私立東海大學資訊工程與科學系研究所

碩士論文

指導教授：朱正忠　教授

以模型為基礎之物件導向需求編輯器

A Model-based Object-Oriented Requirement Editor

研究生：鄭雅文

中　華　民　國　九　十　七　年　七　月　十　五　日

摘　　要

對於應用資訊系統開發者而言，開發合乎使用者需求的軟體始終是一大重要課題。在軟體開發生命周期中，定義出新系統的需求是較為重要的一部分，然而大多數的需求文件仍是以自然語言的文字格式撰寫而成，如此一來易造成充滿模稜兩可(Ambiguity)、不一致(Inconsistency)、不精確(Imprecision)、與不完整(Incompleteness)的情況。除此之外，若將這種需求文件轉換成軟體開發中的物件導向分析以及物件導向設計階段所需的文件時，將會投入更多的轉換成本。

為解決這個問題，本論文提出以模型為基礎之物件導向需求工程。由於物件的繼承特性，可以帶來許多設計上的優勢，物件導向設計方法已儼然成為系統設計的新潮流。我們將需求定義為物件，並將其分類，以預先定義的方式管理需求物件與其屬性。本論文所提出之方法可幫助系統開發者有系統地擷取需求，並且可以讓接下來的分析階段、設計階段使用，而不會有不一致的情況產生。以模型為基礎之物件導向需求編輯器不但可以增加需求階段的效率，也可以幫助簡化軟體開發流程裡的需求階段轉換到分析、設計階段，進而解決不一致的情況產生，降低開發成本。

關鍵詞：需求工程、物件導向、需求管理

Abstract

It is an important subject for information system developer to come up the application software that meets user's requirements. And one of the key tasks in software development cycle is to correctly define the new system requirement. Unfortunately most of the user requirements are written in natural language format. It can easily lead to a situation which is full of ambiguity, inconsistency, imprecision and incompleteness. At the same time, it will incur a lot of conversion cost and time when trying to convert such written requirements into object oriented analysis and design documents.

To resolve such problem, this paper proposes an object oriented structure that can be used to specify, analyze and manage the software requirements. An object has its natural characteristic of inheritance. This characteristic brings certain design advantages. So object oriented design methodology has become a new development trend. We can define the user requirement as an object and classify it. We can then easily manage the requirement object and its characteristic with some predefined methods. The proposed structure will also help system developer to access to the requirement systematically. It can be used furthermore in the analysis and design development cycle without creating any inconsistency. Based on modeling, object oriented requirement engineering can enhance the efficiency during the requirement cycle. It can also simplify the process in software development such as the stage conversion from requirement into analysis and design. It will further resolve the inconsistency issue. It can help to reduce the development cost as well.

Keywords: Requirement engineering, Object-oriented, Requirement management

目　錄

圖目錄

表目錄

第一章　導論

1.1前言

近年來，由於軟體產業的發達，軟體系統的需求量愈來愈大，複雜度與異質性也越來越高。而現今最常見的軟體開發失敗的主要原因幾乎是無法產生完整、正確且清楚的軟體需求規格。雖然軟體工程的概念已經被提出數十年，但始終對於軟體工程的導入無所適從，一方面是軟體工程導入不易，開發人員教育訓練時間過長，另一方面導入軟體工程需撰寫數量眾多的文件，而這龐大的工作量無疑會使忙碌的系統開發師更加忙碌。

在軟體開發生命週期中第一個開發流程即為軟體需求分析，也是整個軟體開發生命週期中較為重要的部份。系統分析師從一開始與客戶訪談，將訪談資料整理分析成需求分析文件，並反覆與客戶確認需求內容，最後產生最終確認的需求分析文件。但現今大多數的需求分析文件都是以自然語言的方式來描述，不同的系統分析師對於相同的需求可能會有不一樣的描述方式。此外，如果系統分析師與客戶對開發的軟體系統的相關背景不太了解，可能就會有需求不一致或是不明確的情況產生，如此一來就會導致後續的系統設計、程式開發與軟體測試等階段產生錯誤。且需求分析階段還會產生數量龐大的人工產出物(Artifacts)，例如

UML 圖、程式碼和文件等，得想要整合或追蹤其他階段的產物只能倚賴人工，進而造成時間上的浪費以及容易發生錯誤，這個缺點也讓我們在導入 CMMI 相關活動時，會花費相當高的成本。

有鑑於此，為使軟體開發流程更加順利、簡便，我們提出以模型為基礎之物件導向需求工程的概念，並開發了一套以模型為基礎之物件導向需求編輯器。在本系統中，每個軟體需求即為一個物件，並設定相關屬性來描述此軟體需求。物件定義完畢後，即可重複呼叫使用此物件，若未來物件的內容有所改變，位於不同位置的相同物件則會自動更新內容，如此即可解決需求不一致的情況產生。本研究還根據不同的系統軟體特徵、過去開發系統的經驗與特定領域的專業知識，建構出多樣化的專家需求樣板。有了專家樣板後，就算接觸到不熟悉的資訊領域，也可透過專家樣板內容引導系統分析師撰寫出簡易的軟體需求。除了使用現有的專家樣板，系統分析師也可自行定義新的需求樣板。未來希望藉由本研究所開發之以模型為基礎之物件導向需求編輯器協助軟體開發專案順利進行，節省系統開發成本。

1.2 研究動機與目的

在任何軟體開發生命週期中第一個步驟便是定義出新系統的軟體需求，從一開始與開發專案有關的專案關係人(Stakeholder)蒐集需求資訊，到後期分析所蒐集的需求並加以分類、整理與文件化，使軟體需求成為軟體開發生命週期後期的設計、實作與測試的參考標準。

從需求的擷取、分析、正規化到最後的需求確認，這些與需求相關的步驟統稱為需求工程[6]。在需求工程的各個階段，會遭遇到許多的困難與問題，這都需要一一被克服，才能完成滿足使用者需求的軟體系統，而不好的軟體需求(Software requirement) [10][16][3]也已經被認定為是造成成本浪費及行程延誤的主要原因。由此可知，需求階段的活動其實是最需要專業的投入，但目前大多數的軟體產業都是以非專業的方式來處理相關之工作，這都歸於大部分的專案關係人若不是沒有受過良好的訓練，就是缺乏相關的領域知識。

由於現在大部分需求文件都是由 Microsoft Office 的 Word 來做編輯的動作，而 word 跟大部分文字編輯器一樣，只能將使用者所想要的需求記錄下來，而需求之間的關係也只能藉由文字來描述。而在本研究中，透過物件導向技術與需求的結合，需求將會用物件的觀點來表示，每個需求是由最基本的物件來表示，而需求所需要的相關數值或是條件可以

利用物件導向技術當中的屬性來表示。而物件與物件之間的關係可以經由整合物件導向技術的編輯器裡的格式，如字型、大小、顏色等等來表現。而相較於傳統編輯器的不同，整合物件導向的編輯器可以表現出物件導向之間的特性-繼承。而在撰寫需求時，如果需求文件裡面有相同的需求時，亦或是有大部份相同的需求，就可以利用繼承的觀念，讓有比較特殊限制的需求去繼承比較一般化的需求，然後再藉由屬性的運用來達到需求特製化的目的。而在需求階段，如果當中的需求有所改變，根據繼承的特性，只要上層的需求修改，那麼底下所有繼承的子需求，便會同步更新，就不會造成前後不一致的狀況發生。另外，由於有繼承的特性，那麼便表示也會有階層的特性存在，如果真的在需求階段當中，出現了不一致的狀況發生，或是其他的錯誤產生，那麼就可以逐層檢查，找出發生問題的地方。這與傳統編輯器來做比較，編輯能力更為強大。此外，導入 model-based 還有一項好處就是，讓成功的設計與架構能夠輕易的被再利用，如此一來，每一階段的人工產出物(Artifact)品質便能夠獲得改善，維護的成本也能夠降低。

1.3 研究方法與步驟

本研究的進行，先調查現有軟體工程相關文獻，包括系統模組化、需求工程、需求管理、需求追蹤等方法、理論與相關的商用或是學術研究工具。並著手計畫、設計出以模型為基礎之物件導向需求架構，再加以描述與架構中各個模組(module)相關的定義，以及各部份系統元件(component)的說明。

接下來，則實作我們所提出的架構，並且進行個別元件的測試，確定無誤之後，整合所有的元件做整合性的系統測試。最後，使用依實際案例對系統進行評估，使我們的研究能得到更客觀的測試結果、對系統有更正確的評估。

1.4章節導讀

論文的章節安排分述如下：

第二章　背景知識與相關研究，此章節說明與本論文相關的研究，並介紹物件導向與需求工程的觀念及其相關的應用。

第三章　說明本研究最重要的元素-需求如何建立與其分類，並以物件的方式表示所有的需求，與各個需求物件與需求物件間的關係，亦於此章節詳細說明。

第四章　透過一個實際案例來講解本研究所提出之方法。

第五章　結論及未來方向，此章將本論文之研究成果做總結，並提出未來研究之方向。

第二章　背景知識與相關研究

2.1軟體工程

2.1.1軟體工程簡介

近幾年來，隨著電腦科技的快速發展，軟體與硬體產業愈來愈發達。目前全世界幾乎所有的國家都必須依賴著複雜的電腦化系統來處理大大小小的事務，並且也有愈來愈多的商品結合了電腦的軟硬體技術。隨著科技的進步，人們對於電腦科技的需求卻從未減少過，反而更加期望電腦能夠幫忙人類處理更多、更複雜的事情。一套完整的資訊系統，只有少部分的成本花費在硬體上面，而軟體佔系統總成本的比例卻非常高，並且逐漸增加中。目前軟體系統的需求量越來越大，複雜度與異質性也愈來愈高，若開發軟體並無使用標準化的開發方法，或是開發流程不夠完善，則會導致軟體開發時間延遲、成本比預估時超出許多、執行效率不佳、難以維護等問題產生。

根據美國一間著名的市場研究機構 Standish Group 對軟體專案團隊進行大規模的調查[32]， 在 2006 年的報告(如圖 1)中顯示出只有 35%的軟體專案能符合預算且準時交付軟體；有 19%的專案則因為程式偏離需求，或是無法解決問題，導致在專案完成之前就被取消；佔最大比例(46%)的情況即為，軟體雖順利交付，但卻超出預算或時間，或是只完成某部

份的功能和特性。雖然 Standish Group 所調查的單位是美國的軟體開發組織，但目前台灣的軟體開發情形也是相同的，延遲交貨、品質不良、或不符合客戶需求等事情也時有所聞。

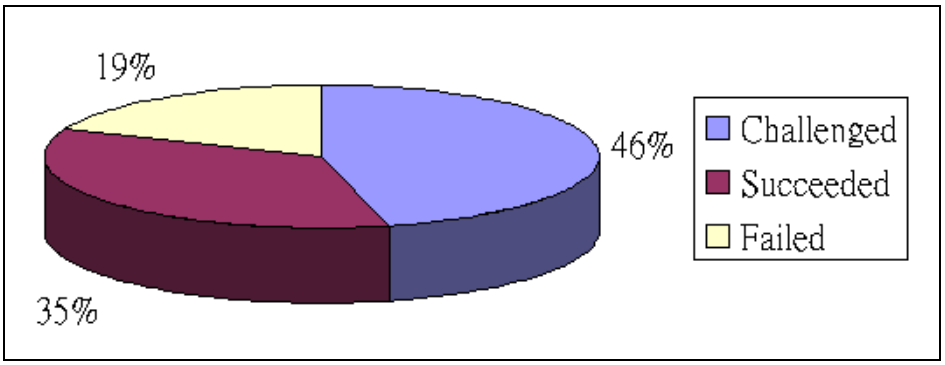

圖 1 軟體專案開發的成功率[32]

為了解決上述軟體系統開發所遭遇到的困難，北大西洋公約組織(North Atlantic Treaty Organization, NATO)招集了近 50 名的計算機科學家、程式設計師和工業界巨頭於 1968 年時在德國舉辦了首次的軟體工程學術會議。此會議主題為討論軟體危機的對策，並提出了以「軟體工程」來規範軟體開發時所需的專業知識，同時也建議了軟體開發應該是以類似工程的活動來進行。

2.1.2軟體工程的定義

軟體工程主要包含了軟體專案管理與軟體開發技術這兩方面的內容，如圖 2 所示，而這兩方面的專業領域則來自於管理科學與資訊科學。軟體專案管理包含了軟體量測、項目估算、進度管控、組織人員管理等，軟體開發技術則包括軟體開發方法論、軟體工具和軟體工程環境。但隨

著焦點的不同，各個學者及專家對於軟體工程則有著不同的見解，以下為目前已知的定義：

- 創造並使用健全的工程原則，以便經濟地獲得可靠且高效率的軟體[1]。
- 利用系統化、遵循規則、可量化的方法來開發、操作及維護軟體[13]。
- 建造由工程師團隊所開發之大型軟體系統有關的知識或學科[7]。
- 對軟體分析、設計、實做與維護的一種系統化方法[29]。
- 目標是生產品質優良、可準時交貨、符合預算，並滿足客戶需求的軟體開發知識與方法[26]。

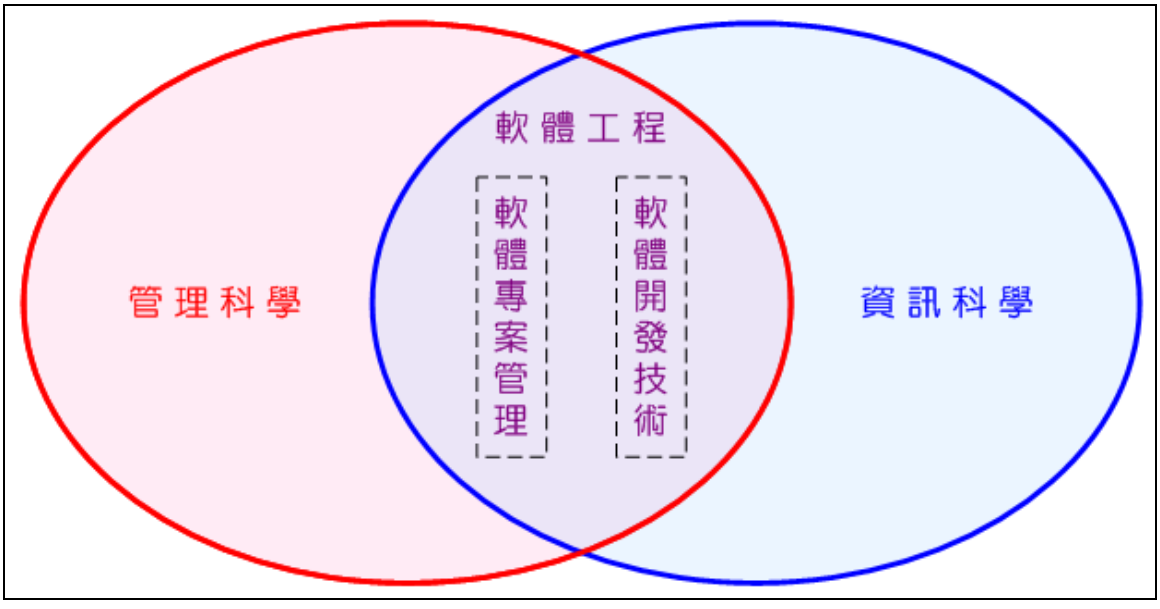

圖 2 軟體工程定義

總結上述各學者與專家的定義，軟體工程可被描述為是一門研究如

何系統化、正規化和量化等工程原則和方法來達到經濟效益之軟體開發和維護的理論。

2.1.3軟體工程的核心知識

在前兩小節中簡單的介紹了軟體工程的由來與定義，但實務上僅瞭解基本的定義是不夠的。軟體工程是一門獨立的學問，要依照軟體工程的流程來進行軟體開發會遭遇到許多問題，而要解決這些問題需要專門領域的知識。在 1993 年美國 IEEE 的計算機協會和 ACM 組成了軟體工程協調委員會(Software engineering Constituent Committee, SWECC)，並在 2004 年 6 月完成了軟體工程知識體(Software engineering body of knowledge, SWEBOK)2004 版全文。SWEBOK 完整的描述出實踐軟體工程所需具備的核心知識，詳細的介紹如下[29]：

- 軟體需求(Software Requirements):

  找出與確認欲製作之軟體所需具備的所有功能，必加以分析且撰寫成正式的文件。本論文主要針對此部份開發一套協助系統分析師撰寫正式需求文件的需求編輯器。

- 軟體設計(Software Design):

  根據所分析的軟體需求定義出軟體系統的基本架構，需將每項細節與系統細分為模組，並定義出每一模組的界面。

- 軟體建構(Software Construction):

  軟體的實作階段，包含有程式撰寫與除錯。

- 軟體測試(Software Test):

  經由軟體測試階段來找尋軟體的瑕疵，並評估此軟體的效能。

- 軟體維護與更新(Software Maintenance):

  改善現有的軟體功能，且須同步修改相關文件與執行適當的軟體測試。

- 軟體配置管理(Software Configuration Management, SCM):

  包含所有與軟體相關的程式和文件的修改與版本控制。

- 軟體工程管理(Software Engineering Management):

