私 立 東 海 大 學
資 訊 工 程 研 究 所
碩 士 論 文

指導教授：朱正忠 教授

以代理人為基礎之安全監控技術用於購物網站
Agent-based Security Monitoring Technologies for On-line Shopping System

研究生：李守彧

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 李 守 彧 所提之論文

以代理人為基礎之安全監控技術用於購物網站

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 王豐堅 簽章

委 員 [illegible]

[illegible]

[illegible]

指 導 教 授 [illegible] 簽章

中華民國 101 年 6 月 29 日

摘要

網路購物是一種方興未艾的商業模式，其結合了許多現代生活的便利與機動，為許多小額資本創造更多機會。然而，過去的技術沿用，帶來了許多陳舊的問題。然而在軟體開發的經驗中，安全性常常在實際應用被屏除，成為一個獨立且非必要的學科，導致訓練及經驗不足之工程師忽略相關議題或是沿用有缺陷的程式碼，給予有心者進行破壞的空間，進而造成許多企業級的軟體開發出現不必要的安全威脅。在本篇論文中，將利用一個架構於雲端的代理人模組，提供企業在進行商業交易行為的介面上，一個可能的安全性解決方案。此模組將提供以下的特性：

1. 代理人可以照使用者自定的規則進行測試與定義輸出文字紀錄檔。
2. 提供一個代理人管理平台，讓不間斷測試形成一個監控模組。
3. 提供知識庫儲存已使用的規則以作為未來使用。
4. 代理人介面提供可攜性，可配合系統內容做移植。

此模組預計做為建構成一個半完整的監控執行程序之基礎，結合其他既存的知識，提供購物網站安全性管理的一個解決方案。

關鍵詞: 網路購物、網路服務、Java EE、智慧型代理人、系統監控

Abstract

Internet shopping is a business model which combines the convenience of modern life and mobility, and creates more opportunities for many small capitals. However, the old pattern of technologies in use brought to many security problems that still exist. Also, while security is often an issue easy to be undercover, the lack of training and experience for the engineers often result in unnecessary security threats in enterprise-class software development. In this paper, a framework for agent module, acts as a possible security solutions. This module will provide the following features:

1. An agent can test according to the rules and output documents that users defined.
2. An agent management platform to allow uninterrupted test of a monitoring module.
3. Knowledge base to store rules for future use.
4. Agent interface provides portability, with the system content easy to transplant.

This module is expected to construct procedures for monitoring with existing technology, to provide a solution of the security management of a shopping site.

Keywords: On-line Shopping, Web Service, Java EE, Intelligent Agent, System Monitoring

誌謝

兩年寒暑，一眨眼就過去了。每天看著日出日落，雖難以感受季節變化，心底的回憶，卻是點滴在心頭。來到東海兩年，很少痛苦，大多是溫暖與歡笑。

我很榮幸接受朱老師的教導，他的好脾氣和豐富學養不知道幫助了我多少次。系上的老師們，總是不厭其煩地解釋講義的細節。不辭辛勞遠道而來的志宏學長與瑞男學長，從你們身上我學到了不少。IS Lab 和 HPC Lab 的同仁，有你們一起奮鬥真好！住隔壁的軟工中心同仁，雖然有時候會丟事情過來，更多的是幫我們處理不少麻煩事。實驗室的總管東暘，大小事一把抓，讓我們省了不少時間在研究上。堪稱萬能的廷肇，你一直是本實驗室的支柱，靠你我們才有今天。實驗室最潮的信豪，很享受跟你一起討論課業和打球的日子。實驗室唯一的女生景宜，你是我們的經濟來源！實作很厲害的學弟羿杰和柏涵，你們讓實驗室專業了不少。還有已畢業的學長們，你們教了我很多，不管好的壞的。最後還是感謝我的家人，在我最困難的時候拉了我一把，讓我走在正確的道路上。願大家都能有顆平靜的心靈。

