國立交通大學

工業工程與管理學系

碩士論文

建構多重使用者之生產與運籌管理網路遊戲

The Construction of a Multi-user Web-based Pedagogical Game for Production and Logistics Management

研 究 生：藍國維

指導教授：張永佳 博士

中華民國九十六年六月

建構多重使用者之生產與運籌管理網路遊戲

研究生：藍國維　　指導教授：張永佳教授

國立交通大學工業工程與管理學系碩士班

摘要

生產與運籌管理遊戲平台(Simulation of production and logistics environment, 以下簡稱 SIMPLE 1.0)於 2006 年由張永佳所發展出來，其主要目的在於提供工業工程相關領域的教師一個教學輔助的工具，透過遊戲的參數設定可以適用於不同的教學課程。不過 SIMPLE 1.0 目前為僅限於單人使用，玩家無法進行多人連線遊戲，且由於遊戲設定與遊戲執行皆在同一個介面上，因此教師所能掌控遊戲與參與者的部份略嫌不足；因此為了使 SIMPLE 1.0 更具備彈性，更方便教師與遊戲參與者使用，本研究將擴充及改善 SIMPLE 1.0，並稱改善後的系統為 SIMPLE 2.0。

本研究以軟體生命週期模型中的漸增模型作為軟體的開發模型，除了可以讓不同遊戲參與者透過網路在同一場遊戲中進行遊戲之外，並分隔出各種使用者型態並賦予不同權限，以方便遊戲的監控及管理。此外，本研究利用軟體架構中的主從式模型來建構遊戲系統及網路，讓遊戲參與者可以透過網路連接至遊戲伺服器進行遊戲；而資料庫部份則是利用關聯式資料模型來建構。

在實作方面本研究的遊戲系統是利用 JAVA 物件導向的功能來建構，而系統管理頁面則是利用 JSP 網頁開發技術撰寫，系統資料庫實作則是透過 HSQL 建構。

關鍵字：教學輔助工具，漸增模型，主從式模型，關聯式資料模型。

The Construction of a Multi-user Web-based Pedagogical Game for Production and Logistics Management

Student：Kuo-Wei, Lan　　Advisor：Yung-Chia, Chang

Department of Institute of Industrial Engineering & Management
National Chiao Tung University

Abstract

Simulation of production and logistics environment (referred as SIMPLE 1.0 hereafter) was developed by Chang (2006) as a teaching aid applicable in courses related to Industrial Engineering and Management in order to increase teaching effectiveness. SIMPLE 1.0 is designed for single-user application thus is lack of the ability of multiple-user interaction. In order to extend SIMPLE 1.0 to make it more flexible and convenient for users, we constructed an improved version of SIMPLE (referred as SIMPLE 2.0 hereafter) in this research.

We chose the incremental model among the software development life cycle models to construct SIMPLE 2.0. The system is designed such that the number of players allowed in one game and the game settings are fully controllable by the game administrator via game management pages separated from game-running environment. Through the implementation of a server-client model, multiple players are able to access SIMPLE 2.0 through internet connection and play interactively with one another. We also designed a relational database system to store game results and user information so that system administers are able to retrieve data from the database for further analysis. The main program is written by Java, the game management pages are implemented by JavaServer Page (JSP) technique, and the database is implemented by HSQL.

Key words: Teaching aid, incremental model, server-client model, relational database.

目錄

圖目錄

表目錄

第一章　緒論

1.1 研究動機與背景

隨著時代的進步，教師傳授學生知識的方式也有所變化。以往，學生只能藉由課本及老師上課所描述的內容去做學習，但這樣的學習效果畢竟有限。於是乎從國小教育開始，甚至到國中、高中，大學，就有許多的實習課程，目的就是想讓學生透過實驗或是實際操作藉此加深對於該門科目的認識。不過，仍然有一些科目無法透過「做實驗」或「實際操作」的方式進行，大專院校的生產管理及供應鏈管理這兩門科目就是最好的例子。如大專院校的生產管理課程，學生無法實際透過實驗瞭解所謂的「經濟訂購量模式」、「報童模式」等存貨管理策略；又如供應鏈管理課程，學生沒有辦法實際扮演供應鏈中的一個廠商，藉此親身體會「長鞭效應」對整條供應鏈造成的影響。因此學生從課本中所學到的，都只有抽象的概念，並沒有辦法透過實驗了解其中的涵義。有鑒於此，許多教學輔助的工具被發展出來，而「遊戲」就是教學輔助中的工具之一。

目前一般大專院校的教師在教授生產管理、庫存管理及供應鏈管理碰到的挑戰，並非像以往只需單純的傳授知識，而在於如何應用現有的教學輔助工具來提高傳授知識的效率，並且讓學生能快速學習並且將所學的知識融會貫通。而「遊戲」不失為現有的教學輔助工具中，一個有效的教學輔助教材。

遊戲提供一個有趣的學習情境（Martocchio and Webster, 1992），而利用遊戲的情境下，大多數學生會保持高度的注意力進行遊戲（葉盛昌，2003）。藉由學生專注於遊戲並且積極想要贏得勝利，學生必須想出方法以解決遊戲中所會碰到的問題，而往往解決問題的方法就隱藏在教師授課的內容中，如此一來可以提高學生的學習動機。而 Quinn（1996）也提出

遊戲有「提高內在動機」的特性，使得學生更容易理解授課內容並且快速吸收。因此，透過遊戲的方式除了可以提昇學生學習的興趣、建立良好的學習態度之外，亦可以訓練學生的思考能力，讓學生將其在課程中所學的知識充分吸收並且在遊戲中加以利用，以達到最佳的學習效果。

在過去的十幾年來，已經有許多有關生產管理、存貨管理及供應鏈管理等教學輔助遊戲被設計出來，提供教師在教學上的方便。在這個領域中最著名的遊戲莫過於由麻省理工學院（Massachusetts Institute of Technology）史隆管理學院（Sloan Management school）於 1960 年間所發展出來的啤酒遊戲（beer game），其主要的目的希望使用者能夠藉由遊戲了解供應鏈中長鞭效應的存在，與遊戲中各決策者的行為表現如何反應在供應鏈的系統。

目前雖然有許多有關生產、存貨和供應鏈管理等遊戲應用在教學上，但是大多數的遊戲常常是為了某一個特定課程或是某一個課程章節的需要而被設計出來，甚少有遊戲可以依照教學範圍而對遊戲進行調整。因此一般的教師在利用遊戲進行教學時，往往因應課程需要而使用不同的遊戲；當然，學生也需要隨著不同課程，學習操作不同的遊戲。如此一來，不但增加教師備課上的不便，亦會降低學生學習的興趣以及學習的效率。有鑑於此，張永佳(2006)建立了一個以網路為介面的生產與運籌管理遊戲平台(simulation of production and logistics environment, 以下稱為 SIMPLE 1.0) 。其主要目的在於提供教學上使用，讓使用者可以依照不同課程的需要使用相同的遊戲介面來調整遊戲參數，使其適用於不同的課程，藉此提高老師教學以及學生學習的效率。SIMPLE 1.0 提供了兩種遊戲進行的方式，分別為「工廠模式」及「供應鏈模式」。在工廠模式中，玩家扮演一間工廠的廠長，負責訂定存貨及訂貨策略，目標在於最小化其工廠的營運成本。而在供應鏈模式中，SIMPLE 1.0 設計了一條由「供應商」、「製造商」、「物流中心」及「零售商」組成的供應鏈，玩家可以選擇扮演這四

種角色中的一種進行遊戲，其目標在於使整條供應鏈的成本最小化。除了上述功能外，SIMPLE 1.0 提供了數種遊戲進行策略、倉儲空間限制選擇及市場需求設定等功能。

SIMPLE 1.0 目前為僅限於單人使用，玩家無法進行多人連線遊戲，且由於遊戲設定與遊戲執行皆在同一個介面上，教師所能掌控遊戲與參與者的部份略嫌不足；除此之外，其遊戲介面目前也僅為中文語系，無法讓非中文語系國家的其他遊戲參與者可以順利進行遊戲。因此為了使 SIMPLE 1.0 更具備彈性，更方便教師與遊戲參與者使用，本研究將擴充及改善 SIMPLE 1.0，並稱改善後的系統為 SIMPLE 2.0。

1.2 研究目的

本研究主要的目的為擴充與改善 SIMPLE 1.0 的功能使其較具備彈性及方便教師與遊戲參與者使用，以下稱本研究所發展的系統為 SIMPLE 2.0。SIMPLE 2.0 擴充及改善的功能如下：

1. 將遊戲的供應鏈模式擴充成可自由彈性調整角色數量。
2. 將遊戲擴充成多語化介面，使得非中文語系國家的遊戲參與者亦可執行遊戲。
3. 讓不同的遊戲參與者彼此可以透過網路連線共同進行遊戲。
4. 將遊戲設定部份與遊戲介面分開，額外設置遊戲設定及管理介面。

1.3 研究架構

本論文的研究架構如圖 1所示。在第一章的緒論中，包含研究動機與背景、研究目的和研究架構；第二章則是相關文獻的探討，這其中包括本研究在開發遊戲時所決定使用的軟體開發模型及架構；第三章則進行遊戲系統的分析與設計；第四章則進行遊戲系統的實作及展示；最後於第五章作結論並說明未來系統的發展方向。

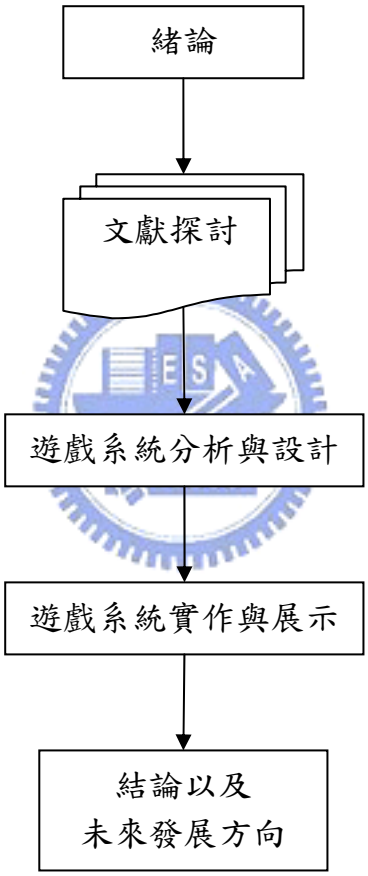

圖 1　研究架構圖

第二章　文獻探討

在本章有三小節，2.1 節探討和生產與運籌管理相關的遊戲。2.2 節則探討在開發軟體時所會使用到軟體開發模型；2.3 節則是對軟體的架構設計作探討與分析。

2.1　生產與運籌管理相關遊戲之介紹

過去十幾年來，已經有許多生產與運籌管理的遊戲被應用在教學上，在表 1中，為這十幾年來針對生產與運籌管理所設計的一些遊戲。遊戲可以將原本實際複雜的體系變得簡單而且有趣，若更進一步以競賽的方式進行，可以提高學生參與的興趣，使學生在操作過程中，對於生產與物流中所發生的實際狀況更能深刻體會（張永佳，2006）。

表 1　生產與運籌管理相關遊戲

遊戲名稱	描述	決策範圍	發展者	發展時間
MIT beer game http://beergame.mit.edu/	啤酒的生產跟配銷在多階配銷通道	扮演製造商、經銷商、零售商去決定啤酒生產或訂購量	Massachusetts Institute of Technology, U.S.A.	1988
MA beer game http://www.masystem.com/beergame	與啤酒遊戲類似	可選擇扮演製造商、經銷商、零售商去決定啤酒生產或訂購量	MA-system (Sweden)	1976
Hulia game http://hulia.haifa.ac.il/Eng/hulia.html	與啤酒遊戲類似，模擬動態決策系統	扮演製造商、經銷商、零售商去決定生產或訂購量	The University of Haifa, Israel	2000
Trading Agent Competition http://www.sics.se/tac	模擬多個市場線上貿易的供應鏈遊戲	決定對客戶最佳的服務	Swedish Institute of Computer Science	2003
Littlefield Technology http://littlefield.responsive.net	模擬接單後製造的生產系統	扮演製造商去決定利用率、生產排程以及存貨	Stanford University, USA.	1996
Distributor Game http://www.gscg.org	強調配銷者的責任的互動	扮演配銷商去決定產程、路徑以及排程等議題	Delft University of Technology, The Netherlands	2005
SIMPLE 1.0	強調可以應用在不同生產與運籌管理的課程	可依照不同模式扮演不同角色	National Chiao Tung University, Taiwan	2006

由於生產與運籌管理的遊戲眾多，因此在本節中只針對 MIT 的啤酒遊戲（beer game）、Trading Agent Competition（TAC）遊戲和 SIMPLE 1.0 做介紹。

2.1.1 啤酒遊戲

啤酒遊戲是在 1960 年由麻省理工（Massachusetts Institute of Technology）史隆管理學院（Sloan Management School）所發展出來，它是類似「大富翁」的紙上遊戲，只是它比大富翁更簡單，因為參與者只需要作一項決策就是下訂單與上游遊戲參與者訂購貨品。此遊戲主要的目的則是希望使用者能夠藉由遊戲了解供應鏈中長鞭效應（bullwhip effect）的存在，與遊戲中各決策者的行為表現如何影響整體供應鏈的系統 (Sterman, 1984)。在 1988 年的 2 月，Pecos River Learning Center才發展出電子版的啤酒遊戲，如圖 2為電子版啤酒遊戲的畫面。

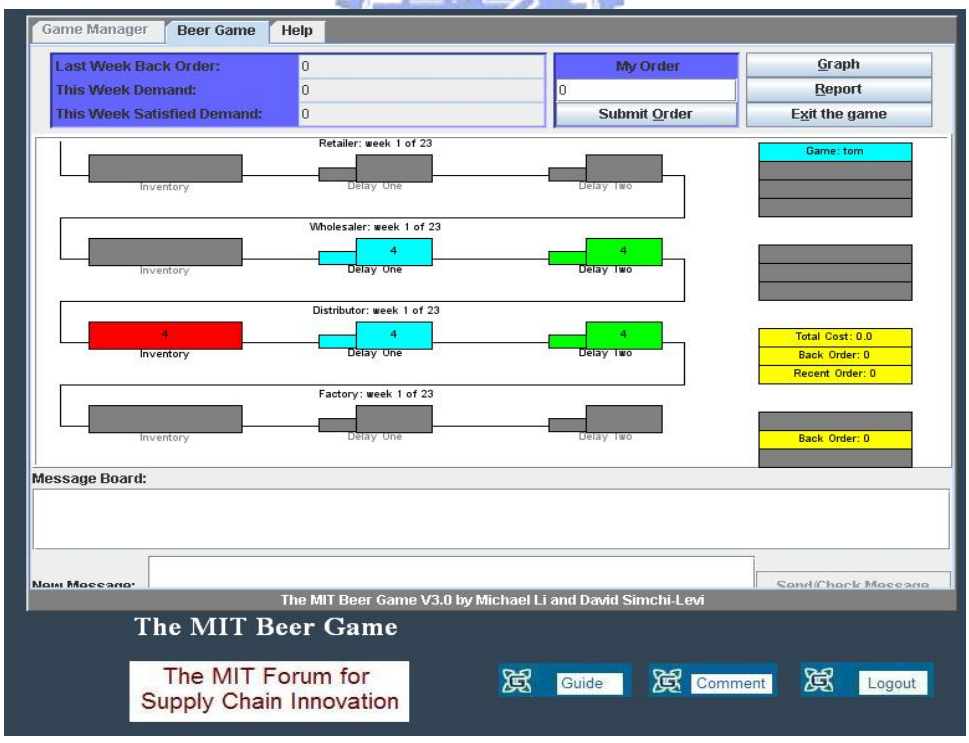

圖 2　電子版啤酒遊戲（http://beergame.mit.edu/）

電子版啤酒遊戲的供應鏈上主要由工廠、配銷商、批發商及零售商四種角色所組成，在遊戲開始前遊戲參與者可以選擇想要扮演的角色和設定

遊戲的週數，並且可以透過資訊透明化功能，觀察其他遊戲參與者的遊戲進行狀況。在啤酒遊戲中每個角色的遊戲參數設定皆相同，系統只計算存貨成本以及缺貨成本(存貨的單位成本為 0.5 元,缺貨的單位成本為 1 元)，並且有 1 週的資訊延遲時間與 2 週的物流前置時間。

