國　立　交　通　大　學
工業工程與管理學系

碩士論文

建立自動化製造系統的開放資料伺服器連接架構

Building the Open Data Server Connectivity Architecture for Automated Manufacturing Systems

研究生：黃柏勳
指導教授：梁高榮博士

中華民國九十九年二月

建立自動化製造系統的開放資料伺服器連接架構

研究生：黃柏勳　　指導教授：梁高榮博士

國立交通大學工業工程與管理學系

摘要

本論文提出一個介於裴氏圖導向控制器及製造系統之間的「開放資料伺服器連接」的標準介面。在現今的產業裡，自動化製造系統通常是透過可程式控制器來控制的。又可程式控制器的品牌選項非常多，而設計控制器的裴氏圖軟體模組選項也不少。每一項兩者的組合都需要一個特別設計的連接介面。在產業界裡，這項令人怯步的工程事實已成為推動裴氏圖控制器普及化的最主要阻礙。在本論文所提出的「開放資料伺服器連接」標準中，其裴氏圖介面是依據裴氏圖軟體模組的產業標準，即裴氏圖加註語言來設計的；其可程式控制器介面則是設計成一個容易安裝各種可程式控制器專屬驅動程式的平臺。這種新的解決之道使得任意裴氏圖軟體模組及可程式控制器的組合部署變得非常容易。為了展示此新想法的可行性，本論文已藉由 Java 程式語言完成此「開放資料伺服器連接」架構的實作。此外一個小型自動化製造系統已完全地透過此實作出來的「開放資料伺服器連接」架構來進行其控制。

關鍵字：
開放資料伺服器連結(Open Data Server Connectivity)
裴氏圖(Petri Net)
裴氏圖加註語言(Petri Net Markup Language)
可程式控制器(Programmable Logic Controller)
自動化製造系統(Automated Manufacturing System)

Building the Open Data Server Connectivity Architecture of Automated Manufacturing Systems

Student: Po-Hsun, Huang Advisor: Dr. Gau-Rong, Liang

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

Abstract

A standard interface named Open Data Server Connectivity (ODSC) has been proposed between a Petri Net (PN) based controller and its underlying manufacturing system. Today an automated manufacturing system is usually controlled through a Programmable Logic Controller (PLC) in industry. However, many PLC brands are available, and there exist various PN software modules for designing controllers. As a result, a specially designed interface is needed for each combination of a chosen PN software module and a dedicated PLC. This terrible engineering fact has become the main barrier for the popularity of PN controllers in industry. In the proposed ODSC interface, the PN side is designed according to Petri Net Markup Language (PNML) which is an industrial standard for various PN software modules, and the PLC side is a well-designed platform suitable for installing the given driver of any dedicated PLC. The new solution makes the combined deployment of an arbitrary PN software module and a PLC easily. For showing the feasibility of this new idea, the proposed ODSC architecture has been implemented by Java programming language. Moreover, a small-scale automated manufacturing system has been completely controlled under the implemented ODSC architecture.

Keywords：
Open Data Server Connectivity
Petri Net
Petri Net Markup Language
Programmable Logic Controller
Automated Manufacturing System

致謝

本論文得以順利完成，首先要感謝我的指導教授梁高榮老師對於我的幫助。一直以來在我的研究方面，梁老師總是給予我最大的寬容度讓我自由發揮，並且也提供我許多資源配合我研究上的需求，這使我能夠在艱辛的研究過程中盡興的投入在我的研究之中。此外，也要感謝他竭心竭力督促我的研究進度以及不辭辛勞指導我論文的方向，讓我順遂的完成此篇論文。我也要感謝兩位口試委員張永佳老師、陳文智老師在百忙之中抽空替我審核論文、參予我的論文口試並給予我許多寶貴的建議，這讓我的論文內容得以更加充實。

接著我要感謝實驗室裡的所有研究夥伴，這包含 95、96 級的學長、姐謝昇晏、陳音帆、林潔妤、范植宇、蔡耿全，和我同一屆的 97 級同學鄭瑋廷、何青樺、魏良縈、顧佳樺、鄭惠華，以及 98 級學弟、妹連惠珍、彭思瑜、劉思宇、鄭仲元、朱明典，由於和這些伙伴們彼此之間的心得交換與互相激勵、學習，這才讓我的研究所生涯不至於過於苦悶。

最後，我要感謝交通大學六年來的培養，在這個聚集眾多優秀人才以及匯集大量研究資源的校園內，所見的人、事、物都不斷的刺激我迅速成長，感謝它給予我這樣的一個機會以及提供一個如此良好的學習環境讓我能夠有幸與這些頂尖人士一同學習成長。

目錄

摘要 i
Abstract ii
致謝 iii
目錄 iv
圖目錄 vi
表目錄 ix
第一章 緒論 1
1.1 研究動機 1
1.2 問題界定與研究目的 3
1.3 研究方法與論文架構 4
第二章 文獻回顧 5
2.1 自動化製造系統的 SMT 架構 5
2.2 SMT 架構控制程式 8
2.2.1 即時型專家系統 G2 平臺控制程式 8
2.2.2 J2ME 平臺控制程式 10
2.2.3 J2SE 平臺控制程式 10
2.2.4 不同平臺控制程式比較 11
2.3 裴氏圖 13
2.3.1 詮釋性裴氏圖的定義 14
2.3.2 詮釋性裴氏圖與 Grafcet 16
2.3.3 詮釋性裴氏圖與法則 17
2.3.4 裴氏圖加註語言 18
2.3.5 優缺點說明 19
2.4 開放資料庫連接架構 21
第三章 開放資料伺服器連接架構的設計 22
3.1 開放資料伺服器連接架構 22
3.2 「資料伺服器」的規格設計 25
3.2.1 修正詮釋性裴氏圖的符號定義 25
3.2.2 修正詮釋性裴氏圖的同步性設定 26
3.2.3 修正詮釋性裴氏圖的法則轉換 27
3.3 決策系統連接介面設計 29
3.4 驅動程式連接介面設計 31
第四章 ODSC 資料連接介面的實作 34
4.1 PNML 技術簡化 ODSC 實作的開發流程 34
4.2 PNML 導向資料伺服器連接軟體開發流程 36
4.3 PNML 導向資料伺服器連接軟體需求分析 37
4.3.1 軟體功能分析 37
4.3.2 軟體資料分析 38

4.3.3 軟體操作流程 39
4.4 PNML 導向資料伺服器連接軟體規格設計 40
4.4.1 軟體架構 40
4.4.2 軟體套件設計 42
4.4.3 操作介面的設計 55
4.4.4 軟體執行流程 59
第五章 自動化製造系統範例的分析與操作 63
5.1 灌模系統的功能說明 63
5.2 灌模系統的監控員設計 67
5.3 灌模系統的偵查員與維修員設計 74
5.4 灌模系統可程式邏輯控制器的驅動程式設計 76
5.5 灌模系統裴氏圖同步性設定 79
第六章 結論與未來研究方向 83
參考文獻 85
附錄一 灌模系統 IDEF0 規格圖設計 87
附錄二 灌模系統監控法則 94

圖目錄

圖 1.1 論文研究方法流程 4
圖 2.1 SMT 架構、開放資料伺服器連接架構與開放資料庫連接架構 5
圖 2.2 遠端監控系統架構 6
圖 2.3 彈性輸送機系統 8
圖 2.4 彈性灌模系統 9
圖 2.5 整合型 SMT 方法流程 9
圖 2.6 手機監控流程 10
圖 2.7 裴氏圖控制程式開發系統監控流程 11
圖 2.8 裴氏圖元素 13
圖 2.9 轉移點激發規則 13
圖 2.10 詮釋性裴氏圖 14
圖 2.11 輸送帶系統與詮釋性裴氏圖 15
圖 2.12 Grafcet 步階表示法 16
圖 2.13 Grafcet 轉移點與箭號表示法 16
圖 2.14 Grafcet 與詮釋性裴氏圖狀態轉換一 17
圖 2.15 Grafcet 與詮釋性裴氏圖狀態轉換二 17
圖 2.16 詮釋性裴氏圖與法則的轉換規則 17
圖 2.17 輸送帶詮釋性裴氏圖及法則轉換 18
圖 2.18 可加註語言範例 18
圖 2.19 裴氏圖加註語言表達法 19
圖 2.20 兩機台製造系統裴氏圖及法則 20
圖 2.21 開放資料庫連接架構 21
圖 3.1 開放資料伺服器連接架構 22
圖 3.2 資料伺服器標準規格設計流程 23
圖 3.3 修正詮釋性裴氏圖 26
圖 3.4 輸送帶系統與修正詮釋性裴氏圖 27
圖 3.5 修正詮釋性裴氏圖監控法則轉換 28
圖 3.6 輸送帶系統監控法則 28
圖 3.7 資料伺服器與決策系統的互動模式 29
圖 3.8 fireTransition 方法的作業流程 29
圖 3.9 updateMarking 方法的作業流程 30
圖 3.10 errorRecovering 方法的作業流程 30
圖 3.11 資料伺服器與驅動程式的基本通訊流程 31
圖 3.12 openPLCConnection 方法的作業流程 31
圖 3.13 getPLCInputState 方法的作業流程 32
圖 3.14 Handle 方法的 ON、OFF 操作指令說明 32
圖 3.15 Handle 方法的作業流程 33
圖 3.16 closePLCConnection 方法的作業流程 33

圖 4.1 PNML 導向開放資料伺服器連接架構開發流程 34
圖 4.2 PNML 導向資料伺服器連接軟體開發流程圖 36
圖 4.3 PNML 導向資料伺服器連接軟體功能 37
圖 4.4 PNML 資料伺服器連接軟體的 IDEF0 規格圖 38
圖 4.5 Controller 資料夾架構 38
圖 4.6 PNML 導向資料伺服器連接軟體操作流程 39
圖 4.7 以類別圖表達 GUI、PetriNet、XMLTools、FileGenerator、Main 套件部署情形 41
圖 4.8 以類別圖表達 Controller 套件部署情形 42
圖 4.9 PetriNet 套件架構 43
圖 4.10 Component 套件類別圖 43
圖 4.11 Tools 套件類別圖 46
圖 4.12 XMLTools 套件架構 46
圖 4.13 PNMLpaser 類別的 parseFile 方法作業流程圖 47
圖 4.14 XMLTools 套件類別圖 47
圖 4.15 FileGenerator 套件架構 48
圖 4.17 DataServerGenerator 類別的 fireTransition 方法自動化撰碼模式 49
圖 4.18 DataServerGenerator 類別的 setup 方法自動化撰碼模式 50
圖 4.19 DataServerGenerator 類別作業流程 50
圖 4.20 FileGenerator 套件產生的資料伺服器軟體與 PetriNet 類別 51
圖 4.21 PetriNetGenerator 類別的作業流程 51
圖 4.22 manifest.mf 檔案內容格式 52
圖 4.23 JarGenerator 類別作業流程 52
圖 4.24 FileGenerator 套件類別圖 53
圖 4.25 Controller 套件架構 54
圖 4.26 PNML 導向資料伺服器連接軟體的操作介面 56
圖 4.27 暫存點同步性設定畫面 57
圖 4.28 轉移點同步性設定畫面 57
圖 4.29 驅動程式選取對話框 58
圖 4.30 Controller.jar 執行檔操作介面 58
圖 4.31 不變量監控器 58
圖 4.32 PNML 檔載入功能執行流程 59
圖 4.33 虛擬變數設定、控制器節點數量設定功能執行流程 60
圖 4.34 暫存點、轉移點同步性設定功能執行流程 61
圖 4.35 PNML 導向資料伺服器連接軟體 PNML 檔匯出執行流程 62
圖 5.1 灌模系統照片 63
圖 5.2 RFID 讀取機 IF5 及標籤 64
圖 5.3 灌模系統 64
圖 5.4 可程式邏輯控制器 SoftPLC 65
圖 5.5 階層轉換法—IDEF0 轉換裴氏圖 67
圖 5.6 階層轉換法自動化軟體操作畫面 68

圖 5.7 灌模系統裴氏圖 68
圖 5.8 灌模系統法則矩陣 71
圖 5.9 灌模系統可達狀態圖 72
圖 5.10 灌模系統最短週期時間分析 73
圖 5.11 putBits 方法運算流程 76
圖 5.12 SoftPLC 驅動程式 77
圖 5.13 控制器節點數量設定 79
圖 5.14 虛擬變數設定 79
圖 5.15 暫存點 P_1 同步性設定 80
圖 5.16 暫存點 P_3 同步性設定 80
圖 5.17 轉移點 T_1 同步性設定 81
圖 5.18 轉移點 T_5 同步性設定 81
圖 5.19 驅動程式設定 82
圖 5.20 灌模系統監控軟體 82
圖 6.1 傳統資料伺服器連接架構 83
圖 6.2 開放資料伺服器連接架構 84
圖 A.1 IDEF0 箭號與作業方格關係圖 87
圖 A.2 IDEF0 作業方格階層性示意圖 88
圖 A.3 備料系統 A-0 層 IDEF0 規格圖 88
圖 A.4 備料系統 A0 層 IDEF0 圖 89
圖 A.5 備料系統 A1 層 IDEF0 圖 90
圖 A.6 備料系統 A2 層 IDEF0 圖 90
圖 A.7 備料系統 A3 層 IDEF0 圖 91
圖 A.8 模具輸送系統 A-0 層 IDEF0 圖 92
圖 A.9 模具輸送系統 A0 層 IDEF0 圖 92
圖 A.10 模具輸送系統 A3 層 IDEF0 圖 93

表目錄

表 2.1 控制程式比較 11
表 3.1 輸送帶系統輸入、輸出節點說明 27
表 4.1 Component 套件各類別方法說明 44
表 4.1 Component 套件各類別方法說明(續) 45
表 4.2 Tools 套件各類別方法說明 46
表 4.3 PNMLparser 套件各類別方法說明 48
表 4.4 FileGenerator 套件各類別方法說明 53
表 4.5 Controller 套件各類別方法說明 55
表 5.1 灌模系統硬體裝置資料 66
表 5.2 模具輸送系統裴氏圖暫存點與轉移點的說明 69
表 5.3 備料系統裴氏圖暫存點與轉移點的說明 70
表 5.3 備料系統裴氏圖暫存點與轉移點的說明(續) 71
表 5.4 灌模系統裴氏圖轉移點不變量 74
表 5.5 灌模系統裴氏圖暫存點不變量 74
表 C.1 灌模系統法則 94
表 C.1 灌模系統法則(續) 95
表 C.1 灌模系統法則(續) 96
表 C.1 灌模系統法則(續) 97

第一章 緒論

本章主要說明本論文的研究方向、目的與論文的整體架構，以下共分成三個部份：1.1 節為『研究動機』，1.2 節為『問題界定與研究目的』，1.3 節為『研究方法與論文架構』。

1.1 研究動機

對於製造業技術發展而言，工廠自動化控制一直是主要的研究方向之一。透過自動化作業，生產線能夠有效率的進行各項作業程式並且保持一致性的產品品質，這有助於提升工廠生產力以及降低產品的瑕疵率。一般來說，在進行自動化製造系統的決策系統設計時，主要可以分成以下四個階段：監控(Supervisory Control)、錯誤偵查(Monitoring)、故障診斷(Fault Diagnosis)及故障回復(Error Recovering)[26]。監控階段在於設計控制器正常行為下的控制邏輯，使得控制器能夠按照所制定的規格自動地依序執行作業程式。錯誤偵查、故障診斷及故障回復是設計控制器異常行為下的控制邏輯。錯誤偵查主要觀察系統在運作時是否有異常行為發生，當異常狀況發生時，系統的傷害程度會與故障延續的時間成正比，一個完善的偵錯制度必須能夠迅速並且準確的察覺到系統錯誤發生。故障診斷及故障回復部份是指當系統偵查到錯誤出現時，具有能力自行分析錯誤發生的可能原因以及位置，並且依據分析結果採取適當的因應措施以避免系統損害的擴大以及可能造成的人員危險，甚至於主動進行系統故障回復作業使得系統能夠重新恢復正常運作。

自動化製造系統的控制邏輯可以透過裴氏圖來表達，裴氏圖是一種圖形塑模工具，尤其善於表現系統的平行作業行為以及隨時間進行之下的系統狀態變化，此外，其具有良好的數學性質以及擁有多種相關分析技術，如可達狀態圖分析(Reachability Graph Analysis)、週期時間分析(Cycle Time Analysis)、不變量分析(Invariants Analysis)等[5]，所以在設計系統控制邏輯時，可以藉由這些裴氏圖分析技術驗證控制邏輯的正確性。而通過驗證的裴氏圖則是可以利用物件導向程式語言撰碼成相對應的監控員(Supervisor)、偵察員(Monitor)及維修員(Troubleshooter)軟體[4]分別執行系統監控、偵錯、以及故障診斷與修復的工作。

除了監控員、偵察員、維修員軟體的設計之外，這裡還需要考量資料伺服員(Data Server)軟體的設計。資料伺服員軟體是決策系統與控制器間的溝通介面，其定義了裴氏圖對於控制器的同步性設定，這主要是用來進行決策系統資料與控制器行為之間的連接與轉換。當監控員、偵察員、維修員軟體運行時，資料伺服員會將決策系統的作業指令轉譯成控制器的程式語言作用於控制器。此外，資料伺服員也會接收控制器所回傳的輸入信號，並轉換成決策系統的資料格式提供給監控員、偵察員、維修員軟體進行下一回合的邏輯判斷使用。

以往，在資料伺服員軟體的設計方面，是直接使用控制器供應商提供的應用程式介面(Application Programming Interface, API)裡的函式、方法來與控制器進行連接以及指令轉換。然而，各類型、品牌的可程式邏輯控制器的應用程式介面所接受及傳送的資料格式、操作方法、程式語法都具有不同程度的差異，彼此之間無法完全相容，造成使用者撰寫出的資料伺服員軟體往往只能依附於單一類型的控制器，不能輕易的在各種可程式邏輯控制器之間交換使用，這侷限了整個監控軟體的移植能力。也就是說，監控軟體的設計師必須在軟體設計以前，就選擇一個非常合適的可邏輯程式控制器，否則當軟體撰寫完成後則很難再作更換。

在本論文中，提出一個開放資料伺服器連接架構(Open Data Server Connectivity Architecture, ODSC Architecture)，ODSC 的目的是要標準化裴氏圖控制邏輯與控制器之間的連接模式與通訊介面，這裡強調的開放指的是依據標準規範建置出的一個能夠通用於所有控制器類型、規格的資料伺服器軟體。此架構的設計概念主要是將決策系統與控制器之間的指令轉譯作業切分成兩個階段：決策系統至資料伺服器軟體、資料伺服器軟體至控制器。決策系統所下達的指令會先透過資料伺服器軟體轉譯成一種能在各種類型、品牌控制器之間共通使用的指令格式，傳送給可程式邏輯控制器的驅動程式。驅動程式再將這種標準指令轉譯成該類型、規格控制器所使用的特定程式語言格式，作用在控制器上。這種部署方式使得軟體設計師只需要依循共通指令規格進行資料伺服器軟體的開發，在軟體開發完成後，就可以藉由替換驅動程式來與各種類型的控制器連接與傳達指令，而無須重新製作資料伺服器軟體。如此一來，能夠提升監控軟體的互用性(Interoperability)、重複使用性(Reusability)以及軟體的開發效率。

1.2 問題界定與研究目的

在自動化製造系統裡，決策系統與控制器之間通常會透過資料伺服員軟體作為銜接兩者的連接介面，其負責轉譯雙向傳輸的資料內容。然而，各類型、品牌控制器皆擁有其專屬的傳輸資料內容格式、連接方式與指令語法，彼此之間並不完全相容，也無一個標準規範可循，這使得軟體開發人員一次僅能針對單一種控制器規格進行其資料伺服員軟體的設計，造成所開發出的監控軟體往往會受限於該控制器規格。一但軟體的執行環境被單一種控制器規格給綁住，使用者就無法輕易的在不同控制器之間交換使用其監控軟體，這會喪失了軟體的使用彈性。

在這樣的情況下，當面對軟體移植或是控制器軟、硬體元件升級時，軟體開發人員必須花費龐大的訓練成本重新瞭解每一個控制器的軟、硬體規格以及操作指令的使用，並且耗費大量時間進行軟體程式碼的調整與修改，才能夠讓原本已開發出的監控軟體適應新的執行環境。

本論文主要是提出一個開放資料伺服器的基本連接架構以及訂定此架構中各組成元件的規格，目的是要標準化決策系統與控制器之間的溝通介面、提供開發人員一個明確的資料伺服器軟體設計規範、提升監控軟體與控制器之間的互用性以及減少軟體開發成本，讓軟體開發人員能夠投入更多的資源專致於監控軟體的創新，而非浪費過多的時間在於修改現存監控軟體使其適用在不同的執行環境。此外，在標準規格制定以後，各種監控程式的輔助開發軟體也能夠遵循此規格被開發出來，這有助於提升未來監控軟體開發的便利性。開放資料伺服器連接架構的相關設計方式與規格設計會在第三章中說明。

在開放資料伺服器連接架構實作方面，這裡是使用 Java 程式語言以及開放資料伺服器連接架構的規格建置一個 PNML 導向的開放資料伺服器連接軟體。該軟體能夠藉由使用者所設計的裴氏圖加註語言檔、裴氏圖同步性設定以及所選擇的驅動程式，進行資料伺服器軟體的自動化撰碼以及決策系統、資料伺服器、驅動程式自動化整合，最後輸出一個具標準規格的資料伺服器軟體以及一個簡易的監控軟體。使用者可以直接執行該監控軟體來與控制器通訊，以進行自動化監控作業，或者是更換其驅動程式來控制各種類型的控制器，發揮開放架構的高移植性。有關開放資料伺服器連接架構實作部分會在第四章中說明。

1.3 研究方法與論文架構

本論文的研究方法共分成四個階段，流程圖如 1.1 所示：

第一階段是探討現行裴氏圖導向的監控軟體規格。這裡主要是整理各種監控軟體的設計架構。

第二階段是針對本論文所提出開放資料伺服器連接架構進行說明。

第三階段是定義開放資料伺服器連接架構中各元件的規格。

第四階段是利用所設計的開放資料伺服器連接架構規格以及 Java 程式語言進行資料伺服器連接架構的實作

在論文架構方面共分成六章，各章節內容大綱如下：

第一章：『緒論』主要說明本論文的研究動機、研究目的、研究方法及論文架構等。

第二章：『文獻回顧』在於探討裴氏圖的基本性質以及現有裴氏圖導向監控軟體的程式架構與相關技術。

第三章：『開放資料伺服器連接架構設計』說明本論文所提出的開放資料伺服器連接架構以及定義架構中各項元件的基本函式、通訊介面、資料傳遞格式的規格。

第四章：『ODSC 的資料連接介面實作』是利用 Java 程式語言以及開放資料伺服器連接架構設計出一個 PNML 導向的資料伺服器連接軟體，讓使用者能夠輸入其設計的裴氏圖加註語言文件至軟體內，並由軟體產生一個具標準介面規格的資料伺服器軟體。使用者亦可以在軟體中選擇與更換欲使用的驅動程式，再由軟體自動將驅動程式與資料伺服器軟體進行整合，以方便與不同類型控制器溝通。

第五章：『自動化製造系統範例的分析與操作』會以一個灌模系統作為範例並利用本論文提出之開放資料伺服器連接架構實作出其監控軟體，藉此說明此架構的運作方式。

第六章：『結論與未來方向』說明本論文的研究結果以及未來可延伸的發展方向。

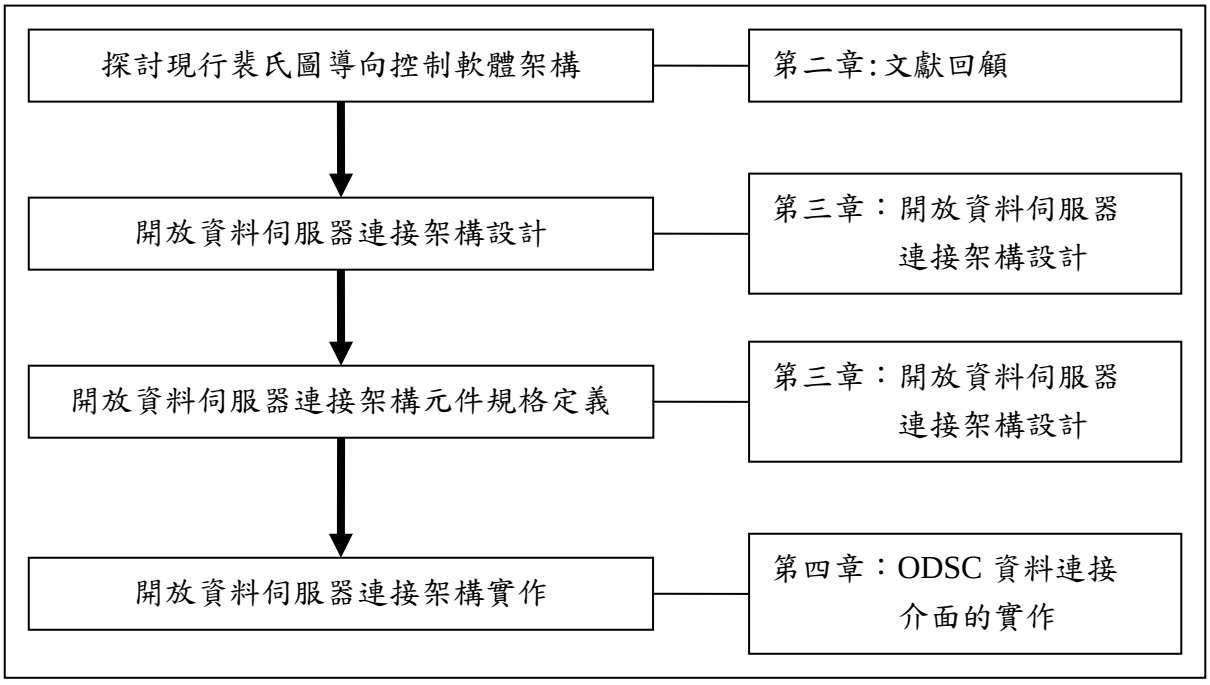


圖 1.1 論文研究方法流程

第二章 文獻回顧

本章主要是回顧自動化製造系統的 SMT 架構、SMT 導向控制系統、裴氏圖(Petri Net)、開放資料庫連接架構等相關文獻。內容共分為四節，第 2.1 節介紹自動化製造系統的 SMT 架構。第 2.2 節介紹採用 SMT 架構所設計出的裴氏圖導向控制系統。第 2.3 節介紹裴氏圖。第 2.4 節介紹開放資料庫連接架構。

2.1 自動化製造系統的 SMT 架構

工廠自動化一向為製造業的發展方向之一，其優點在於可減少人力、降低生產成本、提高產品品質及提昇生產效率。一個良好的自動化製造系統需具備兩項特性—效率性(Efficiency)、可靠性(Reliability)[26]。效率性指系統經由正確設計並且運行時具有良好績效。高可靠度指系統具備故障定位及自我回復能力。針對提升系統效率性與可靠性，自動化製造系統(Automated Manufacturing Systems, AMSs)的設計主要可分為四項議題：(1)監控、(2)偵查異常、(3)故障診斷、(4)錯誤回復[26]。SMT 架構即是整合上述四項議題所提出的一套自動化製造系統設計架構，其中 SMT 分別代表的是自動化製造系統裡決策系統的監控員、偵查員與維修員軟體[26]。

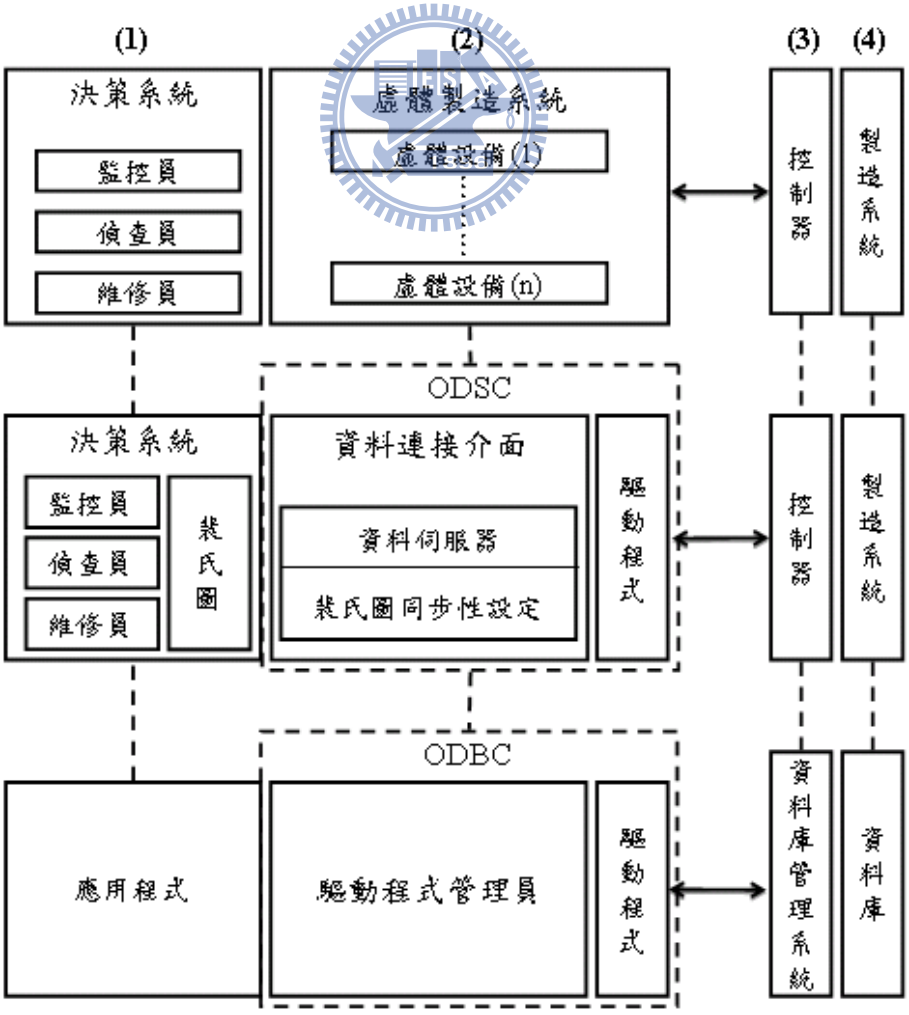


圖 2.1 SMT 架構、開放資料伺服器連接架構與開放資料庫連接架構

在 SMT 架構中，它主要是由四個元件所組成：(1)決策系統、(2)虛體製造系統(Virtual Manufacturing System, VMS)、(3)控制器與(4)製造系統(Manufacturing System)。SMT 架構如圖 2.1(a)所示。對於決策系統而言，其包含了監控員、偵察員、維修員三項軟體，監控員在於規範製造系統的正常行為，其負責接收系統介面所傳達之生產指令並根據所設計的控制邏輯進行實體製造系統的狀態轉換。偵查員在於確保系統維持正常運作，在監控員進行系統狀態控制的同時，偵查員會持續觀察製造系統的狀態以即時察覺錯誤的發生。當系統錯誤發生時，偵查員必須能夠察覺錯誤並且將實體系統的控制權從監控員轉換至維修員，維修員在於診斷系統錯誤，並自動修復可修復的錯誤使系統返回正常狀態。

對於虛體製造系統而言，這是實體製造系統與電腦系統之間的信號交換介面，它主要是模擬實體製造系統的硬體結構設計出一系列與實體系統硬體設備相對應的虛體物件，用來紀錄、儲存各個實體設備的屬性狀態。當決策系統下達監控指令時，虛體製造系統會依據監控指令變更其虛體設備的屬性狀態，並且呼叫可程式邏輯控制器的控制指令去轉換實體設備的屬性狀態，使之與虛體設備的屬性狀態達到一致。

對於控制器而言，它會接受虛體製造系統的所傳送的操作指令去向實體製造系統進行信號輸出的動作，同時也會接收由實體製造系統端所傳入的電子信號，並回傳至虛體製造系統中。

對於製造系統而言，這是由各類型製造設備所組成的，而這些製造設備會連接至控制器的輸出、輸入模組中，所以使用者可以藉由操作控制器電子信號的導通或斷路來控制與之連接的各項設備的作業行為。

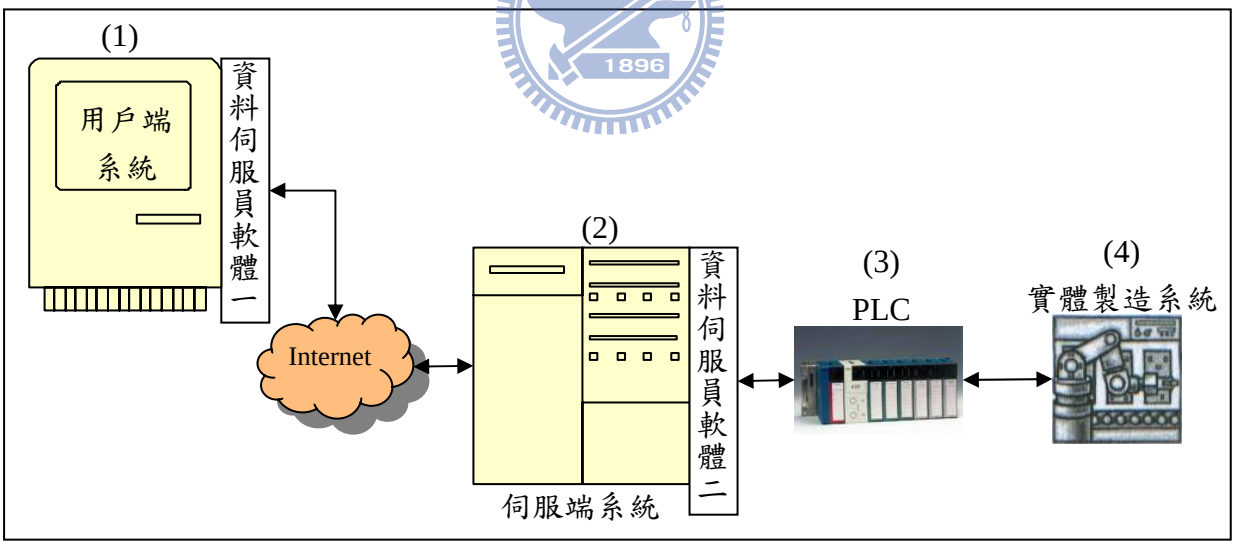


圖 2.2 遠端監控系統架構

在一個典型的遠端監控系統架構裡，其組成元件可以分成(1)伺服端系統、(2)用戶端系統、(3)可程式邏輯控制器與(4)實體製造系統四個部分，如圖 2.2 所示。對於伺服端系統而言，這是指 SMT 架構的決策系統與虛體製造系統部分。在自動化監控功能啟動的期間，伺服器系統會依據使用者所設計的控制邏輯即時判斷實體製造系統應執行的作業，並且透過內部網路(Intranet)或是通訊連接埠(Communication Port, COM)與可程式邏輯控制器連接，藉由操作控制器的輸入、輸出模組向實體製造系統接收或發送電子訊號來轉換製造系統的作業行為。

對於用戶端系統而言，這是指建立在網路傳輸技術上的一套遠端監控軟體，它可以讓使用者透過網路連線接收儲存在伺服端系統的實體製造系統運作資訊，如監視器畫面、不變量狀態與裴氏圖浮標狀態，並且透過影像處理技術將畫面呈現在用戶端，所以操作人員不需要親臨現場就能夠遠距離瞭解實體製造系統的運作情形。

在遠端監控系統底下，資料伺服員軟體的功能性包含了兩種不同的意義。資料伺服員軟體可以指的是裝置在用戶端系統上負責與伺服端系統進行聯繫的資料傳輸介面，如圖 2.2 中的資料伺服員軟體一，它的開發重點項目之一是在於其網路連線功能的穩定性與即時性，這影響到用戶端系統與伺服端系統之間的資料是否能夠同步、使用者所觀看到的影像是否即時。陳宗沂[7]利用 Java 程式語言的插座(Socket)網路傳輸技術設計了五種不同網路傳輸模式的資料伺服員軟體：位元元組串流傳輸模式、字串物件串流傳輸模式、統一編碼轉換格式(Unicode Transformation Format, UTF)資料串流傳輸模式、結合所有製造系統物件資料視為單一物件的物件串流傳輸模式以及視製造系統各物件資料為個別物件的物件串流傳輸模式，並分析在各種傳輸模式下資料伺服員進行資料傳輸時的傳輸時間績效。其分析結果顯示此五種網路傳輸模式之間的績效表現並無太大差異，這表示伺服端與用戶端之間所傳輸的資料型態並不會對於整體資料傳輸速度造成太大的影響。

資料伺服員軟體也可以指的是裝置在伺服端系統上用來與可程式邏輯控制器進行溝通的資料傳輸介面，如圖 2.2 中的資料伺服員軟體二，在 SMT 架構中，這即是虛體製造系統部分。虛體製造系統的開發重點項目之一是在於建立虛擬設備屬性狀態與可程式邏輯控制器操作指令之間的轉換模式，這是要讓虛體製造系統能夠依據其虛體設備的狀態向控制器下達電子信號輸出的指令。然而，在以往監控系統的開發過程中，虛擬製造系統的設計方式存在一項缺點，就是其主要僅針對單一種類型控制器的控制語法為考量設計出控制器操作指令的轉換模式，由於這種轉換模式只能專屬於單一類型控制器使用，所以當面臨控制器等硬體設備的汰舊換新時，就容易發生監控軟體無法相容於新的執行環境的問題。

本研究主要是針對圖 2.2 中資料伺服員軟體二的設計部分進行改良，目的是要建立一個跨控制器規格限制、通用型的決策系統與控制器之間的通訊介面，用來改善以往監控系統在使用上會受到單一種控制器規格侷限的問題以及系統無法有效率移植的問題，所設計出的通訊介面架構稱作開放資料伺服器連接架構，如圖 2.1(b)所示，開放資料伺服器連接架構的概念是來自於資訊系統中的開放資料庫連接架構[28]，這是一種使用者在開發應用程式時可以使用的標準資料庫資料讀寫介面，如圖 2.1(c)所示，它允許使用者使用單一種標準指令語法就能夠與各種類型的資料庫連接，進行相關操作，所以使用者在撰寫其應用程式時無須擔心未來所要使用的資料庫類型，有關開放資料庫連接架構在 2.4 節中詳述。

2.2 SMT 架構控制程式

有關目前以 SMT 為架構所設計的控制程式可依開發環境的不同區分為：即時專家系統 G2 平臺、J2ME(Java 2 Micro Edition)平臺及 J2SE(Java 2 Standard Edition)平臺三類。。2.2.1 節介紹即時專家系統 G2 平臺控制系統。2.2.2 節介紹 J2ME 平臺控制系統。2.2.3 節介紹 J2SE 平臺控制系統。2.2.4 節進行三種平臺的控制系統比較。

2.2.1 即時型專家系統 G2 平臺控制程式

專家系統主要由知識庫(Knowledge Base)以及推理引擎(Inference Engine)組成。知識庫用來儲存解決問題的法則，推理引擎則依據輸入變數與知識庫的法則進行推理並獲得結論，專家系統再依此結論產生相匹配的動作。G2[19]為美國 Gensym 公司於 1986 年所開發的一套商業化即時型專家系統。不同於與一般型專家系統，即時型專家系統被賦予與硬體介面連線溝通的能力，目前已廣泛應用於各個領域如：美國國防部、美國太空總署 NASA、福特汽車、惠普科技等。

對於以 G2 為平臺所建置的 SMT 控制程式而言，洪信銘[1]首先提出以 IDEF/CPN/G2 三階段設計方式設計監控員。IDEF 是指使用 IDEF0 流程圖進行系統需求的分析與設計，CPN 是指使用詮釋性裴氏圖及分析軟體 CPN 進行系統的驗證，通過驗證後的裴氏圖則轉換為相對應的法則撰寫入 G2 系統知識庫內，並經由 G2 的推理引擎進行實體系統的控制。

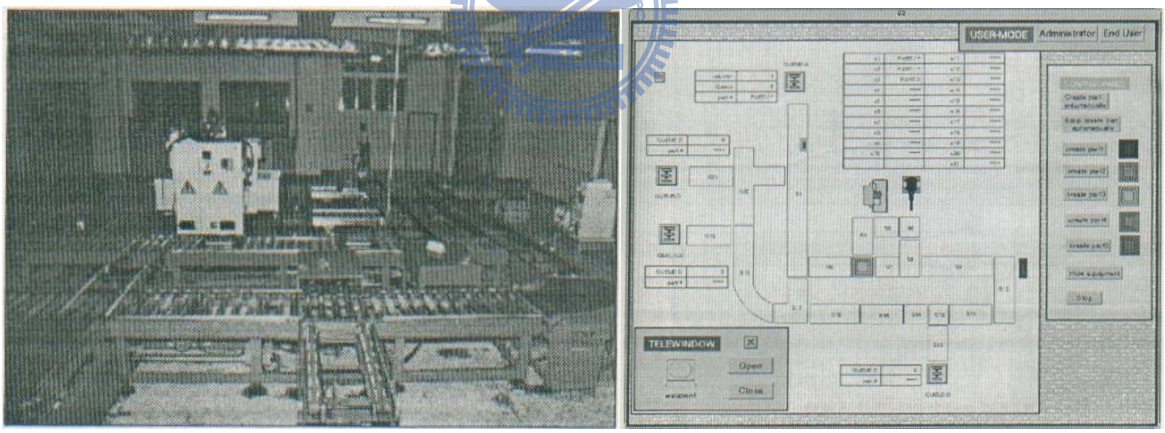


圖 2.3 彈性輸送機系統[1]

針對非重覆性製造系統的控制系統開發，劉育宏[11]採用了 SMT 架構進行控制程式設計。除了監控員的設計外劉育宏也加入了偵查員設計的考量。在監控員的設計上，劉育宏以區段資源(Segment Resource)的概念將實體製造系統分成多個區段，每個區塊都使用階層轉換法(Hierarchy Transformation Method, HTM)[30]來進行分析、設計、驗證並轉換為知識庫的法則。推理引擎會經由知識庫的法則以及虛體製造系統的狀態推理出實體製造系統的下一個狀態。在偵查員的設計方面，這主要是追蹤系統的可達狀態樹(Reachibility Tree)，當系統狀態不在可達狀態樹的可達狀態集合內時即判定系統出現錯誤。劉育宏以一彈性輸送機為例說明其設計原理，如圖 2.3 所示。

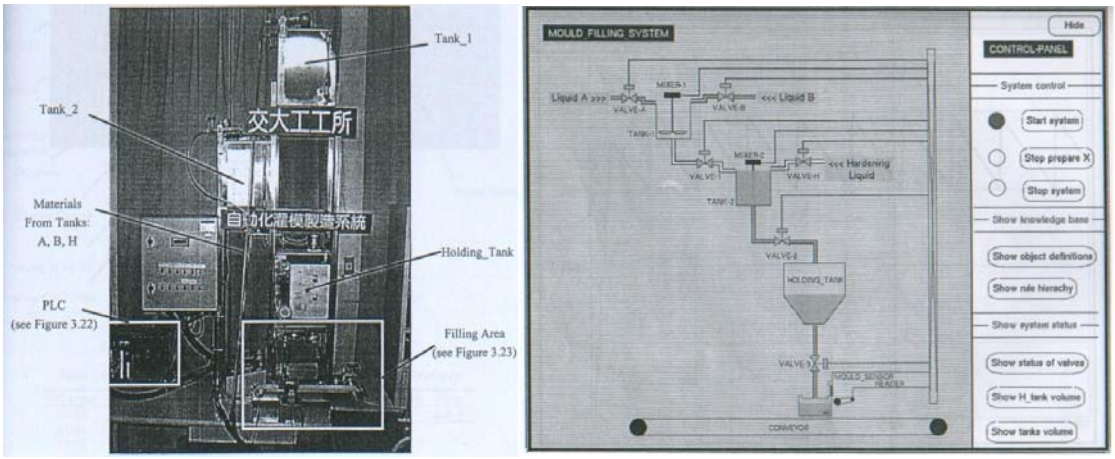


圖 2.4 彈性灌模系統[12]

鄭瑋弘[12]以 SMT 架構為基礎發展一個四階段的整合型 SMT 方法用於設計自動化製造系統，設計流程如圖 2.4 所示。第一階段為監控員的設計，其使用階層轉換法設計規範出系統正常行為並轉換成知識庫的法則。第二階段為偵查員的設計，其將系統的可能狀態定義為合法的 World 並訂定各個 World 應遵循的第一法則，法則則為一定性或定量的標準。偵查員經由觀察系統動態是否合法及第一法則是否被違背來偵查錯誤徵兆。第三階段為維修員的設計，其使用貝氏定理(Bayes' Theorem)[27]計算出最有可能發生錯誤的原因，並藉由擬定回復計畫進行自動回復工作。第四階段則是實體系統的安裝以及實體製造系統與虛體製造系統的連接。鄭瑋弘亦以彈性灌模系統作為其設計實例。

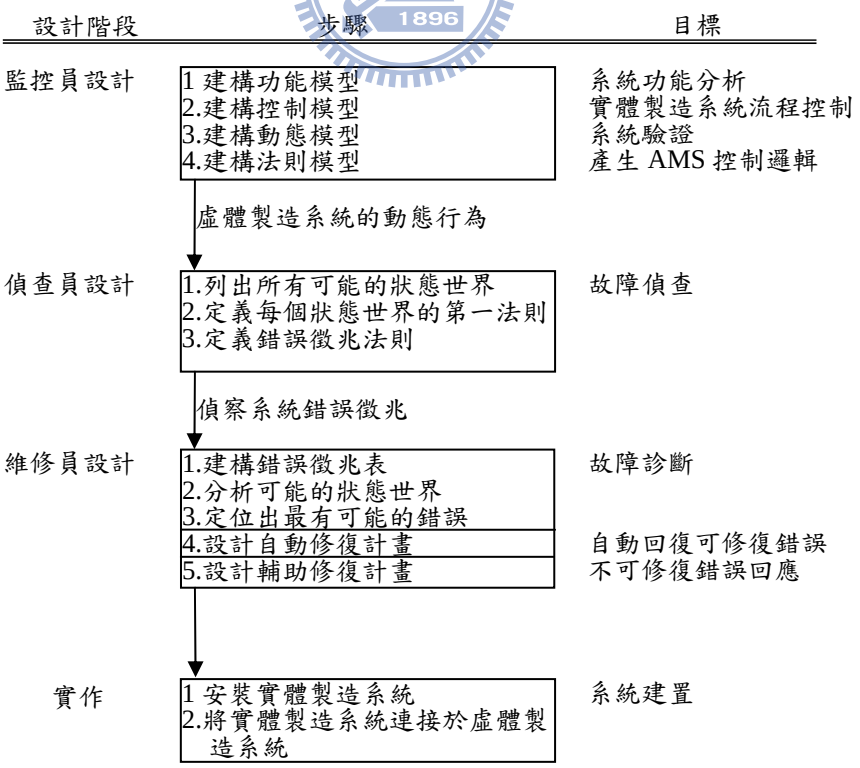


圖 2.5 整合型 SMT 方法流程[12]

