.3 系統功能

總共實做了四個主要的系統功能，包括了資料轉換，初始資料維護，統計結果與實驗平台。

- 資料轉換

  基於 JAVA 語言進行開發，主要實做了兩種檔案格式的讀取，包括了 PDF 與 HTML

  - PDF 使用 Apache 所提供的 PdfBox 套件進行讀取，但對中文 PDF 尚無支持。
  - HTML 使用 Jsoup 的套件進行網頁讀取。

  當資料讀取完畢進行分割後，則透過 Spring Data 套件進行轉存資料庫的動作，Spring Data 整合關聯式資料庫與 NoSQL 資料庫的資料存取提供同一的介面，讓開發者學習一套介面後，就可以存取大多數的關聯式資料庫，例如 MS SQL Server, MySQL, Oracle DB…等，也可以存取 NoSQL 的資料庫，例如 Hadoop, MongoDB, Neo4j…等。

- 初始資料維護

使用者可以對於六篇文章進行三個模型維護包括 AT, SIO, SIOR，並可以維護考題，其畫面如下：

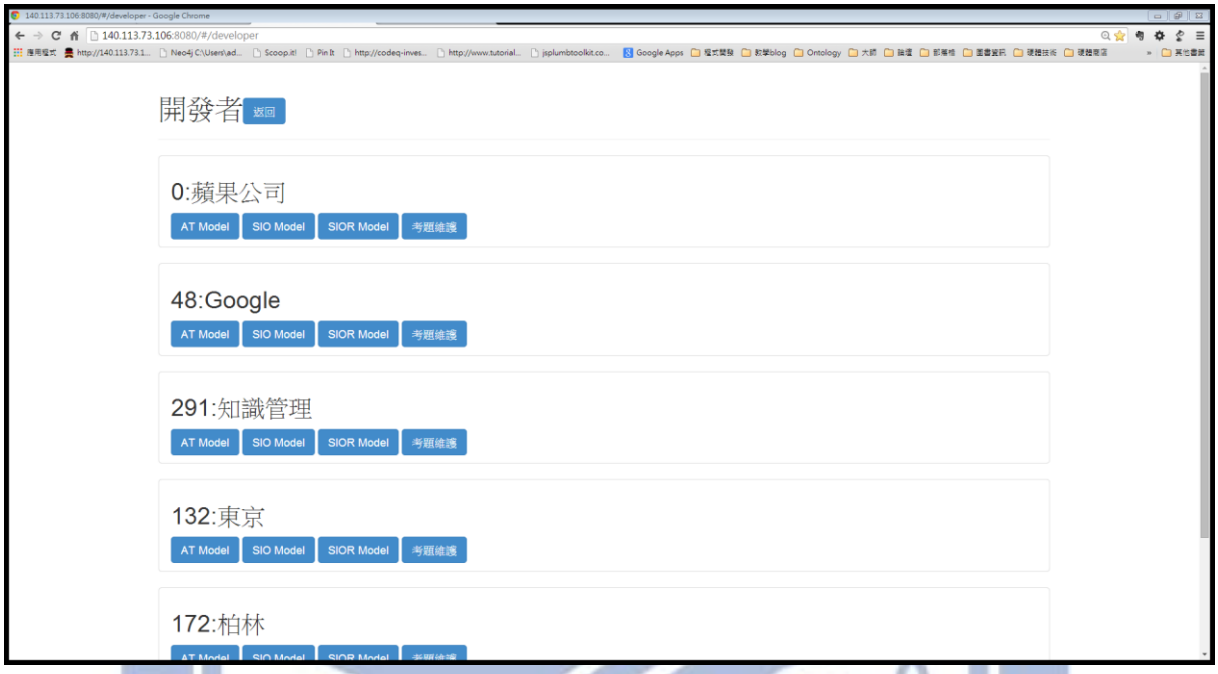


圖 11 實驗系統-資料初始化頁面

- 統計結果

顯示實驗結果，顯示所要量測的指標，包括答題正確率，使用時間以及關鍵字編修狀況，畫面如下：

圖 12 實驗系統-資料統計頁面

- 實驗平台

提供前後測，模型的閱覽，模型的編修，模型測試，畫面分別如下所示：

■ 前後測

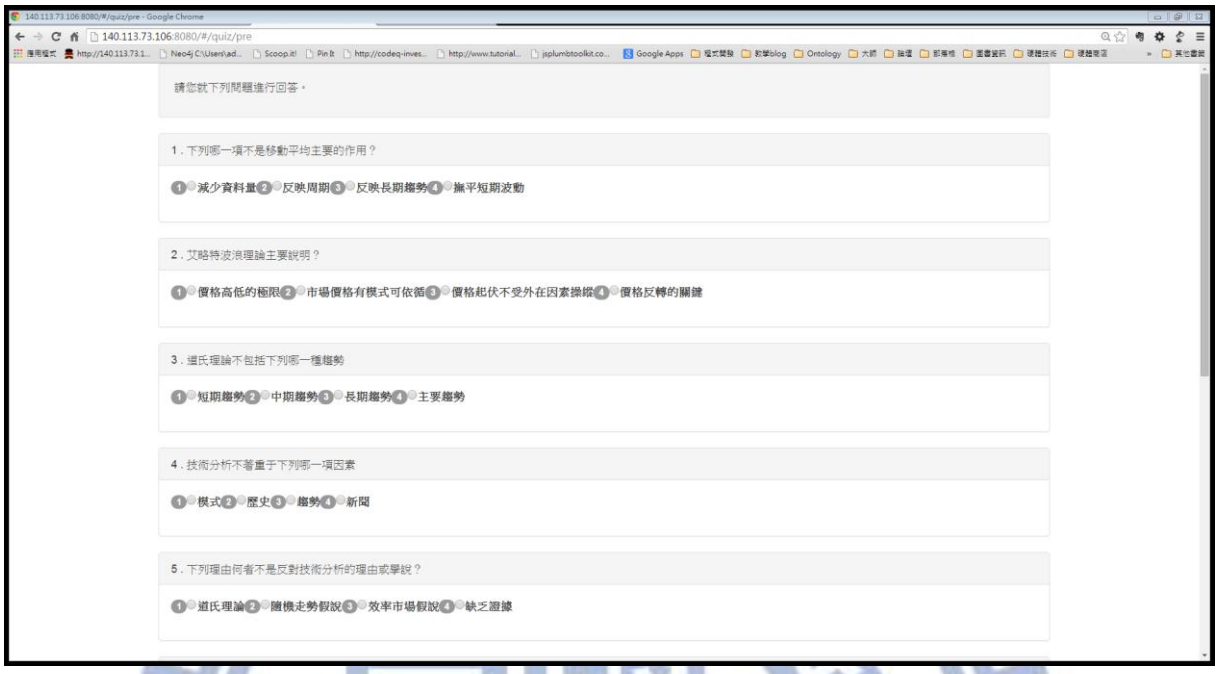


圖 13 實驗系統-問卷頁面

■ 模型閱覽:在畫面顯示上,遵循唯一呈現模式(Unique Presentation)同一個知識元件使用相同的顏色與格式作呈現。並且依點擊知識內容，做知識模型的呈現避免資訊洪流(Information Flood)。

圖 14 實驗系統-模型顯示頁面

- 模型編修：採用一群組一區域(One Group One Area)的方式，將不同知識元件編修做隔離，使編輯時更為清楚。主要提供了重用別人的知識元件，重用自己的知識元件與新增全新知識元件的功能。

