東海大學資訊工程研究所

碩士論文

指導教授:朱正忠 教授

一個基於統一塑模語言狀態機圖的自動化測試方法

An UML State Machine Diagram Based Automatic Testing Approach

研究生: 黃信豪

中華民國 一百零一 年 六月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 黃 信 豪 所提之論文

一個基於統一塑模語言狀態機圖的自動化測試方法

經本委員會審查，符合碩士學位論文標準。

學位考試委員會
召 集 人 楊朝棟 簽章

委 員 孔崇旭

張志宏

薛信林

指 導 教 授 朱延平 簽章

中華民國 101 年 6 月 29 日

摘要

軟體工程的生命週期簡單可分為分析需求，設計架構，實作，測試,維護五個階段。在軟體開發流程中,軟體測試是一項龐大的工程，而且軟體功能測試工作大部分是繁雜而且具重複性,需耗費大量成本及人力，因此如果能夠達到自動化測試，便可以節省大量的人力，成本和避免人為疏失，提高軟體品質，在軟體開發流程中，模型驅動開發的應用越來越廣泛,主要的概念是獲取所有系統設計的重要資訊，利用正規或是半正規的模型描述系統不同的抽象層級,故本研究中引用一套基於高階派翠網路模型自動化測試工具-MISTA(Model based Integration and System Testing Automation)對於系統自動化產生測試程式碼，完成自動化測試。

在本研究中，我們經由擷取統一塑模語言中狀態機圖描述系統行為的資訊,透過可延伸標記語言自動化轉換成高階派翠網路測試模型，再使用自動化測試程式碼生成軟體 MISTA 產生系統程式的所有測試案例和測試程式碼，達到自動化測試，節省軟體測試所需的大量人力與成本，並使軟體開發流程從設計架構到測試作一個銜接。

關鍵詞:軟體測試、統一塑模語言、高階派翠網路、可延伸標記言、模型驅動

Abstract

Software engineering life cycle can be divided into the five stages, analysis requirements, design architecture, implementation, testing, and maintenance. In the process of software development, software testing is a huge project, and most of the software functionality testing is complex and repetitive, should to spend a lot of cost and manpower. Therefore, you can save a lot of manpower and costs, avoid human error and improve software quality if we can achieve automated testing. In the software development process, the application of model-driven developments has grown wider recently. The main concept is to fetch all of the significant properties referring to system design, and use formal or semi-formal model to describe system in different levels of abstraction. In this paper, we apply a tool based on high level Petri Net model of automated testing -MISTA(Model based Integration and System Testing Automation) to automated generate test code for system.

In this paper, through capture the information of the state machine diagram describes the behavior of the system in the Unified Modeling Language. Through the eXtensible Markup Language automation converted to High Level Petri Net test model, and then using automated test code generating software MISTA generation test cases and test code of system to achieve automated testing, and save the software required for testing a large number of human costs, and the software development process from design architecture to the software testing for a bridging.

Keyword: Software testing, Unified Modeling Language, High Level Petri Net, eXtensible Markup Language, Model-Driven

目錄

圖目錄

表目錄

第一章 導論

1.1 前言

軟體系統在我們日常生活中已經漸漸變的重要，故軟體品質顯得越來越重要，而「軟體測試」是目前用來提升軟體服務品質可靠度最常見的方法，軟體測試其目標是找出程式運行過程與結果是否有錯誤，但軟體測試是一項龐大的工程，而且軟體功能測試工作大部分是繁雜而且具重複性，需耗費大量成本及人力，並且可能因人為失誤因素導致軟體開發成本上升，因此本研究將探討設計如何透過自動化的方式，進行軟體自動化測試，自動化產生測試案例跟測試碼，便可節省撰寫測試程式碼的大量時間，和節省人力成本，降低人為失誤因素，提高軟體品質。

1.2 研究動機

為達到自動化測試目的，基於模型測試方法(Model Based Testing-MBT)已受到關注，MBT 使用模型描述系統的行為資訊，模型在整個軟體開發生命週期可以多方面利用，包含改善軟體品質，規格，可靠性分析，測試產生等等。例如有限狀態機[1]和 UML 模型[2]都是 MBT 開發最常使用的模型，透過使用行為模型針對系統行為預測產生測試案例和可執行的程式測試碼，完成自動化測試。

1.3 研究目的

在軟體開發流程中，隨著模型驅動開發(Model-Driven Development-MDD)[3]越來越盛行，統一塑模語言 UML 已經成為模型驅動架構下的核心語言，在本研究中透過 UML 中的狀態機圖描述系統行為資訊，經由轉換成為高階派翠網路測試模型，套入本研究中引用的基於高階派翠網路的自動化測試軟體-MISTA，透過此軟體使軟體開發流程中的測試階段能夠達成自動化產生系統測試案例與測試程式碼，進而節省測試流程中的開發成本，確保軟體品質。

1.4 章節安排

論文的章節安排分述如下：

第二章 背景知識與相關研究，此章節說明與本論文相關的研究。

第三章 此章說明本論文使用之方法。

第四章 系統設計實作與研究案例。

第五章 結論與未來方向。

第二章 背景知識與相關研究

2.1 基於模型測試(Model-Based Testing)

基於模型測試被定義為是一種能夠自動化執行的黑箱測試，但是它不同於一般的黑箱測試，因為測試案例並不是依據需求文件產生，而是透過系統行為模型預測系統的行為模式自動來產生測試案例，使用這模式去產生一連串的測試輸入和預期結果，透過這些預測的輸入於系統的測試上(System Under Test，SUT)，而系統的輸出則和預期結果的輸出做比較以達到系統測試，透過此模式運作下，基於模型測試能夠達成自動化測試設計和測試產生來取代人為測試設計。

軟體測試與基於模型測試類似，在進行測試之前必須需要先決定測試策略，基於模型測試並沒有非常明確的定義，模型測試工作大致分為模型選擇，模型產生，找出測試準則，測試案例產生與執行。

基於模型測試有三個主要的工作[4]:

1. 設計功能測試模型:

測試模型代表對被測試的系統或環境的預測行為，測試設計者可以使用標準塑模語言如 UML 觀察被測試的系統，預期該系統的動態行為，相關的實體測試和對不同的測試配置的測試數據，透過鏈結模型元素如狀態(state)，轉換動作(transitions)和系統需求來確保需求和模型跟隨後產生的測試案例和測試

結果之間的雙向可追溯性，故測試模型必須準確和完整使系統測試足以自動化測試推導。

2. 決定測試產生準則:

模型通常可以產生出無限多的測試，所以測試設計者必須選擇一套準則來限制所產生的測試數量，(例如經由選擇最高優先權級的測試或確保對系統行為的具體覆蓋)，對測試選擇一個常用的方法是基於結構模型覆蓋(例如確定測試的模型元素覆蓋)，另一個有用的準則，則是確保所產生的測試案例能夠覆蓋所有的需求，可能對比較高風險性的需求產生更多的測試，在這種方式，測試人員可以用基於模型測試來實現需求導向或風險驅動的測試。

3. 測試產生:

在通常的情況下，在基於模型測試下的測試產生是完全自動化，產生出來的測試案例是一序列高層次的事件或活動，為每個事件或活動輸入參數，預期的參數輸出跟回傳值，而測試序列是容易理解和完整的足以手動或自動執行測試，如果有必要，測試人員可以把測試工作再細化至更多具體層級來支持自動化執行。

在基於模型測試中的一個關鍵問題是過多的概念和方法，不遵循

一個共同的約定，自 2005 年以來，物件管理組織(OMG)提出 UML Testing Profile(UTP)規範支持基於模型測試整合 UML，目前有幾個自由軟體和商業解決方案實施 UTP，一些研究論文和公司(IBM、Microsoft 等)也引用 UTP，OMG 在 2011 年七月已經完成修訂 UTP。以下為基於模型測試的工具列表，如表 1 [18][19][20][21][22]

表 1 基於模型測試的工具

工具	測試模型	測試產生標準
CertifyIT v5.1 [18]	商務企業流程模型或統一塑模語言	測試資料和驗證點
Conformiq Designer v4.4 [19]	狀態圖	需求驅動測試產生，黑箱測試設計
Tedeso 3.0 [20]	統一塑模語言活動圖和循序圖	模型和資料覆蓋
TestCast Generator Beta [21]	統一塑模語言狀態機圖	狀態，轉換動作和判斷覆蓋
Spec Explorer 2010 [22]	Spec#	轉換動作覆蓋

2.2 派翠網路模型(Petri net model)

派翠網路（Petri-Net）理論是 Petri 於 1962 年的博士論文中所提出，是一種兼具圖形數學特性的系統塑模分析與發展工具，至今已被廣泛用在不同領域作為系統之模組化建構、分析與模擬。由於 Petri-Net 的優異數學特性，使其成為一種著名的塑模工具（modeling tool），常被用來塑模具有同時、非同步、選擇、分散、非決定性以及推測性等特徵之系統，這套工具提供了圖形化的表示方式，和數學理論的基礎，可用來作為分析系統行為的一種工具。

Petri Net[11]是一種有向圖，但這種有向圖包含了初始狀態，稱作初始狀態標記，以 M0 表示。一個 Petri Net 的圖形，是一個有方向性、有權重、雙向的圖，以下為派翠網路圖的四種符號:

1. 位置(place 以圓圈表示)

   指的是派翠網路圖中的狀態，亦是要觸發 Transition 所需要的條件，或是觸發 Transition 之後所產生的條件。

2. 轉變(transition 以長條形或是長方形表示)

   一個轉換動作，表示由一個 Place 到另一個 Place 的轉換過程，表示一些處理動作、行為或會進行的反應。

3. 位置和轉變之間的流向關係(Arc)

   連接 Place 與 Transition 以表示流向關係，每個弧標示上一正整

數表示其權重。

4. 代表系統執行的權杖(Token)

用來標示這個Place是否處於「致能（enable）」的狀態，當圖中Transition所有輸入的Place中均含有Token時，Transition方可以被觸發。

派翠網路以 PN={P,T,A,W, M_0}來定義。

1. P={P1,P2,P3,.....Pn} n≥ 0，代表所有place的集合。
2. T={T1,T2,T3,.....Tm} m≥ 0，P∩ T = ∅, T ∩ P = ∅，代表所有transitions的集合。
3. A= {P × T} ∪ {T × P}，代表路徑，{P× T}為Place到Transition的Arc。
4. W={1,2,3,.....}代表權重函數。
5. M_0:P→ N,N={0,1,2,....}代表系統的起始標記。

在建構派翠網路模型時，會用到兩個觀念：條件(conditions)與事件(events)，在Petri Net 中，places表示條件，transitions表示事件，一個transition 通常有一個或數個的輸入places 和輸出places，分別代表這個事件(transitions)的事前條件(preconditions)和事後條件(post conditions)。對於places和transitions的解釋[11]可以參照表 2

表 2 places 和transitions的解釋

Input Places	Transition	Output Places
Preconditions	Event	Post conditions
Input data	Computation step	Output data
Input signals	Signal processor	output signals
Resources needed	Task or job	Resources released
Conditions	Clause in logic	Conclusion(s)
Buffers	Processor	Buffers

在定義上, place 與 place 不可用 arc 直接連接, transition 與 transition 也不可直接互連，數學定義與圖型定義可以參見 High Level Petri Net ISO Standard[12]。

在本研究中使用高階派翠網路 PrT Net(Predicate/Transition Nets)[13][14][15]或稱為 Function Net[16][17]為測試模型(Test Model)，其定義如下:

PrT net 是一組元素組合$(P, T, F, \sum, L, \varphi, M_0)$

(1)P 是一個有限的集合，代表所有 places 的集合。

(2)T 是一個有限的集合，代表所有 transition 的集合，

$(P \cap T=\emptyset, P \cup T \neq \emptyset)$。

(3)F 代表流動關係，或所有 arcs 的集合，$F \subseteq (P \times T) \cup (T \times P)$。

(4)∑是一個結構，存在一些個別的常數，連同一些操作(operations)和關係(relations)。

(5)L 是在弧(arcs)上的標籤功能，給定一條弧(arc)，弧上有標籤 f，L(f)是標籤的集合，這是個弧上個別的變數集合，代表弧(arc)連接到 place 的變數。

(6)φ是對應到轉換動作的方程式集合，$t \epsilon T$, φ(t)是一個邏輯公式由∑中的變數，操作，關係建構而成。

(7)M_0 代表初始(initial)或目前的標記，$M_0 = \cup_{p \in P} M_0(p)$，$M_0(p)$是在 place 中 tokens的集合。

我們簡單表示一個 Function Net 的圖形如**錯誤! 找不到參照來源。**，在圖中我們看到有三個 place，一個 transition，初始標記(initial marking)為{P1(1,1), P1(1,2), P2(2)}，P1 有兩個 tokens(1,1)跟(1,2)，P2 有一個 token(2)，在 Function Net 中可以有多個初始標記，每個初始標記將產生一組測試套件，這將使得測試資料數據可以被分割成比較小的資料集合，以降低測試模型的複雜度，轉換動作觸發則是在一個轉換動作中有特定滿足條件的輸入時發生，觸發(Firing)啟動轉換動作將匹配的 token 從來源輸入的 place 中移除，並且加入一個