  管控軟體專案的進行與軟體的團隊組織。

- 軟體開發流程(Software Development Process):

  所有關於可改善軟體開發的品質、時間、生產力等活動。

- 軟體工程工具與方法(Computer-Aided Software Engineering, CASE):

  支援軟體開發的工具與方法，如電腦輔助軟體工程(Computer-Aided Software Engineering, CASE)工具。

- 軟體品質(Software Quality):

  評估軟體品質、確認是否具有足夠的可靠性等活動。

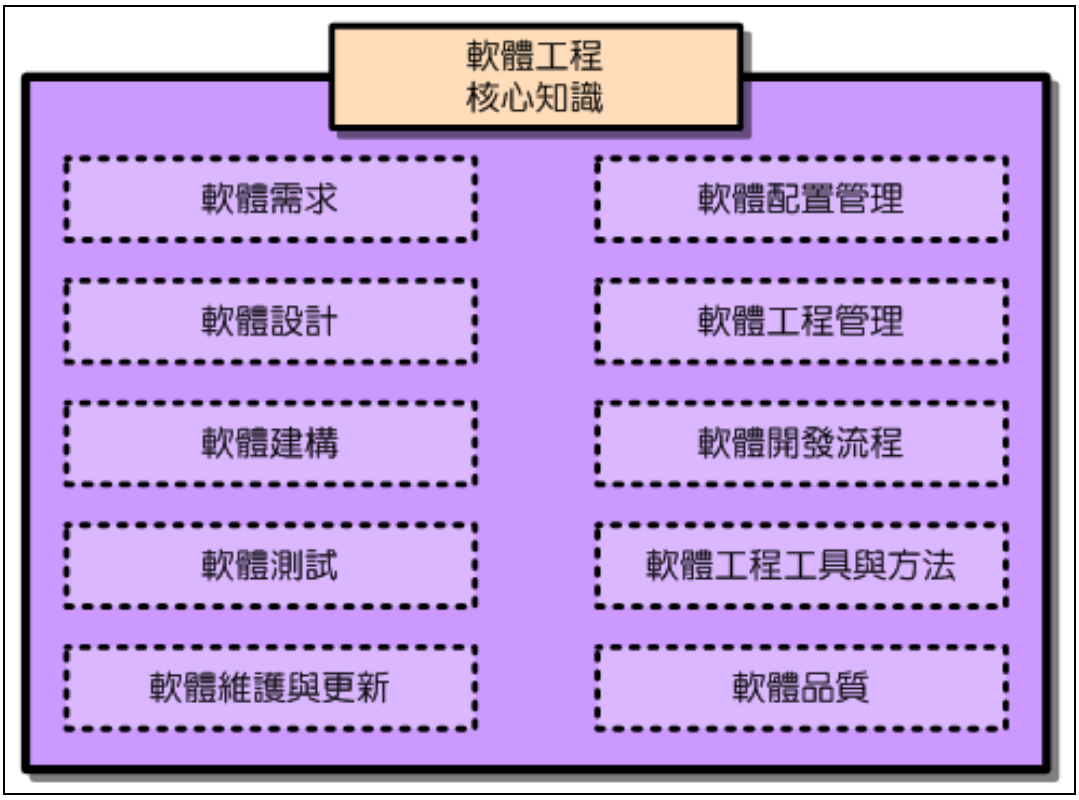

圖 3 軟體工程的核心知識

2.2軟體需求與需求分析

2.2.1軟體需求介紹

所有的軟體系統皆從瞭解使用者需求開始，此需求可能是用戶的簡單描述，或是正式的需求規格。那何謂使用者需求呢?所謂的使用者需求是指使用者對軟體系統在功能性、操作行為、性能、設計約束等方面的期望與要求。系統分析師可透過問卷、訪察和開會討論等需求擷取方式，瞭解使用者想透過此系統解決哪些問題、新系統之目標與限制等資訊，並將結果具體地表達出來，以便讓使用者確認需求是否正確及提供系統分析之使用。

取得使用者需求是整個軟體系統開發過程中較為重要但也是較容易產生錯誤的地方。因為軟體需求具有模糊性、不確定性、易變性與主觀性等特性，因此很難將軟體需求清楚、正確地表達出來。且軟體需求在專案生命週期中，會轉變成系統設計、原始程式碼、測試文件等內容，因此軟體需求會影響專案開發中所有的產出物。因此能愈早掌握正確且完整的需求，是影響軟體系統是否開發成功的主要原因之一。

2.2.2需求分析方法研究

系統分析師透過問題及其軟體操作環境的理解與分析，爲相關的資訊、功能及系統行爲建立模型，讓使用者需求更精緻、更完整，最終形成需求規格說明，這一系列的活動即構成軟體發展生命週期的需求分析階段。

許多需求分析的方法通常對系統模組化或系統設計有直接的幫助，但是需求分析的過程中，涉及許多人為的因素，所以無論應用結構化或是物件導向的需求工程方法，要將使用者需求、系統需求轉換為設計模型(design model)都是很難完全自動化的。

Kotonya[14]提出觀點導向的需求定義(Viewpoint-Oriented Requirement Definition)方法，是屬於需求誘出與需求分析中服務導向(service-oriented)的架構，以系統中的不同觀點來描述需求，並且定義成

屬性、服務或是事件。情境(scenarios)的方式也常被用來描述最終系統的行為需求(behavior requirement)，其中，使用案例圖是最常用的一種，因為它是一種物件導向模組化方法中基本的統一模式語言表示法。使用案例的目的是要畫出使用者的目的而不是系統功能，描述系統外界的觀點，在這種情境中一個人可能會是很多種操作者(actor)、而一個角色也可能是很多人所組成的。統一模式語言的使用案例圖，可用於任何需要了解系統需求的地方，同時也可以協助系統的測試。

傳統的軟體需求規格書是由自然語言撰寫而成，使用案例圖卻要用某個選定的觀點，將軟體的功能、需求明確地表示出。Leffingwell[22]於 2000 年提出較新興的需求表示法，取傳統的軟體需求規格書和使用案例的優點，結合成為 SRS Package 方式，以使用案例表示所有的需求，讓人一目了然，同時以軟體需求規格書輔助說明非功能性需求的部份。

一個導致計畫失敗的原因，主要的因素之一，就是缺乏良善的需求文件[16][29]。因此如何製造良善的工具，幫助開法者製作完善的軟體需求規格，便顯的相當重要。然而，在已存在的軟體需求規格文件中，普遍存在同樣的問題[33]，那就是使用者並無法分辨一份軟體需求文件之間的好壞差別。在本研究中，我們也會針對這個問題加以改進。

2.2.3需求分析的幾個重要階段

Hooper 和 Hsia 認為需求分析主要包含三個活動[11]：

- 需求判斷：

  強調如何判斷正確的需求以及需求的正確性。

- 需求分析：

  分析需求是否有不一致、不完整或互相矛盾等問題產生。

- 需求溝通：

  以最佳的方式組織及描述需求，使需求簡短易懂，且經由相互溝通的方式達到需求確認的目的。

Grosz 認為需求分析可分為兩大步驟[25]：

- 需求擷取：

  針對系統範圍內之各項事物語相關現象加以了解、判斷，並設計成描述性項目。

- 需求轉換：

  將描述性項目以系統模式語言轉換成概念性項目。

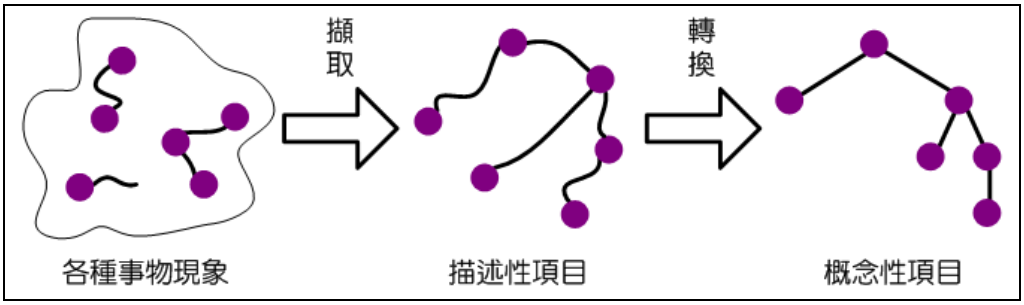

圖 4 需求分析兩大重要步驟

總而言之，需求分析是軟體系統開發的首要階段，此階段的主要工作包含有瞭解現有問題、系統目標、系統限制、使用者操作等。系統分析師與使用者及其管理者需具有良好的互動性，以提升需求分析之效率。良好的需求分析不僅可定義出軟體系統之目標與需求，也可對於軟體系統之功能、介面、品質與限制條件等加以分析與描述。

2.3物件導向與物件導向需求工程

2.3.1物件導向

系統設計方法主要可分為兩大類：一類為傳統的結構化方法，另一類則是後來崛起的物件導向方法。結構化的方式，起源於 1960 年代末期，主要強調如何應用一些概念、策略與工具，來提升系統分析與設計、程式設計與測試之效率與效能等。此方法也重視系統的可維護性，常用系統環境圖(context diagram)、流程圖(data flow diagram)、個體關係圖(entity-relation model)等表示系統概念，並以資料字典(data dictionary)說明所使用資料的定義。

與傳統結構化之軟體開發技術所不同的是物件導向方法，它是以類似於人類思考的模式來描述以及分析系統，可用來開發元件化軟體，也易於修改和不計次數之重複使用，在現今這種強調快速反應的時代，物件導向的技術被廣泛的應用於軟體發展的各個階段。物件導向技術目前

以趨於成熟，各種開發工具幾乎都支援，許多新興的程式語言也需要物件的觀念。

在現今的軟體開發觀點中，有許多的系統分析與設計活動都是透過物件導向的概念來進行。而物件導向應具備下列的特定：

- 物件(Object)與訊息(Message)。
- 繼承(Inheritance) 。
- 多型(Polymorphism)。
- 封裝(Encapsulation) 。
- 動態鏈結(Dynamic Binding) 。

利用物件導向概念所建置的軟體系統，它們主要的構成元素就是物件和類別。在 Coad 與 Yourdon[2]的研究中將物件定義為「物件-問題領域中某些事物的抽象化，可以反應保留系統資訊的能力、系統互動的能力以及兩者兼顧。」類別，則是針對某一組共通的物件所做的一種描述。每個物件應具備有下列三種特性：

- 識別名稱(Identity)：

  此名稱用來和其他物件作區別。

- 狀態(State)：

  描述物件特性的屬性資料。

- 行為(Behavior)：

  提供給外界使用的服務或操作等三部份。

而在物件導向的軟體開發流程中，主要可以分為下列三個階段[15]：

- 物件導向分析(Object-Oriented Analysis, OOA)：

  OOA 主要是在需求階段，從需求文件的字彙裡，透過物件以及類別來對問題領域(Problem Domain)進行塑模(model)。

- 物件導向設計(Object-Oriented Design, OOD)：

  OOD 的目的是為了讓開發者能夠了解系統的架構。在 OOD 的過程中，會導致物件導向分解(Object-Oriented Decomposition)。而 OOD 主要是使用邏輯(Logical)以及實體(Physical)的標記法來表示系統的靜態(Static)及動態(Dynamic)的觀點，並且會將系統之間的物件以及彼此之間的關係描述出來。

- 物件導向程式設計(Object-Oriented Programming, OOP)：

  OOP 是指在軟體發展過程中，將 OOA、OOD 的思想進行表達和實現。

2.3.2物件導向需求工程

將物件導向的觀念結合了資料與處理流程，使軟體系統架構的設計更容易模組化。在物件的定義與分類上，利用物件導向重複使用、繼承

與多型的概念，許多擁有共同基本特性的需求，則不需重新定義，且整體架構的維護性也提高許多。

目前已有許多研究都曾提過將物件導向技術結合需求文件，並且也提出一些工具，驗證其理論的實用性。Lubars[22]等學者，研究出如何從一份使用自然語言(Natural language)的軟體需求規格書中，發展出初始的物件導向需求規格文件(Object-Oriented Requirements)以及初始的物件導向分析模組(Object-Oriented Analysis Model)。他們的方法是將軟體需求規格文件連結到模組裡的物件，但是卻會發生段落重疊(Fragment Overlapped)的現象，這是由於他們的方法是會出現所產生的物件導向需求規格文件會出現語意不明確(ambiguities)、遺漏(omissions)以及重覆的描述(redundant descriptions)。在本研究中，使用者可以根據專家所提出的專家樣板，呈現使用者所需的物件導向軟體需求規格文件，由於已先將詞彙、使用者需求定義、系統需求架構等預先定義並且物件化，因此較為完整，可以降低上述的缺點發生。

另外，也有相關研究指出[26]，大部分的軟體專案之所以會失敗，很大的原因是由於軟體需求規格書的不完整(Incomplete)、語意模糊(Ambiguous)、以及不一致性(Inconsistent)所造成的。在 A.M. Salem 的研究中[26] ，為了提升軟體需求規格書的正確性(Accuracy)、一致性(Consistency)以及完整性(Completeness)，使用一個有規律、逐步逼近並且

有一個明確終點的解決方法來解決這個問題。但在 A.M Salem 的研究裡所提出的方法，在自動化部分仍須不足。因此本研究除了提升軟體需求規格書的正確性(Accuracy)、一致性(Consistency)以及完整性(Completeness)，也會提出一套工具，以加強自動化的部分，讓軟體需求文件能更有效率的產生。

而在 X. Zhu 的研究[35]中指出，如何解決需求文件的不一致(Inconsistency)是軟體系統開發的關鍵因素。因此，本研究提出了以相似詞的方式來改善需求不一致的情況。透過此方法，可以從需求文件中偵測出需求不一致的部分，並且利用物件導向的特性，來確保需求文件內容的一致性，以及最後會製作一套工具，強化自動化的功能。

H. Kaindl[14]在研究裡指出，大部分現實世界裡的專案，大部分都缺乏 OOA 的方法，尤其是在早期的需求文件定義(Definition)以及擷取(Elicitation)部分。因此他結合物件導向技術(Object-Oriented technology)與超文字(Hypertext)，發展出一套實用方法，並提供一套工具(RETH)供使用者使用，改善了需求文件定義的架構以及完整性。但此研究卻缺乏在處理過程中對使用者的引導，也就是說，使用者在定義需求規格文件與 OOA 的期間，常會發生不知如何繼續處理。因此，本研究也會針對此問題，進行研究、改善。

2.4需求管理工具

在軟體發展的過程中，適當的使用工具可以幫助各階段的開發、需求的管理、及維護，可以簡化許多繁雜的工作、並減少重大的錯誤發生。但是工具使用不當，則可能造成反效果。軟體開發、系統分析的工具很多，但針對需求管理專用的商用軟體仍不常見，在此介紹的幾種較著名的幾種商用需求管理工具：CaliberRM、DOORS、RTM Workshop 以及 Rational RequisitePro，目前這些工具仍持續開發更新中，其需求的記錄方式、表示法都有可能再改變。以上所介紹的幾套工具都可以定義各種需求，如：業務需求、使用案例、功能需求、硬體需求、非功能性需求與測試。這些工具某種程度上也都可以和 Microsoft Word 整合使用，典型的方式是在 Word 上添加工具列。這些工具可從文字檔匯入成需求，也可以定義關鍵字以支援階層式的數位條碼，通常是一個字首縮寫，例如 UR 代表 User Requirement 其後再加註一個唯一的整數。也有工具支援類似視窗系統的樹狀結構，DOORS 就是一例，能看出層次結構的軟體需求說明書。

需求管理工具大多都可以把處理過的需求以各種方式、格式匯出，如：軟體需求規格書、文字形態需求、使用者規格和報表等方式。在其他功能上，也可以指定需求改變時可能會互相影響的需求間鏈結，方便專案參與者追蹤查詢用。專案參與者而言，也可以針對不同的群組，設

定其權限；有些工具可以與非文字形態的圖形或圖表物件混合使用，如：Microsoft Excel 的工作表或其他圖片等。這些工具都可以與其他的應用軟體整合，但是沒有哪一個是在整個生命週期都整合得天衣無縫的。

以上所介紹之各項需求管理工具，經過幾年的發展與擴充，在功能上可說是越來越完整，但是卻很少以物件導向的概念思考，以物件為基本單元儲存各項相關的資訊，或以物件的方法及關係協助需求工程活動。因此，我們以不同的角度，提出以模型為基礎之物件導向式需求架構，呈現另一種需求的表示與儲存方式。

表 1 商用需求管理工具功能比較表

Feature	DOORS	RTM	CaliberRM	RequisitePro
Parses a source document to load requirements into database	✓	✓	✓	✓
Imports requirements from Word tables into database	✓	✓	✓	
Incorporates non-textual objects such as Excel worksheets and images into database	✓	✓	✓	
Synchronizes textual SRS with database contents		✓		✓
Defines different attributes for different types of requirements and set attribute values for individual requirements	✓	✓	✓	✓
Defines requirement baselines	✓	✓		
Notifies affected project participants by e-mail about requirement changes	✓	✓	✓	
Defines traceability relationships or links between individual requirements and between requirements and other system elements	✓	✓	✓	✓
Tailors usability options	✓	✓	✓	✓
Includes learning aids, such as a tutorial	✓	✓	✓	✓

and/or sample projects				
Integrates with other tools, such as testing, design, and project management	✓	✓	✓	✓
Defines users and groups and their access privileges	✓	✓	✓	✓
Enables threaded discussions on requirements	✓	✓	✓	✓
Includes web interface for database query, discussion, and perhaps updating requirement attributes	✓	✓	✓	✓
Includes built-in change proposal system	✓	✓		
Includes hardcopy user manuals	✓		✓	✓