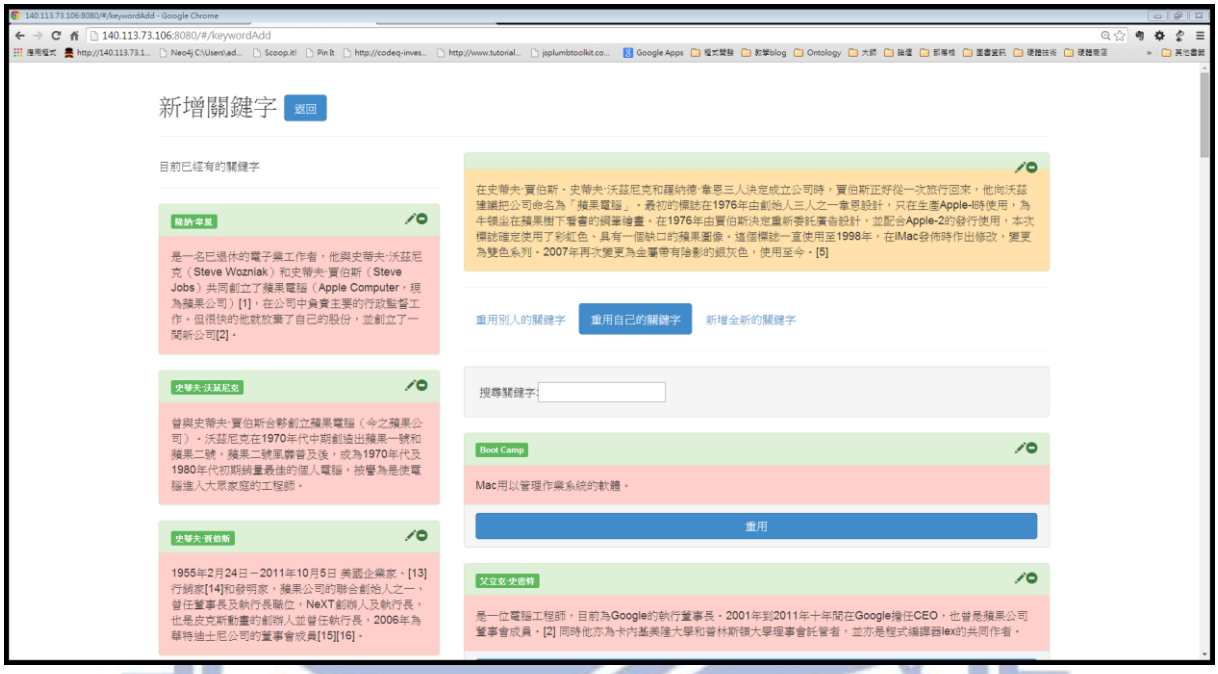


圖 15 實驗系統-新增模型頁面

- 模型測試：於編修結束時，顯示檢覈表，讓使用者進行確認。

圖 16 實驗系統-模型測試頁面

第五章、 實驗結果分析

在本章中，將就知識元件的效益，初學者與非初學者的效益差異性比較，是否有社會化的現象三個大項進行說明。

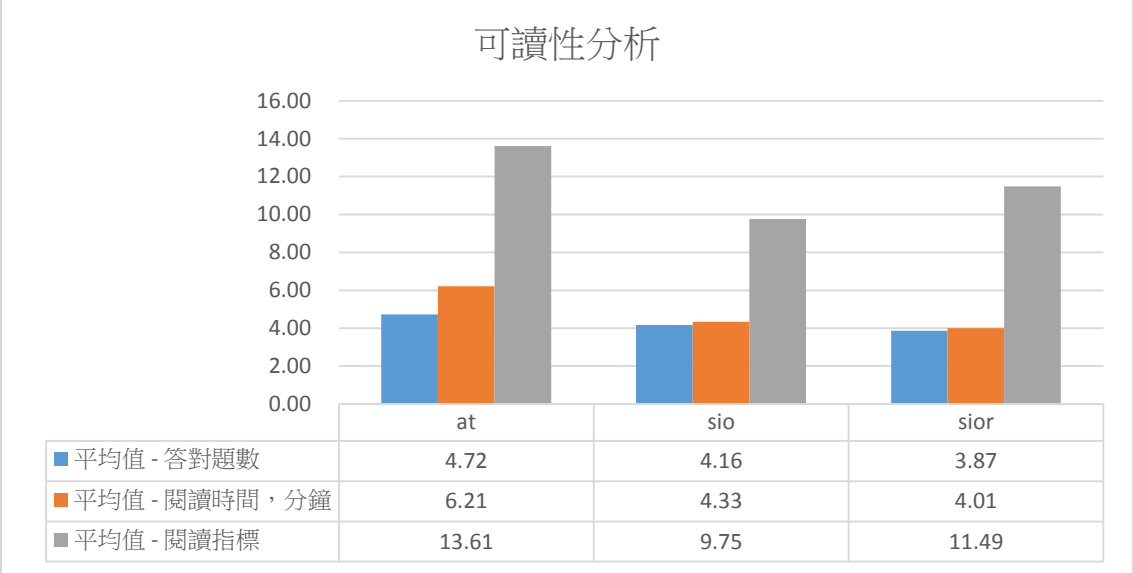
5.1 受測資訊統計

測試期間：2014/6/25~2014/7/2，總共樣本數為 70 個，有效樣本數為 45 個

5.2 知識元件的效益

5.2.1 可閱讀性

可讀性越高，代表模型所表示之知識越容易被理解，下圖展示不同模型下的實驗結果，其閱讀指標代表閱讀成本，為閱讀時間除以答題正確率。



	at	sio	sior
平均值 - 答對題數	4.72	4.16	3.87
平均值 - 閱讀時間，分鐘	6.21	4.33	4.01
平均值 - 閱讀指標	13.61	9.75	11.49