2.2.2 J2ME 平臺控制程式

爪哇語言依據其應用層面的不同可以分為 J2EE、J2SE 及 J2ME[17]三種版本。J2EE 主要應用於企業型伺服器層面，J2SE 應用於個人電腦層面，J2ME 則是針對一般小型消費性電子產品及嵌入式裝置所提供的平臺。由於小型裝置在記憶體、顯示方式、能源供應上有其限制，所以相較與 J2SE 和 J2EE(Java Enterprise Edition)，J2ME 提供的運行環境也較為精簡。

為了在單一處理器上能同時進行虛體製造系統的監控、偵查與維修，首先梁高榮[4]提出虛體製造系統的多緒架構(Multi-thread)，其藉由爪哇程式語言所具有的多執行緒技術使能夠在單一處理器上模擬出平行處理的功能，以達到監控員、偵查員、維修員之間的平行處理。

陳啟宗與梁高榮[8]藉由 J2ME 環境、多緒架構及 SMT 架構在手機裡開發控制程式並能夠遠端監控實體製造系統，其監控流程如圖 2.6 所示。程式中包含有監控員、偵查員、資料伺服員及動畫員。監控員的設計是由階層轉換法產生系統狀態轉換法則，當轉換法則成立時即呼叫相對應的操作員(Operator)轉換虛體製造系統的狀態。偵查員的設計是將系統裴氏圖中的暫存點不變量區分為屬性方程式及偵查法則，藉由觀察偵查法則以瞭解實體系統狀態是否正常運作。資料伺服員作為遠端伺服器的資料傳輸視窗，負責 PLC 伺服器以及遠端控制器之間的訊息交換。動畫員則是將遠端伺服器所接收的訊息以動畫方式呈現於手機螢幕。此外，為實現詮釋性裴氏圖能夠同時激發多個可激發狀態的轉移點特性，程式中的每個操作員之間亦以多緒架構實作。

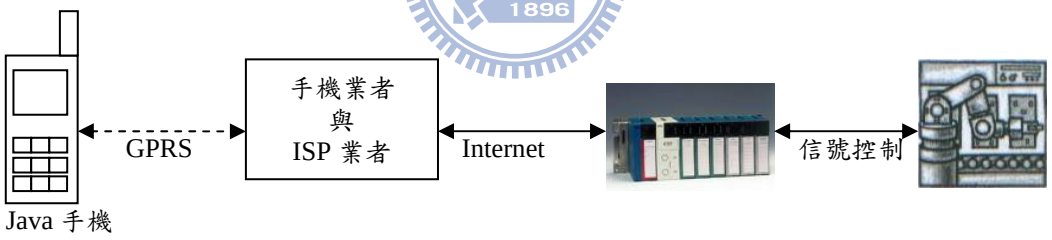


圖 2.6 手機監控流程[8]

2.2.3 J2SE 平臺控制程式

J2SE 為標準版爪哇語言，其適用於一般個人電腦層級的應用程式撰寫。相較於 J2ME 而言，J2SE 提供更為完整且彈性的運行環境能夠用來執行複雜的運算以及系統整合。對於一般自動化製造執行系統的設計與實作而言需要歷經規格(Specification)、驗證(Validation)、撰碼(Coding)、測試及偵查等步驟[6]，過程中常牽涉到多套軟體的操作及分析技術的使用，使得步驟相當繁瑣及費時。陳音帆[9]使用 J2SE 來整合裴氏圖加註語言(Petri Net Markup Language)、裴氏圖分析工具與可程式邏輯控制器等多項技術發展出具備驗證與自動撰碼能力的一套裴氏圖控制程式的開發系統。使用者僅需將其所設計的裴氏圖匯入開發系統內即可進行系統的驗證如死結分析、不變量分析、週期時間分析等並且由系統自動產生出監控員程式。所產生的監控員能夠透過網際網路與遠端的可程式邏輯控制器溝通並監控實體製造系統，其監控流程如圖 2.7 所示。當系統歷經再工程時期時，控制系統的開

發往往須要重新設計、撰碼。但透過裴氏圖控制器開發系統使用者只須修改系統的裴氏圖模型即可再次驗證系統並產生出監控員，這能夠有效的縮短系統開發時間。

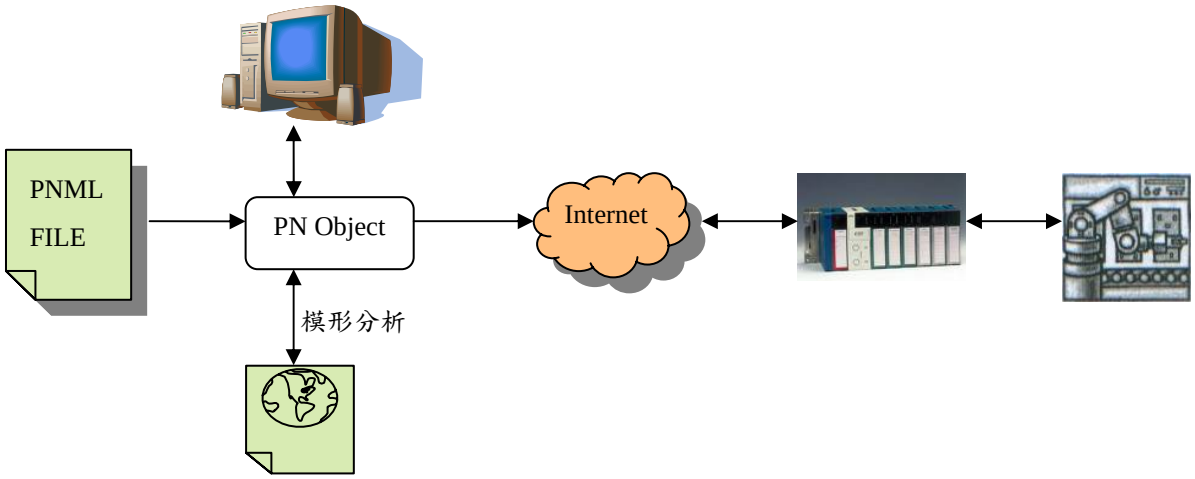


圖 2.7 裴氏圖控制程式開發系統監控流程[10]

2.2.4 不同平臺控制程式比較

表 2.1 控制程式比較

系統 / 功能		G2(鄭瑋弘)	J2ME(陳啟宗)	J2SE(陳音帆)
監控員	規格設計功能			
	系統驗證功能			●
	自動撰碼功能			●
	裴氏圖監控畫面		●	●
	動畫監控功能	●	●	
	遠端監控功能	●	●(可手機監控)	●
	監控員設計原理	階層轉換法	階層轉換法	階層轉換法
偵查員	自動撰碼功能			
	故障偵查功能	●	●	
	偵查員設計原理	狀態世界第一法則	偵查法則	無
維修員	自動撰碼功能			
	故障診斷功能	●		
	自動修護功能	●		
	偵查員設計原理	訂定修復計畫	無	無
可攜性		差	高	中
重複使用性		單一專案使用	單一專案使用	可跨專案使用
硬體設備		PC、PLC	手機、PC、PLC	PC、PLC
成本		高	免費	免費

就系統開發平臺而言，即時型專家系統具備完善的圖型化人機介面、硬體溝通介面、推理引擎、知識庫等功能，使用者僅需輸入控制法則至知識庫即可快速建構出一套控制程式。然而即時型專家系統開發環境的建置成本高且所設計出的程式不具有重複使用性及可攜性(Portability)，這使得每一次在建置及修改控制程式時都需要從頭開始並且難以進行程式的移轉。而爪哇語言所具備的跨平臺特性及物件導向特性則彌補了上述缺點，跨平臺特性使得控制程式只需開發一次即可以在任何作業系統上使用，物件導向程式設計的特性使

得所撰寫程式碼可以模組化重複使用，這兩性特性有助於提升控制程式的擴充性使得程式在修改、調整、增加功能上較為容易而開發出的程式亦能夠彈性的移作他用。不同平臺下控制程式比較結果如表 2.1 所示。

目前使用爪哇語言所設計的控制程式中無論是在J2SE或J2ME平臺上尚缺乏維修員的設計。此外，在 J2ME 平臺當中陳啟宗運用偵查法則設計偵查員的方式雖能判斷出設備屬性之間的互動情形是否正常但難以察覺到系統狀態順序發生錯誤。

2.3 裴氏圖

裴氏圖[31]是 1962 年由 Carl Adam Petri 所提出的圖形塑模工具，其常用來表現離散事件系統的動態行為及平行處理、同步處理能力。

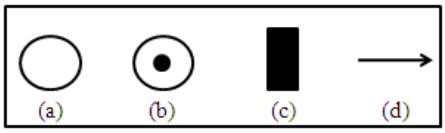


圖 2.8 裴氏圖元素

裴氏圖裡的主要構成元素包含有暫存點(Place)、轉移點(Transition)、箭號(Arc)。暫存點用來表示系統的狀態或條件，圖形中是以圓形作表示，如圖 2.8(a)。當暫存點中包含有浮標(Token)時，即表示系統正處於此暫存點的狀態之下或此暫存點所代表的條件成立，如圖 2.8(b)；反之，則表示系統不處於此暫存點的狀態之下或是暫存點代表的條件不成立。轉移點用來表示系統事件觸發、工作發生或是功能啟動，在圖形中是以長條形作表示，如圖 2.8(c)。暫存點與轉移點之間會以箭號作連接，用來表示系統狀態與系統事件觸發之間的關係，箭號圖形如圖 2.8(d)所示。當箭號的方向是由暫存點指向轉移點時，則稱此暫存點為轉移點的輸入暫存點；反之，則稱為轉移點的輸出暫存點。此外，箭號並不會存在於兩個轉移點或是兩個暫存點之間。

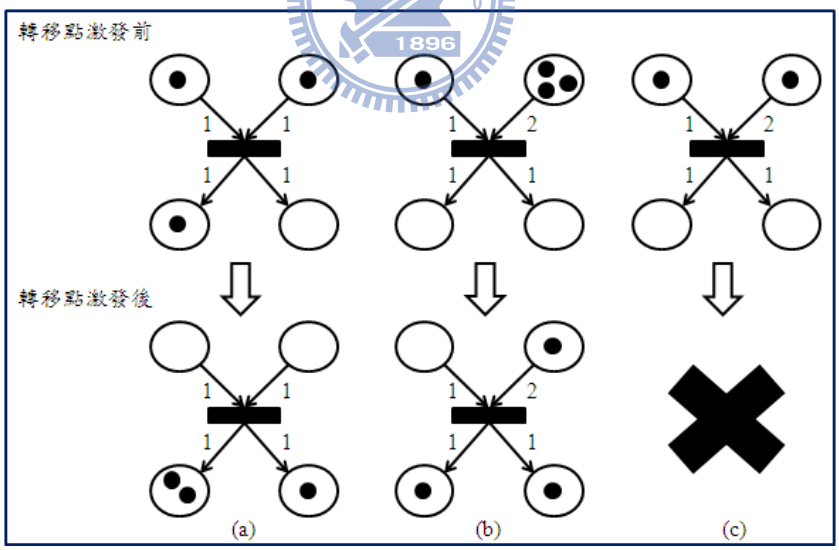


圖 2.9 轉移點激發規則

在裴氏圖的動態行為表現方面，是透過轉移點激發(Firing)來改變圖形中的浮標分佈狀態(Marking)，藉以表達出系統狀態的演化過程。當轉移點的輸入暫存點都包含有足夠的浮標時，稱此轉移點為可激發狀態(Enable)。在可激發狀態下的轉移點可以進行激發的動作。當轉移點激發完成後，會使得轉移點的輸入暫存點浮標數減少、輸出暫存點浮標數增加，其中浮標數增加與減少的數量是由箭號的權重 W 所決定，轉移點的激發規則如圖 2.9 所示。

2.3.1 詮釋性裴氏圖的定義

詮釋性裴氏圖[14]是 1992 年由 René David 與 Hassane Alla 所提出。詮釋性裴氏圖為一般裴氏圖的延伸，主要是用在描述軟體、硬體與可程式邏輯控制器等實體系統的輸出、輸入信號互動模式，以輔助使用者進行系統控制邏輯的設計與分析。在自動化製造系統的應用上，詮釋性裴氏圖可以用在表達 Grafcet。Grafcet[13]為可程式邏輯控制器的圖形式程式語言之一，其表達方式與設計原理源自於裴氏圖，但本身卻不具有如裴氏圖一般的分析技術，所以可先由詮釋性裴氏圖進行系統控制邏輯的設計與驗證，再轉換為 Grafcet 語言載入可程式邏輯控制器，此種作法能夠有助於提升控制器的執行效率。

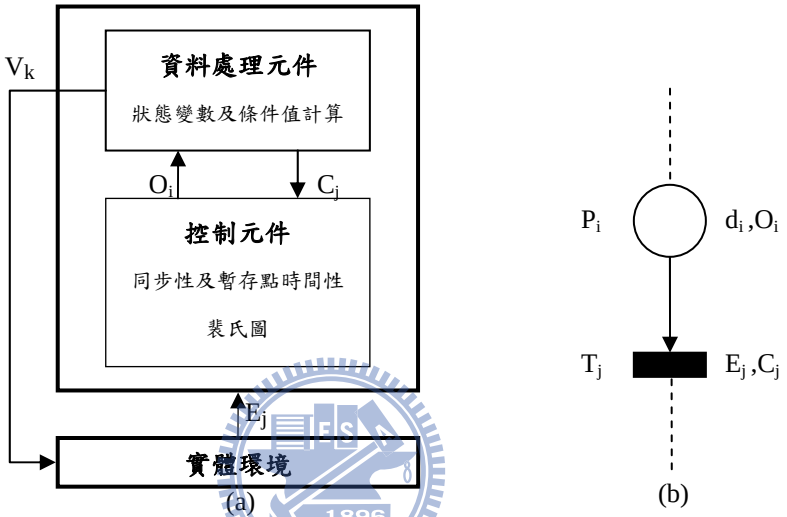


圖 2.10 詮釋性裴氏圖[13]

一個詮釋性裴氏圖的定義包含下列三項性質：同步性(Synchronized)、暫存點時間性(P-timed)、包含資料處理元件。對於同步性而言，是指裴氏圖轉移點激發與外部事件的發生存在關聯。一個同步性裴氏圖只有在轉移點為可激發狀態並且當所對應的外部事件發生時，轉移點才會進入激發狀態。

對於暫存點時間性而言，是指暫存點具有計時的特性。當浮標進入暫存點後即重新啟動計時，計時尚未結束以前進入暫存點的浮標為無法利用(Unavailable)的狀態，計時結束後浮標轉換為可利用(Available)的狀態。

資料處理元件是裴氏圖與實體環境之間的溝通介面。在詮釋性裴氏圖裡，每一個暫存點對應到一項作業行為 O_i；每個轉移點對應到一組激發條件 C_j。資料處理元件會根據目前裴氏圖中暫存點的浮標狀態來調整實體系統的整體運作行為 V_{k-}。此外，資料處理元件也會接收實體環境中所發生的事件來判斷裴氏圖轉移點激發條件滿足與否。當有轉移點的激發條件滿足時，會促使該轉移點進行激發並造成浮標狀態產生變化。圖 2.10(a)顯示一個詮釋性裴氏圖與外部環境的互動關係；圖 2.10(b)說明在詮釋性裴氏圖定義裡，將計時時間 d_i, 及作業行為 O_i 對應至暫存點；外部事件 E_j 及狀態轉移條件 C_j 對應至轉移點。

對於使用詮釋性裴氏圖來描述自動化製造系統的控制行為而言，是將詮釋性裴氏圖的轉移點對應至可程式邏輯控制器輸入模組的節點狀態以及虛擬變數的值。在可邏輯控制器

輸入模組裡，每一個節點藉由連接至感知器(Sensor)裝置接收一組一位元(1 bit)的電子訊號。當訊號回傳為 1 時，代表該節點為導通的狀態；訊號回傳為 0 時，代表該節點為斷路的狀態。可程式邏輯控制器的輸出、輸入模組又分為常開型與常閉型，常開型指的是在一般狀態下其節點皆為導通狀態，只有在感知器接收到事件發生時，如紅外線感知器感應到有遮蔽物或是液位感知感應到液位達到特定高度，會導致節點呈現斷路狀態，反之則稱常閉型。虛擬變數的作用則是協助記錄輸入模組無法表達的狀態，如經由複雜運算過後的數值或是非數值能表達的符號、字串等。控制器透過檢查輸入模組節點狀態以及虛擬變數的值，作為外部事件的發生與否的依據，以決定裴氏圖中哪些轉移點的激發條件成立。

暫存點對應到可程式邏輯控制器輸出模組節點的導通或斷路的狀態以及虛擬變數值。在可邏輯控制器輸出模組裡，每一個節點接收一組一位元的電子訊號，當給予節點的訊號為 1 時，該節點會呈現導通狀態，而連接於該節點的電子設備其電路也將會導通、當給予節點的訊號為 0 時，該節點會呈現斷路狀態，而連接於該節點的電子設備其電路也將會斷路。

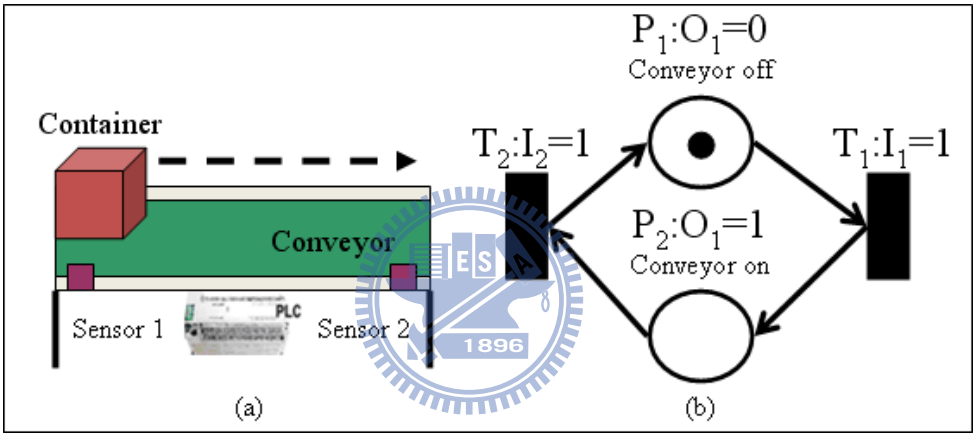


圖 2.11 輸送帶系統與詮釋性裴氏圖

以圖 2.11(a)的輸送帶系統為例，此系統內共包含四項設備—輸送帶、一號紅外線感知器、二號紅外線感知器以及可程式邏輯控制器。輸送帶的啟動與停止是由可程式邏輯控制器的輸出節點 O_1 控制，當 $O_1 = 1$ 時，即表示該輸出節點為導通狀態，此時輸送帶會啟動；反之，當 $O_1 = 0$ 時，表示該輸出節點為斷路狀態，此時輸送帶會停止。一號、二號紅外線感知器分別連接至可程式邏輯控制器的輸入節點 I_1、I_2，節點 I_1、I_2 在正常情況下為斷路狀態，只有當紅外線感知器感應到有容器進入時會回傳給可程式邏輯控制器節點導通的訊號。

對於輸送帶系統的控制邏輯可由圖 2.11(b)的詮釋性裴氏圖表達。其中，暫存點 P_1 表示輸送帶停止狀態對應到作業行為 $O_1 = 0$、暫存點 P_2 表示輸送帶啟動狀態，對應到作業行為 $O_1 = 1$、轉移點 T_1 表示容器進入輸送帶的事件，其激發條件對應到外部事件 $I_1 = 1$、轉移點 T_2 表示容器離開輸送帶的事件，其激發條件對應到外部事件 $I_2 = 1$。

2.3.2 詮釋性裴氏圖與 Grafcet

Grafcet 於 1977 年由法國研究團隊所提出。Grafcet 為法文"Graphe Fonctionnel de Commande Etape Transition "的縮寫，意指步階轉移功能圖(Step Transition Function Chart)，主要用在建構循序系統控制(Sequential System Control)模型。1988 年國際電工委員會(International Electrotechnical Commission, IEC)以 Grafcet 為基礎發展出順序功能圖(Sequential Function Chart, SFC)並列為可程式邏輯控制器 IEC 848 標準程式語言之一。1992 年順序功能圖與階梯圖(Ladder Diagram, LD)、功能方塊圖(Function Block Diagram, FBD)、結構化檔(Structured Text, ST)、指令表列(Instruction List, IL)共同列為可程式邏輯控制器 IEC 61131 標準程式語言[21]。

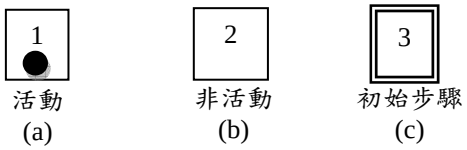


圖 2.12 Grafcet 步階表示法

Grafcet 的基本原理及元素發展至裴氏圖，兩者具有相似的的圖形表達方式。Grafcet 的三項基本構件為：步階(Step)、轉移點(Transition)、箭號(Arc)。步階相當於裴氏圖中的暫存點，其分為兩個狀態：活動(Active)、非活動(Inactive)。當步階狀態為活動時，系統即執行該步階所記錄的輸出動作，步階裡並標記一個浮標(Token)表示，如圖 2.12(a)所示。當步階狀態為非活動時，則系統不執行該步階紀錄的動作，如圖 2.12(b)所示。而系統的所有起始步階則會以兩個重疊正方形表示，如圖 2.12(c)。

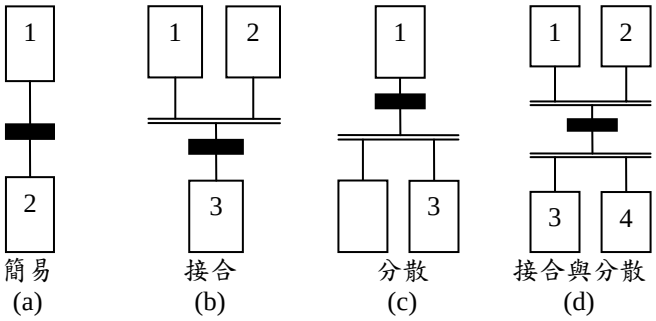


圖 2.13 Grafcet 轉移點與箭號表示法[13]

轉移點在 Grafcet 中以一條狀物表示，在轉移點內記錄著一組輸入變數的函數名為接受值(Receptivity)，當轉移點接收值為真且該轉移點的所有輸入步階的狀態均為活動時，轉移點即進行激發(Firing)。轉移點激發完成後，其所有輸入步階的狀態轉換為非活動、所有輸出步階的狀態轉換為活動。在 Grafcet 裡轉移點與箭號的表示法可分為四類如圖 2.13 所示。

Grafcet 的基本原理雖源自裴氏圖，然而兩者間之間仍有些許不同之處。以下兩點為 Grafcet 與詮釋性裴氏圖之間的主要差異：(1) Grafcet 裡一個步階僅分兩種狀態；詮釋性裴氏圖一個暫存點則依照浮標數量會有多種狀態，如圖 2.14 所示。(2) Grafcet 裡同時間內所

有可激發狀態的轉移點會一起激發；詮釋性裴氏圖裡多個可激發的轉移點間若有資源衝突(In conflict)情形時，僅能有一個轉移點進入激發狀態。如圖 2.15 所示。

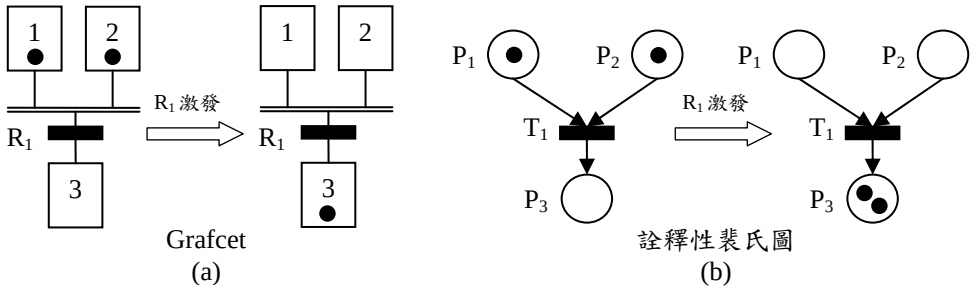


圖 2.14 Grafcet 與詮釋性裴氏圖狀態轉換一[13]

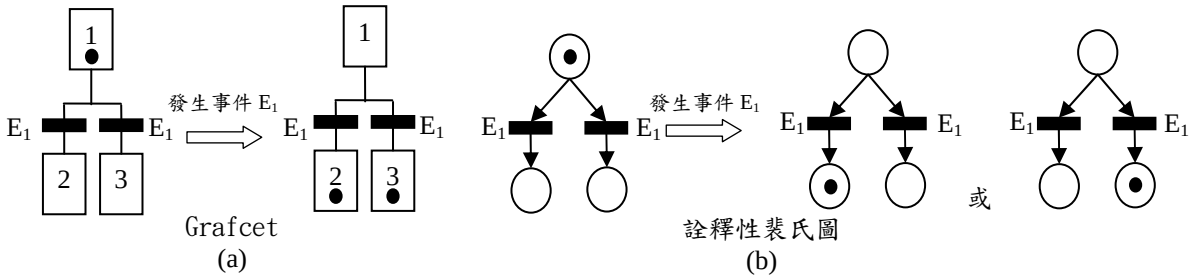


圖 2.15 Grafcet 與詮釋性裴氏圖狀態轉換二[13]

雖然詮釋性裴氏圖與 Grafcet 之間存在上述差異，但仍可由模型的設計來消弭兩者間的不同。如裴氏圖為安全性(Safe)時，換句話說每一個暫存點最多只會擁有一個浮標，則第一項差異不存在。當裴氏圖為確定性(Deterministic)或具有資源衝突(In Conflict)的轉移點所連接的外部事件不會同時發生時，則第二項差異不存在。此時詮釋性裴氏圖在表達上等同 Grafcet，相關的裴氏圖理論就可用以分析其性質。

2.3.3 詮釋性裴氏圖與法則

詮釋性裴氏圖除了可以轉換成 Grafcet 程式語言作為控制器的控制邏輯之外，也可以表達成法則(Rules)的形式說明系統狀態轉換的規則[30]，當這些法則載入即時型專家系統(Expert System)的知識庫(Knowledge Bases)內，或是使用高階程式語言如 JAVA、C、C++等將其進行撰碼之後，亦能夠循著法則所敘述之轉換規則變換控制器狀態。

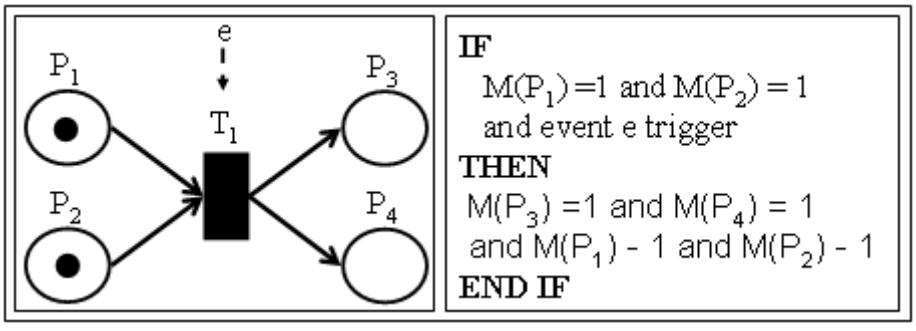


圖 2.16 詮釋性裴氏圖與法則的轉換規則

在詮釋性裴氏圖裡，每一個轉移點都可以視為是一條 IF……THEN 型式的法則，其中轉移點的激發條件及轉移點的輸入暫存點代表法則成立的條件，輸出暫存點表示的是法則條件成立時應執行的動作，詮釋性裴氏圖與法則的轉換規則如圖 2.16 所示。當轉移點的每個輸入暫存點都擁有浮標並且該轉移點的激發條件成立時，換句話說，在所有輸入暫存點所代表的控制器狀態皆滿足並且轉移點所連接的狀態轉換條件成立的情形之下，法則指示控制器應將系統狀態轉換至輸出暫存點所代表的狀態中。

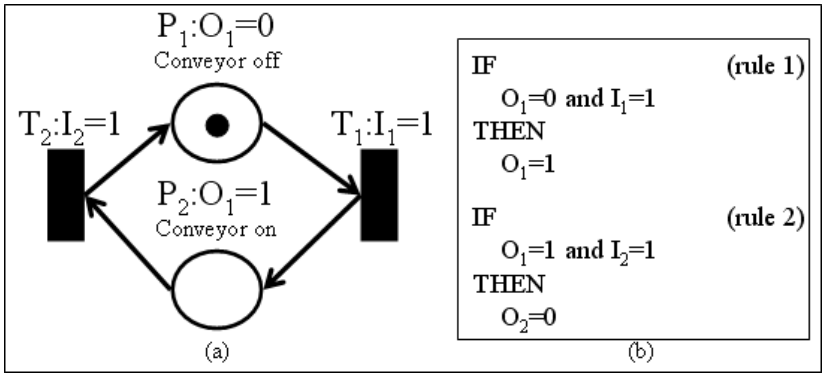


圖 2.17 輸送帶詮釋性裴氏圖及法則轉換

以圖 2.11 的輸送帶系統的詮釋性裴氏圖為例，其可表達成圖 2.17(b)的兩條法則。法則一是由轉移點 T_1 產生，其意指當輸出節點 O_1 為斷路狀態且輸入節點 I_1 為導通狀態時，則導通輸出節點 O_1，也就是指輸送帶為停止狀態並且容器進入輸送帶時啟動輸送帶；法則二是由轉移點 T_2 產生，其意指當輸出節點 O_1 為導通狀態且輸入節點 I_2 為導通狀態時，則斷路輸出節點 O_1，也就是指輸送帶為啟動狀態並且容器離開輸送帶時停止輸送帶。又這裡的兩個暫存點所代表的都是輸送帶的狀態，而單一物件同時間下僅會處於一種狀態中，也就是說當暫存點 P_1 內有浮標時，暫存點 P_2 並定不會有浮標，反之亦然，所以在圖 2.17(b)的法則內容裡就可以主動省略掉轉移點觸發時需要進行輸入暫存點浮標數減少的步驟。

2.3.4 裴氏圖加註語言

可加註語言(eXtensible Markup Language, XML)[23]。是 1996 年由全球資訊網發展協會(World Wide Web Consortium, W3C)[22]所發展出的一種電子檔語法標準，其主要是透過已制定的標籤格式記錄檔內容，使得電腦間能夠利用這種共通的檔型式進行資訊分享，可加註語言檔的撰寫方式如圖 2.18。

```
<?xml version="1.0" encoding="iso-8859-1"?>
<example>
   <tagName>
     <sub-tag1>tag1_attribute_value1</sub-tag1>
     <sub-tag2>tag1_attribute_value2</sub-tag2>
   </tagName>
   <tagName>
     <sub-tag1>tag2_attribute_value1</sub-tag1>
     <sub-tag2>tag2_attribute_value2</sub-tag2>
   </tagName>
</example>
```

圖 2.18 可加註語言範例

圖 2.18 中<標籤名稱>代表宣告標籤開始，</標籤名稱>代表宣告標籤結束，在標籤的開始與結束之間則可記錄標籤的值或者是插入多個子標籤來描述父標籤的屬性。目前已有許多種針對不同目的所定義出的可加註語言格式，包括數學檔使用的數學加註語言(Mathematical Markup Language, MathML)、音樂檔使用的 MusicXML、圖型檔使用的 GraphML，而裴氏圖加註語言則是針對裴氏圖模型所制定出的可加註語言格式[24]。

在裴氏圖加註語言文件內分別是以<place></place>、<transition></transition>、<arc></arc>三種標籤記錄裴氏圖模型中暫存點、轉移點、箭號資訊。暫存點標籤底下包含有暫存點識別<id>、暫存點名稱<name>、座標位置<position>、初始浮標狀態<initialMarking>等屬性的資訊；轉移點標籤底下包含有轉移點識別<id>、轉移點名稱<name>、座標位置<position>等屬性的資訊；箭號標籤底下有箭號識別<id>、箭號名稱<name>、箭號的來源<source>、箭號的目標<target>、箭號的權重<value>、箭號的轉折點位置<arcpath>等屬性的資訊，圖 2.19 為一範例裴氏圖加註語言檔以及與其相對應的裴氏圖模型。

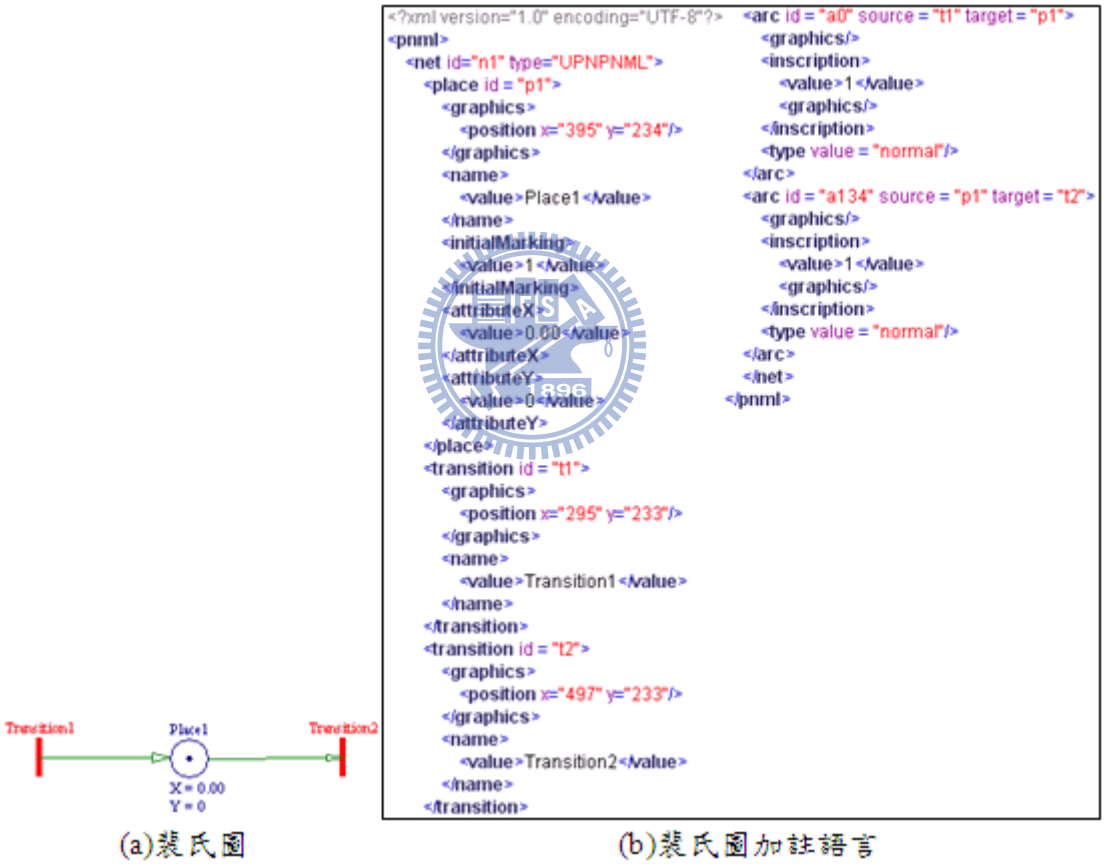


圖 2.19 裴氏圖加註語言表達法

2.3.5 優缺點說明

在設計自動化製造系統時，詮釋性裴氏圖的優點在於其善於表達控制器的平行處理以及動態行為模擬，並且裴氏圖本身擁有良好的數學性質及許多的分析工具可以用來驗證監控模式的設計的正確性。而通過驗證的詮釋性裴氏圖能夠經由轉換成 Grafcet 程式語言或是法則的型式方便實作出控制器的控制邏輯。

然而，詮釋性裴氏圖主要在於規範系統的正常行為。當例外情形出現時，例如裝置未

被控制器順利的啟動、關閉或是裝置受外力幹擾而意外被啟動、關閉，控制器無法單由詮釋性裴氏圖所建構出的控制邏輯察覺錯誤的發生，甚至於進行錯誤的診斷以及錯誤的回復。以圖 2.20(a)的兩機台製造系統裴氏圖為例，其中法則轉移點 T_1 顯示在滿足輸出節點 O--$_1$ 為斷路狀態(O_{-1} = 0)、O_4 為斷路狀態(O_4 = 0)，且輸入節點 I_1 為導通狀態(I_1 = 1)的情形下，控制器會分別導通輸出節點 O_1 及 O_4。但實際上，控制器無法就所產生的法則確認是否一旦輸出節點 O_1 及 O_4 導通後，一號機台以及作業警告鈴能夠順利被啟動或可能發生裝置受損而無法順利被啟動的情形。此外，假使一號機台已順利被啟動，但在作業過程中意外被關閉時，控制器同樣無法偵查到錯誤的發生，而使得控制器繼續激發其他的轉移點。所以在詮釋性裴氏圖中需要增加對於例外狀況的察覺能力，使控制器除了能夠藉由裴氏圖來進行監控作業之外，亦能夠從中研判出例外狀況的發生

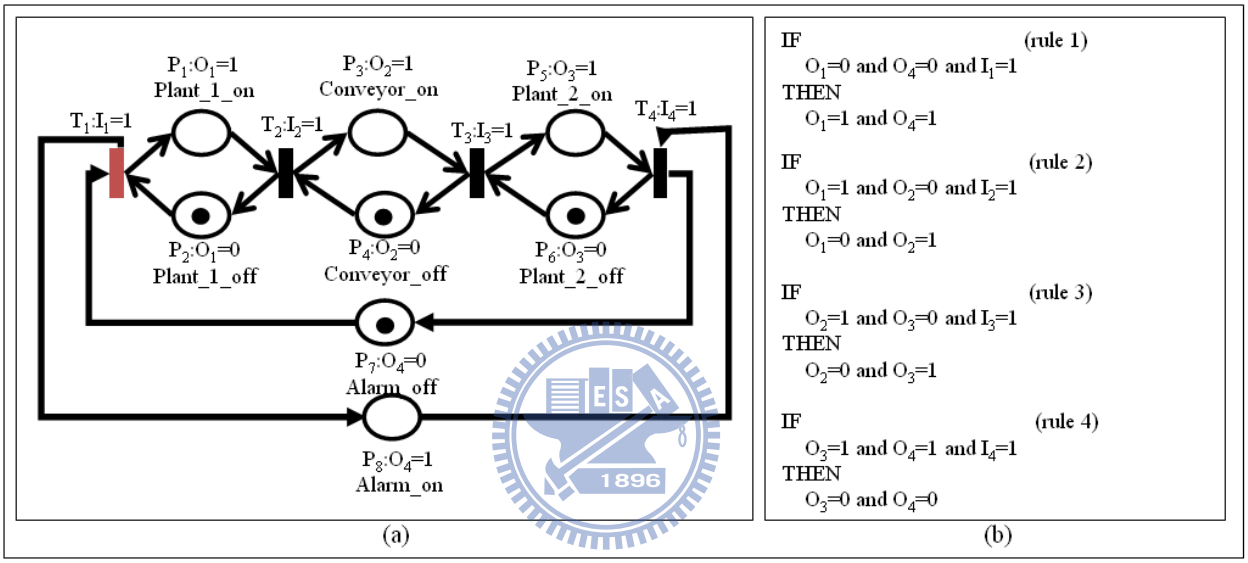


圖 2.20 兩機台製造系統裴氏圖及法則

2.4 開放資料庫連接架構

開放資料庫連接(Open Data Base Connectivity, ODBC)標準是由 X/Open SQL Acess 團體所開發出的一組標準應用程式介面[29]，它是以 SQL(Structured Query Language)標準語法作為標準指令規格，讓使用者在撰寫應用程式時能夠單純藉由 SQL 指令去連接到各種類型資料庫管理系統的資料庫。目前大多數的資料庫管理軟體皆有提供使用者有關開放資料庫連接標準的支援，如 Microsoft SQL Server、Microsoft Access、Oracle、DB2 等等。

在開放資料庫連接架構中，主要可以分成五個元件：(1)應用程式(Application)、(2)驅動程式管理員(Driver Manager)、(3)驅動程式(Driver)、(4)資料庫管理系統(Data Base Management System, DBMS)與(5)資料庫，如圖 2.21 所示。

圖 2.21 開放資料庫連接架構

對於「應用程式」而言，這是用來接收與回覆使用者回應的溝通介面，它會將使用者的命令轉換成 SQL 指令語法，並且呼叫 ODBC API 內的函數進行 SQL 指令的處理。

對於「驅動程式管理員」而言，這是用來載入與卸載資料庫管理系統的驅動程式。此外，它也負責 SQL 指令的驗證，針對已通過驗證的 SQL 指令，驅動程式管理員會再呼叫對應的驅動程式函數進行處理，否則回傳錯誤訊息至應用程式。

對於「驅動程式」而言，這是用來進行 SQL 指令與資料庫管理系統專屬指令之間的轉譯工作以及回傳資料查詢的結果。一般而言，驅動程式是由資料庫管理系統供應商設計並提供給使用者使用。

對於「資料庫管理系統」與「資料庫」而言，這是用來儲存使用者欲紀錄的資料，資料庫管理系統並允許使用者透過線上交易處理(On-Line Transaction Processing, OLTP)[28]的方式來新增、修改、刪除資料記錄。

開放資料庫連接架構的優點在於它讓使用者只需要面對同一種指令語法就可以與各種類型資料庫管理系統連接以及進行資料庫資料的處理，所以即使往後更換使用了其他類型的資料庫管理系統，使用者也不需要擔心必須再重新學習另外一套的指令語法。促成這項優點最大的原因，即在於應用程式與資料庫管理系統之間的連接方式並不是由應用程式直接使用資料庫管理系統的 API 以及其自訂的程式語法與其溝通，而是會先透過驅動程式作一次指令轉譯的動作，將標準 SQL 語法轉譯成為該資料庫管理系統自訂的程式語法，所以在這樣的連接架構底下，就能夠讓應用程式與各種資料庫管理系統之間具有良好的互用性。本研究中，會以開放資料庫連接架構為概念設計一個開放資料伺服器連接架構，目的在於改善以往監控軟體與各種控制器規格之間不具有互用性的問題。

第三章 開放資料伺服器連接架構的設計

本章說明開放資料伺服器連接架構的建置以及架構中各模組的規格開發。3.1 節說明開放資料伺服器連接架構，在該架構中可以得到三大模組，此三大模組會分成三節分析之，即 3.2 節決策系統與控制器之間的“資料轉譯介面”設計，這裡稱之為“資料伺服器”，3.3 節決策系統與資料伺服器之間的“資料傳送介面”設計，這裡稱之為“決策系統連接介面”3.4 節資料伺服器與驅動程式之間的“資料傳送介面”設計，這裡稱之為“驅動程式連接介面”。

3.1 開放資料伺服器連接架構

開放資料伺服器連接架構可以分成五個部分：(1)決策系統、(2)資料連接介面、(3)驅動程式(Driver)、(4)控制器及(5)製造系統，如圖 3.1 所示。對於「決策系統」而言，這是指由裴氏圖導向的監控員、偵查員、維修員軟體所組成的自動化製造系統控制邏輯。決策系統的功能在於隨著控制器的狀態變化，即時判斷應採取的控制行為，並向控制器下達對應的監控指令。

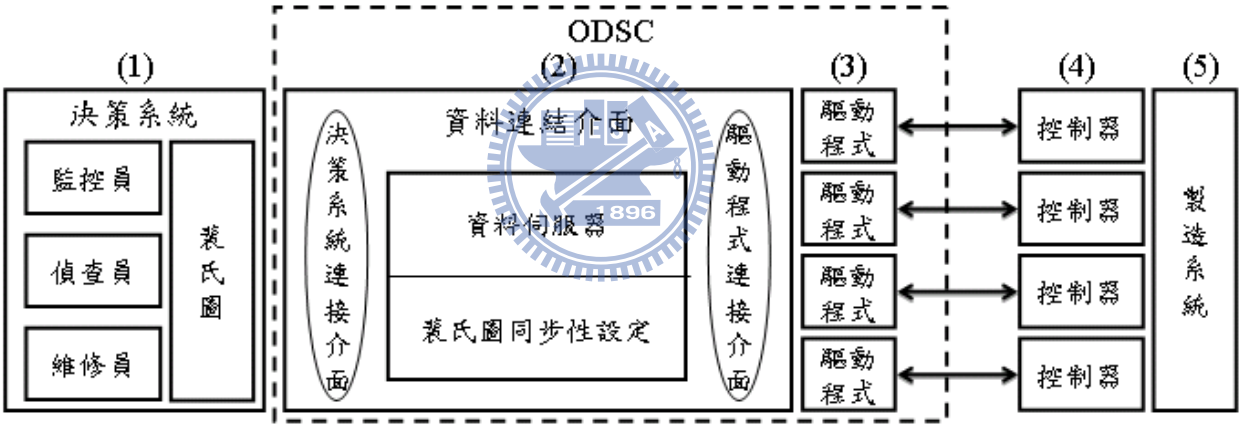


圖 3.1 開放資料伺服器連接架構

對於「資料連接介面」而言，這是指決策系統與控制器之間的一項仲介軟體，其主要功能是將裴氏圖所表達的監控邏輯轉譯成對應的控制器輸出、輸入行為。決策系統所傳送的監控指令會經由資料伺服器內部記錄的裴氏圖同步性設定，轉換為相對應的控制器信號輸出資訊，傳送至驅動程式。另一方面，資料伺服器也接收由驅動程式所傳回的控制器信號輸入資訊，並經由裴氏圖同步性設定轉換為相對應的裴氏圖狀態回傳至決策系統。

對於「驅動程式」而言，這是指資料連接介面以及控制器之間的資料轉譯程式，其內部包含對於特定類型控制器的基本操作指令，如與控制器連線、關閉控制器連線、導通或斷路輸出模組節點以及擷取輸入節點的狀態。當接收到資料伺服器所傳送的控制器輸出資訊時，驅動程式會將其轉譯成該類型控制器接受的指令格式，再傳送至控制器。此外，驅動程式也接收由控制器傳回的資訊，並轉譯成資料連接介面定義的標準資料格式。

對於「控制器」與「製造系統」而言，控制器會依驅動程式下達的指令進行電子信號輸出，藉以控制製造系統裡各項電子設備、裝置的作業行為。此外，控制器也接收電子設

備、裝置所傳回的電子信號並回傳至驅動程式。

四階層的開放連接架構，主要是基於將決策系統與資料伺服器之間，以及資料伺服器與控制器之間兩階段的指令轉譯工作分離為概念所設計而成，其設計方法是透過模組化(Modulize)的方式讓單一軟體只需負責一項單純的功能，並且各軟體之間透過標準介面來和其他軟體溝通，避免與其他軟體間有太多的關聯。具體而言，這是在資料連接介面與控制器之間加入了一個驅動程式作為中繼軟體，負責進行第二次的轉譯動作，所以決策系統下達的指令會先由資料伺服介面轉譯成相對應的控制器行為，驅動程式再利用控制器的專屬指令語法去轉譯資料伺服介面所表達的控制器行為，使資料伺服器軟體與控制器之間達到獨立性。