章節目錄

摘要.....I

Abstract..... II

誌謝.....III

章節目錄.....IV

圖目錄.....VII

表目錄.....VIII

第一章　導論.....1

1.1. 前言.....1

1.2. 研究動機.....2

1.3. 研究目的.....3

1.4. 章節安排.....4

第二章　背景知識與相關研究.....5

2.1. 網路購物之安全問題.....5

2.2. 黑箱測試(Black-Box Testing) 7

2.3. 基於可再用元件的軟體工程.....8

2.4. 智慧型代理人.....9

2.5. Enterprise Java Bean.....10

2.6. Java Persistence API.....11

2.7. JBoss Enterprise Application Platform 12

第三章 研究方法.....14

3.1. 系統模型.....14

3.2. 智慧型代理人.....16

3.3. 測試知識庫.....19

3.4. 應用元件介面.....21

第四章 系統的設計與實作.....23

4.1. 系統規格.....23

4.1.1. 系統環境.....23

4.1.2. 開發環境.....23

4.2. 系統流程.....23

4.3. 使用者介面.....25

4.4. 代理人工廠.....26

4.5. 知識庫實作.....28

4.6. Enterprise Java Beans 與再利用元件.....29

第五章　案例研究與評估.....31

5.1. 購物網站實例.....31

5.2. 初始化代理人.....32

5.2.1 成本檢查代理人.....32

5.2.2 存貨檢查代理人.....33

5.2.3 登入動作檢查代理人.....33

5.3. 測試知識庫學習.....36

5.4. 評估.....37

第六章　結論與未來工作.....39

參考文獻.....41

圖目錄

圖 一 NEGOTIATION BROKER (NB) FRAMEWORK 9

圖 二 JBOSS ENTERPRISE MIDDLEWARE 13

圖 三 系統架構 16

圖 四 代理人運作圖 17

圖 五 知識庫應用示意圖 20

圖 六 代理人介面示意圖 22

圖 七 流程圖 24

圖 八 管理介面圖 26

圖 九 選擇新創 RULE 或使用知識庫 27

圖 十 創造代理人之介面 28

圖 一一 購物網站實例 31

圖 十二 選擇存入之規則 37

表目錄

表 一 規則語法範例 19

表 二 知識庫格式範例 21

表 三 介面架構表 29

表 四 成本規則 33

表 五 庫存規則 33

表 六 登入動作規則 34

表 七 成本運算結果 34

表 八 庫存運算結果 35

表 九 登入監控結果 35

第一章　導論

1.1. 前言

網路商務近年已趨近於成熟，無論是 B2B 或是 B2C 的運作，都藉由網路的便捷性與普遍性的擴展，在業界已儼然成為一種主流，網路購物即為其中一個顯然的例子。網路購物的發展，已讓客戶可以不用出門，即可於線上瀏覽應有盡有的商品，充分享受購物的樂趣。對於銷售方而言，其內部管理，如採購程序等，透過網路，更能便於整合企業之既有系統，如倉儲、客戶系統(Customer relationship management system)等。而對於一般小資本的賣家，網路商店更省下了實體店面的成本，如再與物流服務做結合，可以有效率的鋪貨至全國範圍。網路的流通性，亦可以讓貨品的流通性加大，促進資本的流動或企業規模發展，對於世界經濟不可不說相當重要。

然而，在現有的運作機制中，人為操作是不可避免的一環。因為網路程式需透過網頁做為介面，在此介面之下，包含了客戶與管理者兩方面的使用者的操作。而這些使用者人為的操作，產生了許多誤操作(disoperation)或誤用(misuse)的安全性問題：包括無意的錯誤資訊(ex.商品標價錯誤)或是惡意的攻擊(ex. SQL injection)，直接或間接地造成企業不可計數的損失，或是造成個資流出，給予詐騙集團等犯罪