圖 17 可閱讀性分析

閱讀指標越小代表其可閱讀性越高，以閱讀指標來看可以發現 SIO 優於 SIOR 優於 AT，進一步可以發現，SIOR 與 SIO 在對平均閱讀時間與平均達對題數進行分析比較，其實可以發現兩者相去不大，在閱讀性上兩者差不多。

5.2.2 可維護性

可維護性越高，代表再重新使用知識元件時，所需要花費的修改成本越短，幫助使用者更快的將原本的知識元件，修改成自己所需要的版本，可維護性指標為平均知識元件修改時間。

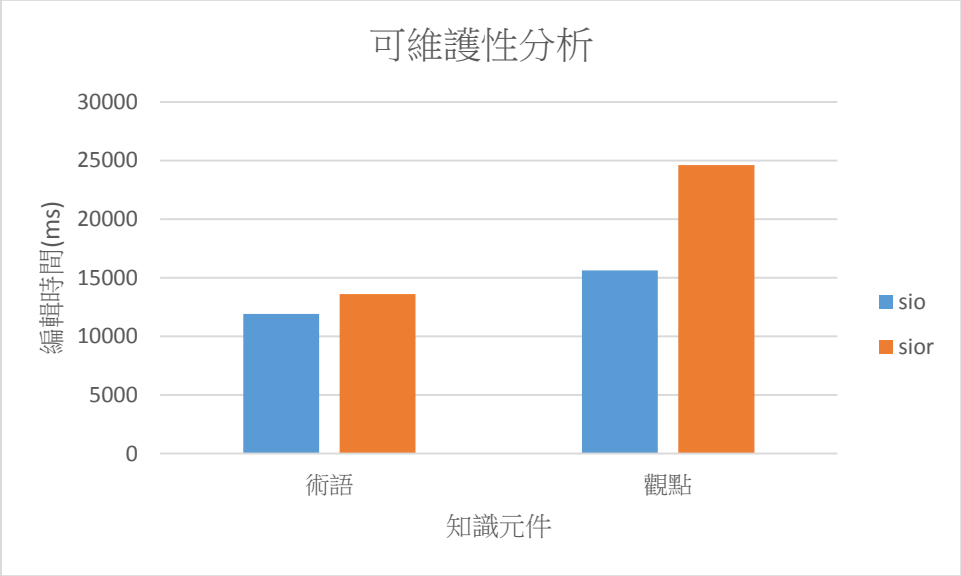


圖 18 可維護性分析

圖中可以發現 SIOR 的平均時間，都大於 SIO 的平均時間，其原因在於要將知識元件修改成為符合 SIOR 的標準所需花費時間較高。

5.2.3 可延伸性