這意味著在資料伺服器軟體的開發過程中，只要軟體開發人員依循所規範的標準規格進行實作，無需在乎未來所要連接的控制器類型，就能藉由替換驅動程式來完成與各類型控制器銜接的工作。整體而言，開放資料伺服器連接架構的優點在於：(1)提升監控軟體的移植能力，其解決了以往資料伺服器軟體與單一機型控制器的作業環境過度相依，所以當控制器類型作變更時，必須藉由大量修改程式碼來讓已設計好的軟體適應不同作業環境的問題。(2)軟體開發人員不用浪費過多的資源在改寫現存的軟體組件以適應不同規格的控制器，只需要針對不同規格控制器搭配使用該控制器的驅動程式。(3)藉由模組化的方式讓軟體間的關聯被限制到最低，所以當單一軟體需要修改時，其他軟體所受到的影響較少，這讓軟體維護作業更為方便。(4)由於使用相同的規格標準，所以每一個模組可以被獨立開發，這在團隊開發上會是一項優點。(5)資料伺服器軟體被開發出來後可以使用在各種規格控制器中，驅動程式被開發出來後也能夠提供給所有相同規格的資料伺服器軟體使用，這提升了軟體的重覆使用性。

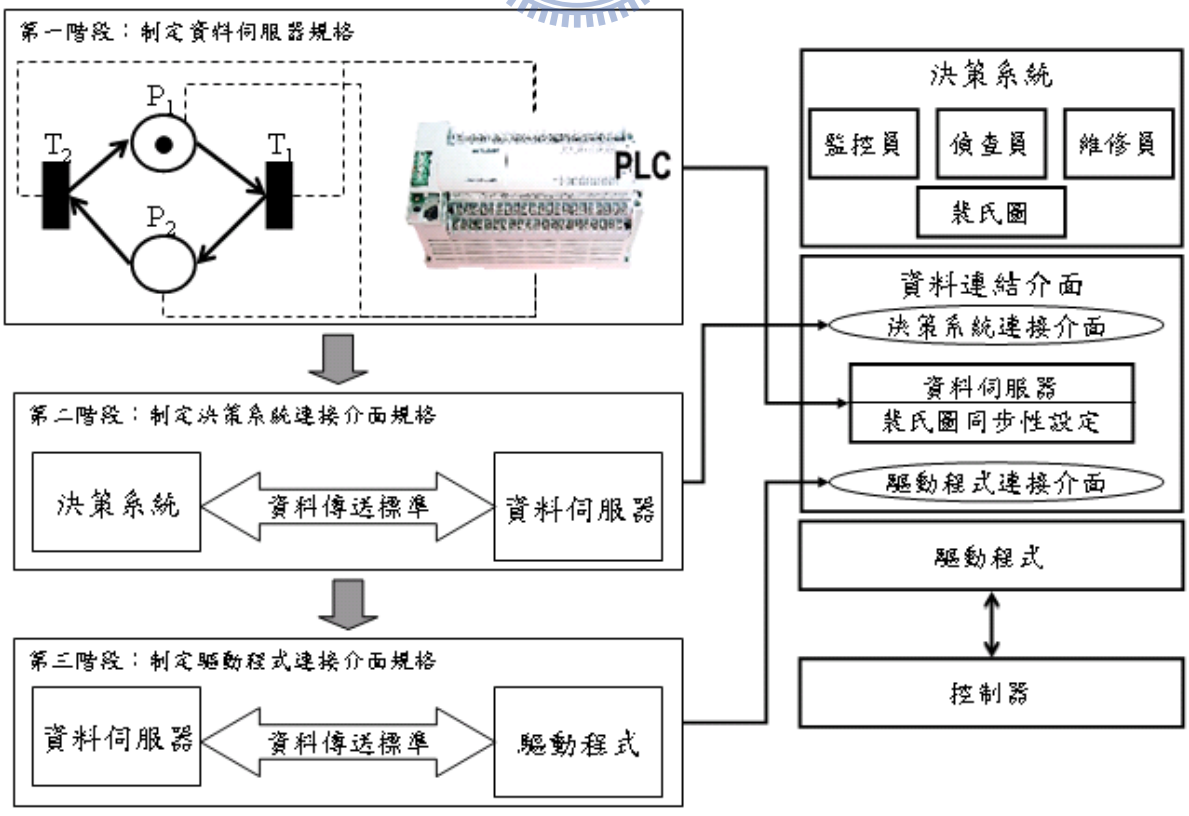


圖 3.2 資料伺服器標準規格設計流程

開放資料伺服器連接架構的建置過程中，最重要的是建立資料伺服介面的標準規格，讓使用者可以依此標準進行資料伺服介面的設計。在資料伺服介面的規格制定過程中可以分成「資料伺服器設計」、「決策系統連接介面設計」、「驅動程式連接介面設計」三個階段，如圖 3.2 所示。

第一階段「資料伺服器規格設計」：是定義裴氏圖與控制器之間的同步性設定方式。由於在這裡所使用的決策系統，是指裴氏圖導向的監控員、偵察員、維修員軟體，同化性設定的目的即是為了將控制器輸出、輸入模組中各節點的狀態與裴氏圖裡的各項元件連接，使得裴氏圖裡的控制邏輯能夠轉譯成控制器端相對應的控制行為。此部分於 3.2 節說明。

第二階段「決策系統連接介面設計」：是針對定義決策系統與資料伺服器之間進行資料傳送時所使用的函式、方法規格以及資料格式，讓資料伺服器開發人員能夠依循此通用標準設計其資料伺服器軟體。此部分於 3.3 節說明。

第三階段「驅動程式連接介面設計」：是定義資料伺服器與驅動程式之間進行資料傳送時所使用的函式、方法規格以及資料格式，使控制器開發人員能夠依循此通用標準設計其驅動程式。此部分於 3.4 節說明。

3.2 「資料伺服器」的規格設計

資料伺服器的主要功能是進行決策系統與驅動程式之間的資訊轉譯，其內部定義了控制器與裴氏圖之間的同步性設定，也就是控制器的輸出、輸入信號狀態與裴氏圖中各個轉移點及暫存點的連接關係定義。資料伺服器會透過裴氏圖的同步性設定將決策系統的指令表示成對應的控制器輸出信號狀態並傳送至驅動程式。另一方面，也從驅動程式端接收有關控制器的輸入信號狀態，並轉譯成裴氏圖資訊傳回至決策系統。

在詮釋性裴氏圖中，同步性設定方式是將控制器的輸入信號狀態連接至轉移點、控制器的輸出信號狀態連接至暫存點。在這裡，首先會進一步把控制器的輸入信號分成事件觸發信號以及設備狀態訊號。事件觸發信號用來表達實體環境中特定事件的發生，設備狀態訊號則來表達各項實體設備的運作狀態。接著調整詮釋性裴氏圖的同步性設定方式，讓設備狀態信號對應至暫存點，事件觸發信號與輸出信號對應至轉移點，所以裴氏圖浮標狀態能夠顯示出實體環境所發生的正常、異常行為資訊，並提供給監控軟體進行分析。調整後的裴氏圖稱為修正詮釋性裴氏圖。3.2.1 節說明修正詮釋性裴氏圖的符號定義。3.2.2 節說明修正詮釋性裴氏圖的同步性設定方式。3.2.3 節說明修正詮釋性裴氏圖的法則轉換方式。

3.2.1 修正詮釋性裴氏圖的符號定義

修正詮釋性裴氏圖共包含十一個元素 PN = (P, T, F, W, M_0, S, E, O, f_p, f_t)。其中，(P, T, F, M_0)代表一組二元標記(Binary Marked)、連接(Connected)、普通(Ordinary)且轉移點時間性(T-timed)的裴氏圖。「二元標記」是指任何狀態下暫存點內最多只會有一個浮標存在。「連接」是指任意兩個節點(Node)間至少存在一條可以到達的路徑。「普通」是指每一條箭號的權重皆為一，「轉移點時間性」是指轉移點具有計時特性。

集合 S 包含實體系統的各元件狀態，S = {$S_{1,1}$, $S_{1,2}$, $S_{1,3}$……$S_{m,n}$}，$S_{k,i}$ 為元件 k 的狀態 i。

集合 E 包含實體系統的觸發事件。E = {e, E_1, E_2, E_3…….E_m}，e 代表一虛擬的無間斷發生的觸發事件。

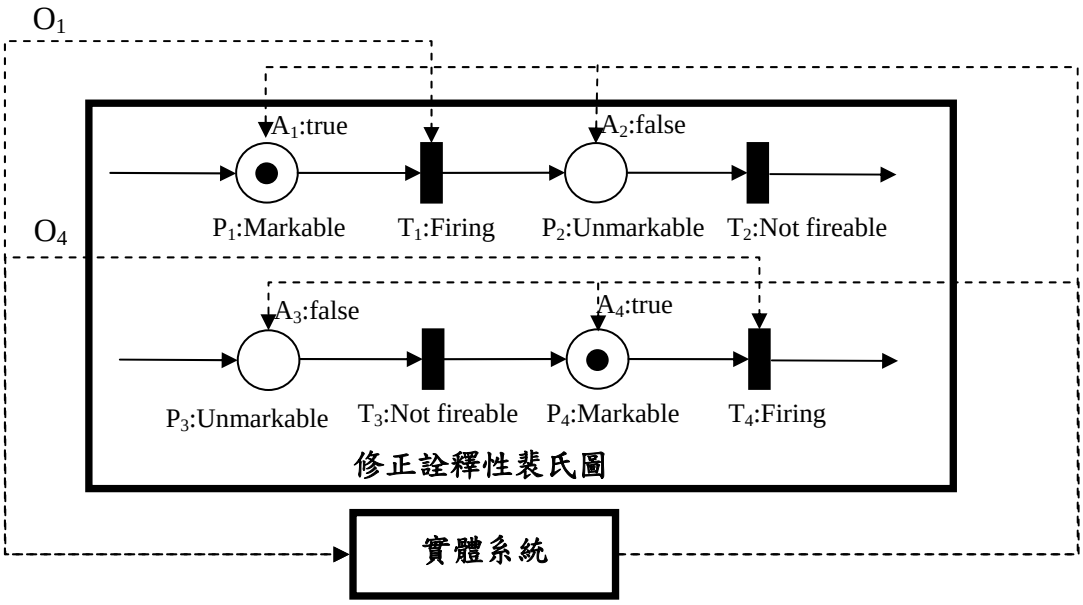
集合 O 包含實體系統的各元件作業。O = {$O_{1,1}$, $O_{1,2}$, $O_{1,3}$…....$O_{m,n}$}，$O_{k,i}$ 代表元件 k 的作業 i。

函數 f_p 為暫存點 P_i 與實體系統元件狀態的對應。$f_p(P_i)$ = S’，S’為元件狀態集合 S 的一組子集合。

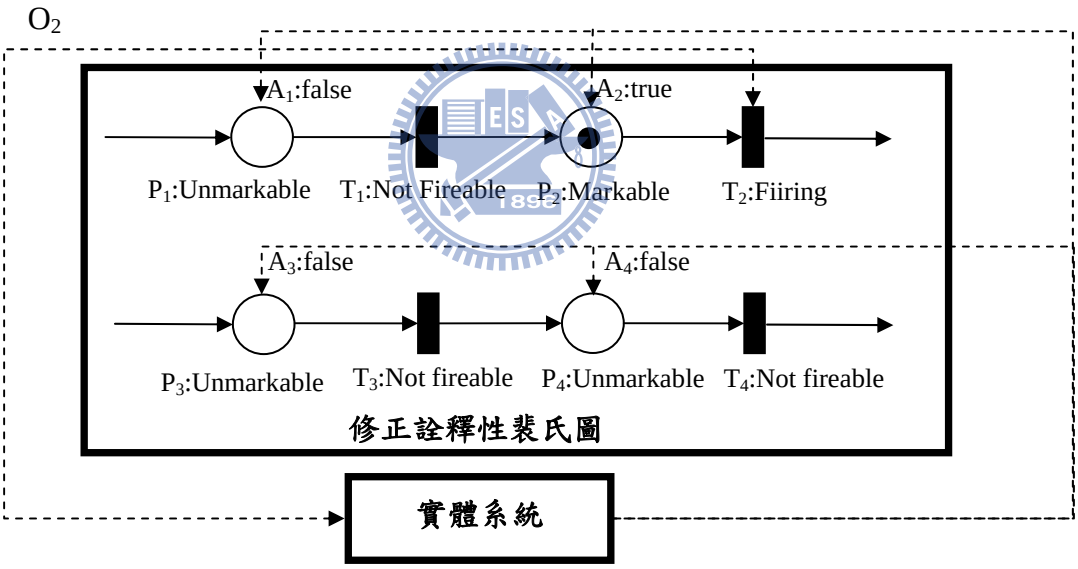
函數 f_t 為轉移點 T_i 與實體系統元件作業、觸發事件以及延遲啟動時間的對應。$f_t(T_i)$ = (E’,O’, ms)，E’為觸發事件集合 E 的一組子集合，O’為元件作業集合 O 的一組子集合。ms 為轉移點的延遲啟動時間，其單位為毫秒。轉移點的計時器會在轉移點為可激發狀態下並且接收到觸發事件後開始計時，當計時終了後才執行元件作業。

對於修正詮釋性裴氏圖的動態行為而言，當轉移點的所有輸入暫存點都擁有浮標時，該轉移點為可激發狀態。所有可激發狀態的轉移點均會同時激發。激發完成的轉移點會將其輸入暫存點轉為不可標記(Unmarkable)狀態、輸出暫存點轉為可標記(Markable)狀態，可標記狀態的暫存點代表實體系統元件目前應該停留的狀態。對於浮標分佈而言，修正詮釋性裴氏圖是依據目前實體系統裡各個元件的狀態資訊以及暫存點是否為可標記狀態給予

浮標，所以當暫存點具有浮標時即表示元件的預期狀態與實際狀態吻合。修正詮釋性裴氏圖與實體系統的互動如圖 3.3 所示。



(a) T_1、T_4 激發



(b) T_1、T_4 激發完畢，T_2 激發

圖 3.3 修正詮釋性裴氏圖

3.2.2 修正詮釋性裴氏圖的同步性設定

對於修正詮釋性裴氏圖與外部環境之間的同步化，是藉由函數 $f_t(T_i) = (E',O')$對應轉移點至一組實體系統的作業以及一組作業的觸發事件，函數 $f_p(P_i) = S'$對應暫存點至一組元件狀態。在控制器裡，系統的作業是指可邏輯程式控制器輸出節點的導通或斷路、變更虛擬變數的值或是呼叫程式函式進行複雜運算。觸發事件以及元件狀態都可以指的是特定的可邏輯程式控制器輸入節點狀態或是位於特定值域的虛擬變數，不同之處在於元件狀態專指由控制器的作業輸出端再接回至控制器狀態輸入端的回授資訊。

以 3.4(a)的輸送帶系統為例，系統中包含了三種元件—輸送帶、一號紅外線感知器以及二號紅外線感知器，各元件與可程式邏輯控制器輸入、輸出節點的對應如表 3.1。

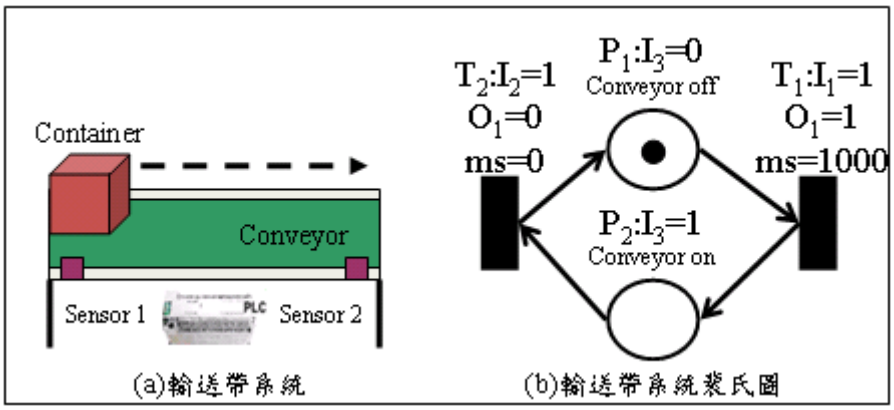


圖 3.4 輸送帶系統與修正詮釋性裴氏圖

表 3.1 輸送帶系統輸入、輸出節點說明

元件	輸入節點	輸出節點
一號紅外線感知器	I_1	無
二號紅外線感知器	I_2	無
輸送帶	I_3	O_1

對於系統的作業流程而言，當一號紅外線感知器偵測到容器進入輸送帶後，會啟動輸送帶直到二號紅外線感知器偵測到容器離開輸送帶後再將輸送帶停止，其修正詮釋性裴氏圖可表達如圖 3.4(b)。

在信號傳遞方面，一號紅外線感知器在偵測到容器進入輸送帶時，會回傳信號 1 至可程式邏輯控制器的輸入節點 I_1 表示「容器進入」事件的發生。控制器接收到「容器進入」的事件發生後，會導通其輸出節點 $O_1(= 1)$以啟動輸送帶。二號紅外線感知器在偵測到容器離開輸送帶時，會回傳信號 1 至可程式邏輯控制器的輸入節點 I_2 表示「容器離開」事件的發生。當控制器接收到「容器離開」的事件發生後，會啟動計時器進行計時，並且於一秒後斷路其輸出節點 $O_1(= 0)$以停止輸送帶。在輸送帶行進過程中，輸送帶會藉由輸入節點 I_3 回傳其回授狀態至控制器，當 $I_3 = 1$ 時表示輸送帶處於啟動狀態、$I_3 = 0$ 時表示輸送帶處於停止狀態。

此範例中，元件狀態集合 S = {輸送帶已停止($I_3 = 0$)、輸送帶已啟動($I_3 = 1$)}、作業觸發事件集合 E={容器進入($I_1 = 1$)、容器離開($I_2 = 1$)}、元件作業集合 O = {啟動輸送帶($O_1 = 1$)、停止輸送帶($O_1 = 0$)}。而裴氏圖的同步性則可以分別定義 f_t、f_p 函數為 $f_t(T_1) = (\{I_1 = 1\}, \{O_1 = 1\}, 0)$、$f_t(T_2) = (\{I_2 = 1\}, \{O_1 = 0\}, 1000)$；$f_p(P_1) = \{I_3 = 0\}$，$f_p(P_2) = \{I_3 = 1\}$。

3.2.3 修正詮釋性裴氏圖的法則轉換

在修正詮釋性裴氏圖內，每一條轉移點可以表達成一組兩層式巢狀結構的 IF…..THEN

型式的監控法則，如圖 3.5 所示。對於法則外層結構而言，這是指當轉移點的輸入暫存點都具有浮標時可執行其內部的敘述式。對於法則內層結構而言，這是指當作業的觸發事件成立時啟動計時器，並且在計時器計時終了時再執行其內部的敘述式。

當裴氏圖為選擇型裴氏圖時，相同的法則外層結構下，可能包含有多組不同的內層法則，這是因為選擇型裴氏圖在同一個浮標狀態下會依據觸發事件的不同，選擇不同的轉移點進行轉移點，而每一個內層法則都會對應到一個特定的轉移點。

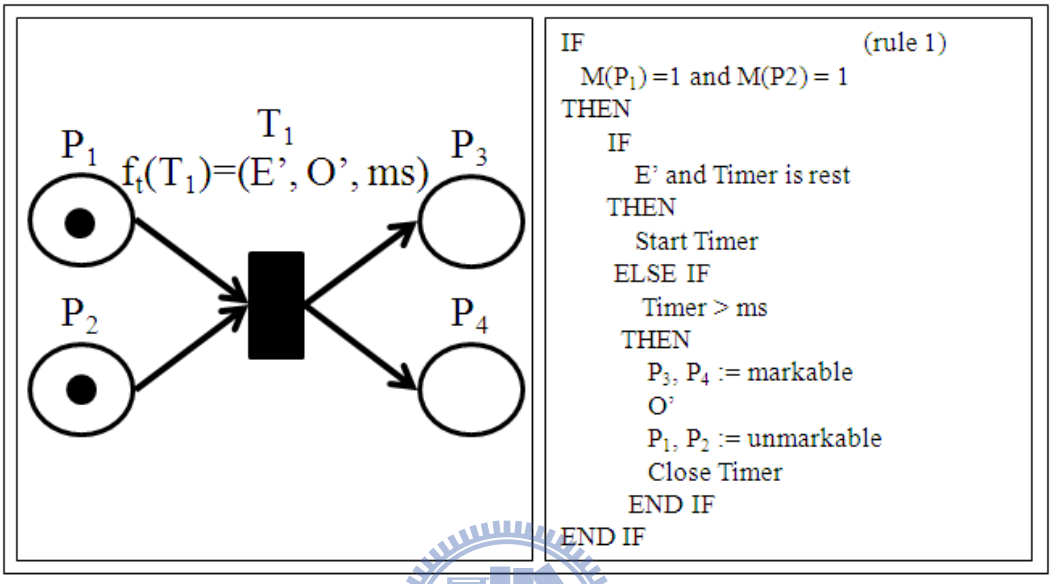


圖 3.5 修正詮釋性裴氏圖監控法則轉換

在法則內部的計時器計時終了後，所會執行的敘述式可以分成四個部分。首先是將轉移點的輸出暫存點更新為可標記狀態。接著執行各項元件作業，並且將輸入暫存點更新為不可標記狀態。最後，關閉計時器。先更新輸出暫存點狀態的作法可以確保浮標分佈轉換不會在控制器進行輸出行為時，導致浮標數目銜接錯誤的情形。以圖 3.4(b)的輸送帶系統裴氏圖為例，其可產生兩條監控法則，如圖 3.6 所示。

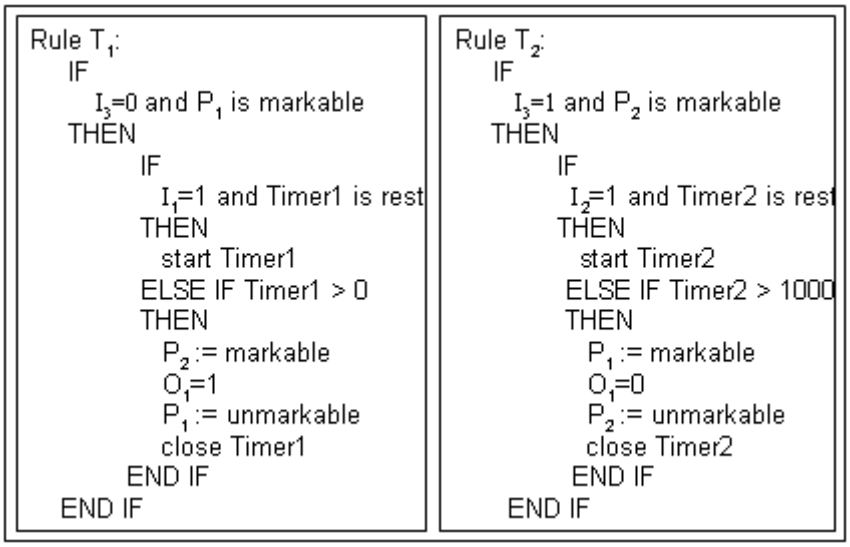


圖 3.6 輸送帶系統監控法則

3.3 決策系統連接介面設計

資料伺服器與決策系統間的互動模式如圖 3.7 所示。在裴氏圖物件裡包含了暫存點與轉移點物件，資料伺服器會將控制器回傳的資訊轉換成暫存點的浮標狀態及轉移點的激發事件儲存於其中。監控員、偵查員軟體再依目前裴氏圖物件的浮標資訊，判斷系統應執行的控制行為，這種方式取代了以往藉由虛體製造系統保存控制器資訊的作法。

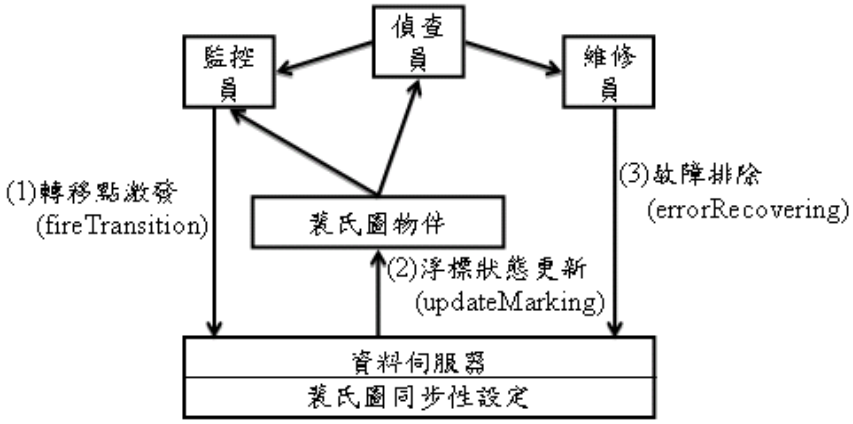


圖 3.7 資料伺服器與決策系統的互動模式

在修正詮釋性裴氏圖中，控制器的輸出信號與事件觸發輸入信號是連接於轉移點，設備的狀態輸入信號連接於暫存點。資料伺服器會根據驅動程式所傳回的控制器輸入信號資訊以及裴氏圖的同步性設定來更新裴氏圖物件的浮標狀態。監控員軟體則是會檢視目前裴氏圖物件的浮標狀態，檢查可以激發的轉移點，並傳送給資料伺服器轉移點的激發指令。資料伺服器接到激發指令後，會對照到該轉移點的同步性設定，並且取得相對應的控制器作業指令傳送給驅動程式進行第二次的轉譯作業。

偵查員軟體會從裴氏圖物件的浮標狀態來判斷控制器的運作情形是否發生異常，當異常情形發生時即呼叫維修員進行系統診斷與修復的作業。維修員會從偵查員所提供的錯誤訊息診斷錯誤發生的可能原因，並且通知資料伺服器進行故障排除。

針對上述決策系統與資料伺服器之間的互動模式，在資料伺服器軟體的設計規格中，分別制定了三個公開的方法作為決策系統與資料伺服器之間溝通介面，例如(1)轉移點激發時，即呼叫 fireTransition 方法、(2)浮標狀態更新時，即呼叫 updateMarking 方法、(3)故障排除時，即呼叫 errorRecovering 方法。

圖 3.8 fireTransition 方法的作業流程

對於「fireTransition」方法而言，這是用來提供給監控員軟體執行轉移點激發作業。在呼叫此方法的同時，必須傳入一個整數型態的參數以告知欲進行激發的轉移點編號。此方法的程式內容主要記錄了各個轉移點所對應的兩層式巢狀法則的內層結構，外層結構條件式的判斷部分則會撰寫在監控員軟體中。當此方法受到監控員軟體呼叫後，會依據所傳入的參數找到該轉移點對應的法則，並依照法則內容傳送信號輸出指令。fireTransition 方法的作業流程如圖 3.8 所示。

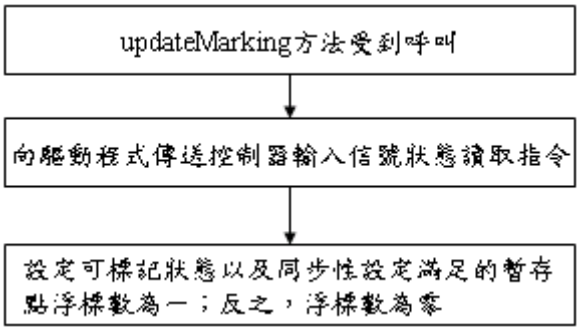


圖 3.9 updateMarking 方法的作業流程

對於「updateMarking」方法而言，這是用來更新裴氏圖物件的浮標分佈狀態。此方法被呼叫後，首先會向驅動程式傳送控制器輸入信號狀態讀取的指令。在接收到驅動程式傳回的資訊後，接著會逐一檢查各個暫存點是否為可標記狀態，以及該暫存點所對應的同步性設定是否滿足。當暫為可標記狀態並且同步性設定滿足時，則設定此暫存點的浮標數為一，否則設定浮標數為零。updateMarking 方法的作業流程如圖 3.9 所示。

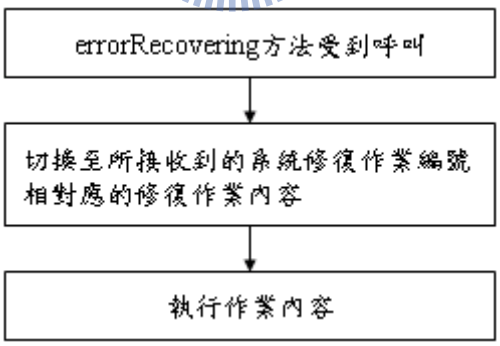


圖 3.10 errorRecovering 方法的作業流程

對於「errorRecovering」方法而言，這是用來進行系統錯誤排除所使用。在呼叫此方法的同時，必須傳入一個整數型態的參數告知欲執行的修復作業編號。此方法的程式內容主要是記載了各種的系統修復流程，當此方法受到維修員軟體呼叫以後，會依據所傳入的參數執行相對應的系統修復作業。errorRecovering 方法的作業流程如圖 3.10 所示。

3.4 驅動程式連接介面設計

圖 3.11 顯示資料伺服器與驅動程式之間的一個基本通訊流程，其中包含(1)建立控制器連線、 (2)擷取輸入節點狀態、(3)導通或斷路輸出節點、(4)結束控制器連線四項作業。針對上述的作業，在驅動程式的設計規格中分別制定了四個公開方法作為資料伺服器與驅動程式之間的溝通介面，例如(1)建立控制器連線時，即呼叫 openPLCConnection 方法、(2)擷取輸入節點狀態時，即呼叫 getPLCInputState 方法(3)導通或斷路輸出節點時，即呼叫 Handle 方法、(4)結束控制器連線時，即呼叫 closePLCConnection 方法。

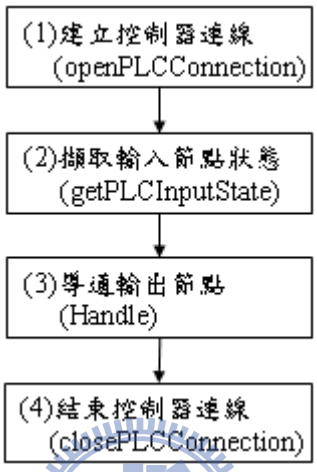


圖 3.11 資料伺服器與驅動程式的基本通訊流程

對於「openPLCConnection」方法而言，這是用來建立與控制器之間的連線。呼叫此方法的同時，必須傳入二個字串型態的參數以告知驅動程式欲與之建立連線的控制器的網際網路通訊協定(Internet Protocol, IP)位置與連接埠(Port)。此方法的程式內容主要是由各類型控制器自訂的連線語法所組成，其會依據所傳入的 IP 位置及連接埠嘗試與遠端控制器進行連線，並於此方法執行結束後會回傳一個布林值說明本次連線的作業狀態。當所傳回的布林值為真(True)時，代表成功連線、布林值為假(False)時代表連線失敗。 openPLCConnection 方法的作業流程如圖 3.12 所示。

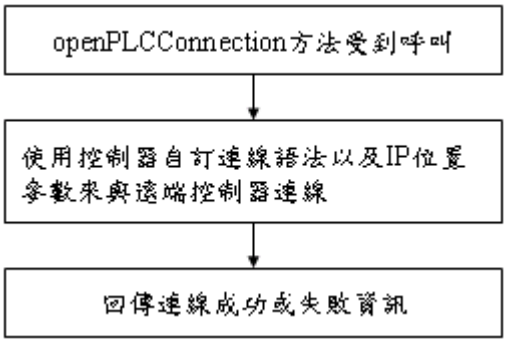


圖 3.12 openPLCConnection 方法的作業流程

對於「getPLCInputState」方法而言，這是用來取得控制器輸出模組狀態資訊。此方法

受呼叫後，會回傳一組以二進位數值表達的字串。如同輸出節點的導通與斷路指令，在二進位元數值中，由右至左的每一個位元依序對應到輸入模組上從編號零開始的節點位置，而值為 1 的位元則代表該節點為導通狀態、值為 0 的位元位該節點為斷路狀態。getPLCInputState 方法作業流程如圖 3.13 所示。

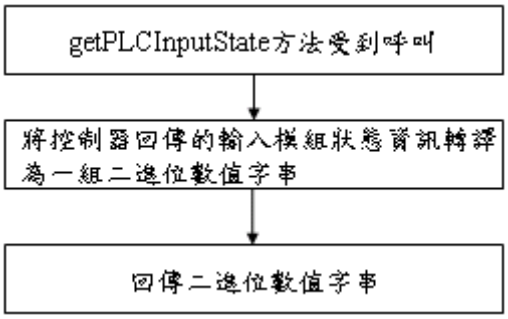


圖 3.13 getPLCInputState 方法的作業流程

對於「Handle」方法而言，這是用來執行控制器輸出、輸入模組操作。呼叫此方法的同時，必須傳入一個字串型態的參數以告知驅動程式欲進行的控制器輸出、輸入模組相關操作指令。傳入的字串參數其格式應為：ON 加上一組二進位數值、OFF 加上一組二進位數值與 GET 三種。ON、OFF 指令是用來指示控制器進行輸出模組節點的導通及斷路動作，所搭配的二進位數值則指示欲導通或斷路的輸出節點位置。

在二進位數值中，由右至左的每一個位元依序對應到輸出模組上從編號零開始的各節點位置。對於 ON、OFF 指令來說，值為 1 的位元代表欲進行狀態變動的節點、值為 0 的位元位置則代表不需變動狀態的節點。例如 ON1011 代表要求控制器導通輸出模組上的零號、一號及三號節點，其餘節點不作任何更動，OFF11001 代表要求控制器斷路輸出模組上的零號、三號及四號節點，其餘節點不作任何更動。ON、OFF 指令規則如圖 3.14 所示。

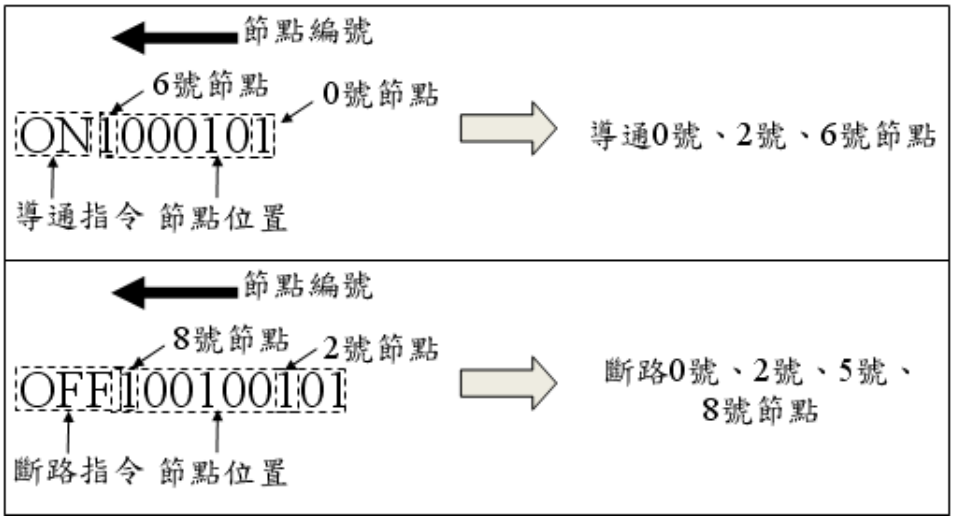


圖 3.14 Handle 方法的 ON、OFF 操作指令說明

GET 指令是用來指示驅動程式向控制器擷取輸入模組狀態。GET 指令發送後，控制器會傳回其輸入模組狀態儲存於驅動程式之中，這時可再使用 getPLCInputState 方法來向驅

動程式取得輸入模組狀態資訊。

此方法執行結束後會回傳一個布林值說明本次控制器指令傳送的狀態。當所傳回的布林值為真時，代表成功傳送指令、布林值為假時代表傳送指令失敗 Handle 方法的作業流程如圖 3.15 所示。

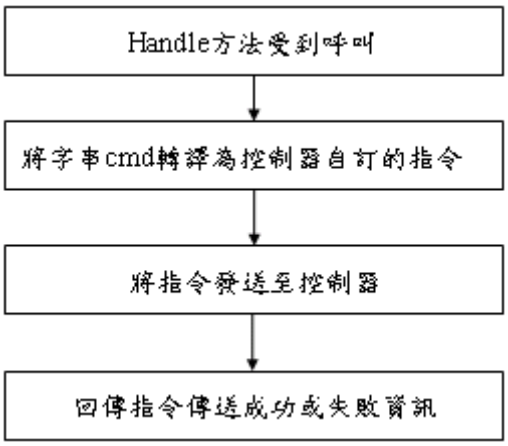


圖 3.15 Handle 方法的作業流程

對於「closePLCConnection」方法而言，這是用來關閉與控制器的連線。此方法內部主要是由各類型控制器自訂的關閉控制器連線語法所組成，其會嘗試與遠端控制器關閉連線，並於此方法執行結束後會回傳一個布林值說明本次結束連線的作業狀態。當所傳回的布林值為真時，代表成功關閉連線、布林值為假時代表關閉連線失敗。closePLCConnection 方法的作業流程如圖 3.16 所示。

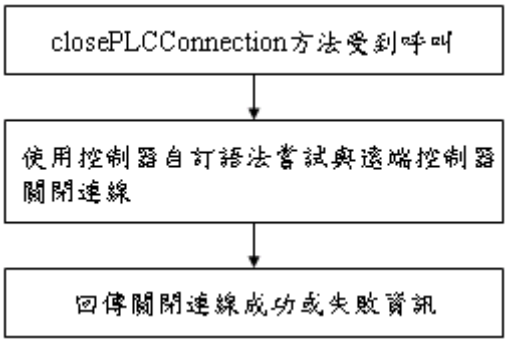


圖 3.16 closePLCConnection 方法的作業流程

第四章 ODSC 資料連接介面的實作

本章說明使用物件導向程式語言 Java[15]以及 PNML 技術實作開放資料伺服器連接架構的過程。4.1 節說明以 PNML 技術為基礎的開放資料伺服器連接架構的開發流程。4.2 節定義 PNML 導向資料伺服器連接軟體的開發流程。4.3 節是 PNML 導向資料伺服器連接軟體的需求分析。4.4 節是 PNML 導向資料伺服器連接軟體的規格開發。

4.1 PNML 技術簡化 ODSC 實作的開發流程

在開放資料伺服器連接架構的開發流程裡，共可以分成(1)裴氏圖定義、(2)裴氏圖同步性設定、(3)資料伺服器軟體開發、(4)驅動程式開發以及(5)決策系統、資料伺服器軟體、驅動程式整合五個步驟。其中，資料伺服器軟體開發、驅動程式開發以及最後的軟體整合三個步驟裡需要撰寫大量程式碼，過程複雜繁瑣。為了能夠加快以及簡化資料伺服器連接架構的開發，這裡是以「裴氏圖加註語言」文件技術為基礎設計一套資料伺服器連接軟體，來輔助建置資料伺服器連接架構。透過此軟體的使用，能夠自動化資料伺服器軟體的撰碼以及軟體整合的撰碼作業，使得原本五個步驟的開發流程減少至僅需三個步驟：(1)裴氏圖定義、(2)裴氏圖同步性設定、(3)驅動程式開發。PNML 導向開放資料伺服器連接架構的開發流程如圖 4.1 所示。

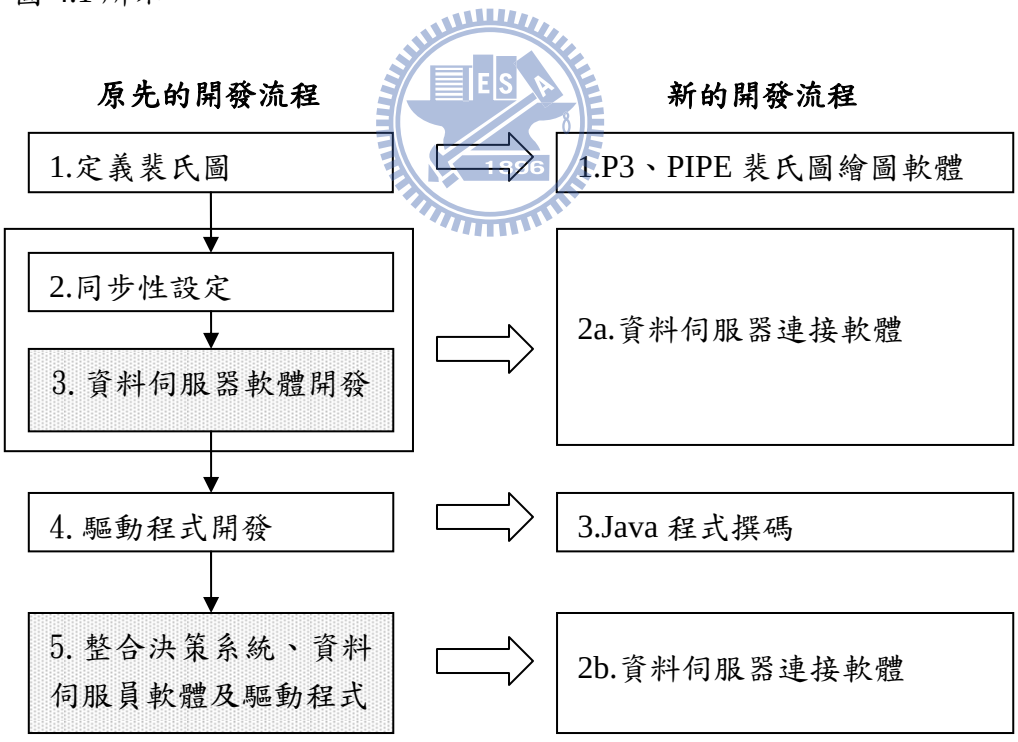


圖 4.1 PNML 導向開放資料伺服器連接架構開發流程

對於「裴氏圖定義」而言，這指的是使用專業的裴氏圖繪製軟體，如 P3、PIPE，設計出自動化系統的裴氏圖，並且以裴氏圖加註語言檔格式保存該模式。

對於「裴氏圖同步化設定」而言，這是指載入所定義出的裴氏圖加註語言文件至資料伺服器連接軟體之中。資料伺服器連接軟體在得到裴氏圖的資訊後，會提供介面給予使用

者進行裴氏圖元件與控制器之間的同步性設定，並且將使用者所設定的同步性配置內容自動撰碼成為資料伺服器軟體。同時，使用者也能夠在此軟體中選擇已撰寫的驅動程式進行決策系統、資料伺服器軟體、驅動程式三者之間的整合作業，整合完成的軟體可以直接用來與控制器連線進行自動化控制。有關 PNML 導向資料伺服器連接軟體的開發流程會在 4.2 節中說明。

對於「驅動程式開發」而言，這是針對尚未擁有驅動程式的控制器類型製作其驅動程式軟體，驅動程式的撰碼規格需依照 3.4 節所述。

4.2 PNML 導向資料伺服器連接軟體開發流程

在 PNML 導向資料伺服器連接軟體的開發流程方面，可以分成(1)軟體需求分析、(2)軟體規格設計、(3)程式撰寫以及(4)軟體驗證四個階段。PNML 導向資料伺服器連接軟體的設計流程如圖 4.2 所示。

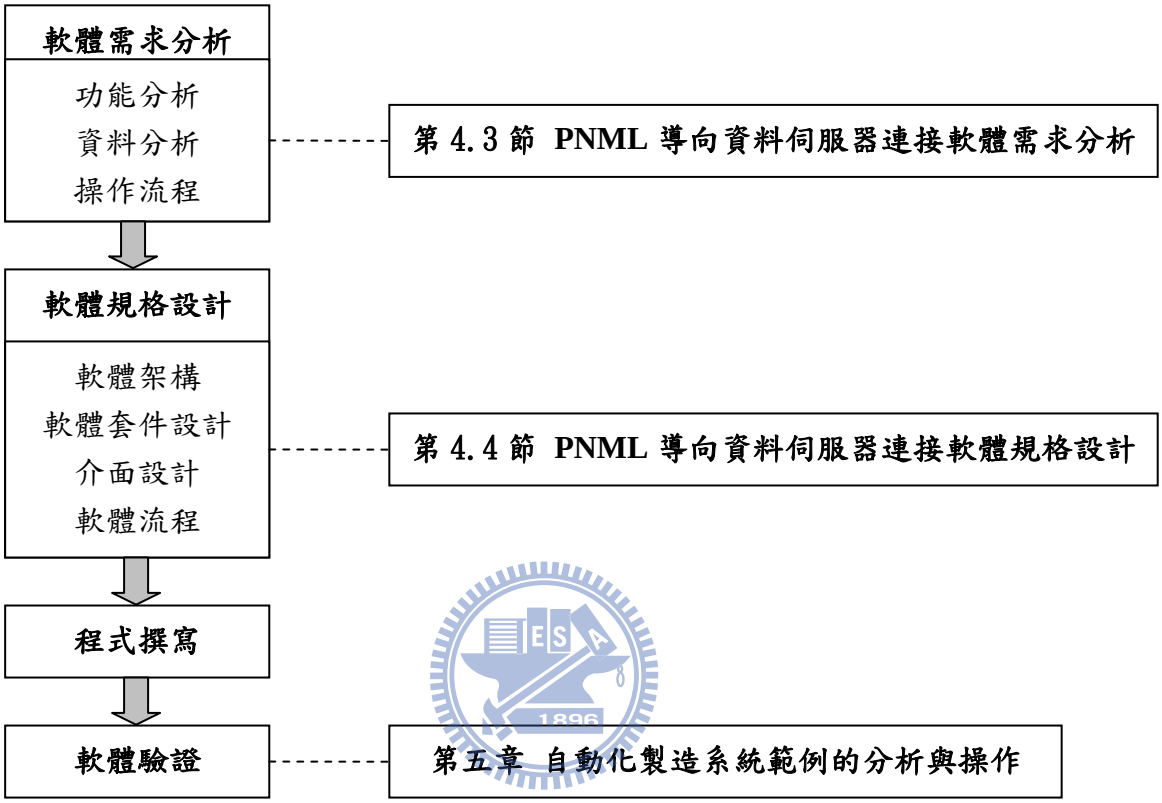


圖 4.2 PNML 導向資料伺服器連接軟體開發流程圖

對於「軟體需求分析階段」而言，這是依照使用者需求規劃軟體所需的開發內容，而其內容包含(1)軟體功能分析、(2)資料分析以及(3)操作流程三個項目。「功能分析」是要針對軟體欲達成的目標定義出軟體應提供的功能與服務。「資料分析」在於界定軟體的輸入與輸出，亦即軟體在使用時所需載入與匯出的資料內容與格式。「操作流程」是說明使用者對於軟體的操作。有關 PNML 導向資料伺服器連接軟體的需求分析部份於 4.3 節中說明。