者從事非法活動的空間。

為了避免許多嚴重的安全性問題，政府已經採取了許多手段防治。但這些由上而下，透過警政機制的防治，終究不過是被動的，可以減少犯罪產生，無法釜底抽薪，準確的解決問題。因此，對於網路程式的開發者而言，應從兩個方向來解決這些網路程式的安全問題，以避免未來造成的損失：1. 正確的程式開發流程，在開發階段找出問題並改善。2. 事後的完善維護，除了管理的完善流程，還要有錯誤改善的機制。

在過去，已有許多相關的方案被提出，其中不乏對症下藥者，在許多框架中扮演安全性的重要部分。但是這些機制，大多都無法到達完善，或是可移植可再利用的元件開發。能夠發展一個不但有效，也能成為標準容易開發的安全管理方式，將是未來在網路商務雲端化後，持續努力的目標。

1.2. 研究動機

現有的網路程式架構與開發框架，主要提供的安全性機制都僅限於開發階段，並過於簡略。如果開發者有安全性上的維護需求，如保護客戶認證、商業表單等等資訊，通常都無法在上線操作(On-the-fly)的階段達成，而僅能透過特定的工具，甚至是人為對於程式碼的檢視。

這樣的作法沒有效率，不但消耗人力，而且容易有忽略誤判的情形。

本論文將希望將網站之程式維護與安全管理，找出一個可能的切入點，建立未來網站應用程式之維護與管理的標準流程(Standard operation procedure)。此流程將包括自動化的測試，與客製化的規則，並將系統組合而成為一個簡單的監控機制。此系統包含的元件，也可以透過一個完整且分散的方式，讓開發者只需要透過簡單的更改，便能適用在其他相關的環境。

在本論文中，此方案將針對於一個網路購物的環境，設計其運作步驟與方式，作為一個應用實例的代表，希望能從中找出可行的方式，並演示其應用的可能性與發展性。

1.3. 研究目的

本論文的目標，在於創造出一個自動、即時、無態(Stateless)並且可再利用的代理人，並應用在一個專為商業系統的監控模型。本論文將透過實作一個網路應用程式，結合架構平台，讓測試程序成為一個不間斷且自動化的過程，留下使用者定義的紀錄，讓此模組形成一個自動化的監控代理人，而不需依賴管理者在場。

本論文也將透過 Java EE 中的 Enterprise Java Beans 技術，達到元件獨立的成果，讓未來的開發者可以使用本論文中的代理人邏輯，建

構在新的系統與理論中。

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，說明過去與本論文相關的研究，以及參考之方法與資訊。

第三章 概述本論文中，基礎理論與做法，並解釋做法的用意與效用，描繪實作系統的概述。

第四章 系統的設計與實作，說明系統的實際架構、在雲端環境下運作方式以及實際操作方式，檢視第三章所述的目標。

第五章 案例研究，以所建構之雛型系統，套用到一個常見的商業案例，並展示功能與介面。

第六章 結論及未來方向，將本論文之研究成果做總結，舉出優點與缺失，並提出未來研究方向。

第二章　背景知識與相關研究

在本章中，整理與本論文相關之相關背景知識。在第一節，解釋網路購物之內容與可能的安全問題，以釐清本論文中將要解決的問題方向。第二節解釋黑箱測試的原理，與其優缺點。第三節解釋再利用元件的重要性與一般方法。第四節解釋代理人的定義，與本論文參考的相關應用。第五節與第六節將概略解釋本論文將使用的關鍵技術：Enterprise java bean 與 Java persistence API。最後第七節則解釋 本論文所使用的平台架構：JBoss enterprise application platform。

2.1. 網路購物之安全問題

網路購物[1][2][3][4]是一種電子商務的形式，讓消費者直接從賣方通過網路購買商品或服務。如一名客戶透過網路商店向背後的實體商店(Bricks and mortar stores)或購物中心購買貨品，這個過程被稱為企業對消費者(B2C)的網路購物；當一個企業購買透過網路向另一家企業採購，它被稱為企業對企業(B2B)的網上購物。