第三章　物件導向需求模型

軟體需求分析在軟體開發生命週期中，是第一個開發流程也是較為重要的一部份，系統分析師透過問卷或訪談的方式取得初步的使用者需求，並反覆與客戶確認需求內容，最後產生最終確認的需求分析文件。現今的軟體需求大多數還是以自然語言的文字格式(Text Form)撰寫而成，較少使用其他方式來呈現。使用自然語言來描述軟體需求，容易有模稜兩可(Ambiguity)、不一致(Inconsistency)、不精確(Imprecision)及不完整(Incompleteness)的情形產生。如此一來易使軟體開發流程的時間增加與浪費，發生需求錯誤的機率也提高許多。

因此，本研究即是在需求階段提出一個物件導向式需求架構，並開發出一套以模型為基礎之物件導向需求編輯器，使用物件導向良好的特性協助開發軟體需求，並著重於系統發展生命週期前期的軟體需求管理。使用者定義為專案經理、系統分析師以及軟體組態工程師等軟體開發相關人員。目的在於協助需求工程活動的進行，且避免在需求階段時，客戶提出各種模糊不清的需求規格，解決開發成本與時間耗費的問題。

圖 5 為本研究之系統架構圖，系統分析師可透過物件導向開發技術擷取、發展軟體需求，使擷取出的軟體需求以物件的方式呈現，這些需求物件描述貼合實際生活環境，其描述諸如: 軟體系統之使用者、實際運

作的平台環境及硬體設備、整合套用的應用軟體等。此方法嚴加定義了這些需求物件之內在特質，外在溝通介面及物件與物件間的合作運行關係。除此之外，這些截取出的需求物件，繼承了物件導向之繼承之特性，使接續型(incremental)開發的軟體需求得到有效的描述及保存，最後可產生需求規格文件。

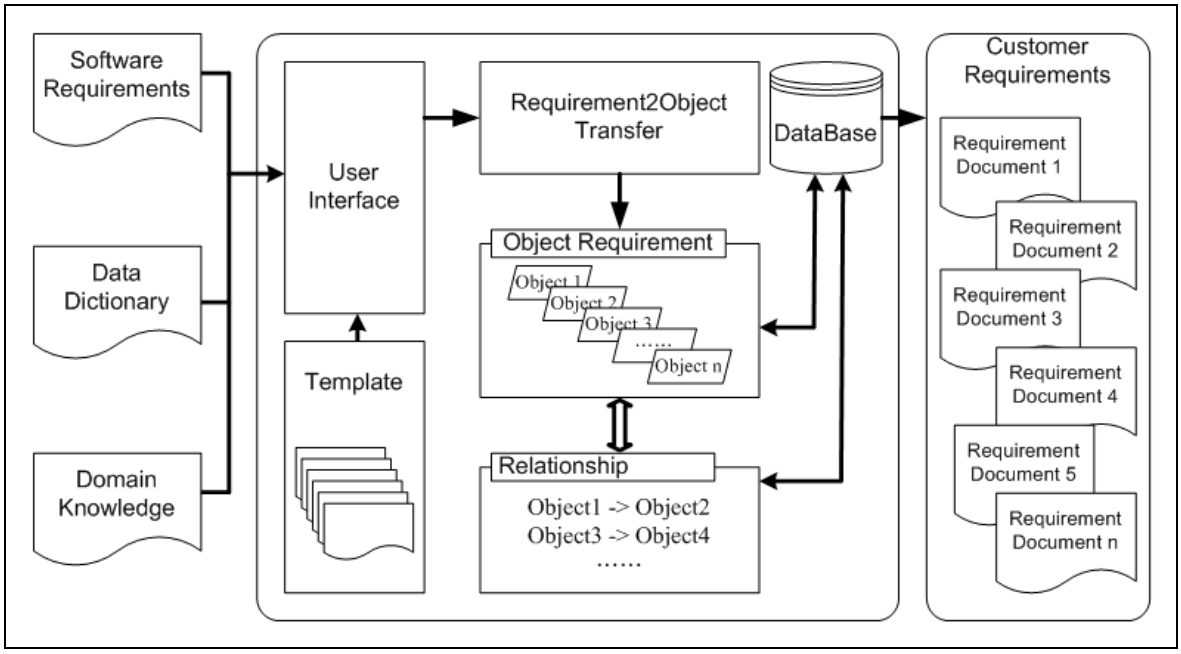

圖 5 系統架構圖

3.1軟體需求擷取流程

在整個軟體開發專案中，第一個要執行的步驟即是擷取顧客需求，如圖 6 所示，在顧客需求擷取階段，主要可分為三個步驟。首先，依據需求訪談所收集的顧客需求資料，針對客戶的需求來做需求的定義；接著，使用需求擷取模型擷取顧客需求以獲得初步的需求文件；之後，依據初步的需求文件進一步的轉換成完整的軟體需求規格。

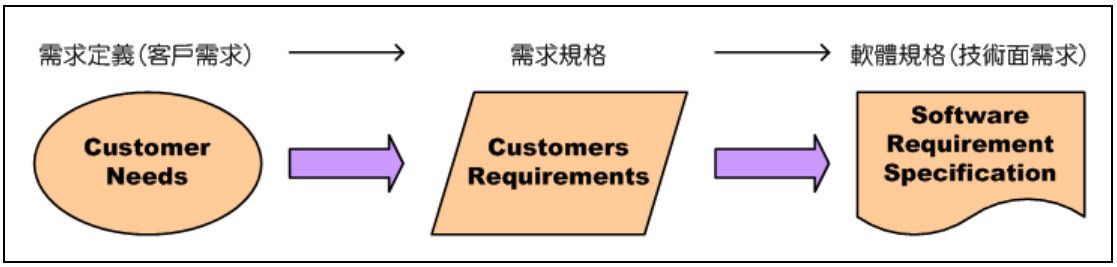

圖 6 軟體需求擷取流程

3.2需求問題樣板

在需求擷取階段，系統分析師蒐集所有顧客所提供的相關資料，這些由顧客所提供的資料稱為顧客需求(Customer needs)，為了簡化顧客需求擷取的步驟，也為了讓經驗不足的系統分析師了解需取得哪些需求，因此本研究在此階段定義了一問題樣板(Customer need elicitation template)，提供系統分析師有效的取得顧客需求，詳細的樣板內容如下：

1. 目的(Goal)：

   開發此系統軟體最主要的目的是什麼？想要藉由什麼方式呈現出來？

2. 系統使用者(User)：

   軟體系統開發完畢後，會有哪些人員操作或使用到此系統。

3. 功能性需求(Functional requirement)：

   主要描述系統該做什麼，也就是系統要提供給使用者的什麼樣的服務內容？如：需具有什麼樣的輸入、處理的流程與步驟、以什麼樣的類型輸出…等。

4. 非功能性需求(Non-functional requirement)：

與系統的執行效率、效能相關之需求，且是可以量度的(Measurable)的項目，如：系統的反應時間、系統可靠度、效能與維護性...等。

5. 系統架設環境(Environment)：

(1) 伺服器端(Server)與客戶端(Client)的電腦最低的硬體需求為何？

(2) 伺服器端(Server)與客戶端(Client)會利用何種作業系統來執行所開發的系統？

(3) 執行系統時，是否須有其他應用軟體支援？如：資料庫或其他應用軟體。

(4) 欲開發的系統是否運用到網路？如果會的話，網路的架構為何？網路速度是否有限制？

此需求問題樣板定義出一系列的問題，問題的設計由軟體系統目標擷取為出發點，透過定義軟體的最終目標，進而找出軟體主要需求(software requirement)，軟體需求包括功能性需求(functional requirement)及非功能性需求(non-functional requirement)。上述三要項(goal, function requirement and nonfunctional requirement)勾畫出此系統之主要藍圖。

除此之外，顧客需求還包含軟體運作環境之定義，其主要元素包括硬體需求、作業系統、應用軟體及網路環境等。在顧客所定義出來的特定環境上，預達到系統最終目標，有效執行軟體功能。上述呈列的這些特點在我們所定義出來的樣板將依輔助軟體工程師有效的擷取這些必要資訊。

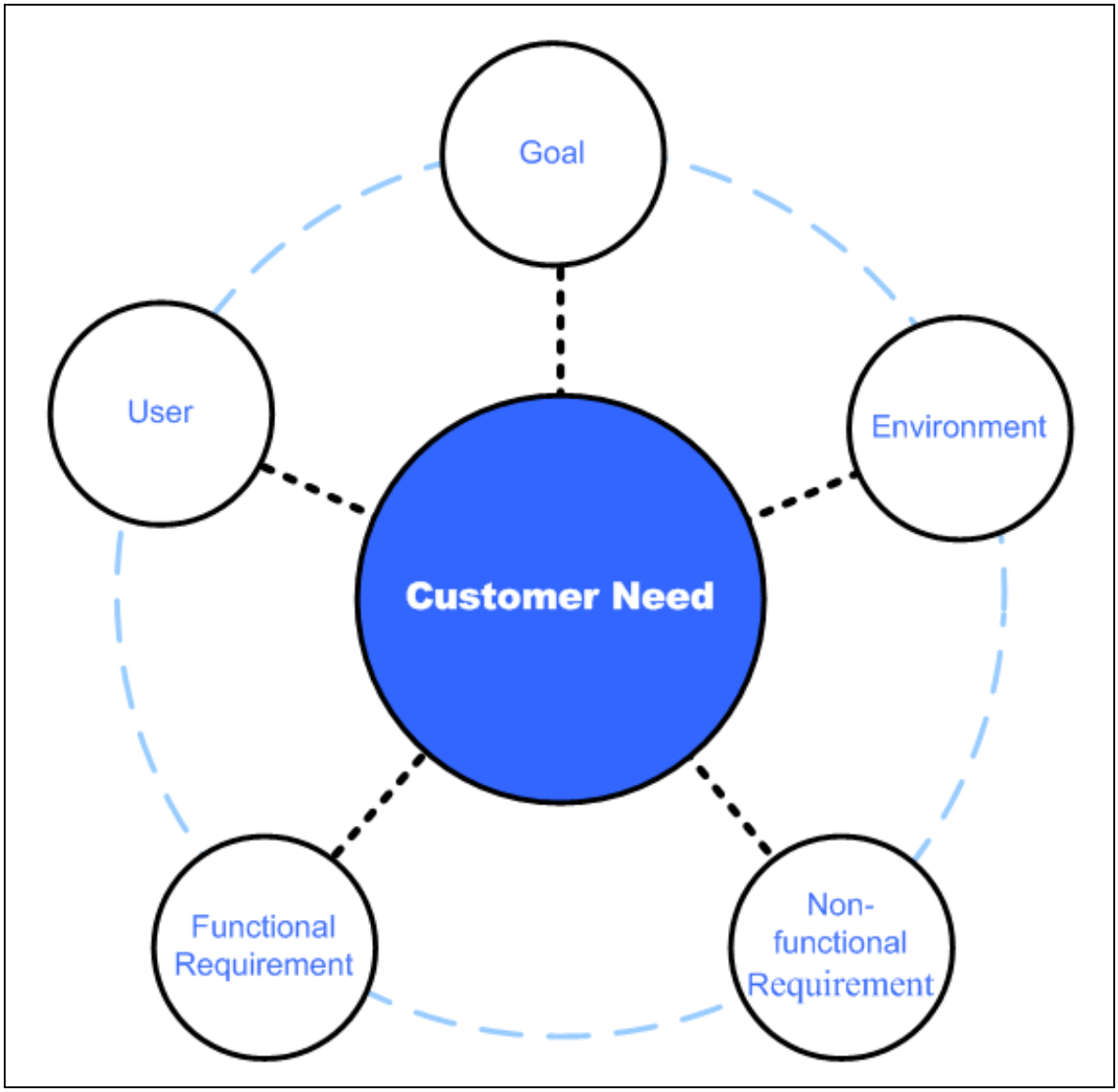

圖 7 需求問題樣板

下表 2 為一案例示範，此範例描述一圖書館管理系統之客戶需求。

表 2 需求訪談內容

項目	內容
系統目標	希望藉由圖書館管理系統達到全校師生與職員可自行辦理借書程序，以節省圖書館櫃檯人力支出。全校師生與職員也可透過網路操作來查詢書籍相關資訊與書籍是否還在館內。而系統管理者也可透過此系統來查詢所有書籍的狀態與了解書籍借閱情況。
系統使用者	1. 系統管理者 2. 訪客 3. 學生 4. 教師 5. 職員
功能性需求	1. 提供訪客查詢書籍資訊功能。 2. 提供系統使用者維護個人帳號資訊功能、登入、登出。 3. 提供系統管理者維護所有帳號資料、維護所有書籍與教材影片資料功能。 4. 提供教師預約書籍、外借書籍、歸還書籍、查詢借閱資訊、預約教材影片、外借教材影片、歸還教材影片功能。 5. 提供同學與職員預約書籍、外借書籍、歸還書籍、查詢借閱資訊功能。
非功能性需求	1. 在網路速度 2M/256k 環境下，使用者執行查詢書籍資訊功能需在 10 秒內顯示回應內容。 2. 圖書館管理系統需能容納 500 人同時使用本系統所有服務。 3. 系統當機率需小於 0.01%。 4. …
建置環境	1. 硬體： 伺服器主機/Ram 4G/硬碟 5T 客戶端主機/Ram 512M 2. 軟體： 伺服器主機/Database SQL Server 2000/Win2000 客戶端主機/WinXP/IE 6 3. 網路： 伺服器主機/100M 客戶端主機/512k

需求問題樣板除定義出一系列的問題來協助系統分析師擷取需求，也定義出相關的需求規格來引導系統分析師撰寫較完整的需求內容，詳

細的介紹如下：

表 3 需求規格說明

項目		內容	範例
非功能性需求	安全性需求	每隔時間單位系統會自動備份需備份資料項目名稱	每隔 1 小時系統會自動備份書籍借閱資訊
		管理者可透過備份回復機制回復需回復資料名稱	管理者可透過備份回復機制回復書籍借閱資訊
		是否需提供使用者權限控管機制(是/否)	☑是 ☐否
	系統效能需求	使用者登入時，需在時間單位內回應	使用者登入時，需在 2 秒內回應
		搜尋項目名稱時，需在時間單位內回應	搜尋書籍資訊時，需在 10 秒內回應
	可靠度需求	系統是否需提供可將轉移資料的方法(是/否)	☑是 ☐否
		系統當機率需小於次數/時間單位	系統當機率需小於 1/半年
	處理能力需求	系統需能容納人數同時使用本系統所有服務	系統需能容納 500 人同時使用本系統所有服務
	其他需求	是否需與現有系統的整合需求(是/否)	☑是 ☐否
介面需求		使用者是透過何種裝置來操作本系統	使用者是透過鍵盤、滑鼠來操作本系統
		是否需提供網頁介面(是/否)	☑是 ☐否
測試需求		測試功能，輸入內容後，系統需回應內容	測試搜尋書籍資訊功能，輸入書籍名稱後，系統需回應此書籍相關資訊

3.3需求情境樣板

在許多相關研究中有提到，針對需求的描述與擷取，必須先鎖定適當的活動者(Actor)，透過這些活動者擷取有效需求[28][1][14][24][29]。軟

體系統的活動者可分為使用軟體系統主要功能的人與支援軟體系統運作的活動者，而支援軟體系統運作的活動者又可分為軟體、硬體、網路。

其次，一個完整的需求描述，可以增加系統完成的可靠度。因此，完整的需求的描述在需求發展階段是重要的影響因素。從許多現行的文件與系統，我們可以發現，一個完整的需求描述需要包含需多面向。首先，我們必須了解此需求的由來(why)，其次我們必須知道此需求是由誰所提出(who)，他們在什麼情況下產生這樣的需求(when)，他們在什麼樣的環境底下會有這樣的需求(where)，他們真正的需求內容為何(what)，以及軟體設計師面對這樣的需求我們應該如何因應(how)。這些描述是構成軟體需求的重要元素，缺一不可。因此，本研究定義一需求情境樣板，系統分析師將透過需求問題樣板所擷取的初步需求，再參考此需求情境樣板進行較詳細的軟體需求分析。其詳細的情況如下所示：

- 情境名稱：

  簡單的情境描述，其規則為主詞+動詞(+受詞)的型式。

- 活動者：

  觸發此功能的人或支援欲開發軟體系統運作的其他軟體系統。

- 操作時間：

  使用者可操作此系統之時間，或事件觸發時間。

- 操作地點：

  使用者可在何處操作此系統。

- 功能性需求：

  此情境需使用到哪些功能性需求?

- 相關性需求：

  與此情境相關的所有需求，包含非功能性需求、測試需求、系統限制等。

- 輸入：

  需有輸入哪些資料?

