國立暨南國際大學　資訊管理學系

碩士論文

模擬平台之設計與實作

Design and Implementation of a Simulation System

指導教授：俞 旭 昇 博士

研 究 生：蒲 泰 宏

中 華 民 國 九 十 七 年 六 月

國立暨南國際大學碩（博）士論文考試審定書

資訊管理 學系（研究所）

研究生 蒲泰宏 所提之論文

模擬平台之設計與實作

Design and Implementation of a Simulation System

經本委員會審查，符合碩（博）士學位論文標準。

學位考試委員會

委員兼召集人

委員

委員

委員

委員

中華民國 97 年 06 月 23 日

博碩士論文電子檔案上網授權書

（提供授權人裝訂於紙本論文書名頁之次頁用）

本授權書所授權之論文為授權人在 暨南國際大學 資訊管理學系 _____組 96 學年度第 二 學期取得 碩士 學位之論文。

論文題目： 模擬平台之設計與實作
指導教授： 俞旭昇

茲同意將授權人擁有著作權之上列論文全文（含摘要），非專屬、無償授權國家圖書館及本人畢業學校圖書館，不限地域、時間與次數，以微縮、光碟或其他各種數位化方式將上列論文重製，並得將數位化之上列論文及論文電子檔以上載網路方式，提供讀者基於個人非營利性質之線上檢索、閱覽、下載或列印。

- 讀者基非營利性質之線上檢索、閱覽、下載或列印上列論文，應依著作權法相關規定辦理。

授權人：蒲泰宏
簽　名：蒲泰宏　　中華民國 97 年 07 月 24 日

誌謝

於埔里的三年歲月，一轉眼就過去了，回首從前，多少莘莘學子為了研究所的窄門擠破了頭，放榜時的幾家歡樂幾家愁，以及來到新環境與新生活的忐忑與喜悅，這些景象，彷彿歷歷在目。

「不論人生有多少曲折與轉角，如果我能看的更遠，要感謝老師讓我踩在他的肩膀上～」，對於一個政治系的門外漢，縱使大學曾經修習資管輔系，來到了這裡，才發現資訊領域的浩瀚，並非啃食一兩本教課書即可迅速地消化「應該」具有的基礎知識，有幸能拜師於指導教授俞旭昇老師的門下，在老師的開放風格指導下，可以從多元的面向去探索研究問題與領域，不僅開創了視野，更讓學術研究的過程富饒了趣味與生命力！此外，老師的處事待人更是匡正我的行為與價值觀的學習典範，做事的嚴謹與細膩，將是我在人生另一段旅途的一把尺。對於老師的感謝之意，並非字裡行間可以詳盡，但老師的恩惠，將永存於心！

特別感謝於論文口試期間，承蒙中央大學吳曉光老師、中央大學蔡孟峰老師、台灣大學薛智文老師與暨南大學尹邦嚴老師所給予的精闢指正與寶貴意見，使自己的論文能夠更臻完善，並體認到作研究的嚴謹與問題意識的重要性，在此向您們至上最深的謝意。

本論文可以順利完成，要特別感謝劍霞學姐不厭其煩的指正與教導，更提供了寶貴的生物知識與論文寫作的經驗分享；在埔里的生活，要感謝李大哥、哲嘉學長、佑寧學長以及小玫學姐，謝謝你們的鼓勵與支持，讓我更有信心完成學業。此外，感謝同窗好友阿暐、繼昱、克文、茂楠、凱育及博盛等同學的鼓勵；也要感謝Lab B11的所有成員：智泉、明勳、洛豪、劭睿、益民、書楠、克威以及孝萱，有你們的參與讓我的碩士生活更加精彩！在生命中，還有許多幫助過的貴人，也許無法一一載述于此，但是對於曾經給予的溫暖，我會僅記在心，也由於您們的幫助，使我在學術的道路上，步伐更穩健。

最後，要誠摯地感謝所有疼愛我的家人：疼愛我的大姑婆，在我論文完成的期間，您已經辭世，但這份喜悅想與您分享；感謝我的父母，感謝您們一直以來對我的鼓勵、支持與無怨無悔的照顧，讓我能更專心的完成學業；感謝大舅、六姑、咸宜表姐，以及其他疼愛我的家人。此外，還要感謝小蓉，在學業上互相鼓勵與打氣，雖然我們只能在所謂「國家公園」的附屬圖書館，埋首於堆積如山的 papers 與書堆中，不停的敲打鍵盤，但最後也順利度過這漫長的碩士生涯，在此祝福妳未來在中央的博班生涯，坦途一片光明，能夠更順心如意！

蒲泰宏　謹誌

于　暨南大學管理學院　B11 研究室

民國九十七年七月

論文名稱：模擬平台之設計與實作

校院系：國立暨南國際大學資訊管理研究所 頁數：65

畢業時間：中華民國 97 年 6 月 學位別：碩士

研究生：蒲泰宏 指導教授： 俞旭昇博士

摘要

由於現實生活中存在著許多事物侷限於時間、金錢或無法實際實驗，因此必須借助模擬來研究。模擬乃為藉由電腦模仿真實系統的運作過程，透過模擬的結果，不僅節省了許多不必要的成本，也可以作為研究參考之依據。但目前的模擬軟體大多數存在著無法處理複雜的模擬環境，且僅適用於單一領域的問題。因此，本研究希望建立一套具有易用性、有用性及泛用性的模擬平台。

在模型選擇的策略上，微分方程(ODEs)最主要的困難點在於模擬大型網路時，任意增減節點時必須重新撰寫整個模型，而且不具有圖形化也不易學習，因此，本研究選擇了具有圖形化與可以驗證模型正確性的 Petri Nets。Petri Nets 主要由 Transition、Place、Arc 與 Token 所組成，雖然容易使用，但是僅具有一種運作機制，塑模方式無法由使用者自由延伸，即使是改良過的 High level Petri Nets 能無法避開其缺陷，因此對於許多真實現象皆無法確切的表達與模擬。

有鑑於此，本研究將 Java 的繼承概念，引進了 Transition，使其語意(semantic)能夠改變，因此使用者可以依據需求而實做不同的 sub Transition；不論何種 Petri Nets，皆必須事先定義 Arc 的方向性才能模擬，本研究提出了動態 Arc 與資料庫的配合，產生整個模擬環境，而且 Arc 不需具有方向性；對於 Token 也可用連續型別處理，別於傳統只能傳遞離散型別的 Token；最後透過支援多核心硬體，將可縮短模擬時間。

關鍵字：模擬、Petri Nets、物件導向、Java、繼承

Title of Thesis: Design and Implementation of a Simulation System

Name of Institute: National Chi-Nan University, Pages: 65

Institute of Information Management

Graduation Time: 06/2008 Degree Conferred: Master

Student Name: Tai-Hong Pu Advisor Name: Shiuh-Sheng Yu

Abstract

A computer simulation is an attempt to model a real-life or hypothetical situation on a computer so that it can be studied to see how the system works. By changing variables, predictions may be made about the behavior of the system.

In the thesis, we propose the simulation software and it is based on Petri Nets. The proposed method is extending Petri Nets' semantic of transition by Object Oriented. Compared with original Petri Nets, the proposed method can not only execute the original transition but also can execute other different transitions that can be implemented via Java' inheritance. In addition, user can define fire rule in transition and change the original Petri Nets' execution semantic. Petri Nets' direction of arc is dynamic when a transition finished and determined the arc's direction. In other words, the relationship between transition and place is undirected, arc's direction is computing dynamic by transition. For example, in a chemical equation, the reaction is forward or reverses according to equilibrium constant, and the equilibrium constant is computed in a transition.

Furthermore, we propose a new way to generate simulation model quickly and automatically by combined dynamic arc with database, so user do not necessary generate model step by step. Lastly, the software can support with multicore processors and it can improve the simulation performance significantly.

Key words: Simulation, Petri Nets, Object Oriented, Java, Inheritance

目錄

圖目錄

表目錄

第一章　緒論

1.1　研究背景與動機

根據莫爾定律（Moore's Law）表示，一顆晶片上的電晶體數目將會以每十八個月的速度呈現指數性地倍增。這樣的預測，不僅僅只是造福個人使用電腦上的成本降低，硬體效能得以提昇，更帶動了所有科學嶄新的一大邁進。

也因此，電腦的強大運算能力也促成了模擬(Simulation)更廣泛的使用。不論是軍事武器殺傷力之探究、地質結構之分析、工業排程之管理、教育電腦輔助學習之使用、以及目前熱門的領域－藥物開發以及癌症機轉之探討等，都可以窺見一二。電腦模擬具有如此之魅力，究竟何謂電腦模擬？學者Alessi與Trollip於1985年提出，電腦模擬是一種利用電腦來模擬實物的反應模式，而不必經由實物製作亦能達到有效測試的一種工具；Reigeluth與Schwartz於1989年稱道，電腦模擬最基本的定義即是一種模式（model），此為一種事物的呈現，而其運作包含兩個或兩個以上變項的動態關係；Williams在1995指出，電腦模擬乃是一種應用電腦程式模組（model），來研究不同實物的反應動作，亦即研究系統本身對不同輸入的反應模式[53]。

電腦模擬的優點，不外乎對於時間與金錢成本的節省之外，欲使用模擬之技術，必定於研究範圍與假設有其困難與侷限，而衍生出一個較佳的解決方案。與大家息息相關的例子如同醫藥工業，他們所面臨的一大挑戰即是如何滿足市場、開發更有療效的新藥、以及創造新的收益，是醫藥工業所急迫追求的目標[4][13]。但是，藥物的開發並不是皆能以活體做實驗，況且必須對其使否具有致命的高風險做評估與考量，因此，模擬藥物與活體的交互作用，便是一個良好的解決之道。同理可證，舉凡不能在現實生活中實驗的如兩國交戰、彗星是否撞上地球、核子武器之殺傷力等，皆可體會模擬系統存在與使用之必要。

系統開發所面臨的挑戰，即是否容易使用以及是否具有助益。學者Davis於1989年提出的科技接受模式[5][6]，可以說明一個資訊系統的成敗，其模型如圖1-1。在此

模型中，外部變數會影響使用者對於資訊系統的觀感，使用者會對資訊系統產生認知，對於其本身是否易於操作？以及這樣的系統，對其工作或研究上是否有助益？這將會影響使用者是否願意繼續使用這套系統，這也關係到這系統能否延續或者就此腰斬。

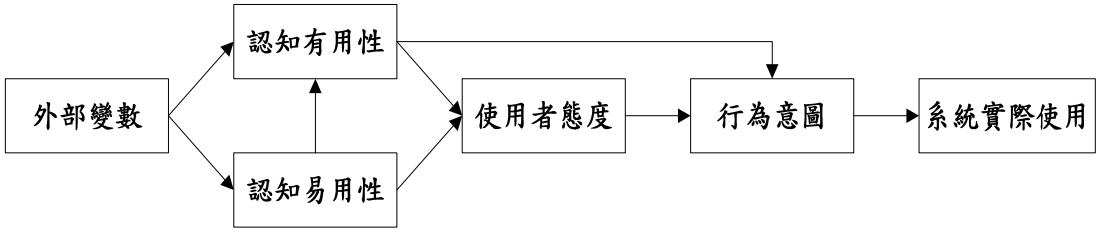


圖 1-1 科技接受模式

資料來源： Davis(1989)

模擬系統亦是如此，模擬系統首先必須具備容易使用的特性，例如親切的圖形使用者介面、簡潔的操作指令與功能，以及結果的呈現能夠直觀的表達，讓使用者方便解釋。除了系統便於操作之外，尚須考慮系統模擬結果的精確度與準確性。模擬系統之目的，就是能夠提出一個仿真的模型，透過這個模型去預測真實系統的情形，所得到的結果愈精準，愈能解釋以及描述真實的狀況，但是模擬系統並非一般性的系統，欲研究的變數具有相依關係。因此，結果的精準度並非容易可得，唯有透過不斷的修正模型與驗證模型，才能提高系統的價值，這個部份，也是影響模擬系統成敗的一大關鍵。綜合上述，雖然開發模擬系統具有許多的問題與挑戰，但仍然值得探討，故本論文將提出一個模擬平台的架構並且實做之，期許能夠應用其於不同領域之研究。

1.2 研究目的

目前多數的模擬軟體存在著不易操作與系統彈性的問題，例如：使用者介面的功

能過於複雜，需要一段時間的學習才能上手；在建構模型時，有的模擬軟體的模型是無法依據使用者的需求做修改，而有的軟體則是需要使用者透過圖形化介面，將整個環境徒手建立，尤其模型僅具有一種語意，塑模的方式無法由使用者自由延伸，也缺乏了透過資料庫建立模擬環境的能力。上述問題，不僅耗費時間且容易造成錯誤，次者，多數軟體在計算方面速度太慢，並且缺乏對於多核心硬體的支援，加上目前的模擬軟體大多只能侷限於某個領域，如生化網路之研究。有鑑於此，本研究目的將建立一個同時具有易用性、有用性及泛用性的模擬系統，不僅能夠容易操作之外，更能應用於不同之領域研究。而目前本研究將以簡單的代謝網路－醣酵解作用作為簡單的測試範例。

1.3 研究架構

本研究共分成五個章節，架構如下：

- 第一章為緒論。說明本論文的研究背景與動機、研究目的和研究架構。
- 第二章為介紹與整理文獻探討。主要簡介 Petri Nets 的定義與分類 、Low level Petri Nets 所產生的問題、High level Petri Nets 的改良，以及概述系統生物學的觀念。
- 第三章為系統設計。將說明系統之組成：Shell、Transition library 和 Kernel。
- 第四章為系統實作。將前述所提出概念、模擬策略與演算法等實做平台。
- 第五章為結論與展望。

第二章　文獻探討

2.1 Petri Nets

2.1.1 Low level Petri Nets

Petri Nets 是由 Carl Adam. Petri 於 1962 年的博士論文提出，再藉由多位 MIT 的研究人員於 1968 年至 1976 年的研究，才使理論逐漸完備。Petri Nets 不僅具有嚴謹的數學推導基礎，更可以透過圖形化的方式，利用 Token 動態的模擬系統的行為。因為兼具上述兩者優點，可見其被應用於研究具有同時性(concurrent)、非同步性(asynchronous)、隨機性(stochastic)、分散性(distributed)、平行性(parallel)、非決定性(non-deterministic)等資訊系統[52]。

Petri Nets 是由 Transition、Place、Token 和 Arc 所有組成，圖形與意義如下表 2-1 之整理，表 2-2 則為大部份對 Transition 與 Place 所作的詮釋[32]：

表 2- 1　Petri Nets 基本符號

名稱	圖形	意義
Transition	□	**以方框表示，可以是系統的事件或者工作。**
Place	○	**以圓形代表，表示 Transition 所需要的條件或者是系統內可利用的資源。**
Token	●	**當 Token 出現在 Place 內時，可以視為該 Place 的條件成立或具有的資源數目。**
Arc	→	**為一個有向射線，且賦予權重，為 Transition 和 Place 的連結器。**

資料來源： Murata(1989)

表 2- 2 Transition 與 Place 的詮釋

Input Places	Transitions	Out Places
輸入資料	計算步驟	輸出資料
輸入訊號	訊號處理	輸出訊號
資源需求	工作或處理	資源釋放
前置條件	事件	後置狀態
緩衝器	處理器	緩衝器
條件	邏輯子句	結論

資料來源： Murata(1989)

Petri Nets 在數學的表示法是由下列五種元素所組成，PN=(P, T, F, W, M_0)，其正式定義如下：

1. P = {$p_1, p_2, \ldots, p_m$}，為 Place 的有限個數集合，m 表示在系統內的 Place 個數。
2. T = {$t_1, t_2, \ldots, t_m$}，有限個數的 Transition 集合，m 表示在系統內的 Transition 個數。
3. $F \subseteq (P \times T) \cap (T \times P)$，為 Arc 的集合，具有方向性，作為 Transition 和 Place 之間的關聯。
4. $W : F \rightarrow \{1, 2, 3\ldots\}$，權重函式，Transition 必須從 Input Place 取出多少 Token 有關。
5. $M_0 = \{M(p_1), M(p_2), \ldots, M(p_m)\} : P \rightarrow \{0, 1, 2, \ldots\}$，代表初始標記，其中 $M(p_i)$表示在位置 p_i 內 Token 的數目。

由上述定義，將可以表達 Petri Nets，若要考慮整個網路的動態行為，將會根據以下的引發規則來改變：

1. 一個 Transition 為待發的狀態，根據其 input Place 內是否具有 Token 而定。當每 Place 至少含有 w(p, t) 個 Token 時，則可稱其為待發狀態(enable)，而 w(p, t)代表

從 Place 至 Transition 的 Arc 上的權重值。

2. 一個待發的 Transition 還會因為其他的事件、條件或其他因素決定是否觸發(fire)，因此一個待發的 Transition 不一定會觸發。

3. 當 Transition 觸發後，會從每一個 input Place 內，根據 w(p, t)傳遞 Token 至 output Place，因此系統的狀態因而改變。