在一般的情況下，消費者藉由訪問零售商網站或搜尋供應商的購物搜索引擎，尋找感興趣的產品。網站上的購物車，可以讓消費者積累多個欲購買的商品項目和數量。結帳時，付款方式(包括信用卡、銀行支票等)與和寄送資訊會被收集與確認，有些商店會允許消費者

註冊一個永久性的網路帳戶，以保留付款與寄送等相關信息，而不需在每筆交易重覆輸入。交易完成後，消費者會收到確認，通常是通過電子郵件。另外也有較簡單的交易方式，透過消費者主動透過電話或電子郵件提供訂單，只是由於安全性的因素，信用卡號碼無法通過這種方式傳送。

網路購物不但提供更便利的購物環境，也提供更廣泛的選擇。另外由於開店成本相較於實體店面遠為低廉，也可以提供較低的價格，增加競爭力。

根據 iThome 網站之專文[5]，購物網站常見之安全威脅可分為六大類：

1. 使用者威脅

   包括使用者習慣不良、惡意程式感染與網路詐騙、釣魚等。

2. 人員威脅

   網站之使用管理與開發時期之管理不當。

3. Web 威脅

   功能設計有錯誤、漏洞。管理員或使用者之身分驗證與權限，資料保護不全被駭客破解。

4. 系統威脅

   系統存在漏洞，或是架構不良。如錯誤處理或共用資源。

5. 內部威脅

機器中木馬或網路架構問題被開後門。

6. 關聯網站

不同企業中網路或是下流權限控管問題，造成個資洩漏。

這些問題可以透過強化軟體開發流程、提供架構的權限控管、使用資安的產品和將使用者納入安全防護圈等方式解決。

2.2. 黑箱測試(Black-Box Testing)

黑箱測試(black-box testing)[6][7][8][9][10][11]是將程式元件當作一個黑色無法透視內部機制的箱子，只依照表現的功能效果，測試是否正確地達成任務。測試人員並不需要對軟體的結構性有足夠深層的瞭解，只須進行著重於功能面的測試，因此又名功能測試。黑箱測試目標著重在驗證功能性的需求、介面的運作與程式初始與終結的正確性。測試案例(Test case)的設計則依據使用者需求之規格設計，以了解程式是否滿足特定使用者需求，測試人員的作業則必須要按測試案例來逐一進行，因此測試案例的設計好壞會直接影響到測試結果。

黑箱測試的優點為：

- 測試案例設計成本較低廉。
- 需求規格導向對於使用者觀點較為實用。

- 不易受到程式架構、內容與實作方法等影響而產生偏見。

黑箱測試的缺點為：

- 需要大量的測試案例以確保涵蓋率(coverage)—測試案例能夠涵蓋程式行為的比率。
- 大量測試案例的設計容易造成功能性重複，造成時間資源的浪費。

2.3. 基於可再用元件的軟體工程

對於軟體開發過程中，基於可再用元件的軟體工程(Reuse component-based software engineering) [12][13][14][15][16]是另外一個值得重點研究的領域。相較於硬體系統開發，軟體開發過程其時程及品質掌控均嫌不足，深究其原因在於傳統軟體開發多採 one-off，造成經驗及成果無法累積，可再用元件技術便是萃取先前所開發軟體系統中成熟之軟體模組，成為可再用的軟體元件，再建立新系統時可以直接採用或修改，建構出新軟體系統。在[17][18][19][20]中，我們提出對於可再用軟體元件之擷取、分類、正規化表示、擷取、整合、調適的問題及相關可行處理方式。而在[21]上，一個完整四階段的 reuse 架構：包含 Architecture level、Design level、Code level 和 Data level 也被提出作為再用元件之運用的完整概念。

2.4. 智慧型代理人

智慧型代理人[22][23]是提供人機交流界面的軟體，來做為使用者在特定工作上的助理，而非單純的工具。雖然其智慧不一定能比美真正的人工智慧，代理人卻可以從與用戶的互動中，自發性的學習，並透過推理預測用戶的使用模式。