新的 token 至輸出的 place 中，這將會再產生出新的標記。在圖 1 中我們使用 P1(1,2)這個 token 來表示轉換動作觸發的條件，T1{x=1，y=2}因為 x 和 y 是 T1 的正式參數且滿足觸發 T1 中 x 不等於 y 的條件，此時會將 P1 中的 token(1,2)移除，並在 P3 中加入一個新的 token(1,2)，經過此觸發 T1 這個動作之後，得到的標記會變成{P1(1,1), P2(2),P3(1,2)}。

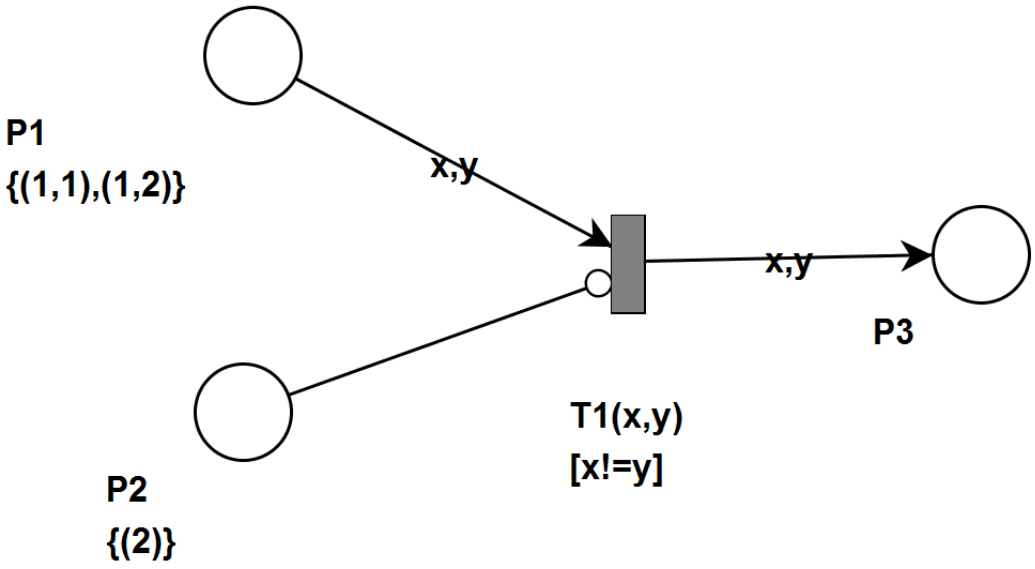


圖 1 Function Net 範例圖

2.3 MISTA(Model based Integration and System Testing Automation)

MISTA[5]是一套基於高階派翠網路(Function nets)為測試模型來產生可執行測試程式碼的工具，MISTA 從 Model-Implementation Description(MID)產生可執行的測試程式碼，MID 格式包含測試模型

(Model Description)，Model-Implementation Mapping(MIM)跟使用者所提供的 Helper code，在 MIM 中將測試模型中的各個元素對應至系統程式碼，而 Helper code 則必須填寫程式碼的標頭檔資訊。

MISTA 有下列幾點特點: (1)Function nets 為測試模型可以表示控制和資料導向測試需求，他們可以建立在不同的抽象層級 (2)測試模型可以透過以圖型或電子表格格式(Excel)編輯並且分析通過驗證 (3)測試可以自動化產生去滿足測試模型不同的覆蓋準則，如可達性覆蓋，狀態覆蓋，轉換覆蓋，深度覆蓋，隨機產生，狀態死結覆蓋，偏序和成對技術來選擇減少測試的數量 (4)自動化產生各種程式語言可執行的測試程式碼，包含 Java，C#，C/C++和 HTML 等 (5)測試可以被產生而且線上執行，這針對非確定系統測試特別有用。

MISTA 主要的元件如圖 2:

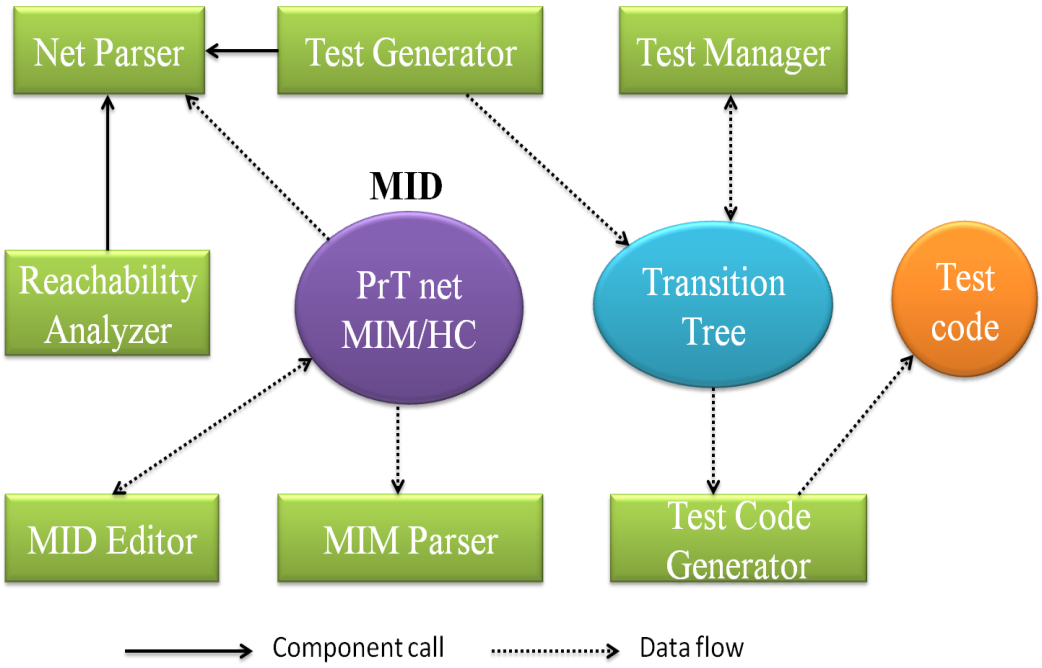


圖 2 MISTA 架構

- MID editor:一個圖形化界面提供使用者編輯 MID 格式,MID 格式也可以透過 Microsoft Excel 編輯。
- MID parser: Net parser 用來解析在 MID 格式中 PrT net 的文字描述，MIM parser 用來解析 MIM 規格和 helper code，這兩個 parser 用來檢查使用者所提供的 MID 規格語法是否有錯誤。
- Reachability analyzer:驗證在 MID 格式中的 PrT net 是否能夠從初始標記到達目的標記(goal marking)，可達性分析可以幫助解決 MID 中的未發現問題，例如:一開始便知道 goal marking 可以從 initial marking 達到，但是可達性分析有不同的結果，則代表 MID 格式中的 PrT net 模型有錯誤。
- Test generator:根據所選擇的覆蓋準則產生 model-level 的測試，這

些測試被組織起來並且可視為 transition tree，MISTA 支持從 Prt net 產生測試的幾種覆蓋準則，如可達性覆蓋(reachability coverage)，狀態覆蓋(state coverage)，轉換覆蓋(transition coverage)，深度覆蓋(depth coverage)，隨機產生(random generation)，狀態死結覆蓋(coverage of deadlock states)等。

- Test manager:允許使用者管理所產生的測試，例如將測試匯出至一個檔案，匯入測試，刪除測試，改變測試順序，重新定義測試的實際參數，測試中從外部插入額外的程式碼。
- Test code generator:從 transition tree 產生所選擇目標語言的測試程式碼，目前 MISTA 支援的語言/測試框架:Java/JUint，Jfcunit(JUnit 是對 Java 程式的單元測試框架，Jfcunit 是 JUnit 的擴充針對 Java 程式的 GUI 測試)，C/C++，C#，VB 跟 HTML。

2.4 統一塑模語言(Unified Modeling Language, UML)

統一塑模語言(UML)[6]是一種 Modeling Language，結合了 G. Booch，J. Rambaugh，與 I. Jacobson 三人的物件導向方法論所提出的一種物件導向語言，並被 OMG 納入為標準。UML 結合了 Booch 的物件導向方法、Rambaugh 的物件塑模技術與 Jacobson 的物件導向軟

體工程，目前 UML 已成了一種通用的物件導向語言(Object Oriented Language)，廣為被應用在各類系統的描述及設計上，成為一種溝通設計理念的語言，除了主導廠商 Rational 的推廣及擴展之外，其他為數眾多的廠商及研究單位也利用 UML 做為表達設計或研究成果的語言。在軟體工程中，UML 已被應用在相關的研究中，譬如：Software Architecture、Framework、Patterns、Software Process。應用在其他領域如：Real-Time System、Embedded System 以及 Workflow。

雖然UML被應用在許多領域，同時也被視為是塑模的重要工具，但 UML 的使用往往過於彈性而缺乏嚴謹，導致需求分析與系統設計的模型容易產生不完整及不一致性[7]。然而不只 UML，包含所有現有的模型建置技術都必須要有一個正規化的方式以達成軟體開發與維護過程中軟體模型的高度整合性，目前 OMG 亦發現這個問題，訂定了 Diagram Interchange 標準[8]，不過僅限於UML相關的圖形轉換，對於整合其他標準或非圖形塑模方式，尚未支援。

2.5 XML(eXtensible Markup Language)

XML(eXtensible Markup Language)是由 W3C(World Wide Web Consortium)所制定的標準[9]，一種延伸式的標記語言，具有擴展性(Extensibility)、結構性(Structure)、描述性(Description)、確認性(Validation)等特性。同時 XML 具有跨平台的功能，對於不同的作業

系統、硬體設備、應用軟體、多元的輸入模式，開發者可以自行制定符合己身需求的標記(Tag)，做結構性的描述，促使相同的一份文件呈現不同的規格，適用於不同的軟體，符合不同的設備、滿足多重的輸入方式。

XML 本身就是一種 Meta Language，可利用 XML 定義各種 Model 的文件語言，在進行各種 Model 轉換時可以利用 XML 延伸出的 XMI 來幫助檔的交換及存取。XMI 是 XML Metadata Interchange 的縮寫，在 XMI 裡 X 代表 XML 跟 extensible 雙重意義。XMI 在 1999 年 3 月 23 日正式被 W3C 所建議為正式標準。

XMI[10]是一種提供以 XML 來交換 Object-Oriented Models 和資料的標準，XMI 同時是 OMG(Object Management Group)所認可的一種新的標準，讓使用者可以根據 XMI 的標準標籤(Tag)將 UML 文件轉換為 XML 及 DTD 文件。XMI 讓 XML 使用起來更加的容易，因為它應用了 UML 圖形化的特性來產生 XML 所需的文件及 DTD 檔。而且 XMI 讓 DTD 檔、軟體設計及 XML 檔資料可以一致。

透過本章節上述幾個小節介紹本文相關的背景知識與相關研究，我們將於第三章節研究方法介紹說明由狀態機圖轉換成高階派翠網路模型的轉換對應規則，最後在第四章節介紹透過由狀態機圖 XML

檔案銜接 MISTA 自動化產生程式測試碼的流程。

第三章 研究方法

此章節為描述本論文研究之方法，本研究為了能夠達成自動化產生系統測試案例和測試程式碼，提高軟體品質，降低軟體測試成本與時間，引用一套基於高階派翠網路為測試模型的自動化產生測試案例和測試程式碼的自由軟體工具-MISTA(Model based Integration and System Testing Automation)針對系統自動化測試。

在軟體開發流程中模型驅動開發(Model-Driven Development, MDD)方法越來越盛行，透過模型描述系統不同的抽象層級獲取所有系統設計的重要資訊，故使用基於模型來測試軟體功能的模型測試(Model-Based Testing)在近幾年已經也漸漸受關注，如有限狀態機和 UML 模型都是 MBT 開發常使用的模型，透過使用行為模型針對系統行為預測產生測試案例，但基於模型測試要自動化產生可執行的測試程式碼，會遇到兩個問題，第一個就是從模型產生的測試往往不夠完整，因為無法確定實際的參數，例如以統一塑模語言(UML)中狀態機圖(State Machine Diagram)或其他模型當作測試模型，因為很難自動預測確定的實際參數測試序列，如此一來便無法預測測試程式碼預期所需要的輸入跟輸出，第二個問題則是因為軟體開發過程中塑模和系統程式碼撰寫所使用的語言不同，如果要自動化執行測試往往需要特定的測試驅動程序或配適器。