可延伸性越高，代表越容易重用知識元件，因此以平均知識元件重用次數作為量測指標。

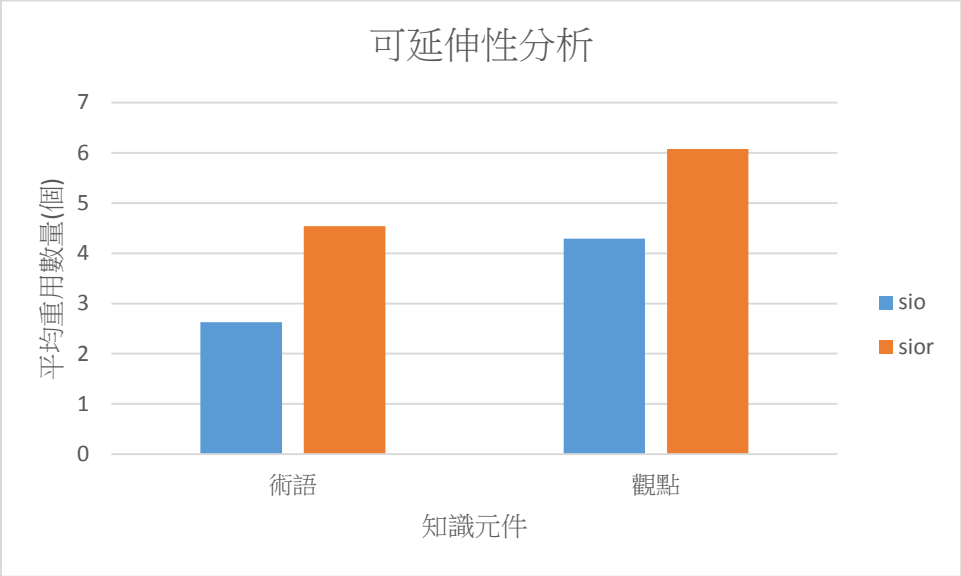


圖 19 可延伸性分析

在延伸性上，SIOR 不論是術語或觀點其重用的數量都高過 SIO，因此依照 SIOR 指示所產生的知識元件較容易被使用者重用。

5.2.4 完整性

完整性越高，代表其知識模型在表現同一個知識概念上越完整，其指標為術語的平均數量，包含了新增，修改與重用的數量。

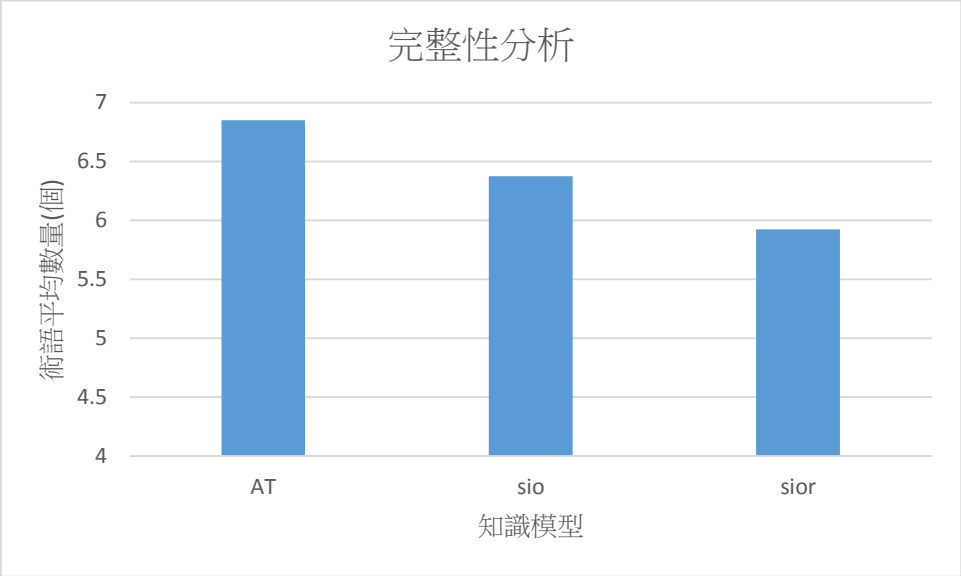


圖 20 完整性分析

如圖所示，與預期的的結果相反，原本預測應為 SIOR > SIO > AT，其原因在於實驗時間過久，SIOR 模型在最後一個，到最後使用者沒有耐心進行，因此本實驗結果反而呈現 SIOR < SIO < AT。