2.1.2 TAC 遊戲

TAC 為 2000 年在密西根大學（University of Michigan）的人工智慧實驗室所發展出來的遊戲,而自 2002 年開始由瑞典電腦科學研究所(Swedish Institute of Computer Science, SICS）每年定期舉辦的 TAC 遊戲的競賽。

這個遊戲主要分成兩種情境：

1. TAC CLASSIC：如圖 3所示，這個模式的遊戲可以讓 8 個遊戲參與者扮演經紀人（agent）的角色，每一個經紀人都分別有 8 位客戶。經紀人主要就是負責將旅遊套裝（travel package）在 5 天內從TAC市(TAC Town)運送到坦帕市(Tampa)。所謂的旅遊套裝包含旅客的來回機票、預定住宿的旅館（有高級旅館和普通旅館兩種）和具有娛樂性質的門票（有美國摔角、娛樂公園和博物館這三種門票）；而機票有 8 種拍賣（auction）方式，旅館有 8 種拍賣方式，娛樂性質的門票則有 12 種拍賣方式。遊戲的評分方式是以客戶的滿意度作為評分標準，每個遊戲參與者的目標就是想辦法完成客戶的要求以最大化顧客滿意度。

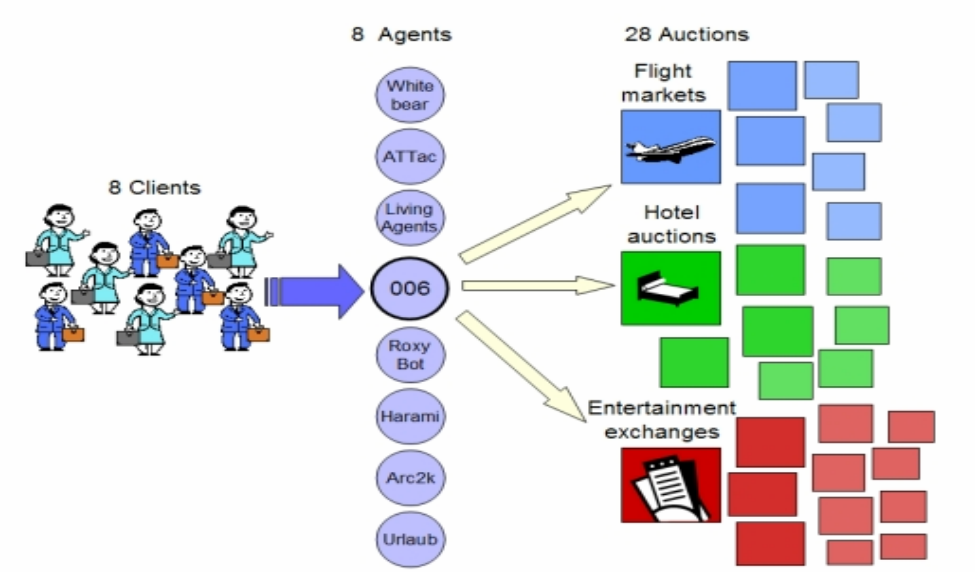

圖 3　TAC CLASSIC（http://www.sics.se/tac/）

2. TAC SCM：這個模式的遊戲主要是模擬電腦經紀人如何銷售個人電腦（personal computer）的情境。遊戲中，一台個人電腦由中央處理器（central processing unit, CPU）、主機板（motherboard）、記憶體（memory）和硬碟（hard disk）所組成；由遊戲參與者扮演的經紀人必須爭取客戶的訂單，同時也要根據不同顧客的需求向供應商購買零組件的材料，再交由工廠製造；除此之外，經紀人的工廠都有產能的限制。

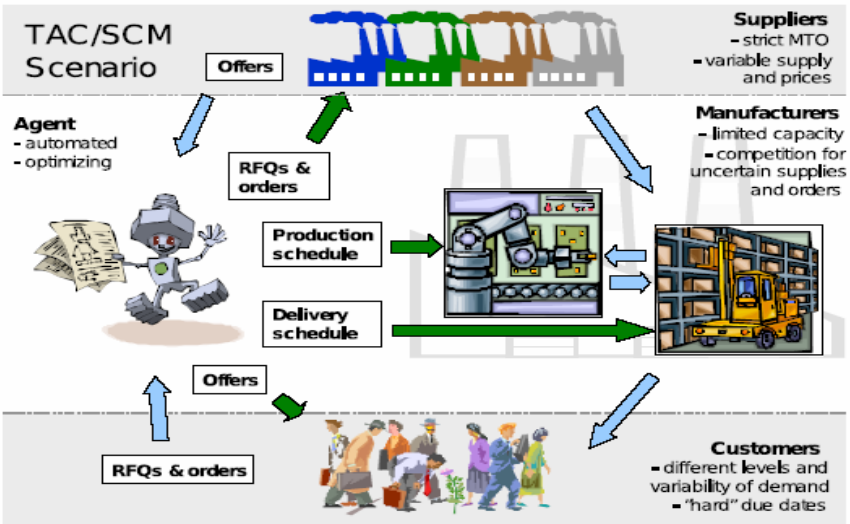

圖 4　TAC SCM（Collins, 2006）

如圖 4所示，TAC的架構主要分成三層：

i. 最上層的是供應商。由於一台個人電腦是由四種零組件組合而成，每一種零組件皆有兩種不同廠牌的供應商，因此共有 8 間不同的供應商，而每間供應商是以接單式生產（make-to-order）的方式來符合經紀人的需求。

ii. 第二層是電腦經紀人，也是遊戲參與者主要扮演的角色。在圖 5中，可以明顯的看出經紀人所要執行的動作。經紀人需要向顧客提出電腦的報價（requests for quotes, RFQs）和交期，以爭取到訂單；另一方面，除了需要控管自己的倉庫零組件的存貨量，若有不足則需要向供應商洽詢零組件價錢之後再決定是否要訂購之外，還需要擬定工廠的生產排程以符合交期。

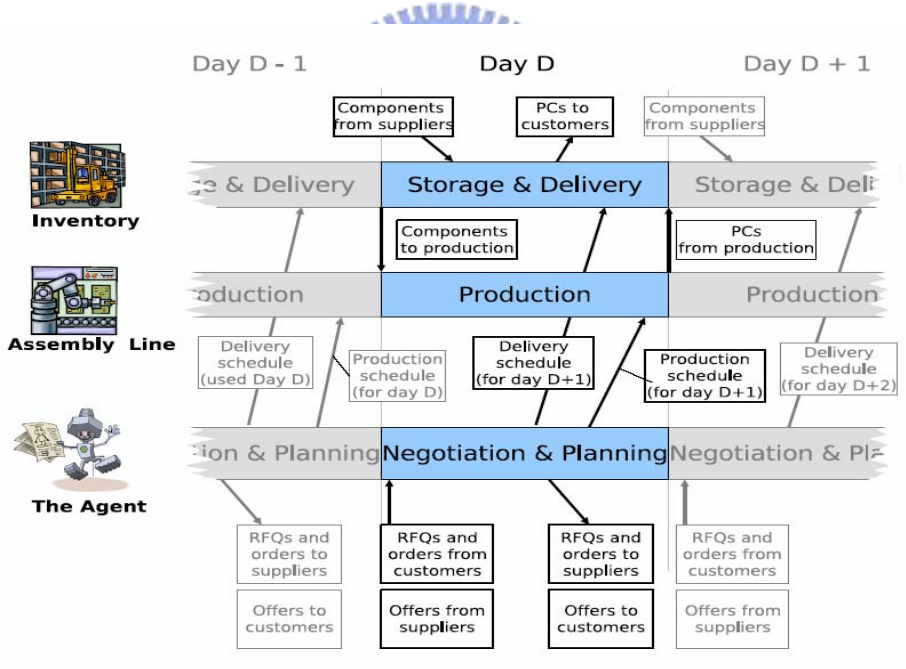

圖 5　經紀人動作分析圖（Collins, 2006）

iii. 最下層是顧客。顧客會有不同的電腦需求，於是向經紀人洽詢；顧客根據經紀人的報價以及可交貨的日期，再決定向哪位經紀人訂購電腦。

除了第二層可以由 6 位遊戲參與者擔任經紀人的角色之外，

供應商及顧客皆是由伺服器控制。在遊戲的最後，哪位玩家所賺的錢最多，則該位遊戲參與者為優勝者。

2.1.3 SIMPLE 1.0

SIMPLE 1.0 為 2006 年由國立交通大學工業工程與管理學系的供應鏈管理實驗室所發展出來的，其主要目的是希望能夠讓教師藉由這套教學遊戲軟體，減少備課上負擔以增進教學的效率（張永佳，2006）。這個遊戲主要分成兩種模式：

1. 供應鏈模式：遊戲在這個模式下主要是由四種角色-供應商、製造商、物流中心及零售商所組成，如圖 6所示。遊戲參與者可以選擇扮演其中一個角色，而其他三個角色由電腦控管。這三個由電腦控管的角色，遊戲參與者可以根據不同的需求，分別設定每一個角色的執行策略。這個遊戲中，遊戲參與者若是選擇扮演製造商之外的角色，其主要的動作就是根據自己擬定的策略與上游角色下訂單；若是扮演製造商，因為工廠有產能的限制，除了下訂單之外，當產能不足時還需要決定要以加班或是外包的方式增加產能。

圖 6　SIMPLE 1.0 系統畫面（張永佳，2006）

2. 工廠模式：遊戲在這個模式下遊戲參與者僅可扮演製造商的角色，遊戲參與者必須依照顧客的不同需求來決定工廠的生產策略，目標是最小化該工廠的總成本。

2.2 軟體工程

所謂軟體工程，就是有系統的進行軟體的規劃、分析、設計、程式撰寫和維護等工作，其目的就是希望運用科學化的方法來提高軟體的品質（Thayer, 1997）。軟體工程所涵蓋的範圍相當廣泛，主要可以分成兩方面：軟體發展技術和軟體專案管理。軟體發展技術包括軟體發展方法、軟體工具和軟體工程環境，而軟體專案管理包含軟體度量、專案預估、專案進度控制、人員組織、配置管理和專案計畫等。軟體工程師的工作則是包含上述的內容。

2.2.1 軟體生命週期模型

軟體工程師對於軟體的開發所執行的策略被稱為軟體發展生命週期模型(software development life cycle model, SDLC)。而軟體發展生命週期模型的選擇，是根據所開發的軟體的性質及需求決定的（陳湘楊，2004）。軟體開發生命週期模型將軟體開發的活動分成數個階段，每一個階段的關係以概念性的模型表現出來，因此不同的模型對於軟體開發的各階層關係皆有不同的定義。

早期的軟體開發並不會遵循一定的程序，而Royce(1970)為最早提出軟體開發的模型的人，他所提出的模型為瀑布模型(waterfall model)。在此之後，由於不同的軟體開發的需求，瀑布模型已不敷各式軟體開發使用，於是便有漸增模型(incremental model)、螺旋模型(spiral model)等模型一一被提出來。軟體開發模型的發展演進如圖 7。

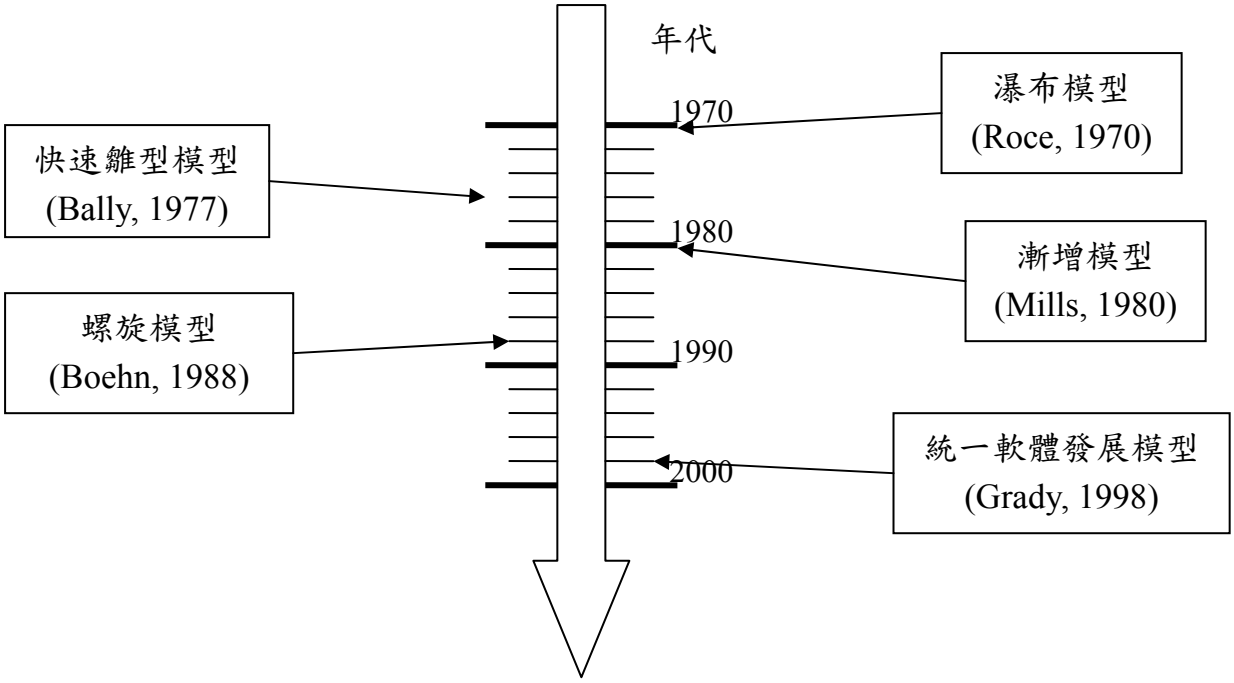

圖 7　軟體開發模型演進圖（陳湘楊，2004）

2.2.2　軟體生命週期模型的種類

由於軟體生命週期模型的種類繁多，本節只探討瀑布模型、漸增模型、螺旋模型和統一軟體發展模型。

（一）瀑布模型

瀑布模型亦可稱為古典生命週期或是線性序列模型，它是最早被提出的軟體生命週期模型，其程序架構如圖 8所示。瀑布模型的軟體開發的方式是採取階段式的，當這一個階段被完成之後才能進入下一個階段繼續執行。這個模型中主要階段可以對應到幾個軟體開發的基本活動：

1. 需求分析及定義：與系統使用者進行溝通協調之後，定義出系統的服務限制及目標，然後再詳細定義成系統規格。
2. 系統與軟體設計：系統的設計程序會將需求分割成硬體和軟體系統，以建立整體系統的架構。
3. 實作與單元測試：在這一個階段，將軟體設計實際做成一系列的

程式或是程式單元；單元測試則測試每一個單元是否符合規格。

4. 整合與系統測試：整合獨立的程式單元或各個程式，並且以完整的系統進行測試，以確保系統是否符合軟體要求。

5. 運作與維護：這個階段通常是花費最長時間的，包含系統的安裝和實際的運作。維護則包括修正之前沒有發現的錯誤，並當新的需求出現時增強系統的服務。

原則上，每一個階段都會產生一份或多份被核准的文件，這個動作稱為「signed off」(Jacobson, 1999)；這份文件主要用意就是確保下一階段必須等到前一個階段完成之後才可以進行。不過實際上，這些階段往往會互相重疊，並且互相提供資訊；在設計階段會發現有關需求的問題，在程式撰寫階段也可能會發現設計上的問題，原因是軟體程序並不是單純的線性模型，它包含了一連串重覆開發動作(Sommerville, 2006)。