如圖 2-1 為一個 Petri Nets 簡單的示意圖，就相對位置而言，P1 對 T1 而言可視為 input place，其內具有一個 Token，Arc 若沒有標示權重時，將預設為 1。因此，當 Transition 的每一個 Input Place 含有 Token 時，可視其為待發的；若滿足觸發的充分條件時(即擁有的 Token 數目大於或等於 Arc 的權重)，將發生轉置動作。因此 P1 內的資源將會透過 T1 傳遞，同理，P2 也將會傳遞 Token，結果如圖 2-2。

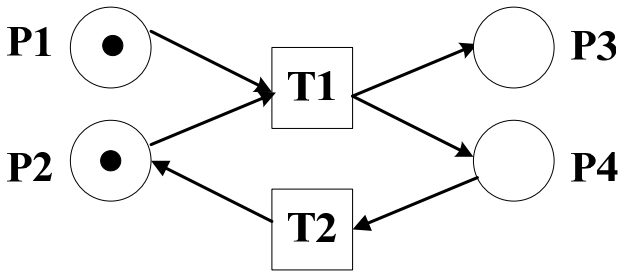


圖 2- 1 Petri Nets 簡單示意圖(a)

資料來源：本研究之整理

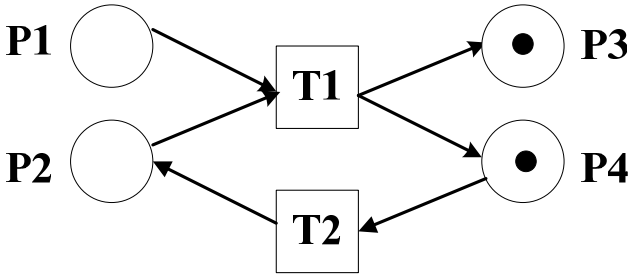


圖 2- 2 Petri Nets 簡單示意圖(b)

資料來源：本研究之整理

除此之外，Petri Nets 具有下列的性質，分述如下表 2-3：

表 2- 3 Petri Nets 的性質

性質	意義
可及性(reachability)	**若存在著一組可觸發的 Transitions，使得 Petri Nets 可由標記 M 推論至 M'時，則稱標記 M'為標記 M 可及的。**
圍束性(boundedness)	**唯有系統內的單元與資源為有限時，才有資格被驗證。因此若某個具有初始標記 M_0 的 Petri Nets，其標記內的資源為有限個數(皆小於或等於一個正整數)，則可稱此系統為 k-bounded。**
安全性(safeness)	**具有初始標記的 Petri Nets 即可稱之。**
活轉性(liveness)	**不論從任一標記開始，每一個 Transition 皆可被觸發，且不會因為曾經被觸發後就不能在被觸發，換句話說，活轉性避免 Transition 發生 dead 與 deadlock 的情形。**
可逆性(reversibility)	**於 Petri Nets 中，若每一個標記 M'皆為可達性，且可以透過任一標記推回初始標記 M_0，則可稱為可逆性。**

資料來源：本研究之整理

Reddy et. Al(1996)為最早提出以 Petri Nets 模擬生化網路的作者[40]，他們以 Petri Nets 為工具，針對紅血球細胞(erythrocyte cell)的糖分解作用(glycolytic)和戊醣磷酸路徑(pentose phosphate pathway)作定性分析(Qualitative analysis)之研究。他們認為 Petri Nets 對於分析複雜的生化網路非常有益，因為這工具不僅容易使用，且圖形化的表達容易讓人理解。文獻[36][11][21][35] [42]同樣以 Petri Nets 應用於生化路徑模擬。爾後

的研究學者，將應用各種 High level Petri Nets，如 Timed Petri Nets、Stochastic Petri Nets、Colored Petri Nets 和 Hybrid Petri Nets，針對不同的生化網路型態與情形作定性分析或定量分析的研究，此部份將於後面小節分別介紹。

2.1.2 High level Petri Nets

2.1.2.1 Timed Petri Nets

Timed Petri Nets 最早由 Ramchandani 於 1974 年的博士論文提出[39]，傳統的 Petri Nets 並沒有時間參數的概念，因此只能描述系統的靜態結構，卻無法在一個有限時間內，計算出系統內各元件在運作時的速度，因此 Time Petri Nets 可以用來描述具有固定執行時間的系統，依據每一個 Transition 延遲時間的不同，為系統排程規劃的工具。

Timed Petri Nets 的基本定義大致與傳統的 Petri Nets 一樣，是由六個元素所組成[37]，即 $Z = (P, T, F, V, m_0, I)$，其中前五項欄位可以定義基本的 Petri Nets，I 為時間函數，定義了最早觸發時間(earliest firing time)以及最晚觸發時間(latest firing time)。研究[38]以研究[22]於 1999 年所提出的概念，針對碳的代謝分別以 Timed Petri Nets 做定性與定量的分析。研究[24]則是利用 Timed Petri Nets，針對訊號傳導網路(signal transductions networks)，提出 12 種不同的反應現象，藉由這個工具去描述之，並且以細胞自毀(apoptosis)作為個案研究。

2.1.2.2 Stochastic Petri Nets

Stochastic Petri Nets 是一種 Timed Petri Nets，被稱之為 Generalized Stochastic Petri Nets[34]，其應用於分散式系統如多處理器、區域網路和自動化製造系統等。在專書[26]中提到了 Stochastic Petri Nets 具有隨機的觀點：待發(enable)的 Transitions 可以依據指數分配(exponentially distributed)來決定其延遲時間(time delays)。在基本定義中，其由七個(P, T, Pre, Post, F, λ, M)值組所構成，除了前五項為基本 Petri Nets 之外：P 表示一組有限的 Place 集合；T 為一組有限的 Transition 集合；Pre 和 Post 分別為 input Arcs

和 output Arcs 的權重值；D 為定義觸發時間的機率密度函數(probability density function)；而 λ 以實數定義，為一組觸發速率(firing rates)，用此計算與 Transition 的機率密度函數。

文獻[9]以此工具對迷你染色體的複製做定量研究，透過不同的時間點去計算出蛋白質的產出數量以及機率分佈。文獻[46]則探討了女性更年期循環個案，他們希望能找出本身基因與外在環境的影響下，所造成停經期後可能產生失調的風險。經由不同的時間與反應速率的計算，測量出不同時期下體內各種激素的變化。Stochastic Petri Nets 雖然具有圖形化且易於使用，但是仍有其侷限－缺乏空間(Compartmentalized)的概念，因此對於物質的傳遞無法表達。

2.1.2.3 Coloured Petri Nets

Jensen(1991)為最早提出 Coloured Petri Nets 的研究者[15][16]，別於傳統的 Petri Nets，Token 可以具有不同的型態，不再只是只有 0 與 1 的區別，賦予 Token 不同的顏色，將可以應付日趨複雜的系統。典型的例子，Coloured Petri Nets 常應用於通訊協定、分散式系統、嵌入式系統、自動化生產系統及工作流程分析。在數學定義中，是由(P, T, Pre, Post, M, C)五個元素所構成，前四個元素與前述相同外，C 代表一組顏色，是對應 Token 的顏色函數。

在生化網路的應用，研究[10]以醣酵解路徑和戊醣磷酸路徑做定性分析，研究[41]延續前者，對其做定量之研究。研究[41]將定性模型(core model)延伸，加入環境因素(分別為型 I 無初始值和型 II 有初始值的環境條件)，此稱之為系統模型(System model)。為了解決模型中某些 Transition 可能發生 dead 情形，可以 Token 內加入時間的概念或相關訊息，解決衝突的發生。研究[8]也以醣酵解路徑為例，以物質名稱與濃度為組合，使 Token 具有不同的彩色值組(colour set)，並透過公開資料庫(利如 Brenda)取得所需的數據，再依照修正後的酵素動力學(Michaelis-Menten law)，實做每一個 Transition 所需的計算法則。

2.1.2.4 Hybrid Petri Nets

研究[3]提出了 Hybrid Petri Nets 的數學定義為 Q=(P, T, Pre, Post, M, h)，同樣地，(P, T, Pre, Post, M)是基本的 Petri Nets 組成元素外，h 是 hybrid function，它表示著每一個網路中的節點(Transition 或 Place)的屬性為離散型節點(discrete node)或者是連續型節點(continuous node)。

就生化意義而言，Discrete Place 可以非負整數(non negative integer)的型態表示一個蛋白質是否與 DNA 結合或者多少分子量位於此胞器中；Continuous Place 可以為物質的濃度，以實數表示(real number)；Discrete Transition 具有交換機制(switching mechanism)，處理接收與傳遞訊息；Continuous Transition 則根據所計算出的速率，對接收或傳遞的物質做消耗與堆積的處理。

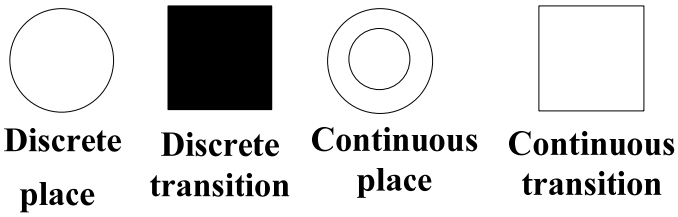


圖 2- 3 Hybrid Petri Nets 元素符號

資料來源：Chen(2003)

[3]此篇文獻作者建立了基因調控網路模型(Model of Genetic regulatory networks)，雖然 hybrid Petri Nets 具有親切的使用者介面，但是仍點出了一個問題：對於基因接受到訊息後，完成編碼形成產物的過程，因為無法清楚的了解其機制，所以將其視為黑箱處理。

文獻[28]、[33]、[31]與[29]為同一作者團隊，針對生化網路所提出的研究。在文獻中，他們希望藉由此工具，能夠模擬三大路徑：代謝路徑(Metabolic Pathway)、基

因調控路徑(Gene Regulation Pathway)和訊號傳導路徑(Signal Transduction Pathway)。除了電腦的輔助可以幫助使用者觀察細微且複雜的動態變化外，透過模擬希望能應用於藥物開發，改進藥物的療效。

2.1.3 Petri Nets 應用於生化網路所面臨的問題

傳統的 Petri Nets，基於一個問題－Transition 的運作機制僅具有一種語意，也因此應用於生化網路時，會面臨下列三種現象而無法表達：

- Arc－逆反應的處理。
- Place－反應物同時觸發多個 Transitions。
- Token－對於物質的傳遞，可以連續型別的量化處理。

一個生化反應的進行，會有正反應及逆反應，以及最終達至動態的平衡反應。因此，Transition 的運作，應該包括了反應物與產物的量化計算，這個量化的計算，指的是否可以連續型別的量化處理，因為在傳統的 Petri Nets 對 Token 的處理，只要達到觸發條件，將依權重需求而全數轉置，但就生化現象而言，這是不合理的，因為生命體內的各種物質應處於一種恆定的狀態，若依照此機制，各種物質不是處於濃度過高或者就是匱乏狀態。除此之外，也牽涉了反應進行的方向，反應並非永遠單向，會依照濃度、活性、抑制劑或其他因素，導致反應正向或逆向的進行。至於反應物，有可能同時觸發多個反應的情形。反應的進行，最終使整體生化網路能形成穩態(steady state)的平衡，因此 Transition 的內部機制若不能更改，將無法處理上述現象。

以 Low level Petri Nets，研究[40]為例，是最早使用 Low level Petri Nets 對醣酵解反應做定性分析，既然為定性研究，故沒有涉及量化的問題，但是在建構路徑模型時，並未考慮逆反應的表達，僅以單向反應為推論基礎。

就各種 High level Petri Nets，也同樣面臨了上述的問題。以 Timed Petri Nets 為例，研究[38]針對馬鈴薯莖的碳代謝做了定性研究，雖然有將逆反應(以雙向的 Arc 表示)

納入模型建構，但是對於模型驗證上，卻無描述逆反應如何運作；研究[24]探討訊號傳導網路中的細胞自毀之機制，他們將路徑中的 Transitions 細分為 12 種不同的反應型態，其中亦有代謝反應，但就各種 Transitions 的內部運作而言，並無談及，更遑論當涉及代謝反應時，如何進行反應物的消耗與生產，進而達到動態平衡。對於 Place 同時觸發多個 Transitions 的衝突問題，雖然提出可以 Transition 的觸發頻率×Arc 的權重來計算同時觸發的 Transitions 的可能延遲時間，但是在此文獻中的範例卻仍無法解決此問題，因為計算出的延遲時間仍皆相同，至於 Token 的消耗型態仍以離散型。

在 Stochastic Petri Nets 的研究中，以研究[9][46]為例，主要探討著 Transitions 將以機率分配來決定觸發的延遲時間。當 input Place 已經具有 Tokens，且判斷事件的閘道(gate)為 true 時，即可在時間範圍內發生反應。因此藉由機率的不同，將可以解決 Place 同時觸發多個 Transitions 所產生的衝突，不過對於其它問題則無解決之道。

Coloured Petri Nets，研究[10]以研究醣酵解路徑為標的，作定性分析。在這篇文獻中，他們對於路經中的逆反應做了處理－虛設另一個相同的 Transition 表示為逆反應；對於 Place 同時觸發的衝突問題，因為沒有涉及量化分析，所以 Place 的同一種物質將以不同顏色的離散型態之 Tokens 所構成，解決路徑選擇的問題，但對真實現象而言，生命體內並不可能會將所需的同一種物質做標記，此方法失之常理。研究[8]則是針對醣酵解路徑作定量分析，其透過修正過的酵素動力學模型做計算，因此考量了逆反應以及 Token 可以連續型別量化。對於 Place 所產生的衝突問題，在此文獻中，因為路徑恰巧沒有這種問題存在，因此沒有探討解決的方法。終歸而言，仍未針對 Transition 的內部運作機制做處理，因此無法有效解決上述現象。

Hybrid Petri Nets，文獻[3]以先天性遺傳代謝缺陷疾病(inborn errors of metabolism)為個案研究。此篇幅只介紹了 Hybrid Petri Nets 的基本定義以及表式法則，並未對生化網路中所產生的現象，多做說明，且作者在文中直接表示，基因收到訊息到產生所需物質的這段過程，仍處於深晦不明，因此對於 Transition 仍視為黑箱機制的處理(black box of Transition)。研究[28][29][33]為日本研究團隊，針對三大網路所進行的建

模與分析。在代謝網路的分析上，以醣酵解作用為例，並無探討逆反應的處理；Place 所產生的衝突問題和 Transition 的處理機制，也未提及，唯一改良了傳統 Petri Nets，即 Place 與 Transition 可依照情形視為離散型態或連續型態，因此可以對 Token 做量化分析。表 2-4 為各種 High level Petri Nets 整理之摘要。

表 2- 4　High level Petri Nets 比較表

分類 / 衍生問題	Timed Petri Nets	Stochastic Petri Nets	Coloured Petri Nets	Hybrid Petri Nets
Place 同時觸發 Transitions		◎	◎	
逆反應的處理			◎	
消耗連續型別的 Token				◎
Arc 為動態連結				

註：◎表示雖然針對部份問題解決，但結果仍不符合實際現象

資料來源：本研究之整理

上述所提及各種的 High level Petri Nets 雖然想要針對問題來解決，但是所解決的方式並非符合真實現象的情形，比方說 Place 同時觸發的問題，在生化網路顯而易見，為一種平行處理的概念，但是以機率或顏色來決定路徑的選擇仍不符合實際現象；生化網路總存在著許多可逆反應，但是因為 Petri Nets 本身的侷限而無從表達；對於生化網路中物質的消耗與產出，應該屬於連續型別的處理，經過一連串的震盪反應，可以維持整體性的動態平衡，但是在 Petri Nets 中，一旦達到觸發條件，將全數轉置 Token，若一個生物體內存在著這樣的運作，可能具有致命性的危機。除了這些問題之外，更重要的，Petri Nets 的 Arc 皆為靜態連結，這意謂著要模擬之前，必須先事先定義好整個環境的所有連結，以及 Arc 的方向，均為單向性，因此對應於真實世

界的應用，不但不完美，且在建構模型時，不具有彈性。

2.2 Ordinary Differential Equations

將微分方程(Ordinary Differential Equations, ODEs) 作為描述現象的工具，可由一些例子窺見一二，如細菌或生物的成長、動植物群體的無限制生長、馬爾薩斯的人口定律等[55]。雖然微分方程應用於生物網路的模擬，有著其侷限性：非連續性的狀態改變、物質傳遞與擴散、細胞移動、空間概念的表達等，但對於模擬與時間相依的事件，則是具有相當的能力[27]。利用微分方程所開發的生化網路之模擬軟體已經行之有年，因此文獻[1][54]分別將目前市面上以微分方程建模的軟體作整理，針對其功能面、系統可靠度、系統效能、系統介面等作比較；文獻[23]則介紹了應用於代謝工程中較為有名之軟體，如 Virtual Cell [43]、E-Cell[45]與 Gepasi[30]。Virtual Cell 與 E-Cell 分別考慮了在物質分佈不均勻的系統內，各種物質的濃度變化不僅與時間有關，同時也會受到擴散速率的影響；Gepasi 則適用於模擬物質均勻分佈的環境。