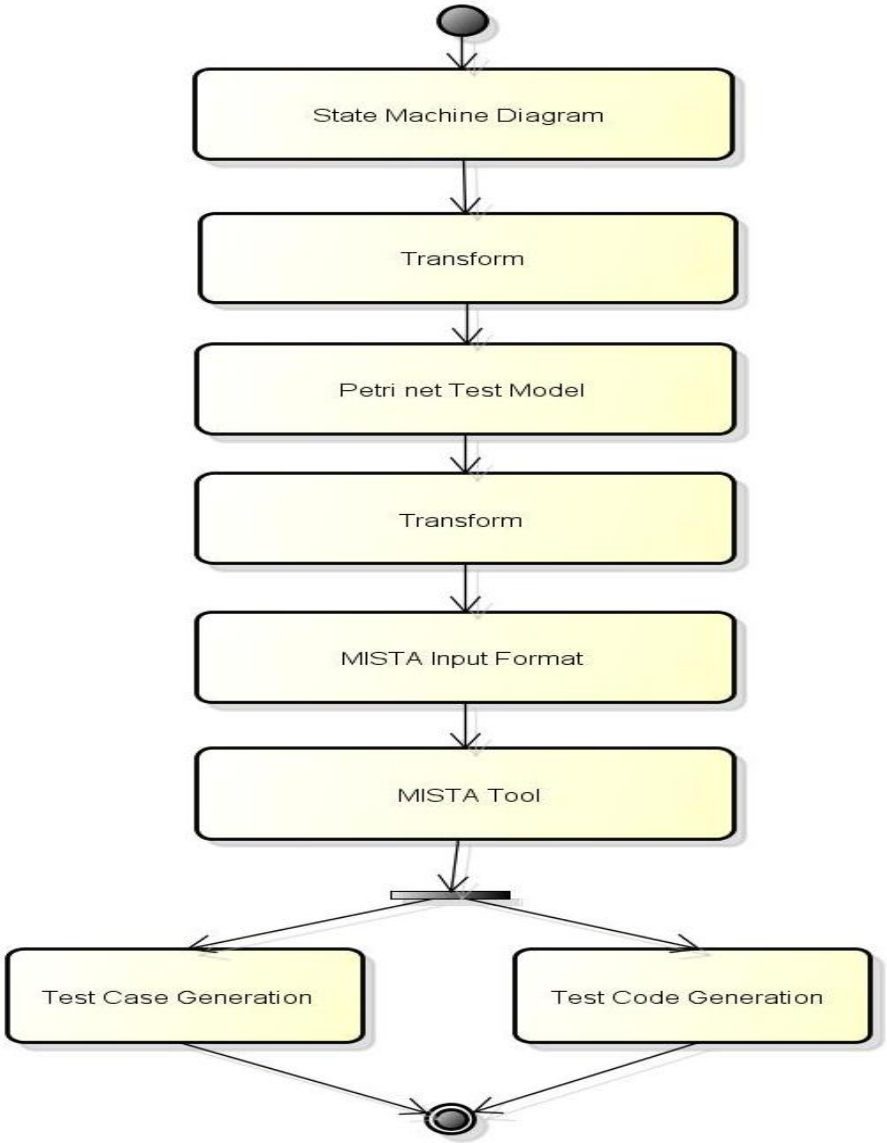


圖 3 自動化產生測試程式碼流程

有鑑於上一段描述基於測試模型要自動化產生程式測試碼所面臨的問題,在本研究中引用工具MISTA自動化產生程式測試碼軟體，此工具是以高階派翠網路模型(High Level Petri Net Model)為測試模型解決無法自動預測實際參數的問題,但是在現今軟體開發塑模過程中統一塑模語言已經變成其主要的核心語言,故本研究想提供一個從

UML 狀態機圖中所描述的系統行為資訊，透過延伸式標記語言(XML)與模型轉換規則將狀態機圖中的元素相對應轉換至高階派翠網路模型的轉換機制，讓開發者可以直接使用狀態機圖銜接 MISTA 軟體，使其自動化產生測試案例和測試程式碼，而不需要再另外花費時間建構高階派翠網路測試模型即可達到軟體自動化測試，本研究方法流程如圖 3。

在接下來的幾個小節將介紹由狀態機圖轉換至測試模型-高階派翠網路模型的相對應轉換規則。

3.1 狀態對應

在狀態機圖中的狀態(State)代表在某一時間下的情況或前後關係，在轉換過程相對應至高階派翠網路中的 Place，如圖 4

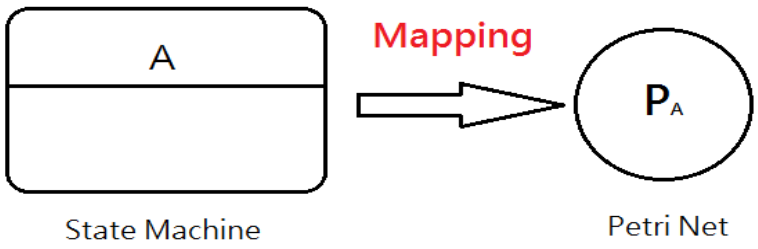


圖 4 狀態對應

3.2 轉換動作對應

轉換動作代表由某個狀態轉到另一狀態，在狀態機圖上每個轉換動作都會有一個標籤，它由三個部分組成:行為辨識資料[成立條件]/活動，在本研究的轉換對應將狀態機圖中的行為辨識資料對應至高階派翠網路中的 Token，而成立條件/活動則相對應至 Transition 中的 guard，如圖 5

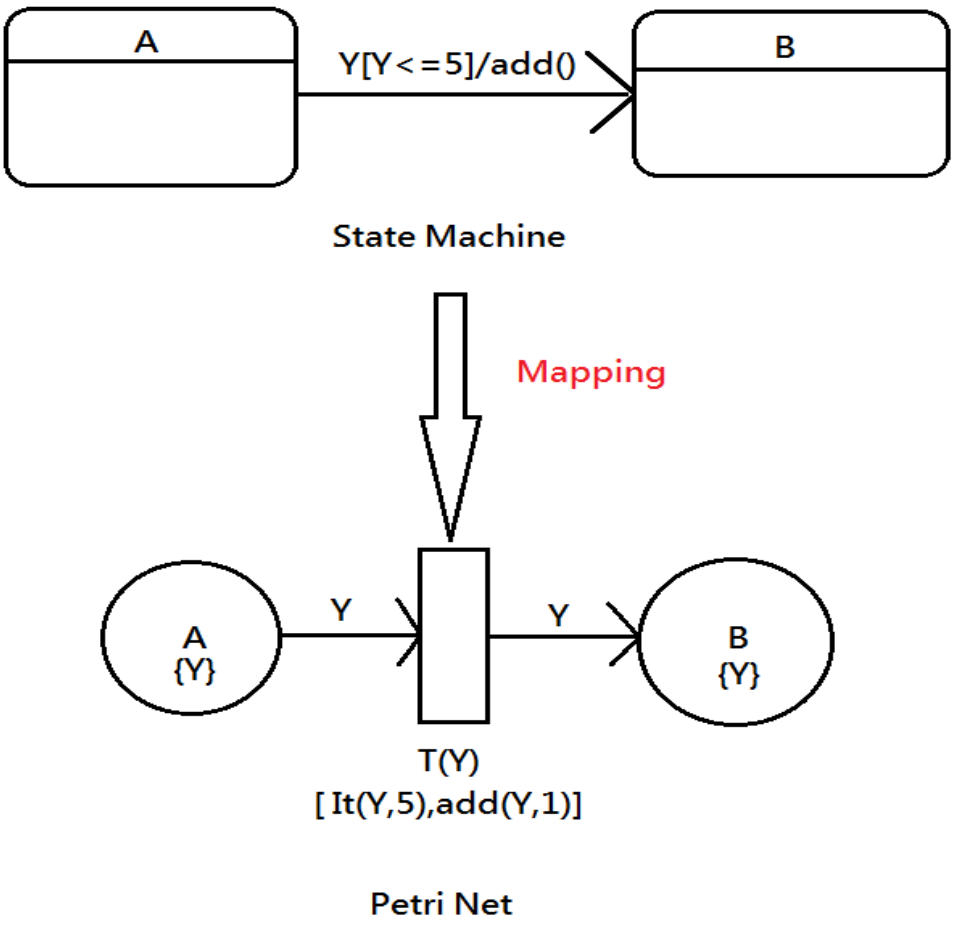


圖 5 轉換動作對應

3.3 內部活動與自身轉換動作

在狀態機圖中狀態回應某事件時不一定需要發生轉換動作，可能會發生內部活動(internal activity)，在這時候我們將事件，成立條件與活動放在狀態圖的方塊裡面，在對應至高階派翠網路的轉換規則中，我們將狀態機圖中的內部活動與自身轉換動作在高階派翠網路中直接視為一個轉換動作，而做完轉換動作之後再回到原本的 place，如圖 6

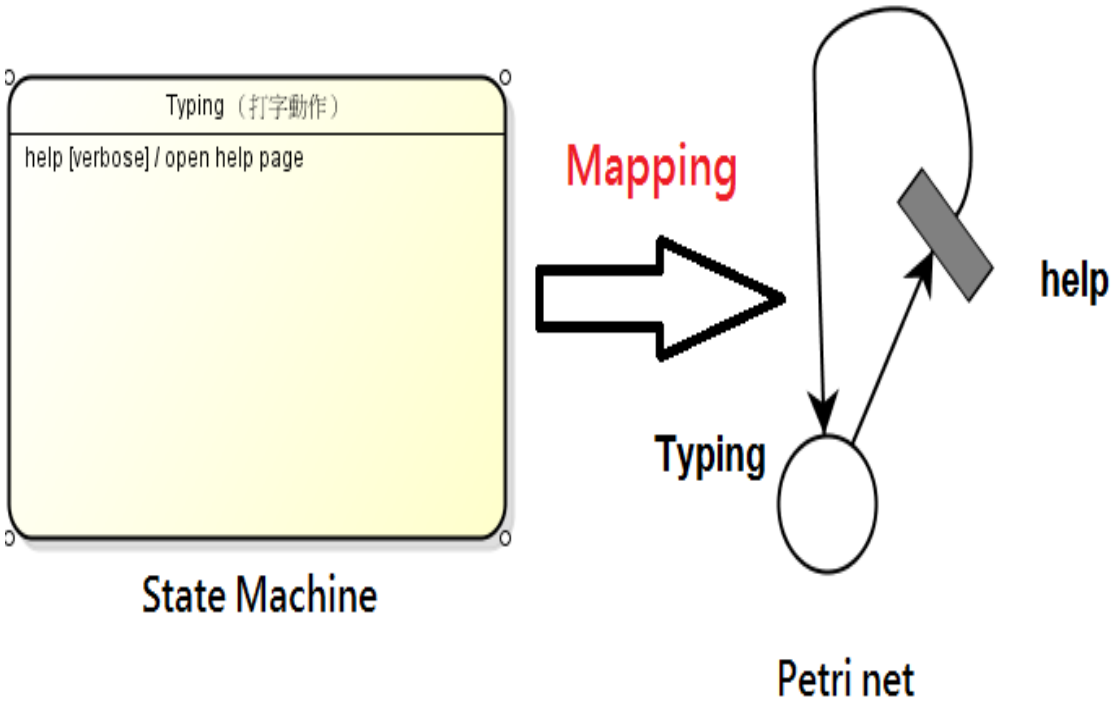


圖 6 內部活動與自身轉換動作對應

3.4 超狀態對應

在 State Machine Diagram 中，我們常常發現好幾個狀態一起分享相同的轉換動作與內部活動的情形，在這種情形下我們可以將這些狀態變成子狀態(substate)，然後把共享的行為放到超狀態(superstate)上，如圖 7 中的狀態機圖，如果沒有超狀態的話，我們必須需要替 B 狀態中的三個狀態畫出 cancel 的轉換動作。

超狀態也叫做合成狀態(composite state)，在超狀態裡所包含的狀態稱為子狀態，最上層的子狀態稱為直接子狀態(direct substate)，第二層以下的子狀態則是有延遞效果的巢狀子狀態(nested substate)，它們必須遵守上層狀態的行為。

超狀態(superstate)對應到派翠網路模型的對應方式如圖 7，在圖中我們將狀態機圖中的超狀態 B 在對應過程中看成一個 place B_input，跟一個 place B_output，由 B_input 的輸出做轉換動作 new，再由 place C，D，E 各自的 cancel 轉換動作給 B_output，完成狀態機圖中的超狀態對應。