對於「軟體規格設計階段」而言，這是指將軟體需求轉換成電腦中的邏輯處理方式，而其內容包含(1)軟體架構、(2)軟體套件設計、(3)介面設計與(4)軟體流程四個項目。軟體架構是說明軟體的組成套件以及各套件的部署方式。軟體套件設計是進一步描述各個軟體套件裡類別的屬性與方法定義。介面設計是規劃軟體與使用者之間的互動方式。軟體流程是說明軟體的運作模式。有關 PNML 導向資料伺服器連接軟體的規格設計部份於 4.4 節中說明。

對於「程式撰寫階段」而言，這是指將軟體規格利用程式語言撰寫實作出來，這裡所使用的程式語言是 Java 程式語言。

對於「軟體驗證階段」而言，這是以一個灌模系統為案例進行開放資料伺服器連接架構的實作，以驗證軟體的可用性及正確性。有關軟體驗證部份於第五章中說明。

4.3 PNML 導向資料伺服器連接軟體需求分析

本節共分成三小節。首先，4.3.1 節進行軟體功能分析。接著，4.3.2 節進行軟體資料分析。最後，4.3.3 節說明軟體操作流程。

4.3.1 軟體功能分析

PNML 導向資料伺服器連接軟體的開發目的是要輔助使用者建置開放資料伺服器連接架構，這包含(1)提供使用者便利的裴氏圖同步性的定義方式、(2)資料伺服器軟體的自動化撰碼功能以及(3)決策系統、資料伺服器軟體、驅動程式的自動化整合功能。

對於「裴氏圖同步性定義方式」而言，這是建構圖形化介面讓使用者只需透過圖形點選方式就能完成與控制器之間的同步性設定。同步性設定方法是遵照修正詮釋性裴氏圖的規則作設計，也就是說每個暫存點上可以設定一組輸入節點狀態或是位於特定值域的虛擬變數用來代表元件的狀態。每個轉移點上可以設定一組輸出節點的狀態或是虛擬變數的值用來代表該轉移點所對應到的一項作業、一組輸入節點狀態或是位於特定值域的虛擬變數用來代表該作業的觸發事件以及一組轉移點激發的延遲時間。

對於「資料伺服器軟體自動化撰碼功能」而言，這是要將使用者所配置出的同步性設定自動轉換成相對應的 Java 程式語言，並產生一個符合標準規格的資料伺服器軟體，以節省使用者所需花費的撰碼時間。

對於「自動化軟體整合」而言，這是讓使用者能夠在已開發的控制器驅動程式中選擇與設定，並且所選的驅動程式會自動與資料伺服器軟體以及軟體中提供的一個簡易的決策系統進行整合。這個整合完成的軟體能夠直接與控制器作連接，進行自動化控制作業。

如此一來，只要藉由以上軟體功能的操作，使用者無須具備相當程度的程式撰寫能力，也可以自行建置出開放資料伺服器連接架構。並且在不同控制器類型的執行環境下，使用者只需要更換匹配的驅動程式，就能夠迅速移轉其控制程式至其他的作業環境中。

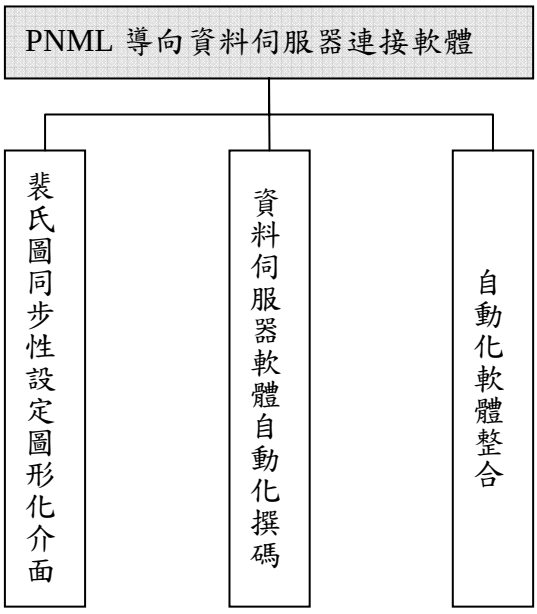


圖 4.3 PNML 導向資料伺服器連接軟體功能

4.3.2 軟體資料分析

在資料輸入方面，軟體接受的資料形式為裴氏圖加註語言檔。有關裴氏圖加註語言檔的製作，這可以藉由裴氏圖繪圖軟體如 P3、PIPE 產生。在讀取裴氏圖加註語言文件中的資訊之後，軟體會擷取出使用者所定義的裴氏圖結構，並且將裴氏圖結構中的各項元件與使用者所設定的同步性資訊進行結合，以產生相對應的資料伺服器軟體。PNML 資料伺服器連接軟體的 IDEF0 規格圖如圖 4.4 所示。

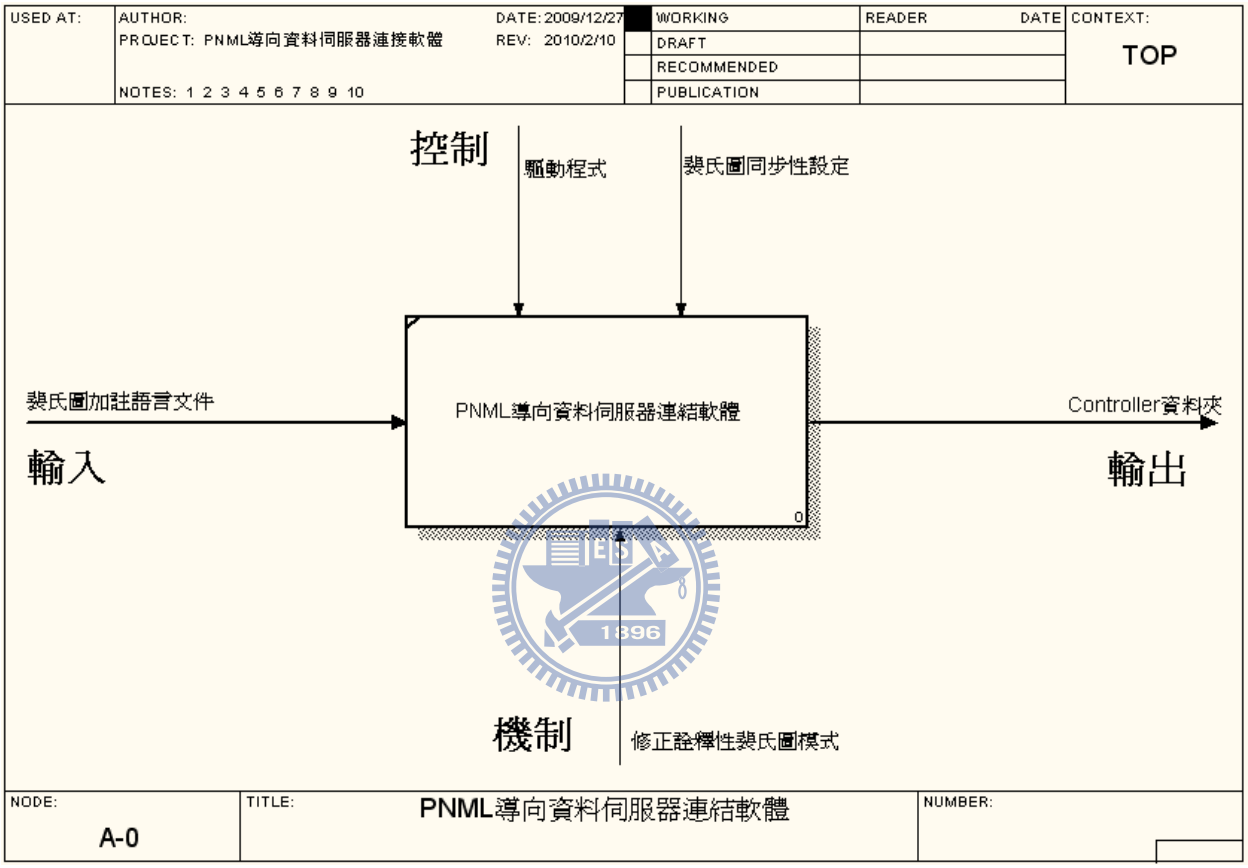


圖 4.4 PNML 資料伺服器連接軟體的 IDEF0 規格圖

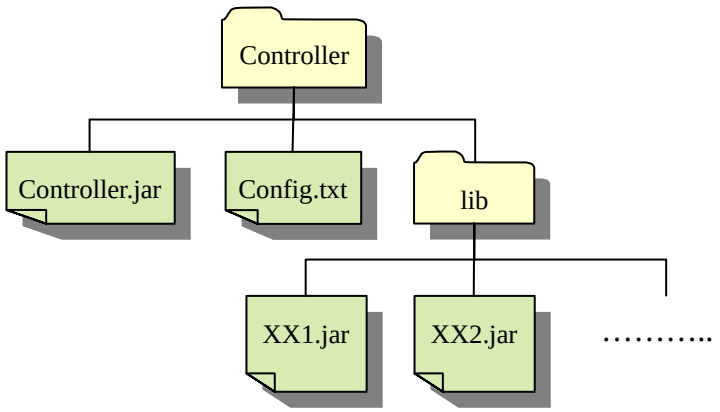


圖 4.5 Controller 資料夾架構

在資料輸出方面，軟體會匯出一個 Controller 資料夾，Controller 資料夾底下包含有兩

個檔案與一個資料夾，分別是(1)控制程式的主執行檔 Controller.jar、(2)控制程式的設定檔 Config.txt 檔以及(3)存放驅動程式所需 API 檔案的資料夾 lib。其中，「Conroller.jar」檔是整合了決策系統、資料伺服器軟體以及驅動程式以後所產生的控制程式執行檔。它主要是利用 Java 程式語言所提供的檔案封裝功能，將多個程式執行時所需要的已編譯類別檔壓縮成單一個可執行檔。這種包裝方式除了能夠省去過多檔案所造成的檔案管理困難之外，也可以提升軟體的安全性，這指的是避免直接開放程式原始碼而可能造成程式碼受到修改，導致程式執行時發生錯誤。「Config.txt」檔是用來保存所控制程式安裝的相關資訊，如控制器的網路位置。「Lib」資料夾則是用來放置驅動程式需要使用到的 API 檔案。Controller 資料夾內的檔案架構如圖 4.5 所示。

4.3.3 軟體操作流程

在軟體執行時，使用者所需要進行的操作包含四項：(1)選擇載入的 PNML 檔案位置、(2)設定裴氏圖同步性、(3)選擇驅動程式與(4)選擇匯出的檔案位置，如圖 4.6 所示。首先在程式一開始時，使用者需要將附檔名為.pnml 或是.xml 的 PNML 文件載入至軟體中。當 PNML 檔正確的載入至軟體後，使用者可以開始透過圖形化介面設定每一個暫存點、轉移點與控制器輸入、輸出模組之間的連接方式。在所有的暫存點與轉移點的同步性都設定完成後，接著是選擇所要搭配的驅動程式類型以及設定與驅動程式連接的控制器的網路位置。最後，使用者必須選擇匯出檔案的存放位置，軟體會在該位置上建立一個 Controller 資料夾並且將決策系統、資料伺服器軟體、驅動程式整合完成後的控制程式檔案放置於資料夾內。

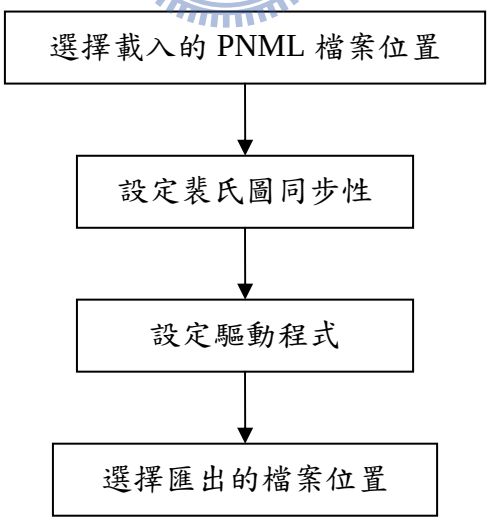


圖 4.6 PNML 導向資料伺服器連接軟體操作流程

4.4 PNML 導向資料伺服器連接軟體規格設計

本節共分成四小節。4.4.1 節說明軟體架構。4.4.2 節說明軟體套件的設計。4.4.3 節說明操作介面的設計。4.4.4 節說明軟體的執行流程。

4.4.1 軟體架構

對於軟體環境建置而言，PNML 導向資料伺服器連接軟體的內部程式依功能性質的不同可以區分成六個套件：(1)GUI 套件、(2)PetriNet 套件、(3)XMLTools 套件、(4)FileGenerator 套件、(5)Controller 套件以及(6)Main 套件。各套件功能說明如表 4.1。有關 PNML 導向資料伺服器連接軟體的套件部署方式如圖 4.7、4.8 所示。

表 4.1 PNML 導向資料伺服器連接軟體套件功能說明

套件名稱	套件功能
GUI	建置使用者圖形化介面。
PetriNet	保存、分析裴氏圖資訊。
XMLTools	解析 PNML 文件的資料。
FileGenerator	輸出檔案。
Controller	建立決策系統
MainFile	軟體的主執行程式套件。

對於「GUI 套件」而言，這主要是用來建置軟體的使用者圖形化介面，它負責管理軟體的畫面呈現以及處理與使用者之間的互動。當使用者對圖形化介面中的元件進行任何動作時，如滑鼠點擊、遊標移動、文字輸入等，GUI 套件會將這些動作視為一次事件的觸發，並且針對所觸發的事件呼叫預先設定的功能進入作業程式，作業程式完畢後可以再將結果反應至圖形化介面中。

對於「PetriNet 套件」而言，其包含各種裴氏圖元件，如轉移點、暫存點、箭號，這是用來保存與分析使用者定義的裴氏圖資訊。在 XMLTools 套件讀取出 PNML 檔案內的裴氏圖資訊之後，就會使用 PetriNet 套件建立一個裴氏圖物件用來保存其中所需的資訊，如元件的名稱、座標、暫存點的初始浮標狀態、箭號的輸入及輸出節點。物件形式的資料會在程式執行階段暫存於電腦記憶體中，並且可以在各個功能之間傳遞使用。此外，使用者所設定的同步性內容也會儲存在裴氏圖物件中。

對於「XMLTools 套件」而言，這主要是用來解析 PNML 檔的資料。當使用者載入 PNML 檔至軟體後，軟體首先會呼叫 XMLTools 套件擷取出保存於 PNML 檔內的裴氏圖資訊，所讀取出的裴氏圖資訊會以物件形式暫存於軟體中，並傳送給其他套件使用。

對於「FileGenerator 套件」而言，這主要是用來執行檔案輸出功能。當接收到使用者的檔案匯出指令後，軟體會呼叫 FileGenerator 套件將使用者所配置的同步性設定轉換成資料伺服器軟體，並且進行決策系統、資料伺服器軟體以及使用者所選的驅動程式的整合作業。在整合作業完畢後，會匯出資料夾至目標目錄底下。

對於「Controller 套件」而言，這是一個簡易的決策系統軟體，其包含了監控階段所需的監控員軟體、偵查員軟體、維修員軟體與操作介面。當 FileGenerator 套件進行整合作業時，會將所產生的資料伺服器軟體與驅動程式檔案放置到 Controller 套件內，再包裝 Controller 套件成一個控制程式執行檔，使用者就能執行該執行檔進行所定義的控制行為。這也表示在 PNML 導向資料伺服器連接軟體的運作過程中，並不會實際使用此套件的任何功能，Controller 套件本身可視為是另一個獨立的 Java 應用程式架構。

對於「MainFile 套件」而言，這是軟體的主執行套件。在 MainFile 套件裡的類別宣告了方法 Main，這是 Java 程式語言所規定的應用程式結構，代表程式的起始位置。

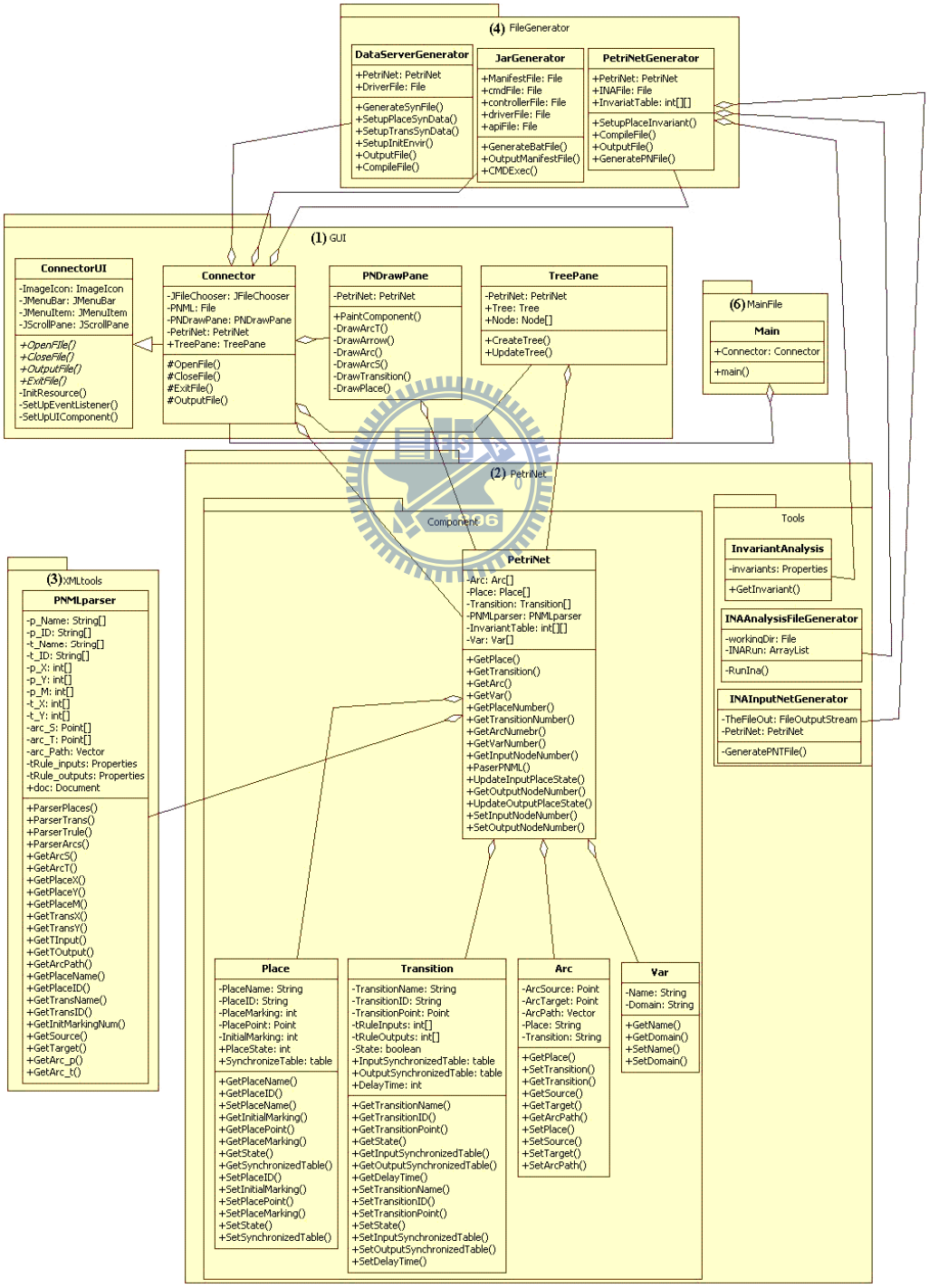


圖 4.7 以類別圖表達 GUI、PetriNet、XMLTools、FileGenerator、Main 套件部署情形

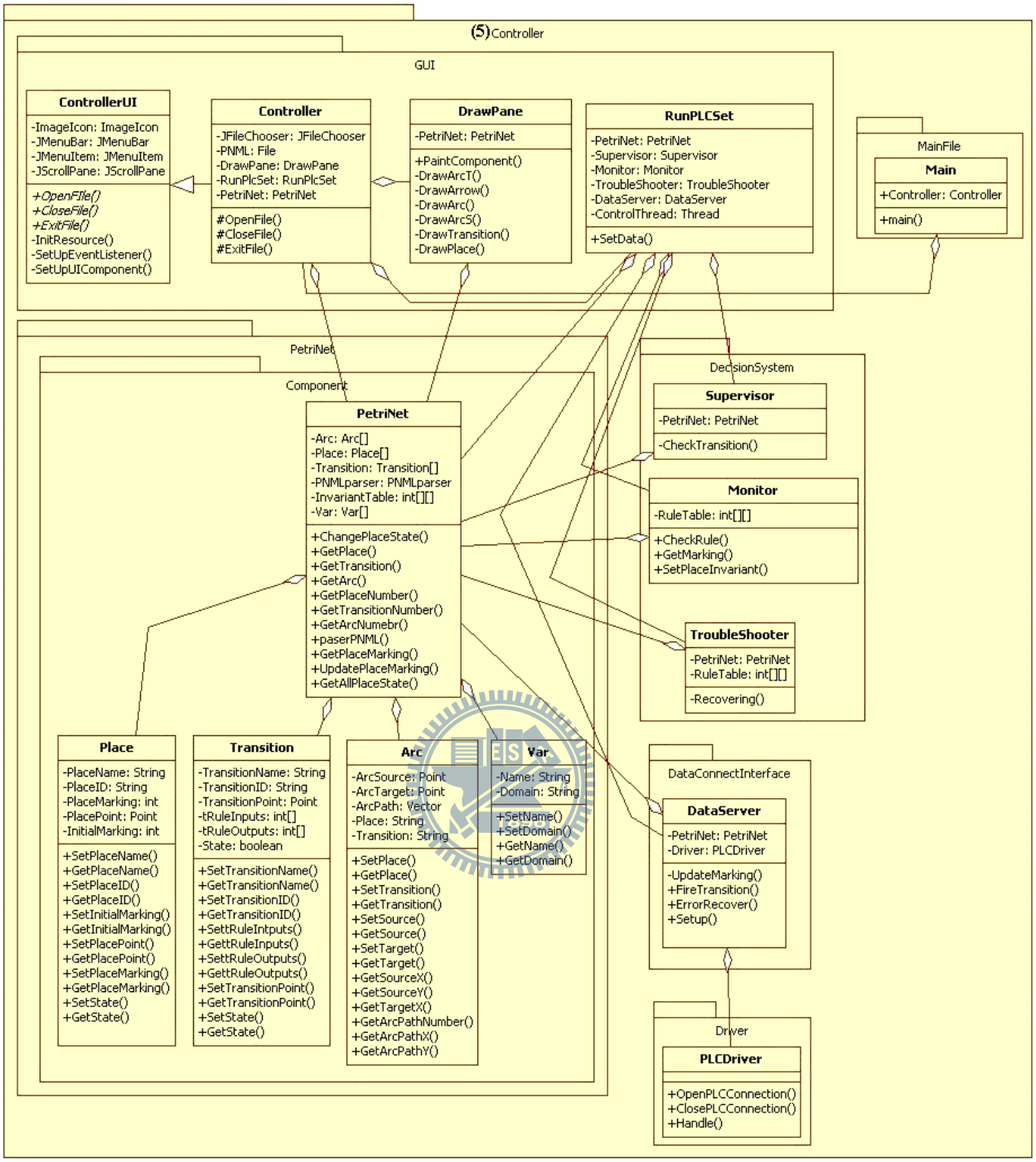


圖 4.8 以類別圖表達 Controller 套件部署情形

4.4.2 軟體套件設計

本節主要是針對 PetriNet 套件、XMLTools 套件、FileGenerator 套件、Controller 套件的設計方式進行說明，有關 GUI 套件部分會另外在 4.4.3 節「操作介面設計」中詳述。而 MainFile 套件僅包含 Java 程式語言的基本應用程式啟動指令，這裡則不再多加闡述。4.4.2.1 節介紹 PetriNet 套件的設計。4.4.2.2 節介紹 XMLTools 套件的設計。4.4.2.3 節介紹 FileGenerator 套件的設計。4.4.2.4 節介紹 Controller 套件的設計。

4.4.2.1 PetriNet 套件的設計

在 PetriNet 套件中包含兩個子套件「Component」與「Tools」。PetriNet 套件的內部結構如圖 4.9 所示。

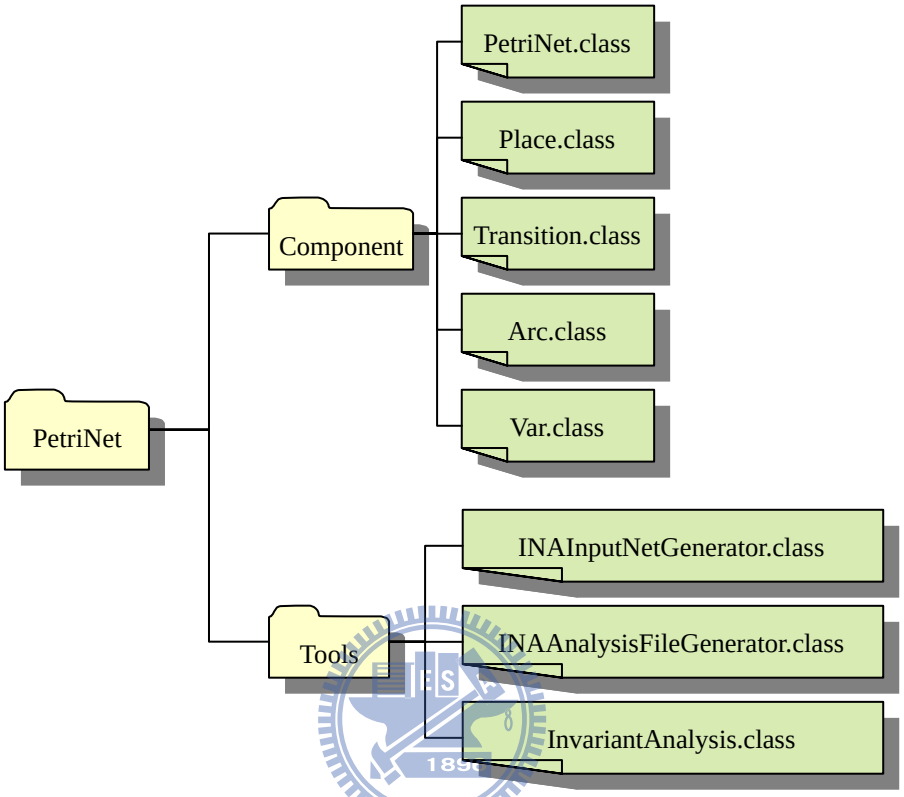


圖 4.9 PetriNet 套件架構

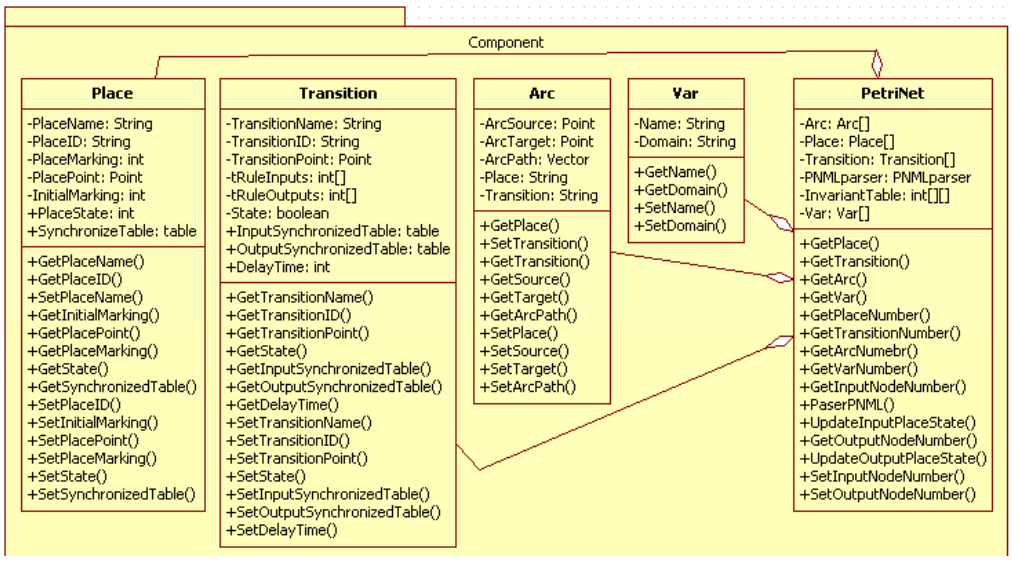


圖 4.10 Component 套件類別圖

「Component 套件」主要用來保存裴氏圖資訊以及紀錄使用者所設定的同步性內容，套件內共包含五個類別 PetriNet、Place、Transition、Arc、Var 分別代表裴氏圖、裴氏圖暫

存點、轉移點與箭號元件以及同步性設定時使用到的虛擬變數。針對各個類別所包含的屬性，如元件名稱、座標、浮標數、可標記狀態，類別裡都設有相關的屬性 set 方法與 get 方法可用來更新與讀取屬性值。Component 套件類別圖如圖 4.10 所示。Component 套件各類別方法說明如表 4.1 所示。

表 4.1 Component 套件各類別方法說明

類別	方法	功能	傳入值	回傳值
PetriNet	GetPlace	取得暫存點物件	暫存點編號	Place 物件
	GetTransition	取得轉移點物件	轉移點編號	Transition 物件
	GetArc	取得箭號資訊	箭號編號	Arc 物件
	GetVar	取得變數資訊	無	虛擬變數表
	GetPlaceNumber	取得暫存點數量	無	暫存點數量
	GetTransitionNumber	取得轉移點數量	無	轉移點數量
	GetArcNumber	取得箭號數量	無	箭號數量
	GetVarNumber	取得變數數量	無	變數數量
	GetInputNodeNumber	取得輸入節點數量	無	輸入節點數量
	GetOutputNodeNumber	取得輸出節點數量	無	輸出節點數量
	ParserPNML	解析 PNML 檔	檔案名稱	無
	SetInputNodeNumber	設定輸入節點數量	輸入節點數量	無
	SetOutputNoderNumber	設定輸出解點數量	輸出節點數量	無
	SetVar	設定變數資訊	虛擬變數表	無
	UpdateInputPlaceState	更新輸入暫存點為不可標記狀態	轉移點編號	無
	UpdateOutputPlaceState	更新輸出暫存點為可標記狀態	轉移點編號	無
Place	GetPlaceName	取得暫存點名稱	無	暫存點名稱
	GetPlaceID	取得暫存點 ID	無	暫存點 ID
	GetInitialMarking	取得初始浮標數	無	初始浮標數
	GetPlacePoint	取得暫存點座標	無	暫存點座標
	GetPlaceMarking	取得暫存點浮標數	無	暫存點浮標數
	GetState	取得暫存點狀態	無	暫存點狀態
	GetSynchronizedTable	取得同步性設定表	無	同步性設定表
	SetPlaceName	設定暫存點名稱	暫存點名稱	無
	SetPlaceID	設定暫存點 ID	暫存點 ID	無
	SetInitialMarking	設定初始浮標數	初始浮標數	無
	SetPlacePoint	設定暫存點座標	暫存點座標	無
	SetPlaceMarking	設定暫存點浮標數	暫存點浮標數	無
	SetSynchronizedTable	設定同步性設定表	同步性設定表	無
	SetState	設定暫存點狀態	暫存點狀態	無

表 4.1 Component 套件各類別方法說明(續)

Transition	GetTransitionName	取得轉移點名稱	無	轉移點名稱
	GetTransitionID	取得轉移點 ID	無	轉移點 ID
	GetTransitionPoint	取得轉移點座標	無	轉移點座標
	GetState	取得轉移點狀態	無	轉移點是否激發
	GetInputSynchronizedTable	取得激發事件同步性設定表	無	激發事件同步性設定表
	GetOutputSynchronizedTable	取得元件作業同步性設定表	無	元件作業同步性設定表
	GetTInputs	取得輸入暫存點	無	輸入暫存點矩陣
	GetTOutputs	取得輸出暫存點	無	輸出暫存點矩陣
	GetDelayTime	取得延遲啟動時間	無	延遲時間
	SetTransitionName	設定轉移點名稱	暫存點名稱	無
	SetTransitionID	設定轉移點 ID	暫存點 ID	無
	SetTransitionPoint	設定轉移點座標	暫存點座標	無
	SetTInputs	設定輸入暫存點	輸入暫存點矩陣	無
	SetTOuptuts	設定輸出暫存點	輸出暫存點矩陣	無
	SetSynchronizedTable	設定同步性設定表	同步性設定表	無
	SetInputSynchronizedTable	設定激發事件同步性設定表	激發事件同步性設定表	無
	SetOutputSynchronizedTable	設定元件作業同步性設定表	元件作業同步性設定表	無
	SetState	設定轉移點狀態	轉移點是否為激發狀態	無
	SetDelayTime	設定延遲啟動時間	延遲啟動時間	無
Arc	GetPlace	取得暫存點物件	無	暫存點物件
	GetTransition	取得轉移點物件	無	轉移點物件
	GetSource	取得來源座標	無	來源座標
	GetTarget	取得目標座標	無	目標座標
	GetArcPath	取得轉折點座標	無	轉折點座標
	SetArcPath	設定轉折點座標	轉折點座標	無
	SetPlace	設定暫存點	暫存點物件	無
	SetTransition	設定轉移點	轉移點物件	無
	SetSource	設定來源座標	來源座標	無
	SetTarget	設定目標座標	目標座標	無
Var	GetName	取得變數名稱	無	變數名稱
	GetDomain	取得變數資料型別	無	變數資料型別
	SetName	設定變數名稱	變數名稱	無
	SetDomain	設定變數資料型別	變數資料型別	無

「Tools 套件」的功能是呼叫出裴氏圖分析軟體 INA 來輔助分析裴氏圖的暫存點不變量性質，這是為了要產生決策系統中的偵查員軟體所使用。Tools 套件裡共包含了三個類別 INAInputNetGenerator、INAAnalysisFileGenerator、InvariantAnalysis。對於暫存點不變量分析作業的執行步驟而言，首先必須藉由 INAInputNetGenerator 類別的 writeNet 方法建立一個符合 INA 軟體輸入資料格式的檔案，接著是使用 INAAnalysisFileGenerator 類別的 runINA 方法來啟動 INA 軟體進行不變量分析，最後再利用 InvariantAnalysis 類別的 getInvariant 方法解析 INA 軟體所產生的輸出檔案，並取得暫存點不變量性質。Tools 套件類別圖如圖 4.11。Tools 套件的各類別所含方法說明如表 4.2 所示。

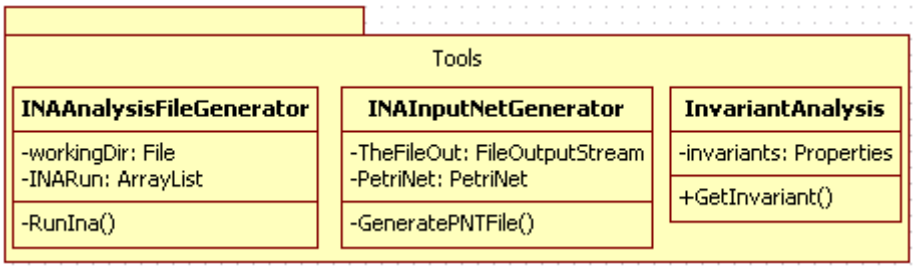


圖 4.11 Tools 套件類別圖

表 4.2 Tools 套件各類別方法說明

類別	方法	功能	傳入值	回傳值
InvariantAnalysis	GetInvariant	取得暫存點不變量資訊	暫存點不變量檔案存放位置	暫存點不辨量資訊
INAAanlysisFileGenerator	RunIna	執行 INA 軟體分析裴氏圖	無	是否成功執行 INA 軟體
INAInputNetGenerator	GeneratePNTFile	產生 INA 軟體所需的輸入資料檔案	裴氏圖物件、檔案存放位置	無

4.4.2.2 XMLTools 套件的設計

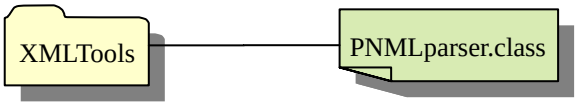


圖 4.12 XMLTools 套件架構

在 XMLTools 套件中包含一個類別「PNMLparser」，該類別是運用 dom4j API[18]的可加註語言文件編輯技術所開發出的裴氏圖加註語言文件解析程式。當使用者載入裴氏圖加註語言文件至軟體後，軟體會呼叫 PNMLpaser 類別的 parseFile 方法來擷取所需的裴氏圖資訊，這包含暫存點與轉移點的 ID、名稱、座標、暫存點初始浮標數、箭號的輸入、輸出節點名稱與座標，而擷取出的資訊會再儲存至裴氏圖物件內。XMLTools 套件的內部架構

如圖 4.12 所示。PNMLpaser 類別的 parseFile 方法作業流程如圖 4.13 所示。XMLTools 套件的類別圖如圖 4.14 所示。PNMLparser 類別所含方法說明如表 4.3 所示

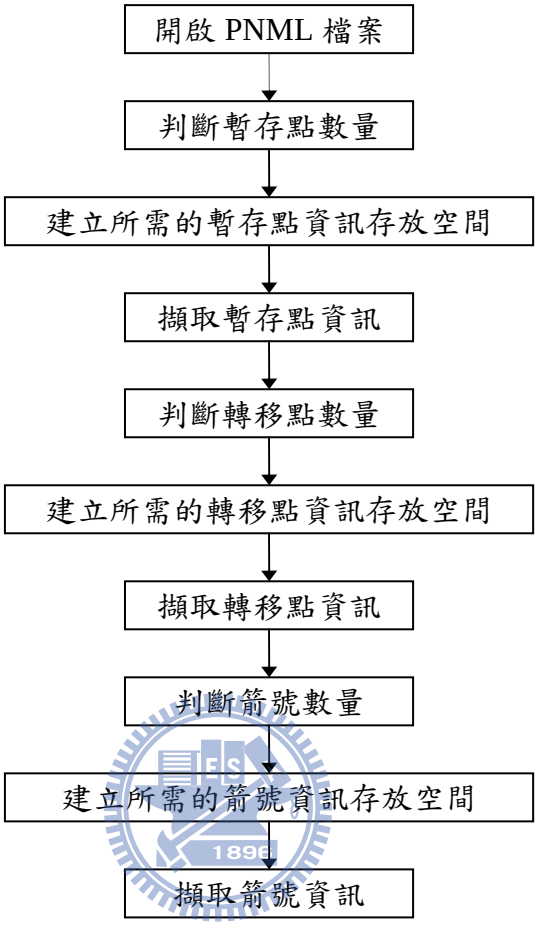


圖 4.13 PNMLpaser 類別的 parseFile 方法作業流程圖

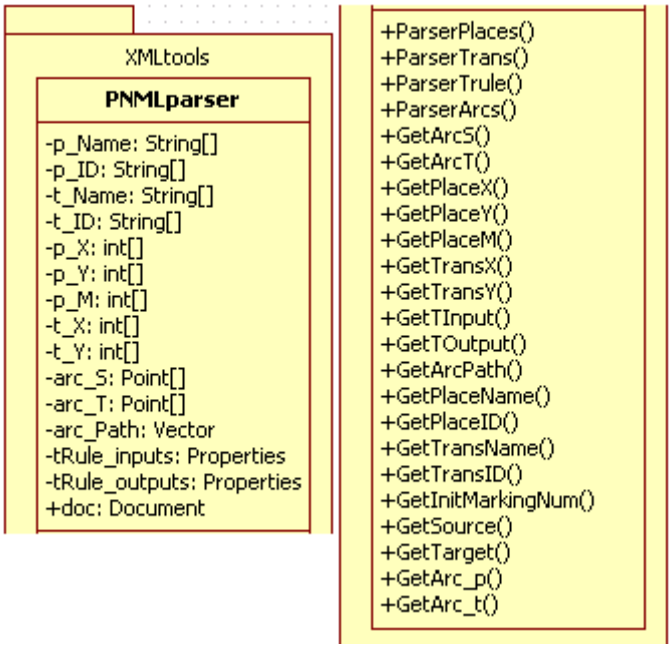


圖 4.14 XMLTools 套件類別圖

表 4.3 PNMLparser 套件各類別方法說明

類別	方法	功能	傳入值	回傳值
PNMLparser	ParserPlaces	擷取 PNML 文件暫存點資訊	無	無
	ParserTrans	擷取 PNML 文件轉移點資訊	無	無
	ParserArcs	擷取 PNML 文件箭號資訊	無	無
	GetArcS	取得箭號來源座標	無	箭號來源座標矩陣
	GetArcT	取得箭號目標座標	無	箭號目標座標矩陣
	GetPlacePoint	取得暫存點座標	無	暫存點座標矩陣
	GetTransPoint	取得轉移點座標	無	轉移點座標矩陣
	GetArcPath	取得箭號轉折點	無	箭號轉折點矩陣
	GetPlaceName	取得暫存點名稱	無	暫存點名稱矩陣
	GetPlaceID	取得暫存點 ID	無	暫存點 ID 矩陣
	GetTransName	取得轉移點名稱	無	轉移點名稱矩陣
	GetTransID	取得轉移點 ID	無	轉移點 ID 矩陣
	GetInitMarkingNum	取得初始浮標數	無	初始浮標數矩陣
	GetSource	取得箭號來源名稱	無	箭號來源名稱矩陣
	GetTarget	取得箭號目標名稱	無	箭號目標名稱矩陣
	GetArc_p	取得箭號暫存點名稱	無	箭號暫存點名稱矩陣
	GetArc_t	取得箭號轉移點名稱	無	箭號轉移點名稱矩陣

4.4.2.3 FileGenerator 套件的設計

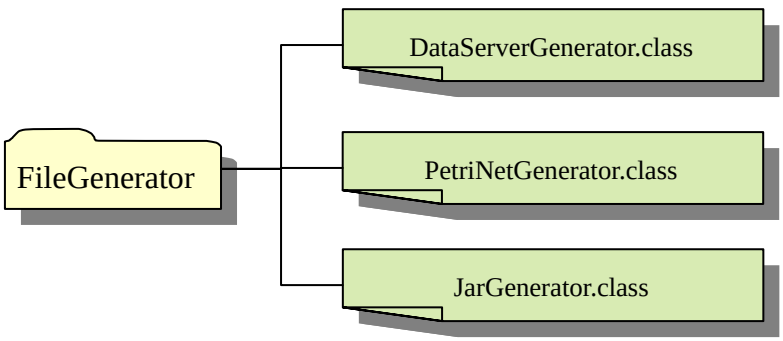


圖 4.15 FileGenerator 套件架構

在 FileGenerator 套件中共包含三個類別：(1)DataServerGenerator、(2)PetriNetGenerator 與(3)JarGenerator。FileGenerator 套件架構如圖 4.15 所示。

對於「DataServerGenerator 類別」而言，這是用來整理使用者所設定的同步性資料並轉換成相對應的資料伺服器軟體，轉換過程可以分成四個階段。首先，第一階段是進行資料伺服器軟體的 updateMarking 方法自動化撰碼。updateMarking 方法的目的是要按照暫存點同步性所述，將控制器輸入模組以及虛擬變數的狀態轉換成裴氏圖物件裡各個暫存點的浮標狀態。在 updateMarking 方法的內容撰寫方面，這裡會將使用者對於每一個暫存點所

設定的同步性個別寫成一組 if….then 形式的條件判斷式，當該暫存點所設同步性條件成立並且暫存點為可標記狀態時即設定此暫存點擁有浮標，反之則無。updateMarking 方法的自動化撰寫模式如圖 4.16 所示。

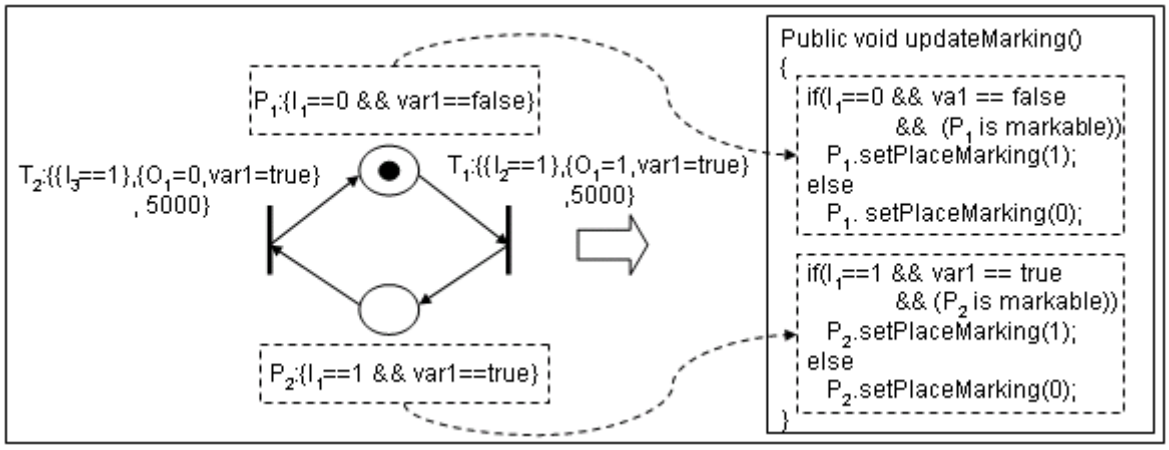


圖 4.16 DataServerGenerator 類別的 updateMarking 方法自動化撰碼模式

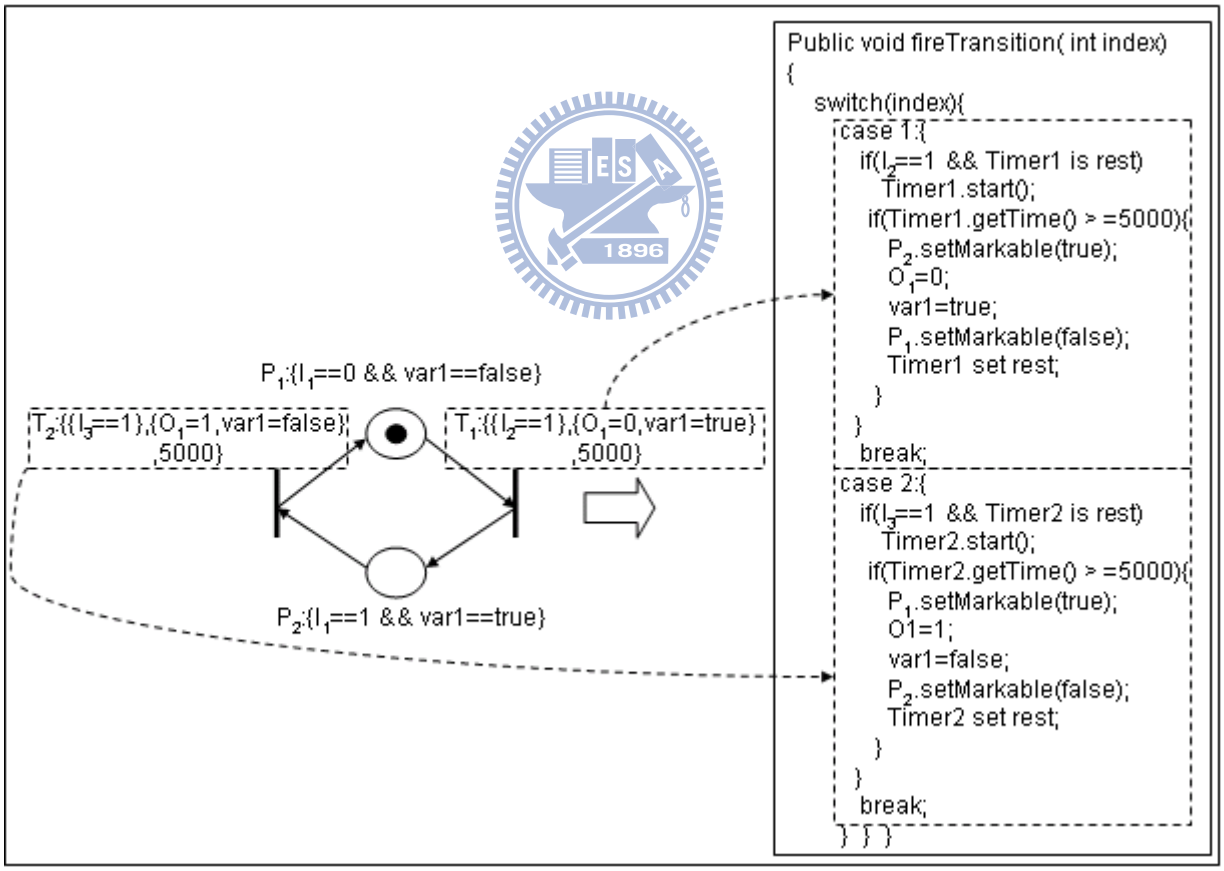


圖 4.17 DataServerGenerator 類別的 fireTransition 方法自動化撰碼模式

轉換過程的第二個階段是進行資料伺服器軟體的 fireTransition 方法的自動化撰碼。fireTransition 方法的目的是要按照轉移點同步性的敘述，將決策系統所下達的轉移點激發指令轉換成相對應的控制器的輸出行為或是更新虛擬變數的值。在 fireTransition 方法的內容撰寫方面，這裡以 Switch…..case 的語法結構為基礎去設定每一個轉移點的監控法則內