但是在使用微分方程模擬時，對於計算所需的參數，如反應速率、擴散速率、相關濃度等，在估計上有其困難；對於模擬生化網路時，任意新增或減少節點時，則必須重新撰寫方程式，使用與計算徒增了困難與費時，不具有易用性；當然，最重要的，使用微分方程往往需要具有良好的數學基礎，因此對於大多數的人而言，在使用上有其困難度。

2.3 Systems Biology

系統生物學最早是由參與推動人類基因體計畫(the Human Genome Project)的 Hood, L.所提出，對於系統生物學的框架做了以下四點說明[12][14]：

1. 定義關於這個系統內的所有元件：

首先透過以往所得的生化知識，利用生命的元件去建構一個模型。建構模型是以

全域(a global approach)的方法達到兩個目標，分別為(a)透過系統內的元件互動來描述系統的整體行為，(b)準確地預測系統經由特別的擾動後的相關特性。對於建構模型，可以透過描述、圖形化或數學式等方法，而元件的資料收集可以透過高通量的技術。

2. 系統性地擾動以及監測這系統內的元件：

內在的基因遺傳或者外界的環境變遷將可以對生命體作特別的干擾，輔以大規模的探索儀器觀察各種元件受到擾動後的生化訊息，並將所收集的實驗數據整合後與模擬結果作比較。

3. 調和實驗所觀察的現象與模型所產生的預測結果：

透過修正模型以期待所預測的數據更能貼近實驗的觀察結果。當模擬結果無法與實際的觀察配適時，將選擇其它的假設作實驗，以求減少這兩者之間的不一致程度。當模型太複雜或者無法獲得，則觀察的結果將被推論這些特別的元件所產生的功能是被系統所需要的；相對的，若模型已經定義良好，則所預測出的結果也許和觀察的結果有良好的近似性，則這裡的差異僅在於預測範圍的改變。

4. 設計和執行新的擾動實驗，用來區別多樣的或者只能擇一的模型假設：

為了能與大多數所觀察到的現象近似或吻合，修正模型即是為了達到此目的。當目前的數據無法辨別模型的差異時，就必須運用新的擾動實驗。透過這樣的測試，將可以發現哪個模型的預測結果最不準確。透過重複上述的 2.到 4.的步驟，持續的擴大以及淬煉模型，使模型更完臻。

簡單來說，系統生物學即是整合了技術、生物學和電腦計算。人類基因體計畫帶動了生物學的革新，使得生物學成為了資訊生物學(informational biology)。基因體是生物的數位資訊，了解基因體不但對體認生物有益之外，基因體還帶給我們許多資訊：透過基因可以了解相關的蛋白質的催化機制，以及基因在調控網路中的行為。而 Hiroaki Kitano 對於系統生物學也有其見地，對於了解生物細胞或組織的靜態結構和動態功能，必須以全系統的觀點著手[20]。系統的複雜性不可能

單就組合部份元件就可以了解系統全體，他認為以系統的角度的研究方法具有四個特性：

1. 系統結構：包括了系統內生化網路中所有的互動元件，這些元件在細胞內以及在多細胞所呈現結構上不同的生理特性。
2. 系統動態性：隨著時間轉移，透過不同的分析方法如代謝分析、敏感度分析、動態分析等，觀察系統內各元件的變化。
3. 控制方法：系統性的控制細胞狀態能夠使機能失常最小化，藉此找出疾病治療的潛在標靶。
4. 設計方法：透過定義好的設計原則以及模擬方法，取代非盲目的試錯法，可以幫助我們策略性的修正以及推導出生物系統，這樣的系統也具有我們所期待的特性。

在資訊爆炸的時代，雖然豐富大量的生物資料可以透過文獻或者公開的資料庫取得，藉由這些資料可以了解生命體的結構以及因為環境的變化所產生的不同現象，但是對於其原因，仍不能輕易了解。因此，透過假設驅動的方法研究，從靜態結構，利用預測推論至動態行為。科技資訊的創新，高通量的技術快速產出大量的實驗數據，藉由這樣結果，可以使得模擬方法更加完臻。以上的概念可由圖 2-4 說明：

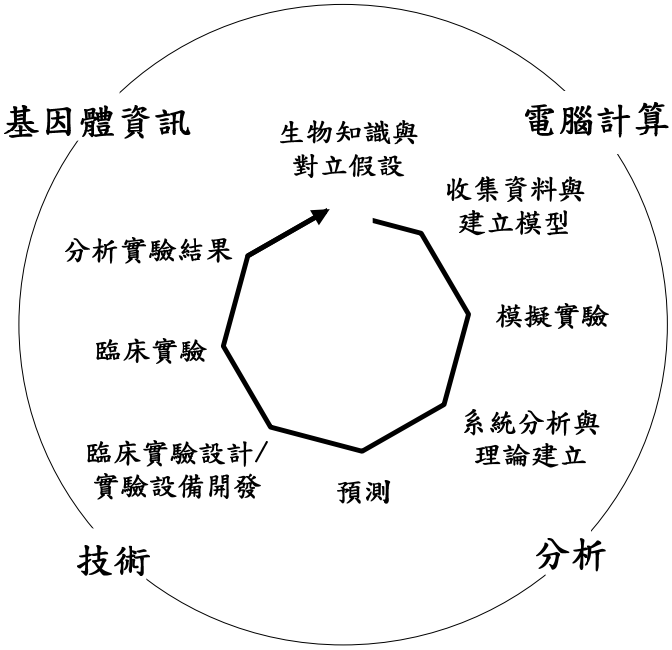


圖 2- 4　系統生物學

資料來源：本研究之整理

第三章　系統設計

在設計方面，我們模擬平台主要分為 Shell、Transition library、Kernel，以及所需連結的 Database 或 Files 等，其系統架構以圖 3-1 表示之。本章節將對系統各部份進行介紹，為何藉由此架構將不僅可以表達 low level Petri Nets，更能解決原本無法描述於生化網路的缺陷。

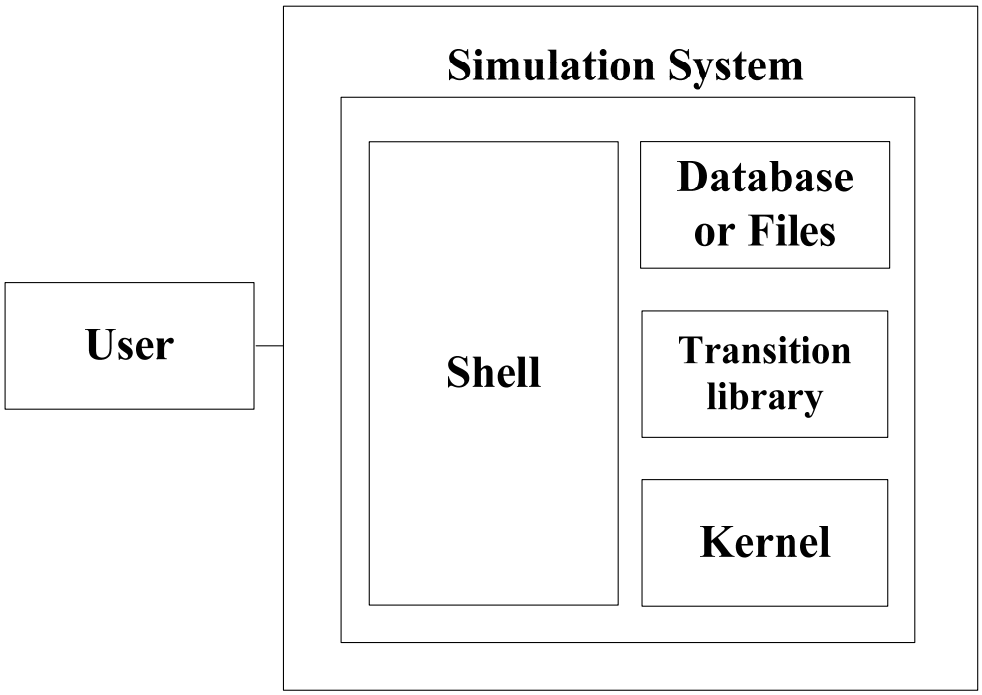


圖 3- 1　系統架構圖

資料來源：本研究

3.1 Shell

Shell是使用者直接接觸的部份,也是系統的進入點,使用者將透過shell 與Kernel溝通和互動。使用者可以依據不同的需求去定義且產生不同的 Shell，例如建構模型的方式有透過關聯式資料庫、讀取 XML 格式(如 SBML、CellML 等)、藉由純文字檔或其格式的檔案等，因此可以依照不同的讀取方法建構不同的 Shell。此外介面可以提供模擬計算、存檔和讀檔等基本功能，或由使用者自行定義開發其他功能。圖 3-2 為使用者透過 Metabolic Shell 與系統互動的情形，此 Shell 是透過關聯式資料庫讀取資料，而使用者可以使用建立模擬環境(build)、執行模擬(start)、停止(stop)、存檔(save)、讀檔(load)和離開系統(exit)等功能，當然使用者可以視實際情況作新增或修改功能。

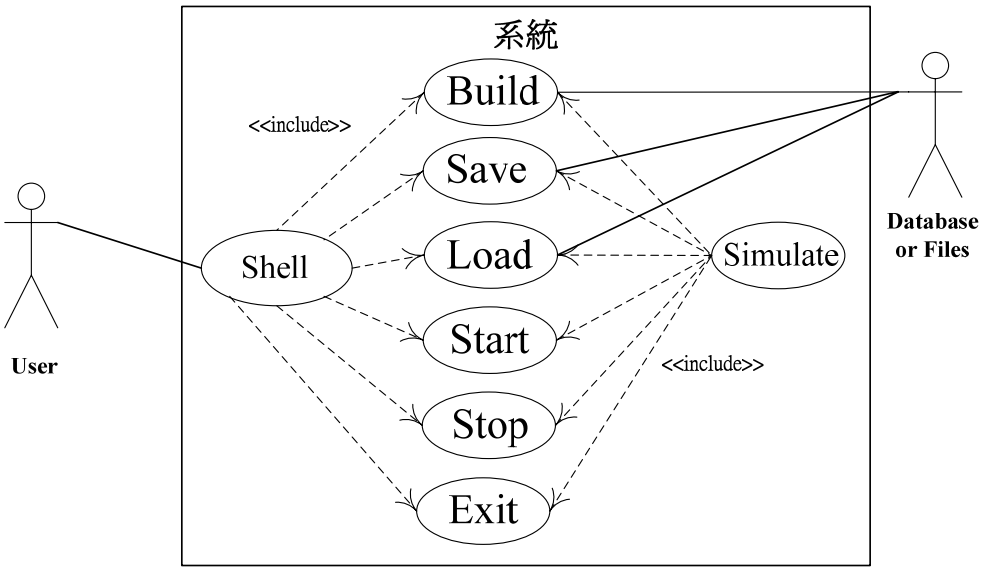


圖 3- 2 使用案例圖

資料來源：本研究

3.2 Transition

就 Low level Petri Nets 而言，Transition 僅有一種運作機制，如圖 3-3 和 3-4 所示，只要當 Place 擁有的 Token 數目大於 Arc 的權重時，且滿足 Fire Rule 時，Transition 將轉置該 Token。圖 3-5 為本模擬系統所提出的定性模型結構，此模型中所涉及的為 Transition 與 Place，別於傳統，本研究特別將計算物件的概念嵌入 Transition 中。

因此，本研究引用了物件導向(Object-Oriented)的技術，藉由 Java 的繼承改變了 Transition 的行為，修正後的 Transition，不僅能夠模擬 Low level Petri Nets，使用者可依照環境的假設和條件的需求，繼承這個 super class，接著實做不同的計算法則即可。本系統中，即針對代謝網路的反應實做了平衡解的計算物件。

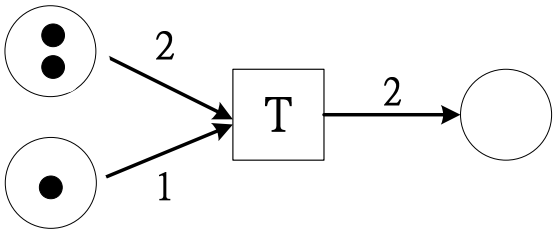


圖 3- 3 轉置前

資料來源：本研究

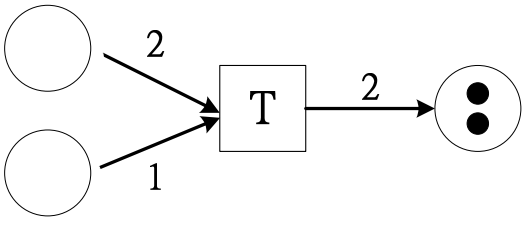


圖 3- 4 轉置後

資料來源：本研究

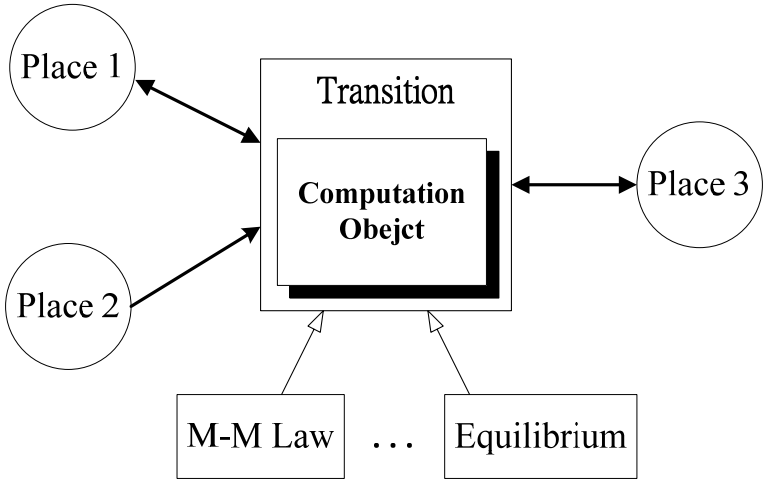


圖 3- 5　修正後的 Transition 模型

資料來源：本研究

在生物網路上會面臨以下問題：

1. 反應是否可逆和平衡的處理？
2. 產物與代謝物量化的處理？
3. 一個反應物是否能同時觸發多個反應？

傳統的 Petri Nets 均無法表達以上的概念和問題。生化網路上反應物的消耗與產物的增加，並非單純的只視 Arc 的權重即可決定一切，因為這牽涉到網路的平衡問題，反應平衡並非靜態，況且 Transition 與 Place 之間互相聯繫，牽一髮而動全身，是必須經由無數次的震盪反應而達到的動態恆定狀態。因此大多數以 Petri Nets 建模者，只是描述一個 Transition 的反應速率如何計算，以決定反應的先後次序，卻無明確提及這反應的計算機制。對於反應為可逆的處理，為了避免麻煩，多半假設為單向反應；而會正視此問題者，在一般定性模型會將 Arc 表示為雙向型態，但是實際在運作的情形多半會虛設一個 Transition，代表著逆向的反應，並且分別給予不同的反應速率。這個方法並非不可行，只是在建構模型時會徒增麻煩，一旦參與反應的數量過大，可能會造成設計上的錯誤。

修正過的 Transition 將不受此限制，Transition 可以與其相關的 Place 動態地產生關聯，因此在執行期間時可加以變動，換言之，Arc 不必如傳統的 Petri Nets 事先定義完整，而 Transition 之間的關聯性也由共同的 Place 所建立；反應的進行，也不受 Arc 的方向所影響，透過 Transition 動態計算後，檢查觸發條件與相鄰的 Transition 而決定路徑的走訪。而且對於 Place 的 Token 消耗，也不只是單純的整數型態的處理，畢竟在生命體內的物質，應該是可以量化的，透過這樣的機制，將可以更準確的預測 Place 的質與量的變化。

最後關於一個 Place 如何觸發多個 Transition？目前也是個懸而未解的問題，然而生化反應往往是同時間的大規模作處理，並非只是單純的循序執行且資源無法量化處理，因此，本研究將透過平行處理的技術和反應速率等概念來解決此問題。

本研究也對 Place 提出不同的闡述，傳統在模擬生化網路方面，總是單純的把 place 視為反應物或產物，為 Transition 所必須的資源而已，若我們可以加入空間的概念，將可以在不同場域(Compartment)傳遞物質。假設 H_2O 將從細胞質傳遞至粒線體內，在這裡我們考慮反應的擴散速率因為太快可以忽略不計，因此在模型上的表達可以如圖 3-6 所示。