- 系統回應內容：

  系統需有哪些回應資訊以及回應方式?

- 前置作業：

  執行此需求前，需有哪些前置作業需完成?

- 後置作業：

  此需求執行完畢後，會有哪些後置作業需完成?

表 4 以圖書館管理系統中的預約書籍功能為例子，描述此系統為達到提供學生預約書籍所做的所有互動行為，其中包括驗證學生的帳號密碼、要求輸入所預約書籍名稱、確認書籍是否在館內及顯示預約是否成功等。在劇情的描述語法，從 6W(Who、when、what、how、why、where)面向觀之，學生(who)在任何時間(when)都可以在家中透過網路(where)輸入帳號密碼(how)進行身分確認(why)來進行預約書籍(what)。透過此需求情境樣板，系統分析師即可撰寫出格式較正確且內容較完整的軟體需求。

表 4 學生預借書籍需求情境

項目	內容
情境名稱	學生預借書籍
活動者	學生
操作時間	任何時間
操作地點	客戶端電腦(必須連接上網際網路)
功能性需求	預借書籍
相關性需求	1. 系統回應時間應小於 2 秒 2. 需能容納 100 人同時使用 3. 系統當機率需小於 0.01% 4. …
輸入	書籍名稱
系統回應內容	預借成功、此書籍已外借、館內尚無此書
前置作業	1. 學生須先登入系統 2. 此書籍資料已建置完成
後置作業	1. 預借成功時，將此書籍狀態設定為已預借

3.4需求物件化定義

軟體需求往往來自於許多不同的提供者，各個需求提供者有著個人獨特的表述方式，多半以自然語言的方式表達，這樣的問題不容易在需求開發之初始階段就透過定義字彙(define glossary)[29]的方式進行解決，然而需求開發初始階段，必定是透過自然語言進行描述，一辭多義、一義多表的情況層出不窮[29]，不同的表述方式對不同的開發者而言，很可能有多重解讀，開發者的自我解讀及需求提供者的多變描述增加了軟體開發困難度。

如表 2 所示，圖書館管理系統的使用者描述，在表 2 中運用了以下兩種字彙”學生”與”同學”，但是實際上，這兩種字彙最終要表達的是相同

的意義，諸如此類之案例經常出現在軟體需求的描述當中。

我們有鑑於此，本研究提供一方法：同義詞彙，並讓系統設計師定義出需求同義詞。透過此方法，系統可自動找出軟體需求中所有擁有相同意義的辭彙，並且同步更新為統一的辭彙，如此可有效的排除一義多表所造成的困擾，及一辭多義所造成的錯誤。

除了同義詞外，在本研究中，我們選擇較適用於物件導向架構的方式，把描述在需求文件裡的資訊建構成模組化物件，這些物件描述了真實環境中有形或無形的實體(entity)，一個物件表示一種概念(concept)；再者，物件與物件間具有拼裝組合能力，透過物件的多重拼裝擴充物件能力。物件中的方法可以自動的完成訊息的傳遞、傳值等工作，而毋須由額外的手動方式來操作。而物件與物件之間的關係，更省去關連式資料庫表單透過合併(join)方式檢視不同資料表的麻煩。

以物件方式儲存需求，每一個需求就是一個物件，這些物件都具有一些特性，如：需求內容的描述、由哪些人所提出、被誰修改過、版本資訊...等都需要被記載與描述。我們將這些用來描述需求特性的資訊項目，定義為需求物件的屬性。而通常一個使用者需求很有可能因為其內容的不同，被切割成幾個部分去實現。這種物件的作法具有管理上的優勢，且這樣的方式更貼近使用者需求的本質。若本研究所提出之特性不足系統分析師使用時，系統分析師也可自行新增特性，以符合不同軟體

專案開發時擷取與分析需求使用。

需求必須記載描述需求內容的基本資訊，又因為需求本身變動性大，也常需要被修改，而有修改資訊。另外，需求與需求之間也存在著不同的關係，這些關係也需要被說明，所以一個需求物件基本的屬性如表 5 所示。

表 5 需求物件屬性表

基本資訊	專案名稱
	物件名稱
	物件分類
	物件描述
	提出人
	提出日
修改資訊	修改者
	修改日
	修改原因
	修改內容
物件資訊	目前版本
	先前版本

定義需求物件不僅可查詢此物件位於整份需求的何處，若有需求變更的情況產生，系統也會自動更新所有的物件，不需再花費額外的人力與時間更新文件內容。物件不只限定為單一名詞，也可以是段落、圖示、檔案等。

需求物件定義完畢後，系統分析師可透過物件辭典直接引用物件。圖 8 顯示出同一文件中，不同的章節處使用了相同的物件示意圖，若其中一處之內容有變更，系統會自動變更相同物件之內容。需求物件也具有繼承的特性，如圖 9 所示，系統管理者、學生和訪客都繼承了使用者的屬性"操作系統的人"。

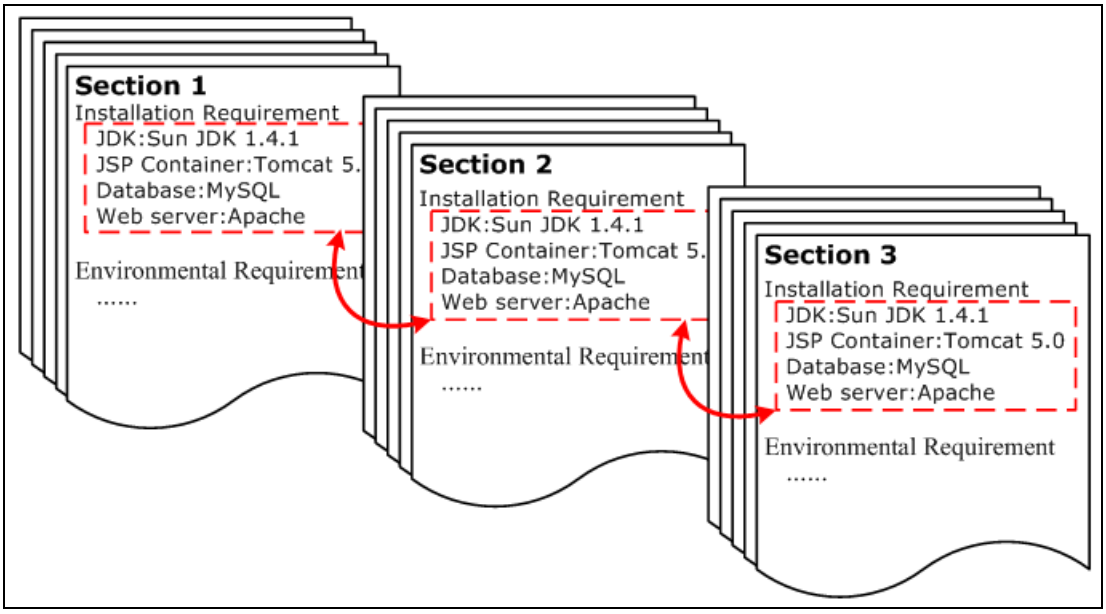

圖 8 引用需求物件

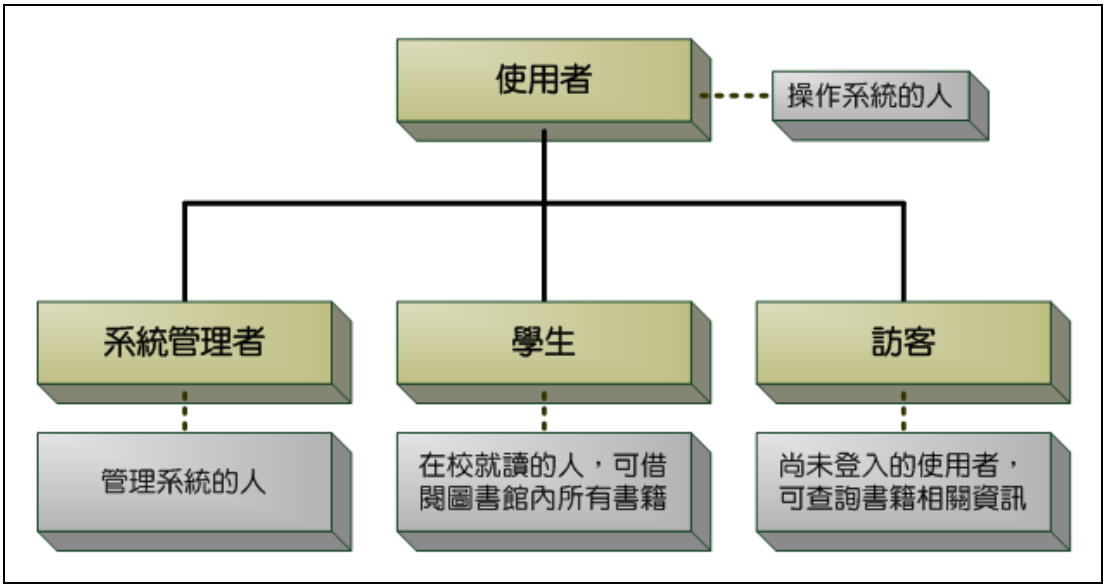

圖 9 需求物件繼承圖

需求物件除了可使用在同一專案中，也可在其他專案中重覆使用(reuse)此需求物件。如圖 10 所示，圖書館系統、線上測線系統、電子信箱系統與選課系統皆有”學生登入系統”此需求，因此可引用現有的需求物件。透過此方法可有效幫助分析同類型系統之必要軟體功能，換句話說，軟體功能也就是在這邊所指的軟體需求。同時，它也針對輔助性功能及非必要性功能提供評估方式。透過此方法，系統開發者可以參考現行系統他們所具有的基本功能，並且評估這些現行系統隱藏之非功能性需求(Non-functional Requirements)。這些參考依據可以幫助開發者，快速擷取系統主要需求，針對這些主要需求進行徹底了解。如此一來，不僅可以妥善利用現有經驗，同時可以減少需求擷取時間，更能夠增加需求可信度，最後促使提升軟體成功的可能性。

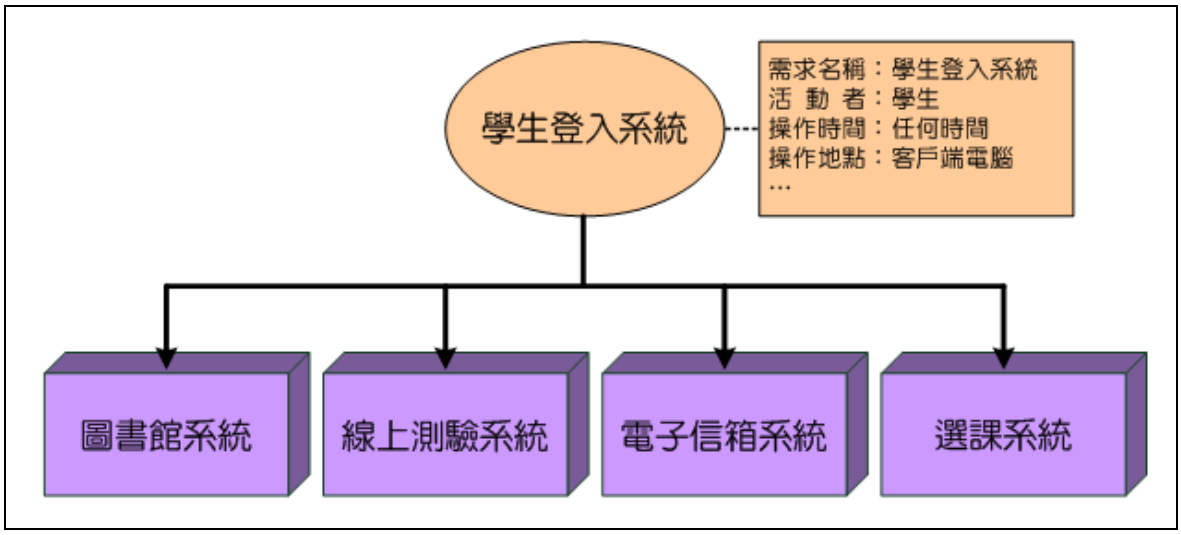

圖 10 不同專案系統引用相同需求物件

3.5需求關聯性

需求管理活動在系統發展生命週期中，大多都會遭遇反覆和不可預料的需求改變，需求儲存與需求間通常會相互鏈結。面對越來越大量的資料量、即時的、複雜的、甚至是密集系統(intensive system)，對於需求追蹤(requirement traceability)的維護是必須的。

需求追蹤主要是追蹤需求對其他需求、模組或其他相關追蹤項目間的關係，如設計與使用者文件間的關係。需求追蹤定義眾說紛紜，大致上可分以鏈結(link)和關係(relationship)兩種觀點區隔：欲透過鏈結追蹤，需要預先定義可能會相互影響的需求之間的鏈結，這種方式的追蹤，可以得知改變的源由、了解設計的理念；利用需求與需求間自然存在的關係追蹤，如需求間的組合關係，則能發覺需求的變動對後續開發階段或是相關的文件等影響為何。除了組合關係外，需求關係還分為前置條件關係與後置條件關係。若執行此需求前，需有某先需求先達成目標，此為前置條件關係；反之，若執行此需求後，會產生其他的需求，則為後置條件關係。

第四章 系統展示

本章節將以一簡易之圖書館管理系統為案例，說明以模型為基礎之物件導向需求編輯器的操作方式。

4.1 開發環境

本研究所實作以模型為基礎之物件導向需求編輯器，採用微軟公司的 Windows XP 作業系統，Sun 所推出的 JAVA 程式語言來撰寫系統，並使用 IBM 所推出的 Eclipse 整合開發環境作為系統程式開發平台，後端資料庫則使用 MySQL 來進行系統的實作。

Java 是由美國 Sun 公司所發展出的一種程式語言，Java 具有物件導向、豐富的程式庫或自動記憶體管理等多樣特性，還擁有比其它程式語言更好的支援性、語法更簡潔與跨平台的高移植性，因此本研究使用此程式語言來撰寫系統。

Eclipse 主要針對 JAVA 語言所推出的整合開發環境工具，且擁有可插件(Plug-in)機制，可外掛不同工具或程式語言作發展，本研究即再加上可讓 Eclipse 支援 GUI 的外掛程式(Jigloo)，Jigloo 可協助系統開發師在 Eclipse 上輕鬆的建立圖形化使用者介面應用程式，因此本研究選擇以 Java、Eclipse 與 Jigloo 作為前端的系統開發工具。

MySQL 為一套免費的程式，是由 MySQL AB 公司所開發的資料庫伺服器，可以連結 C、C++、Java、Perl、PHP 等程式語言，而且也可在許多平台上運作，如：Linux、Windows、Sun Solaris ... 等，且支援微軟的 ODBC 規格的資料庫整合。權限的使用也是 MySQL 特別的地方，對不同使用者設定權限，在資料庫中必須依權限的設定才能進入資料表，提高了安全性。

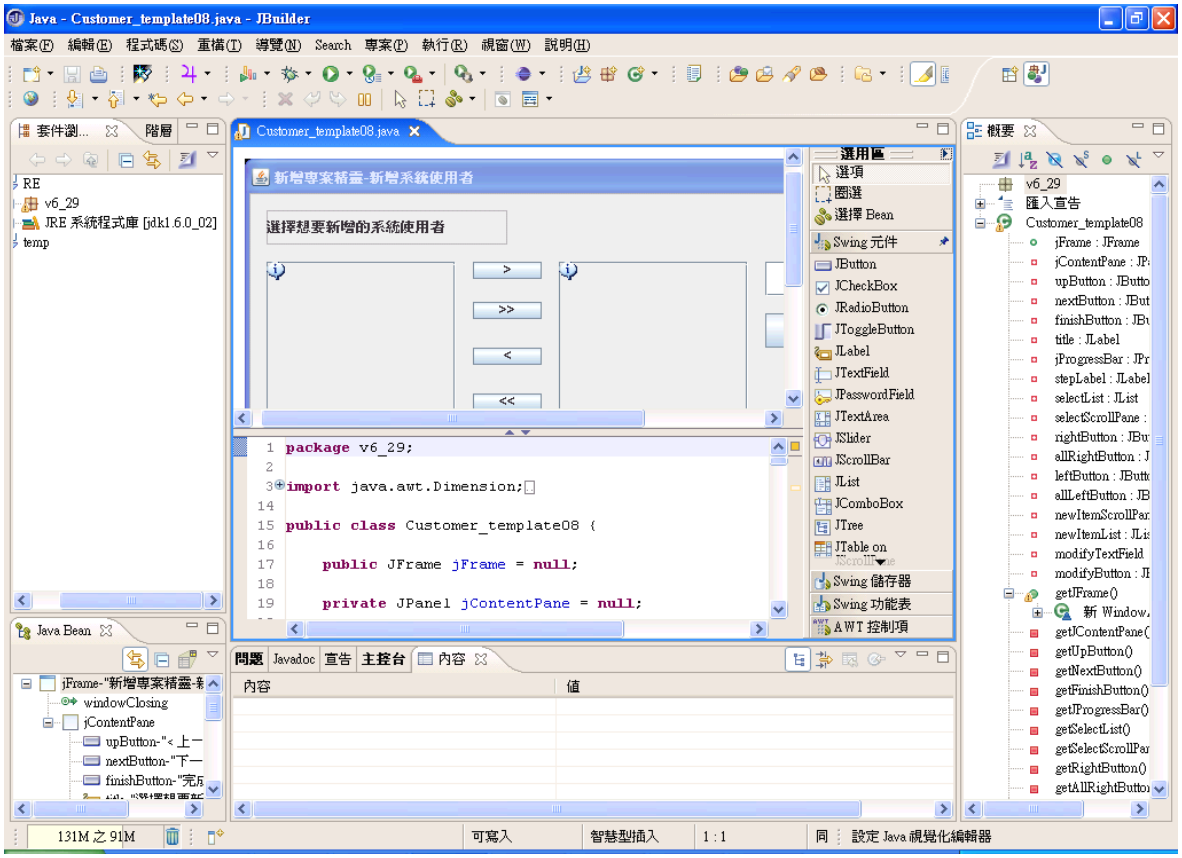

圖 11 Borland JBuilder 2007 系統畫面

4.2 系統登入與新增專案

本系統的起始畫面即為使用者登入程序畫面，使用者在輸入帳號、密碼之後，系統會查詢該使用者帳號與密碼是否有誤，是否為專案中的

成員，並且判斷使用者在專案中所扮演的角色。

圖 12 登入程序畫面

建立新軟體專案需求的第一步即將此專案命名，並可選擇專案範本來協助撰寫軟體需求。

圖 13 新專案命名

圖 14 選擇範本類型

4.3 系統主畫面圖

專案新增完畢後，系統畫面如圖 15 所示。本系統主要分為三個部份；A 部分為系統工具列，所有的操做工具皆顯示在此區域；B 部分為需求/物件列表區，在此區域可看見所有的需求以及物件列表；C 部分為需求/物件描述區，此部分可撰寫詳細的需求定義、需求狀態、需求關連性等資訊。