層結構為一個不同編號的 case。當 fireTransition 受到呼叫時，Switch…..case 語法會切換到編號與 index 相同的轉移點監控法則進行激發作業。fireTransition 方法的自動化撰寫模式如圖 4.17 所示。

轉換過程的第三個階段是進行資料伺服器軟體 setup 方法的自動化撰碼。setup 方法的目的是要在資料伺服器軟體啟動前，將虛擬變數還原成初始值，以避免因為虛擬變數值不正確的緣故，而造成資料伺服器軟體無法取得正確的初始浮標狀態。在 setup 方法的內容撰寫方面，這是按照初始狀態下具有浮標的暫存點的同步性設定來初始化各個虛擬變數。setup 方法的內容撰寫方式如圖 4.18 所示。

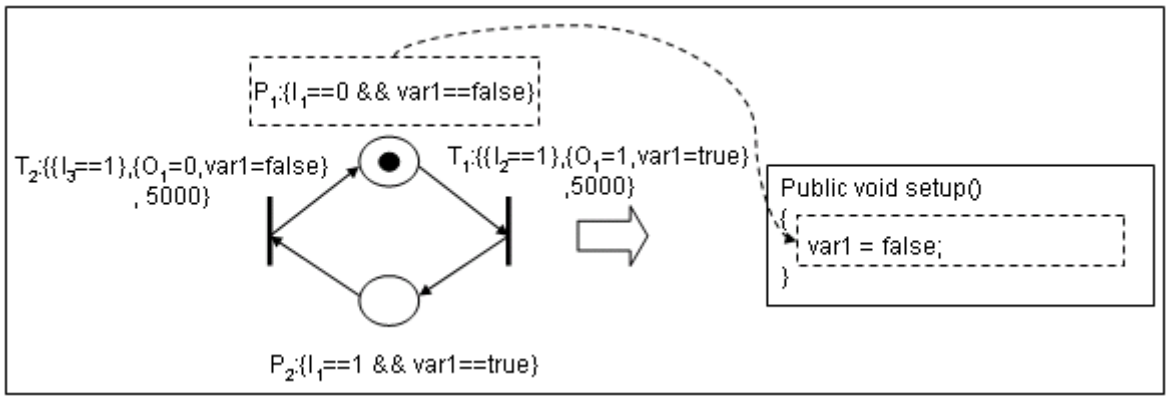


圖 4.18 DataServerGenerator 類別的 setup 方法自動化撰碼模式

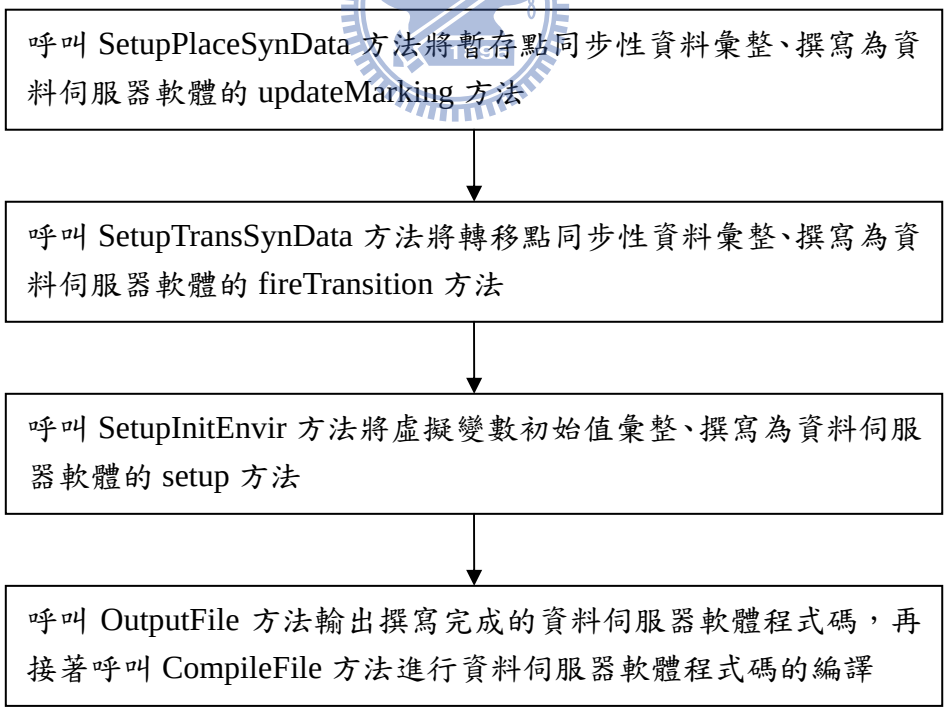


圖 4.19 DataServerGenerator 類別作業流程

最後，轉換過程的第四個階段是將 updateMarking、fireTransition、setup 方法的程式碼與資料伺服器軟體其餘較制式的程式碼，如套件載入、類別名稱、建構子等程式內容合併以後，匯出成一個 DataServer.java 檔。並透過程式碼編譯(Compile)作業獲得所需的

DataServer.class 檔。DataServerGenerator 類別的作業流程如圖 4.19 所示。編譯完成的 DataServer 類別圖如圖 4.20(a)所示。

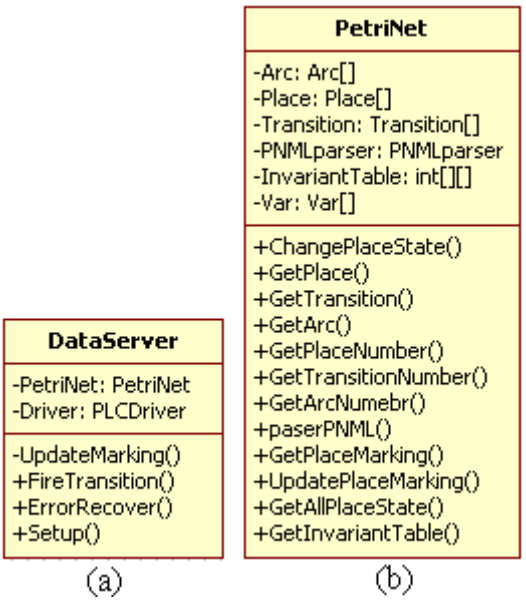


圖 4.20 FileGenerator 套件產生的資料伺服器軟體與 PetriNet 類別

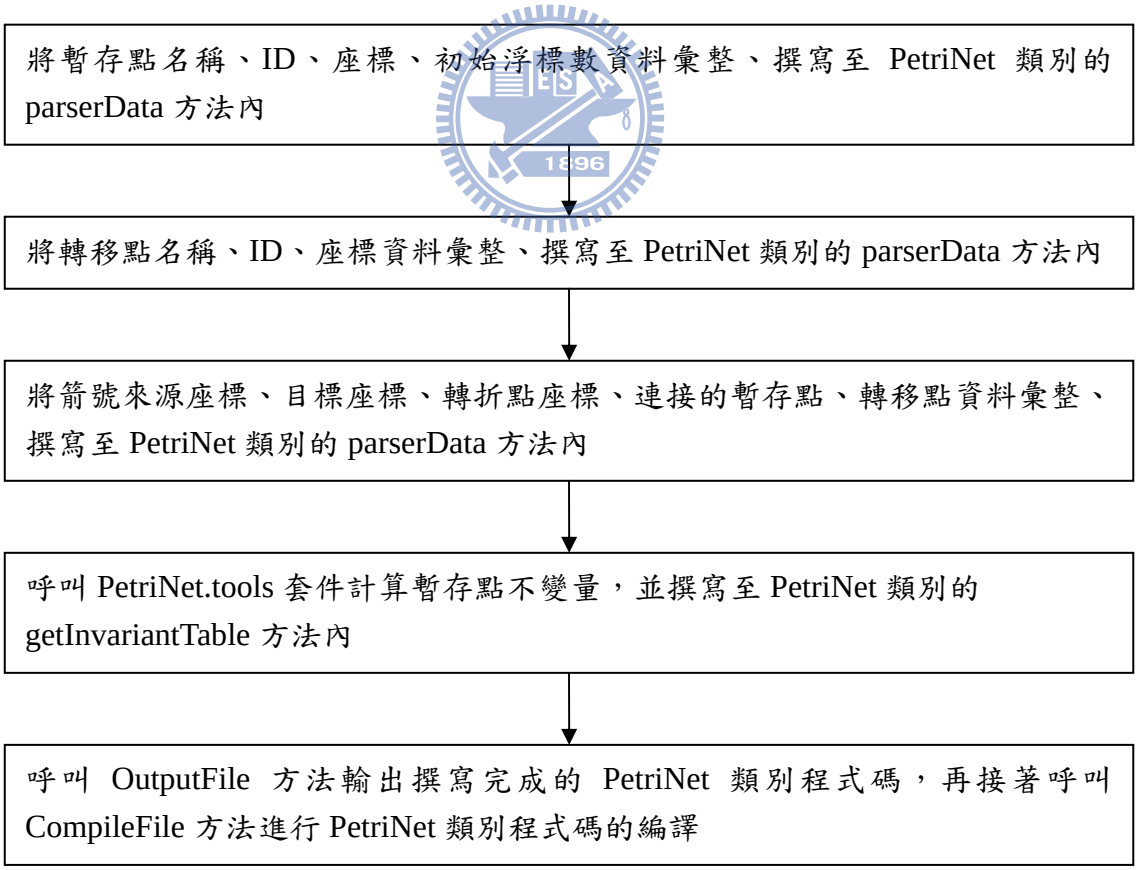


圖 4.21 PetriNetGenerator 類別的作業流程

對於「PetriNetGenerator 類別」而言，這是用來將裴氏圖物件所記錄的暫存點、轉移點、箭號資訊匯出成一個 PetriNet.java 檔，讓控制軟體啟動時，不需要另外載入 PNML 檔

就可以從 PetriNet 類別裡得到相同的裴氏圖資訊。此外，PetriNetGenerator 類別在資料匯出過程中會利用 PetriNet.Tools 套件所提供的暫存點不變量分析功能計算系統暫存點不變量，並且一併將此資訊保存在 PetriNet 類別中。PetriNetGenerator 類別的作業流程如圖 4.21 所示。由 PetriNetGenerator 類別所產生的 PetriNet 類別與 PNML 導向資料伺服器連接軟體本身的 PetriNet 類別不同之處在於其多了一個 getInvariantTable 方法可用來取得暫存點不變量資訊，如圖 4.20(b)所示。

對於「JarGenerator 類別」而言，這是用來整合決策系統、資料伺服器軟體、驅動程式，並且將它們壓縮、匯出成一個 Controller.jar 執行檔。在 JarGenerator 類別的作業流程方面共分成二個步驟。步驟一是要產生出一個 manifest.mf 檔案，這個檔案是在使用 Java 程式語言的內建程式碼包裝指令 Jar 時，必須附帶的一個環境設定檔。該檔案裡說明瞭方法 main 的所在位置以及控制器驅動程式所需使用的 API 檔案名稱，這會指示軟體在啟動時載入這些 API 檔案。所以當 JarGenerator 類別受到呼叫時，首先會依據使用者所選擇的驅動程式，將該驅動程式必須搭備使用的 API 檔案名稱輸入至 manifest.mf 中。manifest.mf 檔案內容格式如圖 4.22 所示。

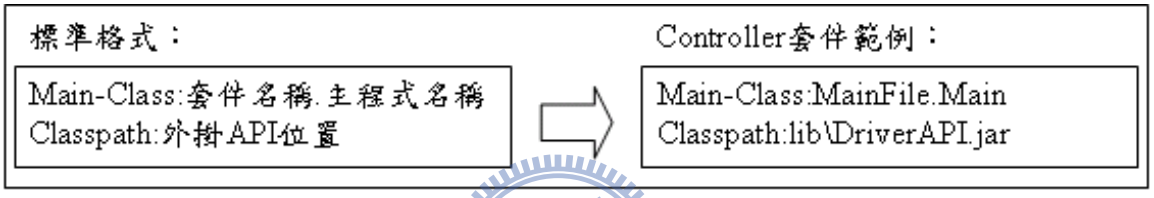


圖 4.22 manifest.mf 檔案內容格式

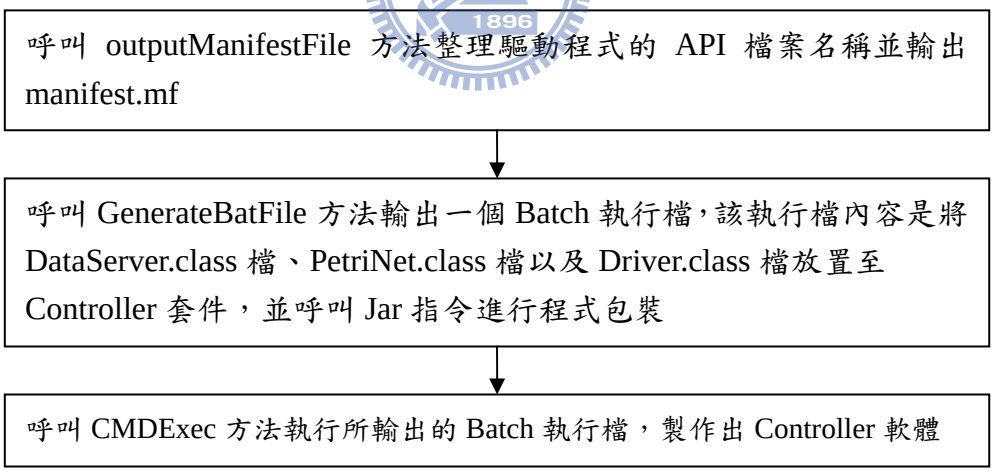


圖 4.23 JarGenerator 類別作業流程

JarGenerator 類別作業程式的步驟二是要呼叫 Jar 指令進行程式碼包裝動作。這首先會將已經編譯成位元組程式碼(bytecobe)的 DataServer.class 檔、PetriNet.class 檔以及 Driver.class 檔放置到 Controller 套件。接著，呼叫出 Jar 指令將 Controller 套件、manifest.mf 檔包裝成一個 Controller.jar 執行檔，並存放在目標目錄底下。JarGenerator 類別的作業流程如圖 4.23 所示。FileGenerator 套件的類別圖如圖 4.24 所示。FileGenerator 套件各類別方法說明如表 4.4。

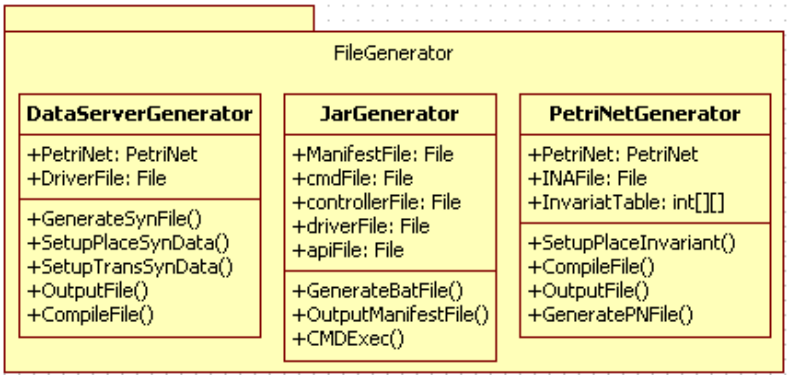


圖 4.24 FileGenerator 套件類別圖

表 4.4 FileGenerator 套件各類別方法說明

類別	方法	功能	傳入值	回傳值
DataServerGenerator	GenerateSynFile	產生 DataServer 檔程式碼	無	無
	SetupPlaceSynData	暫存點同步性資料彙整	無	無
	SetupTransSynData	轉移點同步性資料彙整	無	無
	SetupInitEnvir	彙整虛擬變數初始狀態資料	無	無
	OutputFile	DataServer.java 檔案輸出	無	無
	CompileFile	將 DataServer.java 檔編譯成 class 檔	無	無
PetriNetGenerator	SetupPlaceInvariant	分析暫存點不變量	無	無
	GeneratePNFile	產生 PetriNet.java 檔程式碼	無	無
	OutputFile	輸出 PetriNet.java 檔	無	無
	CompileFile	將 PetriNet.java 檔編譯成 class 檔	無	無
JarGenerator	OutputManifestFile	輸出 Manifest 檔	所使用的 API 檔名稱	無
	GenerateBatFile	產生 Jar 檔指令執行 Batch 檔	無	無
	CMDExec	執行 Batch 檔	無	無

4.4.2.4 Controller 套件的設計

Controller 套件的功能是將使用者設定出的資料伺服器軟體、驅動程式搭配 Controller 套件提供的決策系統整合建置成一個簡易的開放資料伺服器連接環境。該套件內共有六個子套件：(1)PetriNet、(2)DecisionSystem、(3)DataConnectInterface、(4)Driver、(5)GUI 以及(6)MainFile。Controller 套件架構如圖 4.25 所示。

對於「DecisionSystem 套件」而言，它所代表的是開放資料伺服器連接架構的決策系統，其中包含有監控員軟體、偵查員軟體與維修員軟體。監控員軟體能夠判斷裴氏圖中有哪些轉移點是可激發狀態，這是檢查是否轉移點的輸入暫存點內都有浮標，當有轉移點為可激發狀態時，即呼叫資料伺服器軟體進行轉移點激發動作。偵查員軟體能夠檢查系統的暫存點不變量是否遭到破壞，當暫存點不變量遭到破壞時，偵查員軟體會呼叫維修員軟體進行系統修復。而維修員軟體受到呼叫後，則是會提示使用者系統不變量已遭到破壞，並

且暫停所有的控制器動作。

對於「DataConnectInterface 套件」與「Driver 套件」而言，它們所代表的是開放資料伺服器連接架構的資料伺服介面與驅動程式。在使用者執行檔案匯出功能後，軟體會將產生出的 DataServer.class 檔、Driver.class 檔、PetriNet.class 檔分別放入 DataConnectInterface 套件、Driver 套件與 PetriNet 套件中，所以 Controller 套件就能夠藉由使用者所設定的裴氏圖、裴氏圖同步性、驅動程式與該類型的控制器進行溝通。當所使用的控制器類型變換時，使用者只需要更新 Driver 套件中的 Driver.class 檔至相對應的控制器類型，就能繼續使用相同的裴氏圖、裴氏圖同步性設定在新的控制器類型上執行監控作業，而不須重新進行資料伺服器的程式撰碼。

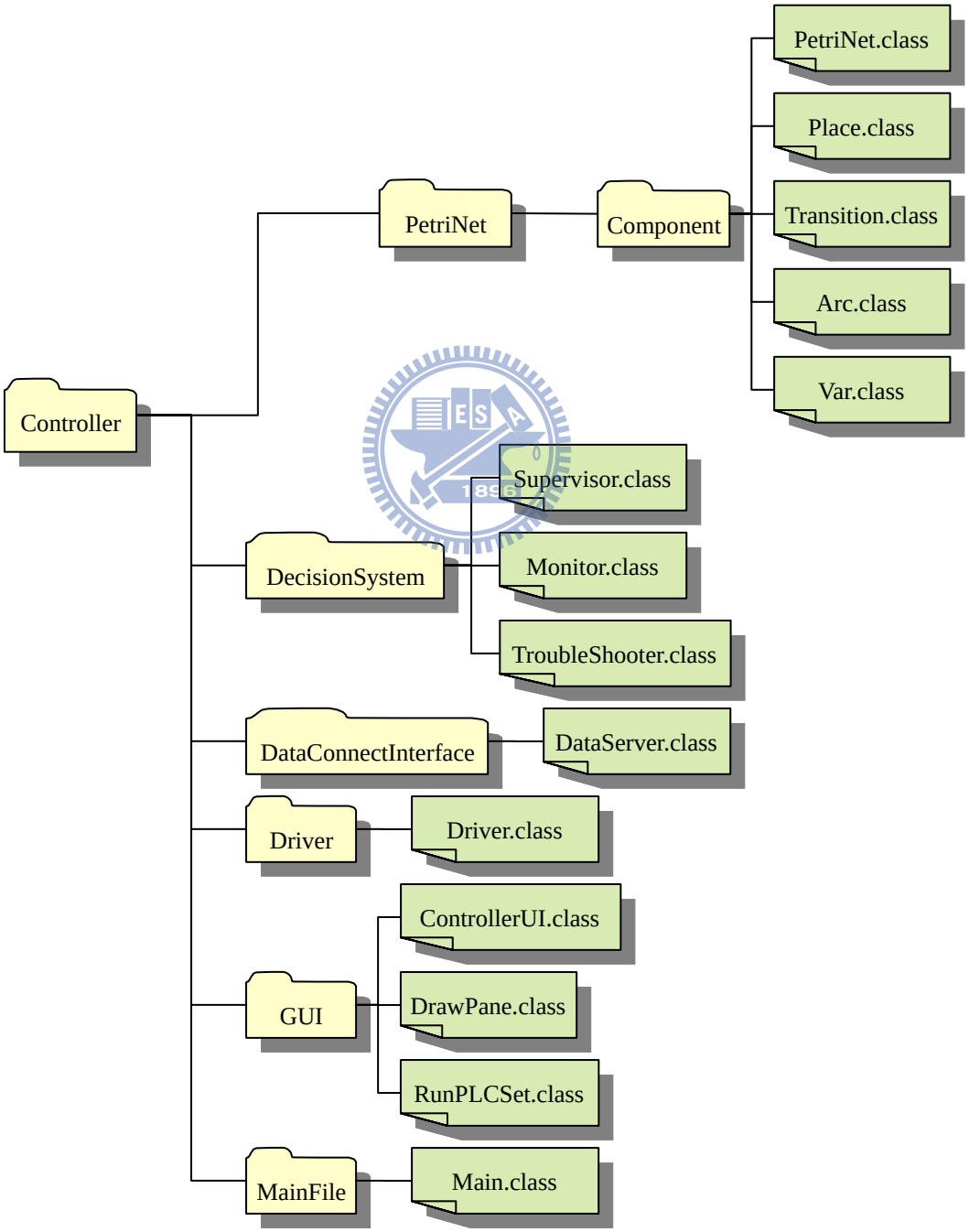


圖 4.25 Controller 套件架構

對於「Controller 套件」而言，其擁有自己的 Main 套件與 GUI 套件，這表示說 Controller 套件本身就可以視為是一個與 PNML 導向資料伺服器連接軟體之間獨立運作的應用程式。事實上，在 PNML 導向資料伺服器連接軟體的作業過程中，並不會實際使用到 Controller 套件內所包含的任何監控功能，而是針對 Controller 套件的程式碼進行編輯，開發出一個客製化的 Controller 應用軟體，並且這個應用軟體會符合開放資料伺服器的架構。Controller 套件類別圖如圖 4.8 所示。Controller 套件各類別方法說明如表 4.5。

表 4.5 Controller 套件各類別方法說明

類別	方法	功能	傳入值	回傳值
Supervisor	CheckTransition	檢查是否有可激發的轉移點	無	轉移點激發狀態矩陣
Monitor	CheckRule	檢查是否有暫存點不變量遭到破壞	無	不變量狀態矩陣
	GetMarking	取得目前浮標狀態	無	無
	SetPlaceInvariant	設定暫存點不變量	暫存點不變量矩陣	無
TroubleShooter	Recovering	進行系統診斷與修復	不變量狀態矩陣	錯誤修復作業編號
DataServer	UpdateMarking	更新浮標狀態	無	無
	FireTransition	進行轉移點激發	轉移點激發作業編號	無
	ErrorRecovering	進行系統錯誤修復	錯誤修復作業編號	無
	Setup	將系統還原至初始狀態	無	無
Driver	OpenPLCConnection	與 PLC 進行連線	控制器 IP 位置與連接埠號碼	無
	ClosePLCConnection	中斷與 PLC 的連線	無	無
	Handle	輸出、輸入信號處理	信號處理指令	無

4.4.3 操作介面的設計

PNML 導向資料伺服器連接軟體的操作介面可以分成三個部分：在畫面最上方的是工具列，工具列裡包含有[檔案]功能選單與[關於]功能選單，[檔案]功能選單主要是用來載入裴氏圖加註語言檔、匯出 Controller 資料夾以及關閉軟體，[關於]功能單則是說明軟體的版本資訊。

在畫面左側的是樹狀目錄面板，該面板會顯示使用者匯入的裴氏圖暫存點、轉移點資料，還有在同步性設定中所使用到的輸入節點、輸出節點數量與虛擬變數資料，使用者可以在樹狀目錄的各節點上雙擊滑鼠左鍵，針對該節點進行相關設定，如暫存點、轉移點的同步性設定、輸入、輸出節點的擴充或縮減、虛擬變數新增與刪除。

在畫面右側的是裴氏圖圖像面板，該面板會顯示使用者所匯入的裴氏圖圖形，使用者可以點選圖形裡的暫存點或是轉移點進行同步性設定。對於尚未設定同步性的暫存點而言，在裴氏圖圖像面板中該暫存點會顯示為白色，並且名稱前方會標記有(I)的字樣，這代表此暫存點尚無對應到任何的元件狀態。反之，暫存點會顯示為紅色，並且名稱前方的(I)字樣會消失。

對於尚未設定同步性的轉移點而言，在裴氏圖圖像面板中該轉移點會顯示為灰色，並且名稱前方會標記有(I, O, T)的字樣，這分別代表轉移點尚未對應到的任何的觸發事件、元件作業並且所設定的啟動延遲時間為零毫秒。當轉移點的觸發事件信號、元件作業以及啟動延遲時間個別設定完成後，轉移點前方的(I, O, T)符號也會陸續消失，並且轉移點會顯示為紅色。PNML 導向資料伺服器連接軟體的操作介面如圖 4.26 所示。

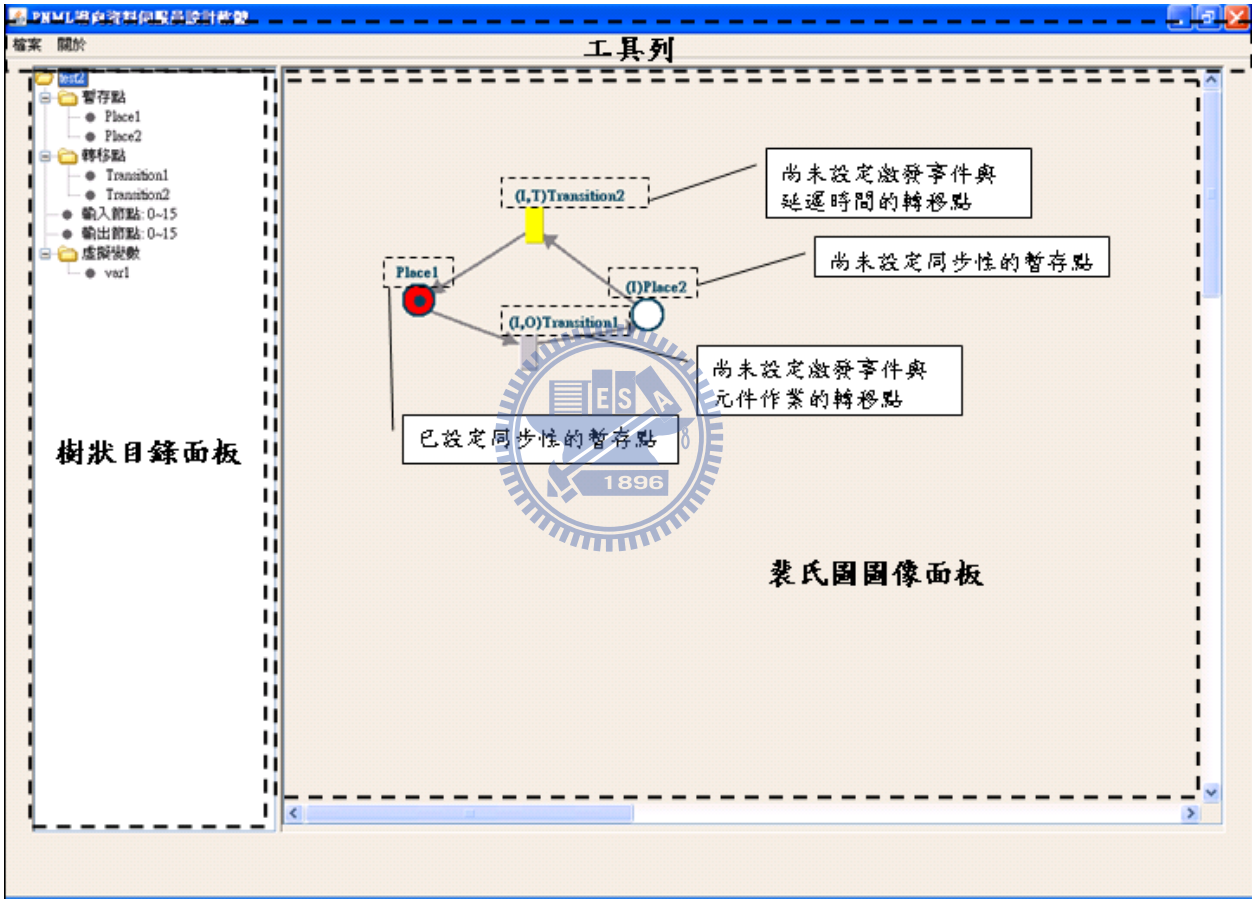


圖 4.26 PNML 導向資料伺服器連接軟體的操作介面

在暫存點同步性設定的畫面裡，包含了該暫存點的同步性資料表以及增加、編輯、刪除與關閉四個按鈕，這些按紐可以用來進行資料表內容的設定。當滑鼠點擊增加按鈕時，會出現新增對應關係的對話框，這裡面共包含五個下拉式選單：連接詞、前括弧、節點編號/變數名稱、運運算元、後括弧，以及一個變數值的文字輸入區。連接詞選單裡有&&(AND)與||(OR)兩個項目，可以用來定義與前一項同步性資料之間的邏輯關係。前括弧與後括弧則可以用來設定同步性資料間的運算順序。在節點編號/變數名稱選單中，可以選擇編號零號開始的控制器輸入節點狀態或是由自行定義的虛擬變數，若選擇的是虛擬變數時則需要再選擇運運算元並且填入適當的變數值進入文字輸入區。當設定完成後，則可以點擊確定鍵

完成設定。暫存點同步性設定畫面如圖 4.27 所示。

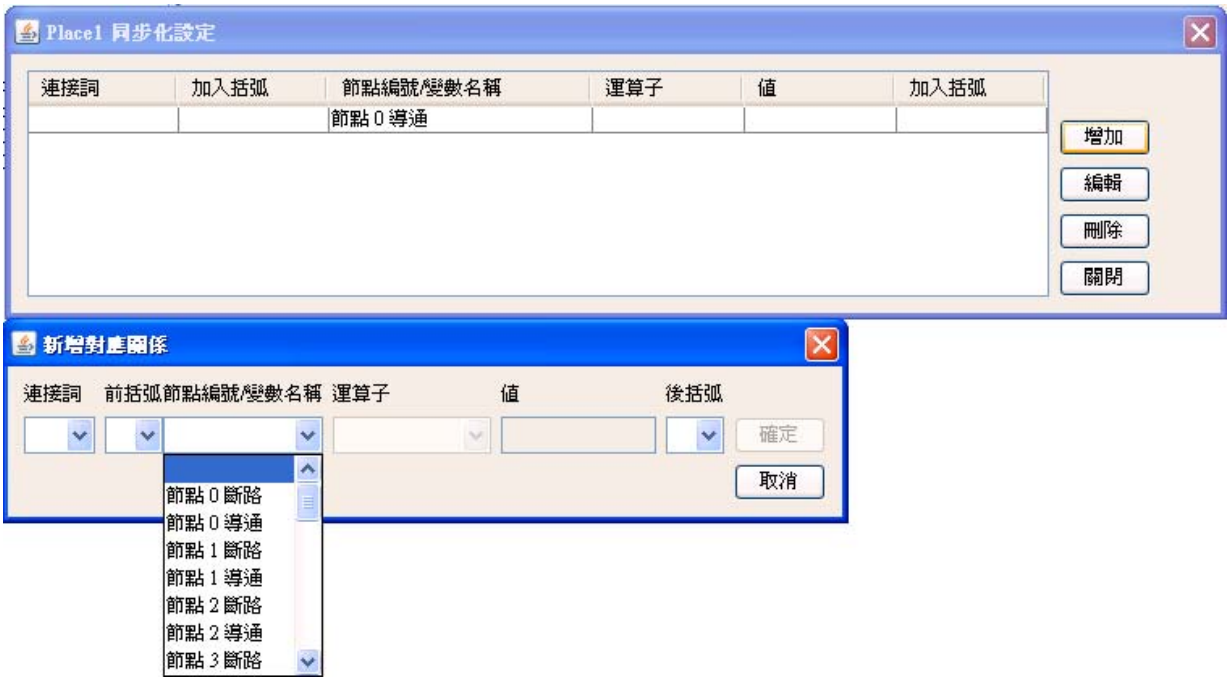


圖 4.27 暫存點同步性設定畫面

在轉移點同步性設定的畫面裡，包含了兩個同步性資料表：位於上方的是有關觸發事件設定的同步性資料表，位於下方的是有關元件作業設定的同步性資料表。與暫存點同樣的，這裡可以點選兩張同步性資料表旁的增加、編輯、刪除與關閉四個按鈕，來進行資料表內容的設定。轉移點同步性設定畫面如圖 4.28 所示。

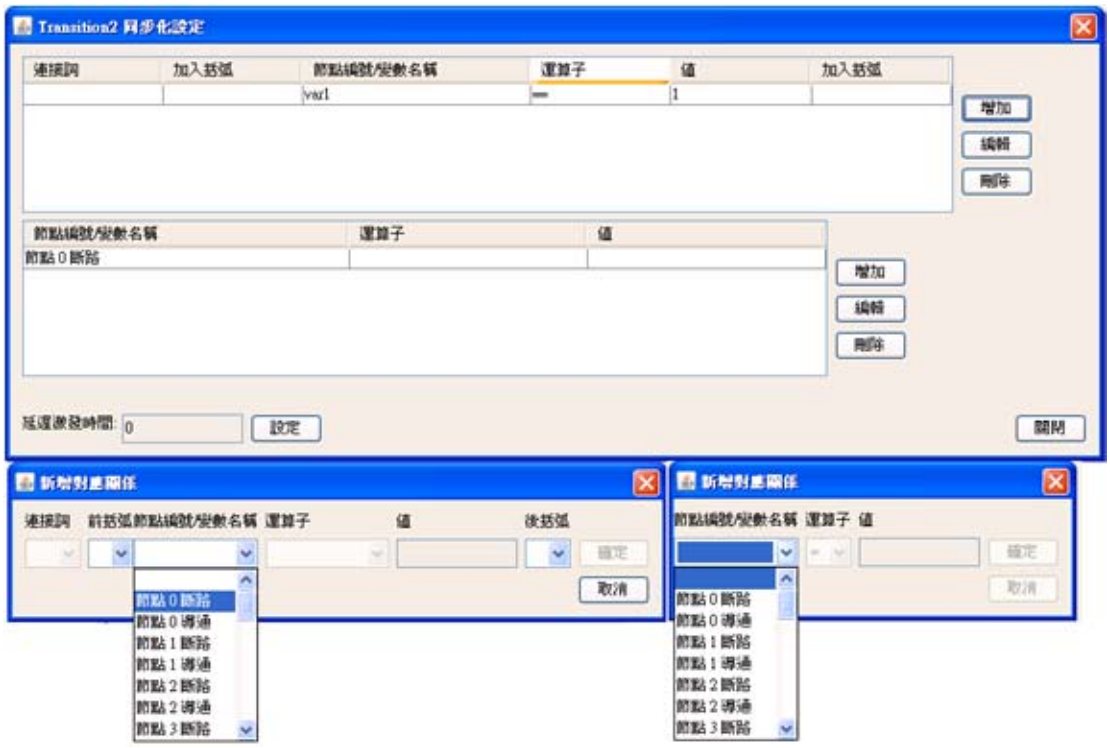


圖 4.28 轉移點同步性設定畫面

當使用者執行工具列[檔案]功能選單的匯出功能後，畫面會出現驅動程式的選取視窗，視窗內要求使用者選取所需使用的控制器驅動程式 class 檔以及驅動程式啟動需要的 API 檔案。在使用者選取完檔案後，可以點選[確定]按鈕匯出檔案。驅動程式選取畫面如圖 4.29 所示

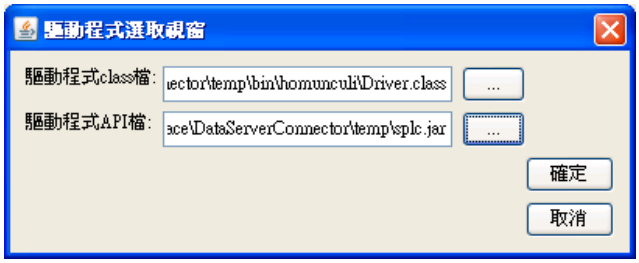


圖 4.29 驅動程式選取對話框

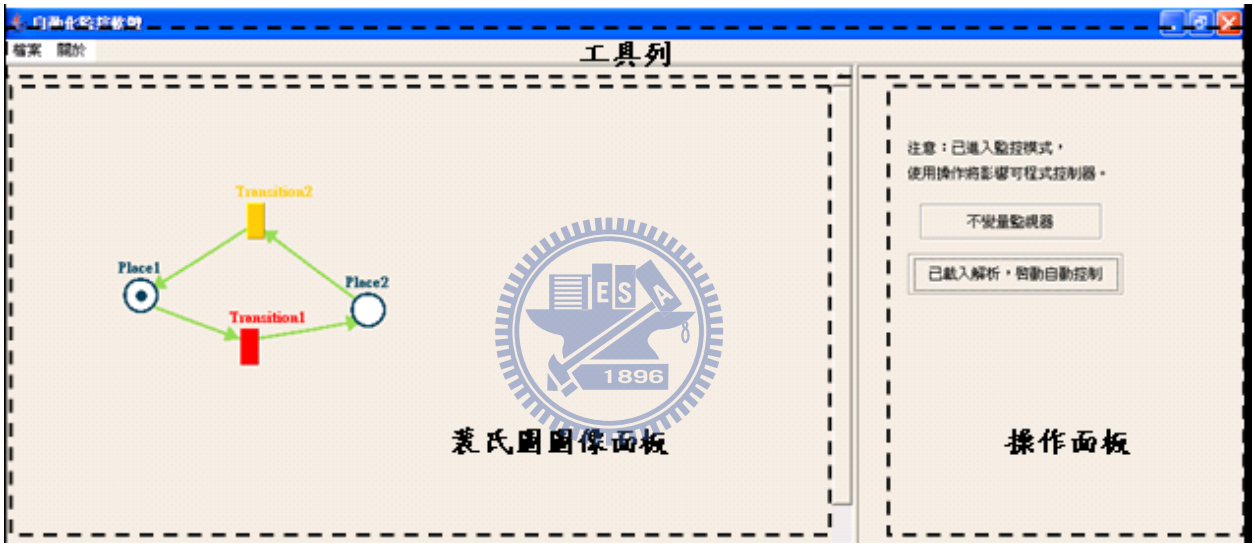


圖 4.30 Controller.jar 執行檔操作介面

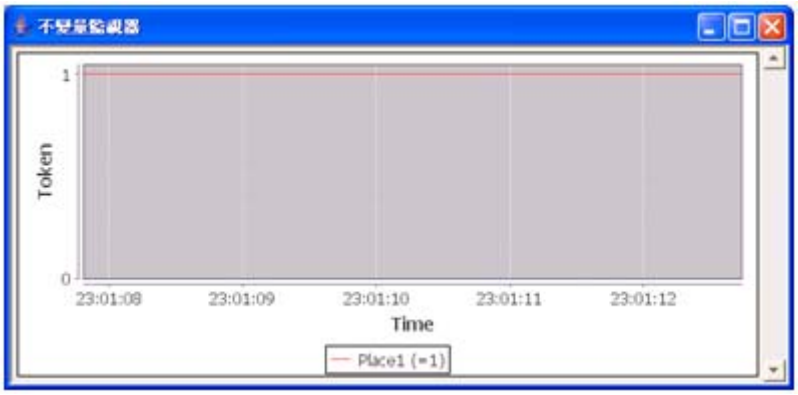


圖 4.31 不變量監控器

在所匯出的 Controller.jar 執行檔的操作介面方面，同樣也分成三個部分：在畫面上方的是工具列，工具列中的[檔案]功能選單僅包含關閉軟體選項，而[關於]功能選單則用來說

明軟體的版本資訊。在畫面左側的是裴氏圖圖像面板，該面板會顯示裴氏圖圖形狀態。其中，激發中轉移點在面板中會呈現為紅色、未激發轉移點會呈現為黃色，在自動控制功能啟動後，該圖形的浮標狀態以及轉移點狀態也會隨著控制行為的進行產生變化。在畫面右側的是操作面板，操作面板裡有不變量監控器啟動按鈕以及自動控制功能啟動按鈕，這分別可以用來啟動不變量監控器啟動按鈕以及啟動自動控制功能。Controller.jar 執行檔操作介面如圖 4.30 所示。不變量監控器如圖 4.31 所示。

4.4.4 軟體執行流程

在 PNML 導向資料伺服器連接軟體的作業程式方面，這可以依照使用者所操作的功能項目的不同區分成六項流程。

對於「PNML 檔案載入功能」而言，當使用者執行工具列檔案選單的載入檔案功能後，GUI 介面首先會呼叫 PetriNet.Component.PetriNet 類別的 parserPNML 方法解析 PNML 檔案裡儲存的資訊。parserPNML 方法裡則會再呼叫 XMLTools.PNMLparser 類別的 parserPlaces、parserTrans、parserArcs 方法擷取出 PNML 檔案內的暫存點、轉移點、箭號資訊，並且呼叫 XMLTools.PNMLparser 類別各個屬性的 Get 方法，如 GetPlaceName、GetTransitionName、GetArcSource 等，將這些所需的資訊回傳至 PetriNet 類別中儲存。GUI 介面再利用 PetriNet 類別存放的資訊將裴氏圖圖形繪製於圖像面板中。PNML 檔案載入功能的執行流程可以用 UML 中的順序圖表達如圖 4.32 所示。

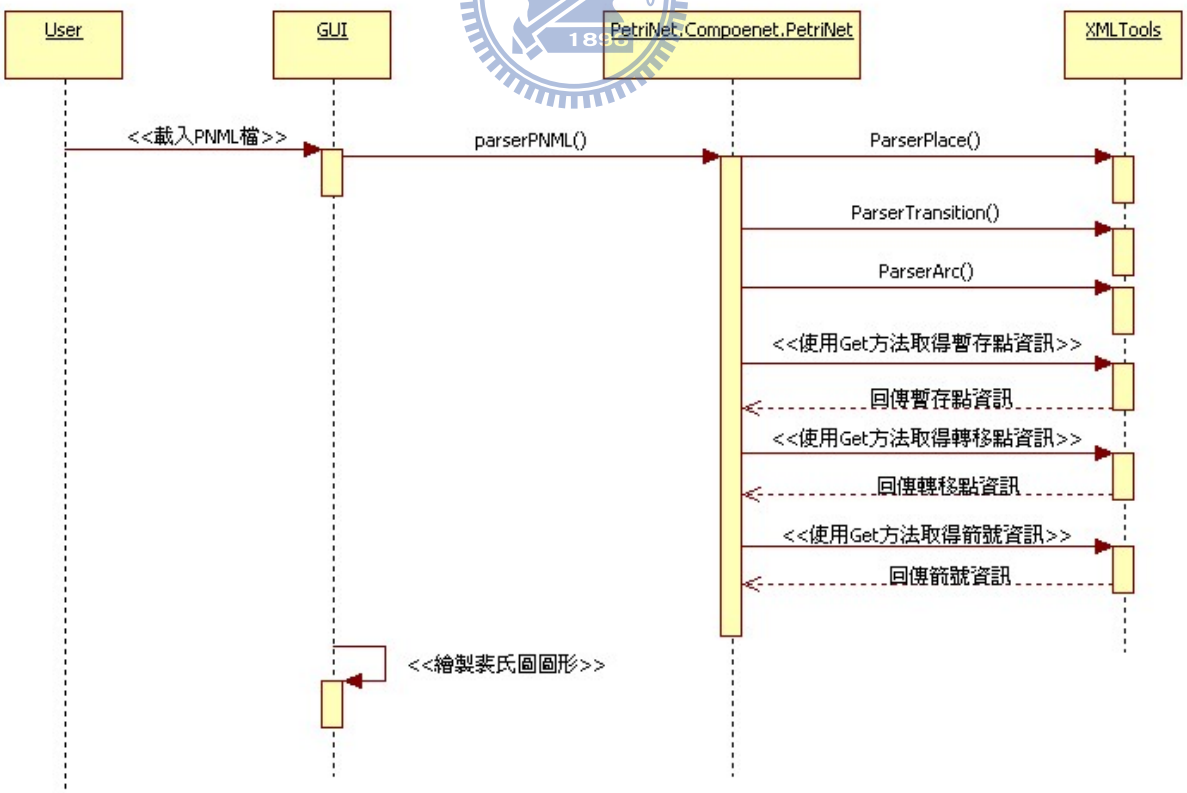


圖 4.32 PNML 檔載入功能執行流程

對於「虛擬變數設定」的作業流程而言，當 GUI 介面感應使用者滑鼠雙擊樹狀目錄面

板的虛擬變數選項時，即會顯示出虛擬變數設定對話框，並且呼叫 PetriNet 類別的 GetVar 方法取得虛擬變數資訊表，載入到虛擬變數對話框裡。接著，使用者輸入完成所需的虛擬變數之後，GUI 介面會再呼叫 PetriNet 類別的 SetVar 方法儲存新的虛擬變數資訊表。