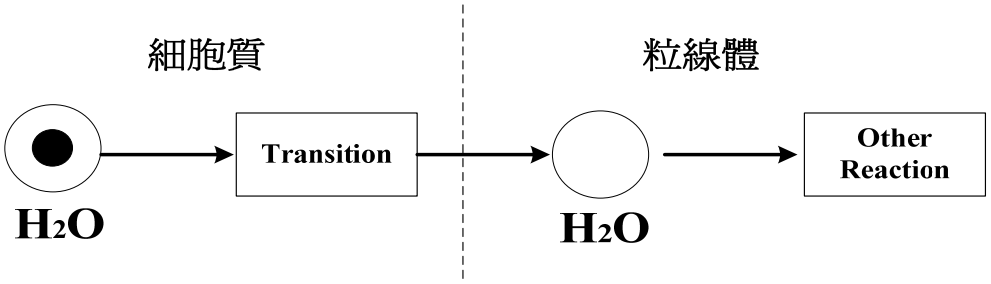


圖 3- 6 Place 的空間概念表示圖

資料來源：本研究

3.3 Fire Rule Parser

Fire Rule Parser 的功能，是將 Transition 觸發規則(Expression)剖析後並驗證，判斷 Transition 是否達到觸發條件。觸發規則的表示法假設有以下幾種型態：

```
Expression   ：：＝ ( Expression )
                  ｜(Expression ) and Expression
                  ｜(Expression ) or Expression
                  ｜SimpleExpression
                  ｜SimpleExpression and Expression
                  ｜SimpleExpression or Expression
SimpleExpression   ：：＝ literal > value
                        ｜literal < value
                        ｜literal >= value
                        ｜literal <= value
                        ｜literal = value
```

圖 3- 7　觸發規則表示法

資料來源：本研究

當 Transition 建立的時候，並需對其觸發規則做驗證，因此透過 Fire Rule Parser 的機制，會先將其傳入剖析器做字串處理，經過字串的處理可以分辨屬於何種 Expression，這個動作會持續的將規則分析成 Expression 或 SimpleExpression，最後依據其條件驗證是否達到觸發條件。以代謝網路中，如圖 3-8，參與糖酵解作用第一個反應的酵素為 HK，假設此反應的觸發條件為 HK ＞ 0。則經過處理後，此為

SimpleExpression，因為‘HK’為 literal，‘>’為運算符號，‘0’為 value，因此當 HK 的濃度大於零時此反應將會發生。

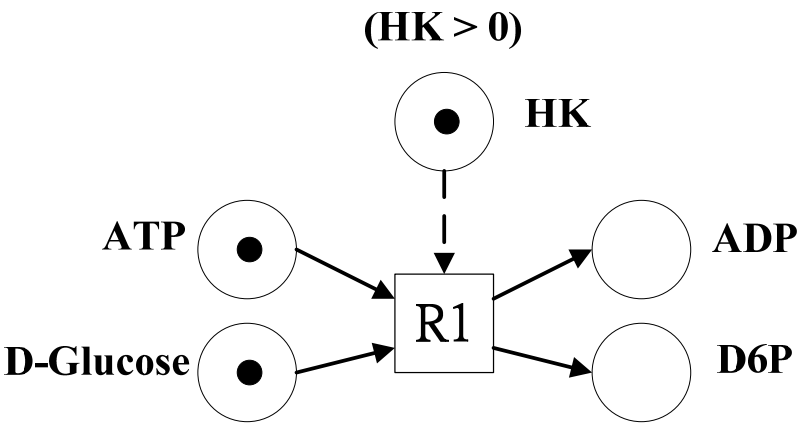


圖 3- 8 Fire Rule Parser 示意圖

資料來源：本研究

3.4 Kernel

做為整個系統的管理者，可以視其為類似作業系統中的角色，kernel 的主要工作在於負責分派 Places 與規劃 Transitions 做排程管理，直到所有的 Transition 執行完畢為止。

因此，在使用者透過 GUI 建立起模擬環境後，Kernel 也會同時產生，這個核心物件將作為模擬系統的監控單元，因此對於所有的 Transition 都必須向 Kernel 註冊。此外，Kernel 具備取得所有資源的功能，這項服務將提供 Transition 設定其相關資源的參數值。不僅如此，Kernel 可以預視資源(peek Place)的能力，透過這個功能，GUI 將可以找出每個資源的初始值。

Kernel 在排程運作機制，必須透過兩種 Queue 來處理，分別為 Priority Queue 和 Runnable Queue。以 Priority Queue 將所有 Transition 依照其反應完成時間的順序做為優先權排程；而 Runnable Queue 則在於當 Transition 可以取得並鎖住所需資源時，且獲得了 CPU 資源後，將由 FIFO 的特性，從 Queue 取出並執行。因此，在 Kernel 中，將運作兩種 Thread：Main Thread 和 Executor Threads。Main Thread 的目的在於產生一個 Priority Queue 和執行 Executor Threads；而 Executor Threads 的目的則在處理 Runnable Queue。

3.5 模擬排程之機制

誠如上述，在排程方面必須實做一個 Priority Queue 和等待執行的 Runnable Queue。Priority Queue 作為所有 Transition 處理次序的依據，從中取出最小值放入 Runnable Queue，再經由先進先出(FIFO)的性質執行。藉由文獻摘要整理，本研究將以 Fibonacci heap 為排程的資料結構，圖 3-9 為效能比較摘要表。

表 3- 1 效能比較摘要表

procedure	Binary heap (worst-case)	Binomial heap (worst-case)	Fibonacci heap (amortized)
Make-Heap	**Θ(1)**	**Θ(1)**	**Θ(1)**
Insert	**Θ(lg n)**	**O(lg n)**	**Θ(1)**
Extract-Min	**Θ(lg n)**	**Θ(lg n)**	**O(lg n)**
Decrease-Key	**Θ(lg n)**	**Θ(lg n)**	**Θ(1)**

資料來源：Cormen, Leiserson and Stein(2001)

3.6 Transition 的觸發與執行

圖 3-9 為 Transition 所具有的四種狀態，實線為狀態變化的過程，當 Transition 執行完畢後，可能會觸發其他反應，屆時需要作一些處理，將在後面分別討論以下四種狀態。

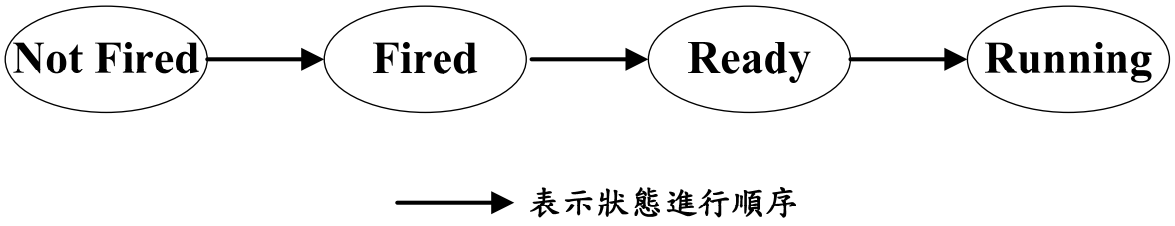


圖 3- 9 Transition 狀態圖

資料來源：本研究

3.6.1 Transition 具有的四種狀態

- **Not Fired**：泛指所有尚未被觸發的 Transition。
- **Fired**：已經被觸發但尚未鎖定 Place 的 Transition，置入於 Priority Queue。
- **Ready**：已經鎖定 Place 但尚未取得 CPU 資源的 Transition，置入 Rnnable Queue。
- **Running**：取得 CPU 資源的 Transition，將從 Ruunable Queue 中移出執行。

3.6.2 Transition 的狀態轉換

- **Not Fired to Fired**：

當一開始模擬時，會評估所有 Transition 的 Fire Rule；或當某個 Transition 執行完後，會觸發其他 Transition 時，將會重新評估之。因此，當 Fire Rule 評估後為成立時，Transition 將由 Not Fired 轉換為 Fired 狀態。

- **Fired to Ready：**

Transition 要從 Fired 轉換為 Ready 狀態，必須確認此 Transition 的 Priority 於 Priority Queue 中為最小，同時必須確定可以取得並且鎖定 Place。此外，必須確保 Ready 或者 Running 的 Transition，其能觸發的 Transition 不能發生搶先的動作，換言之，此被觸發的 Transition 之 Priority 必須大於該狀態轉換的 Transition 的 Priority。

- **Ready to Running：**

當 Transition 依照上述狀態轉換，且取得 CPU 資源時，將依據 FIFO 的原則，從 Ruunable Queue 內取出 Transition 並且執行。當執行完畢後，若會觸發其他 Transition，將重新評估與其相鄰的 Transitions。

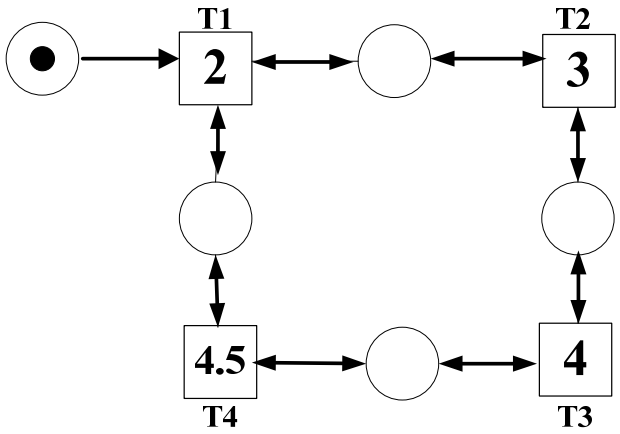


圖 3- 10　簡單網路示意圖

資料來源：本研究

圖 3-10 為一個簡單的生化網路，假設 Transition T1 到 T4 為尚未觸發狀態(Not Fired 狀態)，經過 Fire Rule 判斷後，被觸發的 Transition 將置入於 Priority Queue 中，並且會依照 delay time 由小至大排列，則：

Priority Queue [T1, T2, T3, T4]，此時為 Fired 狀態

接著從 Priority Queue 內取中最小值 T1，確定它取得且鎖定 Place 後，將其放入

Runnable Queue，則：

Priority Queue [T2, T3, T4]

Runnable Queue [T1]，此時的 T1 為 Ready 狀態

當 Runnable Queue 取得 CPU 資源後，Runnable Queue 將依據 FIFO 的特性，取出 T1 執行(Running 狀態)，待 T1 執行完畢後，可能會觸發相鄰的 T2 與 T4，經過檢查皆在 Priority Queue 內，則繼續取出最小值準備執行，因此取出 T2 準備轉換狀態為 Ready，則：

Priority Queue [T3, T4]

Runnable Queue [T2]，此時的 T2 為 Ready 狀態

待 T2 執行完畢後，經由檢查將可能觸發 T1 與 T3。因為 T1 不在 Priority Queue 內，經過計算，其新的累積時間為：2 + 3 = 5，並將其插入(Insert)於 Priority Queue 中；又 T3 已經在 Priority Queue 中且為最小值，所以將把 T3 置入 Runnable Queue 內，故 T3 為 Ready 狀態，則：

Priority Queue [T4, T1]

Runnable Queue [T3]，此時的 T3 為 Ready 狀態

待 T3 執行完畢後，可能會觸發 T2 與 T4，經過計算後發現新的累積時間如下：T2 = 4 + 3 = 7；而原本的 Priority Queue 中已存在 T4 與 T1，其累積時間分別為 T1 = 5 和 T4 = 4.5，又將 T2 插入 Priority Queue 中，則 Priority Queue 結果如下：

Priority Queue [T4, T1, T2]

因此從 Queue 中取出最小值 T4 放入 Runnable Queue 準備執行，則目前兩個 Queue 為：

Priority Queue [T1, T2]

Runnable Queue [T4]，此時的 T4 為 Ready 狀態

待 T4 執行完畢後，檢查被觸發的可能有 T1 與 T3，依此類推。

3.7 系統內的互動情形

圖 3-11 系統循序圖，首先使用者透過 GUI 呼叫 build 函式，這個函式將會產生需要模擬的 Transitions，同時會產生一個 Kernel。產出的 Transitions 會藉由 Parser 剖析驗證 Fire Rule，也同時向 Kernel 作註冊的動作。接著，開始執行模擬，由 Kernel 依據事先定義好的演算法與排程規則做處理，分別執行著 Main thread 和 Executor threads，最後待模擬結束並回傳結果。

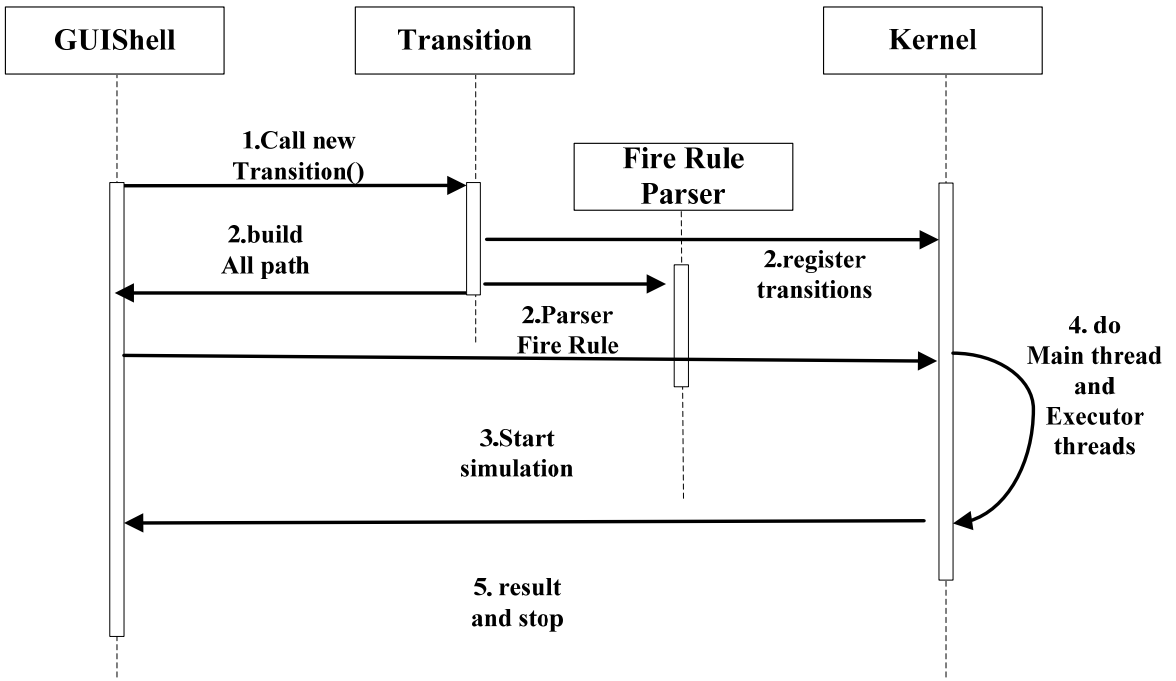


圖 3- 11 系統內之互動情形

資料來源：本研究

3.8 資料庫綱要之設計

圖 3-12 為資料庫的實體－關係模型(E-R Model)，本系統以代謝網路為例，設計共有三個資料表：action、computationConstant 和 defaultAmount。

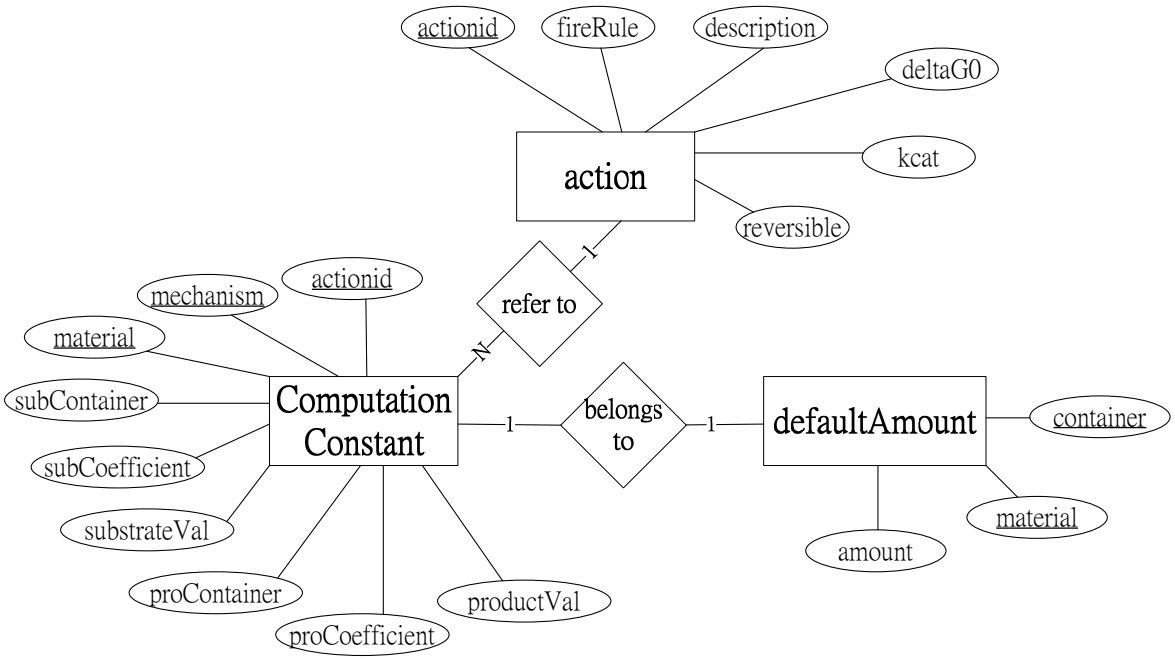


圖 3- 12 資料庫的 E-R Model

資料來源：本研究

action 與 computationConstant 之間有「refer to」的關聯，表示每一個 action 都可以對應一至多個 computationConstant，屬於一對多關係；computationConstant 與 defaultAmount 具有「belongs to」的關係，也就是 computationConstant 將會擁有一個 defaultAmount，屬於一對一的關係。以下圖 3-13、圖 3-14 與圖 3-15 為資料庫表格定義。

```
create table action   --建立 action 表格
(
   actionid varchar(100) not null,   --反應的識別代碼
   fireRule varchar(255) not null,   --觸發反應的條件
   description varchar(200) null,   --關於此反應的環境、條件、物種等描述
   deltaG0 float not null default 1   --計算平衡常數所需之參數
   kcat float not null default 1,   --單位時間中酵素可將反應物轉換成產物的最大數量
   reversible char(1) not null default '0',   --反應是否可逆
   primary key(actionid)   --設 actionid 為主鍵
)
```