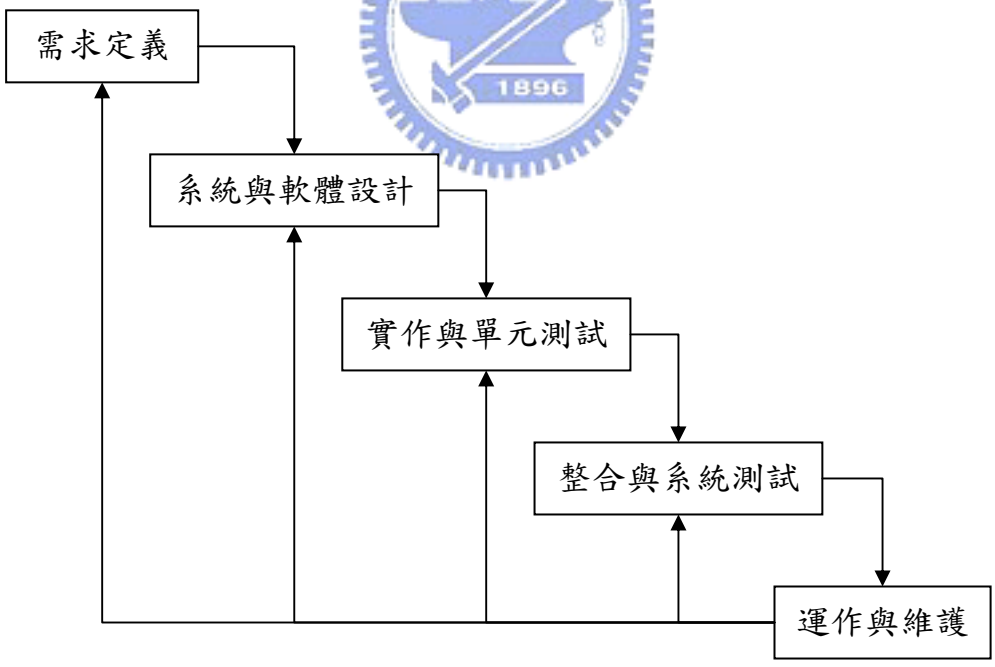

圖 8　瀑布模型圖（翻譯自 Sommerville, 2006）

因為製作與核准文件非常消耗成本，所以重複動作的成本非常高。因此，當某一個階段重複數次之後就會被停止而繼續進行下一個開發階段，

有問題的話則留給後面階段處理或是利用程式解決。在開發程式的運作與維護階段，便有可能在原來的軟體需求中發現錯誤和疏忽的地方；所以系統才必須持續發展及維護，使得軟體可以繼續使用。

使用瀑布模型開發軟體時，系統的需求必須在設計開始之前就要提出來，設計師則必須在實際製作軟體之前提出設計策略；如果開發階段需求有了變更，則系統的需求、設計及實際製作都必需要重頭來過，這是瀑布模型最大的缺點。不過它的優點是容易管理，且因為設計與實際製作分開，因此可以有比較穩固且容易修改的系統。

（二）漸增模型

漸增模型是由Mills *et al.*在 1980 年提出的，目的就是為了避免重做開發程序的動作，讓系統使用者有機會先對系統有一些經驗，然後再提出詳細的需求。圖 9為漸增模型的軟體開發圖。

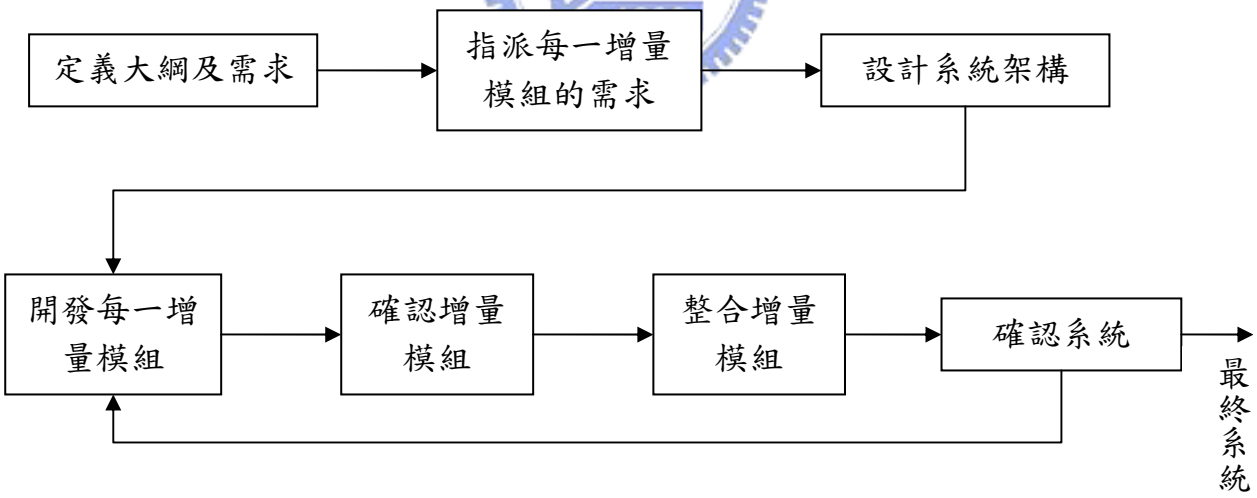

圖 9　漸增模型圖(Aurum, 2005)

使用漸增模型開發軟體時，可以先概略的提出系統的需求，並且決定哪些需求比較重要，哪些需求比較不重要；接著定義出一系列的增量模組。每一個增量模組均提供了某一部份的系統功能，而增量模組所配置的系統需求則是依據優先順序來決定的，比較重要的系統需求需要先提供給

系統使用者使用，讓系統使用者對系統有一些概念跟經驗。

系統的增量模組決定之後，就可以先詳細定義出第一個增量模組的系統需求，接著在利用適當的開發程序開發這個增量模組。在開發階段可以針對下一個增量模組的需求進行分析，但是不允許變更目前這個增量模組的需求。當有增量模組開發完成後，就移交給系統使用者測試使用。系統使用者可以先試用這些完成的功能，以便能夠更清楚的知道下一個增量模組的需求，或是針對目前的增量模組進行改版的需求。當有新的模組完成後，便加入現有已完成的模組中，因此每次有新的增量模組加入後，系統的功能便能更進一步的改進，直至完成系統。

漸增模型的優點有下列幾項(Aurum, 2005)：

1. 系統使用者不需要等到整個系統完成之後才開始使用。因為當第一個最高優先的增量模組完成後，系統使用者便可以開始使用系統該項功能。
2. 系統使用者可以把最一開始的增量模組當作雛型，之後藉由這個雛型取得一些經驗，再將新的需求增加至後續的增量模組中。
3. 用此模型的軟體開發失敗機率相當低。
4. 系統不容易出錯。因為最高優先的增量模組會先開發並交由系統使用者使用，之後的增量模組再與先前的模組做整合，所以系統最重要的部份被使用者測試最多次，這也表示系統最核心的部份出錯的機率很低。

（三）螺旋模型

螺旋模型是由Boehm在 1980 年首先提出的，這種軟體開發模型的程序是以螺旋的方式來表示，如圖 10所示。螺旋中的每一個迴圈代表軟體開發程序中的一個階段，內圈與系統的可行性最有關係，再向外一圈則與系統需求的定義有關，再向外一圈則與系統的設計有關，以此類推（Boehm,

1988)。

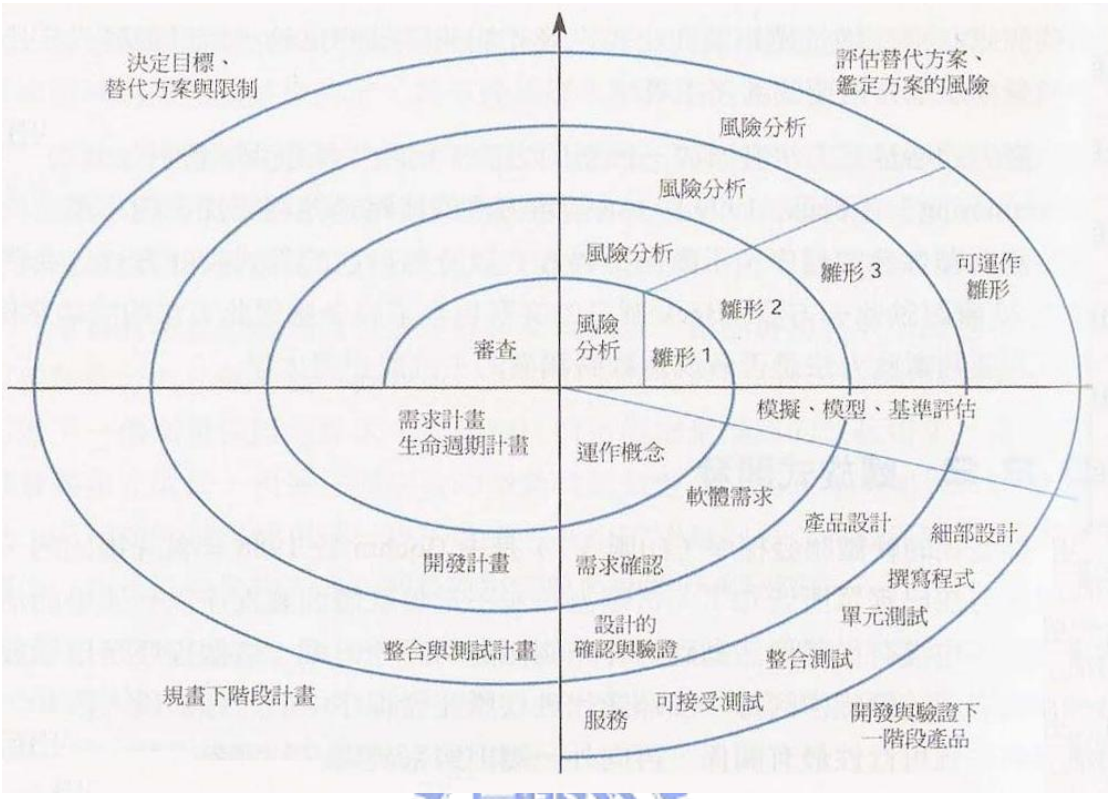

圖 10　螺旋模型的軟體程序（姜子龍，2004）

螺旋中的每一個迴圈可以分成四個部份：

1. 目標設定：定義此專案目前階段的特定目標、辨識程序、產品的限制、詳細的管理計畫、鑑定專案的風險程度以及根據此風險程度規劃出替代策略。
2. 風險評估與降低風險：對每一個被鑑定出的專案風險進行詳細的分析，並且採取風險較低的步驟。
3. 開發與確認：風險評估後就可以對此系統選用適當的開發模型。例如，若發現最主要的風險是子系統的整合，最適當的開發模型則為瀑布式模型。
4. 計畫：審查此專案，並且決定是否進行下一個螺旋迴圈。若決定繼續進行，則必須擬定出此專案的下一個階段的計畫。

螺旋模型和其他軟體開發模型最主要的差異在於螺旋模型在程序中

明確的考慮到專案開發的風險。所謂的風險就是在專案開發過程中，出現不正確的事情。舉例來說，倘若使用新的程式語言，其可能的風險是程式的編譯器（compiler）無法產生非常有效率的程式碼或是常常編譯失敗等問題，而這些問題往往會造成效率的降低和成本的提高；利用螺旋模型可以將這些風險因素都納入考量，藉此降低風險。不過螺旋模型並沒有規格制定或設計等固定階段，因為它可以包含其他的軟體開發模型。比如說，螺旋模型可以使用漸增模型來解決需求不確定的問題，以降低風險程度；之後再利用傳統的瀑布模型來開發軟體。

（四）統一軟體發展模型

統一軟體發展模型又稱為 Rational 統一流程(rational unified process, RUP)，它是由 Grady 在 1980 年所提出來的，主要是利用物件導向的技術來完成軟體的開發。RUP 總共有三大特點、四個階段和九個核心工作流程（core workflows）（Kruchten, 2003）。RUP 的三大特點：

1. 軟體開發是一個迭代（iteratiative）的過程。
2. 軟體開發是使用者所驅動的。
3. 軟體開發是以架構設計（architectural design）為中心。

利用物件導向的分析與設計概念，RUP 將軟體開發的程序分成以下四個階段：

1. 起始階段（initial phase）：定義專案大小以及其涵蓋的範圍，評估專案所需要的能力、時程與經費，以及資訊系統預期達到之效益了解商業模型及需求。
2. 精細規劃階段（elaboration phase）：擬定專案計畫、系統特性與架構，並確認商業模型及需求，然後進行系統分析與設計。
3. 建構階段（construction phase）：建構產品並進行單元測試、整合測試。

4. 移轉階段（transition phase）：將產品分批交付給客戶進行驗收測試，並進行使用者訓練。

以上四個階段並非一次從頭做到尾，而是以重複往返且漸增（iterative and incrementally）的方式進行，採用的是類似螺旋式模型的生命週期。重複往返式與漸增式流程的主要精神在於每一個階段都可重複好幾次核心流程，階段與階段間也可以重複好幾次該模型。這個方法主要目標有兩項：第一，在每一次完成一個階段的時候，增加此軟體的部份功能，送出一個可執行的版本給系統使用者；第二，維持短的傳送時間和頻繁的傳送，以避免專案執行時間過長造成時效性不佳的問題。

RUP 的九個核心工作流程如下：

1. 商業塑模（business modeling）：商業塑模工作流程描述如何為了一個新的組織目標開發一個構想；在商業模型中基於這個構想，定義開發過程、人員角色和責任。
2. 需求（requirements）：需求工作流程的目標是描述系統應該做什麼，並且使開發人員跟系統使用者就這一個描述達成共識。為了達到該目標，要對需要的功能與限制進行定義和其規定範圍。
3. 分析和設計（analyses and design）：分析和設計工作流程是將需求轉化成未來系統的設計，為系統開發一個穩固的結構並調整設計使其與實作的環境吻合。
4. 實作（implementation）：實作工作流程的目的為利用層次化的子系統的方式來定義實作的組織結構、實作的設計元件；並將開發出的元件作單元測試。將元件整合後，便可成為可執行的系統。
5. 測試（test）：測試工作流程的目的是要驗證軟體中所有元件是否皆正確，並檢驗軟體是否有符合需求的設定。
6. 部署（deployment）：部署工作流程的目的是確保將完成的軟體產品安全的送至最終使用者手上；有時，這個工作流程還會包括幫

使用者移除現有的軟體以安裝新的軟體。

7. 配置和變更管理（configuration and change management）：配置和變更管理工作流程描述如何識別管理配置的項目和配置改變的管理。
8. 專案管理（project management）：專案管理工作流程平衡各種可能產生衝突的目標，克服各種限制並交付使用者完美的產品。
9. 環境（environment）：環境工作流程的目的是向軟體開發組織提供軟體開發環境。

使用 RUP 開發軟體的優點是該方法結合了許多成功的模型與方法理論，提供完整的開發流程藉此降低開發失敗的風險，不過 RUP 因為結合太多方法理論，導致其製作程序變得相當龐大且複雜，因此 RUP 適合生命週期較長的中、大型的軟體專案。

2.2.3 小結

在 2.2.2.節中，共提到四種在軟體開發模型。其中瀑布模型雖然發展時間最早，但它的使用彈性較低，無法隨著臨時需求的變動而改變；螺旋模型因無規格制定或設計等固定階段，對於初次開發遊戲軟體的人員比較不適用；而 RUP 也是因為開發流程過於複雜，不適合初次開發遊戲軟體的人員。

相較於上述三種模型，漸增模型不但有規格制定和設計等固定階段，且系統開發人員可以先將部份完成的系統交由使用者測試，之後再根據使用者所提出的需求修改設計，在需求變更方面較具彈性。除此之外，漸增模型無複雜的開發流程使得初次開發遊戲軟體的人員容易上手。因此本研究決定使用漸增模型作為軟體生命週期模型。