對於「控制器節點數量設定」而言，當 GUI 介面感應使用者滑鼠雙擊樹狀目錄面板的輸入節點或輸出節點選項時，即會顯示出輸入或輸出節點設定對話框，並且呼叫 PetriNet 類別的 GetInputNodeNumber 方法或是 GetOutputNodeNumber 方法取得目前輸出、輸入節點數量資訊，載入到對話框裡。接著，使用者輸入完成所需的節點數量之後，GUI 介面會再呼叫 PetriNet 類別的 SetInputNodeNumber 方法或是 SetOutputNodeNumber 方法更新節點數量資訊。虛擬變數設定、控制器節點數量設定功能的執行流程可以用 UML 中的順序圖表達如圖 4.33 所示。

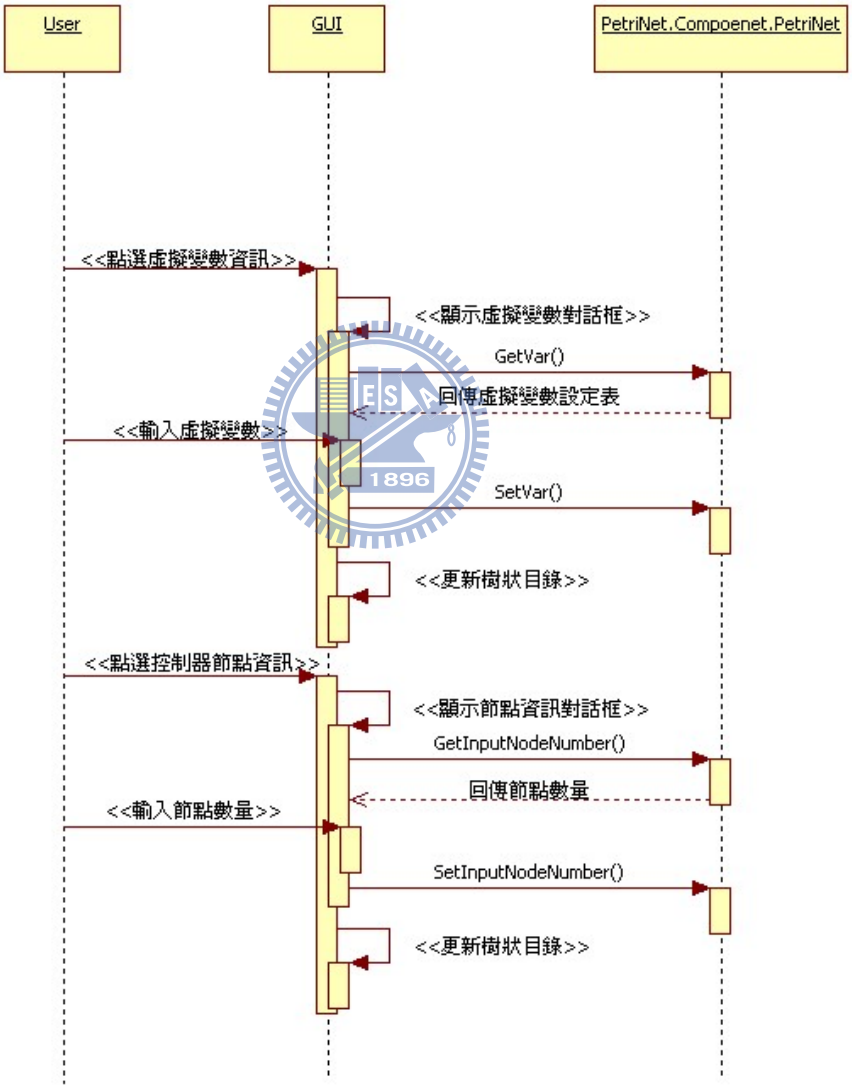


圖 4.33 虛擬變數設定、控制器節點數量設定功能執行流程

對於「暫存點同步性設定」而言，當使用者點選裴氏圖圖形面板中的暫存點圖示或是樹狀目錄面板裡的暫存點名稱時，即會顯示暫存點同步性設定對話框，並且呼叫 PetriNet 類別裡的 GetPlace.GetSynchronizedTable 方法取得暫存點屬性的元件狀態同步性資料表，載入到暫存點同步性設定對話框。接著，使用者輸入完成暫存點所對應的元件狀態資料以

後，GUI 介面會再呼叫 PetriNet 類別裡的 GetPlace.SetSynchronizedTable 方法更新該暫存點屬性的元件狀態同步性資料以及變更圖像面板中暫存點的顏色。

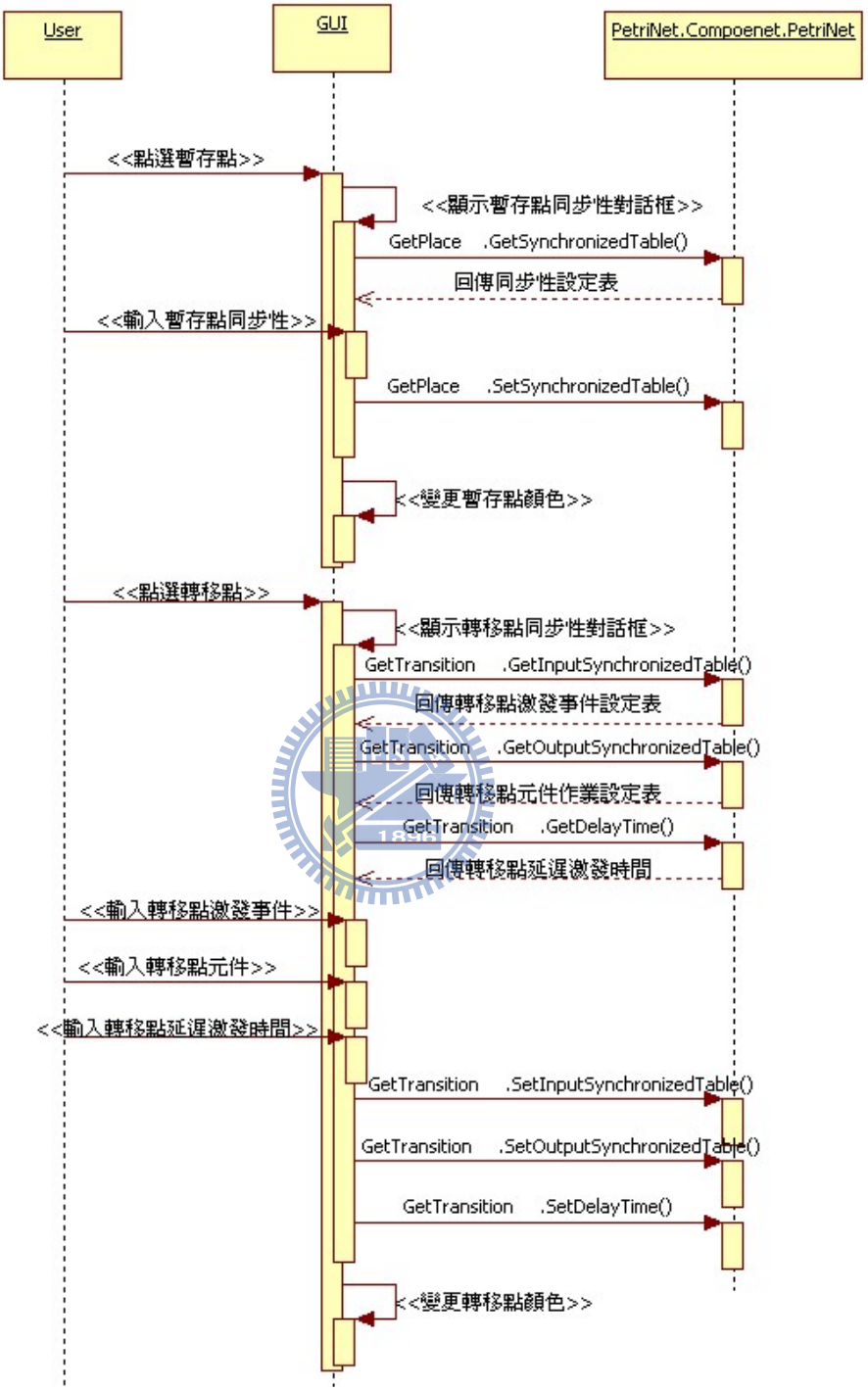


圖 4.34 暫存點、轉移點同步性設定功能執行流程

對於「轉移點同步性設定」而言，當使用者點選裴氏圖圖形面板中的轉移點圖示或是樹狀目錄面板裡的轉移點名稱時，即會顯示轉移點同步性設定對話框，並且分別呼叫 PetriNet 類別裡的 GetTransition.GetInputSynchronizedTable、GetTransition.GetOutputSynchronizedTable、GetTransition.GetDelayTime 方法取得轉移點屬性的激發事件同步性資料表、元件作業同步性資料表與轉移點的延遲激發時間，載入至轉

移點同步性設定對話框。接著，使用者輸入完成轉移點的同步性設定之後，GUI 介面會再呼叫 PetriNet 類別裡的 GetTransition.SetInputSynchronizedTable 、GetTransition.SetOutputSynchronizedTable、GetTransition.SetDelayTime 方法更新該轉移點的激發事件同步性資料、元件作業同步性資料與轉移點的延遲激發時間，以及變更圖像面板中轉移點的顏色。暫存點、轉移點同步性設定功能的執行流程可以用 UML 中的順序圖表達如圖 4.34 所示。

對於「檔案匯出功能」而言，當使用者執行工具列檔案選單的匯出檔案功能後，GUI 介面首先會顯示驅動程式選取對話框與檔案匯出位置選取對話框，讓使用者選擇欲使用的驅動程式以及 Controller 資料夾的匯出位置。接著，依序呼叫 FileGenerator.DataServerGenerator 類別的 SetupPlaceSynData 、SetupTransSynData 、SetupInitEnvir 、OutputFile 、CompileFile 方法匯出資料伺服器軟體，FileGenerator.PetriNetGenerator 類別的 SetupPlaceInvariant、GeneratePNFile、OutputFile、CompileFile 方法匯出 PetriNet.class 檔，以及 FileGenerator.JarGenerator 類別的 OutputManifestFile、GenerateBatFile、CMDExec 方法匯出 Controller.jar 檔案至目標目錄下。檔案匯出功能的執行流程可以用 UML 中的順序圖表達如圖 4.35 所示。

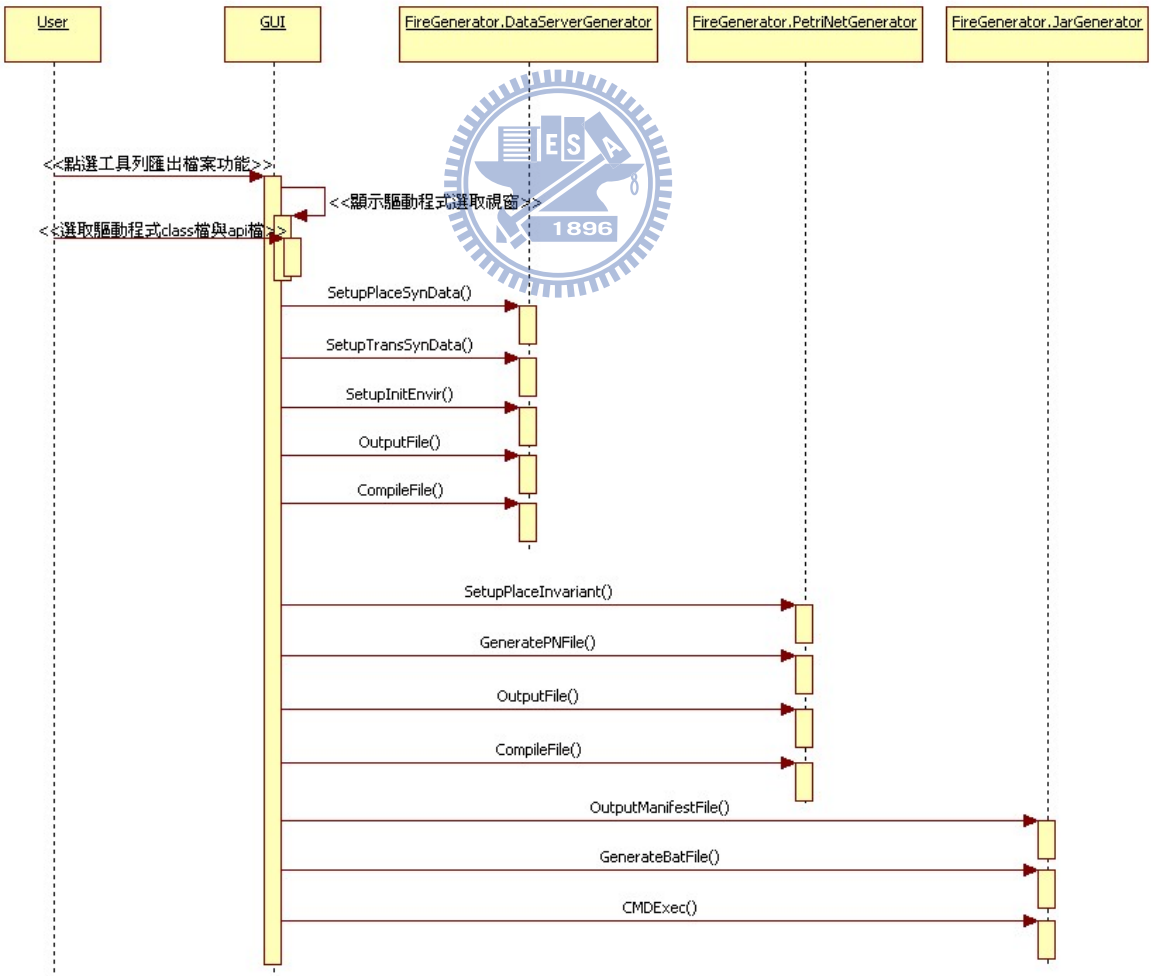


圖 4.35 PNML 導向資料伺服器連接軟體 PNML 檔匯出執行流程

第五章 自動化製造系統範例的分析與操作

本章將會以一個灌模系統作為實體系統範例，說明以開放資料伺服器連接架構為基礎的製造系統自動化監控軟體的建置流程。5.1 節簡介灌模系統所涵蓋的功能，有關灌模系統的詳細規格收錄在附錄一中提供參考。5.2 節設計灌模系統的監控員軟體。5.3 節設計灌模系統的偵查員軟體。5.4 節是以灌模系統所使用的控制器為例，說明其驅動程式的設計。5.5 節是利用 PNML 導向資料伺服器連接軟體設定灌模系統的裴氏圖同步性與驅動程式，並藉由軟體的自動化撰碼功能產生出灌模系統的監控軟體。

5.1 灌模系統的功能說明

圖 5.1 為灌模系統的照片，此系統的功能在於使用灌漿造模方式進行模型的製作。系統的硬體架構是採用鄭瑋弘[12]設計的灌模系統為雛型建置而成。此系統包含三項特色：(1)導入無線射頻識別技術(Radio Frequency Identification, RFID)[20]輔助系統運行、(2)運用擬陣理論設計其偵查員軟體、(3)使用 ODSC 作為裴氏圖控制器與 PLC 之間的連接架構。

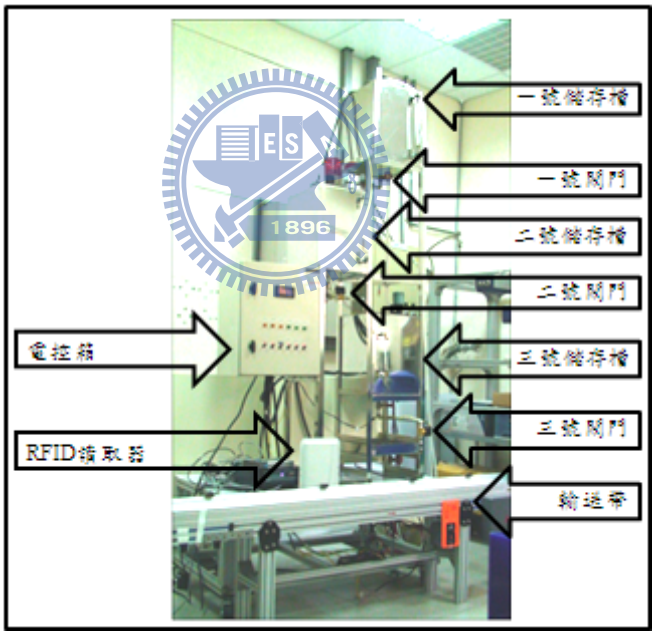


圖 5.1 灌模系統照片

在灌模系統的作業流程裡，無線射頻識別標籤可視為加工物件的作業工單。每個進入灌模作業的模具上，都裝置有一個無線射頻識別標籤用來紀錄該模具在製程中所需使用的參數值。當模具進入工作站後，系統會讀取出標籤所記載的作業參數並進行其作業程式，讓製程裡的加工機台能夠即時性的因應生產線上的各種變化，對於不同類型的待加工產品自動轉換其作業模式。無線射頻識別技術的加入，無疑地有助於提升製程的自動化程度，同時也能夠取代以往在加工機台作業模式轉換上所需的人工作業以及減少人工作業可能造成的失誤，讓系統在運作上能夠更為穩健。圖 5.2 為灌模系統所使用的超高頻(Ultra High Frequency, UHF)RFID 標籤及讀取器 Intermec IF5 fixed Reader[20]，IF5 讀取器的優點在於

具備了 RFID 標籤資訊讀取與寫入的功能，並支援使用者以 Java 程式語言進行相關操作。

(a) IF5 RFID Fixed Reader　　(b)超高頻RFID標籤

圖 5.2 RFID 讀取機 IF5 及標籤

對於灌模系統硬體架構而言，灌模系統是由兩個子系統所構成：模具輸送系統及備料系統，如圖 5.3 所示。模具輸送系統的功能在於運載模具至指定的位置，並根據模具裝置的無線射頻標籤記錄的作業參數進行澆模作業，其硬體設備涵蓋一個閥門、一組輸送帶、一個無線射頻識別標籤讀取器及四個紅外線感知器；備料系統的功能在於調製出模型硬化成型所需的物料，其硬體設備涵蓋三個幫浦、三個液體儲存槽、三個液位感知器、兩個攪拌器以及兩個閥門。灌模系統的各項硬體裝置，除了無線射頻識別標籤的讀取器是透過直接連線的方式與其進行溝通以外，其餘設備皆是整合至可程式邏輯控制器(Programmable Logic Controller, PLC)，透過控制器進行控制。

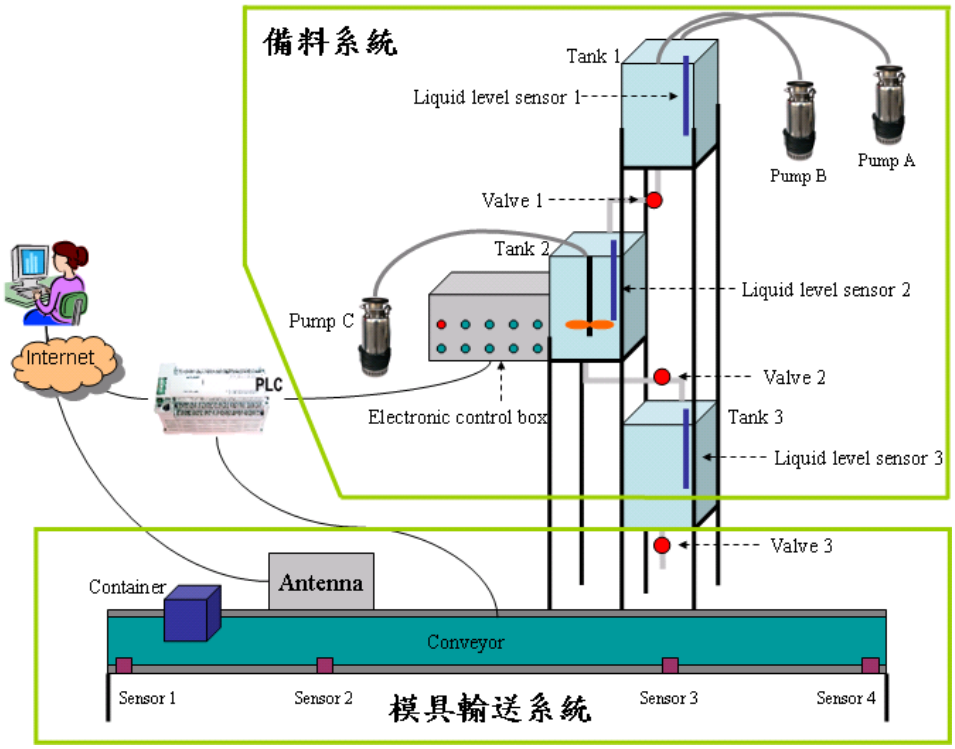


圖 5.3 灌模系統

可程式邏輯控制器的主要功能在於其輸出、輸入模組。輸出模組的功能類似於各項硬體設備的啟動開關，控制器可以透過導通或切斷連接在輸出模組各個端點上的電源線路，來啟動或關閉電子設備。輸入模組的功能如同信號接收器，控制器可以利用輸入模組各個端點接收各項設備的電子訊號以瞭解該設備的狀態。

此外，可程式邏輯控制器的內部安裝有一個中央處理單元(Central Processing Unit, CPU）以及 Java 虛擬機器(Java Virtual Machine)，能夠進行資料處理及邏輯運算，使用者可以自行撰寫控制邏輯至控制器，讓控制器自動進行一系列的輸出、輸入行為控制。圖 5.4 為灌模系統使用的可程式邏輯控制器 SoftPLC。灌模系統內各裝置的詳細資料如表 5.1 所示。

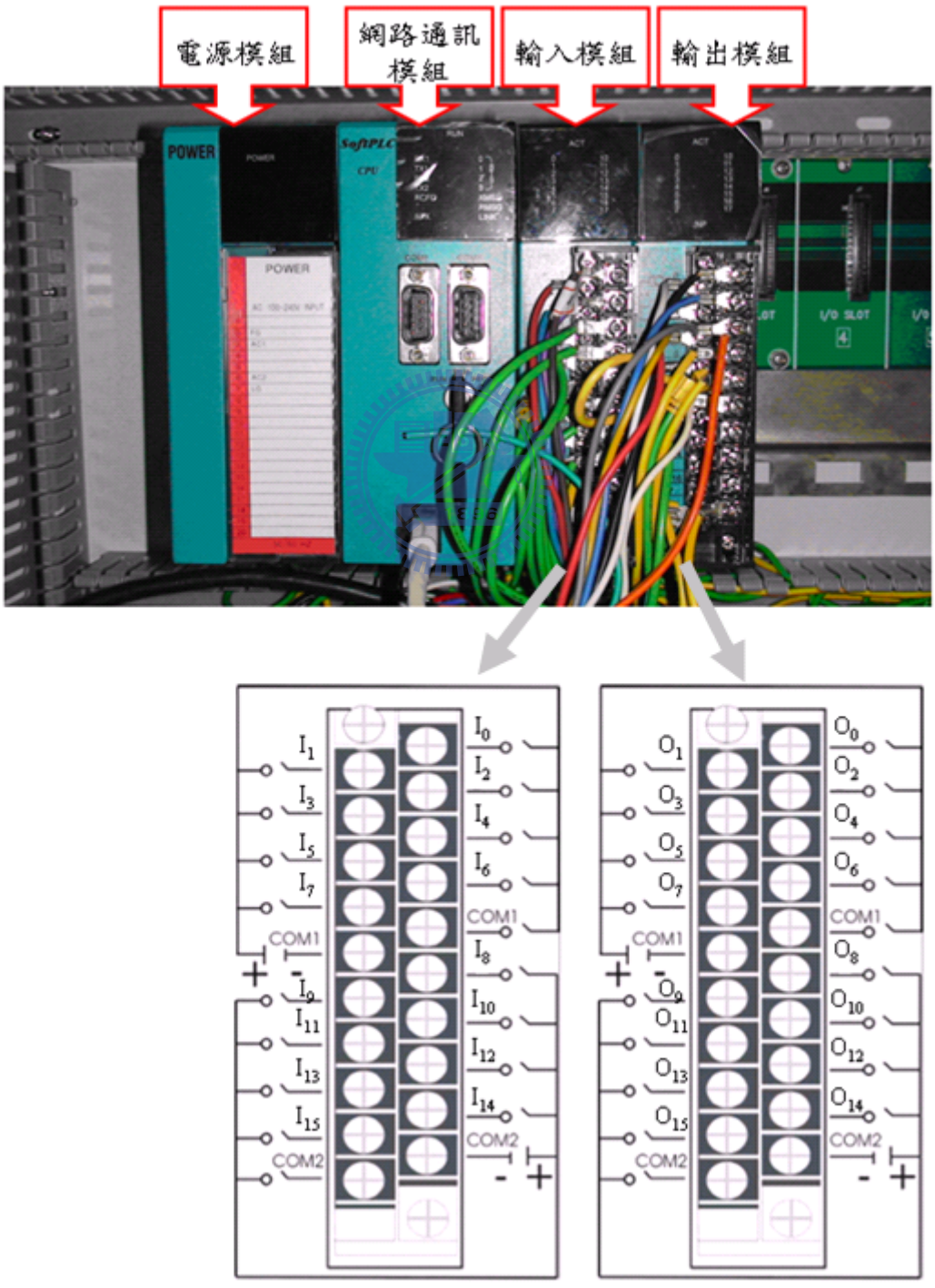


圖 5.4 可程式邏輯控制器 SoftPLC

表 5.1 灌模系統硬體裝置資料

裝置編號	裝置名稱	隸屬系統	信號輸入節點	信號輸出節點	功能
1	一號儲存槽	備料系統	無	無	調製物料 Y
2	二號儲存槽	備料系統	無	無	調製物料 X
3	三號儲存槽	備料系統	無	無	儲存物料 X
4	幫浦 A	備料系統	I_0	O_0	導入物料 A
5	幫浦 B	備料系統	I_1	O_1	導入物料 B
6	幫浦 C	備料系統	I_2	O_2	導入物料 H
7	一號閥門	備料系統	I_3	O_3	控制物料 Y 的流入二號儲存槽
8	二號閥門	備料系統	I_4	O_4	控制物料 X 的流入三號儲存槽
9	三號閥門	模具輸送系統	I_5	O_5	控制物料 X 的流入模具
10	一號攪拌器	備料系統	I_6	O_6	混合物料 A、B
11	二號攪拌器	備料系統	I_7	O_7	混合物料 Y、H
12	一號液位感知器	備料系統	I_8	無	感應一號儲存槽最高水位
13	二號液位感知器	備料系統	I_9	無	感應二號儲存槽最高水位
14	三號液位感知器	備料系統	I_{10}	無	感應二號儲存槽最高及最低水位
15	輸送帶	模具輸送系統	I_{11}	O_8(正向) O_9(反向)	運送模具
16	RFID 讀取器暨天線	模具輸送系統			讀取模具 RFID 標籤資訊
17	一號紅外線感知器	模具輸送系統	I_{12}		感應模具進入模具輸送系統
18	二號紅外線感知器	模具輸送系統	I_{13}		模具 RFID 讀取位置定位
19	三號紅外線感知器	模具輸送系統	I_{14}		模具澆模位置定位
20	四號紅外線感知器	模具輸送系統	I_{15}		感應模具離開模具輸送系統
21	可邏輯控制器	備料系統 模具輸送系統			整合硬體裝置與控制程式

5.2 灌模系統的監控員設計

在系統裴氏圖建立方面，可以使用階層轉換法來將系統的 IDEF0 規格圖轉換成裴氏圖。IDEF0 規格圖主要是提供使用者一種靜態方法表達系統內所涵蓋的功能以及各功能的執行順序，較難分析系統在時間進行下可能導致的狀態變化。在系統的動態行為模擬與驗證方面，這可以藉由裴氏圖分析。階層轉換法的轉換邏輯是將 IDEF0 圖裡代表系統功能及動作的作業方格轉換成裴氏圖的轉移點，而代表系統資訊流的線段轉換成裴氏圖的暫存點，最後依據系統的初始狀態給予各暫存點浮標數，轉換方式如圖 5.5。有關灌模系統的 IDEF0 規格圖設計可參考附錄一。

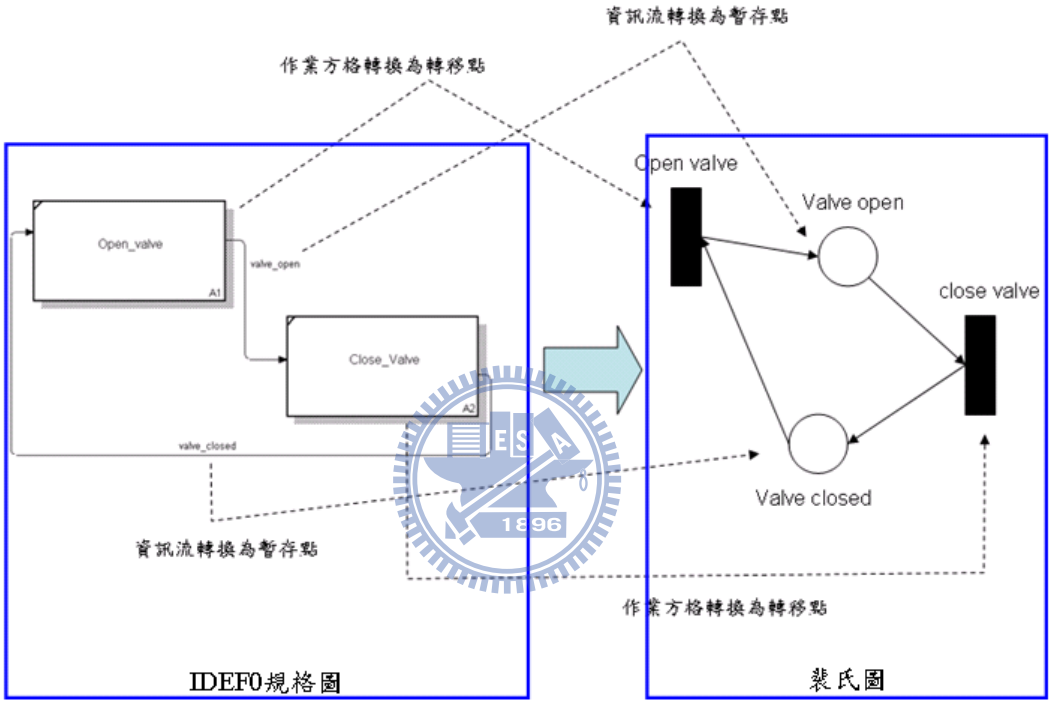


圖 5.5 階層轉換法—IDEF0 轉換裴氏圖

當系統 IDEF0 規格圖過於複雜時，以人工作業方式進行階層轉換法需要花費大量的時間，並且容易在轉換期間發生資訊遺漏的情形。階層轉換法自動化軟體[2]提供使用者將繪製完成的 IDEF0 規格圖載入軟體內，並自動產生一個相對應的裴氏圖加註語言格式檔案。對於不同檔案格式之間的轉換，該軟體採用的是樹形轉換法，其原理是展開 IDEF0 檔以及裴氏圖加註語言檔的樹狀資料結構，再依據階層轉換法的轉換規則進行兩種樹狀結構之間的資料對應與輸出。圖 5.6 為階層轉換法自動化軟體的操作畫面。

由階層轉換法自動化軟體所產生的灌模系統裴氏圖如圖 5.7 所示。此模式分成上下兩個部份，其中下半部代表的是模具輸送系統、上半部代表的是備料系統。模具輸送系統共包含 7 個轉移點、16 個暫存點，初始狀態下共有 7 個浮標分別在暫存點 2、3、6、8、10、12、16 內。模具輸送系統的轉移點及暫存點意義如表 5.2 所示。備料系統共包含 10 個轉移點、24 個暫存點，初始狀態下共有 12 個浮標分別在暫存點 18、20、22、24、26、28、20、32、34、36、38、39 內。模具輸送系統的轉移點及暫存點意義如表 5.3 所示。

模具輸送系統與備料系統裴氏圖的狀態方程式可以表達為 $P' = P + R^T t$，P 為目前暫存

點浮標數向量、P'為狀態轉換後的暫存點浮標數向量，t 為轉移點激發向量，R 為法則矩陣(Rule Matrix)。備料系統與模具輸送系統的法則矩陣如圖 5.8 所示。灌模系統的監控法則可參考附錄二。

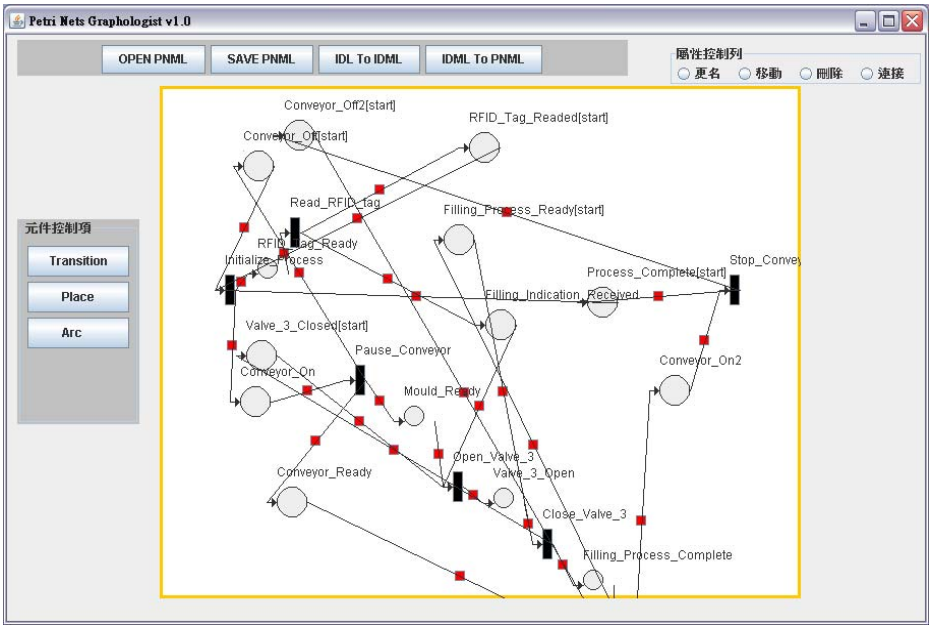
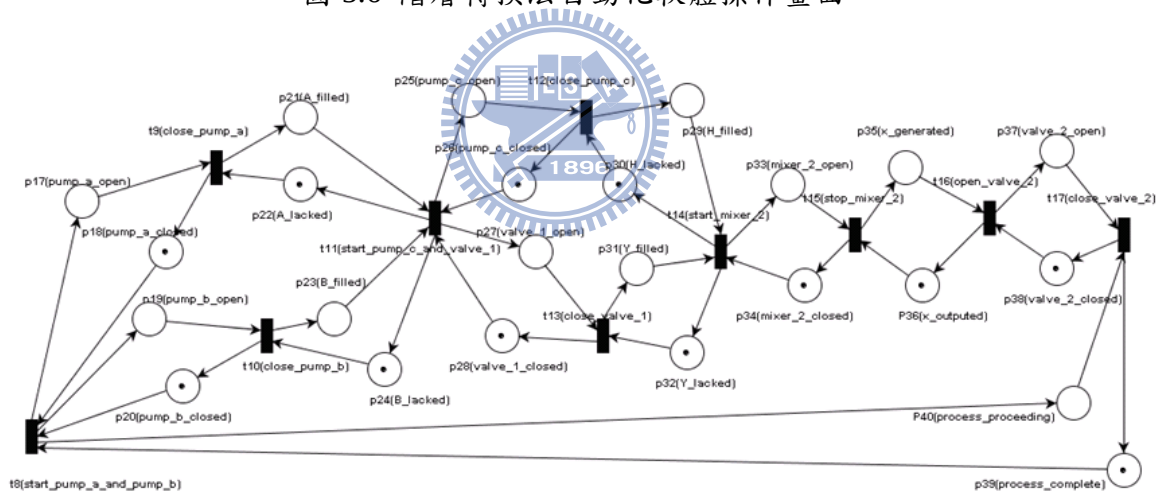
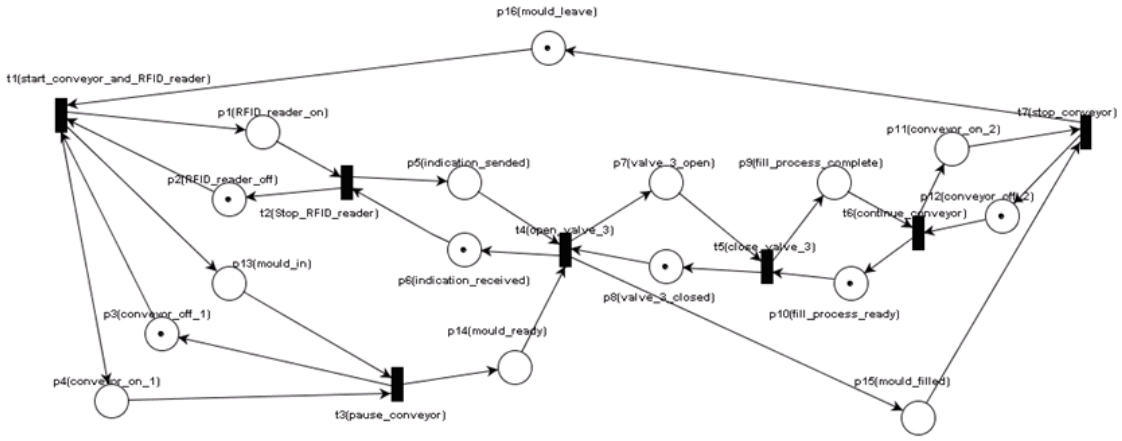


圖 5.6 階層轉換法自動化軟體操作畫面



(a)備料系統裴氏圖



(b)模具輸送系統裴氏圖

圖 5.7 灌模系統裴氏圖

表 5.2 模具輸送系統裴氏圖暫存點與轉移點的說明

暫存點之意義		$f_p(P_i)$	轉移點之意義		$f_t(T_i)$	轉移點激發時間(秒)
P_1	RFID 標籤讀取器已開啟	Var:Reader = on	T_1	啟動輸送帶及 RFID 標籤讀取器	({I_{12}=1},{$O_{8:}$=1,mold:=in, Reader:=on},0)	0.5
P_2	RFID 標籤讀取器已關閉	Var:Reader = off,	T_2	停止 RFID 標籤讀取器	({e},{Reader:=off,讀取資訊 Indication:= tag 資訊},0)	7
P_3	輸送帶已停止(1)	I_{11}=1 or Var:Switch!=1	T_3	暫停輸送帶	({I_{14}=1},{O_8=0,Switch:=2, Mold:=ready},0)	13.6
P_4	輸送帶已啟動(1)	I-$_{11}$=0,Var:Switch=1	T_4	開啟三號閥門	({e},{O_4=1,indication:=null Mold:=filled},0)	0.5
P_5	作業指令已傳遞	Var:Indication != null	T_5	關閉三號閥門	({e},{O_4=0, Process_1:= complete },0)	5
P_6	作業指令已接收	Var:Indication == null	T_6	繼續啟動輸送帶	({e},{O_8=1, Process_1:= ready },2000)	2.5
P_7	三號閥門已開啟	I_5=0	T_7	停止輸送帶	({I_{15}=1}{O_8=0,Switch:=1, Mold = leave}0)	5.5
P_8	三號閥門已關閉	I_5=1				
P_9	澆模程式已結束	Var:Process_1 = complete				
P_{10}	澆模程式待命中	Var:Process_1 = ready				
P_{11}	輸送帶已啟動(2)	I_{11}=0 ,Var:Switch=2				
P_{12}	輸送帶已停止(2)	I_{11}=1 or Var:Switch=1				
P_{13}	模具進入	Var:Mold = in				
P_{14}	模具就定位	Var:Mold = ready				
P_{15}	模具已填滿	Var:Mold = filled				
P_{16}	模具離開	Var:Mold = leave				

：系統初始狀態

表 5.3 備料系統裴氏圖暫存點與轉移點的說明

暫存點之意義		$f_p(P_i)$	轉移點之意義		$f_t(T_i)$	轉移點激發時間(秒)
P_{17}	幫浦 a 已啟動	I_0=0	T_8	啟動幫浦 a 及幫浦 b	({I_{10}=1},{O_0=1, O_1=1, Process_2:=proceeding},0)	0.5
P_{18}	幫浦 a 已關閉	I_0=1	T_9	關閉幫浦 a	({I_8=0},{ O_0=0, m_a:=filled },30000)	50
P_{19}	幫浦 b 已啟動	I_1=0	T_{10}	關閉幫浦 b	({I_8=0},{ O_1=0, m_b:=filled },35000)	50
P_{20}	幫浦 b 已關閉	I_1=1	T_{11}	啟動幫浦 c 及一號閥門	({e},{O_2=1,O_3=1 m_a:=lacked, m_b:=lacked },0)	0.5
P_{21}	物料 A 已填充	Var:m_a=filled	T_{12}	關閉幫浦 c	({I_9=0},{O_2=0,m_h:=filled},30000)	239
P_{22}	物料 A 已不足	Var:m_a=lacked	T_{13}	關閉一號閥門	({I_9=0},{O_3=0,m_y:=filled},90000)	239
P_{23}	物料 B 已填充	Var:m_b=filled	T_{14}	啟動二號攪拌器	({e},{O_7=1, m_h=lacked, m_y=lacked },0)	0.5
P_{24}	物料 B 已不足	Var:m_b=lacked	T_{15}	停止二號攪拌器	({e},{O_7=0,m_x:=generated},20000)	10
P_{25}	幫浦 c 已啟動	I_2=0	T_{16}	開啟二號閥門	({e},{O_4=1, m_x:=outputed },0)	0.5
P_{26}	幫浦 c 已關閉	I_2=1	T_{17}	關閉二號閥門	({I_{10}=0},{O_4=0, Process_2:=complete},90000)	2877
P_{27}	一號閥門已開啟	I_3=0				
P_{28}	一號閥門已關閉	I_3=1				
P_{29}	物料 H 已填充	Var:m_h=filled				
P_{30}	物料 H 已不足	Var:m_h=lacked				
P_{31}	物料 Y 已填充	Var:m_y=filled				
P_{32}	物料 Y 已不足	Var:m_y=lacked				
P_{33}	二號攪拌器已啟動	I_7=0				
P_{34}	二號攪拌器已停止	I_7=1				
P_{35}	物料 X 已調製完成	Var:m_x=generated				

：系統初始狀態

表 5.3 備料系統裴氏圖暫存點與轉移點的說明(續)

暫存點之意義		$f_p(P_i)$
P_{36}	物料 X 已輸出	Var:m_x=outputed
P_{37}	二號閥門已開啟	I_4=1
P_{38}	二號閥門已關閉	I_4=0
P_{39}	備料程式已結束	Var:Process_2 = complete
P_{40}	備料程式已啟動	Var:Process_2 = proceeding

：系統初始狀態

	t1	t2	t3	t4	t5	t6	t7
p1	1	-1	0	0	0	0	0
p2	-1	1	0	0	0	0	0
p3	-1	0	1	0	0	0	0
p4	1	0	-1	0	0	0	0
p5	0	1	0	-1	0	0	0
p6	0	0	1	-1	0	0	0
p7	0	0	0	1	-1	0	0
p8	0	0	0	-1	1	0	0
p9	0	0	0	0	1	-1	0
p10	0	0	0	0	-1	1	0
p11	0	0	1	0	0	-1	0
p12	0	0	0	0	0	1	-1
p13	0	0	0	0	0	-1	1

(a)模具輸送系統法則矩陣

	t8	t9	t10	t11	t12	t13	t14	t15	t16	t17	t18	t19
p15	1	-1	0	0	0	0	0	0	0	0	0	0
p16	-1	1	0	0	0	0	0	0	0	0	0	0
p17	1	0	-1	0	0	0	0	0	0	0	0	0
p18	-1	0	1	0	0	0	0	0	0	0	0	0
p19	0	1	0	-1	0	0	0	0	0	0	0	0
p20	0	0	1	-1	0	0	0	0	0	0	0	0
p21	0	0	0	1	-1	0	0	0	0	0	0	0
p22	0	0	0	-1	1	0	0	0	0	0	0	0
p23	0	0	0	0	1	-1	0	0	0	0	0	0
p24	0	0	0	0	-1	1	0	0	0	0	0	0
p25	0	0	0	0	0	1	-1	0	0	0	0	0
p26	0	0	0	0	0	-1	1	0	0	0	0	0
p27	0	0	0	0	0	1	0	-1	0	0	0	0
p28	0	0	0	0	0	-1	0	1	0	0	0	0
p29	-1	0	0	0	0	0	0	1	0	0	0	0
p30	0	0	0	0	0	0	1	0	-1	0	0	0
p31	0	0	0	0	0	0	0	1	-1	0	0	0
p32	0	0	0	0	0	0	0	0	1	-1	0	0
p33	0	0	0	0	0	0	0	0	-1	1	0	0
p34	0	0	0	0	0	0	0	0	0	1	-1	0
p35	0	0	0	0	0	0	0	0	0	-1	1	0
p36	0	0	0	0	0	0	0	0	0	0	1	-1
p37	0	0	0	0	0	0	0	0	0	0	-1	1
p38	-1	0	0	0	0	0	0	0	0	0	0	1

(b)備料系統法則矩陣

圖 5.8 灌模系統法則矩陣

在裴氏圖的驗證方面，首先可以進行鎖死分析，此外模具輸送系統與備料系統的裴氏圖均具有記號圖(Marked Graph)性質，這指的是裴氏圖圖形中每個暫存點前後的箭號數都為一。對於具有記號圖性質的裴氏圖，可以進行最短週期時間分析以及不變量分析[5]。