圖 3- 13　action 的資料庫表格定義

資料來源：本研究

```
create table defaultAmount   --建立 defaultAmount 表格
(
   container varchar(100) not null,   --物質所在之容器
   material varchar(100) not null,   --物質之名稱
   amount float not null default 0,   --物質之濃度(量)
   primary key(container, material)   --設 container, material 為主鍵
)
```

圖 3- 14　defaultAmount 的資料庫表格定義

資料來源：本研究

```
create table computationConstant    --建立 computationConstant 表格
(
    mechanism varchar(10) not null,    --計算物件之名稱
    actionid varchar(100) not null, --反應的識別代碼
    material varchar(100) not null,    --參與反應的物質
    subContainer varchar(100) not null default '',    --反應物所在容器
    subCoefficient int not null default 1,    --反應物係數
    substrateVal float not null default 0,    --反應物反應常數
    proContainer varchar(100) not null default '',    --產物所在容器
    proCoefficient int not null default 1,    --產物係數
    productVal float not null default 0,    --產物反應常數
    primary key(subContainer, mechanism, actionid, material),    --設 subContainer,
                                                                  mechanism,
                                                                  actionid, material 為
                                                                  主鍵
    foreign key(actionid) references action,    --關聯 action 表格中的 actionid
)
```

圖 3- 15　ComputationConstant 的資料庫表格定義

資料來源：本研究

第四章　系統實作

4.1 Shell

Shell 是以 Class GUIShell 為主，Class PlaceList 和 Class Plotter 為輔所構成的使用者介面，使用者將透過這個介面與 Kernel 產生互動。因此本節將介紹關於 Shell 的相關函式。圖 4-1 為 GUIShell 所相關的類別關聯圖。

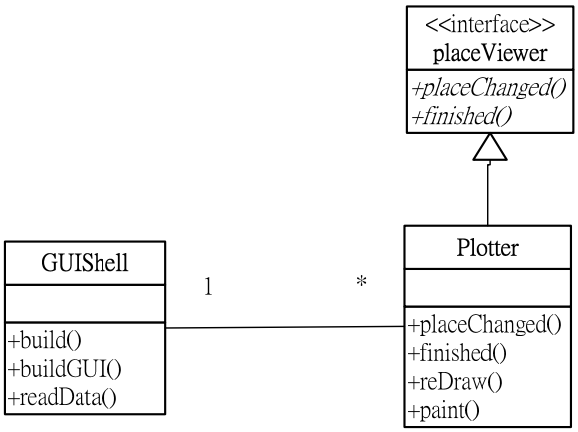


圖 4- 1　GUIShell 等類別關聯圖

資料來源：本研究

Class GUIShell 是系統的進入點，因此在 main( ) method 內除了一些連結資料庫所需的相關設定之外，還有呼叫 GUIShell 建構子，因此建構子需要一個 connection 作為參數，用來連接相關的資料庫。最後，GUIShell 的物件產生時，將會呼叫 init( ) method 把整個 GUI 的環境建立起來。

透過 init( ) method，將呼叫 buildGUI( ) method，這個 method 將把整個 GUI 的畫面建立起來，圖 4-2 為 GUIShell 的範例圖。這個 method 將會建立三個區塊：drawArea、optionArea 和 placeArea。drawArea 作為結果的呈現，所有欲觀察的 place 皆會在此以線圖的方式呈現；optionArea 為功能選單，目前提供了 build(建立反應路徑)、save(儲存參數檔案)、load(讀取參數檔案)、start(開始模擬)、stop(停止模擬)和 exit(離開)等功能；placeArea 則會產生與此反應路徑所相關的所有 place，使用者可以依照需求勾選

此 place 是否為常數？是否欲觀察？以及更改它的初始值，如圖 4-3。

當 listener 收到 build 的 event 後，將呼叫 build( ) method，這個 method 會透過 readData( ) method 建立模擬的反應路徑，同時也會產生一個 Kernel 物件，這個 Kernel 物件將會對所有參與此次模擬的 Transition 作註冊的動作。Class PlaceList 和 Class Plotter 會依照使用者的決定而產生對應的勾選物件和繪圖物件於 GUI 上。

因此，readData( ) method 會在 listener 收到 build 的 event 後開始讀取資料，建立模擬的環境。這個函式可以依照所讀取檔案格式的不同，繼承 Class GUIShell 時而覆寫之，本研究實做的方式是從關聯式資料庫讀取資料，藉由這個函式，開始產生所需的 Transition 或其他的計算物件，接著針對每一個 Transition 或計算物件的 place，開始作參數設定的動作，如此一來將可以把整個模擬環境建立。

當 listener 收到 start 的 event 後，必須先確保模擬環境已經建立，此時會呼叫 initPlace( ) method，這個函式會對該 place 的原點時間與初始量作紀錄，接著在 Thread 取得 CPU 資源時，即可開始模擬。下表 4-1 為 Class GUIShell 函式的整理與描述。

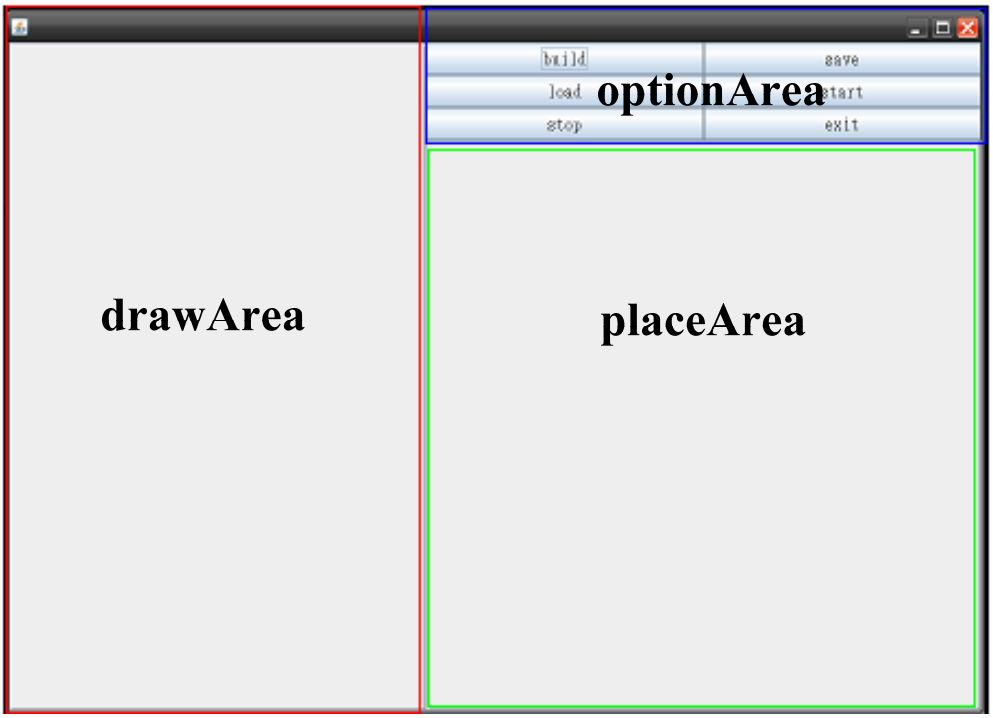


圖 4- 2　GUIShell 範例圖之一

資料來源：本研究

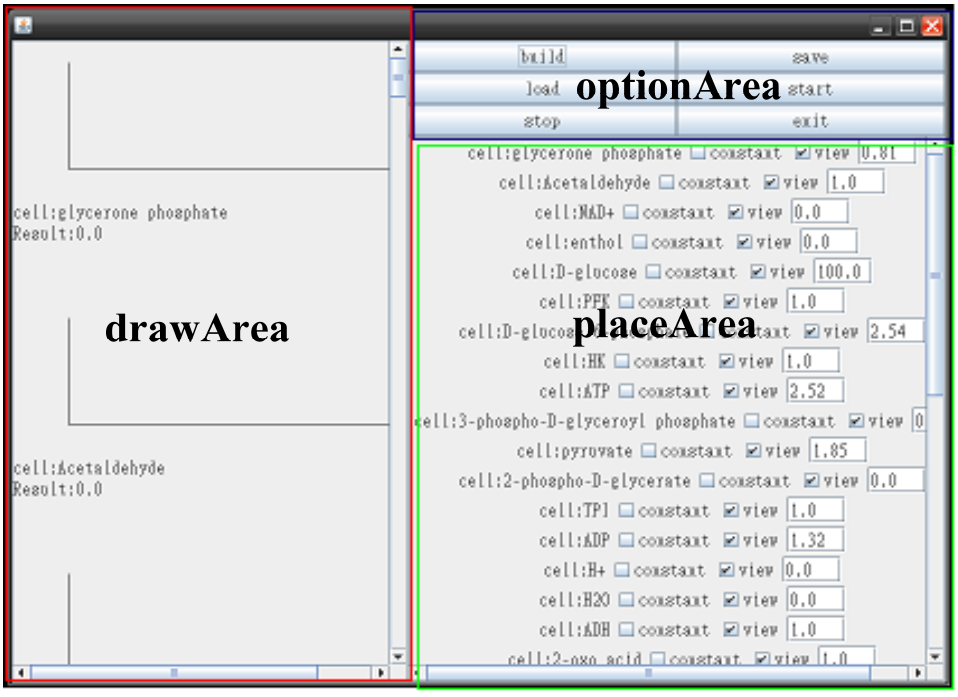


圖 4- 3　GUIShell 範例圖之二

資料來源：本研究

表 4- 1 GUIShell 函式整理

函式	描述
static void main( )	程式進入點，除了設定資料庫連線外，同時將建立 GUIShell 物件。
void readData( )	讀取所需的模擬資料，使用者可以依據建模檔案格式的不同，可覆寫此函式。
void build( )	呼叫 readData( )建立環境；產生 Kernel 物件用以註冊的 Transition；透過 Class PlaceList 和 Class Plotter 建立實際會勾選和呈現結果的物件。
void initPlace( )	初始化 place 的值，紀錄當原點時間時，place 的初始量為多少。
void init( )	呼叫 buildGUI( )，建立使用者介面。
void buildGUI( )	關於 GUI 的設計與功能等建立。

資料來源：本研究

4.2 Transition

在 Petri Nets 的架構中，主要由 Transition 和 place 所構成的網路，因此本小節將分別討論關於 Transition 與 place，包括了 Class Transition、Class Equilibrium、Class TransitionParam 以及 Class Place、Interface PlaceViewer，

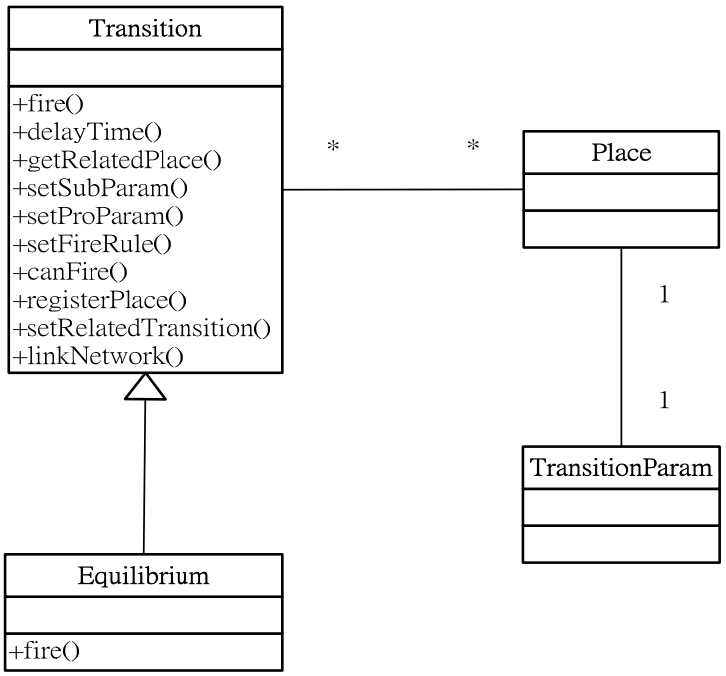


圖 4- 4 Transition 等類別關聯圖

資料來源：本研究

Transition 在系統扮演著計算的角色，也是系統排程的基本單元。因此 Class Transition 的建構子除了一些需要的參數如代號(tid)、關於此 Transition 的描述(des)、完成 Transition 所需的時間(delay)，最重要的，還需要一個 Kernel 參數，將會對此 Transition 作註冊的動作。delayTime( ) method 為計算 Transition 完成所需的時間並回傳該值，這個值將作為所有 Transition 等待執行的排程依據。此外，使用者可以依據需求覆寫這個函式，以代謝網路的反應為例，其 delayTime 為轉化數(turnover number)

的倒數，即 1/kcat。

fire( ) method 為量化的函式，它需要一個參數 timeAt，意謂此刻 Place 其量的變化情形。在 Transition 的計算方式，fire( ) method 必須計算一個 delta，這個值代表著至少可以轉置的比例，藉由所有的 input Place 所擁有的 Token 數目除以 Arc 的權重即可求得 delta，再經由比較找出最小值，最後 input place/output Place 將依照 delta 作增加/減少 Token。以圖 4-5 為例，分別計算 P1 與 P2 的 delta 與圖 4-6，轉置後的結果：

因為 P1's delta = 2 / 2 = 1 且 P2's delta = 1 / 1 = 1，故 delta 設為 1，

所以 P1 = 2 – (2 * 1) = 0，P2 = 1 – (1 * 1) = 0，P3 = 0 + (2 * 1) = 2 則為轉置後的結果。

fire( ) method 也可以依據使用者的需求而覆寫，例如研究者可以實做酵素動力學或化學平衡的計算模型。fire( ) method 有一個布林的回傳值，當回傳 true 時，表示將觸發其他 Transition。

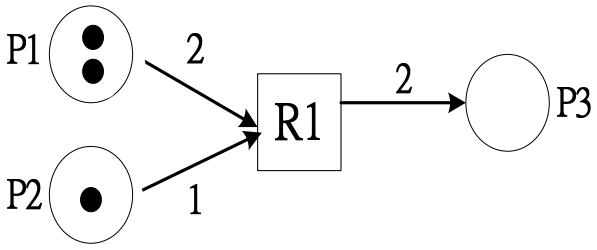


圖 4- 5 Transition 轉置計算

資料來源：本研究

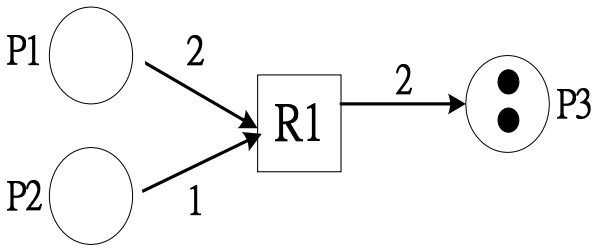


圖 4- 6 Transition 轉置計算後結果

資料來源：本研究

setFireRule( ) method 是處理該 Transition 的 fire rule，從資料庫讀取後當做參數，再經由 Fire Rule Parser 剖析。而 canFire( ) method 會依據剖析的結果判斷此 Transition 是否可以觸發。

setSubParam( ) method 與 setProParam( ) method 為設定該 Transition 相關的 place 參數，透過所在位置(container)、物質名稱(material)和相關係數(coefficient)，藉由 Class TransitionParam 建構該 place 的相關資訊。

getRelatedPlace( ) method 的概念很簡單，因為一個 Transistion 所需的資源也許是獨享，也有可能是必須共有。因此，透過這個函式將可以找出與此 Transistion 相關的資源為何？也因此，整個模擬環境不畢事先定義好 Arc，透過此 method 將可以動態產生 Transition 與 Place 的關聯。而 Kernel 將透過呼叫 registerPlace( ) method，對參與模擬的 Transition，其需求的資源做註冊的動作，因此呼叫 registerPlace( ) method 時，將可以藉由 getRelatedPlace( ) method 取得相關資訊。

setRelatedTransition( ) method 可以讓此 Transition 透過相連的 places，找出與其相關聯的其它 Transitions。而 linkNetwork( ) method 會被 Kernel 所呼叫，透過 place 將相關的 Transitions 互相連結在一起，且 linkNetwork( ) method 會計算出相鄰中反應最快的 Transition。表 4-2 為 Transition 函式的整理與描述。

表 4- 2 Transition 函式整理

函式	描述
boolean fire(double timeAt)	計算模型。參數代表當下時間內所要做的變化，回傳的布林值表示是否會觸發其它 Transitions，使用者可以需求而覆寫。
Double delayTime( )	Transition 所需的完成時間。
Place[ ] getRelatedPlace( )	與此 Transition 相關的 Places
void setSubParam(String subContainer, String material, double subCoefficient)	設定 input Places 的參數。
void setProParam(String proContainer, String material, double proCoefficient)	設定 output Places 的基質。
void setFireRule(String rule)	剖析 Fire Rule。
boolean canFire( )	確認此 Transition 是否可以被觸發。
void registerPlace( )	Place 註冊相關的 Transitions。
void setRelatedTransition( )	找出與本身相關的其它 Transitions。
void linkNetwork( )	連結相關的 Transitions.