5.2.5 關連明確性

關連明確性越高，代表其知識模型在表現同一個知識概念上關聯訊息表現越完整，其指標為平均產生的關聯數量，包含所有術語與觀點，同一個術語可以再多個知識內容被重複使用。

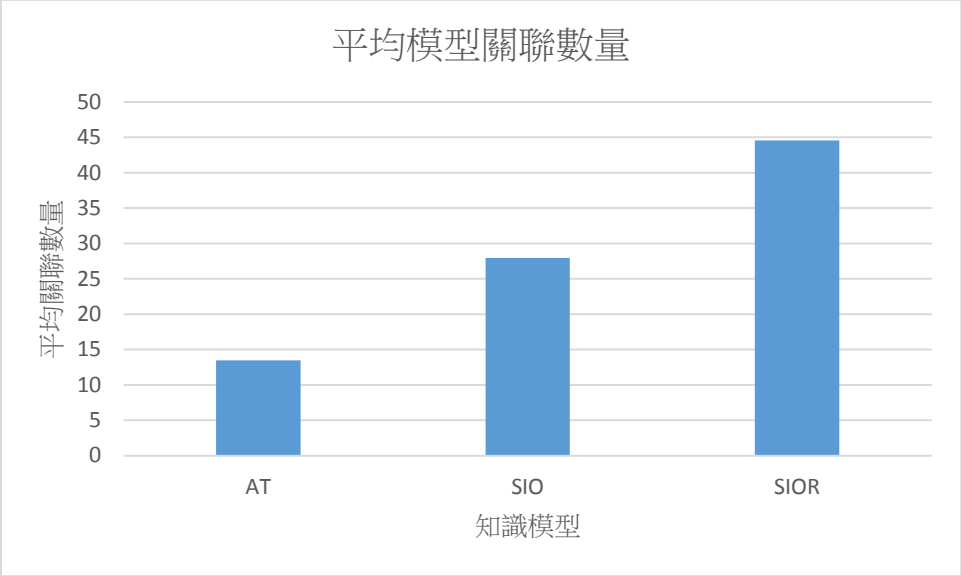
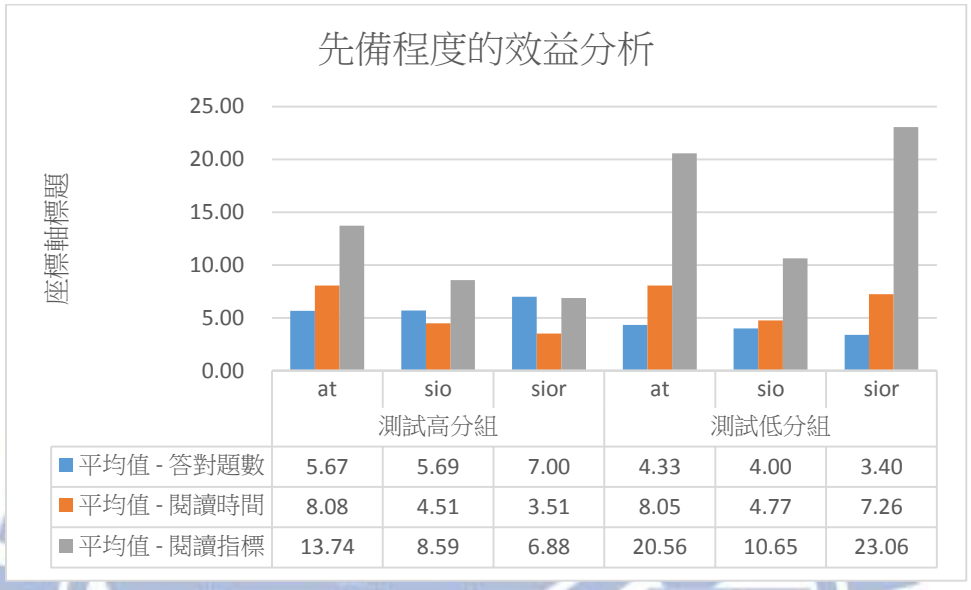


圖 21 關連明確性分析

這邊的關聯包含了術語與觀點的關聯數，從這邊可以觀察到 SIO 與 SIOR 的關聯數量約略為 AT 模型的兩倍到五倍間，關聯性越明確有助於使用者在搜尋時，能夠透過術語或重點，快速的連接到知識內容，反過來也可以透過所參考到的某一個知識內容，找到其他的術語或觀點。這些關聯也是相似度分析的基礎。

5.3 先備程度的效益分析

將平均達對題數，平均閱讀時間，與平均閱讀指標依前測高分群與低分群進行整理，如下表所示：



	測試高分組			測試低分組		
	at	sio	sior	at	sio	sior
平均值 - 答對題數	5.67	5.69	7.00	4.33	4.00	3.40
平均值 - 閱讀時間	8.08	4.51	3.51	8.05	4.77	7.26
平均值 - 閱讀指標	13.74	8.59	6.88	20.56	10.65	23.06

圖 22 先備程度的效益分析

可以在這邊發現先備程度較低，模型對其幫助效益不大，主要還是看個人理解的能力，但是如果提高其先備程度，例如對於該知識載體已經有建立過知識模型，則效果則會非常明顯。

5.4 實驗總結

首先先整理本次實驗所發生的問題，在對實驗內容進行總結。在本次的實驗當中發現下列問題：

- 實驗時間過長

由其對於網路受試者而言，受試者所願意花費的時間是有限的，由於本實驗一次需要驗證三個模型，對於受試者而言在固定的時間狀況下，越後面的模型其樣本數越少。

- 實驗過於複雜：

可以發現本實驗對於網路受試者來說，閱讀部分願意花比較多的時間來進行，但是對於知識建模，願意仔細進行的受試者不多，有不少的測試者一個知

識模型都沒有建立，必須把這些樣本排除在外。

可閱讀性的部分整體來看，SIO 與 SIOR 並沒有明顯的差別，但優於 AT 模型，但是對於一開始就比較了解這些文章內容的人，SIOR 比 SIO 更有效果，對於可延伸性的部分 SIOR 高於 SIO，明確關聯性部分，可以證明 SIO 與 SIOR 均比 AT 模型產生更多的關聯性。完整性的部分與可維護性的部分與設想的相反，完整性受到實驗過程影響因而無解釋力，可維護性的部分則因 SIOR 條件比 SIO 更為複雜，因而需要花費更多的時間。