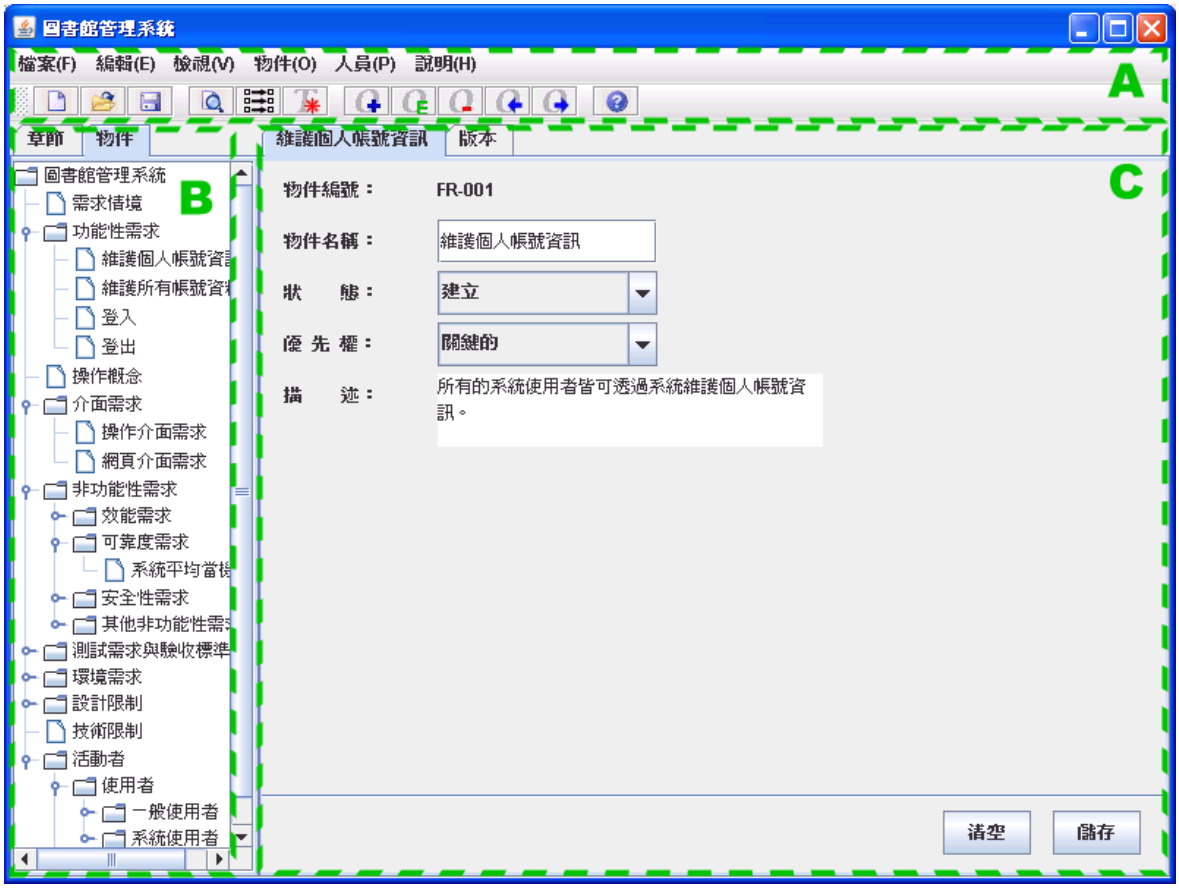

圖 15 系統主畫面圖

4.4 建立系統目標與相關活動者

擷取軟體需求的第一個階段即是建立系統目標，用以說明開發此軟體專案最主要的目的為何？與系統要如何呈現？

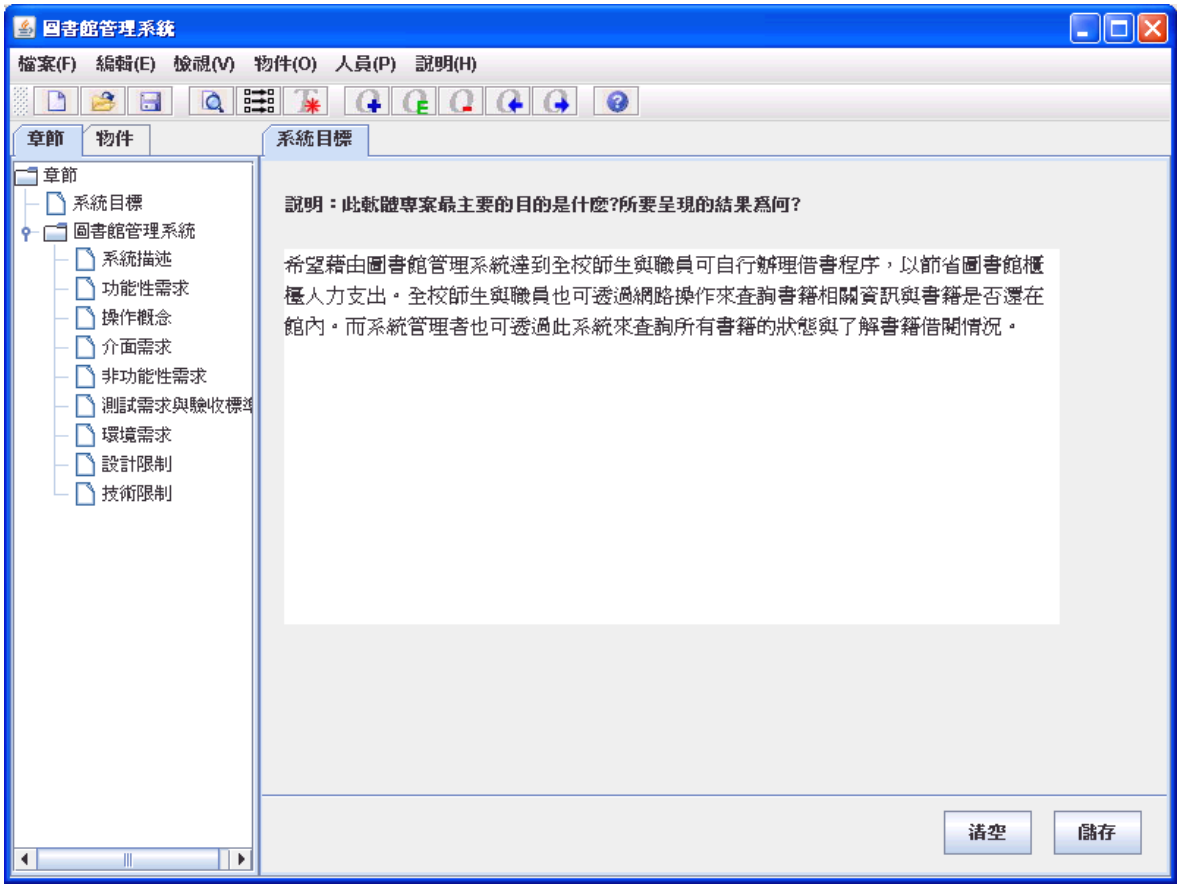

圖 16 編輯系統目標

系統目標建立完成後，第二個步驟就是建立與系統相關的活動者。活動者包含有真實的使用者與協助此系統運作的其他應用軟體。

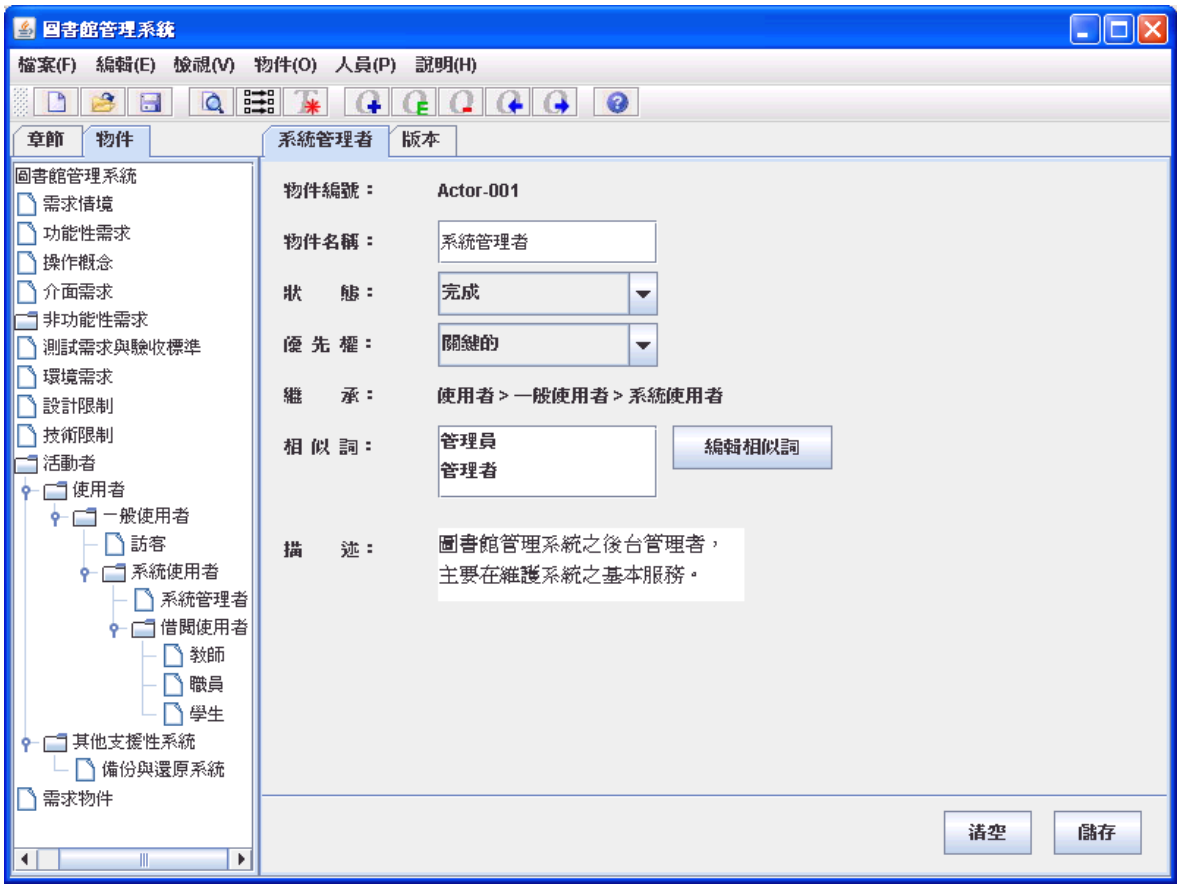

圖 17 編輯"系統管理者"

若系統分析師對於需求分析流程不太了解，或不曉得需撰寫哪些需求項目，此時可開啟引導精靈功能來協助系統分析師逐步完成需求分析流程，項目提示欄位會顯示需撰寫哪些內容並會顯示相關範例。

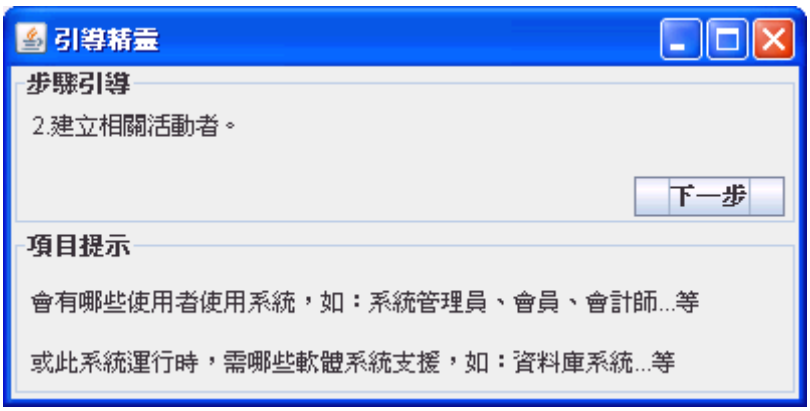

圖 18 引導精靈

4.5 使用需求樣板新增需求

圖書館管理系統屬於校園資訊系統的一部份，且在一開始時我們也選擇了校園資訊系統的專家範本，此時可以點選工具列上的樣板精靈  按鈕來新增一些需求。首先依據樣板精靈所提供的需求項目，選擇開發圖書館管理系統所需之需求，透過此樣板精靈可新增功能性需求、非功能性需求、介面需求、測試需求、環境需求、設計限制與操作限制，如圖 19 所示。圖 20 則是透過樣板精靈所產生的”維護個人帳號資訊”需求，並且也做了小幅度的修改。