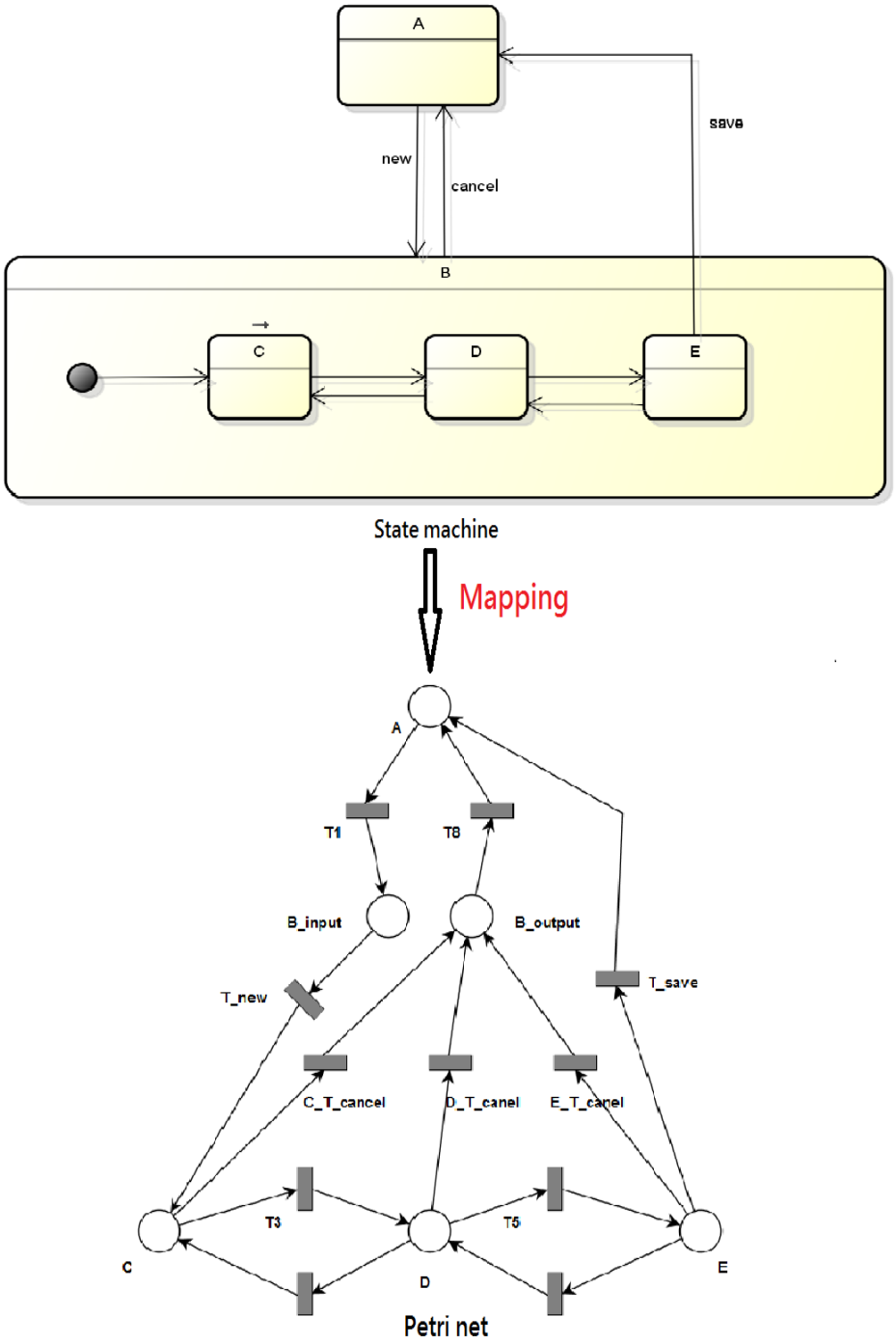


圖 7 超狀態對應

3.5 分岔虛擬狀態與會和虛擬狀態對應

為了處理某個進入轉換動作分岔成多個轉換動作的情況，我們可

以用分岔虛擬狀態來表示，同樣的，為了處理多個離開轉換動作會和成單一的轉換動作情況時使用會和虛擬狀態來表示。

分岔虛擬狀態與會和虛擬狀態與高階派翠網路對應方式本方法採用高階派翠網路中的 And -join-split 規則來表示，對應方式如圖 8: 分岔虛擬狀態對應至 And-split，會和虛擬狀態則對應到 And-join。

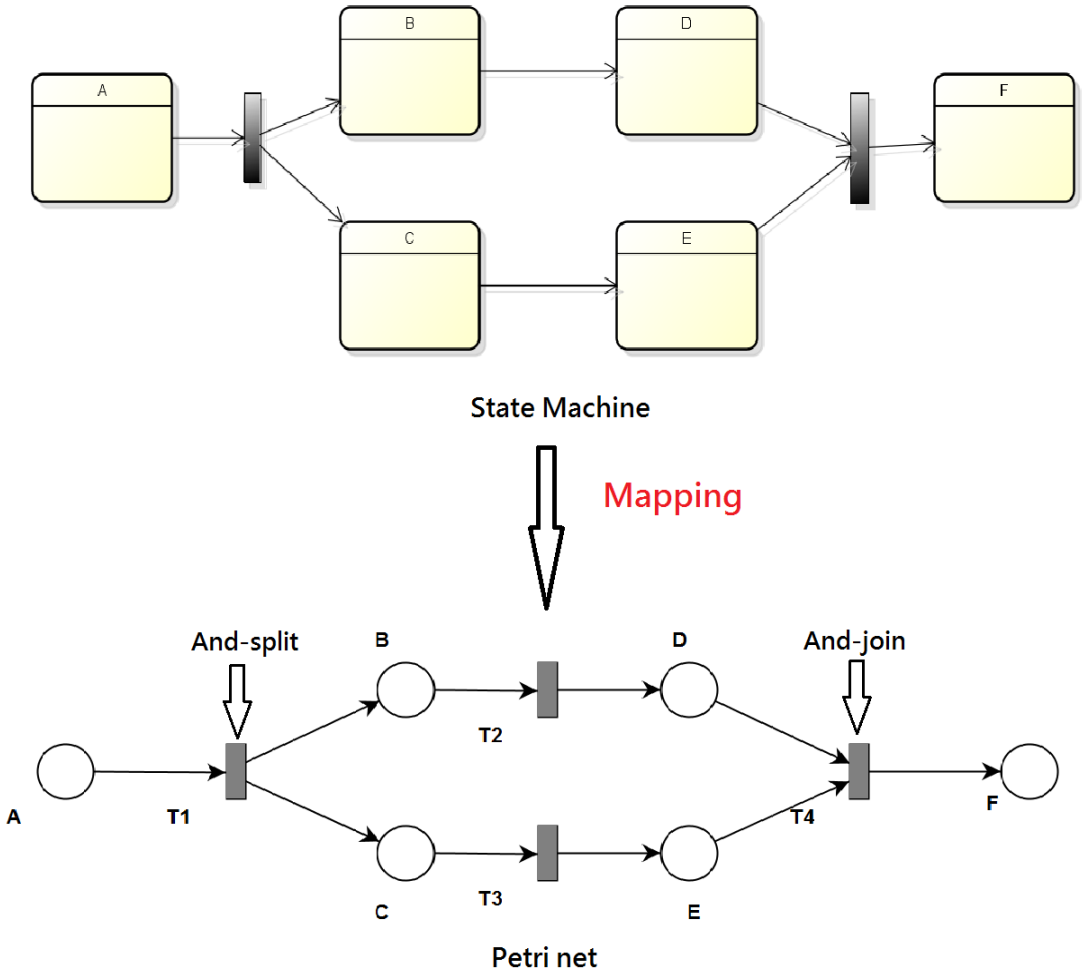


圖 8 分岔，會和虛擬狀態對應

最後我們將使用一個狀態機圖案例，套用本章節所介紹的轉換規則，並於第四章說明由狀態機圖 XML 資訊自動轉換成 MISTA 中的 MID 格式的轉換工具，透過 MISTA 工具自動化產生程式測試碼的實驗流程介紹。

第四章 系統實作與研究案例

本研究最後將在 Eclipse 環境底下實作一個從狀態機圖轉換成派翠網路模型的模型轉換工具，套用上第三章節介紹的狀態機圖與高階派翠網路模型的轉換對應規則，使用狀態機圖資訊的 XML 儲存格式，將所需要的資訊擷取，自動化填入 MISTA 工具所需要的輸入格式 MID[5](Model- Implementation Description)中用來描述模型資訊的 Model Description 欄位(可用 Microsoft Excel 編輯)，再透過手動編輯 MIM 與 Helper Code 後匯入給 MISTA 工具自動化產生測試碼，其架構圖如圖 9。

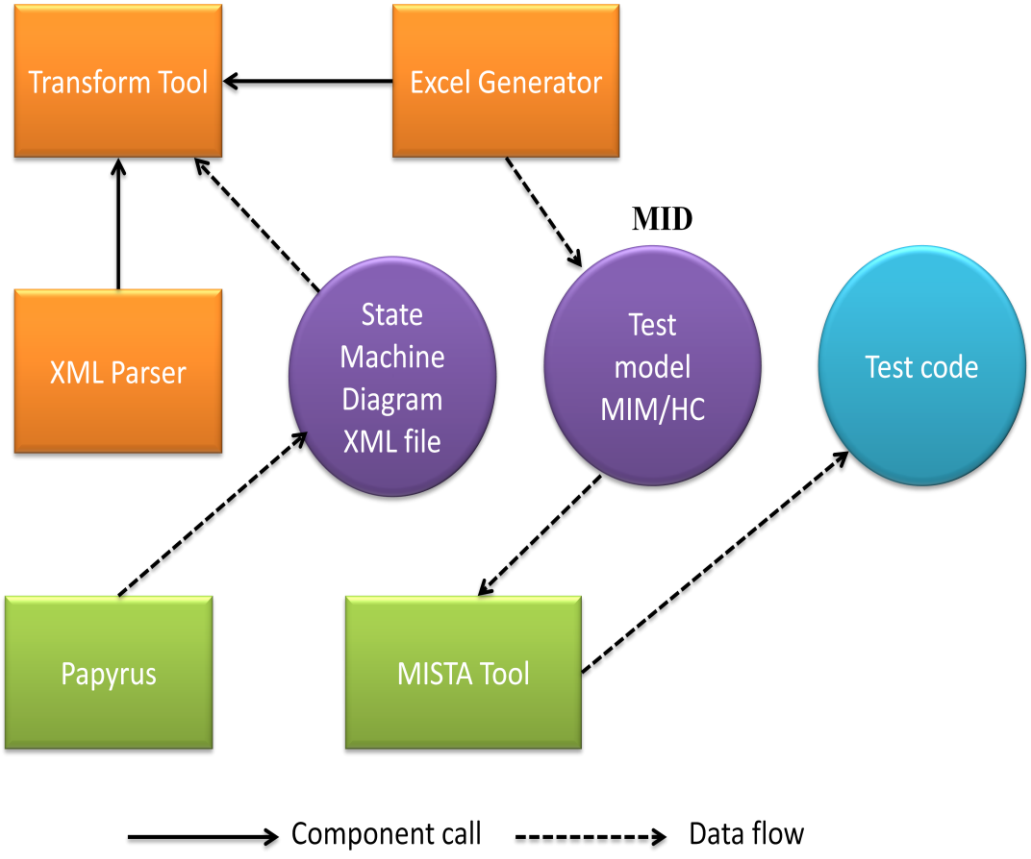


圖 9 系統架構圖

在本研究案例中我們使用 Papyrus UML 塑模工具畫製狀態機圖並透過 Papyrus 產生其狀態機圖的 XML 資訊如圖 10 和圖 11，在圖 10 中我們可以看到左邊欄位中有副檔名名稱分別為.di 與.uml 的檔案，.di 檔即為在 Papyrus 中的狀態機圖模型的圖形檔案，而.uml 檔則為此狀態機圖模型的 XML 檔案資訊，而在下面的部分則為狀態機圖模型中轉換動作的詳細資訊如識別資料，成立條件和活動，在圖 11 XML 檔案中紅色框線部分所描述的是狀態機圖中各個狀態的型態，狀態 ID 與狀態名稱，藍色框線中所描述的為狀態機圖中的轉換動作資訊，包括轉換動作的 ID，名稱與轉換動作中來源的狀態 ID 與目的狀態 ID，紫色框線部分代表狀態機圖中轉換動作的成立條件(使用者在繪製狀態機圖時必須將轉換動作中的成立條件語法明確填寫進入，不能透過自然語言描述成立條件)，褐色框線則代表狀態機圖中的識別資料，綠色框線部分則代表超狀態(superstate)與子狀態間(substate)的相關資訊。