第六章、 結論與建議

本章分為四個部份，第一部份為本研究的貢獻，第二部分為管理意涵，第三部分為如何進行實務應用，第四部份則說明未來研究之建議與方向。

6.1 研究貢獻

本研究貢獻分別敘述如下：

- 整合不同領域知識包括知識工程中的本體(Ontology)、軟體工程中的重構(Refactor)與電子商務中的推薦(Recommend)，提出知識漸進式演進的循環架構，能夠幫助使用者對知識進行歸納與了解，該架構可以不斷循環，讓知識載體與知識模型都可以不斷演進。
- 整理不同的知識模型，提出一個新的基於 Semi-informal Ontology 的知識模型，幫助使用者能夠快速的進行理解，整理所需要理解的知識。
- 詳細描述知識重構的方式，讓使用者在建立知識模型時，能夠有所依循。並提出以知識反模型與模型特性做為測試案例，將測試驅動導入建構知識模型的程序中。
- 提供推薦的方法讓知識可以重用，降低建立新知識模型與搜尋知識的成本。

6.2 管理意涵

對於知識管理而言，本研究提出新的知識模型，運用其架構可以幫助知識閱讀者快速的理解與重用，提升其閱讀的效率，由其是當先備知識足夠的時候，其幫助更是顯著。透過知識元件化與元件化，使得知識可以再利用，避免使用者重工，將有助於知識的架構與創造。透過知識的循環架構，可以不斷的對既有的知識進行品質的改善，讓知識管理能夠更加的落實。

舉例來說，由於知識密集性的產業時常定義新名詞，對於這些新增的術語，如果沒有好的儲存方法來進行整理，隨著公司的成長與人員的流動，會變成沒有人完全理解一份知識載體內所敘述的內容，會變成東拼西湊，最後產生大量的誤解，導致開發新產品遭遇許多定義混淆的問題。

依照知識建構理論，知識的創立，會先經過前三個程序包括搜尋，鞋盒與評估，

再由後面的三個階段所產生新知識，包括建立架構，訂定假說與呈現，由於本研究可以透過重用，加速前三個階段的速度，因此可以減少知識的創立的周期。

組織知識隨著組織的成長與人員的流動而造成品質低落，透過本研究的知識循環架構，將可以增進組織知識的品質，避免新進人員無法快速與公司知識圈進行接軌，不但造成人力資源的浪費，更有可能會讓新進人員感到無法適應而離職。

6.3 實務應用

以公司知識管理來說，可以將已經存在的許多知識載體進行整理，這些知識載體很多具備了知識泡泡，冗余知識，重複的知識等知識反模式，經過整理過後，可以提升使用者對這些知識的利用效率，還有這些知識載體的品質。當知識能夠活用，公司才不會不斷的空轉，才會有餘力去進行創新，對於生存於這個快速變遷的時代，如果沒有創新，很容易就會被取代而消滅。

以教育學習來說，教材可以使用本研究的知識模型，讓學生能夠協同合作建立共通的知識模型，增加學習效率，改善原有教材的品質。由於經過學生協同合作編修，可以去除教材編輯者的盲點，讓教材更加的適合學生閱讀。

以學術研究來說，論文的整理可以使用本研究的知識模型，再透過知識建構的理論，使用元件化的方式整理論文架構，並且可以透過協同合作，讓整個實驗室的知識可以進行流通，累積對於實驗室領域共通的論文知識模型，對於重要的論文都可以加以整理，讓後進者可以快速的了解實驗室的研究領域，加速學術研究的效率。

6.4 未來研究方向

本研究著重於文字知識載體的部分，知識載體還包括了圖片，聲音及影像等類型，未來可以就這些知識載體做進一步的研究。對於概念層，本論文只有整理出應該具備的項目，對於實際應用只有提出關於知識概念的推薦，後續可就概念層進行更多的研究與實驗。

對於知識螺旋的知識管理模型，本研究已經包含了知識的內化，外化與組合化，只有社會化並沒有討論與驗證，社會化為隱性知識的轉移，在淺移默化中，所有人員達成一致的共識並遵循，而本研究提供的他人知識的重用，如果這些重

用的選擇會集中於某些知識元件，或著是某些知識的創作者，那代表著在本研究的知識框架具備有社會化的現象，如此在知識管理上，本研究架構則可以完整的映射到知識螺旋所表達的內化，外化，社會化與組合化，其管理意涵可以更加的豐富。