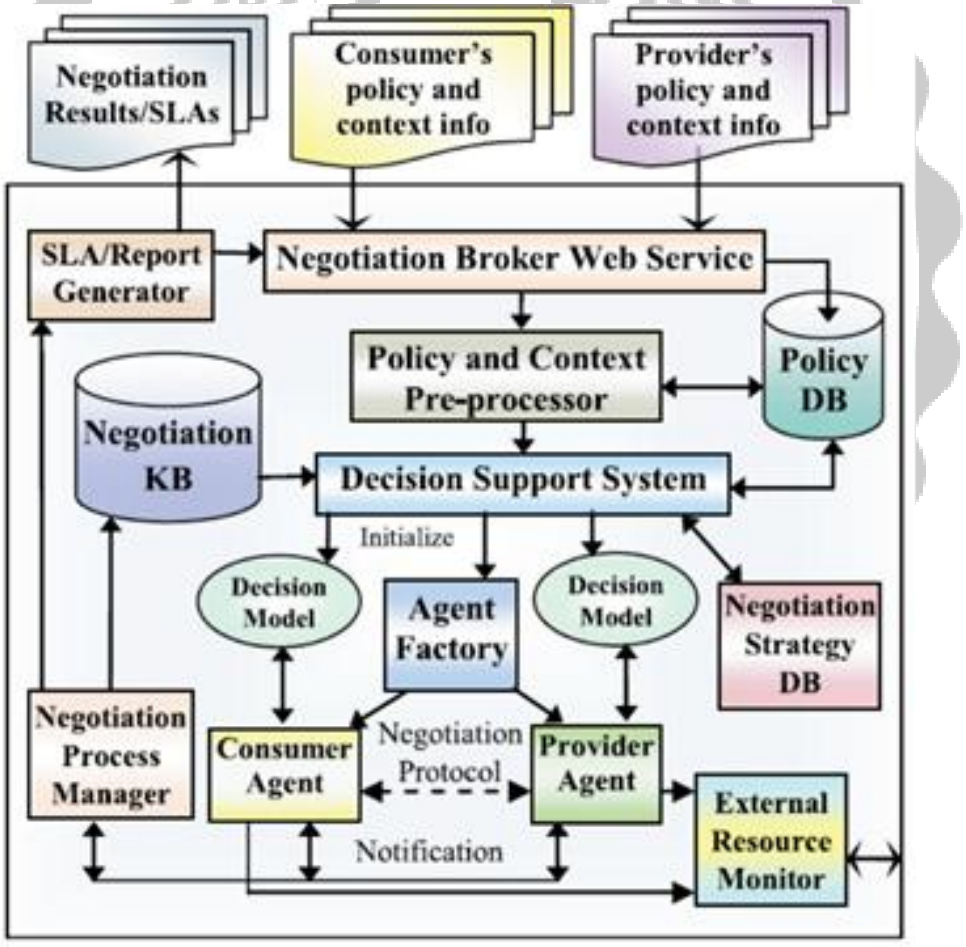

圖 一 Negotiation broker (NB) framework

Zulkernine, et al. [24][25][26]亦指出針對異質的網路環境與元件，

需要一個自主運算的方法，提供伺服器與客戶端進行 SLA 的協調。該研究進一步的將此構想實作而成為一個 Negotiation broker (NB) 框架，其結構如圖 一。該框架含有一個決策輔助系統(Decision support system, DSS)，根據定義的 policy 檔案，透過決策方程式，算出一個價值(Utility value)最高的方案。最終會根據每次的協調策略，存入一個策略資料庫中，做為未來策略選擇的參數。該框架的優點是運用了自動協調的智慧代理人(Intelligent agent)和具可塑性的系統，可以根據使用者的需求做適當調整。不過在文意的辨識與多邊協調的方法沒有確切的解答，可能將會透過策略學習模組或可信賴模組來解決。