資料來源：本研究

除了 Class Transition，還有 Class Place 與 Interface PlaceViewer，圖 4-7 是 Place 與 PlaceViewer 的類別關聯圖。在 Class Place 中，提供了一些基本的函式，讓使用者對 Place 做設定或存取之用，因此將分述如下：

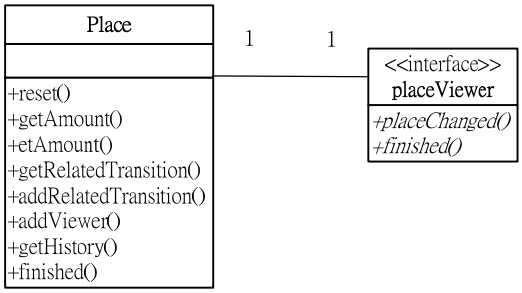


圖 4- 7 Place 等類別關聯圖

資料來源：本研究

reset( )method 會於模擬開始執行時被間接呼叫。當 listener 接受到 start 的 event 時，將會呼叫 initPlace( ) method，此時便會開始對 Place 做初始化的動作，一方面會設定初始值，同時也會將資料存於一個 Vector 中，以便之後做為計算以及繪圖的根據。

getAmount( ) method 可以取得 Place 的量；setAmount( ) method 為設定 Place 的量，可以分成三種情形處理：1.若量為負值，則列印結果並終止程式；2.若該資源為常數型態(意謂著不會隨著時間或條件而增減其量)，則不做任何動作；3.正常情況下，會把時間與量存於一維陣列中(名稱為 history 的 Vector)，作為繪圖之用。

getRelatedTransition( ) method 的目的為回傳一個 Vector，其內容為與此 Place 相關的 Transitions；addRelatedTransition( ) method 則是紀錄與此 Place 相關的 Transitions。至於 addViewer( ) method，也是利用一個 Vector 物件，存放著需要觀察的 Place；getHistory( ) method，透過之前存放於 Vector 的歷史資料，並且回傳此物件；finished( ) method，則是對每個觀察的 place 告知模擬將終止。表 4-3 為 Place 函式的整理與描述。

表 4- 3 Place 函式整理

函式	描述
void reset(double init)	初始化時，紀錄時間原點與 place 的初始值。
double getAmount( )	取得 place 的量。
void setAmount(double time, double newAmount)	設定該時間點下，Place 的量。
Vector getRelatedTransition( )	取得與此 Place 相關的 Transitions。
void addRelatedTransition(Transition t)	註冊與此 Place 相關的 Transitions。
void addViewer(PlaceViewer x)	增加觀察此 Place。
Vector getHistory( )	紀錄著 Place 的不同時間與量。
void finished( )	告知該 Place 即將結束模擬。

資料來源：本研究

4.3 Fire Rule Parser

Fire Rule Parser 是由 Class Token、Class Lexical、Class Evaluate、Class Expression 和 Class SimpleExpression 所組成，圖 4-8 為 Expression 等類別關聯圖，Parser 的運作以 Class Expression 和 Class SimpleExpression 為主，皆實做了 interface Evaluate，將實做 eval( ) method 判斷 Fire Rule 是否成立。

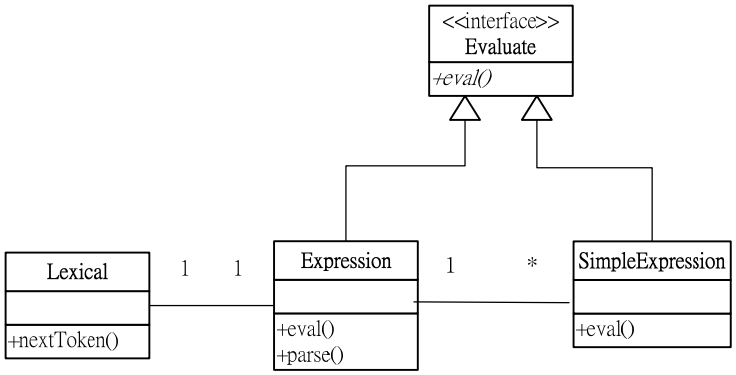


圖 4- 8 Expression 等類別關聯圖

資料來源：本研究

Fire Rule 的表達可以是一個敘述、多個敘述的組成、或簡單的判斷式等，如第三章圖 3-7 所示，不容贅述。Class Exprssion 將提供兩個函式：eval( )method 和 parse( ) method，分別剖析 Fire Rule 和驗證是否可以觸發。

parse( ) method 需要三個參數，分別代表 Kernel、該 Transition 以及其 Fire Rule。Class Expression 藉由 overloading 再定義另一個 parse( )method，除了需要參數 Kernel 和 Transition 外，會建構一個 Lexical 物件，準備分析剛傳進的 Fire Rule。

Class Lexical 提供了 nextToken( ) method，這個函式負責切割 Fire Rule。除了空白字元，將分析 Fire Rule 各個組成的子成份是否為字串、數字、判斷符或括符。而 Class Expression 內的 parse( ) method，將藉由其分析，將 Fire Rule 拆解成最小單位：

Simple Expression，再判斷是否滿足條件。eval( ) method 會回傳一個布林值，當經過判斷後確認觸發條件成立，則回傳 true。Class Token 則定義了字串、數字與各種判斷符的對應。表 4-4 為 Fire Rule Parser 相關的函式整理。

表 4- 4 Fire Rule Parser 函式整理

函式	描述
boolean eval( )	**觸發條件是否成立的判斷。在 Class SimpleExpression 內藉由簡單的條件式判斷後回傳佈林值；Class Expression 內會將各部份的判斷結果以 And 或 Or 計算後回傳布林值。**
Expression parse(Kernel parent, Transition act, String d)	**被呼叫後，將產生一個 Lexical 物件處理 Fire Rule，並且呼叫 parse(Kernel parent, Transition act, Lexical l)。**
Expression parse(Kernel parent, Transition act, Lexical l)	**剖析 Fire Rule。**

資料來源：本研究

4.4 Kernel

Kernel 部份以 Class Kernel 為主體，負責整個系統的資源管理與排程規劃，因此，還需要其它的資料結構配合—Class ActionQueue 負責排程所需的兩種 Queue：Priority Queue 和 Runnable Queue，而搭配 Priority Queue 的資料結構將透過 Class Fheap 與 Class FheapNode。

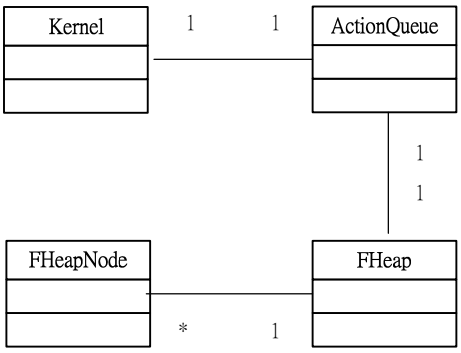


圖 4- 9 Kernel 等類別關聯圖

資料來源：本研究

Class Fheap 為實做 Fibonacci Heap，而 Class FheapNode 則定義了一個節點所需要的資料結構。誠如第三章 3.4 所示，在系統的排程方面，以完成時間當作優先權的依據，因此當 Transition 即將要執行時，會依序從小至大的放入 Runnable Queue，因此需要透過 Extract-Min 的動作。而 Transition 從產生到執行的過程中，每一個時期都有不同的狀態，待執行完畢時，可能會觸發其他 Transition，屆時必須做適當的處理，具有相關功能之資料結構的效能也已於第三章 3.5 比較，因此不容贅述。

Class ActionQueue 提供了一些基本的功能來處理兩種不同的 Queue。insert( ) method，顧名思義就是將 Transition 置入 Queue 中，但是這裡必須作判斷，若欲插入的 Transition 已經確定在 Priority Queue 或 Runnable Queue 中，則作 Decrease-Key 的處理；若不在 Queue 內，則直接加入。

ready( ) method 主要負責確認 Transition 是否可以將 Place 鎖定，deleteMin( ) method 將會用到此函式。deleteMin( ) method 是將最小的 Transition 從 Priority Queue 取出放入 Runnable Queue，在 Runnable Queue 內必須確保 Transition 已經可以鎖定 Place。

deQueue( ) method 說明了當所有的 Transition 已經鎖定資源，且 Thread 也取得 CPU 資源時，將依照 FIFO 的特性取出 Transition 並執行。若 Queue 為空時，且還有 Transition 等待進入，則 Thread 必須釋放 CPU 資源並等待。SetDrained( ) method 的目的是透過改變 Variable drain 設定，當 drain 為 true 時，表示不會再有 Transitions 加入 Queue 中。

doneAction( ) method 會當 Transition 執行完畢時作相關的處理。首先必須將 Transition 所鎖定的資源釋放，接著檢查是否觸發其它反應？若觸發的反應不在 Queue 內，則將其置入。

Class Kernel 主要就是套過上述的資料結構與機制來管理資源與執行 Transitions，並且提供了一些函式如下所述。 SetTimeLimit( ) method 會對 Variable timeLimit 作設定，主要是提供使用者對系統模擬的時間做調整，使用者決定一個終止時間，可避免不必要的計算資源的浪費。

addCron( ) method 提供了在實務上一些特殊的應用，例如某些 Transition 會在特定的時間內做固定的觸發反應，是一種規律性的動作，因此，此函式提供了這樣的服務。使用者可以設定該 Transition 的起始時間，包括多久觸發一次，以及觸發的次數，在生化網路上，某些酵素可能會在某個固定時間點做訊號傳遞的工作；在藥物投試的研究，也可以透過此觀察某個節點的測試變化。

registerTransition( ) method，因為 Kernel 負責整體的排程管理，因此必須對參與此模擬的所有 Transitions 做註冊的動作。

getPlace( ) method 這裡採用了 overloading 的技術，分別定義了兩種不同的參數：只透過 material 或透過 container 與 material 的組合。只透過參數 material 的 getPlace( )

method，可以用在剖析 Fire Rule 時，在 Fire Rule 的表達式中找出相關的 place，並產生對應的 place 物件；而透過 container 與 material 的組合可以用於 Transition 要設定相關的 place 時，產生所需的 place 物件。

peekPlace( ) method 與 getPlace( ) method 相同，也採用了 overloading 的技術，但是兩者最大的不同點是 peekPlace( ) method 不會產生任何的 place 物件。其可以透過參數 id 或者是 透過 container 與 material 的組合找出對應的 place，在對 place 做初始化的動作時，因為整個 place 已經存在，只需對其的初始值做設定，因此不需要再產生新的 place 物件，故透過 peekPlace( ) method 可以查看該 place，並對其做設定。

getAllPlaces( ) method 會取得所有參與反應的 places，在 kernel 中存在一個 Hashtable，用來存放所有的 places，因此透過轉換存至一個一維陣列中，將可以取得所以的 places。表 4-5 為 Kernel 相關的函式整理。

表 4- 5 Kernel 函式整理

函式	描述
void setTimeLimit(double lim)	使用者可以設定系統模擬的整體時間。
void addCron(double start, double period, double times, Transition tran)	使用者可以依照需求對 Transition 做相關設定，start 表示開始的時間；period 表示觸發間隔；times 表示觸發次數；tran 表示此 transition。因此該 transition 會在特定的系統時間內檢查是否被觸發。
void registerTransition(Transition t)	transition 對 Kernel 做註冊。
Place getPlace(String material)	此函式用於剖析 Fire Rule 時，可以“：”找出所在的 Container 與 material 再透過 getPlace(String container, String material) 產生對應的 place 物件。
Place getPlace(String container, String material)	由 Container 與 material 產生對應的 place 物件。
Place peekPlace(String id)	透過 id 查看此 place。
Place peekPlace(String container, String material)	透過 container 與 material 查看此 place。
Place[] getAllPlaces( )	取得所有相關的 places。

資料來源：本研究

4.5 系統佈署

本系統的環境架構如下：處理器為 Intel Pentium 4 CPU 3.20GHz、記憶體為 1.46GB、作業系統為 Microsoft Windows XP、資料庫為 Microsoft SQL Server 2005。