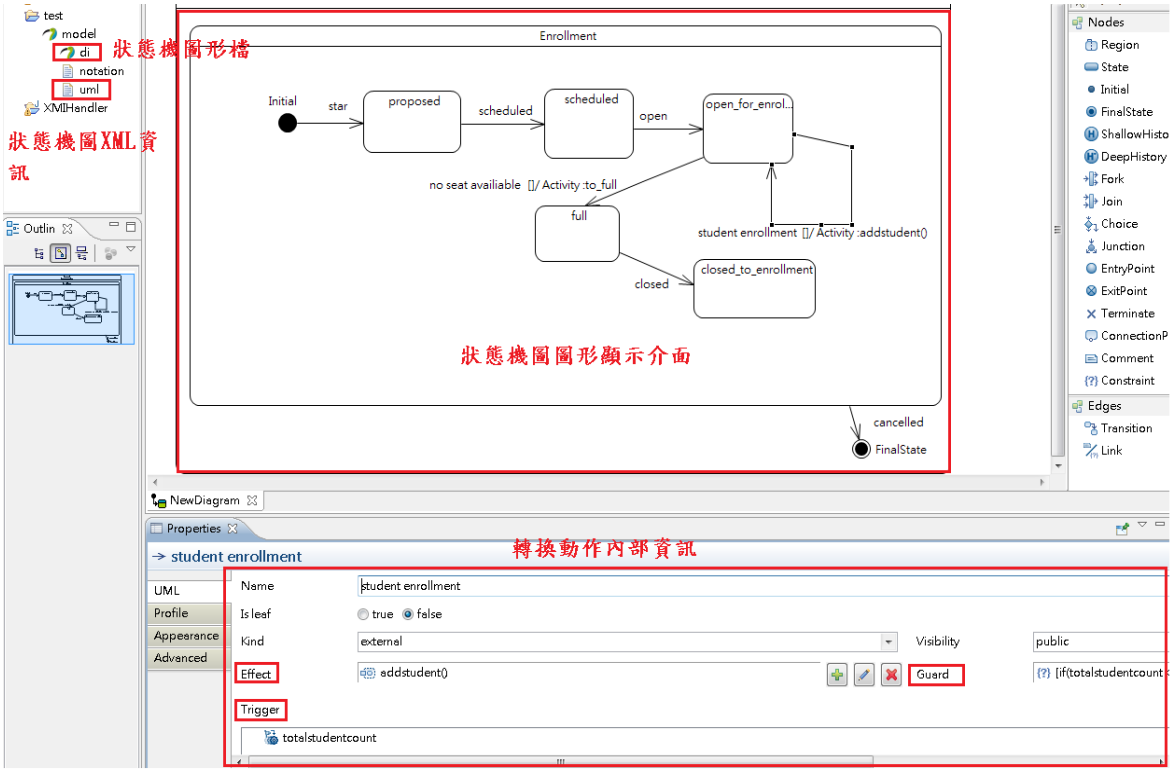


圖 10 使用 Papyrus UML 畫製狀態機圖模型

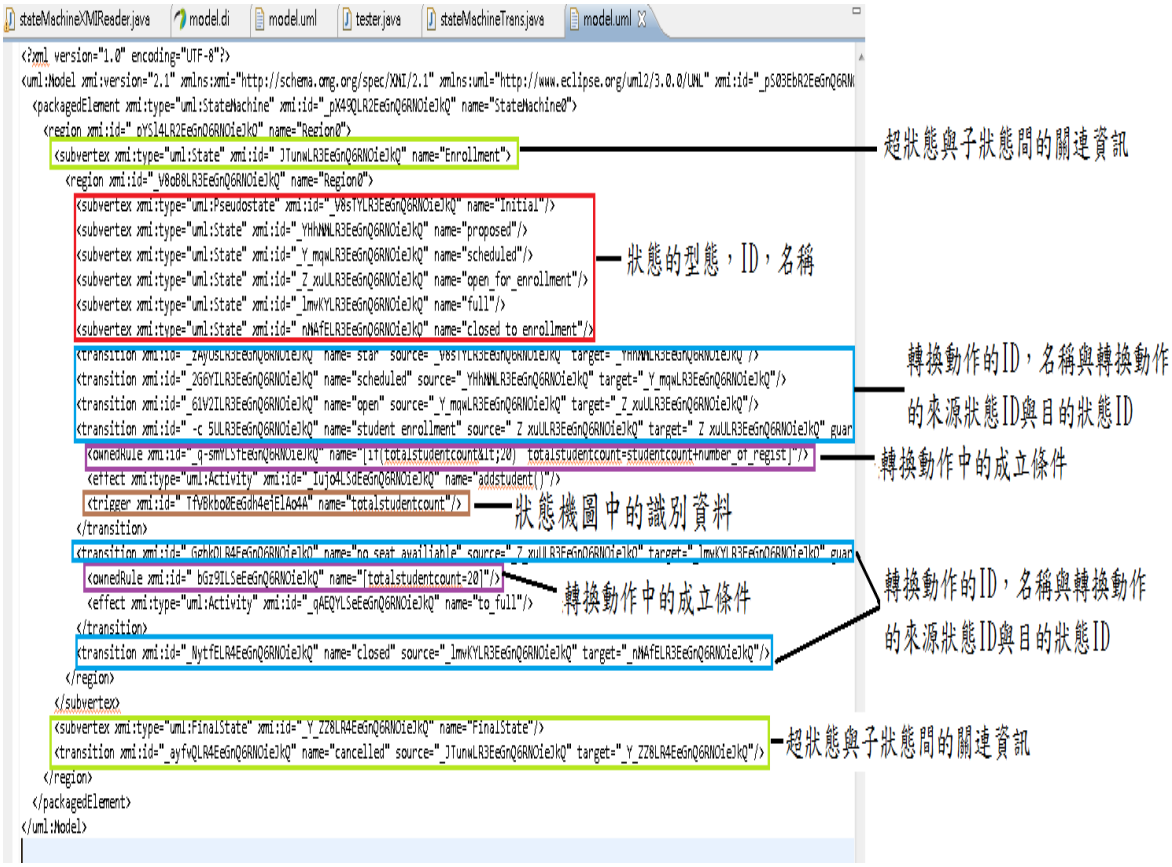


圖 11 使用 Papyrus UML 工具所產生之狀態機圖 XML 格式

在完成狀態機圖模型產生出其 XML 檔之後，便將.uml 檔放進本研究案例所實作的模型轉換工具，此轉換工具使用 eclipse 下的 jexcelapi(jxl)API 與 Simple API for XML(SAX)API 組成，我們使用 jexcelapi 自動產生編輯 MID 格式的 Excel 文件框架，再透過 SAX API 剖析狀態機圖 XML 文件，經由第三章介紹的轉換規則擷取要填入 MID 中 Model Description 欄位中相關所需要的資訊。轉換工具的程式碼畫面如圖 12，執行結束後則會在所指定的儲存位置產生出 MID Excel 格式文件。

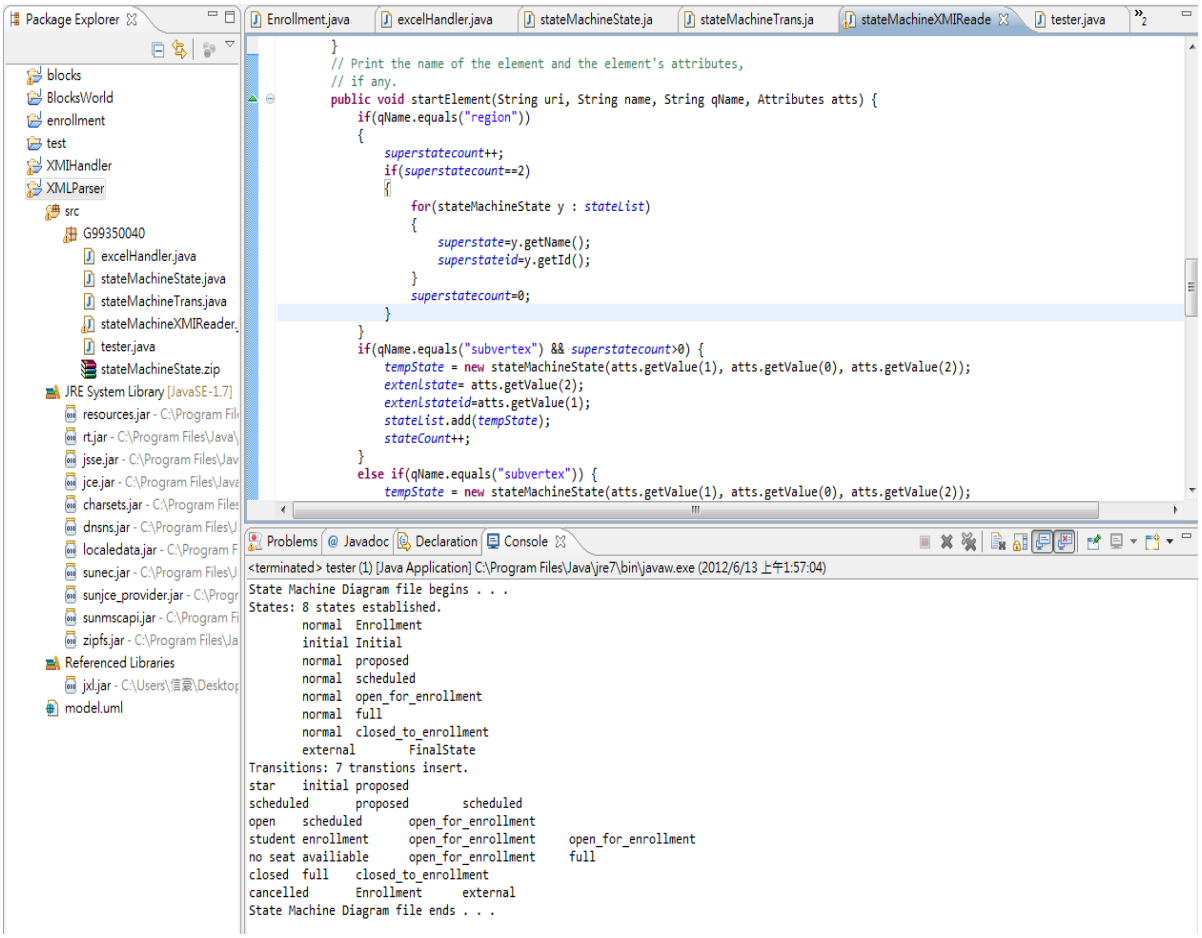
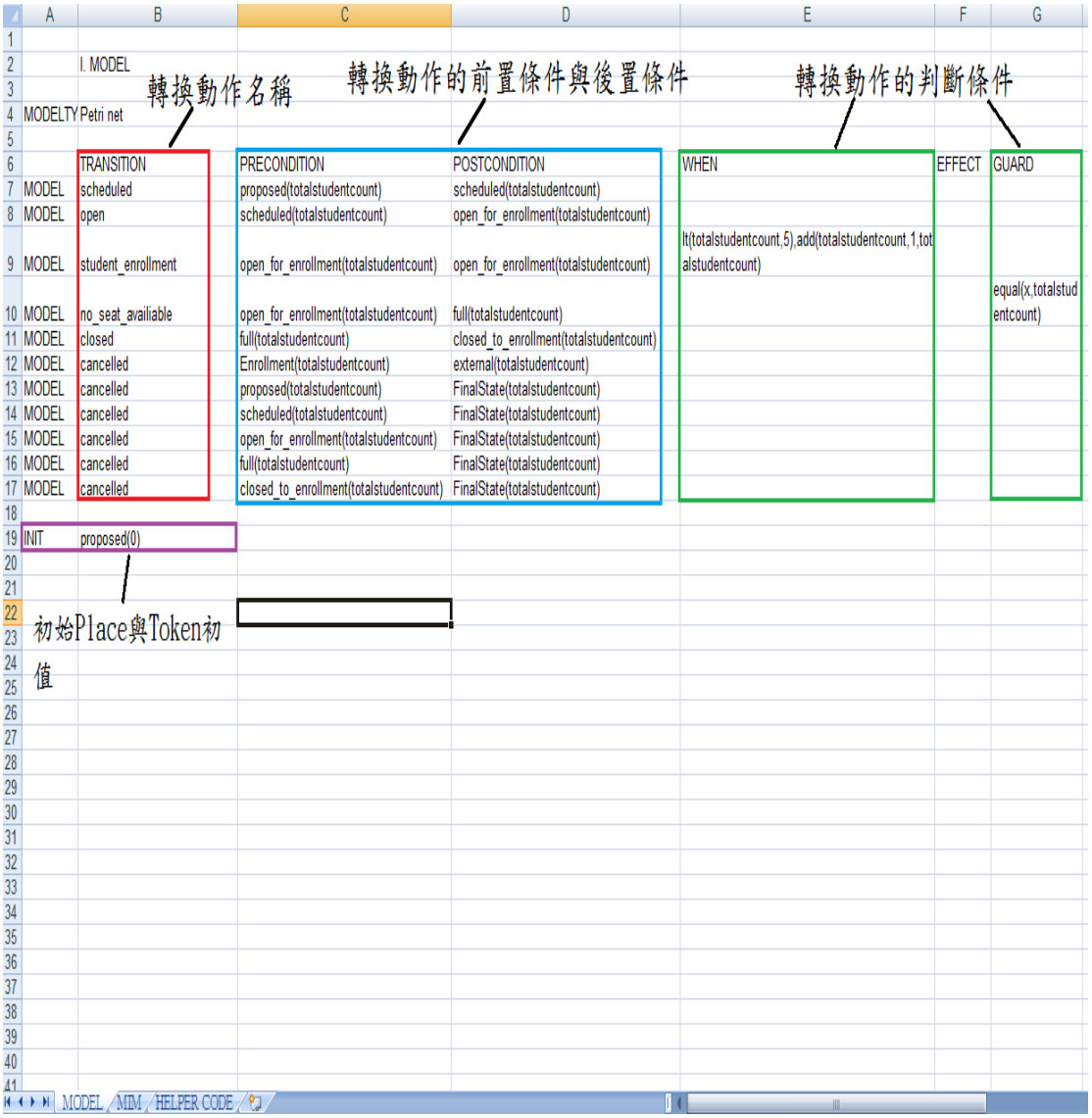