2.3 軟體的架構設計

大型系統通常會被分解成數個子系統，而辨識這些子系統的初始設計程序以及建立子系統之間控制與通訊的骨架則稱為架構設計（architectural design）。軟體架構設計的重點是為系統建立一個基本的結構化骨架，以用來辨識子系統控制與溝通的框架（Sommerville, 2006）。Bass(2003)曾經提出明確的設計與紀錄一個軟體架構會得到以下三個好處：

1. 與利害關係人進行溝通：架構等於是系統比較高階的表示方法，所以可以當成各不同領域的利害關係人討論時的聚焦所在。
2. 系統分析：在系統開發初期階段時，明確的建立系統架構，分析此系統架構是否符合系統重要的需求。系統重要的需求包括執行效能、系統可靠性及系統可維護性等。
3. 大規模再利用：系統架構是對系統的組織方式和系統間元件的互動做一個簡單且容易管理的描述。如此一來，當這個系統需要轉移至其他相同架構的系統時，可以重複的利用。

在本節中，將會分成三個小節來探討三種架構設計的模型：倉庫模型（repository model）、主從式模型（server-client model）和抽象機器模型（abstract machine model）。

2.3.1 倉庫模型

組成一個系統的子系統之間一定要交換資訊才能有效的一起運作，交換資訊有兩個基本方法：

1. 將共用的資料保存在一個集中的資料庫，讓所有子系統都可以存取資料。以共用資料庫為基礎的軟體架構模型則稱為「倉庫模型」。
2. 每一個子系統都有各自的資料庫保存資料，需要交換資料的時候則透過子系統之間的訊息傳遞。

大多數使用大量資料的系統都會使用倉庫模型作為軟體的架構

（Pfleeger, 2001），因為這個模型適合於資料由某一個子系統產生之後存回共用資料庫，而其他子系統則透過共用資料庫再將資料取回使用。圖 11是一個利用倉庫模型來進行專案架構設計的案例圖。利用倉庫模型作為軟體架構的優點是可以有效分享大量的資料且子系統不用管資料如何產生；其缺點是執行效能不佳，且將來要轉換成其他架構，成本會非常昂貴。表 2整理了利用倉庫模型作為軟體架構的優缺點。

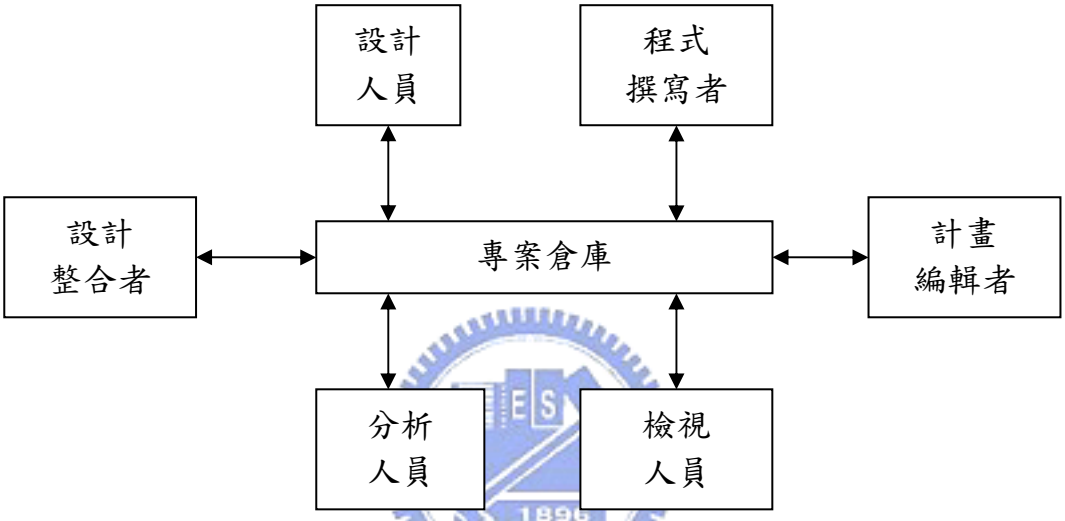

圖 11　倉庫模型案例圖（Sommerville, 2006）

表 2　倉庫模型之優缺點

倉庫模型的優點	倉庫模型的缺點
1. 有效的分享大量的資料 2. 子系統不用管資料如何產生 3. 集中式的管理，例如備份、機密性等	1. 各個子系統必須同意使用儲存庫模型，不可避免的，有些子系統必須妥協 2. 將來若要轉換成其他架構，成本會非常昂貴 3. 執行效能不佳

2.3.2 主從式模型

主從式架構模型式一種分散式的系統模型（Pressman, 2001）。這個模型的主要組成元件有：

1. 提供服務給其他子系統的一些獨立伺服器。例如提供列印服務的列印伺服器和提供檔案管理的檔案伺服器等。
2. 向伺服次要求提供服務的客戶端。這些客戶端通常是子系統，同一個客戶端程式可能在同一個時間有多個實例同時執行。
3. 可讓客戶端存取這些服務的網路。這個元件不是絕對需要，因為客戶端和伺服器可以放在同一部機器上執行，因此客戶端不需透過網路存取伺服器的資料。

在這個模型架構下，客戶端要知道可用的伺服器名稱以及伺服器所能提供的服務；伺服器則不需要知道客戶端的名稱及數量。主從式方法也可以用來實做一個以倉庫模型為架構的系統，只要將資料庫以一個系統伺服器來提供即可，而存取資料庫的子系統就是客戶端。圖 12是一個主從式架構為基礎所建構的系統範例，它是一個能夠提供影片和圖片資料庫的多使用者超文件系統。系統中有一些負責管理不同媒體類型的伺服器，另外有伺服器是提供超文件資訊系統的連結，以方便客戶端查詢目錄。客戶端程式則只是為了這些服提供一個整合的使用者介面。

主從式模型最大的好處就是它是屬於分散式架構（Christensen, 2002），這種以網路連結的系統可以有效的利用許多分散的處理器來執行，而且假如要加入一個新的伺服器或是與系統的其他部份整合都非常容易，若要升級伺服器，也不會影響到系統其他的部份。

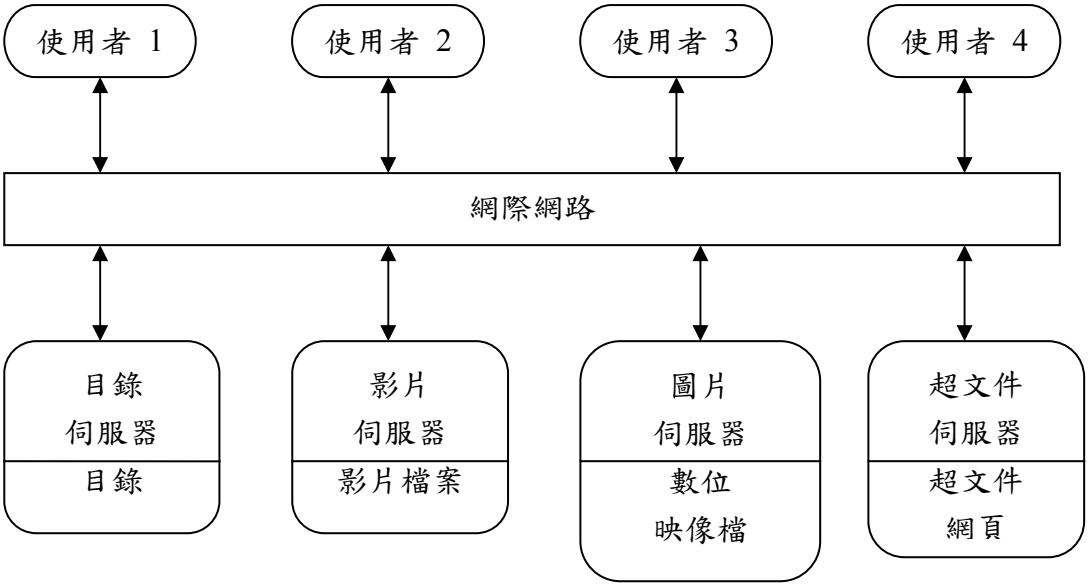

圖 12　利用主從式模型建構的系統範例-影片與圖片之系統（Thayer, 1997）

2.3.3　抽象機器模型

抽象機器模型亦稱作為分層式模型（Sommerville, 2006），它可以用來建立子系統間的介面，並且將一個系統分解成許多分層，讓每一層都提供一些服務；而每一層則定義成一個抽象機器。

使用這種系統架構的開發時，分層中的某一些服務可以讓使用者使用。如果分層有介面，則每一個分層都可以被另一個分層所取代；當分層的介面有所變更的時候，只有鄰近的分層會受到影響。由於抽象機器模型的架構系統可以將機器之間的相依性侷限在分層中的內層，所以可以在比較低階的電腦上進行實作。圖 13為一個版本管理系統利用此模型的範例。

抽象機器模型的缺點就是建構系統比較困難（Biswas and Tseng, 1998），因為所有的抽象機器所需要的一些基本功能必須由內層提供，如此一來，使用者所需要的服務就必須透過好幾層的存取才能到達抽象機器，當外層與它相鄰內層之間沒有簡單的關係時，就很難使用這種架構模式。

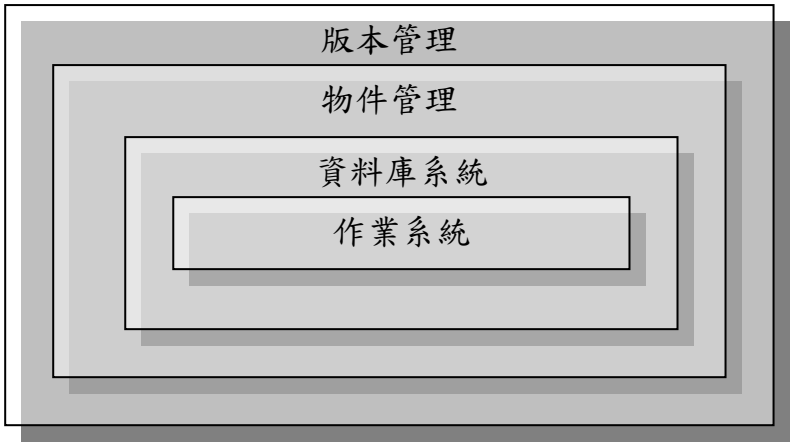

圖 13　版本管理系統的抽象機器模型（Sommerville, 2006）

2.3.4　小結

不同的軟體工程師會採用不同的架構設計，這些設計會根據軟體工程師對應用程式的瞭解和熟悉程度而有所不同。然而，大部分的大型系統架構並不只符合單一模型，系統中不同部份的地方可能會使用不同的架構模型來設計。因此有些系統的架構可能是一個合成架構，它是由各種不同的架構組合而成。軟體工程師必須找出最適合的模型，然後依照問題的需求加以修改。

在 2.3.3 節中，共提到三種軟體架構設計的方式。抽象機器模型雖然可以有效的保護系統，使其不易受到不明使用者的破壞，但由於其建構系統的方式困難且費時，較不適用於本研究所要發展的軟體；而倉庫模型雖然管理容易，但其執行效能不佳，且若要改變其架構，必須花費具大成本。相較於上述兩種架構設計，主從式模型克服了倉庫模型的缺點及保留其優點，在軟體架構設計上容易執行；且目前許多大型的線上遊戲皆是採用這種架構（曾華堃，2005）。因此本研究決定採用主從式模型作為架構遊戲的方式。

第三章　遊戲系統分析與設計

本研究主要目的在於擴充與改善張永佳(2006)所建構之SIMPLE教學輔助遊戲的功能，使得該遊戲中之供應鏈模式內的角色數（即階層數）能由管理者彈性調整，並讓遊戲參與者能透過網路及不同語系的遊戲介面進行連線互動遊戲。除此之外，亦另行設計遊戲管理之功能，分隔出各種使用者型態並賦予不同權限，以方便監控及管理遊戲。以下將張永佳所開發的遊戲稱為SIMPLE 1.0，而本研究所建構之遊戲軟體稱為SIMPLE 2.0。本研究採用漸增模型作為軟體生命週期模型來發展SIMPLE 2.0 並為SIMPLE 2.0 規劃四個軟體增量，依序為「遊戲系統之設計（第一增量）」、「遊戲網路化（第二增量）」、「遊戲管理之設計（第三增量）」以及「資料庫設計（第四增量）」，並依規劃期程陸續發展各增量內容，如圖 14所示。

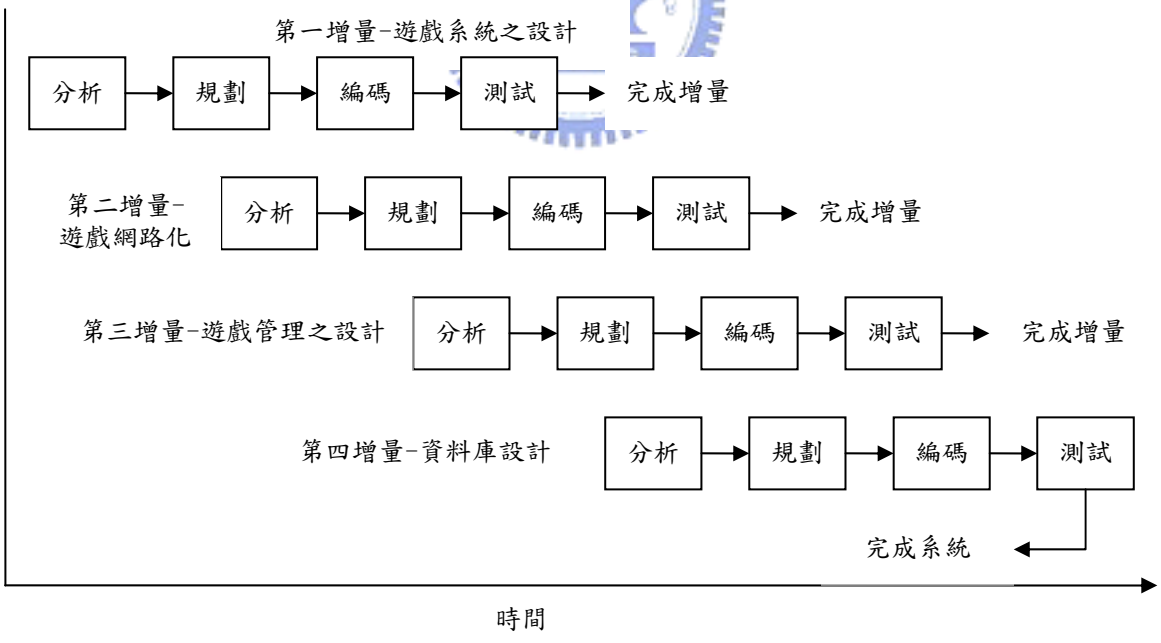

圖 14　SIMPLE 2.0 使用漸增模型之軟體增量圖

以下之 3.1 節說明第一增量-遊戲系統之設計，而第二增量-遊戲網路化在 3.2 節中做詳細的介紹；在完成第一與第二增量模組的開發之後，3.3

節說明第三增量-遊戲管理介面，而於 3.4 節中介紹第四增量-資料庫設計。

3.1 第一增量 – 遊戲系統之設計

本研究將遊戲系統分成兩個部份進行設計，第一個部份的設計為遊戲中之供應鏈模式內的角色數能夠彈性的調整，第二個部份為遊戲多語化介面的設計。

（一）彈性調整遊戲供應鏈角色數

SIMPLE 1.0 的「供應鏈模式」由四種角色所組成，分別為供應商、製造商、物流中心及零售商，各角色位處不同階層而形成一條供應鏈。而在這四個角色中，除了製造商的角色之外，其他三個角色所執行的動作及其各項遊戲數值的計算方式皆相同。因此，本研究將遊戲的角色分成兩類：第一類為一般角色，它所具有的操作方式只限於決定每回合（週）訂購量以及查詢遊戲歷史資料等功能；而第二類為生產角色，它除了擁有一般角色的功能之外，並且可以生產、加班及外包方式調整其產能。