2.5. Enterprise Java Bean

Enterprise Java Bean(EJB)[27][28]，是一個用 Java 程式語言編寫，封裝了應用程序的商業邏輯，分散式的伺服器端元件。商業邏輯包括可以實現特定應用目的之程式碼，遠程客戶端可通過調用這些方法，訪問庫存之應用程式。

EJB 分成三種種類：

- Session beans：代表 J2EE 伺服器內的單一客戶。訪問伺服器上部署的應用程式時，客戶端程式會為用戶調用 Session beans 的方法，減少從伺服器內執行商業任務的複雜度。

- Message driven beans：作為一個接收 message 的事件偵聽器，允許 J2EE 應用程式非同步處理訊息。Message 可以來自任何 J2EE 元件、JMS 應用程序或不使用 J2EE 技術的任何系統。Message driven beans 目前只能處理 JMS 消息，但在未來或許可以擴充處理其他類型的 message。
- Entity beans：代表業務對象如客戶、訂單和產品等的持續存儲機制。在通常情況下，此機制是一個 J2EE SDK 中的關係資料庫，Entity beans 對應資料庫中的一個表單，而每個 bean 實例對應於該表單中的一行。

2.6. Java Persistence API

Java persistence API[30]是在 Java 語言中，用以管理有相關性資料的框架。該框架經常被應用在 Java EE 的平台上，尤其是與 EJB 等動態網路程式共同應用。一般定義之 JPA 包含三個面向：

- API 本身，定義於 javax.persistence 套件中。
- Java Persistence Query Language (JPQL)，將傳統 SQL 存取資料庫表單的方式化成存取實體。
- 物件/關係對應的資料處理詮釋資訊(metadata)，實現資料存取物件導向化的可能，並用於取代 EJB 中傳統的實體 Entity

beans。

目前屬於 Red Hat 企業平台架構下的 Hibernate[31][32]是應用此一概念的開源框架，並且為了 Java 實作了 object-relational mapping (ORM) 的函式庫，給 Java EE 一個使用將 Java 類別映射於傳統關聯性資料庫存取機制如 JDBC 等的方法，取代舊有資料存取機制，還提供物件導向的數據檢索機制，大大地減短開發商業邏輯的程式的時間。

2.7. JBoss Enterprise Application Platform

JBoss 企業級中介軟體平台[33][34][35][36][37]是一個涵蓋設計開發、運行佈署到維運提供的中介平台，穩定、並可應用於關鍵任務上，如美國紐約泛歐證交所(NYSE)在 JBoss 平台上執行最為重要的管控系統 StockWatch，每秒即時監督超過 100,000 個訊息， 圖 二為 JBoss enterprise middleware platform 架構。

JBoss enterprise application platform(JEAP)提供符合 Java EE 架構的平台，功能包括 JDBC、交易處理(JTA/JTS)、訊息服務(JMS)、目錄介面(JNDI)及管理框架(JMX)等，並能在運行中部屬新的程式，讓使用者不須重新運行來更新內部資訊。其內部應用包括，Hibernate、Seam framework、JBoss web framework kit，以及 HTTP/HTTPS 支援、

Virtual host 支援、SSL v2/v3 支援、TLS v1 支援、ANT 及命令列支援、Cold deploy/Hot deploy/Farm deploy 支援、SNMP adapter 支援等。