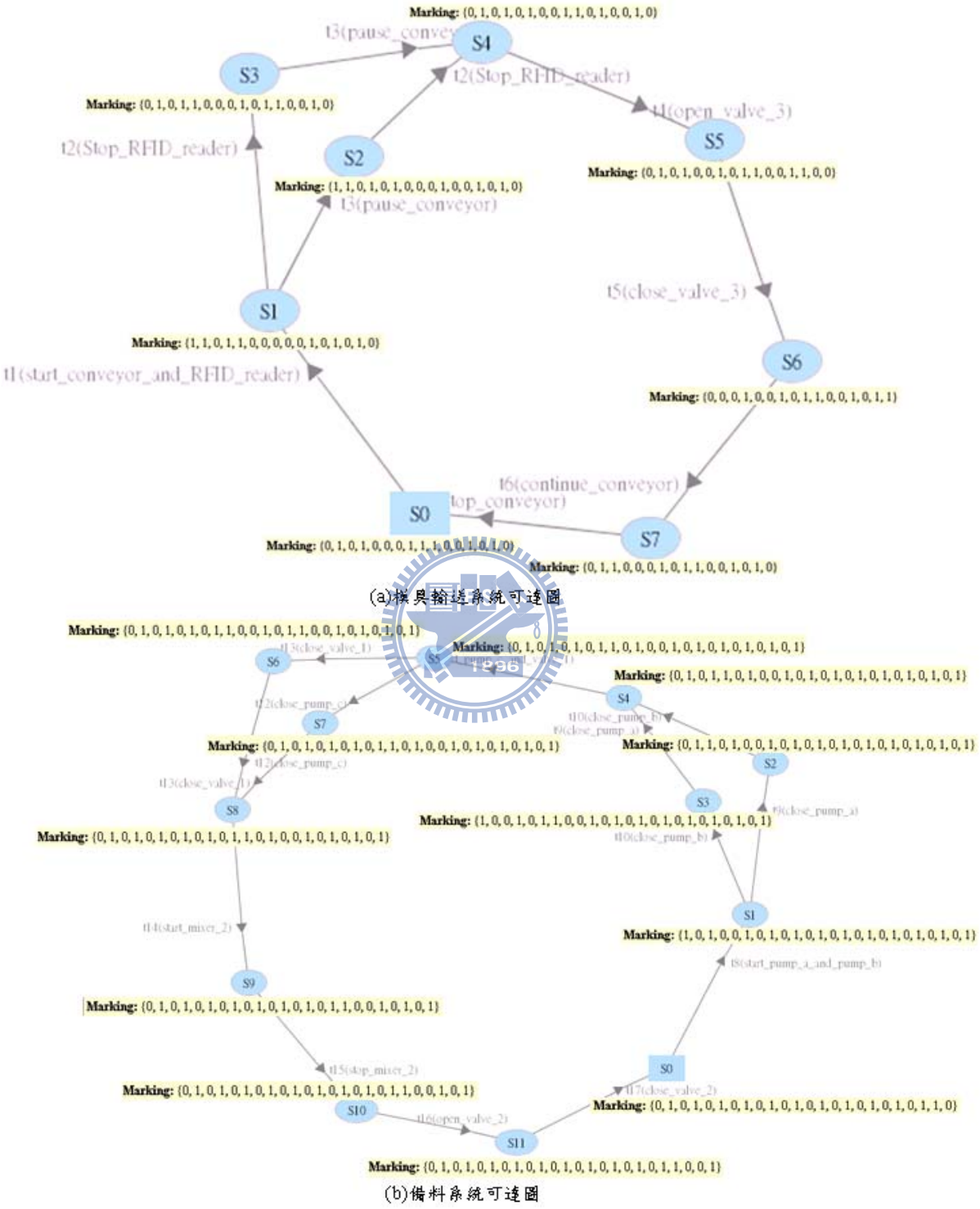


圖 5.9 灌模系統可達狀態圖

對於鎖死分析而言，這是用於驗證系統的設計是否未包含鎖死狀態(Deadlock-free)。當系統運行時，一但進入鎖死狀態後就不會再脫離該狀態，換句話說系統會一直停滯在現行的狀態裡而導致後續的作業程式無法進行，也無法返回初始狀態，這是一種系統設計上的

錯誤，在製造系統設計中應避免鎖死狀態的發生。這裡主要是觀察灌模系統的可達狀態圖裡是否每一個狀態都存在一條離去路徑，不存在離去路徑的系統狀態即代表鎖死狀態。在可達狀態圖中，模具輸送系統包含 8 種可達狀態、備料系統包含 14 種可達狀態，兩個系統中都未包含鎖死狀態。模具輸送系統、備料系統可達狀態圖如圖 5.9 所示。

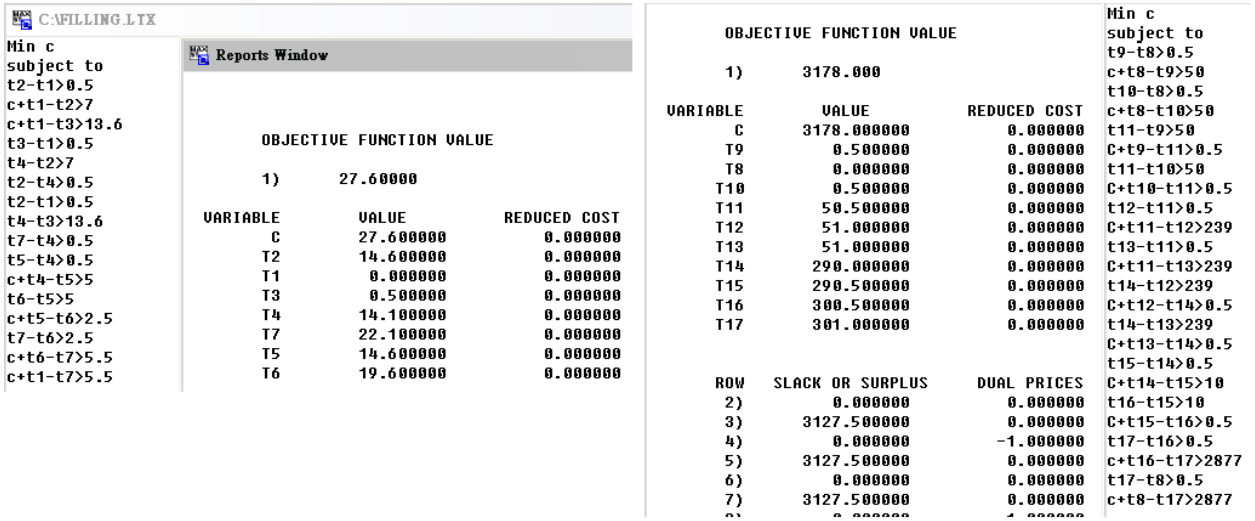


圖 5.10 灌模系統最短週期時間分析

對於最短週期時間分析而言，這是分析裴氏圖自初始標記狀態經系統不斷演化後，再次返回初始標記狀態所需花費的最短時間。藉由瞭解每一批次的生產週期時間，有助於系統生產排程的規劃。有關裴氏圖最短週期時間的計算，這裡是藉由線性規劃方法進行求解，詳細的線性規劃模式建立方式可參考[5]。模具輸送系統的最短週期時間的計算結果為 27.6 秒，如圖 5.10(a)所示，這與實際測量結果 27.4 秒相差僅 0.2 秒。備料系統的最短週期時間計算結果為 3178，如圖 5.10(b)所示，這與實際測量結果 3186 秒相差 8 秒，推測造成這樣的差距的原因可能在於液態原料的流速不固定，所以備料系統在進行每回合作業時會因流速不同而造成備料時間上的些微差距。模式中各參數的設定則可參考表 5.2、5.3。

5.3 灌模系統的偵查員與維修員設計

不變量分析是指找出裴氏圖中的轉移點不變量(T-invariant)及暫存點不變量(P-invariant)，有關不變量的求解方式可參考[5]。轉移點不變量代表的是從系統初始狀態出發後的一組轉移點激發順序，經由這組激發順序能夠使系統再次返回初始狀態。從轉移點不變量當中可以瞭解到系統的循環(Cyclical)行為，以及一個作業循環裡每一個轉移點可能被激發的次數。灌模系統轉移點不變量整理如表 5.4。

表 5.4 灌模系統裴氏圖轉移點不變量

模具輸送系統轉移點不變量	備料系統轉移點不變量
t_1(start_conveyor_and_RFID_reader), t_2(stop_RFID_reader), t_3(pause_conveyor), t_4(open_valve_3), t_5(close_valve_3), t_6(continue_conveyor), t_7(stop_conveyor)	t_8(open_pump_a_pump_b), t_9(close_pump_a), t_{10}(close_pump_b), t_{11}(open_pump_c_valve_1), t_{12}(close_pump_c), t_{13}(close_valve_1), t_{14}(start_mixer_2), t_{15}(stop_mixer_2), t_{16}(open_valve_2), t_{17}(close_valve_2),

表 5.5 灌模系統裴氏圖暫存點不變量

	模具輸送系統暫存點不變量	備料系統暫存點不變量
屬性方程式	P_{11}(conveyor_on_2) + P_{12}(conveyor_off_2) = 1 P_{13}(mould_in) + P_{14}(mould_ready) + P_{15}(mould_filled) + P_{16}(mould_leave) = 1 P_1(RFID_reader_on) + P_2(RFID_reader_off) = 1 P_3(conveyor_off_1) + P_4(conveyor_on_1) = 1 P_5(indication_sended) + P_6(indication_received) = 1 P_7(valve_3_open) + P_8(valve_3_closed) = 1 P_{10}(fill_process_ready) + P_9(fill_process_complete) = 1	p17(pump_a_open) + p18(pump_a_closed) = 1 p19(pump_b_open) + p20(pump_b_closed) = 1 p21(A_filled) + p22(A_lacked) = 1 p23(B_filled) + p24(B_lacked) = 1 p25(pump_c_open) + p26(pump_c_closed) = 1 p27(valve_1_open) + p28(valve_1_closed) = 1 p29(H_filled) + p30(H_lacked) = 1 p31(Y_filled) + p32(Y_lacked) = 1 p33(mixer_2_open) + p34(mixer_2_closed) = 1 p35(x_generated) + p36(x_outputed) = 1 p37(valve_2_open) + p38(valve_2_closed) = 1
偵查法則	P_{13}(mould_in) - P_4(conveyor_on_1) = 0 P_1(RFID_reader_on) - P_{13}(mould_in) - P_{14}(mould_ready) + P_5(indication_sended) = 0 P_{10}(fill_process_ready) - P_{11}(conveyor_on_2) + P_{15}(mould_filled) - P_7(valve_3_open) = 1	p17(pump_a_open) - p19(pump_b_open) + p21(A_filled) - p23(B_filled) = 0 p25(pump_c_open) - p27(valve_1_open) + p29(H_filled) - p31(Y_filled) = 0 p17(pump_a_open) + p21(A_filled) + p25(pump_c_open) + p29(H_filled) + p33(mixer_2_open) + p35(x_generated) + p37(valve_2_open) + p39(process_complete) = 1

暫存點不變量指的是由暫存點所構成的一組聯立方程式。這組方程式在系統正常浮標

狀態演進下會具有浮標數守恆的特性，藉由觀察暫存點不變量是否維持守恆可以得知系統是否保持在可能的正常狀態裡，有關灌模系統偵查員的設計即是藉由觀察系統暫存點不變量是否守恆來判斷系統運作狀況是否正常。模具輸送系統包含 10 條暫存點不變量，其中有 7 條為屬性方程式、3 條為偵查法則；備料系統包含 14 條暫存點不變量，其中有 11 條屬性方程式、3 條為偵查法則。灌模系統暫存點不變量整理如表 5.5。

在維修員方面，當暫存點不變量遭到破壞後，維修員軟體會依造被破壞的暫存點不變量編號以及事前所定義的故障診斷計畫分析可能的故障原因，並且發出警報通知現場作業人員進行修復或是主動執行故障排除作業。在灌模系統維修員中，這裡針對 RFID 標籤讀取異常情形定義了故障診斷與排除計畫，其計畫內容是指當不變量 P_1(RFID_reader_on) + P_2(RFID_reader_off) = 1 遭到破壞時，會先停止輸送帶的正向移動以及啟動輸送帶的反向移動功能。接著，當二號紅外線感知器偵測到容器已就定位時停止輸送帶的反向移動，這時模具會定位於 RFID 標籤讀取器之前，而系統會再啟動 RFID 讀取器並且重新讀取 RFID 標籤內存放的資訊。最後，當 RFID 標籤內的資訊順利被讀取出之後，則是再讓輸送帶回復正向移動並結束故障排除作業。而針對其餘未定義故障診斷與排除計畫的異常情形發生時，灌模系統則是會自行暫停所有的作業流程並且發出警報通知作業人員進行維修。有關維修員的作業示範可參考附錄 DVD 中的示範影片。

5.4 灌模系統可程式邏輯控制器的驅動程式設計

在控制器驅動程式的製作方面，這是要針對 3.4 節制定出的驅動程式標準介面裡所需的四個方法(1)openPLCConnection、(2)closePLCConnection、(3)Handle 與(4)getPLCInputState 各別進行設計。

灌模系統的可程式邏輯控制器 SoftPLC 提供使用者一組操作控制器的 SoftPLC API 檔，在 API 檔案裡包含有程式語法 RemoteDataTable 可以用來建立一個與遠端控制器連線的物件。此語法需要輸入三項參數：localInterface、remoteInterface、timeout，分別是指本地網路位置、遠端控制器網路位置以及等待連線時間，在等待時間內尚未與遠端控制器連線成功的話，則系統會視為是連線失敗。

在 SoftPLC 控制器裡，輸入、輸出模組的狀態資訊會各以一組二進位數值紀錄在記憶體裡。這個二進位數值由右至左順序上的字元分別代表著輸入或輸出模組上零號節點到最終一號節點的狀態。當字元值為 1 時，這代表該節點是導通的；當字元值為 0 時，這代表該節點是斷路的。連線成功以後，這可以使用 SoftPLC API 檔裡面 RemoteDataTable 物件的 handle 方法，藉由輸入 file、element、word 三項參數來取得控制器內部資料的存放位置。當 file 參數為 0 時，這代表的是輸出模組資料的存放位置；當 file 參數為 1 時，這代表的是輸入模組資料的存放位置。

獲得輸入、輸出模組資料的存放位置後，接著可以再使用 RemoteDataTable 物件的 getInt 將資料值從指定的存放位置內取出，並且轉譯成整數型態。或者使用 putBits 方法去更改該資料存放位址內所儲存的值。putBits 方法在使用時，會先將所輸入的 andMask 值與該資料存放位址裡的值進行 AND 邏輯運算，接著再將 AND 邏輯運算結果與所輸入的 orMask 值進行 OR 邏輯運算。如此一來能夠產生出一組新的二元數值存放在該資料位置內，而控制器會依據新的資料值去進行相對應的輸出行為。舉例來說，假如現階段 PLC 的 16 個輸出節點狀態為 1101 1001 0111 1001，這表示節點 0、1、3、4、7、9、10、11、12、15 為導通狀態，其餘為斷路狀態。現在若要將節點 2、6、13 更改為導通狀態而其餘節點則維持前一個階段的狀態，這可以輸入 andMask 為 1111 1111 1111 1111，orMask 為 0010 0011 0000 0101。這時 PLC 的輸出節點狀態會更新為 1111 1011 0111 1101，這表示節點 0、1、2、3、4、6、7、9、10、11、12、13、15 為導通狀態、其餘為斷路狀態，計算流程如圖 5.11 所示。

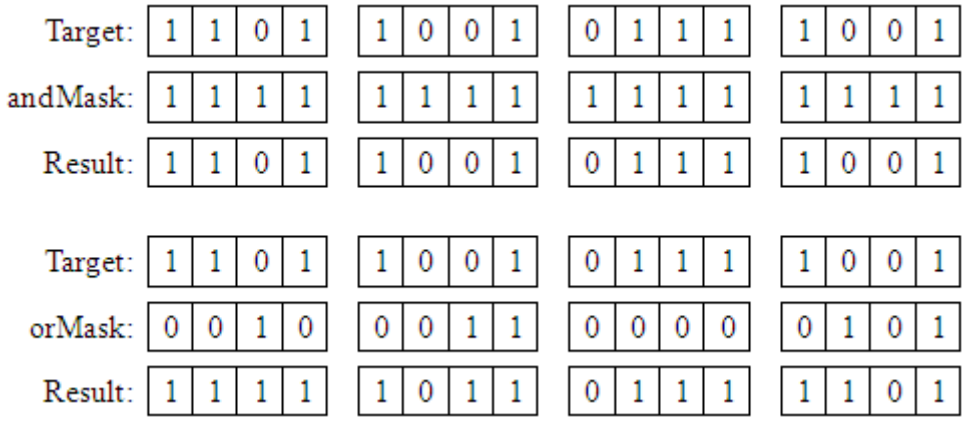


圖 5.11 putBits 方法運算流程

最後，欲結束與 SoftPLC 控制器之間的連線時，可以使用 RemoteDataTable 物件的 finalize 方法來中斷連線。根據 SoftPLC API 所提供的控制器操作語法設計出的驅動程式如圖 5.12 所示。

```
public class SoftPLCDriver {
    RemoteDataTable RDT; //softPLC連線物件
    int inputNodeState;  //以整數型別紀錄控制器輸入模組狀態資訊
    Long inHandle,outHandle;  //儲存輸入,輸出資料儲存位置

    public boolean openPLCConnection(String ip,String port){
        try {
            RDT = new RemoteDataTable(null,java.net.InetAddress.getByName(ip),5000);
                //初始化連線物件
            inHandle = RDT.handle(1, 0, 0);  //取得輸入資料存放位置
            outHandle = RDT.handle(0, 0, 0); //取得輸出資料存放位置

        } catch (UnknownHostException e) {
            return false;                    //錯誤的ip位置回傳false
        } catch (IOException e) {
            return false;                    //連線失敗回傳false
        }
        return true;                         //連線成功回傳true

    }

    public boolean closePLCConnection(){
        RDT.finalize();                      //關閉PLC連線

        return true;
    }

    public boolean Handle(String cmd){

        if (cmd.equals("GET")) {
            //指令為GET使用getInt()方法取得輸入模組狀態資料
            inputNodeState = RDT.getInt(inHandle);
            return true;
        }
        else if(cmd.startsWith("ON")){
            //指令為ON時,擷取ON指令後方的二進位數值更改控制器輸出模組狀態
            int node = Integer.parseInt(cmd.substring(2));
            RDT.putBits(outHandle, (0xFFFF), node);
            return true;
        }
        else if(cmd.startsWith("OFF")){
            //指令為OFF時,擷取OFF指令後方的二進位數值更改控制器輸出模組狀態
            int node = Integer.parseInt(cmd.substring(3));
            RDT.putBits(outHandle, (0xFFFF - node), 0x0);
            return true;
        }
        return false;                        //指令錯誤回傳false

    }

    public String getPLCInputState(){
        return Integer.toBinaryString(inputNodeState);    //以二進位形式回傳輸入模組狀態
    }

}
```

圖 5.12 SoftPLC 驅動程式

對於 openPLCConnection 方法而言，這是建立一個 RemoteDataTable 物件與遠端控制器進行連線，其中在執行 RemoteDataTable 物件的初始化時，會將第一個參數設為 null 這

代表本地端使用預設的網路位置，第二個參數設為使用者所傳入的控制器網路位置，第三個參數設定為 5000 代表連線時間超過 5 秒鐘後，則視為連線失敗。此外，並使用 RemoteDataTable 物件的 Handle 方法取得輸入、輸出模組資訊的存放位置。

對於 closePLCConnection 方法而言，這是呼叫 RemoteDataTable 物件的 finalize 方法結束與遠端控制器的連線。

對於 Handle 方法而言，這首先會判斷使用者所傳送的指令 cmd 其字首為 ON、OFF 或者是 GET。當字首為 GET 時呼叫 RemoteDataTable 物件的 getInt 方法取得輸入模組資訊。當字首為 ON 或 OFF 時，則擷取 ON、OFF 後方的二進位數值並呼叫 RemoteDataTable 物件的 putBits 方法修改控制器的輸出模組資訊。

對於 getPLCInputStat 方法而言，這是使用整數資料的 toBinaryString 方法回傳一個二進位數值的字串，其代表著控制器輸出節點的狀態。

5.5 灌模系統裴氏圖同步性設定

經驗證過的裴氏圖，可以使用 PNML 導向資料伺服器連接軟體進行該裴氏圖的同步性設定。在同步性設定作業中，首先要設定欲使用的控制器輸出、輸入節點數量。根據表 5.1 可以得知在輸入節點方面總共會使用到 0 號到 15 號的 16 個節點。輸出節點方面，這會使用到 0 號到 9 號的 10 個節點，所以這裡設定所需的控制器節點數量為 16，如圖 5.13 所示。

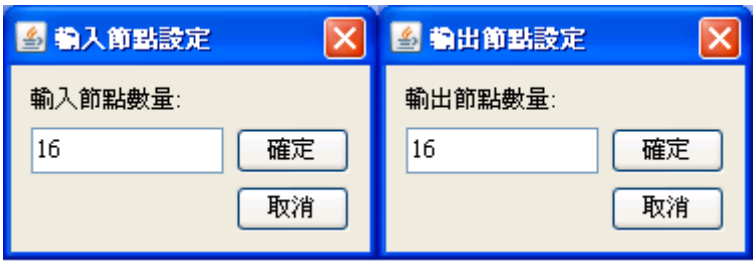


圖 5.13 控制器節點數量設定

其次是設定欲使用到的虛擬變數，根據表 5.2、5.3 可以得知在模具輸送系統方面，會使用到虛擬變數 Reader、Switch、Indication、Process_1、Mold 五個。備料系統方面，會使用到虛擬變數 M_a、M_b、M_h、M_y、M_x、Process_2 六個。十六個虛擬變數的資料型別皆設定為字串型別，其設定如圖 5.14 所示。

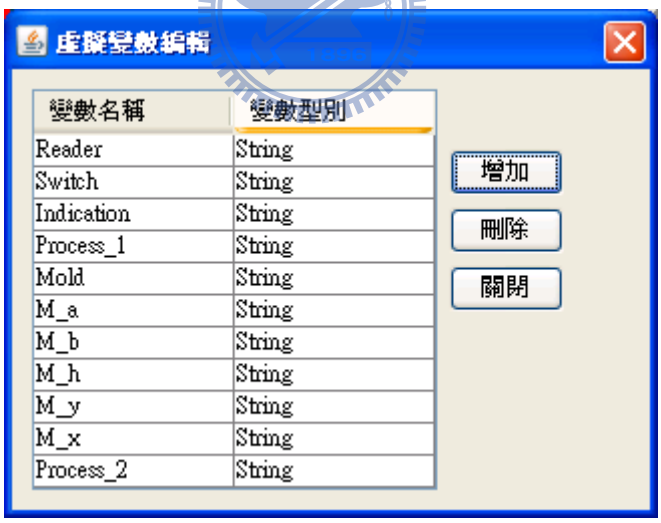


圖 5.14 虛擬變數設定

依據表 5.2、5.3 可設定暫存點、轉移點的同步性。在暫存點同步性設定方面，這裡以灌模系統的暫存點 P_1、P_3 為範例說明設定方式。圖 5.15 顯示的是暫存點 P_1 的同步性設定。暫存點 P_1 表達的是 RFID 標籤讀取器已開啟狀態，它的同步性設定是指當虛擬變數 Reader 值為 on 時，暫存點 P_1 內會有浮標。所以這必須在暫存點 P_1 同步性設定表的[節點編號/虛擬變數]表單中選擇虛擬變數 Reader，在[運算子]選單中選擇.equal(String)進行字串的比較，在值欄位裡輸入 on 完成設定。

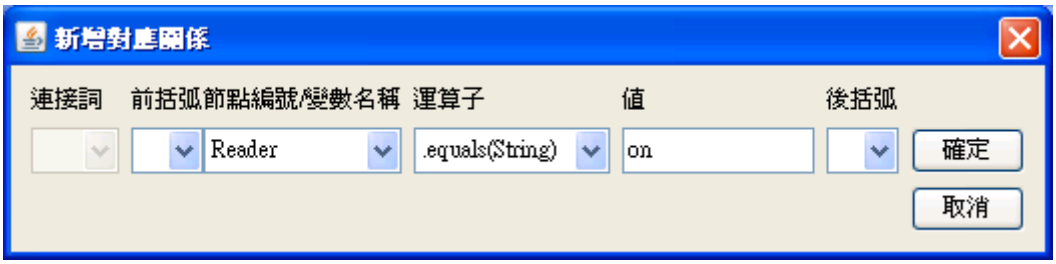


圖 5.15 暫存點 P_1 同步性設定

暫存點 P_3 表達的是輸送帶已停止(1)狀態，它的同步性設定是指當輸入節點 I_{11} 為導通狀態或者是虛擬變數 Switch 值為 2 時，暫存點 P_3 內會有浮標。所以這必須設定兩條暫存點 P_3 的同步性設定，首先是在同步性表的[節點編號/虛擬變數]表單中選擇節點 11 斷路狀態即完成第一條設定，接著再創造另一條同步性設定，並且在[連接詞]表單中選擇||代表 OR 邏輯運算，在[節點編號/虛擬變數]表單中選擇虛擬變數 Switch，在[運運算元]選單中選擇.equal(String)進行字串的比較，在值欄位裡輸入 2 完成第二條設定，如圖 5.16 所示。

圖 5.16 暫存點 P_3 同步性設定

在灌模系統轉移點同步性設定方面，這裡以灌模系統的轉移點 T_1、T_5 為範例說明設定方式，圖 5.17 顯示的是轉移點 T_1 的同步性設定。轉移點 T_1 代表啟動輸送帶及 RFID 標籤讀取器作業，它的同步性設定是指當輸入節點 I_{12}-為導通狀態時，設定輸出節點 O-$_0$ 為導通狀態以及設定虛擬變數 Mold 值為 in、Reader 值為 on，延遲執行時間為 0。所以這必須在轉移點 T_1 的激發事件同步性設定表的[節點編號/虛擬變數]表單中選擇節點 12 導通狀態，並且在元件作業同步性設定表裡設定三項作業。第一項作業是在[節點編號/虛擬變數]表單中選擇節點 0 導通狀態完成設定，第二項作業是在[節點編號/虛擬變數]表單中選擇虛擬變數 Mold，在值欄位裡輸入 in 完成設定，第三項作業是在[節點編號/虛擬變數]表單中選擇虛擬變數 Reader，在值欄位裡輸入 on 完成設定。最後，設定延遲激發時間為 0。

圖 5.18 顯示的是轉移點 T_5 的同步性設定。轉移點 T_5 代表繼續啟動輸送帶作業，它的同步性設定是指設定輸出節點 O_8 為導通狀態以及設定虛擬變數 Process_1 值為 complete，並且不需要等待任何觸發事件，延遲執行時間為 2000 毫秒。所以在轉移點 T_5 的激發事件同步性設定表裡不需要作設定，在元件作業同步性設定表裡需要設定兩項作業。第一項作業是在[節點編號/虛擬變數]表單中選擇節點 8 導通狀態完成設定，第二項作業是在[節點編號/虛擬變數]表單中選擇虛擬變數 Process_1，在值欄位裡輸入 complete 完成設定。最後，設定延遲激發時間為 2000 毫秒。

圖 5.17 轉移點 T_1 同步性設定

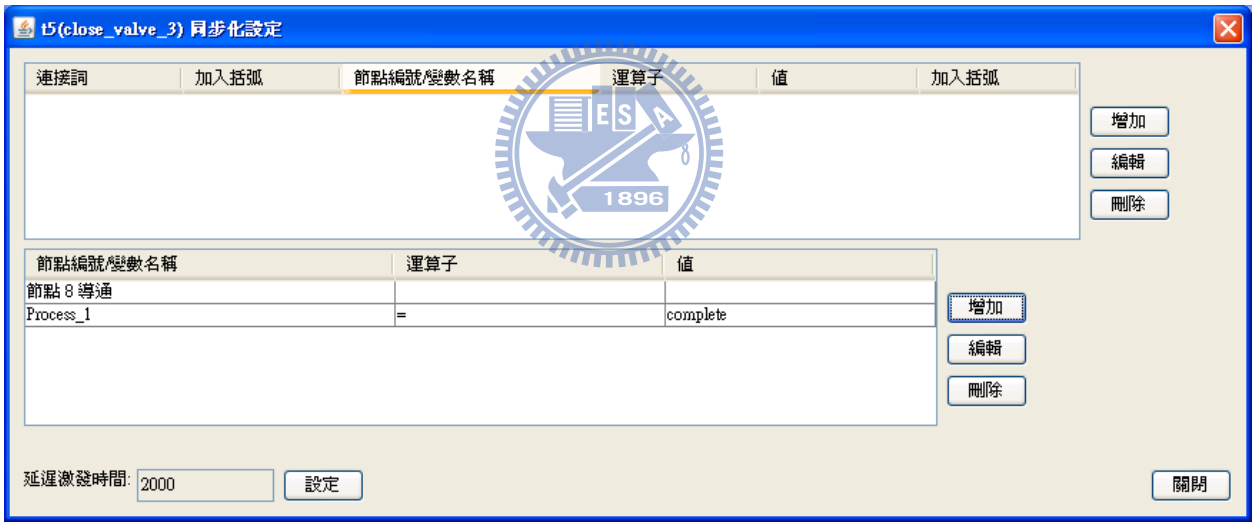


圖 5.18 轉移點 T_5 同步性設定

此外，在灌模系統的轉移點 T_2 方面，其主要的元件作業是呼叫 IF5 RFID 讀取器內部的標籤資訊回傳指令取得存放在模具所裝置 RFID 標籤內的作業指令參數，並且在讀取完畢之後停止 IF5 RFID 讀取器的讀取作業。但由於 IF5 RFID 讀取器並無法透過灌模系統的可邏輯控制器與其進行溝通以及進行指令操作，所以在轉移點 T_2 的同步性設定上，是在 PNML 導向資料伺服器連接軟體所輸出的資料伺服軟體裡，針對轉移點 T_2 進行程式碼的修改，使其能夠與 IF5 RFID 讀取器進行連接並執行標籤資訊讀取作業。

當裴氏圖同步性設定完成之後，即可以點選檔案匯出功能由軟體自動產生相對應的資料伺服器軟體，並且選擇欲搭配的 SoftPLC 控制器驅動程式進行決策系統、資料伺服器軟體以及驅動程式三者的整合，最終輸出一個簡易的灌模系統自動化監控軟體。圖 5.19 為驅動程式設定畫面，圖 5.20 為灌模系統自動化監控軟體的操作介面。

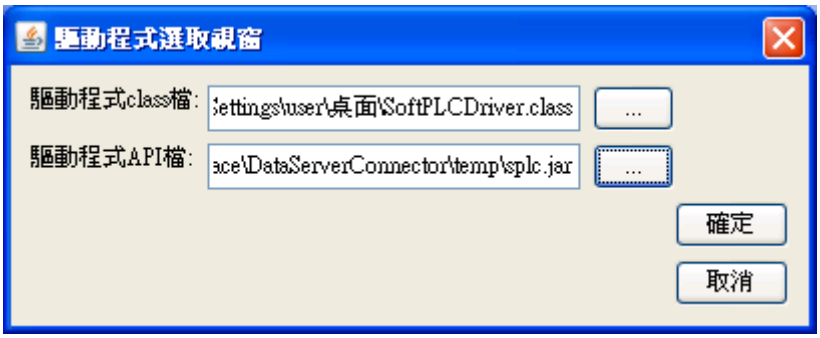


圖 5.19 驅動程式設定

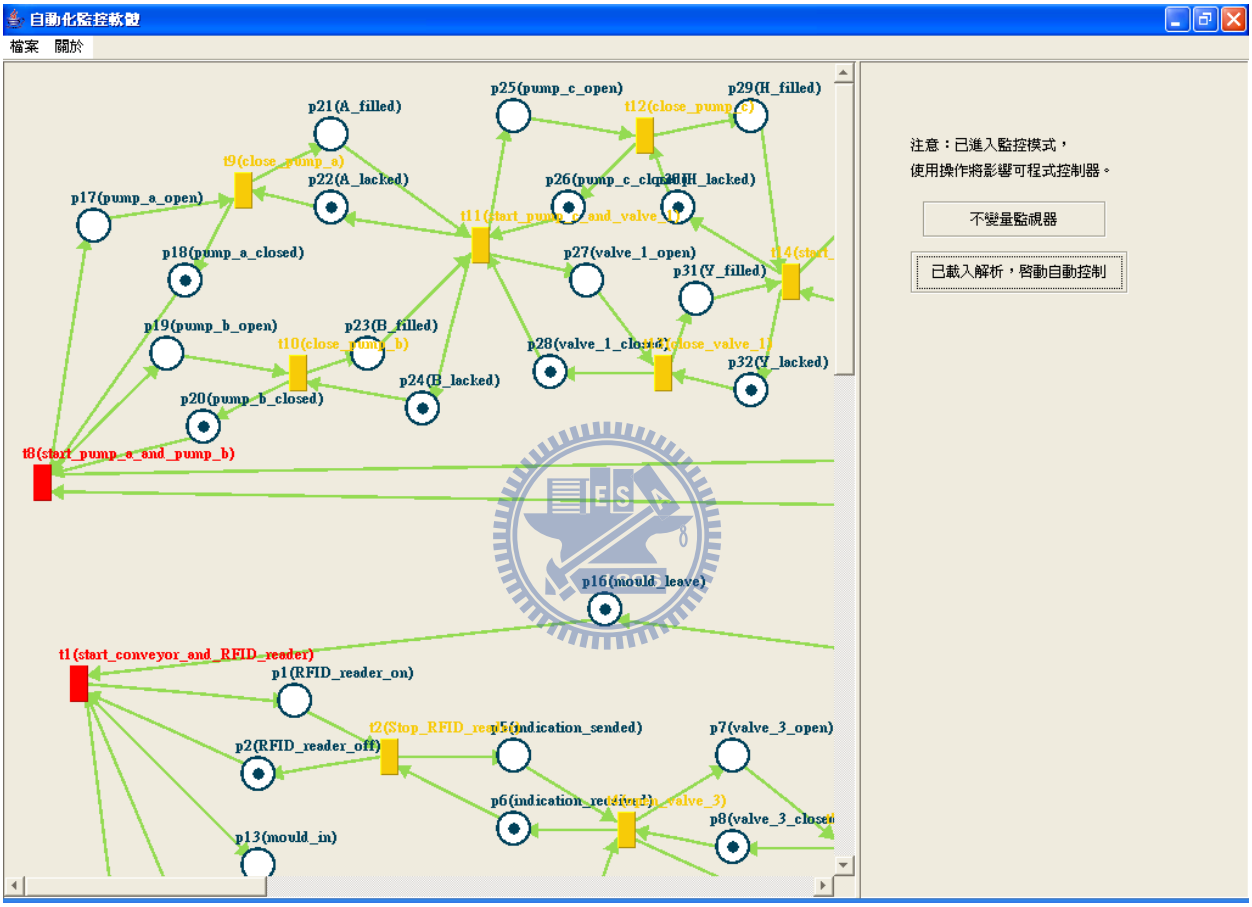


圖 5.20 灌模系統監控軟體

第六章 結論與未來研究方向

本研究主要是提出一個開放資料伺服器連接架構的規範，其目的在於將監控軟體與可程式邏輯控制器之間的通訊介面規格標準化，標準化的 ODSC 可以提升監控軟體的通用性，讓所開發出的監控軟體能夠廣泛的使用在各種類型、品牌的可程式邏輯控制器中。

在開放資料伺服器連接架構的設計過程中，這是藉由模組化的方式將監控軟體內部的控制邏輯、控制邏輯指令轉譯、控制器指令語法轉譯三種功能劃分成決策系統、資料伺服介面與驅動程式三個元件，再標準化各元件之間的溝通介面規格。由於各元件的功能性獨立並且彼此擁有標準介面可以進行溝通，所以每個元件都能夠被獨立開發，元件之間也會易於整合。這樣的設計將有利於團隊開發。此外，當單一元件存在某些錯誤需要修改某時，不會牽連到其他的模組的功能，維護起來也會比較方便。

開放資料伺服器連接架構與傳統資料伺服器連接架構的主要差異在於前者強調了軟、硬體之間的互用性；這是單一監控軟體能夠在不同廠牌、類型的可程式邏輯控制器之間交互使用的能力。在以往監控軟體開發過程中，軟體開發人員主要是針對特定一種控制器規格進行其資料伺服器軟體的設計，所以當面臨執行環境轉移、或者是軟、硬體升級的問題時，軟體開發人員就必須花費大量時間在重新修改現存的監控軟體使其能夠適應新的執行環境，或是撰寫大量的邏輯條件式，讓監控軟體能夠經由條件判斷來執行特定類型控制器的作業方式。但無論使用哪一種方法，都需要耗費大量的學習成本在於重新瞭解各種控制器所使用的指令語法與資料格式等等資訊，這也顯示出傳統資料伺服器連接架構底下的監控軟體會具有通用性差的缺點。傳統資料伺服器連接架構如圖 6.1

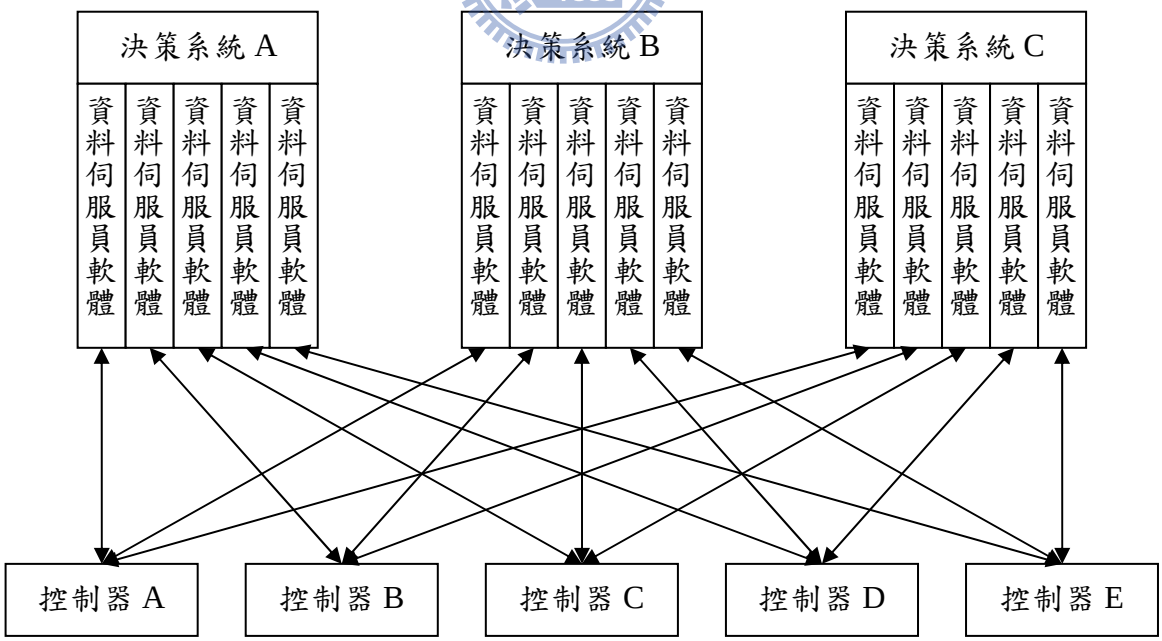


圖 6.1 傳統資料伺服器連接架構

然而，透過本論文提出的決策系統-資料伺服介面-驅動程式的三階段開放連接架構以及所制定的元件規格，這讓控制器廠商能夠依據驅動程式介面規格各自開發其驅動程式提供給使用者使用，而監控軟體開發人員也只需要依循該設計規格進行資料伺服器軟體的開

發，不用擔心最終所使用的控制器類型為何，只要監控軟體開發人員與控制器廠商使用的是相同一套規格標準，所開發出資料伺服器軟體就能藉由搭配不同控制器的驅動程式來與各種類型控制器進行溝通與執行監控作業，如此一來提升了監控軟體的通用性。開放資料伺服器連接架構如圖 6.2 所示。總結使用開放資料伺服器連接架構的優點共可以整理成以下五點：(1)良好的系統移植能力，(2)利於團隊開發，(3)方便軟體維護，(4)提升軟體的重覆使用性，(5)節省軟體開發成本。

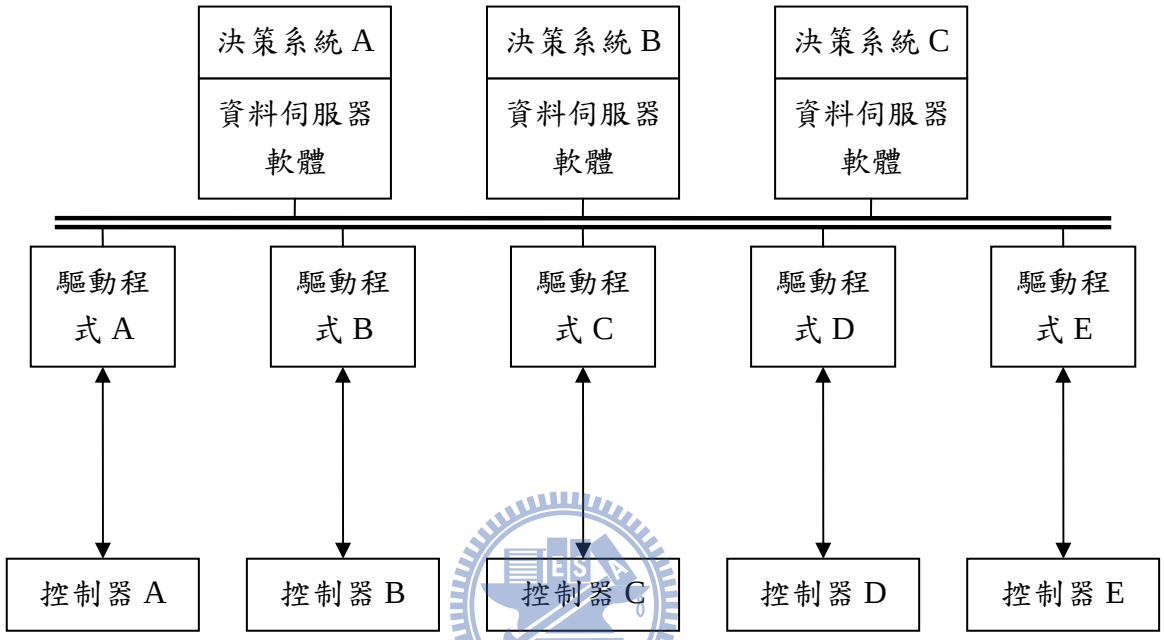


圖 6.2 開放資料伺服器連接架構

在未來研究方向方面，隨著無線射頻識別技術的迅速發展，在自動化工廠中導入無線射頻識別技術的運用議題已廣泛的被提出，這意味著在自動化監控軟體中結合無線射頻標籤資訊的讀寫功能將會成為往後監控軟體開發上的一個研究重點，所以同樣需要思考的是如何在已建立的開放資料伺服器連接架構中設計無線射頻識別讀取器的連接介面的規範，使得所開發出來的監控軟體能夠不受限於特定類型的無線射頻識別讀取器規格。

此外，在本研究中所提出的開放連接架構中，雖然監控軟體可以利用更換驅動程式的方式來與不同類型控制器連接，但是目前仍無法作到一次使用多組驅動程式來與多種不同類型的控制器同時連接的能力。而在自動化工廠中卻可能因為機台的汰舊換新而使用了多種不同類型的控制器，若沒辦法同時使用多組驅動程式來與各個控制器連接，則會導致監控軟體沒辦法監控完整的製造系統流程，所以在多類型控制器同時連接方面，仍有需要作進一步的研究與改進。

參考文獻

[1] 洪信銘，「利用 IDEF/PN/G2 設計方式實作即時型現場監控系統」，國立交通大學工業工程與管理研究所碩士論文，1993 年。

[2] 范植宇，「階層轉換法的自動化：從規格到設計的樹形轉換」，國立交通大學工業工程與管理研究所碩士論文，2009 年。

[3] 梁高榮，「自動化製造系統內的即時故障察覺」，機械工業，五月，127-138 頁，2006 年。

[4] 梁高榮，「虛體製造系統的多緒架構」，機械工業，五月，241-261 頁，2004 年。

[5] 梁高榮，「裴氏圖與記號圖：可程式控制器的分析工具」，機械工業，十月，119-130 頁，2009 年。

[6] 梁高榮，「擬陣理論在現場監控的應用」，機械工業，六月，176-188 頁，2001 年。

[7] 陳宗沂，「網路整合製造系統的資料伺服員設計」，國立交通大學工業工程與管理研究所碩士論文，2005 年。

[8] 陳啟宗，梁高榮，「利用手機控制遠端的製造系統」，機械工業，六月，127-138 頁，2005 年。

[9] 陳音帆，梁高榮，「自動化製造系統裡斐式圖模式的撰碼自動化」，機械工業，十二月，111-121 頁，2006 年。

[10] 陳音帆，「斐氏圖導向控制器開發系統的設計與實作」，國立交通大學工業工程與管理研究所碩士論文，2008 年。

[11] 劉育宏，「利用虛體製造裝置作彈性輸送機的多遠端控制」，國立交通大學工業工程與管理研究所碩士論文，1995 年。

[12] 鄭瑋弘，「以整合型 SMT 方法論設計有效可靠的自動化製造系統」，國立交通大學工業工程與管理研究所博士論文，1997 年。

[13] David, R., “Grafcet: A Powerful Tool for Specification of Logic Controllers,” IEEE Transaction on Control Systems Technology, Vol. 3, No. 3, pp. 253-268, Sept. 1995.

[14] David, R. and Alla, H., Petri Nets & Grafcet: Tools for Modeling Discrete Event Systems, Prentice-Hall, 1992.

[15] Deitel, H. M., Java: How to Program, 5th Ed., Prentice Hall, 2005.

[16] Heinecke, H., Schnelle, K., Fennel, H., Bortolazzi, J., Lundh, L., Leflour, J., Mate, J., Nishikawa, K., and Scharnhorst, T., “Automotive Open System Architecture - An Industry-wide Initiative to Manage the Complexity of Emerging Automotive E/E-architectures,” Proceeding of the 2004 International Congress on Transportation Electronics, Vehicle Electronics to Digital Mobility: The Next Generation of Convergence, pp. 325-332, 2004.

[17] http://java.sun.com/javame/technology/index.jsp, Java ME Technology, Feb. 1, 2010.

[18] http://www.dom4j.org/, dom4j, Feb. 1, 2010.

[19] http://www.gensym.com/, G2 Platform, Feb. 1, 2010.

[20] http://www.intermec.com/learning/technologies/rfid/, Intermec IF5 fixed Reader, Feb. 1, 2010.