在環境佈署方面，如圖 4-10 所示，透過資料庫選取網路，以及環境因子與實際數據作為假設，模擬之後的結果與實際情況做比較，進而修正系統與模型。

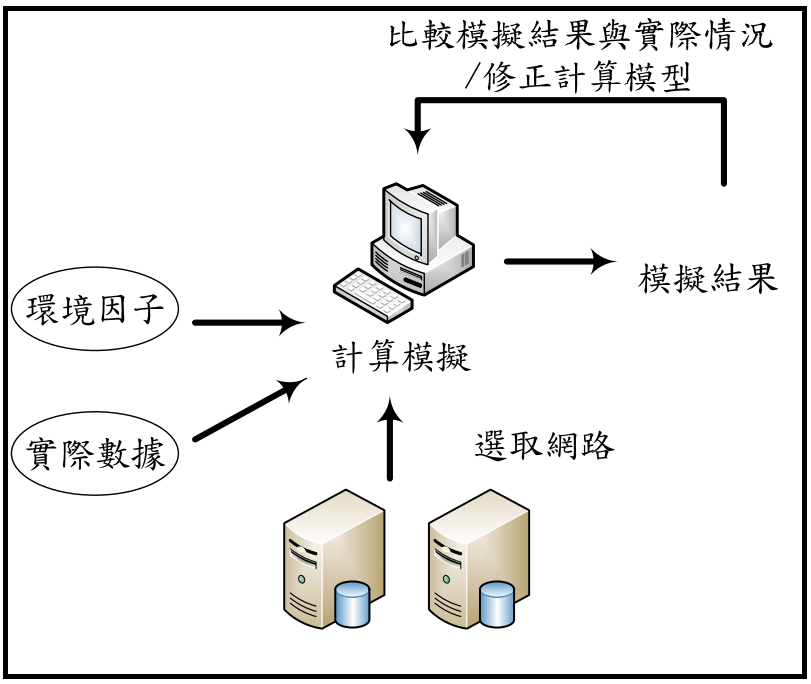


圖 4- 10 系統佈署圖

資料來源：本研究

4.6 模擬畫面與結果呈現

下圖分別為圖 4-11 為系統的啟動畫面、圖 4-12 為模擬畫面和圖 4-13 的結果呈現；醣酵解作用之測試結果請見附錄Ａ。

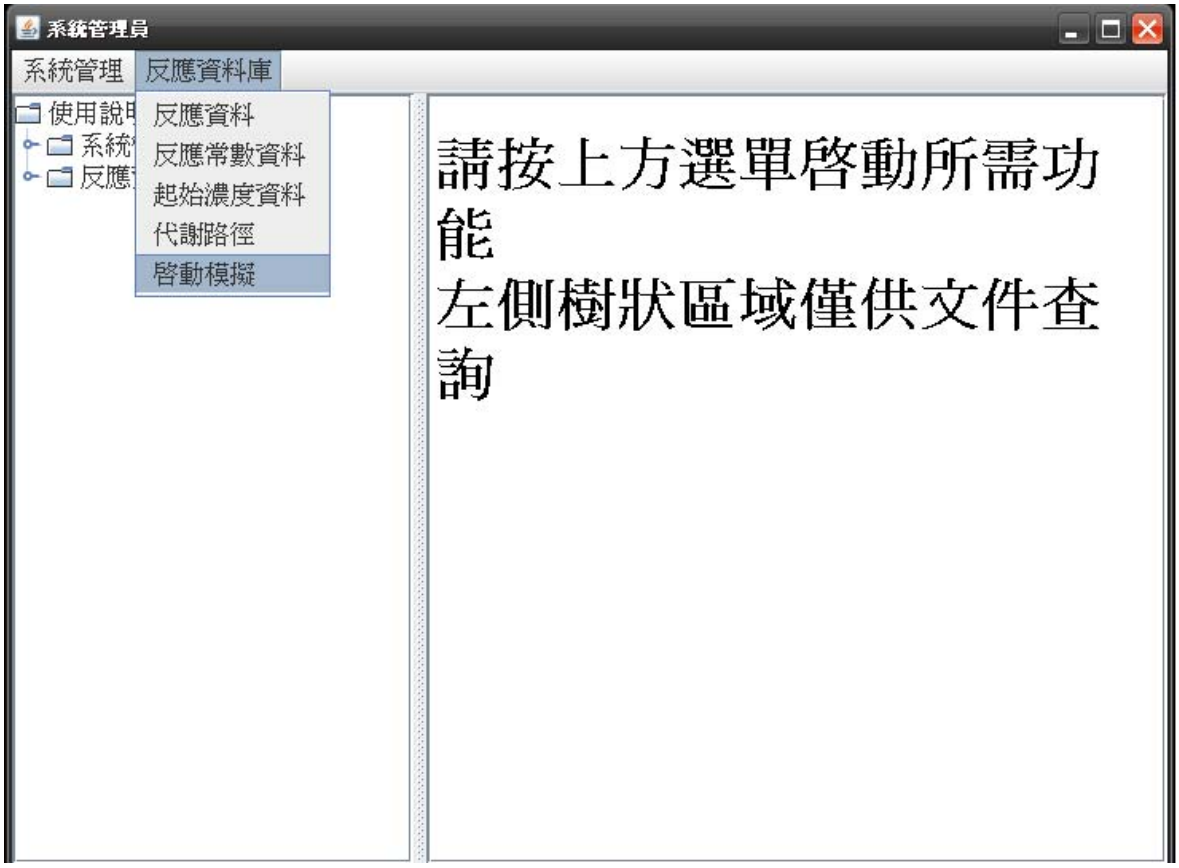


圖 4- 11 啟動畫面

資料來源：本研究

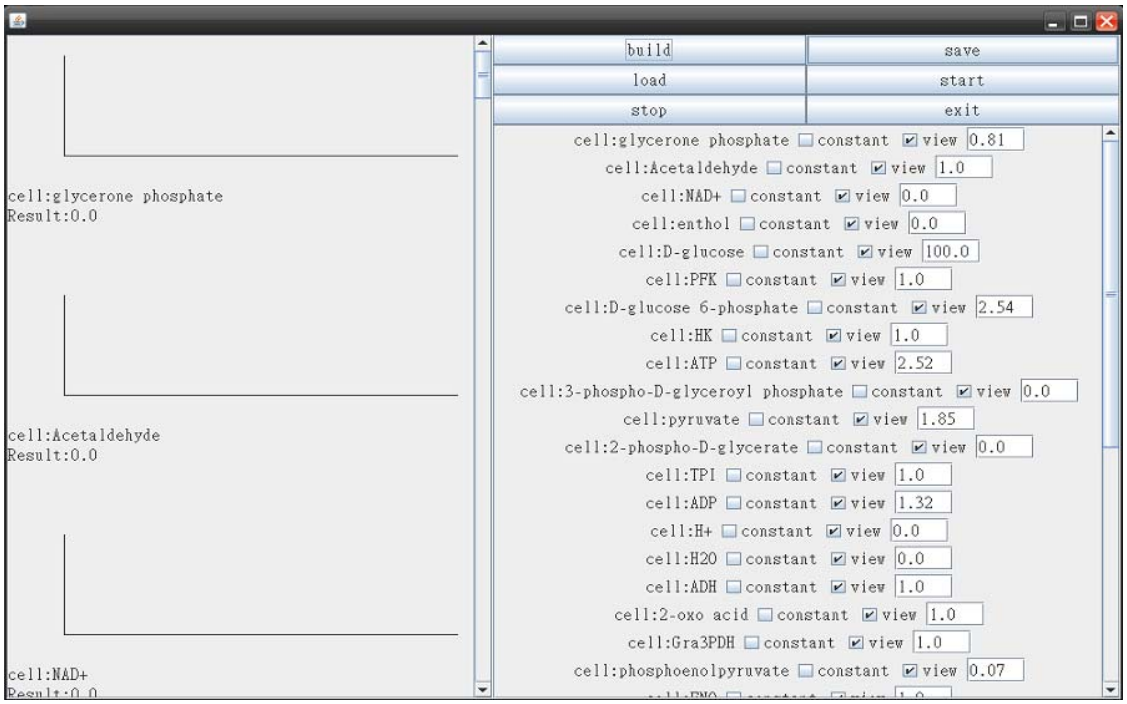


圖 4- 12　模擬畫面

資料來源：本研究

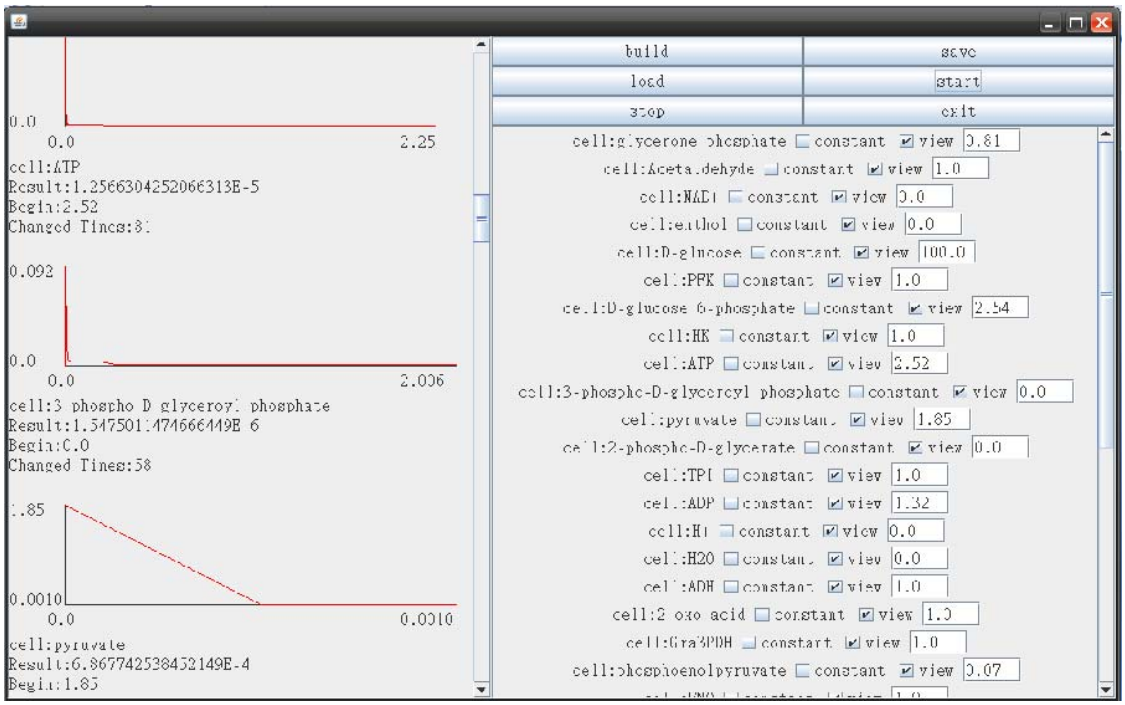


圖 4- 13　模擬結果

資料來源：本研究

第五章　結論

5.1　結論

本研究欲開發一個具有泛用性的模擬平台，因此模擬策略的選擇將以具有泛用性的工具為主，除了傳統模擬所使用的微分方程外，另一個為 Petri Nets。

雖然可以藉由微分方程作為模擬的策略，但是往往會根據研究人員對現象觀察的詮釋不同，而提出不同的模型。對於動輒數以萬計的網路節點結構而言，透過數學建構具有相當大的困難，此人員不僅需要堅實的數學背景而且茲意地對網路新增或減少節點時，數學方程式必須重新撰寫；而對於所需代入或估計的參數，收集的困難度更不在話下，更別談及之後的計算推演，工程之浩大可見一斑。

因此，本研究採用了 Petri Nets，作為主要的建模用語，相較於其他具有圖形表達能力的模型，此工具不僅具有數學驗證模型正確性的能力，同時更可以表達網路的動態情形。

Low level Petri Nets 雖然具有圖形化，可以表現出系統整體的動態情形，也具有驗證模型正確性的能力，但是因為 Petri Nets 本身具有一些缺陷，例如必須事先定義好 Arc 方向，且 Arc 為靜態連結，換言之，使用者必須先將整個網路事先定義完備，才能開始模擬；在 Transition 方面，不但只有一種運作機制，而且僅限於離散型別的 Token 轉置；至於在現實世界中常會面臨的同步問題，Low level Petri Nets 也是無從解釋。

其他研究希望藉由 High level Petri Nets 的改良機制，來處理上述問題，仍屬於治標不治本的方法，因為問題的根源在於 Transition 的語意只有一種，使得上述問題皆無法解決，故本研究從問題的根源，修正了 Transition 的運作機制，導入了物件導向的概念，讓使用者可以依據需求而定義不同的 Transitions，表 5-1 為與其他 Petri Nets 之比較。

表 5- 1 與其他 Petri Nets 之比較

	Timed Petri Nets	Stochastic Petri Nets	Coloured Petri Nets	Hybrid Petri Nets	Ours
	問題：Transition 可以改變其運作機制				
					●
	生化網路現象的模擬				
Place 同時觸發 Transitions		◎	◎		●
逆反應的處理			◎		●
消耗連續型別的 Token			◎	◎	●
Arc 為動態連結					●

註：◎表示雖然針對部份問題解決，但結果仍不符合實際現象

●表示可以完全解決

資料來源：本研究之整理

因此，使用者可以透過 Java 的繼承，自由延伸其塑模方式；傳統的 Petri Nets 必須事先定義 Arc 的關聯與方向性，才能開始模擬，本研究透過動態 Arc 與資料庫的配合，相較於其它模擬策略，將可以快速建立模擬環境；最後是透過 Java 的技術支援核心硬體，將可以大幅縮短模擬時間。

5.2 未來研究方向

1. 應於生化網路之研究

在生化網路的研究上，期望可以擴大網路的模擬。這裡將分為水平面與垂直面：

- 水平面－

針對每一個領域的網路(如代謝網路、訊號傳導網路、基因調控網路、蛋白質交互作用網路等)做定性與定量的模擬與分析。

- 垂直面－

期望能將各種不同領域的網路所結合，由上而下的一連串地全面性模擬。除此之外，可以針對實驗的需求而實做更多不同的 Transitions，表達不同的反應計算模型。

2. 其他領域之應用

應用 Petri Nets 所建構的模擬環境為一種網路形式，因此可以將本研究延伸至一般網際網路的通訊協定做探討。是否可以藉由修改 Transition 的機制，實做不同的通訊協定，以及收集各種資料來源，如 web、ftp、telnet…等，來模擬整個網路環境，進而針對結果做分析與探討。

參考文獻

[1] Alves,, R. Antunes, F. and Salvador, A., "Tools for kinetic modeling of biochemical networks", *Nature Biotechnology*, Vol.24, No.6, 2006, pp.667-672.

[2] Bonabeau, E., "Agent-based modeling: methods and techniques for simulating human systems", *PNAS*, Vol.99, NO.3, 2002, pp.7280-7287.

[3] Chen, M. and Hofestaedt, R., "Quantitative Petri Net model of gene regulated metabolic networks in the cell", *In Silico Biology*, Vol.3, No.0029, 2003, pp.347-365.

[4] Davidov, E.J., Holland, J.M., Marple, E.W. and Naylor, S., "Advancing drug discovery through systems biology", *drugdiscoverytoday*, Vol.8, 2003, pp. 175-183.

[5] Davis, F.D., "Perceived usefulness, perceived ease of use, and user acceptance of information technology", *MIS Quarterly*, Vol. 13, No. 3, 1989, pp. 319-340.

[6] Davis, F.D., Bagozzi, R.P. and Warshaw, P.R., "User acceptance of computer technology: A comparison of two theoretical models", Management Science, Vol. 35, No.8 (Aug., 1989), pp. 982-1003.

[7] Emonet, T., Macal, C.M., North, M.J., Wickersham, C.E. and Cluzel, P., "AgentCell: a digital single-cell assay for bacterial chemotaxis", Bioinformatics, Vol.21, No.11, 2005, pp.2714-2721.

[8] Genrich, H., Küffner, R. and Voss, K., "Executable Petri Net models for the analysis of metabolic pathways", *int J STTT*, Vol.3, No.4, 2001, pp.394-404.

[9] Goss, P.J. and Peccoud, J., "Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets", Proc. Natl. Acad. Sci. USA, Vol.95, 1998, pp. 6750-6755.

[10] Heiner, M., Koch, I. and Voss, K., "Analysis and simulation of steady states in metabolic pathway with Petri Nets", *in Third WorkShop and Tutorial on Practical*

Use of Colored Petri Nets and CPN, 2001, pp.15-34.

[11] Heiner, M., Koch, I. and Will, J., "Model validation of biological pathways using Petri Nets-demonstrated for apoptosis", BioSystems, Vol.75, 2004, pp.15-28.

[12] Hood, L., "Systems biology: Integrating technology, biology, and computation", *Mechanisms of Ageing and Development*, Vol.124, 2003, pp. 9-16.

[13] Hood, L., Heath, J.R., Phelps, M.E. and Lin, B., "Systems biology and new technologies enable predictive and preventative medicine", *SCIENCE*, Vol.306, 2004, pp. 640-643.

[14] Ideker, T., Galitski, T., and Hood, L., "A new approach to decoding life: systems biology", *Annual Review of Genomics and Human Genetics*, Vol.2, 2001, pp. 343-372.

[15] Jensen, K., "Coloured Petri Nets: A high-level language for system design and analysis", *Lecture Notes in Computer Science*, Vol.483, 1991, pp.342-416.

[16] Jensen, K., "Coloured Petri Nets", IEEE, 1993, pp.5/1-5/3.

[17] Khan, S., Makkena, R., McGreary, F., Decker, K., Gillis, W. and Schmidt, C. "A mulit-agent system for the quantitative simulation of biological networks", *AAMAS*, 2003, pp.385-392.

[18] Kier, L.B., Cheng, C.K., Testa, B. and Carrupt, P.A., "A cellular automata model of enzyme kinetics", *Journal of Molecular Graphics*, Vol.14, 1996, pp.227-231.

[19] Kier, L.B., Cheng, K.K. Testa, B. and Carrupt, P.A., "A cellular automata model of diffusion in aqueous systems", *Journal of Pharmaceutical Sciences*, Vol.86, No.7, 1997, pp.774-778.

[20] Kitano, H. , "Systems biology: a brief overview", *SCIENCE*, Vol.295, 2002, pp. 1662-1664.

[21] Koch, I., Junker, B.H. and Heiner, M., ".Application of Petri net theory for modelling

and validation of the sucrose breakdown pathway in the potato tuber", *Bioinformatics*, Vol.21, No.7, 2005, pp.1219-1226.

[22] Koch, I., Schuster, S. and Heiner, M., "Simulation and analysis of metabolic networks using time-dependent Petri Nets", *Proc. Of the German Conference on Bioinformatics(GCB 99)*, Hannover, 1999

[23] Kurnaz, I.A., "Biochemical modelling tools and applications to metabolic engineering", *Turkish Journal of Biochemistry*, Vol.30, No.2, 2005, pp.200-207.

[24] Li, C., Ge, Q.W., Nakata, M., Matsuno, H, and Miyano, S., "Modelling and simulation of signal transductions in an apoptosis pathway by using timed Petri Nets", *J. Biosci.*, Vol.32, No.1, 2007, pp.113-127.

[25] Lollini, P.L., Motta, S. and Pappalardo, F., "Discovery of cancer vaccination protocols with a genetic algorithm driving and agent based simulator", *BMC Bioinformatics*, Vol.7, No.352, 2006, pp.352

[26] Marsan, M.A., Balbo, G., Conte, G., Donatelli, S. and Franceschinis, G., "Modelling with Generalized Stochastic Petri Nets", New York: John Wiley & Sons, 1994

[27] Materi, W. and Wishart, D. S., "Computational systems biology in drug discovery and development methods and applications", *drug discovery today*, Vol.12, No.7/8, 2007, pp.295-303.

[28] Matsuno, H., Aoshima, H., Doi, A., Tanaka, Y., Matsui, M. and Miyano, S., "Biopathways representation and simulation on hybrid functional Petri net", In Silico Biology, Vol.3, 2003, pp.389-404.

[29] Matsuno, H., Li, M. C. and Miyano, S., "Petri Net based descriptions for systematic understanding of biological pathways", *IEICE TRANS. FUNDAMENTALS*, Vol.E89-A, No.11, 2006, pp.3166-3174.