SIMPLE 1.0 是以物件導向(object-oriented)概念的方式設計。由於物件導向設計是將軟體系統視為具有它們私有狀態的互動式物件的集合而不是像函式集合共享整體(Sommerville, 2006)，而且每個物件的狀態及操作方法都是獨立封裝，因此不論在系統的維護上或是在改變物件的實作上都更加容易。為了可以有效率地完成 SIMPLE 2.0 系統，本研究決定採用 Java 作為開發遊戲的程式語言。由於 Java 具備了橫跨各平台(platform independent)的能力，能在各種不同型態的電腦、作業系統開發並執行，且 Java 是純物件導向的程式語言，因此本研究可以延續 SIMPLE 1.0 概念設計利用物件導向來完成 SIMPLE 2.0。

將物件導向設計的觀點套入在開發SIMPLE 2.0 中，遊戲模式中的每一個角色即可視為一個物件，而這些物件皆是較大類別(class)物件的成員(member)之一。圖 15中為SIMPLE 2.0 類別階層概念圖，其中最基本的類

別物件為遊戲角色，在這個類別之下的物件成員則有一般角色以及生產角色兩個物件。

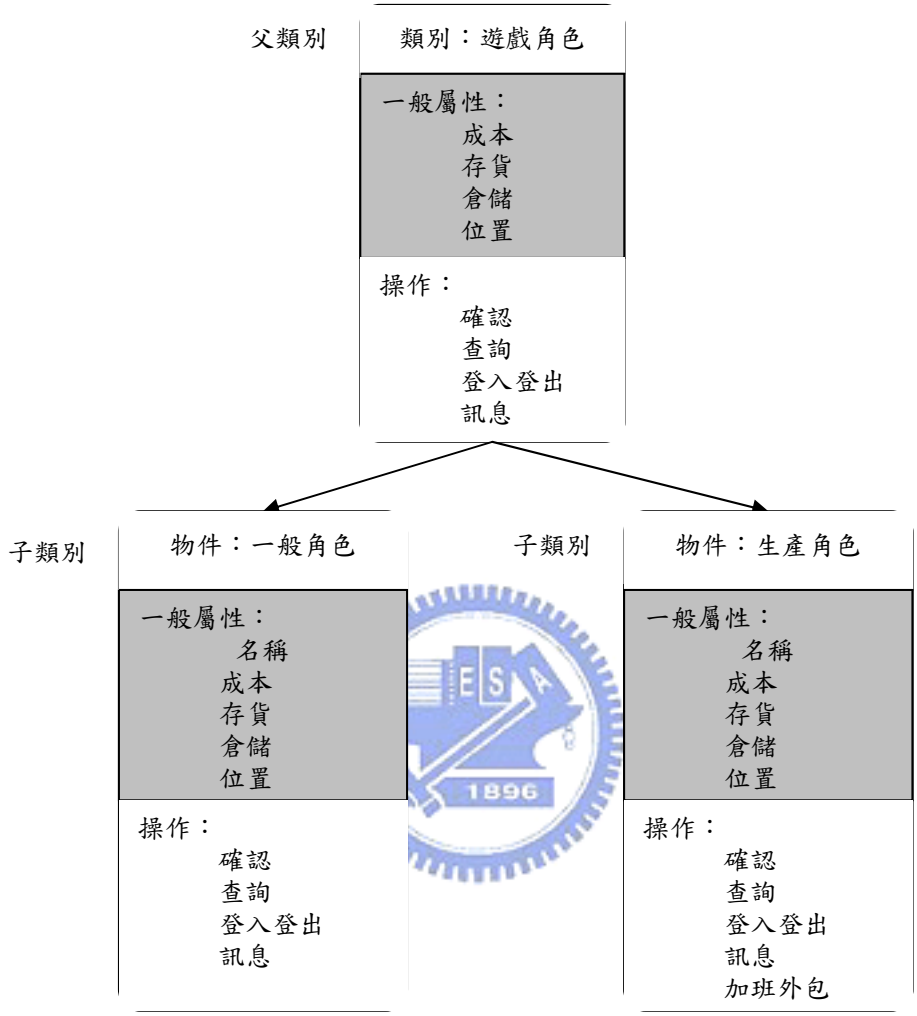

圖 15　SIMPLE 2.0 類別階層概念圖

物件導向設計的另一個主要特性為繼承(inheritance)，其最大的優點在於可提高程式碼的再用性(reuse)，以加快程式發展的速度並減少程式的錯誤。利用繼承的特性，圖 15中之一般角色物件與生產角色物件為遊戲角色的子類別，而使得遊戲角色類別成為其父類別。子類別會繼承父類別所有的屬性及操作，這也表示原本在遊戲角色的資料結構及其演算法操作可完全適用於一般角色和生產角色這兩個物件上。除此之外，任何在父類別(遊戲角色)中屬性或是操作的修改同時會影響其子類別(一般角色及生產角

色)；但對於子類別的任何修改卻不會影響其父類別。因此，在SIMPLE 2.0的設計上，本研究將遊戲角色定義成基本單元(basic unit)，一般角色類別物件完全繼承遊戲角色；而生產角色除了繼承遊戲角色的屬性及操作之外，另加入「加班外包」的操作(如圖 16所示)。根據此設計，SIMPLE 2.0在供應鏈模式下可以由物件繼承的方式產生新的角色，而角色的數量則不受限制(如圖 16所示)，藉此達到彈性調整供應鏈角色數量的目的。

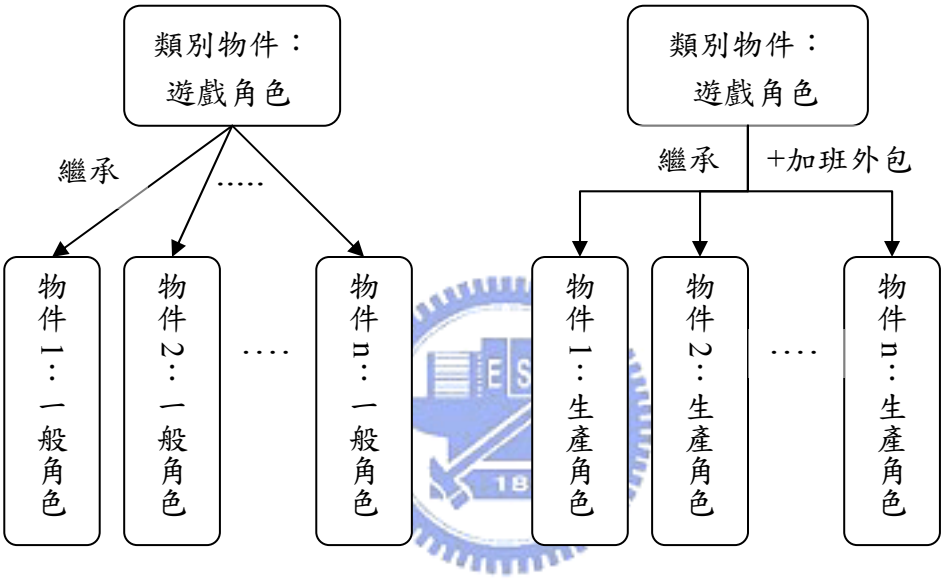

圖 16　SIMPLE 2.0 之物件繼承關係圖

雖然完成彈性調整角色數量的設計，但每個角色的聯繫(物件之間訊息的傳遞)也是相當重要的，而這個動作是由物件的操作執行去完成的。物件之間的互動描述如圖 17所示是透過操作彼此交換訊息完成的；而在物件之間的訊息傳遞格式如下：

message：[destination, operation, parameters]

destination 是用來定義訊息的接收端物件，operation 則是用來接收訊息的方法，parameters 則是要讓操作成功所需提供的資訊。透過這樣的方式，訊息和物件導向系統才可以緊密結合。

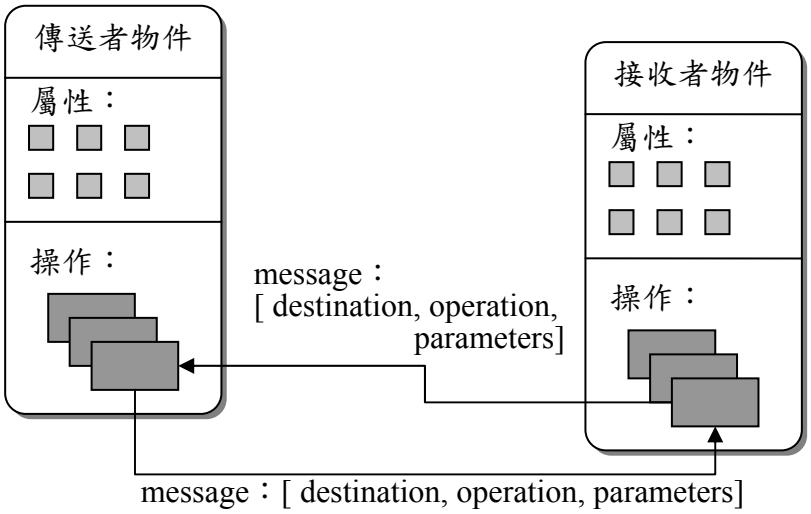

圖 17　物件之間訊息傳送

（二）遊戲多語化介面

SIMPLE 1.0 的遊戲介面僅為中文語系，對於非中文語系國家的人來說，若想要執行本研究所建構的遊戲會有語言上的困難。為了推廣本研究的遊戲系統至各個國內與國際學校的相關科系，因此建立多語化環境的系統。

雖然遊戲參與者可以選擇扮演一般角色或是生產角色進行遊戲，但其遊戲介面顯示的資訊皆無太大差異，因此本研究在基本物件(遊戲角色)的新增語言選擇的操作，如圖 18所示。在這項操作中，本研究會先建立語言特性檔(properties)，在這檔案中會定義各項遊戲資訊的顯示文字；之後透過物件的操作選擇不同的語言特性檔進而完成多語化介面的設計。

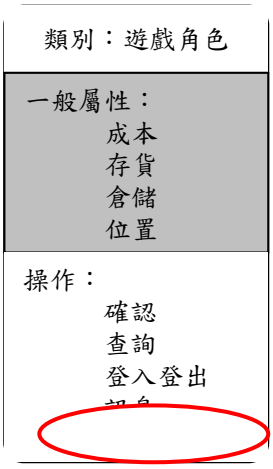

圖 18　基本物件之操作新增圖

3.2 第二增量 – 遊戲網路化

在完成第一增量的規劃之後，接著規劃第二增量模組-遊戲網路化。在各種軟體架構中，由於主從式模型是屬於分散式架構（Christensen, 2002），此種以網路連結的系統可以有效的利用許多分散的處理器來執行，若要加入一個新的伺服器或是與系統的其他部份整合都非常容易，在升級伺服器端設備時，也不會影響到系統的其他部份。除此之外，主從式模型在軟體的架構設計上容易執行，目前許多大型的線上遊戲皆是採用這種架構（曾華堃，2005），因此本研究決定採用主從式模型作為架構遊戲的方式。

SIMPLE 2.0 在主從式模型的架構下，是以伺服器（server）當作連線的中心，而其他透過網路連接至伺服器的電腦則稱為客戶（client）。本研究將依主從式模型將SIMPLE 2.0 的遊戲系統程式與資料庫放在伺服器中，在遊戲進行期間玩家的各電腦都必須連接至伺服器並以伺服器作為連線的中心，其架構如圖 19所示。

在主從式模型的架構下，伺服器端不會主動發送資料給客戶端，只有在客戶端對伺服器端發出請求後，伺服器端才會傳送該客戶端所需要的資料；而當伺服器端未收到客戶端請求時，客戶端與伺服器端之間的連線是中斷的。因此所有資料的傳送都需要由伺服器端發送，且所有的資料處理及計算皆在伺服器端。使用這種網路技術的好處是伺服器與客戶端之間不必經常保持連線，所以可以大幅度減少網路的負荷量；除此之外，因為資料皆需透過伺服器接收與傳送，所有的資料皆儲存在伺服器端，故資料存取與管理相當容易。

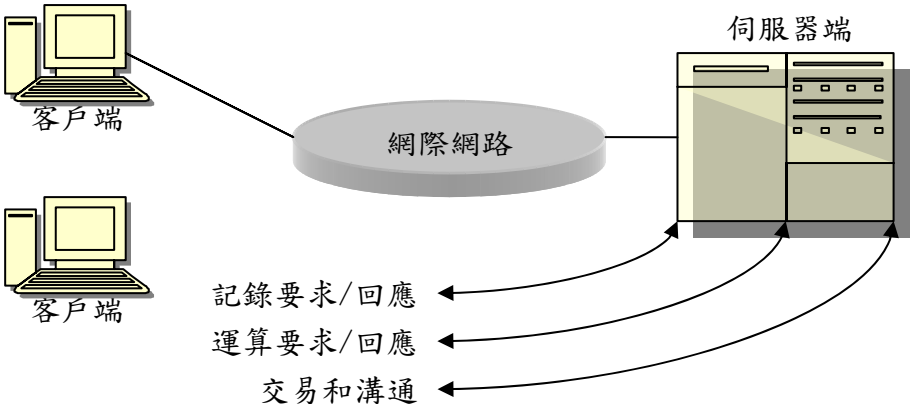

圖 19　SIMPLE 2.0 主從式架構圖

在本文3.1.節中討論到第一增量是用物件導向的方式設計，其中每一個遊戲角色是由客戶端的電腦執行，且每個遊戲角色即是一個物件，而物件之間的互動(包括與伺服器互動)則是透過網路來實行。物件要求仲介者(object request broker, ORB)是一種中間設備元件，它能夠讓在客戶端的物件送訊息給伺服器端的物件。基本上，ORB會把訊息攔截下來，並處理所有通訊及協調活動，以便找出訊息中所指的物件，再找出其操作方法，把適當的資料傳給所指的物件，最後再把結果資料傳回原來產生訊息的物件。

結合第一增量（遊戲系統之設計）與第二增量（遊戲網路化）的網路連線架構如圖 20所示。當客戶端向伺服器端發出請求時，系統便透過介面描述語言(interface description language, IDL)以建立客戶端與伺服器端的IDL存根(stubs)，成為在主從式系統中物件間傳送的通道。而介面描述語言可以定義伺服器端與客戶端的物件類別、物件屬性、物件操作以及利用訊息啟動物件的方式。由於物件透過網路的要求是發生在遊戲執行期間，因此需要透過介面貯存庫(interface repository)儲存物件間的資訊，以便客戶端或是伺服器可以隨時取得適當的資訊。