[21] http://www.plcopen.org/pages/tc1_standards/iec_61131_3/, IEC 61131-3, Feb. 1, 2010.
[22] http://www.w3.org/, World Wide Web Consortium, Feb. 1, 2010.
[23] http://www.w3.org/XML/, Extensible Markup Language, Feb. 1, 2010.
[24] http://www2.informatik.hu-berlin.de/top/pnml/about.html, Petri Net Markup Language, Feb. 1, 2010.
[25] IDEF0, Federal Information Processing Standards Publication 183, USA, 1993.
[26] Jeng, W. H. and Liang, G. R., "Reliable Automated Manufacturing System Design Based on SMT Framework," Computers in Industry, Vol. 35, No. 2, pp. 121-147, 1998.
[27] Johnson, R. A., Probability and Statistics for Engineers, 7th Ed., Pearson Prentice Hall, 2005.
[28] Kroenke, D. M., Database Processing, 10th Ed., Prentice Hall, 2006.
[29] Geiger, K., Inside ODBC, Microsoft Press, 1995.
[30] Liang, G. R. and Hong H.M., "Hierarchy Transformation Method for Repetitive Manufacturing System Specification, Design, Verification, and Implementation," Computer-Integrated Manufacturing System, Vol. 7, No. 3, pp. 191-205, 1994.
[31] Murata, T., "Petri Nets: Properties, Analysis and Application," Proceedings of the IEEE, Vol. 44, pp. 541-579, 1989.
[32] Zimmermann, H., "OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection," IEEE Transactions on Communications, Vol. 28, No. 4, pp. 425-432, 1980.

附錄一 灌模系統 IDEF0 規格圖設計

IDEF0(Integrated Definition Methods)規格圖[25]是 1970 年代美國空軍在執行電腦整合製造計劃(Integrated Computer-Aided Manufacturing, ICOM)時所發展出的一項圖形建模技術，其主要用於描述系統內部功能與作業的運作架構，該技術並於 1993 年被美國國家標準與技術局列為標準建模方式之一。

IDEF0 模型主要由兩種元件所構成—作業方格及箭號，作業方格是用來表示系統的功能、作業或是動作，並且通常會將動詞置於作業方格內以說明其意義；箭號則用來表示系統的資訊流、物流、限制與機制，當箭號由左方進入作業方格時此箭號代表的是該作業方格的輸入、當箭號由上方進入作業方格時此箭號代表的是該作業方格的控制條件、當箭號由下方進入作業方格時此箭號代表的是該作業方格的作業機制、當箭號由右方離開作業方格時此箭號代表該作業方格的輸出，如圖 A.1 所示。

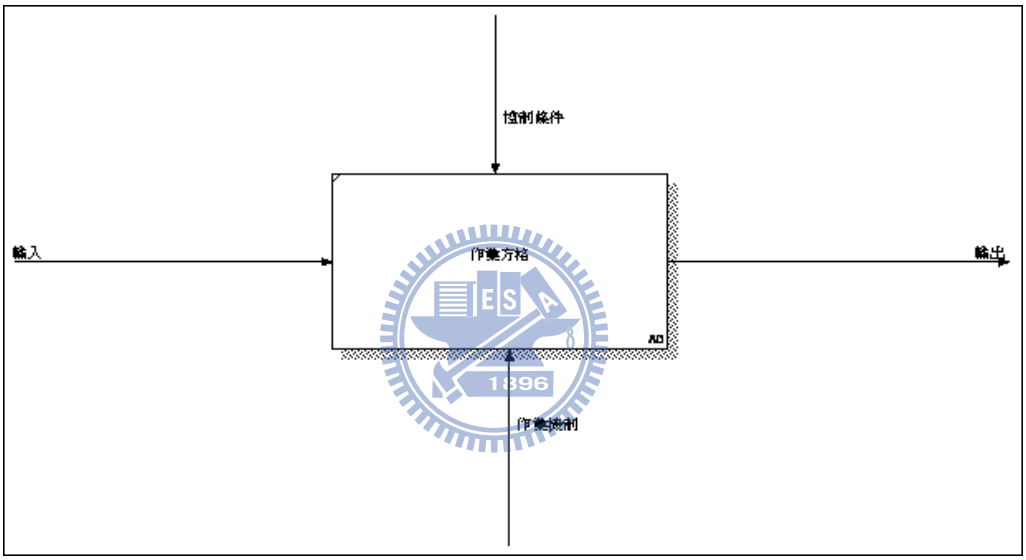


圖 A.1 IDEF0 箭號與作業方格關係圖

此外，IDEF0 模型具有階層性，模型中裡的每個作業方格都可以向下展開出多個子作業方格讓使用者定義每個作業方格更詳細的作業內容，如圖 A.2 所示。

在備料系統的執行流程方面，圖 A.2 顯示備料系統一共包含三種物料輸入即物料 A、物料 B 與物料 H，這三種物料經系統加工處理後會輸出物料 X。圖 A.3 說明備料系統內部的細部作業流程，圖中的粗線部分代表的是系統的物流、細線代表的是系統的資訊流，從物料 A、B、H 輸入至系統到最終系統輸出物料 X 之間的流程共可分成三道程式—調製物料 Y、調製物料 X 以及儲存物料 X。在物流方面，系統會使用物料 A 及物料 B 進行物料 Y 的調製程式，接著系統再使用已調製好的物料 Y 以及物料 H 進行物料 X 的調製程式，最後是儲存已調製好的物料 X。

在資訊流方面，系統會在三號儲存槽的液位到達最低水位時啟動物料 Y 的調製程式，並且在調製程式結束時輸出訊息來啟動下一階段的調製物料 X 程式，調製物料 X 程式結束後除了會輸出訊息來啟動下一階段的儲存物料 X 程式之外亦會輸出回饋訊息至上一階段的調製物料 Y 程式，當儲存物料 X 程式執行結束後會輸出回饋訊息至前兩階段的程式。由圖

中可發現，資訊流在系統中會存在兩種方向，流向後續作業方格的資訊流主要是在於啟動系統的後續程式，而流向先前作業方格的資訊流則是為了將作業完成的資訊告知先前程式以避免前後互相衝突的程式同時發生的情形。

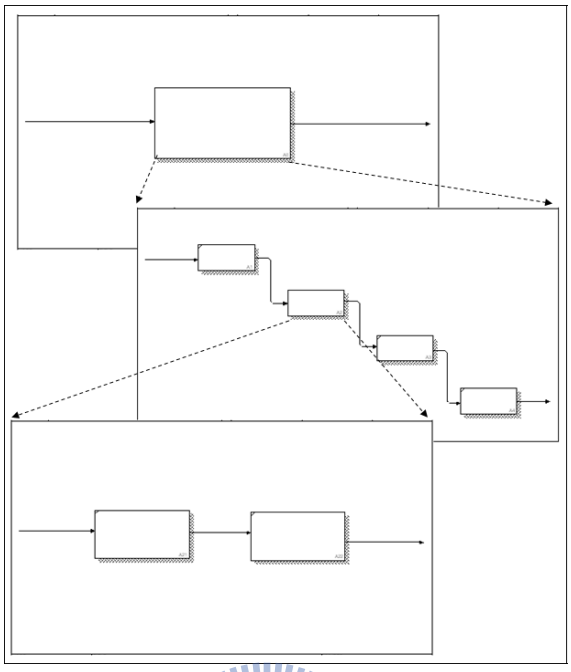


圖 A.2 IDEF0 作業方格階層性示意圖

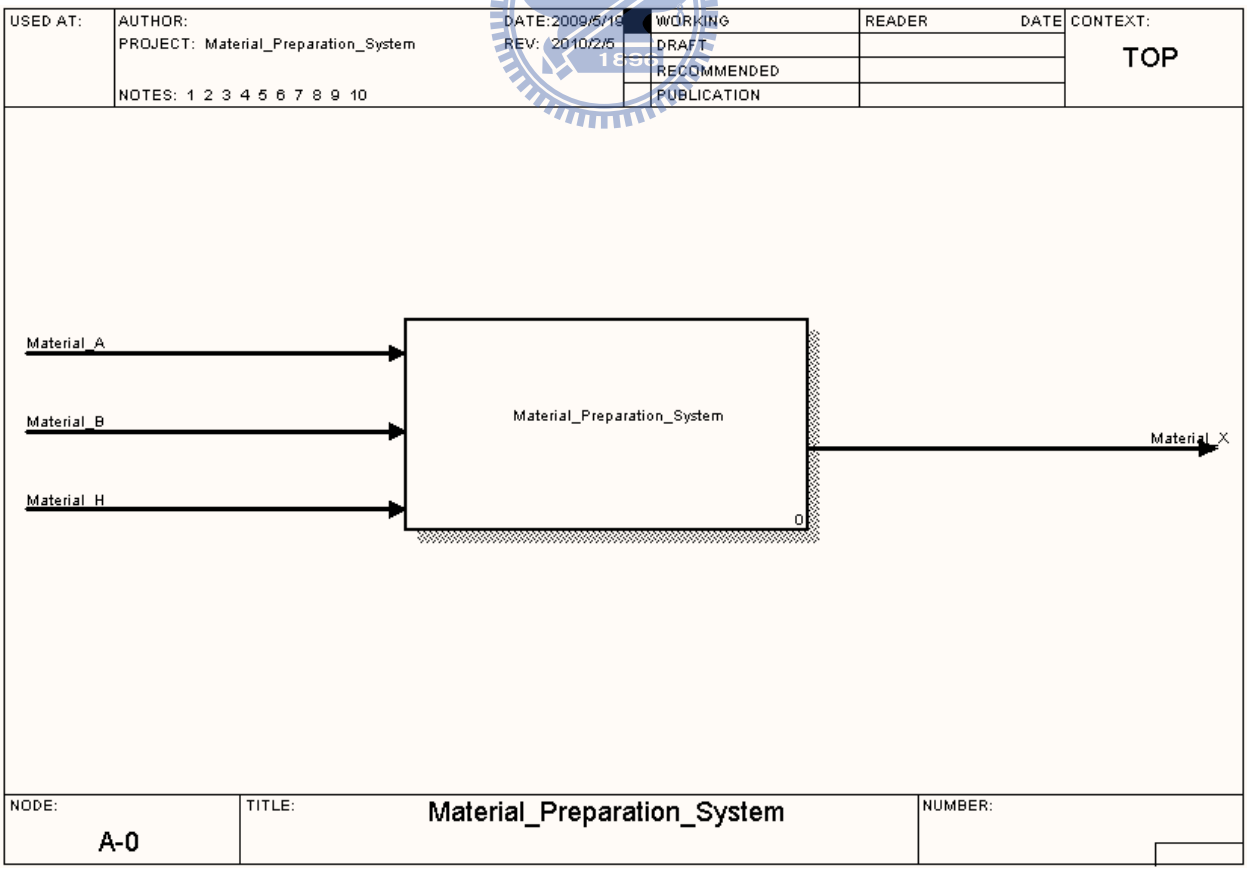


圖 A.3 備料系統 A-0 層 IDEF0 規格圖

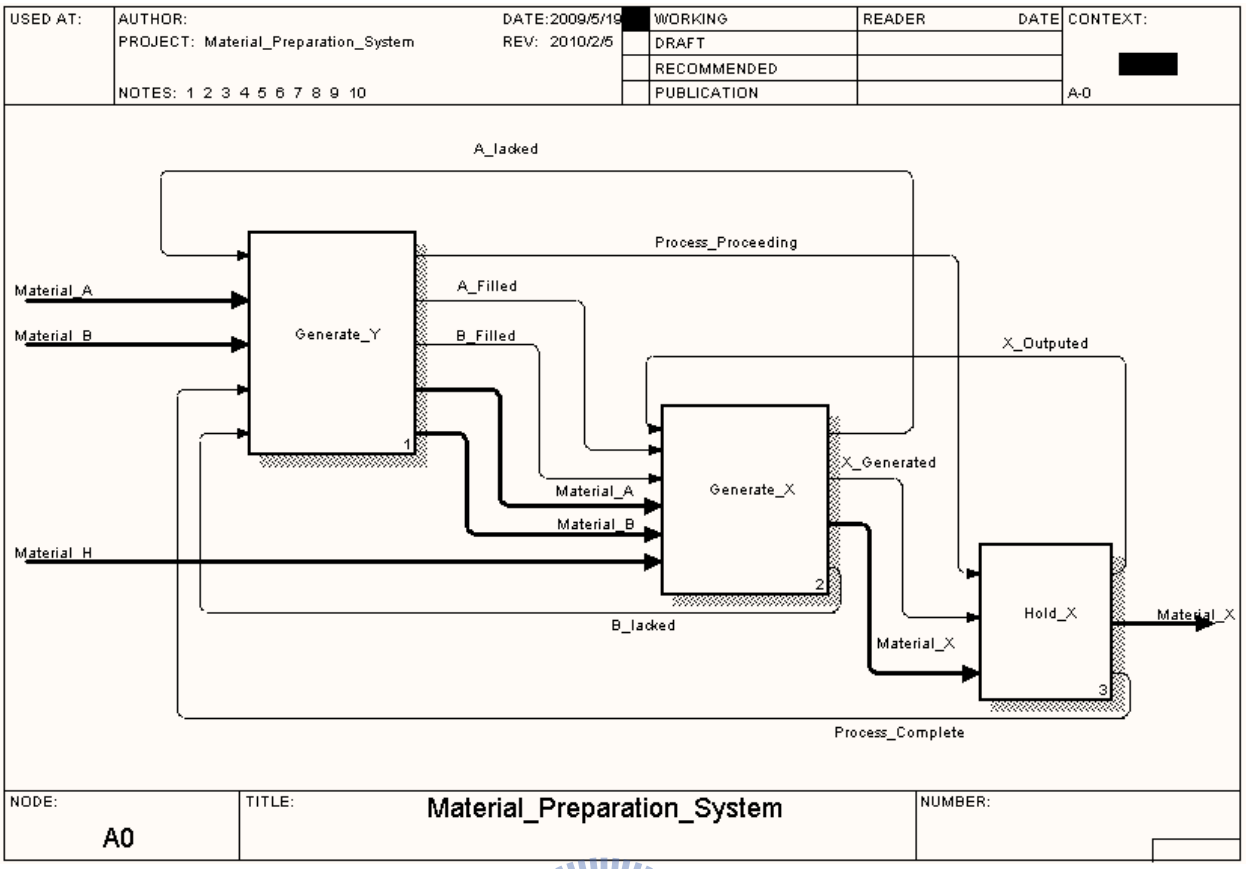


圖 A.4 備料系統 A0 層 IDEF0 圖

圖 A.5、A.6 及 A.7 分別為物料 Y 調製程式、調製物料 X 程式、保存物料 X 程式的細部作業程式。對於物料 Y 調製程式而言，系統會在三號儲存槽的液位到達最低水位時啟動幫浦 A 及幫浦 B 來將物料 A 與物料 B 導入一號儲存槽內，當物料 A 流量大於 60 公升後即關閉幫浦 A、當物料 B 流量大於 30 公升後即關閉幫浦 B；對於調製物料 X 程式而言，系統會先啟動幫浦 C 與一號閥門來將物料 H 及物料 Y 導入二號儲存槽，當物料 H 流量大於 10 公升後即關閉幫浦 C、當物料 Y 流量大於 30 公升後即關閉一號閥門，接著系統會啟動二號攪拌器進行持續 5 分鐘的攪拌作業；對於保存物料 X 程式而言，系統會開啟二號閥門來讓物料 X 流入三號儲存槽直到三號儲存槽容量高於最高水位時再關閉二號閥門。

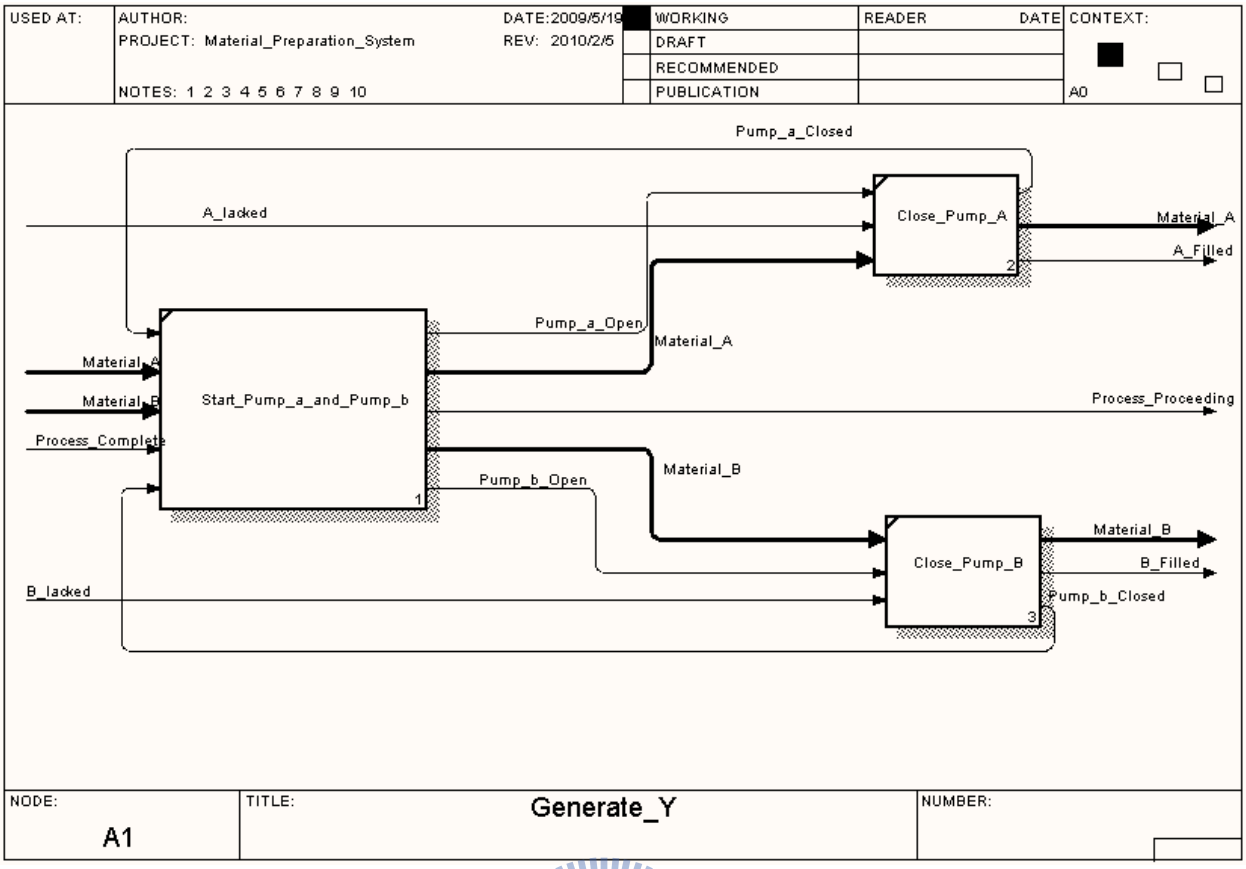


圖 A.5 備料系統 A1 層 IDEF0 圖

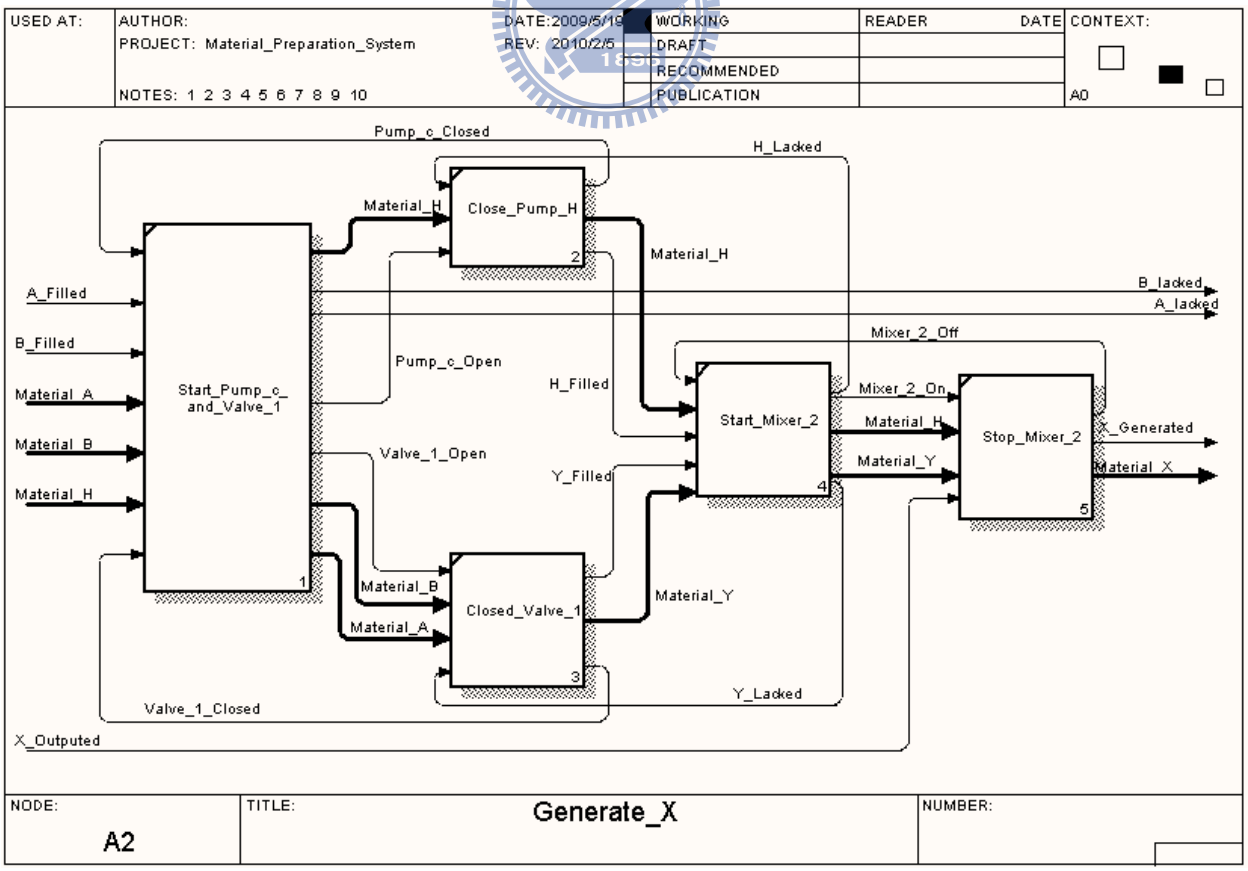


圖 A.6 備料系統 A2 層 IDEF0 圖

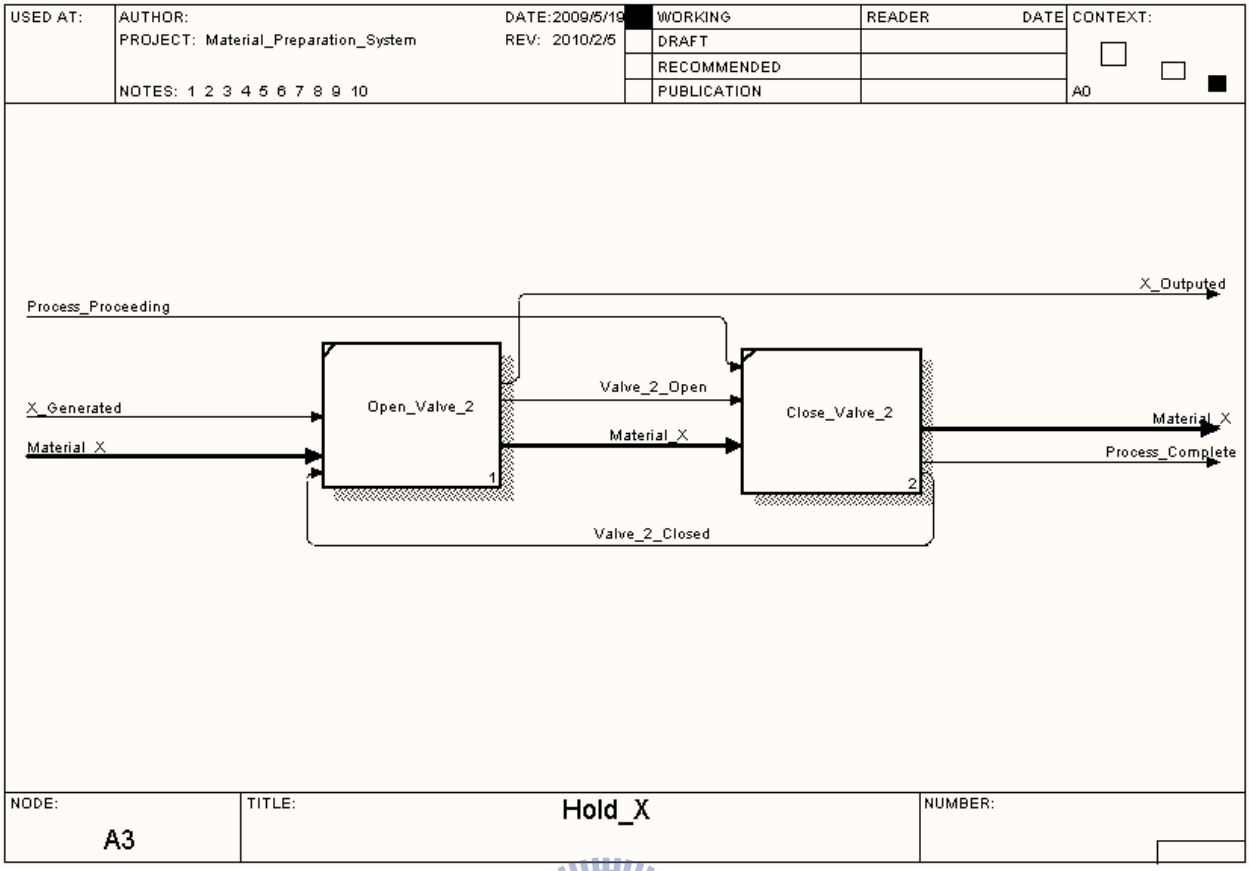


圖 A.7 備料系統 A3 層 IDEF0 圖

在模具輸送系統的執行流程方面，圖 A.8 顯示模具輸送系統所需輸入為空模具經系統作業後會輸出已填充的模具。圖 A.9 說明模具輸送系統內的細部作業流程，模具輸送系統內部包含了四道程式—啟動輸送帶與 RFID 標籤讀取器、停止 RFID 標籤讀取器、澆模、停止輸送帶。首先系統會在一號紅外線感知器受到觸發時，也代表有模具進入系統時啟動輸送帶同時呼叫 RFID 標籤讀取器開始持續進行資料讀取作業以獲得模具上 RFID 標籤裡所記錄的作業指令，RFID 讀取器若能夠在八秒鐘內讀取到標籤資訊時即會結束讀取，否則由於輸送帶載送模具的運行距離已超出 RFID 讀取器的讀取範圍這將導致系統發生 RFID 資訊讀取的錯誤而非按造正常程式停止 RFID 讀取功能。

讀取到的作業指令將會被傳送至下一階段的澆模程式，當三號紅外線感知器受到觸發，代表當模具已抵達所設定的位置時，系統會依據作業指令所記載的秒數來進行澆模程式。最後，系統會在四號紅外線感知器受到事件觸發，代表模具已抵達輸送帶終點時，會停止輸送帶完成一循環的模具輸送系統作業程式。

澆模程式可以再進一步拆解成四道作業流程—暫停輸送帶、開啟三號閥門、關閉三號閥門及恢復啟動輸送帶，如圖 A.10 所示。當三號紅外線感知器受到觸發時系統會停止輸送帶前進，接著開啟三號閥門直到澆模秒數到達後再關閉三號閥門並且啟動輸送帶。

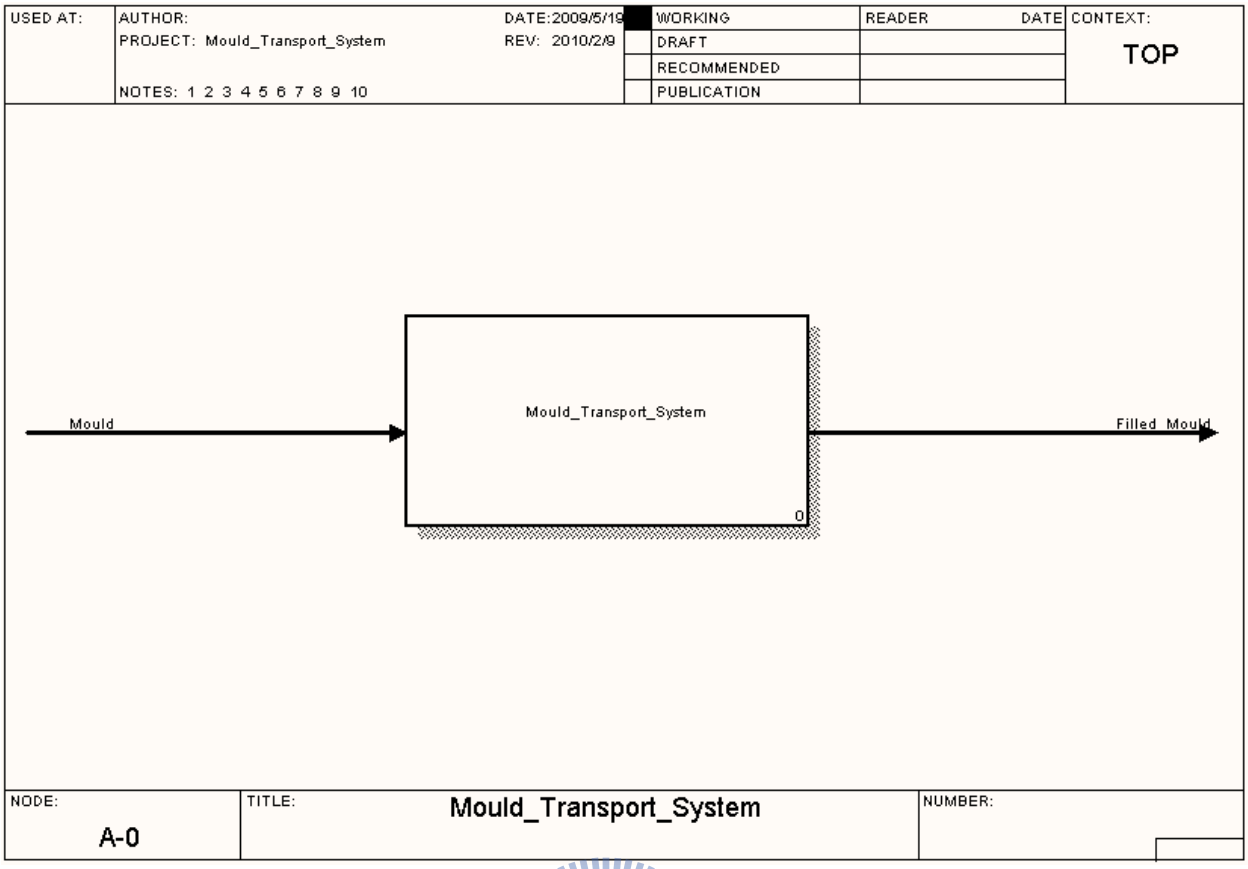


圖 A.8 模具輸送系統 A-0 層 IDEF0 圖

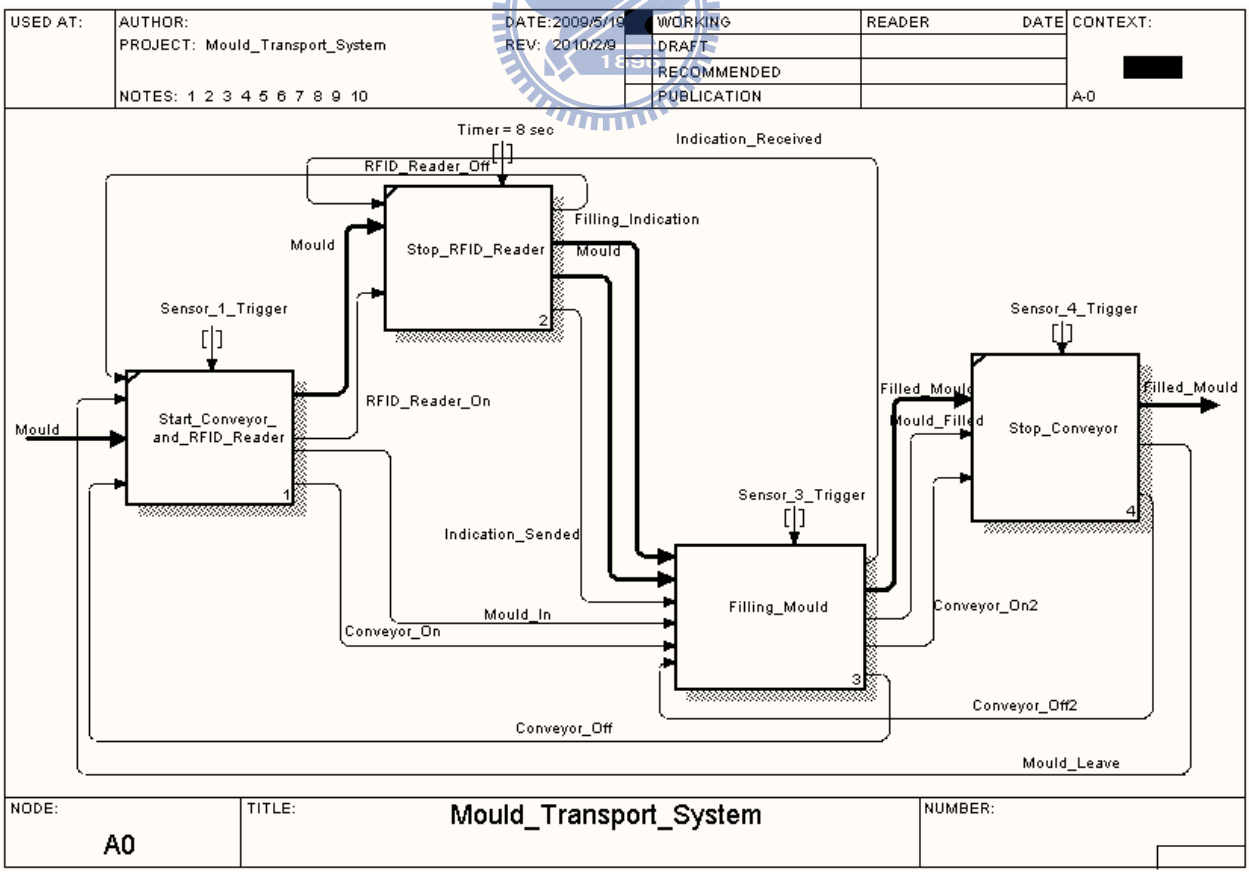


圖 A.9 模具輸送系統 A0 層 IDEF0 圖

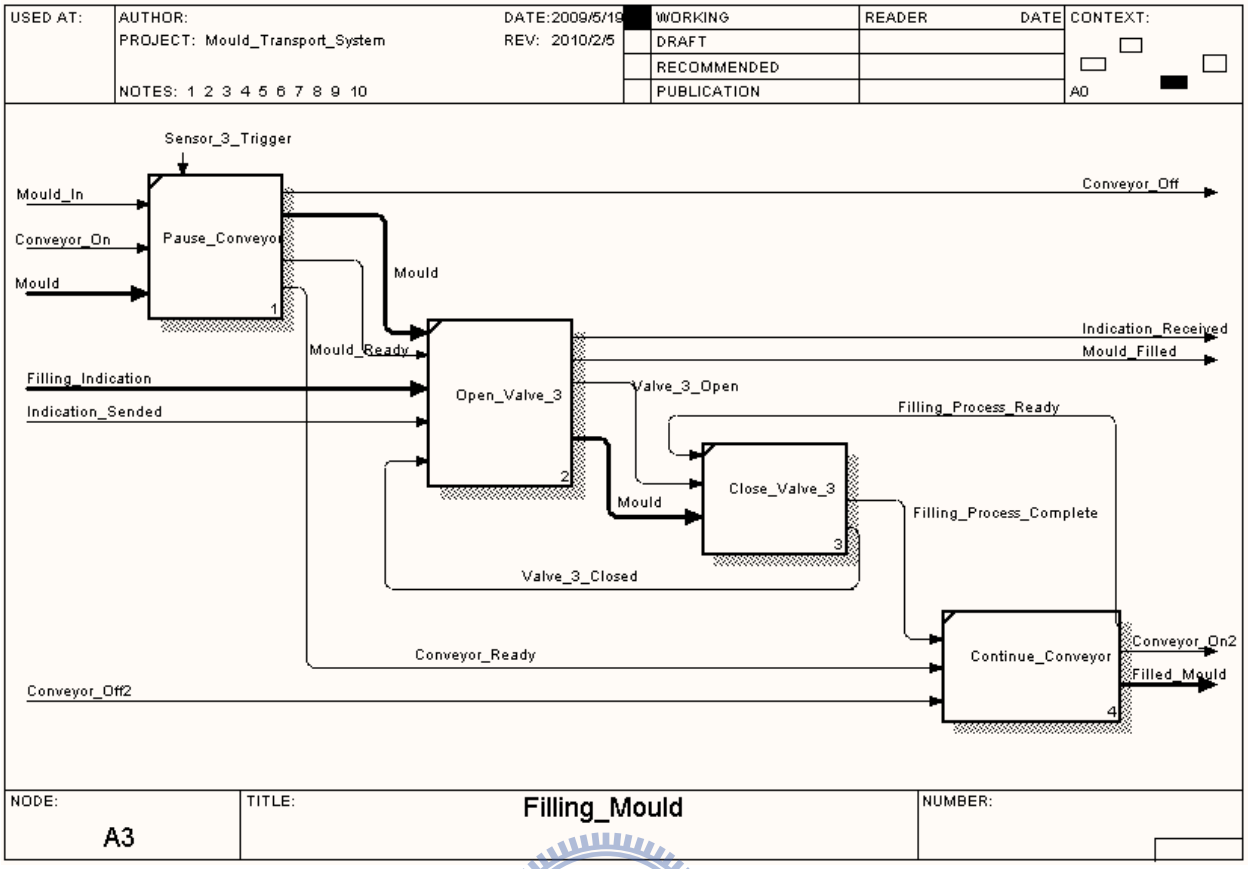


圖 A.10 模具輸送系統 A3 層 IDEF0 圖

附錄二 灌模系統監控法則

本附錄說明由灌模系統裴氏圖所產生的法則，總計包含有 19 條法則，如表 C.1 所示。以轉移點 T_1 為例，轉移點 $T\text{-}_1$ 代表的是啟動輸送帶及 RFID 標籤讀取器，其產生的法則 Rule T_1 包含兩層巢狀結構。在外層結構是指當暫存點 P_2、P_3、P_{16} 皆有浮標，亦即 P_2、P_3、P_{14} 均為可標記狀態，同時回授資訊滿足時，則執行內層法則。在內層結構是指當紅外線感知器接收到容器進入輸送帶的信號時(I_{12}=1)，則啟動輸送帶(O_8=1)、修改虛擬變數 Mold 狀態為 in、虛擬變數 Reader 為 on，並且將暫存點 P_2、P_3、P_{16} 狀態更改為不可標記狀態，暫存點 P_1、P_4、P_{13} 狀態更改為可標記狀態。其餘法則均可依此方式類推。

表 C.1 灌模系統法則

Rule T_1(啟動輸送帶及 RFID 標籤讀取器):
IF
Reader = off and P_2 = markable
and $I\text{-}_{11}$ = 1 and Switch != 1
and P_3 = markable and Mold = leave
and P_{16} = markable
THEN
IF
I_{12}=1 and Timer1 is rest
THEN
Start Timer1
ELSE IF Timer1 > 0
P_1 := markable
P_4 := markable
P_{13} := markable
O_8 := 1
Mold := in
Reader := on
P_2 := unmarkable
P_3 := unmarkable
P_{16} := unmarkable
Close Timer1
END IF
END IF

Rule T_2(停止 RFID 標籤讀取器):
IF
Reader = on and P_1 = markable
And Indication = null and P_6 = markable
THEN
IF
true and Timer2 is rest
THEN
Start Timer2
ELSE IF Timer2 > 0
P_2 := markable
P_5 := markable
Reader := off
Indication:= tag 資訊
P_1 := unmarkable
P_6 := unmarkable
Close Timer2
END IF
END IF

Rule T_3(暫停輸送帶):
IF
I_{11}=0 or Switch =1 and P_4 = markable
And Mold = in and P_{13} = markable
THEN
IF
I_{14}=1 and Timer3 is rest
THEN
Start Timer3
ELSE IF Timer3 > 0
P_3 := markable
P_{14} := markable
O_8 = 0
Switch := 2,
Mold := ready
P_4 := unmarkable
P_{13} := unmarkable
Close Timer3
END IF
END IF

Rule T_4(開啟三號閥門):
IF
Indication != null and P_5 = markable
and I_5 = 0 and P_8 = markable
and Mold = ready and P_{14} = markable
THEN
IF
true and Timer4 is rest
THEN
Start Timer4
ELSE IF Timer4 > 0
P_6 := markable
P_7 := markable
P_{15} := markable
O_4 = 1
Indication := null
Mold := filled
P_5 := unmarkable
P_8 := unmarkable
P_{14} := unmarkable
Close Timer4
END IF
END IF

表 C.1 灌模系統法則(續)

Rule T_5(關閉三號閥門):
IF
I_5 = 0 and P_7 = markable and
Process_1 = ready and P_{10} = markable
THEN
IF
true and Timer5 is rest
THEN
Start Timer5
ELSE IF Timer5 > 0
P_8 := markable
P_9 := markable
O_4 = 0
Process_1 := complete
P_7 := unmarkable
P_{10} := unmarkable
Close Timer5
END IF
END IF

Rule T_6(繼續啟動輸送帶):
IF
Process_1 = complete and P_9 = markable
and I_{11}=0 and Switch = 2 and P_{12} = markable
THEN
IF
true and Timer6 is rest
THEN
Start Timer6
ELSE IF Timer6 > 2000
P_{10} := markable
P_{11} := markable
O_8=1
Process_1 := ready
P_9 := unmarkable
P_{12} := unmarkable
Close Timer6
END IF
END IF

Rule T_7(停止輸送帶):
IF
I_{11} = 0 and Switch=2 and P_{11} = markable
and Mold = filled and P_{15} = markable
THEN
IF
I_{15} = 1 and Timer7 is rest
THEN
Start Timer7
ELSE IF Timer7 > 0
P_{12} := markable
P_{16} := markable
O_8 = 0
Switch_1 := 1
Mold = leave
P_{11} := unmarkable
P_{15} := unmarkable
Close Timer7
END IF
END IF

Rule T_8(啟動幫浦 a 及幫浦 b):
IF
I_0 = 1 and P_{18} = markable
and I_1 = 1 and P_{20} = markable and
Process_2 = proceeding and P_{39} = markable
THEN
IF
I_{10}=1 and Timer8 is rest
THEN
Start Timer8
ELSE IF Timer8 > 0
P_{17} := markable
P_{19} := markable
P_{40} := markable
O_0=1
O_1=1
Process_2 := proceeding
P_{18} := unmarkable
P_{20} := unmarkable
P_{39} := unmarkable
Close Timer8
END IF
END IF

Rule T_9(關閉幫浦 a):
IF
I_0 = 0 and P_{17} = markable
and m_a = lacked and P_{22} = markable
THEN
IF
I_8 = 0 and Timer9 is rest
THEN
Start Timer9
ELSE IF Timer9 > 30000
P_{18} := markable
P_{21} := markable
O_0 = 0
m_a := filled
P_{17} := unmarkable
P_{22} := unmarkable
Close Timer9
END IF
END IF

Rule T_{10}(關閉幫浦 b):
IF
I_1=1 and P_{19} = markable
and m_a = lacked and P_{24} = markable
THEN
IF
I_8 = 0 and Timer10 is rest
THEN
Start Timer10
ELSE IF Timer10 > 35000
P_{20} := markable
P_{23} := markable
O_1=0
m_b:=filled
P_{19} := unmarkable
P_{24} := unmarkable
Close Timer10
END IF
END IF

表 C.1 灌模系統法則(續)

Rule T_{11}(啟動幫浦 c 及一號閥門):
IF
m_a = filled and P_{21} = markable
and m_b = filled and P_{23} = markable
and I_2 = 1 and P_{26} = markable
and I_3 = 1 and P_{28} = markable
THEN
IF
true and Timer11 is rest
THEN
Start Timer11
ELSE IF Timer11 > 0
P_{22} := markable
P_{24} := markable
P_{25} := markable
P_{27} := markable
O_2 = 1
O_3 = 1
m_a = lacked
m_b = lacked
P_{21} := unmarkable
P_{23} := unmarkable
P_{26} := unmarkable
P_{28} := unmarkable
Close Timer11
END IF
END IF

Rule T_{12}(關閉幫浦 c):
IF
I_2=0 and P_{25} = markable
and m_h = lacked and P_{30} = markable
THEN
IF
I_9 = 0 and Timer12 is rest
THEN
Start Timer12
ELSE IF Timer12 > 30000
P_{26} := markable
P_{29} := markable
O_2 = 0
m-_h := filled
P_{25} := unmarkable
P_{30} := unmarkable
Close Timer12
END IF
END IF

Rule T_{13}(關閉一號閥門):
IF
I_3 = 0 and P_{27} = markable
and m_y = lacked and P_{32} = markable
THEN
IF
I_9 = 0 and Timer13 is rest
THEN
Start Timer13
ELSE IF Timer13 > 90000
P_{28} := markable
P_{31} := markable
O_3 = 0
m_y := filled
P_{27} := unmarkable
P_{32} := unmarkable
Close Timer13
END IF
END IF

Rule T_{14}(啟動二號攪拌器):
IF
m_h = filled and P_{29} = markable
m_y := filled and P_{31} = markable
I_7 = 1 and P_{34} = markable
THEN
IF
true and Timer14 is rest
THEN
Start Timer14
ELSE IF Timer14 > 0
P_{26} := markable
P_{30} := markable
O_7 = 0
m_h := lacked
m_y := lacked
P_{29} := unmarkable
P_{31} := unmarkable
P_{34} := unmarkable
Close Timer14
END IF
END IF

表 C.1 灌模系統法則(續)

Rule T_{15}(停止二號攪拌器):
IF
I_7=1 and P_{33} = markable
and I-$_{11}$=0 and m_x=outputed
and P_{36} = markable
THEN
IF
true and Timer17 is rest
THEN
Start Timer17
ELSE IF Timer17 > 20000
P_{34} := markable
P_{35} := markable
O_7 = 0
m-_x := generated
P_{33} := unmarkable
P_{36} := unmarkable
Close Timer17
END IF
END IF

Rule T_{16}(開啟二號閥門):
IF
m_x = generated and P_{35} = markable
and I_4 = 0 and P_{38} = markable
THEN
IF
true and Timer18 is rest
THEN
Start Timer18
ELSE IF Timer18 > 0
P_{35} := markable
P_{38} := markable
O_4 = 1
m_x := outputed
P_{36} := unmarkable
P_{37} := unmarkable
Close Timer18
END IF
END IF

Rule T_{17}(關閉二號閥門):
IF
I_4=1 and P_{37} = markable
I_4=1 and P_{40} = markable
THEN
IF
I_{10} = 0 and Timer19 is rest
THEN
Start Timer19
ELSE IF Timer19 > 90000
P_{38} := markable
P_{39} := markable
O_4 = 0
Process_2 := complete
P_{37} := unmarkable
P_{40} := unmarkable
Close Timer19
END IF
END IF