圖 12 模型轉換工具實作與執行結果

在模型轉換工具執行結束後，便會自動產生 Excel MID 格式，其中用來描述模型資訊的 Model Description 欄位會被自動填寫如圖 13，在圖 13 中紅色框線部分為轉換動作的名稱，藍色框線代表轉換動作的前置條件(precondition)與後置條件(post condition)，代表轉換動作的來源 place 與目標 place，其中各個 place 中的(totalstudentcount)即是 place 中的 token，綠色框線部分則代表轉換動作中的條件判斷式，紫色框線部分則代表標註整個流程起始的 place 與 token 數的初值(初值因無法從狀態機 XML 中得知，故需要設計人員手動填寫)，從圖 13 中的第七行的描述中，我們可以得知轉換動作 scheduled 的來源 place 跟目的 place 分別為 proposed 跟 scheduled，而 scheduled 這個轉換動作中並沒有條件判斷式，所以這個轉換動作是一定會被觸發，而第九行的轉換動作 student_enrollment 中有填寫條件判斷式，所以此轉斷動作必須滿足條件判斷式才會被觸發，而第 19 行的 INIT proposed(0) 則是標示 proposed 為起始的 place，token 的初值為 0。



	A	B	C	D	E	F	G
1							
2		I. MODEL					
3							
4	MODELTY	Petri net					
5							
6		TRANSITION	PRECONDITION	POSTCONDITION	WHEN	EFFECT	GUARD
7	MODEL	scheduled	proposed(totalstudentcount)	scheduled(totalstudentcount)			
8	MODEL	open	scheduled(totalstudentcount)	open_for_enrollment(totalstudentcount)			
9	MODEL	student_enrollment	open_for_enrollment(totalstudentcount)	open_for_enrollment(totalstudentcount)	lt(totalstudentcount,5),add(totalstudentcount,1,totalstudentcount)		
10	MODEL	no_seat_availiable	open_for_enrollment(totalstudentcount)	full(totalstudentcount)			equal(x,totalstudentcount)
11	MODEL	closed	full(totalstudentcount)	closed_to_enrollment(totalstudentcount)			
12	MODEL	cancelled	Enrollment(totalstudentcount)	external(totalstudentcount)			
13	MODEL	cancelled	proposed(totalstudentcount)	FinalState(totalstudentcount)			
14	MODEL	cancelled	scheduled(totalstudentcount)	FinalState(totalstudentcount)			
15	MODEL	cancelled	open_for_enrollment(totalstudentcount)	FinalState(totalstudentcount)			
16	MODEL	cancelled	full(totalstudentcount)	FinalState(totalstudentcount)			
17	MODEL	cancelled	closed_to_enrollment(totalstudentcount)	FinalState(totalstudentcount)			
18							
19	INIT	proposed(0)					

圖 13 Model Description 格式文件

在 Model Description 格式文件完成之後，還必須透過設計人員人工填寫編輯 MIM (Model-Implementation Mapping)與 Helper code 的格式文件，在圖 14 中 MIM 文件中紫色框線部分用來確認系統測試(System Under Test)的本體(Identity)，例如:網路應用程式的 URL，物件導向程式的 class name 或 C 語言程式的 system name 等，紅色框線部分代表轉換動作中的 Event/guard 判斷條件規則去對應到系統程式

碼中的方法(Method Mapping)，藍色框線部分則代表高階派翠網路測試模型中的 Place 對應到系統程式碼來驗證所描述 place 的狀態，而綠色框線部分則是對 token 做初始化動作的程式碼方法名稱。

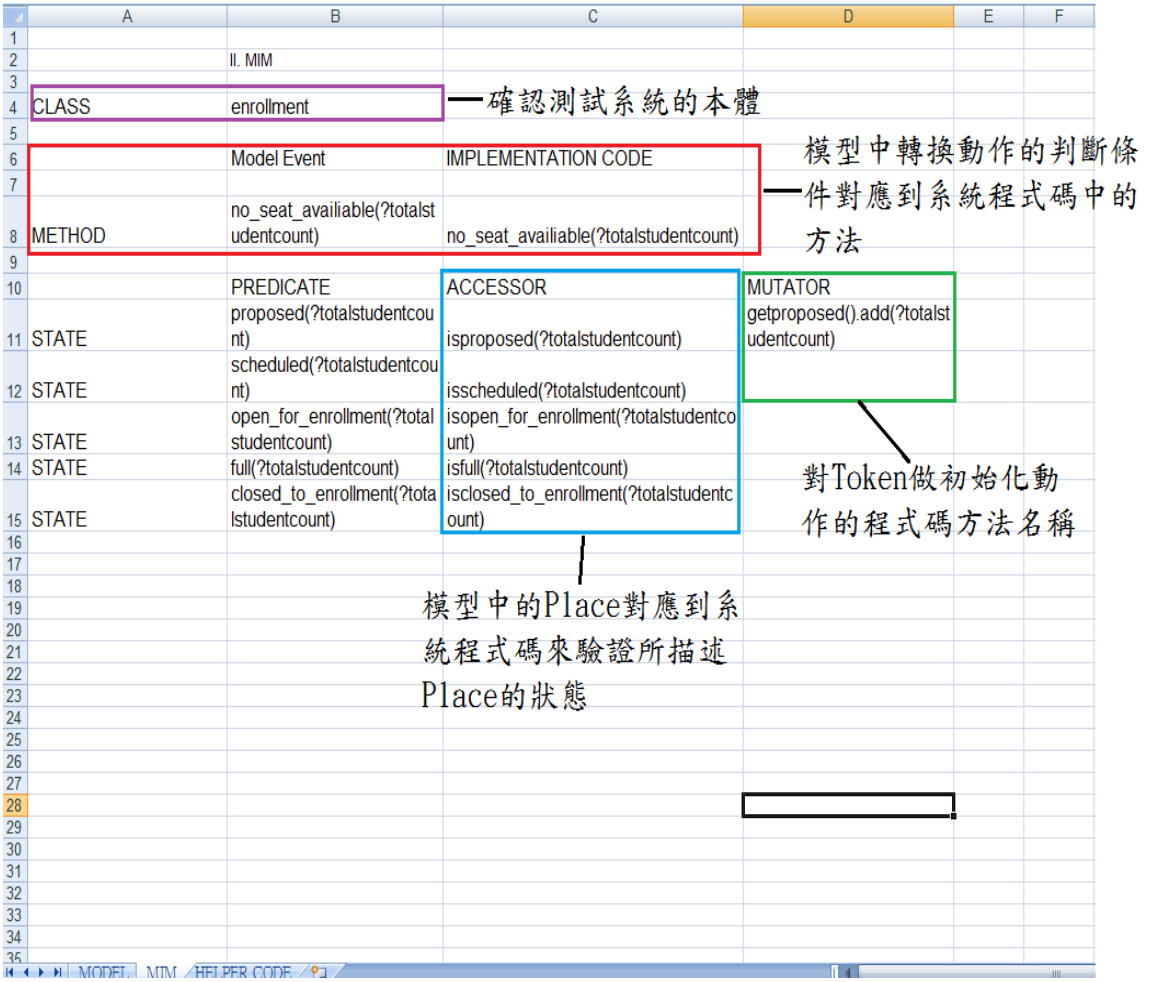
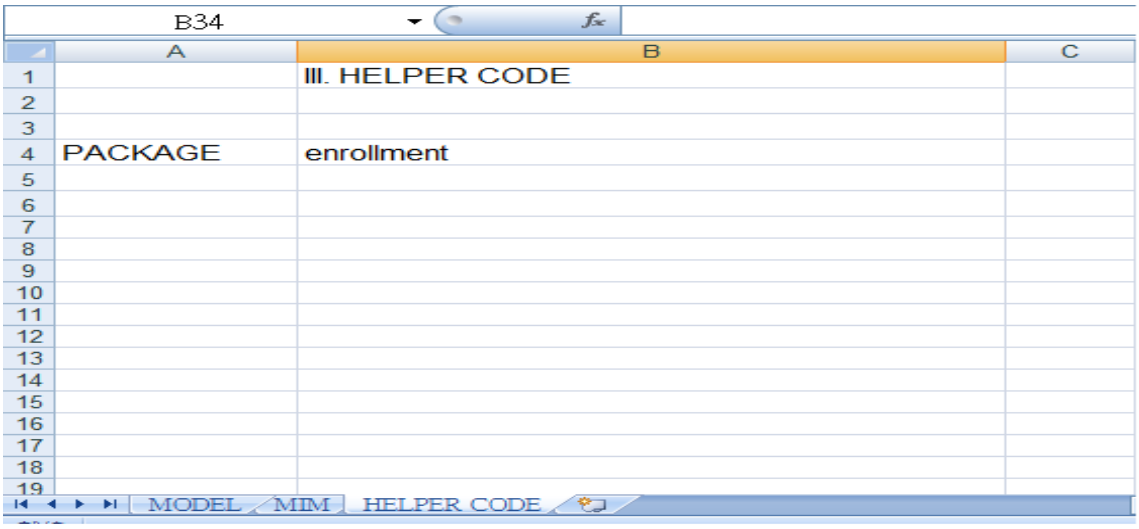


圖 14 MIM 格式文件

在完成 MIM 文件之後，接下來還須填寫 Helper code 文件，其目的是使其自動化產生出來的測試碼是能夠執行的，文件內包含標頭檔(header)例如:java 中的 package/import，C 的#include…等。

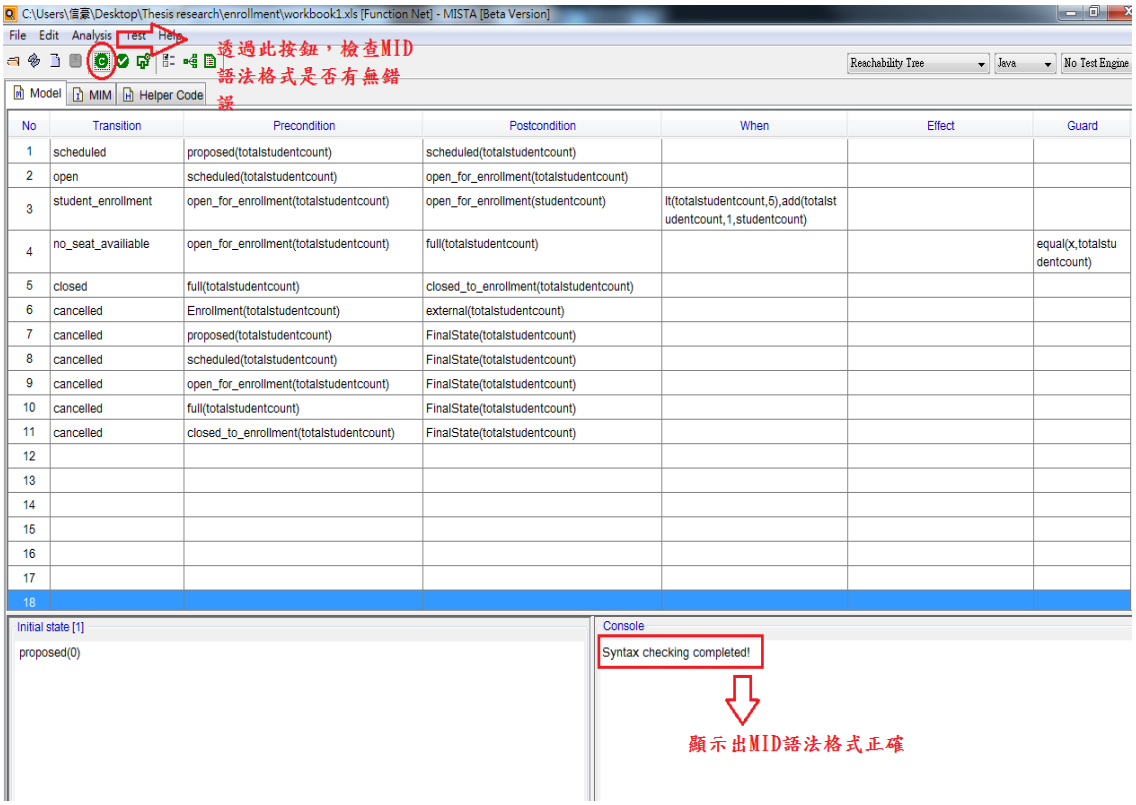
Helper code 文件如圖 15



	A	B	C
1		III. HELPER CODE	
2			
3			
4	PACKAGE	enrollment	
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			