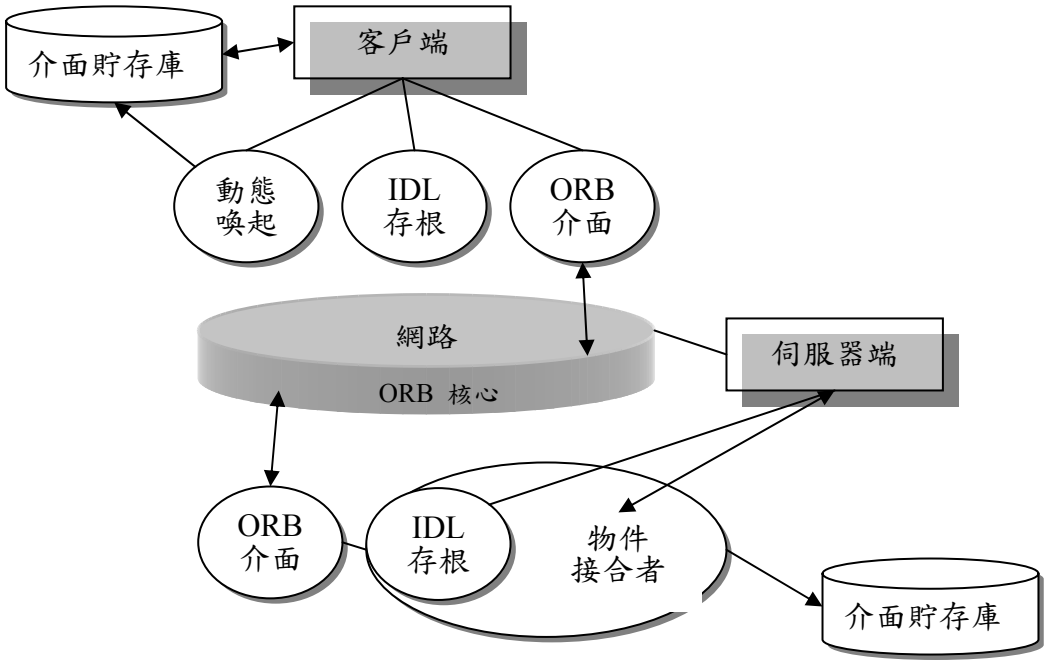

圖 20　SIMPLE 2.0 網路連線架構圖

當客戶端應用程式必須啟用系統中某個物件時，便用動態喚起，以便從介面貯存庫中取得想要採用的方法及其相關的資訊，接著建立資料結構和傳送到物件的參數，最後透過網路及 ORB 核心來完成傳送。而在伺服器端，物件接合者(object adapter)會在伺服器的介面貯存庫中儲存類別及物件資訊，接受並管理由客戶端發送的需求，並執行物件管理的功能。

在完成上述的架構設計之後，本研究以 JWS(Java web start)的方式呈現遊戲。JWS 是在 Java 下執行，它提供一個平台獨立且安全的應用程式封裝技術，使得 SIMPLE 2.0 可以利用 Java 程式撰寫後經過封裝呈現在遊戲參與者面前；此外，JWS 在遊戲參與者每次啟動 SIMPLE 2.0 時便會重新檢查伺服器，比較伺服器中的遊戲版本與客戶端版本的差異，在版本不同時自動下載最新版本。如此一來，在 SIMPLE 2.0 有任何版本異動時，遊戲參與者不會因程式版本不同而不能執行遊戲。

3.3 第三增量 - 遊戲管理之設計

在完成第一與第二增量的設計之後，接著設計第三增量-遊戲管理之設計。如圖 21中所圈選的部份所示，SIMPLE 1.0 的遊戲設定是與遊戲介面放在一起的，在遊戲設定部份也只限於策略調整與角色選擇，其他如各項的單位成本設定、市場需求的設定、玩家人數的控制等參數直接設計在系統中，使用者無法修改與控制。對於想要將SIMPLE 1.0 應用在教學上的教師來說，最大的不便在於無法掌控遊戲的參數設定，而必須由遊戲參與者自行控制。有鑑於此，本研究將遊戲設定的部份與遊戲介面分開，另外設置遊戲參數設定的網頁以方便修改SIMPLE 2.0 的設定，並增加可更動的遊戲參數，使遊戲更具彈性。除此之外，本研究另外設計了遊戲管理的功能，使得遊戲參與者必須先透過網頁註冊後才能進行遊戲，使得系統能追蹤並監控遊戲參與者，以方便系統的維護與管理。

圖 21　SIMPLE 1.0 的遊戲介面與遊戲設定圖

在遊戲管理方面，本研究將 SIMPLE 2.0 的使用者依使用權限分為三個階層，由高至低分別為系統管理人員、遊戲管理人員及一般參與者。其中系統管理人員擁有最高的使用權限，可執行的動作有以下幾項：

(1) 增加/刪除遊戲參與者的資料：為了避免有心人士在遊戲進行中惡意破壞，造成系統的不穩定，因此系統管理人員擁有直接刪除資料庫中遊戲參與者資料的權限，以保護系統穩定進行。

(2) 開放/關閉 SIMPLE 2.0 系統：系統管理人員擁有關閉 SIMPLE 2.0 系統的權限，以方便進行 SIMPLE 2.0 的維護工作。

(3) 存取資料庫中的遊戲數據：系統管理人員擁有存取資料庫資料的權限，可依需要取出遊戲進行後的數據，以分析各玩家的行為與決策模式，或利用此資料作更深入的研究。

(4) 設定遊戲情境：系統管理人員可以直接修改遊戲的參數設定。

(5) 增加/刪除遊戲管理人員的資料：統管理人員可以開放部份權限給遊戲管理人員，供其自行設計所需要的遊戲情境。

遊戲管理人員對系統的權限等級僅次於系統管理者，其可以執行的動作如下：

(1) 觀察線上遊戲參與者的狀態：遊戲管理人員可以觀察線上遊戲參與者的狀態，以方便掌握遊戲的進行。

(2) 設定遊戲情境：遊戲管理人員可以透過遊戲參數的修改，設計其所需要的遊戲情境。

至於一般參與者所開放的權限等級最低，可以選擇參與由遊戲管理人員所設計的遊戲情境進行遊戲。

3.4 第四增量 - 資料庫設計

資料模型(data model)是指一組工具，可用於描述資料庫的架構（蔣定安，2000）。而所謂資料庫的架構是指資料庫中各種不同的資料類型(data type)和資料及資料間各種不同的關係(relationships)。在眾多資料模型中，由於關聯式資料庫最廣泛的被應用且容易設計(Hoffer and Prescott, 2003)，因此本研究決定採用關聯式資料模型。

3.4.1 關聯式資料模型的結構

簡單來說，關聯式資料模型是用一群表格(table)來代表一個關聯式資料庫，而這種表格稱之為關聯表。資料庫中的每個關聯表都有唯一的名稱，表 3為一關聯表名稱為玩家的例子。在每一個關聯表中，每一列皆代表一個實體(entity)，在每一行則代表一個屬性(attribute)(如表 3中的學號、姓名和電子信箱)，關聯表中的每個屬性都有唯一的名稱；而每一個屬性皆有其特定的定義域。

表 3　玩家關聯表

玩家	學號	姓名	電子信箱
	U123456789	王小明	a@yahoo.com.tw
	U987654321	林小美	b@yahoo.com.tw
	U111122222	陳小華	c@yahoo.com.tw

所謂實體可分為實際存在的物體和具有抽象觀念的個體（Dayal and Chen, 1995）。舉例來說，對於一個姓名為王小明而學號是 U123456789 的學生來說，他就是一個實際存在的實體；對於一門課號為 ABC-123 的供應鏈管理課程來說，它就是一個抽象的實體。而屬性就是描述該實體的特性。以王小明的例子來說，他是一個實體玩家，具有姓名及學號兩個屬性；因此我們可以藉由姓名及學號描述該實體的特性。

在關聯式資料庫模型中，鍵(key)的觀念是相當重要的。鍵有很多種類，不過由於在本研究中，主要用到的鍵為主鍵(primary key, pk)和外鍵

(foreign key, fk)。主鍵是一個實體中的一個屬性，此屬性可以區別出關聯表中的每一列，而每一個實體只能存在「唯一」的主鍵。在關聯表中，本研究以底線表示出該關聯表中當作主鍵的屬性，如在表 3中，學號就是該關聯表中的主鍵。而外鍵就是各實體可以透過它建立起彼此的關係性。

3.4.2 設計資料庫

為了完成 SIMPLE 2.0 資料庫的建構，本研究透過正規化(normalization)的步驟來決定哪些屬性應該放在同一個關聯表中，如此可以將關聯表分解成較小且有良好結構之關聯表。

SIMPLE 2.0 資料庫的正規化總共有四個步驟要執行，其執行步驟如下：

Step 1：以表格形式呈現

在執行資料庫正規化之前，先要將資料利用單一的關聯表呈現，如表4所示。

表 4　初始關聯表

遊戲編號	人數設定	成本設定	設定編號	帳號	玩家資料	歷史資料
1021	3	A-1	I	Injon	Type-1	3
1021	3	A-1	I	Ahhway	Type-2	2
1021	3	A-1	I	Alumi	Type-3	4
1022	2	B-1	II	Tom	Type-7	1
1022	2	B-1	II	Yuri	Type-8	5

Step 2：轉換成第一正規化形式(first normal form, 1NF)

當關聯表中沒有重複群組，且每個關聯表都有可以辨識的主鍵時，則完成第一正規化的步驟。因此，我們從表 4中，找出三個決定性的屬性，分別為「帳號」、「設定編號」與「遊戲編號」，其功能相依性如下：

帳號 → 遊戲編號，玩家資料，歷史資料

設定編號 → 成本設定，人數設定

遊戲編號 → 成本設定，人數設定，設定編號

圖 22為根據上述決定性屬性帳號、設定編號、遊戲編號他們的功能相依性所繪製的功能相依圖。

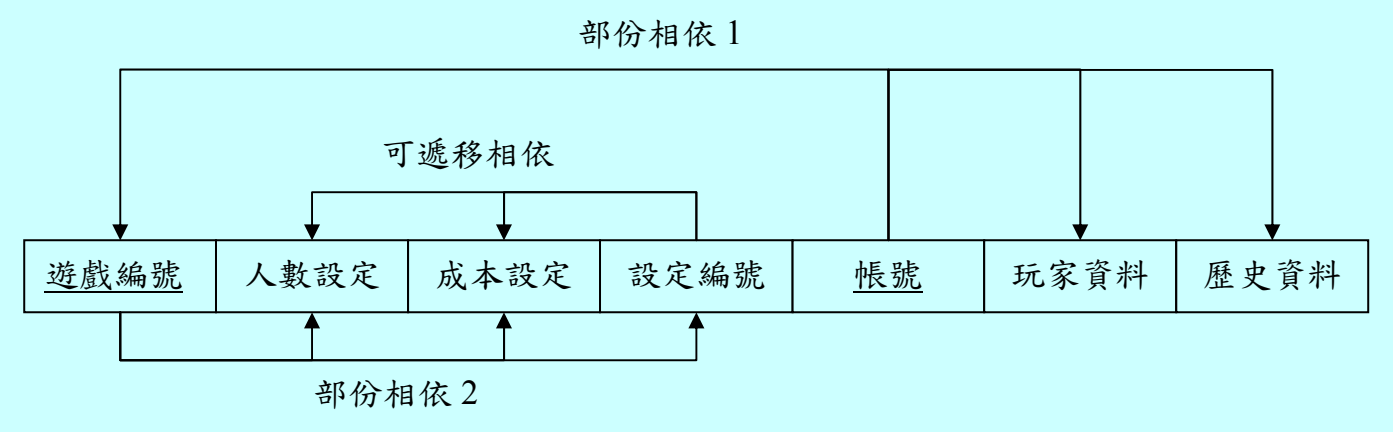

圖 22　功能相依圖

Step 3：轉換成第二正規化形式(second normal form, 2NF)

當關聯表符合第一正規化形式，且屬性間無部分相依性時，就完成第二正規化的步驟。在圖 22中有兩個部份相依，分別是由「帳號、玩家資料、歷史資料及遊戲編號」所構成的部分相依 1 及「遊戲編號、人數設定、成本設定及設定編號」所構成的部份相依 2，將這兩個部分相依的部份從原本的關聯表中分開後則產生圖 23中兩個新的關聯表，其名稱分別為「帳戶」以及「遊戲」。

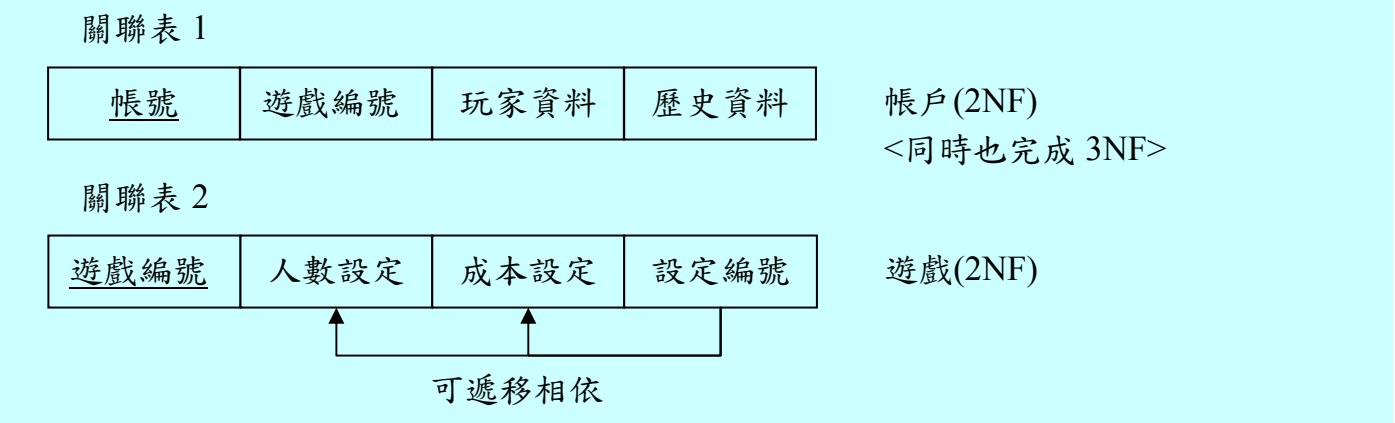

圖 23　移除部份相依

Step 4：轉換成第三正規化形式(third normal form, 3NF)

如果關聯表符合第二正規化形式，且屬性間無可遞移相依性時，就完成第三正規化的步驟。在圖 23中，帳戶這個關聯表在完成第二正規化的同時，因為沒有可遞移相依部份，同時完成第三正規化形式；而關聯表 2 中仍有可遞移相依，因此將遊戲這個關聯表的可遞移相依部份從關聯表 2 中分開產生新的關聯表，並將此關聯表命名為「編號」，其結果如圖 24所示。

<u>帳號</u>	遊戲編號	玩家資料	歷史資料	帳戶(3NF)

<u>遊戲編號</u>	設定編號	遊戲(3NF)

<u>設定編號</u>	人數設定	成本設定	編號(3NF)

圖 24　移除可遞移相依部份

完成上述四個步驟之後，則合併圖 24中的三個關聯表；由於透過帳戶關聯表中的遊戲編號可以找到遊戲關聯表中的設定編號，透過遊戲關聯表中的設定編號可以找到編號關聯表中的人數設定及成本設定，因此帳戶關聯表中的遊戲編號和遊戲關聯表中的設定編號為外鍵，最後產生的資料關聯式綱要如圖 25所示。

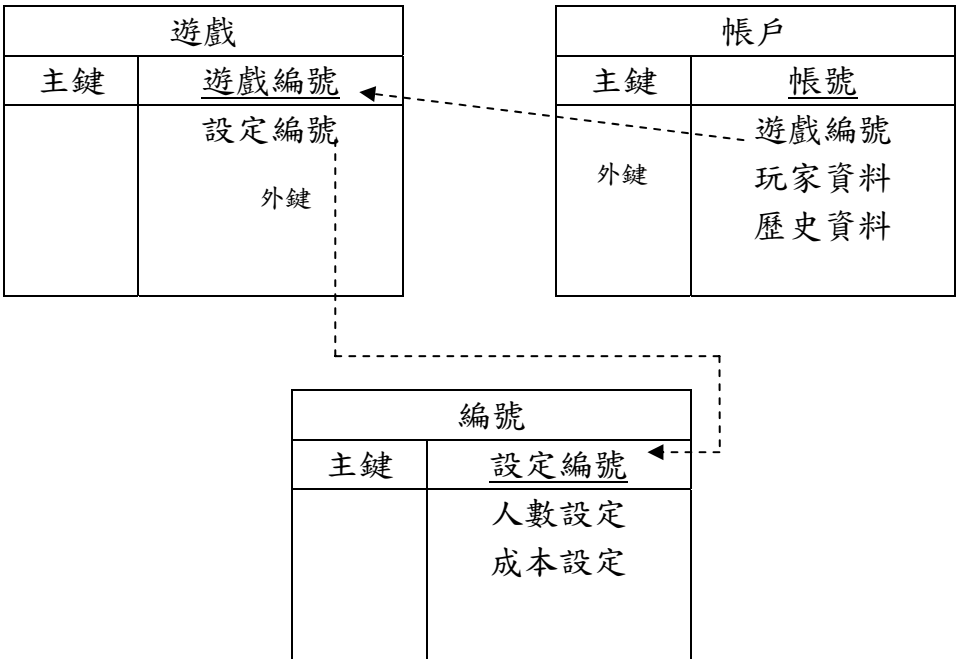

圖 25　資料的關聯式綱要

第四章　遊戲系統之實作

本研究採用由 SUN Microsystems 公司所發展出來的 Java 程式語言作為開發 SIMPLE 2.0 系統的工具，並利用 JSP(JavaServer Page)以及 HSQL (hypersonic structure query language)分別建構遊戲管理網頁以及系統資料庫。在本章中將介紹本研究所發展出之 SIMPLE 2.0 的系統以及其實作方式。由於本研究將系統使用權限分成三個層級，因此前三節將依照層級的高低依序介紹 SIMPLE 2.0 系統。