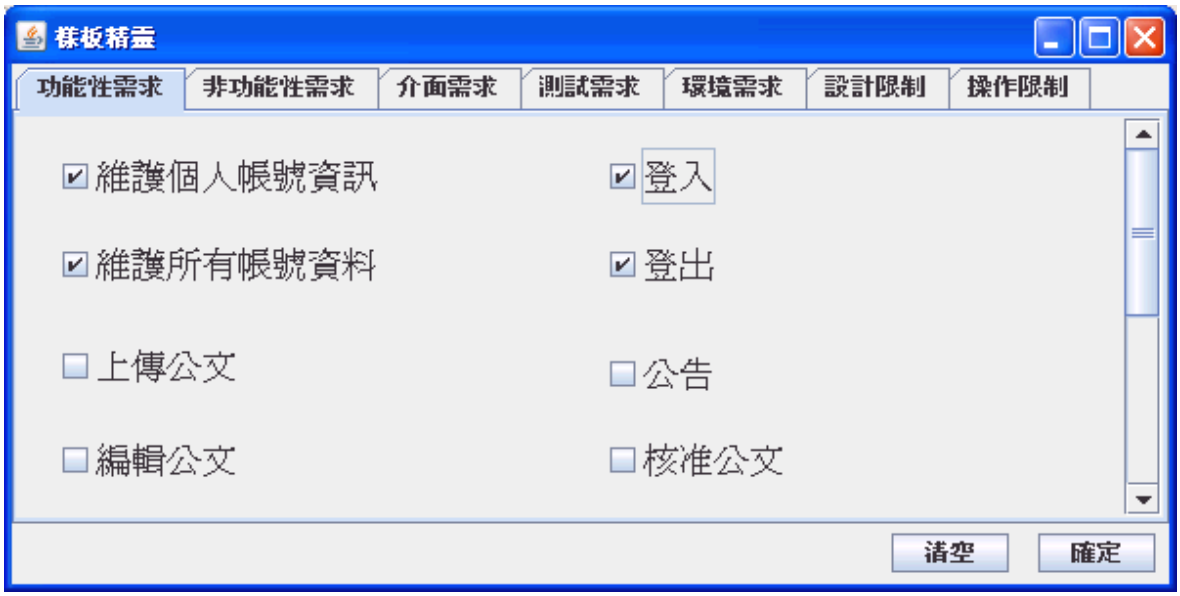

圖 19 使用樣版精靈新增需求

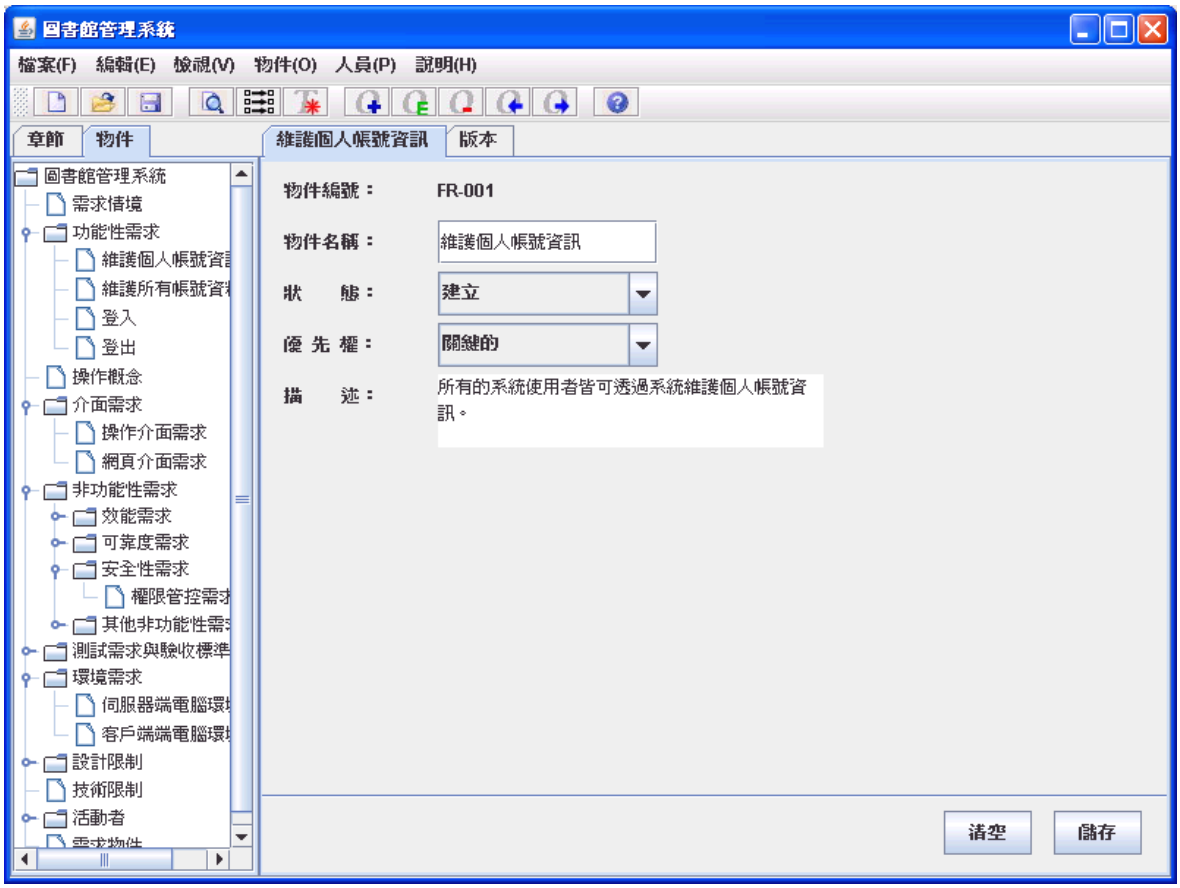

圖 20 修改完成的功能性需求

圖 21 非功能性需求

4.6 新增需求情境

基本的需求新增完畢後，系統分析師即可透過需求情境樣板來編輯需求情境。若系統分析師在編輯情境時，發現缺少某些需求內容，也可先返回新增此需求，再繼續撰寫需求情境。系統分析師也可在需求情境樣板中選擇與此情境相關連之其他需求，當各個需求建立好關連性後，未來只要修改某一需求，系統則會主動提醒是否需修改其他需求。

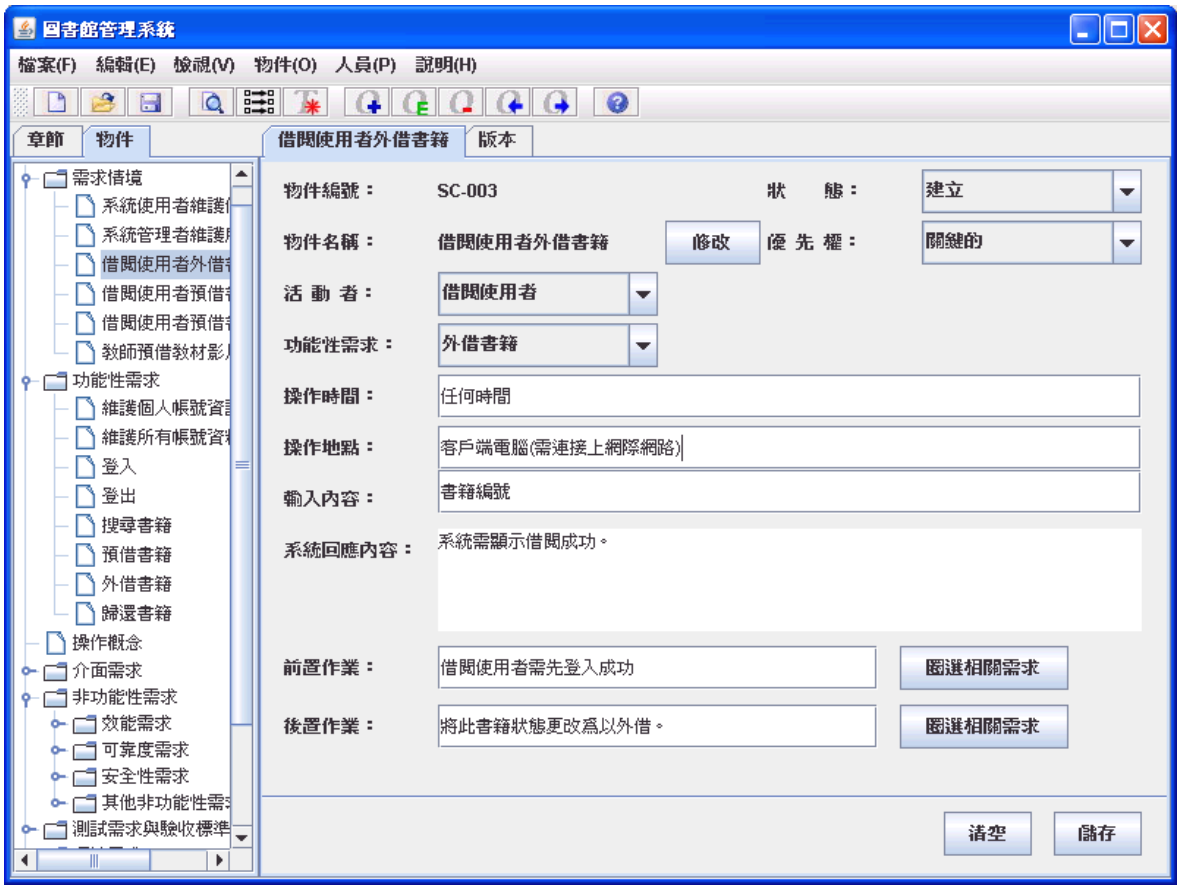

圖 22 編輯”借閱使用者外借書籍”需求情境

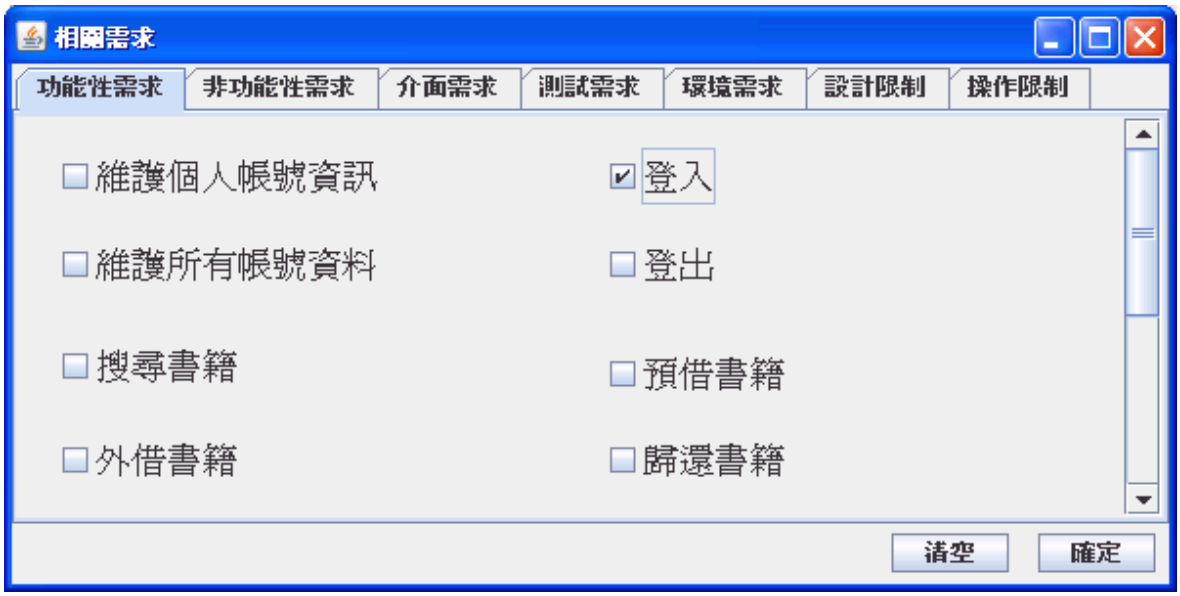

圖 23 選擇相關連的需求項目

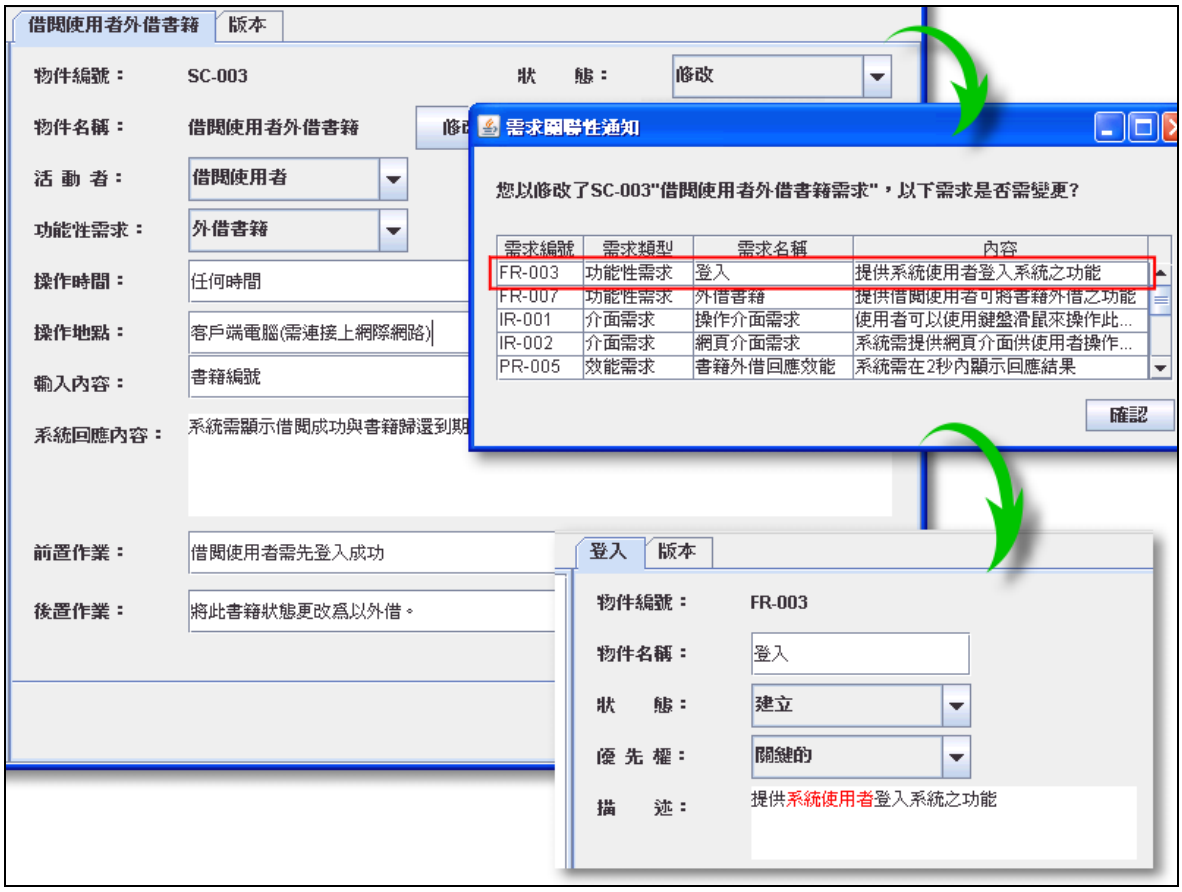

圖 24 需求關連性通知

4.7 建立物件化需求

系統分析師透過樣板精靈產生基本的軟體需求後，系統會自動將基本的軟體需求放置於系統預設的存放位置。接著系統分析師需手動選取特定的需求關鍵字將需求物件化，並定義其相關屬性。

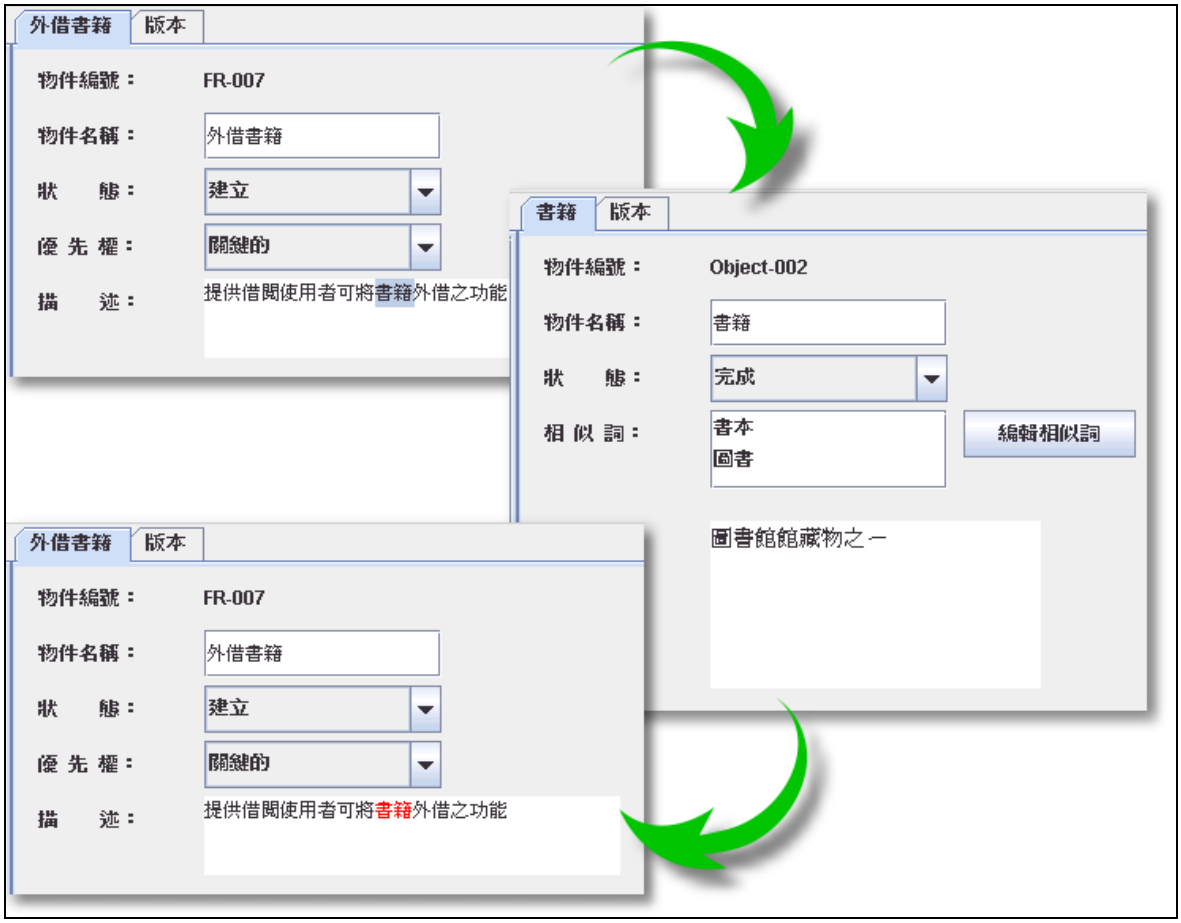

圖 25 新增”書籍”物件

物件新增完畢後會自動儲存在系統中，系統分析師除了可檢查或修改物件內容外，系統分析師也可按下”更新物件”按鈕，可以搜尋現有的需求內容中，是否有與此物件相似詞相同的文字敘述，若搜尋到相同之文字敘述，系統會建議是否更換成此物件名稱。如此可以達到需求描述較一致，不會產生相同的意思卻產生兩種描述方式。