圖 15 Helper code 文件

最後再將編輯好的 MID 格式文件匯入至 MISTA 工具，如圖 16



No	Transition	Precondition	Postcondition	When	Effect	Guard
1	scheduled	proposed(totalstudentcount)	scheduled(totalstudentcount)			
2	open	scheduled(totalstudentcount)	open_for_enrollment(totalstudentcount)			
3	student_enrollment	open_for_enrollment(totalstudentcount)	open_for_enrollment(studentcount)	lt(totalstudentcount,5),add(totalstudentcount,1,studentcount)		
4	no_seat_availiable	open_for_enrollment(totalstudentcount)	full(totalstudentcount)			equal(x,totalstudentcount)
5	closed	full(totalstudentcount)	closed_to_enrollment(totalstudentcount)			
6	cancelled	Enrollment(totalstudentcount)	external(totalstudentcount)			
7	cancelled	proposed(totalstudentcount)	FinalState(totalstudentcount)			
8	cancelled	scheduled(totalstudentcount)	FinalState(totalstudentcount)			
9	cancelled	open_for_enrollment(totalstudentcount)	FinalState(totalstudentcount)			
10	cancelled	full(totalstudentcount)	FinalState(totalstudentcount)			
11	cancelled	closed_to_enrollment(totalstudentcount)	FinalState(totalstudentcount)			
12						
13						
14						
15						
16						
17						
18						

圖 16 MID 文件匯入至 MISTA

MID 文件語法檢查並無錯誤之後，便可透過 MISTA 工具自動化產生測試案例樹如圖 17 與測試程式碼圖 18。

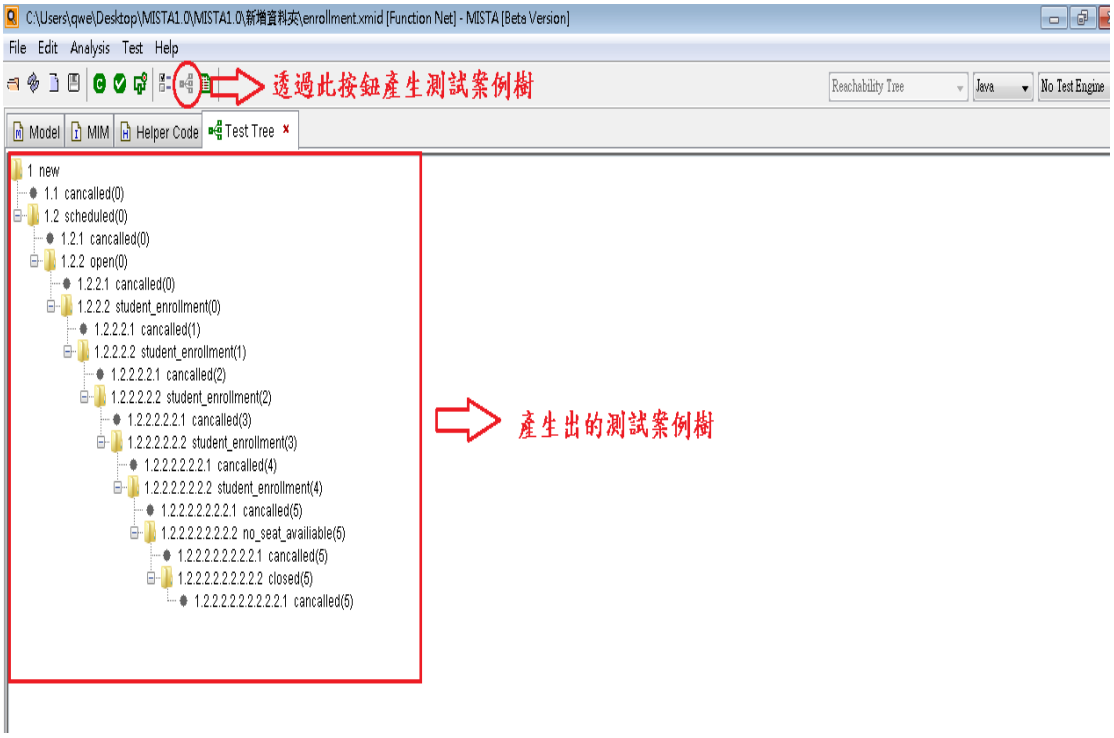


圖 17 測試案例樹

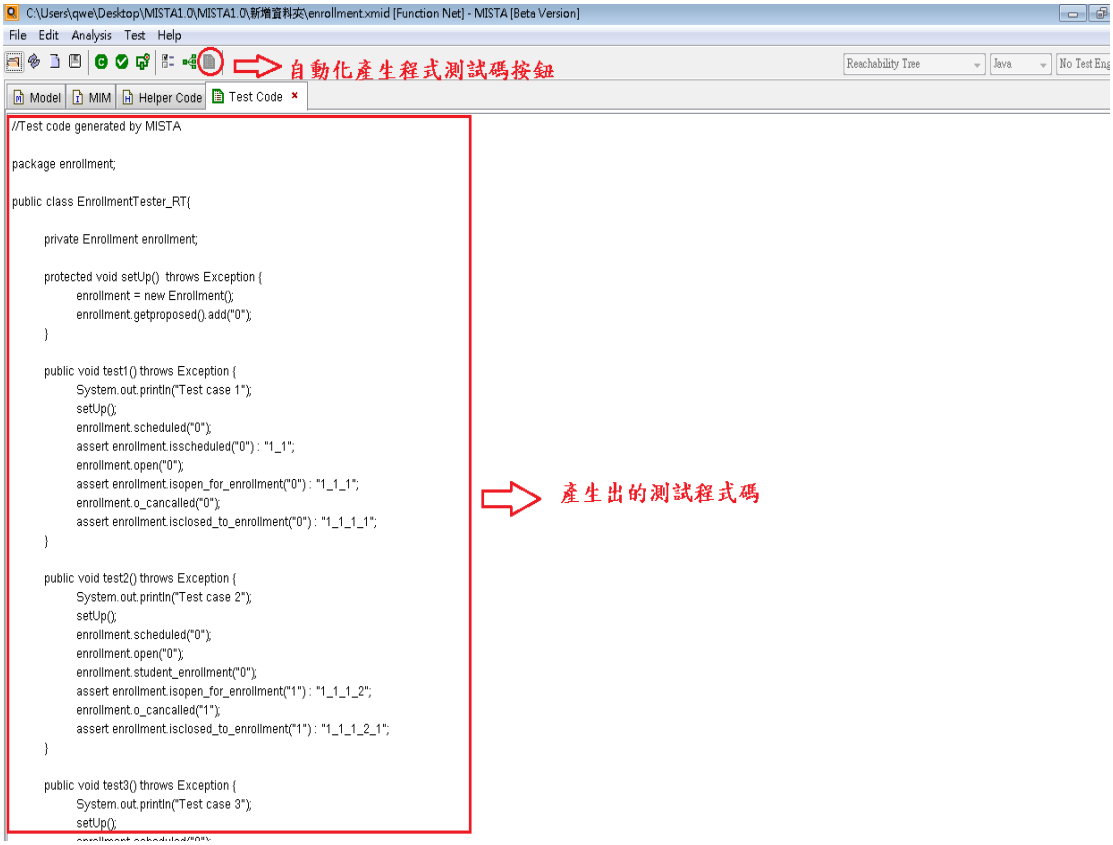


圖 18 自動化產生程式測試碼

在產生程式測試碼之後，會在來源資料夾裡面產生一個程式碼檔案，此案例是產生 java 程式測試碼，所以在來源資料夾裡會有一個副檔名.java 檔,將此檔案匯入進所要進行測試程式的資料裡如圖 19,即可執行 MISTA 所產生的測試程式碼，再執行程式測試碼之前，因為 MISTA 工具產生出的測試碼中使用 assertion 功能來判斷預期參數是否符合程式結果，故必須在 eclipse 底下做一個參數設定的動作，在 Run Configurations 下的 VM arguments 輸入-ea 開啟 assertion 斷言功能如圖 20，執行程式測試碼之後如果測試程式碼測試程式主體沒有錯誤的話，即會完整跑完所有測試案例如圖 21，但是如果測試程式碼測試程式主體並發現錯誤的話，即會跳出中斷訊息，並告知在由 MISTA 產生的哪一個測試案例所自動產生的代號路徑測試出程式結果發生錯誤，例如:我們故意將測試程式碼預期出的參數做改變，使其與程式跑出結果不相同，如圖 22，如此一來測試人員便可依此資訊檢查被測程式的正確性。