另外對於人工智慧為基礎的知識庫也可以做為互補，只要能夠將原有的人工智慧知識庫與本研究的知識模型進行關聯的對應與整合，透過人工智慧的知識庫可以進行知識推論，並且使用本研究的知識模式進一步的幫助使用者理解其推論結果幫助知識的使用。

在知識的建置上面，可以研究透過知識工程中的知識萃取進行初步知識的擷取，畢竟建立知識模型需要花費大量時間，如果可以進行部分程度的自動化，則可降低使用者建立知識模型上的困難，讓系統的導入更加的順利。

網際網路上的實驗有其限制，因此就實驗的部分，可以於公司內部或是學校內重做，可以得到比較強力的證明。此外由於本實驗系統重無到有花費太多時間，對於實驗的設計上，僅能做到效益初步的證明，後續由就可就效益實驗進行更進一步的分析，諸如一個知識元件多少字數較為適合，不同語系是否實驗結果也會有所不同。

參考文獻

1. Bates, M. J. (1989). The design of browsing and berrypicking techniques for the online search interface. *Online Information Review, 13*(5), 407-424.
2. Beck, K. (2003). *Test-driven development: by example*: Addison-Wesley Professional.
3. Biggerstaff, T. J., & Richter, C. (1989). *Reusability framework, assessment, and directions.* Paper presented at the Software reusability: vol. 1, concepts and models.
4. Bobillo, F., Delgado, M., & Gómez-Romero, J. (2008). Representation of context-dependant knowledge in ontologies: A model and an application. *Expert systems with applications, 35*(4), 1899-1908.
5. Chaudhri, V. K., Farquhar, A., Fikes, R., Karp, P. D., & Rice, J. P. (1998). *OKBC: A programmatic foundation for knowledge base interoperability.* Paper presented at the AAAI/IAAI.
6. Devedžić, V. (1999). A survey of modern knowledge modeling techniques. *Expert systems with applications, 17*(4), 275-294.
7. Fowler, M., & Beck, K. (1999). *Refactoring: Improving the Design of Existing Code*.
8. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns: elements of reusable object-oriented software*: Pearson Education.
9. Gruber, T. R., & Olsen, G. R. (1994). An Ontology for Engineering Mathematics. *KR, 94*, 258-269.
10. Hilbert, M., & Lopez, P. (2011). The world's technological capacity to store, communicate, and compute information. *Science, 332*(6025), 60-65. doi: 10.1126/science.1200970
11. Jenkins, C., Corritore, C. L., & Wiedenbeck, S. (2003). *Patterns of information seeking on the Web: A qualitative Study of domain expertise and Web expertise." IT& Society 1 (3): 64.* Paper presented at the IT & Society.
12. Kerievsky, J. (2005). *Refactoring to patterns*: Pearson Deutschland GmbH.
13. Neches, R., Fikes, R. E., Finin, T., Gruber, T., Patil, R., Senator, T., & Swartout, W. R. (1991). Enabling technology for knowledge sharing. *AI magazine, 12*(3), 36.
14. Nielsen, J. (2003). Information foraging: Why Google makes people leave your site faster. *Jakob Nielsen's Alertbox*.
15. Opdyke, W. F. (1992). *Refactoring object-oriented frameworks.* University of Illinois.
16. Pirolli, P., & Card, S. (2005). *The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis.* Paper presented at the Proceedings of International Conference on Intelligence Analysis.

17. Ramer, S. L. (2005). Site-ation pearl growing: methods and librarianship history and theory. *Journal of the Medical Library Association, 93*(3), 397.
18. Rech, J., Decker, B., Ras, E., Jedlitschka, A., & Feldmann, R. L. (2007). The quality of knowledge: Knowledge patterns and knowledge refactorings. *International Journal of Knowledge Management (IJKM), 3*(3), 74-103.
19. Robertson, S. E. (1977). Theories and models in information retrieval. *Journal of Documentation, 33*(2), 126-148.
20. Schafer, J. B., Konstan, J., & Riedl, J. (1999). *Recommender systems in e-commerce.* Paper presented at the Proceedings of the 1st ACM conference on Electronic commerce.
21. Uschold, M., & Gruninger, M. (1996). Ontologies: Principles, methods and applications. *The knowledge engineering review, 11*(02), 93-136.
22. 潘旭偉, 顧新建, 仇元福, & 程耀東. (2003). 面向知識管理的知識建模技術. *計算機集成制造系統, 9*(7), 517-521. doi: 10.3969/j.issn.1006-5911.2003.07.003