本實例中，”書籍”這物件，擁有兩項別名”書本”與”圖書”。原始需求

文件內容如圖 26 所示，需求描述中是使用”圖書”這字眼來描述。更新物件內容後如圖 27 所示，系統使用” 書籍”來替換” 圖書”這字眼，並將其設為物件。

圖 26 預借書籍需求原始內容

圖 27 預借書籍需求物件更新完畢

需求物件具有物件導向中繼承的屬性，例如：有一功能性需求名稱為”搜尋”，所具有的屬性有向上搜尋、向下搜尋、區分大小寫與完全符合此四個屬性，而繼承”搜尋”的功能性需求有”查詢書籍資訊”、”查詢教材影片資訊”與”查詢系統使用者資訊”等功能性需求。上述所列出之四功

能性需求，除本身特定的屬性外，如”查詢書籍資訊”的特定屬性為書籍名稱、書籍編號、書籍作者，還具有”搜尋”物件所有擁有的向上搜尋、向下搜尋、區分大小寫與完全符合這四個屬性項目。繼承需求物件除可節省需求撰寫時間，且當父類別的物件有所更改時，子類別物件也會同時一起更改，不僅節省需求修改的人力，更可減少不小心遺漏而尚未修改的內容，增加需求的一致性。

4.8 匯出與匯入需求物件

需求物件除了可在同一專案中使用外，也可將制定好的需求物件匯出，以利未來專案匯入使用。

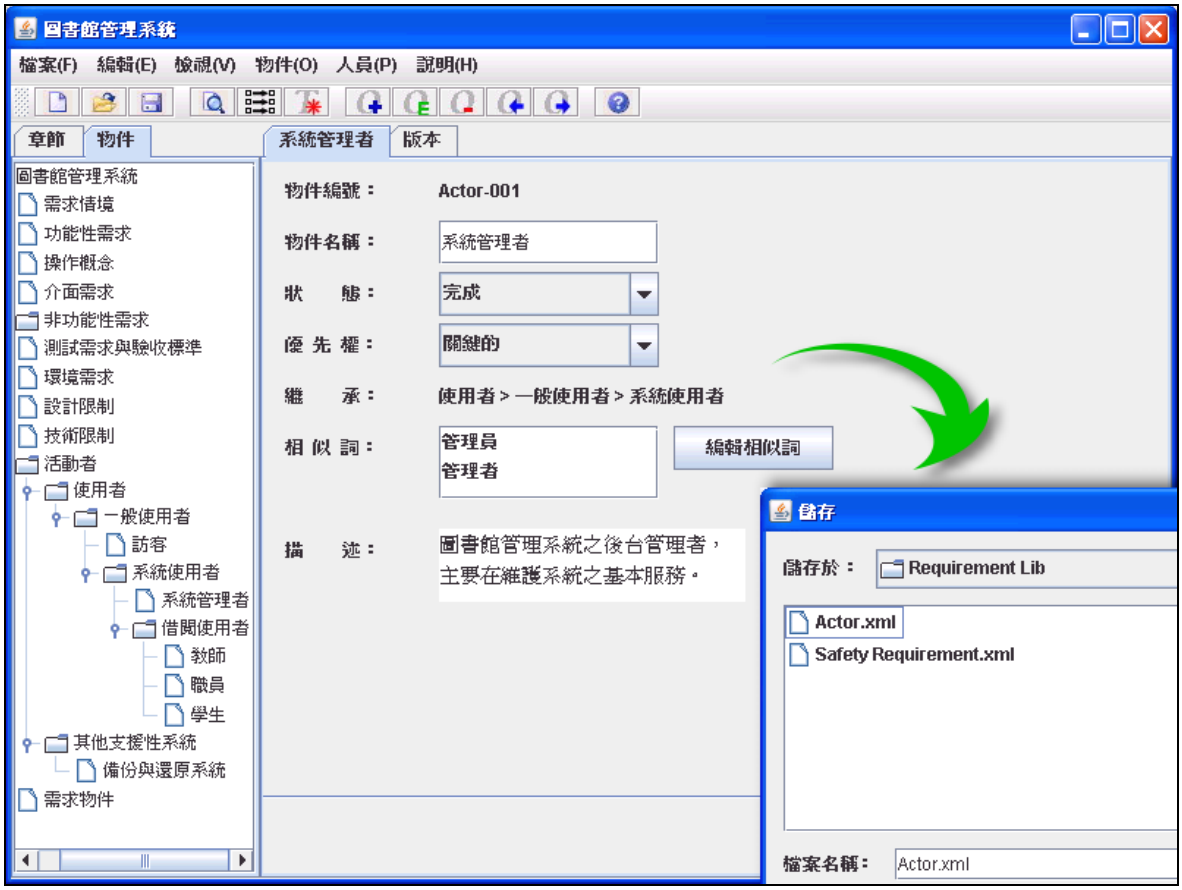

圖 28 匯出”活動者”物件

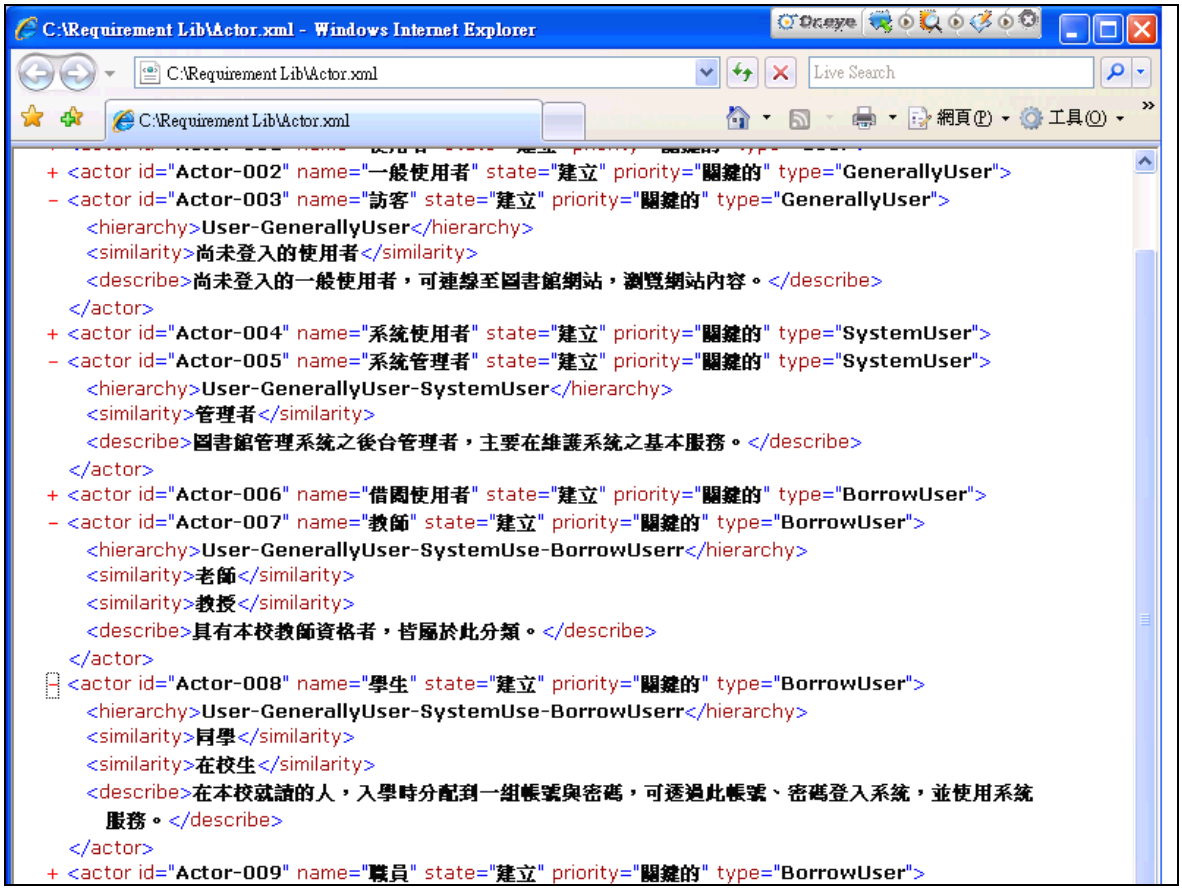

圖 29 ”活動者”物件檔案內容

本系統除可將需求物件匯出外，也具有需求物件匯入功能。系統分析師可選擇現有的需求物件檔案，並進行匯入的動作，如此一來可重複利用現有的需求物件，而系統分析師則不必再花費額外的時間撰寫相同的需求內容。

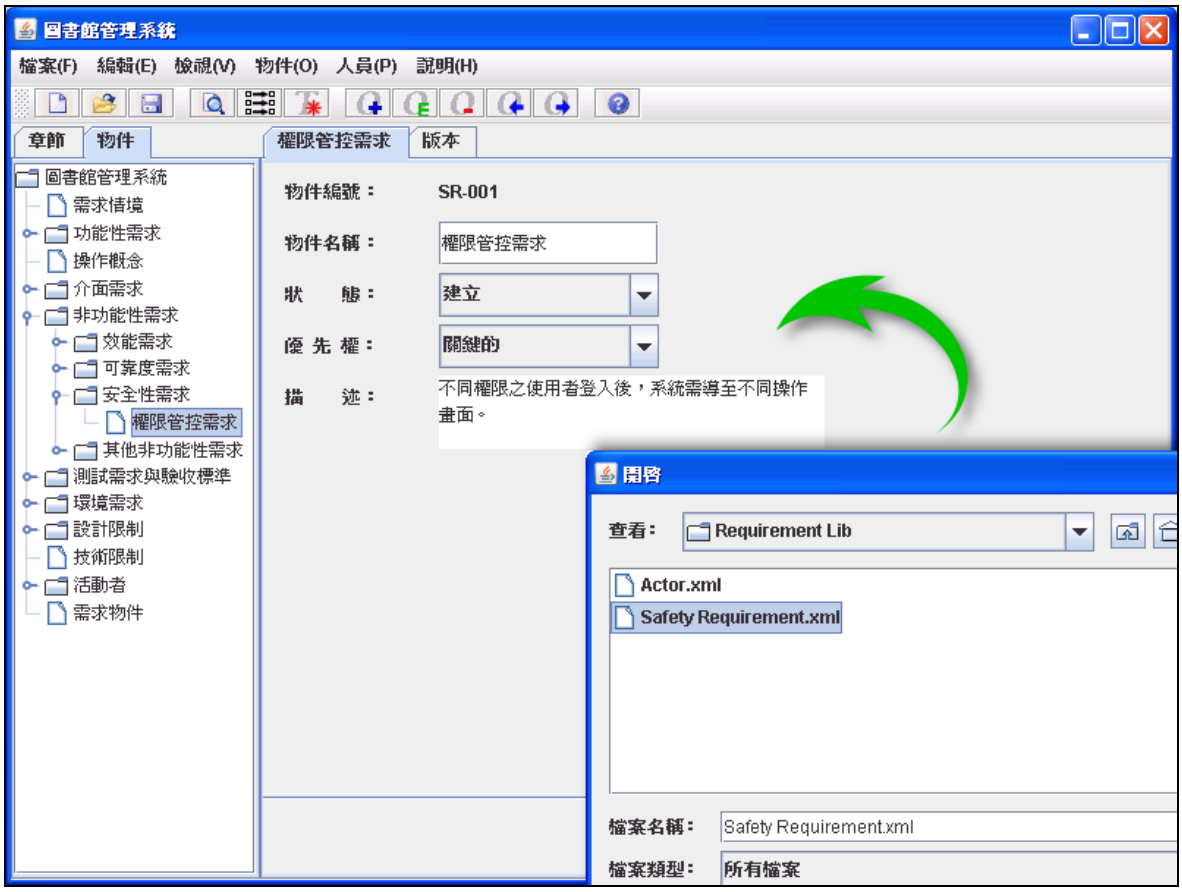

圖 30 匯入”權限管控需求”物件

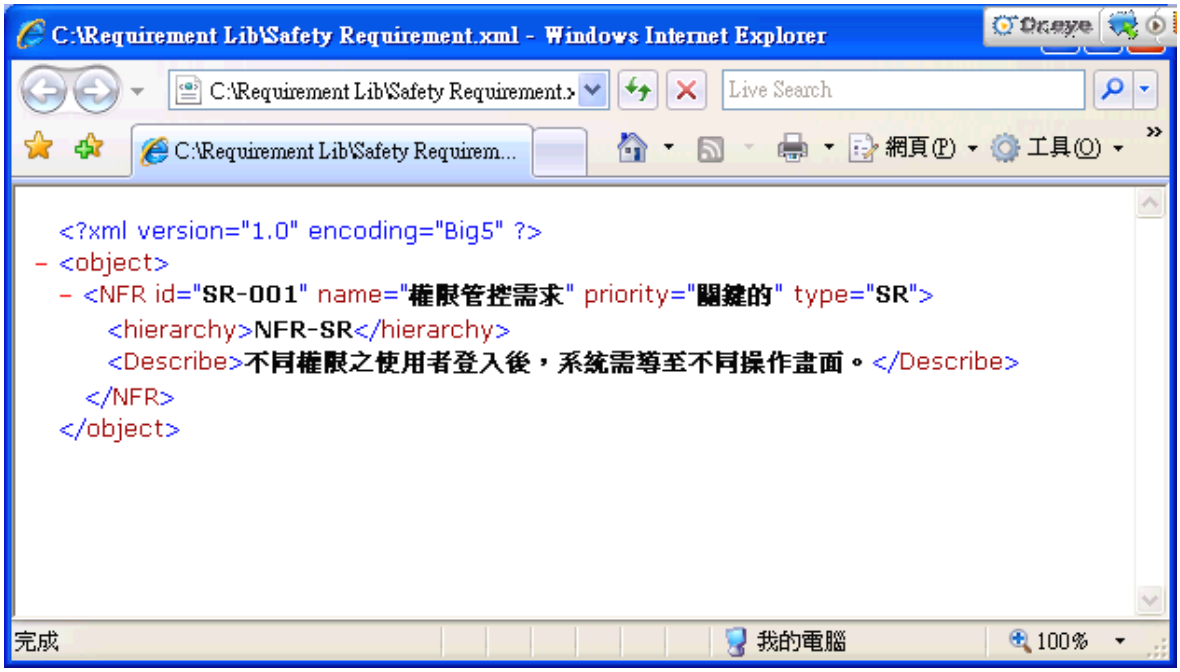

圖 31 “權限管控需求”物件檔案內容

4.9 搜尋需求物件

系統分析師可透過物件搜尋功能來搜尋所有的物件內容，如圖 32 所示，系統分析師填入欲搜尋的物件名稱”學生”，並選擇欲搜尋的需求狀態”完成”後，系統則會列出符合物件名稱為”學生” 且需求狀態為”完成”的所有項目。若想查看完整的需求內容，可點選搜尋完成的項目，系統即跳出完整的需求內容視窗，方便系統分析師查看。