[30] Mendes, P., "Biochemistry by numbers: simulation of biochemical pathways with

Gepasi 3", *Trends Biochem. Sci.*, Vol.22, No.9, 1997, pp.361-363.

[31] Miyano, S., and Matsuno, H., "How to model and simulate biological pathways with Petri Nets – A new challenge for systems biology ", *in the 25th International Conference on Application and Theory of Petri Nets*, 2004

[32] Murata, T., "Petri Nets: Properties, analysis and applications", *Proceedings of the IEEE*, Vol.77, No.4, 1989, pp.541-583.

[33] Nagasaki, M., Doi, A., Matsuno, H. and Miyano, S., "A versatile Petri Net based architecture for modeling and simulation of complex biological processes", *Genome Informatics*, Vol.15, No.1, 2004, pp.180-197.

[34] Narahari, Y., Suryanarayanana, K. and Subba-Reddy, N.V., "Discrete event simulation of distributed system using stochastic Petri Nets", *IEEE*, 1989, pp.622-625.

[35] Peleg, M., Tu, S., Manindroo, A. and Altman, R. B., "Modeling and analyzing biomedical processes using workflow / Petri Net models and tools", *IOS Press*, 2004, pp.74-78.

[36] Pinney, J.W., Westhead, D.R. and Mcconkey, G.A., "Petri net representations in systems biology", *Biochemical Society*, Vol.31, 2003, pp.1513-1515.

[37] Popova-Zeugmann, L. and Schlatter, D., "Analyzing paths in time Petri Nets", *Fundamenta Informaticae*, Vol.34, 2005, pp.1-17.

[38] Popova-Zeugmann, L., Heiner, M. and Koch, I., "Time Petri Nets for modelling and analysis of biochemical networks", *Fundamenta Informaticae*, Vol.67, 2005, pp.147-162.

[39] Ramchandani, C., "Analysis of asynchronous concurrent systems by timed Petri Nets.", *Phd Dissertation, MIT*, 1974

[40] Reddy, V.N., Liebman, M.N. and Venkatramana, M.L., "Qualitative analysis of

biochemical reaction systems", *Comput. Bio. Med.*, Vol.26, No.1, 1996, pp. 9-24.

[41] Runge, T., "Application of coloured Petri Nets in systems biology", *in Fifth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, 2004, pp.77-96.

[42] Sackmann, A., Heiner, M. and Koch, I., "Application of Petri Net based analysis techniques to signal transduction pathways", *BMC Bioinformatics*, Vol.7, No.482, 2006.

[43] Schaff, J. and Loew, L.M., "The virtual cell", *Pacific Symposium on Biocomputing*, Vol.4, 1999, pp.228-239.

[44] Sozinova, O., Jiang, Y., Kaiser, D. and Alber, M., "A three-dimensional model of myxobacterial aggregation by contact-mediated interactions", *The Nation Academy of sciences of the USA*, Vol.102, No.32, 2005, pp.11308-11312.

[45] Tomita, M., Hashimoto, K., Takahashi, K., Shimizu, T.S., Matsuzaki, Y., Miyoshi, F., Saito, K., Tanida, S., Yugi, K., Venter, J.C. and Hutchison III, C.A., "E-CELL: Software environment for whole-cell simulation", *Bioinformatics*, Vol.15, No.1, 1999, pp.72-84.

[46] Tsavachidou, D. and Lieban, M.N., "Modeling and sinulation of pathways in menopause", *Journal of the American Medical Informatics Association*, Vol.9, No.5, 2002, pp.461-471.

[47] Vallurupalli, V. and Purdy, C., "Agent-Based modeling and simulation of biomolecular reactions", *SWPS*, Vol.8, No.2, 2007, pp.185-196.

[48] Walker, D.C., Hill, G., Wood, S.M., Samllwood, R.H. and Southgate, J. "Agent-Base computational modeling of wounded epithelial cell monolayers", *IEEE Transactions on Nanobioscience*, Vol.3, No.3, 2004, pp.153-163.

[49] Wishart, D.S., Yang, R., Arndt, D., Tang P. and Cruz, J., "Dynamic cellular automata:

n alternative approach to cellular simulation", *In Silico Biology*, Vol.5, 2004, pp.139-161.

[50] Xia, X.Q. and Wise, M.J., "DiMSim: A discrete-event simulator of metabolic networks", *J. Chem. Inf. Comput. Sci.*, Vol.43, No.3, 2003, pp. 1011-1019.

[51] Zorzenon dos Santos, R.M. and Coutinho, S., "Dynamics of HIV infection: a cellular automata approach", *Physical Review Letters*, Vol.87, No.16, 2001, pp.168102.

[52] 李允中、王小璠、蘇木春，「模糊理論及其應用」，*全華*，2003。

[53] 陳郁雯，「電腦模擬對學生學習成效影響之後設分析」，碩士論文，國立新竹師範學院國民教育研究所，2004。

[54] 彭曉芳, 金一栗, 梁逸曾, 盧紅梅, "生物化學動力學網絡體系的計算機模擬", *Computers and Applied Chemistry*, Vol.23, No.4, 2006, pp.317-332.

[55] 趙邦杰, 張志讓. 趙晟, "生物學中的數學建模", *Journal of mathematics for technology*, Vol.18, No.6, 2002, pp.1-8.

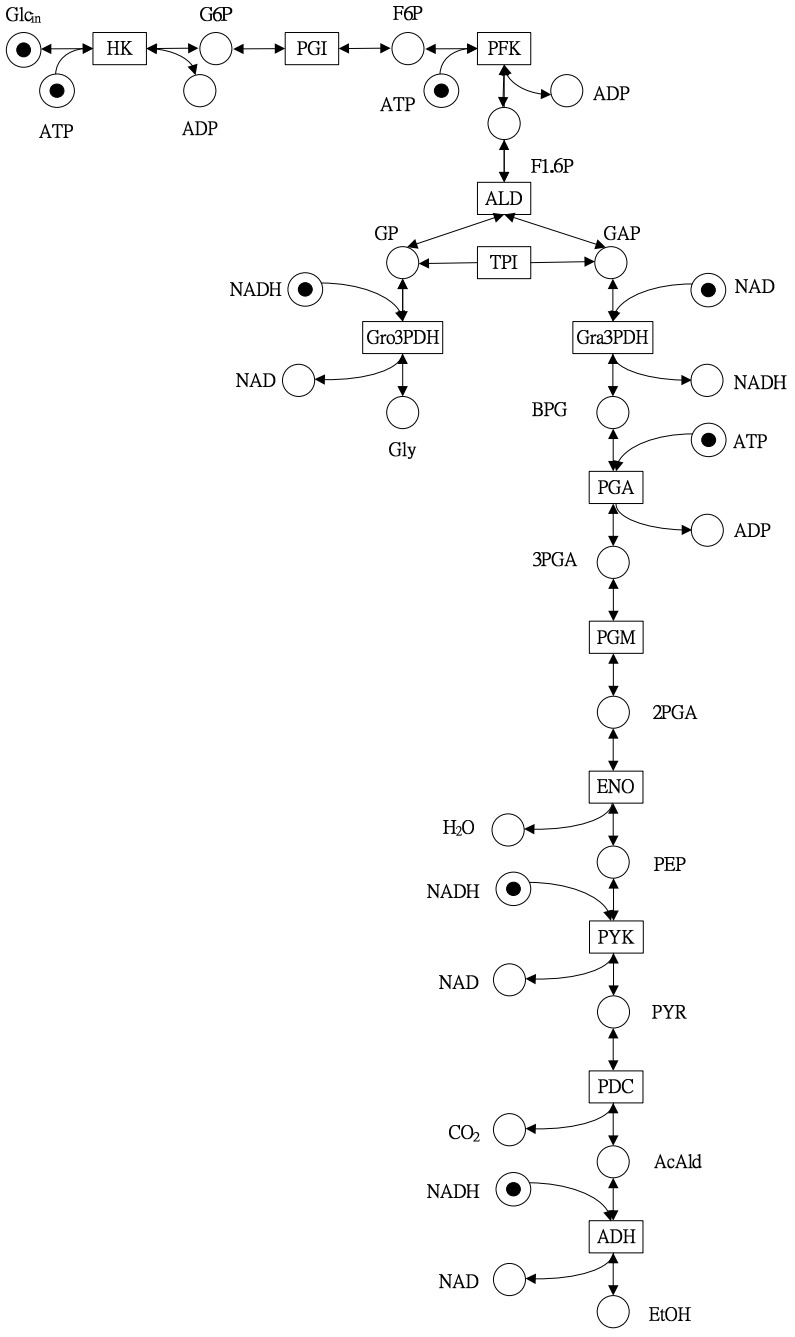
[56] 蔡劍霞，「以系統生物學修正Petri Nets於代謝網路之應用」，碩士論文，國立暨南國際大學資訊管理研究所，2005。

附錄 A 醣酵解作用

1. 本測試醣酵解作用又稱為EMP(Embden-Meyerhof-Parnas pathway)，為生物體內主要供給能量來源之一，其主要淨反應式如下：

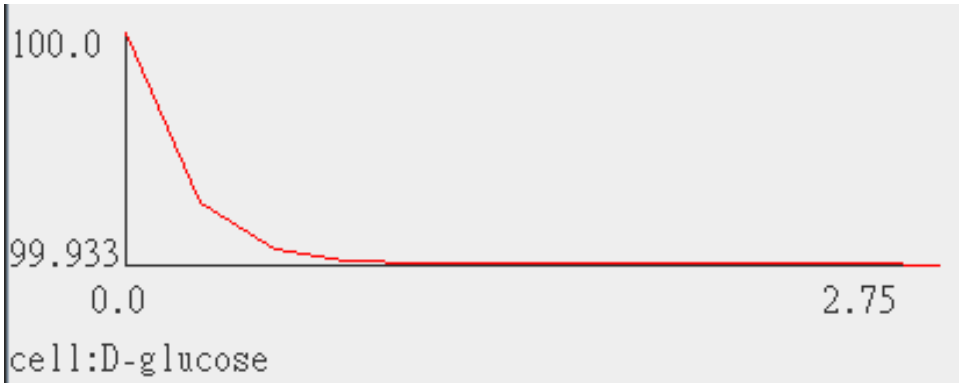
$$Glucose + 2NAD^+ + 2ADP + 2P_i \rightarrow 2pyruvate + 2NADH + 2H^+ + 2ATP + 2H_2O$$

2. 以Petri Nets建構醣酵解作用之模型：



3. 各反應物模擬結果如下：

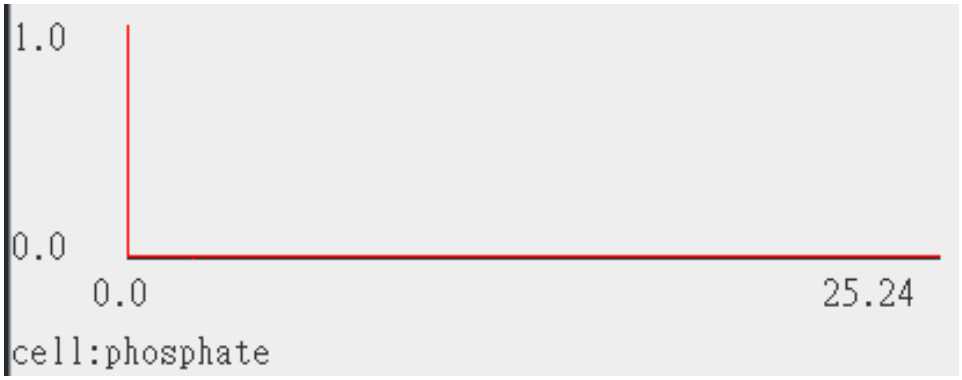
■ **Glucose**



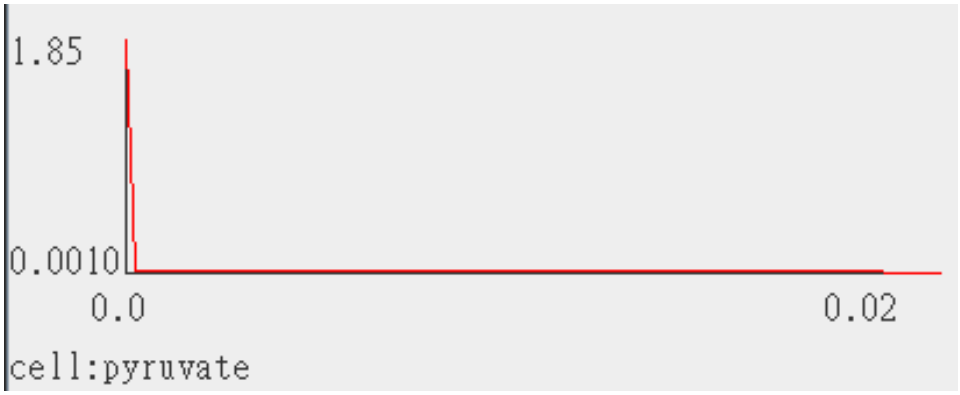
■ **NAD$^+$**

■ **ADP**

■ P_i

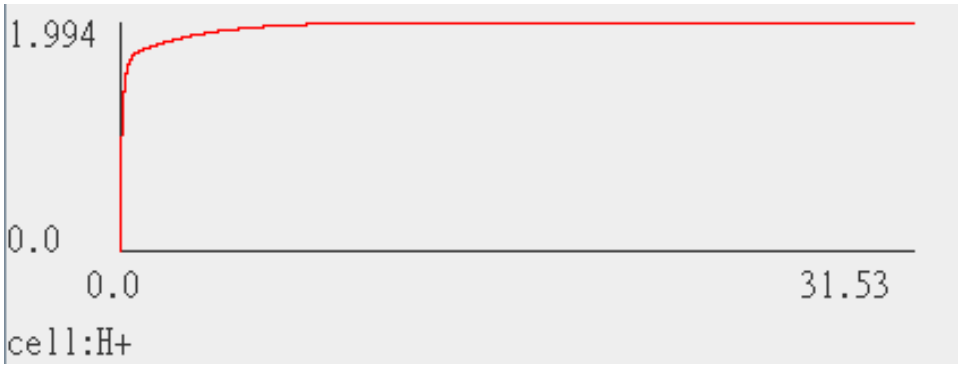


■ pyruvate

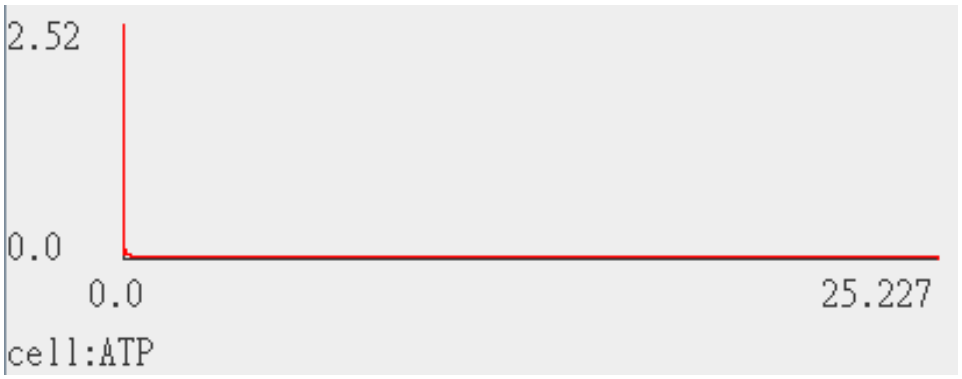


■ NADH

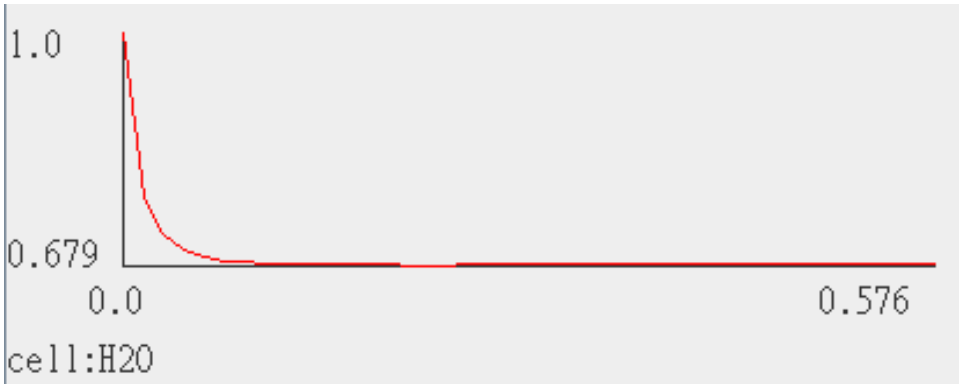
■ H^+


1.994
0.0
0.0
31.53
cell:H+

■ ATP


2.52
0.0
0.0
25.227
cell:ATP

■ H_2O


1.0
0.679
0.0
0.576
cell:H20