4.1 系統管理頁面

系統管理人員對系統擁有最高的權限,除了可以直接設定 SIMPLE 2.0 的遊戲參數，並可操作遊戲資料庫。在本節中先介紹遊戲資料庫的部份，而於 4.2 節中詳細說明遊戲參數設定的部份。在 4.1.1 節中說明本研究使用 HSQL 的原因以及資料庫連線實作；在 4.1.2 節中則展示系統管理頁面。

4.1.1 系統資料庫之實作

本研究採用 HSQL 建構系統的資料庫。HSQL 具有以下的特點：

(1) HSQL 是純 Java 所開發的資料庫，可以完全支援 Java 所架構的軟體。

(2) HSQL 可以透過 JDBC driver 來存取資料。

(3) HSQL 支援 ANSI-92 標準的 SQL 語法，因此若要將資料移植至其他資料庫則相當容易。

(4) HSQL 為免費的開放式程式碼，因此取得相當容易。

由於本研究所建構的系統是採用Java程式語言所撰寫，因此由純Java所開發的HSQL與本系統的相容度極高。為了將HSQL透過JDBC driver進行連線，本研究撰寫了連線的程式，其程式碼如圖 26所示：

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class HsqldbJdbcLink {
    public static void main(String[] args) {
  // load the jdbc driver
  try {
      Class.forName("org.hsqldb.jdbcDriver");
  } catch (Exception e) {
      System.err.println("ERROR: failed to load HSQLDB JDBC driver.");
      return;
  }
  // connect to database
  try {
      Connection c = DriverManager.getConnection(
        "jdbc:hsqldb:hsql://localhost/mydbaliasname", "sa", "");
  } catch (SQLException e) {
      System.err
        .println("ERROR: failed to connect database, please check the database alias na
me");
  }
    }
}
```

圖 26　JDBC 連線程式碼

由圖 26的程式碼可以發現，HSQL的JDBC連線方式與SQL的連線方式差異不大，因此以後系統若要移植至資料庫也相當容易。

4.1.2 系統管理頁面之展示

系統管理人員透過圖 27中的系統管理頁面操作與維護系統，其中頁面左邊的框架中有七個連結，分別為「線上狀態」、「基本設定」、「成本設定」、「玩家設定」、「玩家資料」、「遊戲紀錄報表」及「清除遊戲紀錄」。由於前面四個連結屬於遊戲參數設定部份，因此在本節中只針對玩家資料、遊

戲紀錄報表及清除遊戲紀錄作說明。

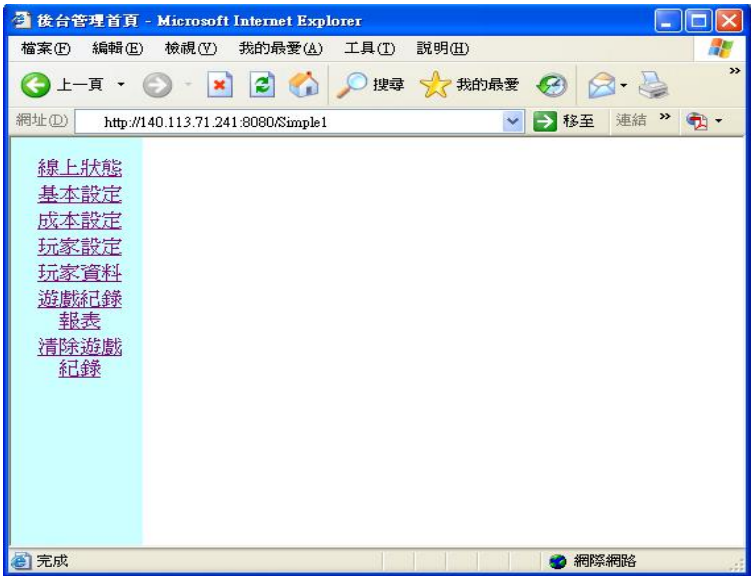

圖 27　系統管理頁面

（一）玩家資料

圖 28為玩家資料的頁面，此頁面會列出目前在資料庫中已經註冊帳號的使用者資料，而系統管理人員可以增加或刪除使用者的資料。

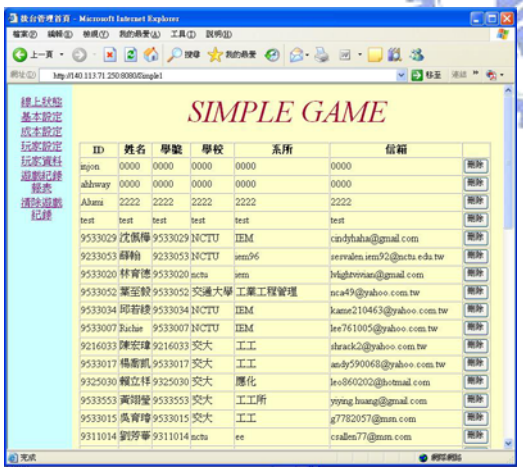

圖 28　玩家資料頁面

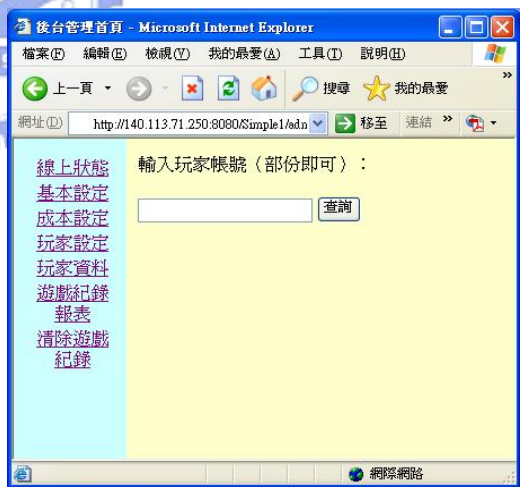

圖 29　資料庫查詢

（二）遊戲紀錄報表

圖 29 為進入資料庫中查詢遊戲紀錄的基本頁面。當系統管理人員進入該頁面時，會先詢問系統管理人員欲查詢的使用者帳號，送出查詢之後系統便會出現圖 30的報表，其中報表會詳細顯示遊戲時間、遊戲的參數設定及遊戲中的各項數據，以便系統管理人員作分析。

圖 30　SIMPLE 2.0 資料庫查詢結果

（三）清除遊戲紀錄

當 SIMPLE 2.0 的遊戲歷史資料均已分析檢視過後，系統管理人員若想清除資料庫中遊戲的歷史數據，便可利用此項連結將資料庫中所有遊戲的數據清空，但這個動作不會連帶清除資料庫中使用者已經註冊過的帳號。

4.2 遊戲管理頁面

在本節的 4.2.1 中說明本研究如何透過 JSP 所建構的網頁修改 SIMPLE 2.0 遊戲中的參數設定；在 4.2.2 節中則展示本研究所建置的遊戲管理網頁。

4.2.1 遊戲管理頁面之實作

JSP頁面是由html程式碼和嵌入其中的Java程式碼所組成。當客戶端透過JSP網頁向伺服器發出請求時，其網頁的JSP程式碼透過網路伺服器(web server)後傳送後進行編譯動作，再透過servlet執行該請求，最後再將結果送

回給客戶端。利用上述方式，本研究先建立一個文件檔案(.txt)，接著將在網頁中的遊戲參數設定傳送至這個文件檔，其網頁傳送的示意圖如圖 31所示，而圖 32則為程式碼。

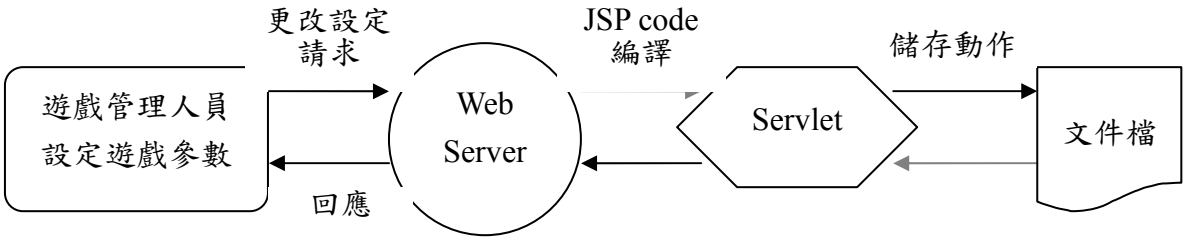

圖 31　SIMPLE 2.0 參數傳送示意圖

```
String path = application.getInitParameter("settingPath");
String originalSetting[] = pt2.util.StringTool.fileToStringArray(path,
"playerParam.txt");
request.setAttribute("originalSetting", originalSetting);
```

圖 32　網頁傳送設定值至文件檔之程式碼

在完成傳送之後，將SIMPLE 2.0 遊戲的讀取參數位置指定該文件檔，其web.xml的設定如圖 33所示：

```
<context-param>
        <description>設定檔的所在目錄</description>
        <param-name>settingPath</param-name>
        <param-value> C:\Program Files\Tomcat5.0 \webapps\Simple\
                playerParam.txt </param-value>
</context-param>
```

圖 33　web.xml 指定路徑之程式碼

如此一來，遊戲設定便可透過網頁進行調整，遊戲管理者透過系統管理人員其權限的開放，可透過網頁進行遊戲參數的設定。

4.2.2 遊戲管理頁面之展示

SIMPLE 2.0 的遊戲管理人員是設計由教師所擔任的，當教師得到系統管理人員所授權的帳號密碼之後，便可登入遊戲管理的網頁，透過遊戲參數的設定營造不同的遊戲模擬情境以因應其課程上的需求，圖 34為遊戲管理者登入後的頁面。在圖 34頁面中的左邊框架裡，有四個連結選項，分別為「線上狀態」、「基本設定」、「成本設定」及「玩家設定」，而右邊框架則顯示左邊框架連結的內容。遊戲管理人員與系統管理人員所登入的頁面最大的不同在於遊戲管理人員不能操作資料庫。

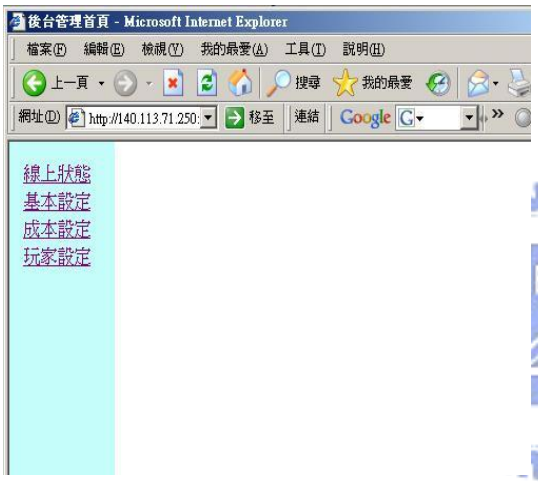

圖 34　遊戲管理頁面

圖 35　遊戲線上狀態

以下分別說明上述四個連結選項：

（一）線上狀態

此項連結是負責顯示目前在線上的玩家列表，以及目前線上有多少局遊戲正在進行和每一局遊戲中玩家的人數，以便遊戲管理者在線上直接觀察遊戲參與者的狀態，如圖 35 所示。

（二）基本設定

圖 36為遊戲基本設定的頁面。在基本設定中，遊戲管理人員可以分別設定遊戲進行的週數、市場的需求量、期出存貨量及在途量、資訊是否透明化、倉儲空間是否有所限制等七項設定。

（三）成本設定

圖 37 為遊戲成本設定的頁面，遊戲管理人員可配合課程需要，透過不同的成本設定而產生不同的遊戲情境。在成本設定中有八項成本參數設定可以進行調整，分別為「存貨成本」、「缺貨成本」、「訂購成本」、「懲罰成本」、「增加額外倉儲成本」、「加班成本」、「外包作業成本」以及「外包產品單位邊際成本」。

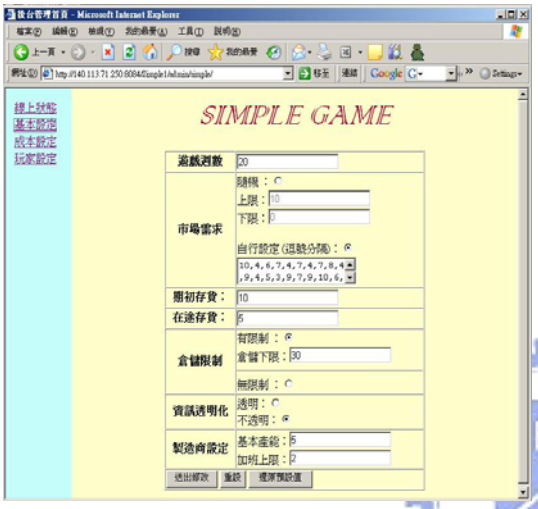

圖 36　遊戲基本設定

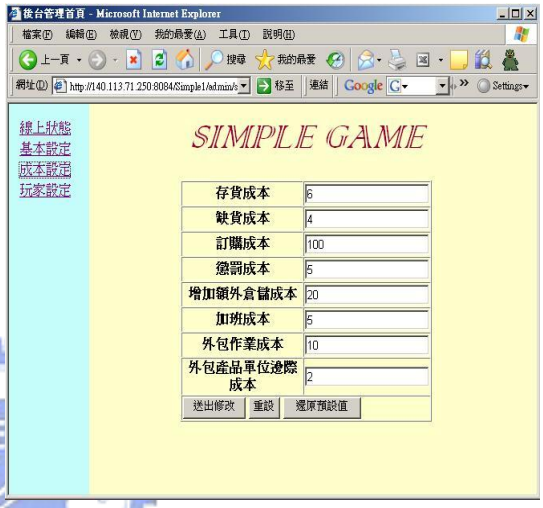

圖 37　遊戲成本設定

（四）玩家設定

圖 38即為遊戲玩家設定的頁面。在玩家設定中除了可以設定供應鏈遊戲的角色數量之外，並且可以設定每個角色為一般角色或是工廠角色，以更符合現實狀況的需求。

圖 38　遊戲玩家設定

4.3 使用者操作頁面

在本節中將分成兩小節作介紹。在 4.3.1.節中將先說明 SIMPLE 2.0 如何利用 Java 將遊戲以 JWS 的方式呈現;在 4.3.2.節中則將展示 SIMPLE 2.0 的遊戲介面。

4.3.1 SIMPLE 2.0 遊戲呈現之實作

SIMPLE 2.0 遊戲的呈現是利用 Java 的 JWS 技術。為了要完成 JWS 實作，本研究先將 SIMPLE 2.0 的完整程式封裝成 jar 檔，並且編輯伺服器的 mime.type 檔案以包含 jnlp 的定義，其 jnlp 的定義如所示：