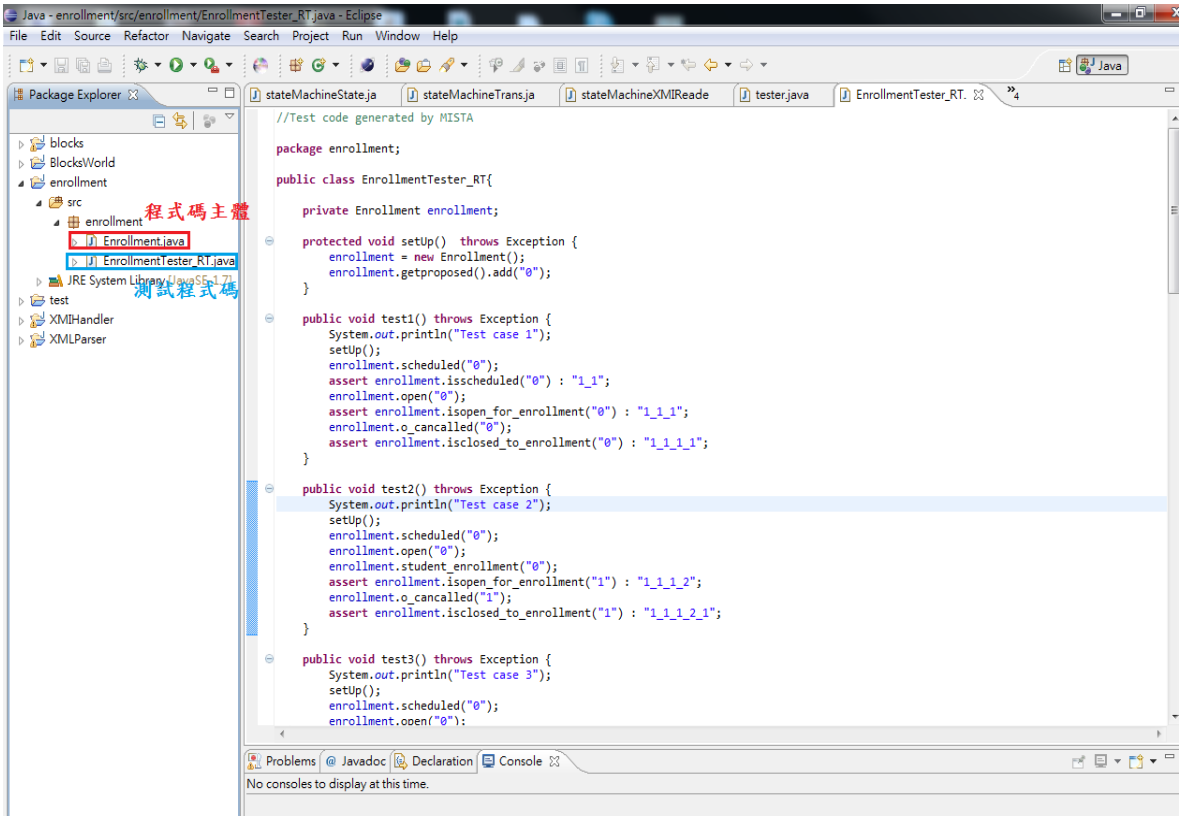


圖 19 測試程式碼畫面

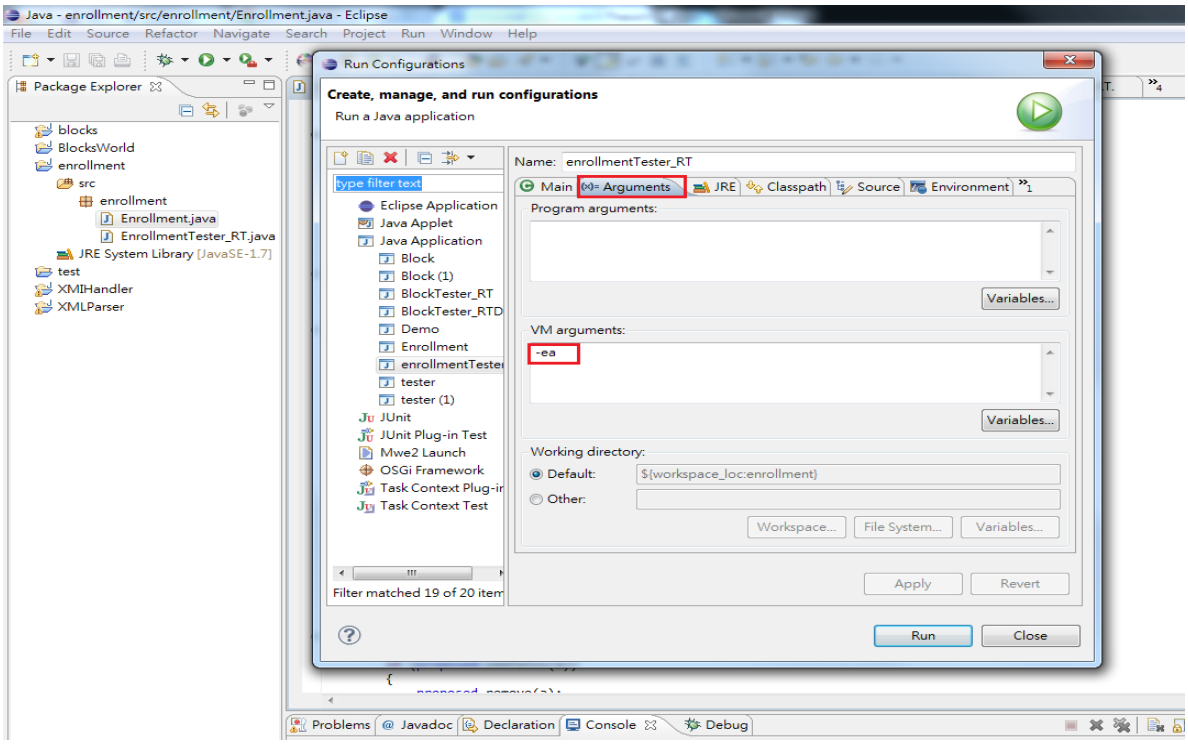


圖 20 assertion 功能開啟參數設定

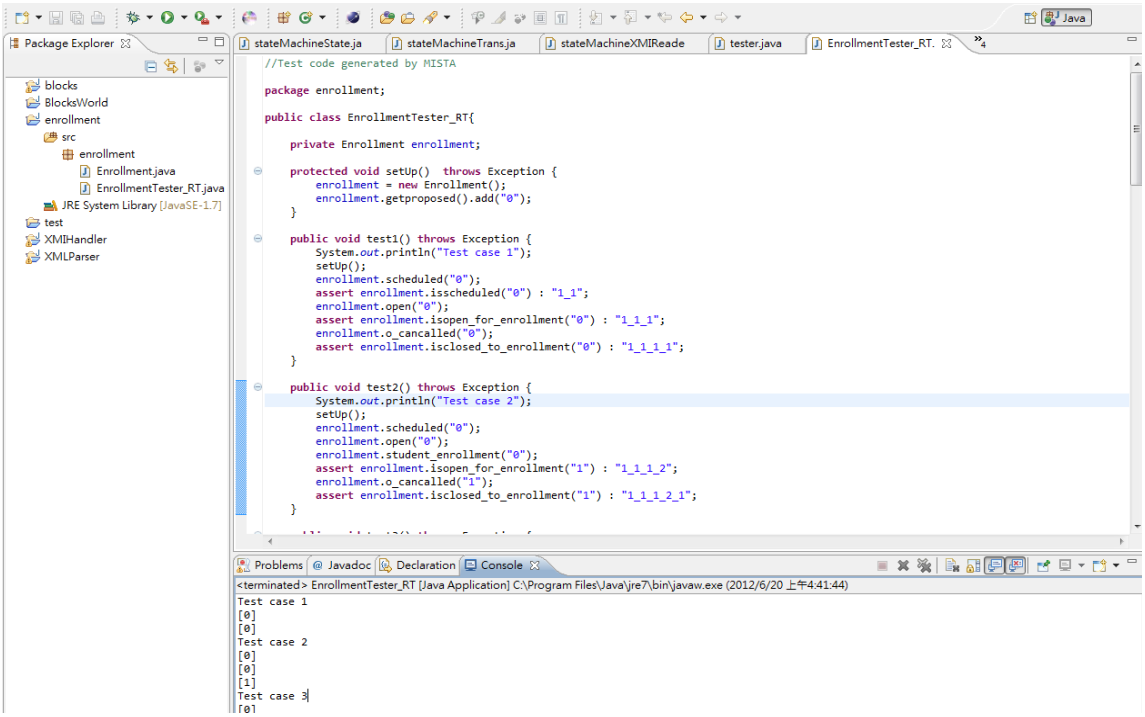


圖 21 測試程式碼沒發現錯誤下的執行結果

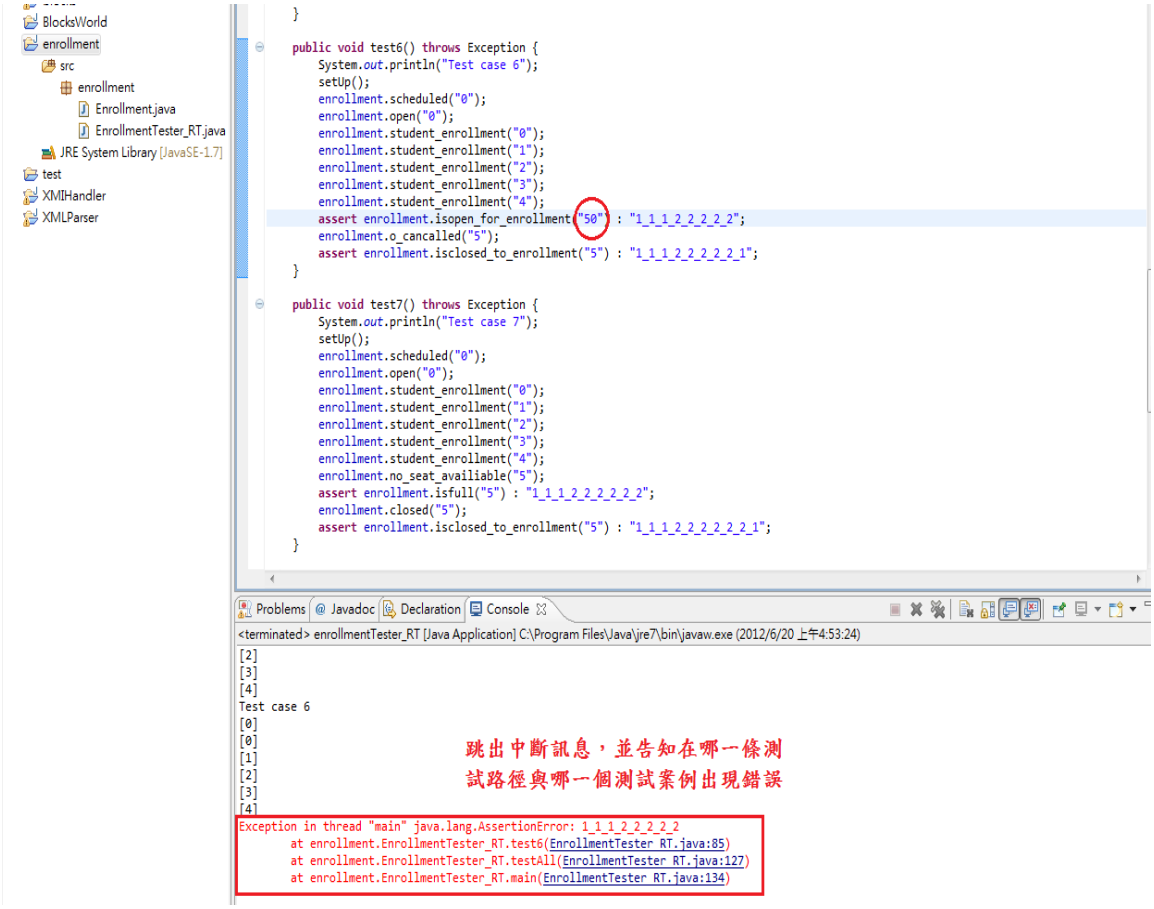


圖 22 測試碼預期參數與主體程式結果不一的錯誤畫面

第五章 結論

在現今軟體開發流程中，UML 已成為軟體塑模開發的主要塑模語言，加上模型驅動開發的興起，更加強調軟體開發過程中模型的建立，軟體開發流程到軟體測試階段便可透過這些模型的建立，再由基於模型測試方法使用行為模型針對系統行為預測產生測試案例和可執行的程式測試碼，在本篇論文裡引用基於模型測試自動化產生測試碼的工具 MISTA，此工具透過高階派翠網路測試模型來解決 UML 模型難以自動預測確定的實際參數測試序列的問題，但 UML 已成為軟體塑模開發的主要塑模語言，故本文提出 UML 模型中的狀態機圖轉換成高階派翠網路模型之對應轉換規則與實作的轉換工具，讓開發者可以在開發軟體流程中直接使用 UML 狀態機圖塑模，並透過本文的轉換工具自動化轉換成高階派翠網路模型銜接至 MISTA 自動化產生測試碼工具，如此一來便可以不用再另外花時間建構高階派翠網路測試模型，透過使用狀態機圖達到自動化測試碼產生，如此一來便可降低軟體測試所需花的人力成本，測試碼撰寫時間成本與降低人為測試失誤的機率。

在未來可接續發展與研究中，可以透過解析系統程式碼，進而從系統程式碼中將所需資訊截取出來自動化填寫進 MISTA 輸入格式中的 MIM 與 Helper Code 部分來取代目前還需要手動填寫的步驟，使

其自動化程度更往上提升，也可透過 UML 中的其他模型如活動圖等資訊，使 UML 中的模型轉換至高階派翠網路測試模型的過程中資訊更加完整。

參考文獻

[1]. D. Xu, W. Xu, W.E. Wong, Automated Test Code Generation from Class State Models. *International J. of Software Engineering and Knowledge Engineering* 19(4), 599–623 2009.

[2]. M. Utting, B. Legeard, *Practical Model-Based Testing: A Tools Approach*. Morgan Kaufmann, San Francisco 2006.

[3]. B. Selic, "The Pragmatics of Model-Driven Development", *IEEE Software*, Vol. 20, No. 5, pp. 19-25, 2003.

[4]. C. Ebert, "Model-Based Testing", *IEEE SOFTWARE*, January/February 2012.

[5]. D Xu. "A tool for automated test code generation from high-level Petri nets." *Proc. of the 32nd International Conference on Application and Theory of Petri Nets and Concurrency (Petri Nets 2011)*, Newcastle upon Tyne, UK, June 20-24, 2011.

[6]. OMG, "Unified Modeling Language: Superstructure", Version 2.3, 2010.

[7]. R.H. Bourdeau and B.H.C. Cheng, "A Formal Semantics for Object Model Diagrams", *IEEE Transitions on Software Engineering*, Vol.21, No.10, pp.799-821, 1995.

[8]. OMG, "UML 2.0 Diagram Interchange Specification", Version 1.0, 2006

[9]. V. Rijsbergen and C. Joost, *Information Retrieval*, Butterworths, pp.24-25, 1979.

[10]. XML Metadata Interchange (XMI) Version 2.1.1, http://www.omg.org/spec/XMI/2.1.1/PDF/index.htm, Retrieved

2010-7-29.

[11]. T Murata, “Petri Net: Properties, Analysis and Applications, ” *Proceedings of The IEEE*,Vol.77,NO.4,April 1989

[12]. Committee Draft ISO/IEC 15909, "*High-level Petri Nets-Concepts*, Definitions and Graphical Notation", Version 3.4,1997

[13]. D. Xu, W. Chu, “A Methodology for Building Effective Test Models with Function Nets”, *Proc. of the 36th IEEE Computer Software and Applications Conference (COMPSAC’12)*, Izmir, Turkey, July 2012.

[14]. D Xu, R.A. Volz, T.R. Ioerger, and J Yen, Modeling and Analyzing Multi-Agent Behaviors Using Predicate /Transition Nets, *International Journal of Software Engineering and Knowledge Engineering*, vol. 13, no. 1, pp.103-124, February 2003.

[15]. H.J. Genrich. “Predicate/transition nets.” Petri Nets: Central Models and Their Properties, 207–247, 1987.

[16]. D. Xu, M. Tu, M. Sanford, L. Thomas, D. Woodraska, and W. Xu. “ Automated security test generation with formal threat models.”IEEE Trans. on Dependable and Secure Computing. In press.

[17]. D. Xu and K.E. Nygard. “Threat-driven modeling and verification of secure software using aspect-oriented Petri nets.” IEEE Trans. On Software Engineering. Vol. 32, No. 4, pp. 265-278, April 2006.

[18]. Smartesting: CertifyIT v5.1

http://www.smartesting.com/index.php/cms/en/home(July 28)

[19]. Conformiq Designer: Conformiq Designer v4.4 http://www.conformiq.com/(July 28)

[20]. Imbus: Tedeso 3.0 http://www.imbus.de/english/produkte/imbus-testbench/modules/managed-model-based-testing/(July 28)

[21]. Elvior: TestCast Generator Beta http://www.elvior.com/motes/generator(July 28)

[22]. Micrsoft Research: Spec Explorer 2010 http://research.microsoft.com/en-us/projects/specexplorer/(July 28)