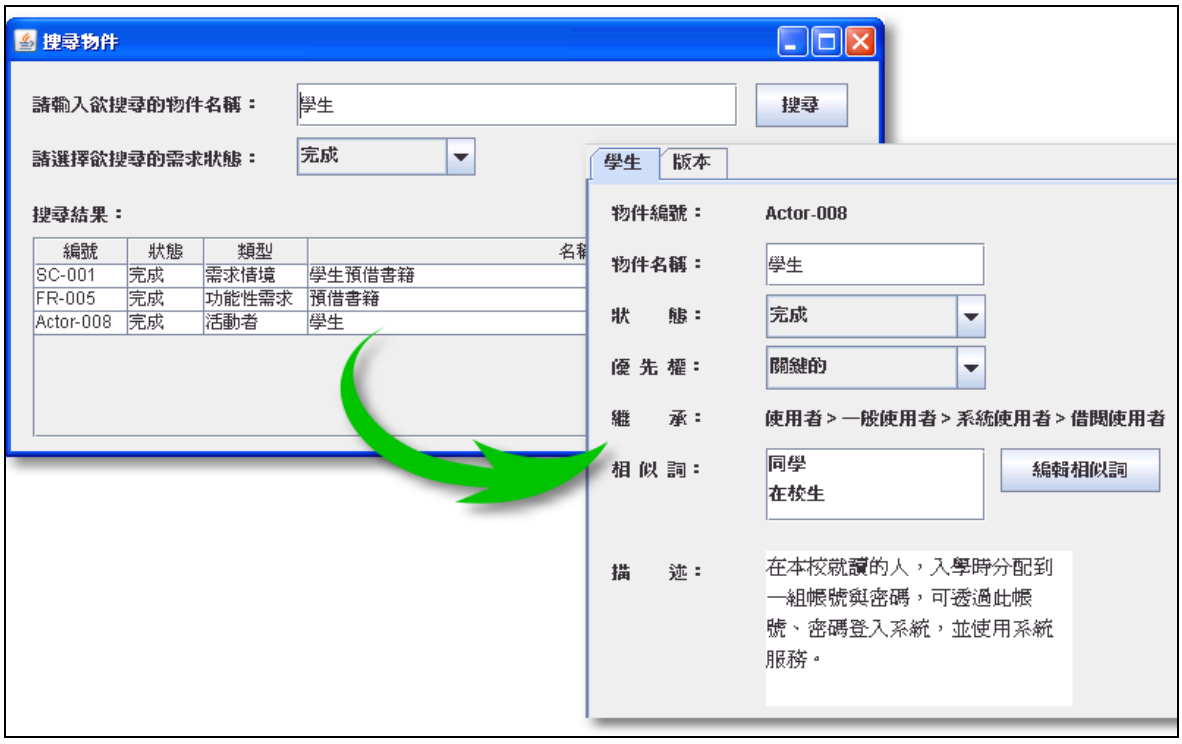

圖 32 搜尋”學生”物件

4.10關連性分析

本系統關連性主要分為三個部份：一般關係、前置條件關係與後置條件分析。只要與指定需求有相關連的關係，皆為一般關係；前置條件

關係則是需完成此前置條件需求後，才能夠完成所指定需求；當所指定的需求完成後，產生此後置條件需求的關係，則為後置條件關係。

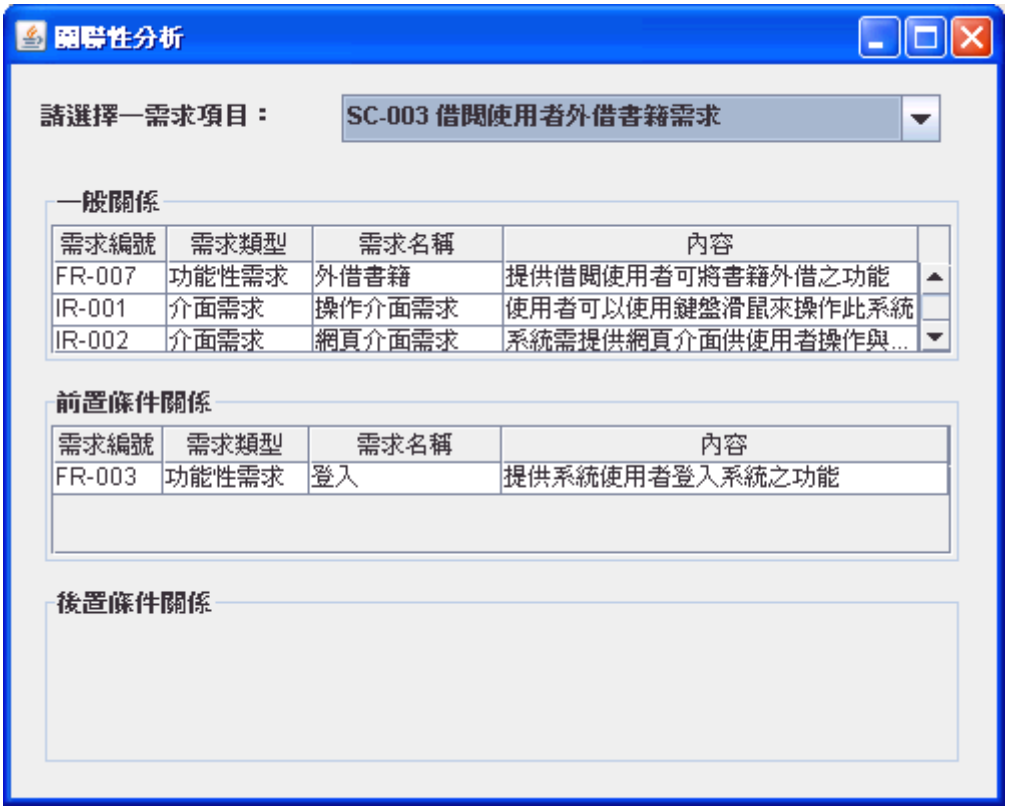

圖 33 關連性分析

4.11編輯使用案例圖

本系統除可以文字和物件來描述需求外，系統分析師還可透過使用案例圖來表達需求。一開始系統分析師可點選想產生一位活動者或多位活動者的使用案例圖，之後系統會自動搜尋其相關的功能性需求，並產生基本的使用案例圖，系統分析師可依據系統自動化產生之使用案例圖進行修改。若系統分析師在此部份有對活動者會功能性需求名稱有加以修改，其原本的需求名稱也會跟著同步修改。

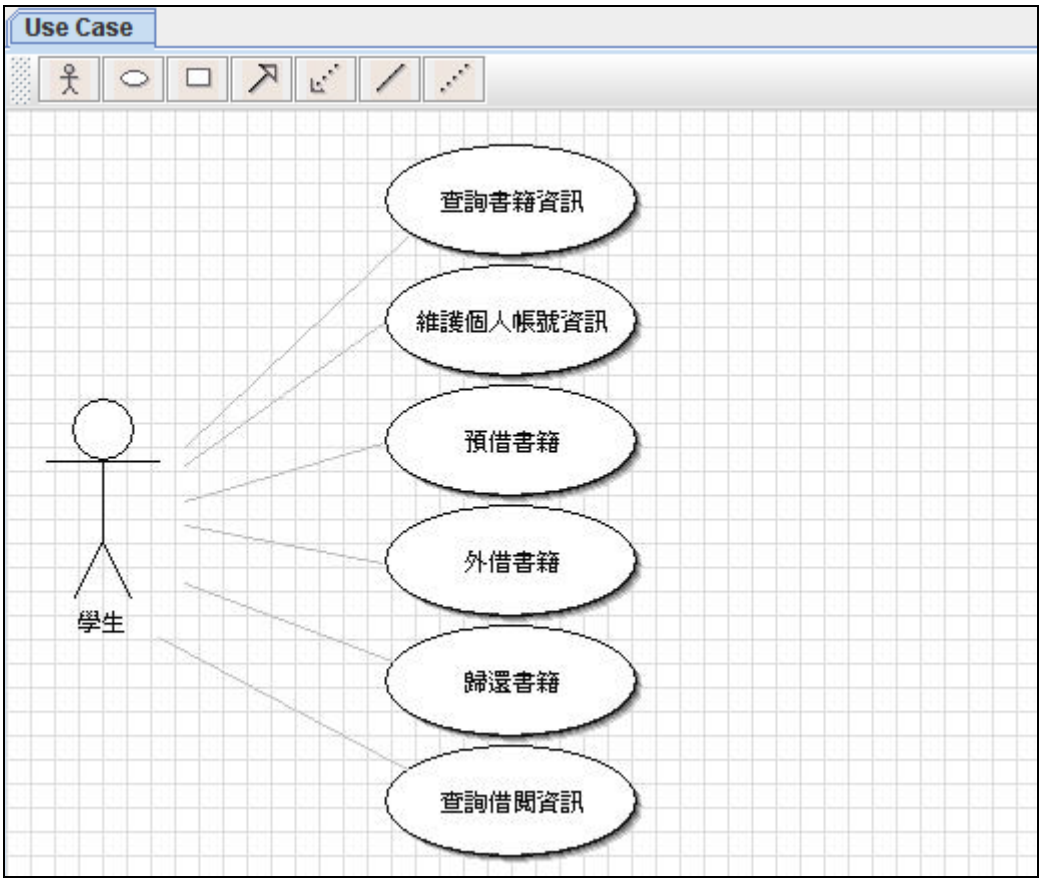

圖 34 編輯使用案例圖

4.12版本紀錄器

在本系統中，不管變更任何內容，版本控制器都會自動產生版本歷史清單以及修改紀錄。未來期望能顯示出每次需求的變更內容，以利整份需求文件的管控。

圖 35 自動化版本控制器

4.13匯出 XML 檔案

需求撰寫完畢時，可將內容匯出成 XML 格式的軟體需求文件。

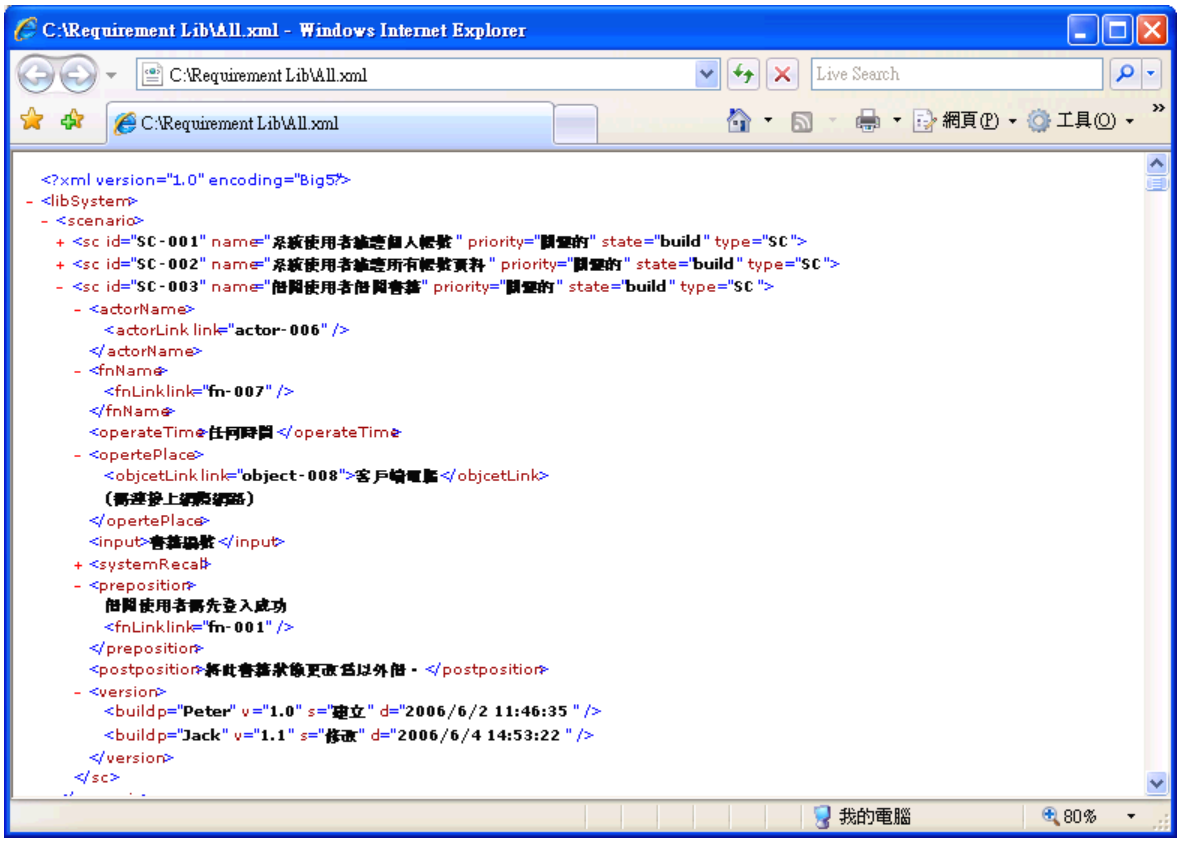

圖 36 軟體需求文件

第五章　結論與未來展望

需求分析是軟體開發生命週期中第一個步驟，但也是最重要的部份。但現今大多數的需求文件都是以自然語言的方式來描述，因此可能會有需求不一致或是不明確的情況產生，進而導致後續的軟體開發與維護成本高昂且易於出錯。有鑑於此，本研究開發出一套以模型為基礎之物件導向需求編輯器，利用物件導向需求模型來描述需求，並加入專家領域的相關知識來協助系統分析師擷取出格式較正確且內容較完整的需求。

因本研究所提出之架構，所以目前只支援到需求分析階段的功能。在未來我們希望能基於本理論基礎，進一步開發出整合需求分析、規劃設計、實作一直到測試、維護等軟體開發流程中的每一個階段，不同標準化模型架構的表示方式，與串聯各軟體開發流程模型的對應機制與規則，建置一套以一致標準格式所架構出的自動化的軟體開發模式，達到整個軟體流程一氣呵成的目的，且降低軟體開發時間與成本。

參考文獻

1 Bauer F.L., NATO Software Engineering Conference, 1968.

2 Coad P., and Yourdon E., OOA – Object-Oriented Analysis 2nd Edition, Prentice Hall, 1990.

3 Carson R.S., “Keeping the Focus During Requirements Analysis,” Proceedings of the 11th International Symposium of the International Council on Systems Engineering (INCOSE), Melbourne, Australia, 2001.

4 Clarke S., Murphy J., and Roantree M., “Composition of UML Design Models: A Tool to Support the Resolution of Conflicts,” Proceedings of the 5th International Conference on Object-Oriented Information Systems, 1998, pp.464-479.

5 Dai L., Cooper K., “Modeling and Analysis of Non-functional Requirements as Aspects in a UML Based Architecture Design,” Proceedings of the Sixth International Conference on Software Engineering, May 2005, pp.178-183.

6 Finkelstein A., Gabbay D., Hunter A., and Nuseibeh B., “Inconsistency Handling in Multi-Perspective Specifications,” IEEE Transactions on Software Engineering, Vol. 20, No.8, 1994, pp. 569-578.

7 Ghezzi C., Jazayeri M., and Mandrioli D., Fundamentals of Software Engineering, 2nd ed., Prentice Hall, 2002.

8 Grosz G., “Building Information System Requirements Using Generic Structure”, IEEE 16th Annual International Computer Software and Applications Conference, 1992, pp.200-205.

9 Heitmeyer C., Jeffords R., and Kiskis D., “Automated Consistency Checking Requirements Specifications,” ACM Transactions on Software Engineering and Methodology, vol. 5, no. 3, pp. 231-261, 1996.

10 Hooks I., “Writing Good Requirements,” the Proceedings of the 3rd NCOSE International Symposium, 2003.

11 Hooper J.W., and Hsia P., “Scenario-based for Requirements Identification”, ACM Sigsoft Software Engineering, Vol.7, No. 5, 1982, pp.88-93.

12 Ian Sommerville, Software Engineering Sixth Edition, Addison-Wesley, 2001.

13 IEEE, IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12, 1990.

14 Kaindl H., “A practical approach to combining requirements definition and object-oriented analysis,” Annals of Software Engineering, Vol. 3, 1997, pp. 319-343.

15 Kaindl H., “Difficulties in the transition from OO analysis to design,” IEEE Software, Vol. 16, No.5, 1999, pp.94-102.

16 Kasser J.E., “Towards improving the recognition and correction of poor requirements,” the Proceedings of SETE 2005, ICE Australia, 2005, pp.1-13.

17 Kasser J.E. and Williams V. R., “What Do You Mean You Can't Tell Me If My Project Is in Trouble?,” Proceeding of Software Metrics (FESMA 98), Antwerp, Belgium, 1998.

18 Kotonya, G. and Sommerville I., “Requirements Engineering with Viewpoints,” BCS/IEE Journal of Software Engineering, Vol. 11, No.

1, 1996, pp. 5–18.

19 Lamsweerde V., Darimont R., and Letier E., “Managing Conflict in Goal-Driven Requirements Engineering,” IEEE Transactions on Software Engineering, Vol. 24, No. 11, 1998, pp. 908-926.

20 Lee J., and Xue, N.L. “FOOM: a fuzzy object-oriented modeling for imprecise requirements,” the Proceeding of Fuzzy Information, 1998, pp.345-349.

21 Lee J. and Kuo J.Y., “A new approach to requirements trade-off analysis for complex systems,” IEEE Transactions on Knowledge and Data Engineering, Vol. 10, No.4, 1998, pp.551-562.

22 Leffingwell, D. and Widring D., Managing Software Requirements, Addison-Wesley, 2000.

23 Lubars, M., Potts C., and Richter C., “Developing Initial OOA Models,” In Proceedings of the Fifteenth International Conference on Software Engineering (ICSE-15), IEEE Computer Society Press, Los Alamitos, CA, 1993, pp. 255–264.

24 Maiden N., Robertsoon S., “Developing Use Cases and Scenarios in the Requirements Process,” Proceedings of 27th International Conference on ICSE’05, May 2005, pp.15-21.

25 Penna G.D., Intrigila B., Larurenzi A.R., Orefice S., “An XML Definition Language to Support Scenario-Based Requirements Engineering,” International Journal of Software Engineering and Knowledge Engineering, Vol. 13, No. 3, Apr. 2003, pp.237-256.

26 Salem A.M., Darter M.O., and Ramanujam B., “A Practical Method for Performing Object Oriented Requirement Analysis,” the Proceedings of International Conference on Computer Science, 2005.

27 Schach S. R., Software Engineering, Aksen Associates Pacific Palisades, CA, USA, 1990.

28 Seybold C., Meier S., “Evolution of Requirements Models by Simulation,” Proceedings of the 7th International Workshop on Principles of Software Evolution, 2004, pp.43-48.

29 Standish, “The Chaos Report," http://www.standishgroup.com/chaos.html, March 19, 1998.

30 Stephane S. Some, “Supporting use case based requirements engineering,” Information and Software Technology, Vol. 48, No. 1, Jan. 2006, pp. 43-58.

31 SWEBOK, http://www.swebok.org, 2007.

32 The Standish Group, Chaos Report, http://www.standishgroup.com/, 2007.

33 Tran X.L, and Kasser J., “'Towards improving the recognition and correction of poor requirements,” Proceeding of Systems Engineering/Test and Evaluation, pp.1-13, 2005

34 Whittle, J., Araujo J., “Scenario Modeling with Aspects,” Proceeding of IEE Software, Vol.151, No.4, Aug. 2004

35 Zhu X., and Jin Z., “Detecting of requirements inconsistency: an ontology-based approach,” Proceedings of the 2005 The Fifth International Conference on Computer and Information Technology, 2005, pp.869- 875.