```
<jnlp spec="1.0+" codebase="http://localhost:8080/Simplegame/">
<information>
<title>Simple Game</title>
<vendor>scmlab</vendor>
<homepage href="none"/>
<description> Simple Program use JWS </description>
</information>
<offline-allowed/>
<security>
<all-permissions/>
                                        </security>

<application-desc main-class="pt2.simple.client.MainApplication">
      <argument>http:// localhost:8080/Simplegame/</argument>
      <argument>tw</argument>
</application-desc>
<resources>
      <j2se version="1.4+" />
      <jar href="jar/Simple.jar"/>
      <jar href="jar/jcommon-1.0.0.jar"/>
      <jar href="jar/jfreechart-1.0.1.jar"/>
</resources>
</jnlp>
```

圖 39　jnlp 定義

其中，codebase 與 application-desc main-class 所指定的位置皆為伺服器 SIMPLE 2.0 的資料夾；而 resources 所指定的 jar 檔，則為經過封裝 SIMPLE 2.0 程式碼。透過此 jnlp 的定義，一般參與者便可在連結伺服器後啟動 JWS 進行遊戲。

4.3.2 SIMPLE 2.0 遊戲之展示

一般參與者所擁有的權限等級最低，其可執行的動作也僅限於執行遊戲。一般參與者要執行遊戲前，必須先連結至圖 40中SIMPLE 2.0 的遊戲網頁註冊遊戲帳號，除此之外，一般參與者的個人電腦中必須有JRE(Java running environment)1.5 以上的版本才可以進行遊戲。

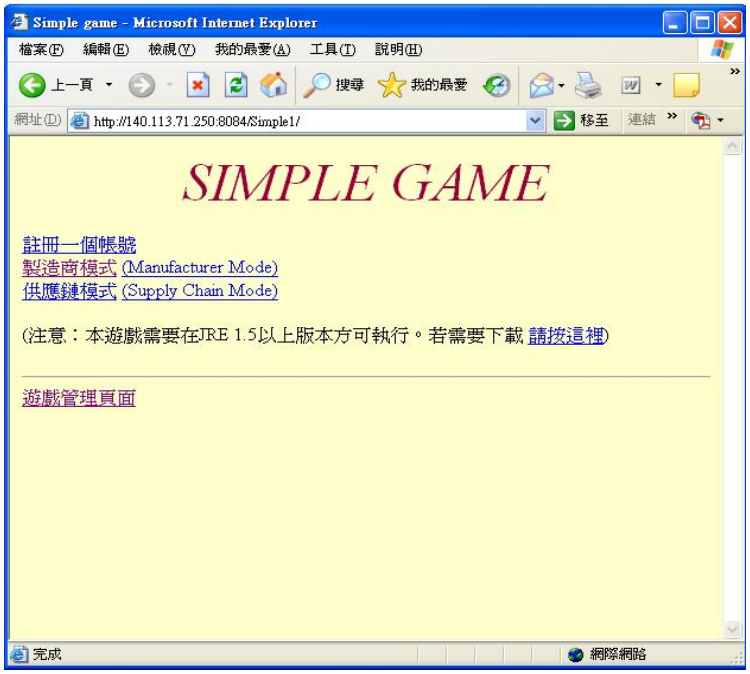

圖 40　SIMPLE 2.0 遊戲網頁

一般參與者註冊之後，便可選擇頁面的兩種遊戲模式中的任一種進行遊戲。由於兩種模式的遊戲介面皆相同，於本文中將以供應鏈模式為範例進行展示。當點選供應鏈模式之後，便會連結伺服器並啟動JWS進行認證，如圖 41所示；認證過後便可透過JWS進行遊戲。

圖 41　JWS 遊戲認證

圖 42為SIMPLE 2.0 的遊戲畫面。本研究利用文字顏色的不同，將遊戲的資訊作明顯的區隔，讓一般參與者的視線可以容易搜尋到自己想要的資訊；除此之外，本研究亦提供燈號顯示，當遊戲參與者於該週尚未做出決策時，屬於該遊戲參與者的燈號會顯示黃色，而當送出決策時，燈號則會變為灰色，以圖 42紅色圈選部份為例，表示供應商已經做出決策，其他遊戲參與者尚未做出決策。

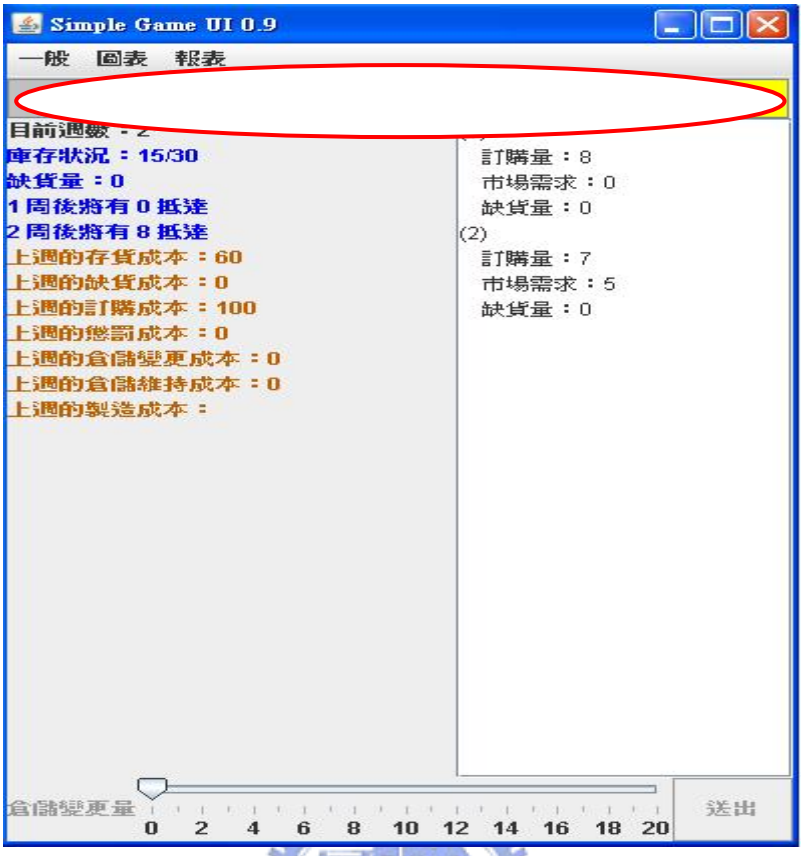

圖 42　SIMPLE 2.0 遊戲畫面(以供應鏈模式為例)

於最後遊戲結束時，如圖 43所示會跳出檢視這回合遊戲歷史資訊的視窗，以供一般參與者在遊戲後的進行檢討分析。

圖 43　遊戲歷史資料檢視視窗(以供應鏈模式為例)

此外，本研究在使用者的操作介面上除了提供中文化介面之外，亦提供英文化介面給各國的遊戲參與者使用；因此在同一場的遊戲當中可以容許不同語系的介面彼此互相連線進行遊戲，圖 44即為兩個不同語系的版本同時進行同一場遊戲。

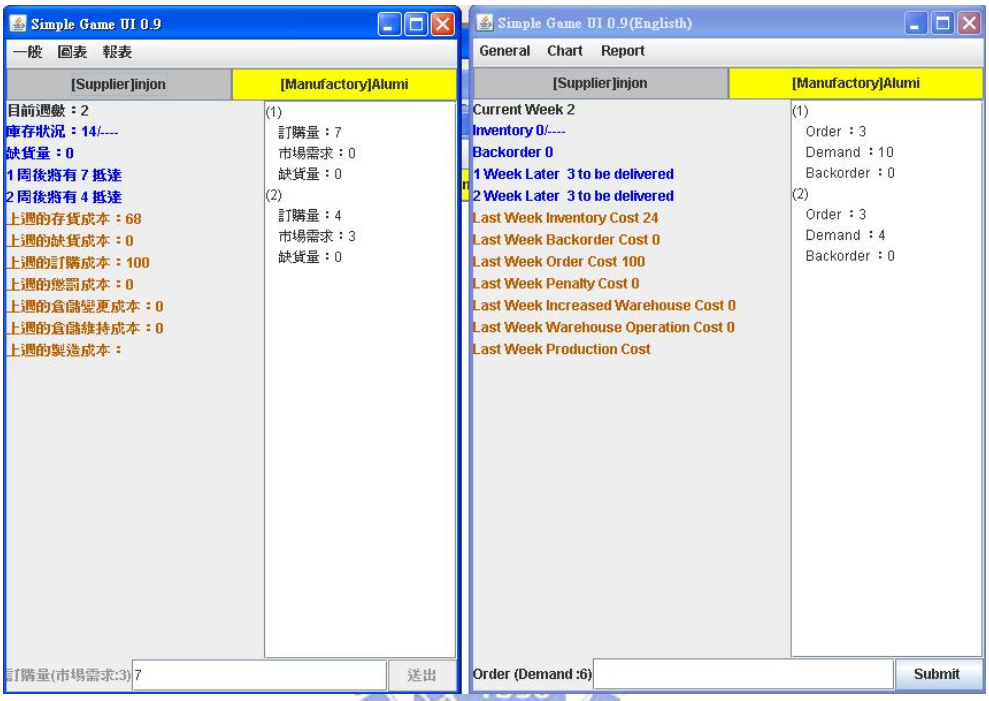

圖 44　不同語系遊戲執行示範

第五章　結論與未來發展方向

在本章的第一節中將作本研究的結論，在第二節中將探討 SIMPLE 2.0 的未來發展方向。

5.1　結論

本研究的目的為擴充及改善 SIMPLE 1.0 的功能，以建構出一個具有彈性的教學輔助軟體，以提供與生產管理、存貨管理及供應鏈管理等相關的課程於教學上使用。因此，本研究以 SIMPLE 1.0 為基礎進行遊戲功能的擴充，利用 Java 物件導向的特性使得遊戲中之供應鏈模式內的角色數能由管理者彈性調整，並讓遊戲參與者能透過網路進行連線互動遊戲；除此之外亦設計遊戲管理之功能，分隔出各種使用者型態並賦予不同權限，以方便監控及管理遊戲；最後再利用 HSQL 建立資料庫使得系統的功能更為完整。本研究成功地發展出 SIMPLE 2.0，取代以往在教授生產管理或供應鏈管理相關課程時，需使用多種輔助遊戲軟體之不便，除了可作為教學輔助工具外，亦可作為個人在自修時的練習教材。

由於實驗操作和實習已經成為各學門(如電機、機械、物理、化學等課程)重要的一環，經由實作可使學生對於該課程的目的和內容有更深的體會；然而在生產與運籌管理中，卻沒有一個完整且可依照教學範圍需要作調整的實習操作工具或模擬環境，而本研究所建構的 SIMPLE 2.0 正好可以滿足此需求。透過 SIMPLE 2.0 遊戲的建立，教師可以藉由遊戲參數的設定，模擬各種在生產與運籌管理時會面對的情境，減少備課的負擔及增加教學的效率，並且經由遊戲中彼此競爭的方式，提昇學生的學習興趣與學習動機，讓學生將課本的理論實際套用在遊戲競賽中，藉此加深對該門課程的理解與應用。

另外，為了推廣 SIMPLE 至各個國內及國外與生產及運籌管理相關科系的學校，本研究提供了遊戲多語化的執行環境，讓各國的教師可以透過

本研究所建構的遊戲模擬環境進行教學，此後可以不再需要花費大量的心思去尋找符合課程及語言需求的教學輔助遊戲，如此一來不但減少教師們在備課上的困難，亦可同時提高其教學效率。而對於各國的遊戲使用者而言，除了不再需要為了課程的需要去適應許多不同遊戲環境的教學輔助軟體之外，同時亦可以透過本研究所建構的遊戲模擬環境作實驗測試，以印證教師於課堂上所教授的理論，藉此提高學習效率。

5.2 未來發展方向

在本研究中，主要的工作在於將 SIMPLE 網路化以及可增加供應鏈角色的數量。而其中供應鏈角色數量調整發展至目前為止，僅就同一層角色中只有一個遊戲參與者擔任，而同層間並沒有競爭者與之競爭。舉例來說，假若目前供應鏈中有兩個角色分別為供應商及零售商，這兩個角色目前都是分別由一位遊戲參與者擔任；但以現實狀況來說，供應商可能不只只有一個，同樣的道理零售商也應該不只一個，在同樣的角色中，應該有其他競爭者相互競爭。

因此，在未來的研究當中，可以利用本遊戲物件導向的架構為基礎，再對系統作擴充，這僅為本研究所提出的一個發展的方向；至於是否還有其他擴充的可能性，這也是值得研究的部份。

參考文獻

姜子龍(2004)。軟體工程。台北市：台灣培生教育出版股份有限公司。

陳湘楊(2004)。軟體工程-物件導向程式設計與UML系統分析實做。台北縣：博碩文化股份有限公司。

張永佳 (2006)。生產與運籌管理遊戲網路平台之構建(1/2)（國科會專題研究計畫期中成果報告，NSC 94-2516-S-009 -001）。新竹市：國立交通大學工業工程與管理學系。

曾華堃(2005)。網路連線角色扮演遊戲產生器的設計與實作。未出版之碩士論文，國立交通大學電機資訊學院資訊組，新竹市。

葉盛昌(2003)。遊戲式數學教學模式對學生數學學習的影響。未出版之碩士論文，台中師範學院教育系，台中市。

蔣定安(2000)。資料庫基本理論與實作。台北市：台灣東華書局股份有限公司。

Aurum, A. (2005). Engineering and managing software requirements. Gubbio: Springer Verlag.

Bass, L. (2003). Software architecture in practice. New York: Addison-Wesley Publishing Company.

Beer game. Retrieved December 9, 2006, from http://beergame.mit.edu/.

Biswas, P. and Tseng , C. C. (1998). LogDf: A data-driven abstract machine model for parallel execution of logic programs. Proceedings of the International Conference on Fifth Generation Computer Systems (pp. 1059-1070). Tokyo.

Boehm, B. W. (1998). A spiral model of software development and enhancement. IEEE Computer, 21(5), 61-72.

Christensen, M. J. (2002). The project manager's guide to software

engineering's best practices. New Jersey: John Wiley and Sons Inc.

Collins, J., Arunachalam, R., Sadeh, N., Eriksson, J., Finne, N., Janson, S. (2006). The supply chain management game for the 2007 trading agent competition. 網址：http://www.sics.se/tac/tac07scmspec.pdf

Dayal, U. and Chen, O. (1995). From database programming to business process programming, in proceedings of the international workshop on database programming languages. Gubbio: Springer Verlag.

Goleman, D.（1995）. Emotional intelligence. New York: Bantam Dell Pub Group.

Group under the IFIP Working Group 5.7 on Integrated Production Management. Retrieved December 13, 2006, from the available protocol: http://iprod.auc.dk/x-proj/gamespm/www-games.html.

Hoffer, J. A. and Prescott, B. M. (2003). Modern database management. New Jersey: Printice Hall.

Jacobson I.(1999). The unified software development process. New York: Addison-Wesley Publishing Company.

Kruchten, P. (2003). The rational unified process: An introduction. New York: Addison-Wesley Publishing Company.

Martocchio, J. J., and Webster, J. (1992). Effect of feedback and cognitive playfulness on performance in microcomputer software training. Personnel Psychology, 45, 553-578.

Pfleeger, S. L. (2001). Software engineering: Theory and practice. New Jersey: Prentice Hall.

Pressman, R. S. (2001). Software engineering: A practitioner's approach. McGraw Hill.

Quinn, C. N. (1996). Designing an instructional game: Reflections on quest for independence. Education and Information Technologies, 1 (3&4), 251-269.

Sommerville, I. (2006). Software engineering. New York: Addison-Wesley Publishing Company.

Sterman, J. D. (1984). Instructions for running the beer distribution game. Retrieved December 12, 2006, from http://www.solonline.org/repository/download/instr.html?item_id=456354

TAC game. Retrieved December 9, 2006, from http://www.sics.se/tac.

Thayer, R. H. (1997). Software Requirements Engineering. New Jersey: John Wiley and Sons Inc.