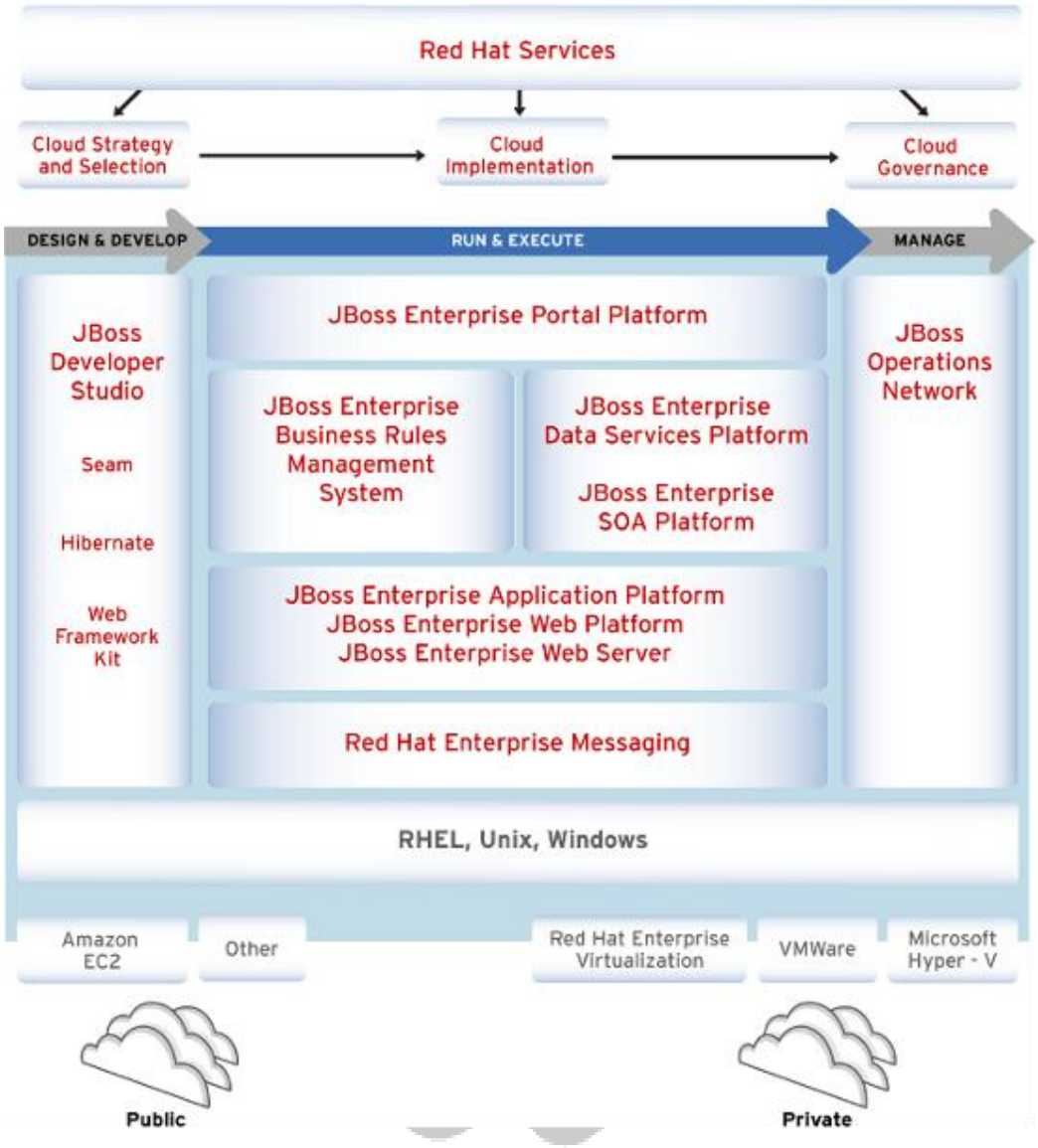

圖 二 JBoss enterprise middleware

第三章　研究方法

一般測試流程視專案規模而定，分為規劃、執行、追蹤與輸出結果四個階段。網路服務如購物網站的測試則需藉由操作外在的網路介面，作為輸入輸出的手段。對於這種介面依存的環境，在黑箱測試的手段中，如要達成自動化測試的目標，只能透過網路機器人(Webbot)、開發者自行定義的測試環境或是既有工具的操作。透過以上方法雖可適用於許多主流的網路程式架構，卻因為過於泛用化而有無法有效融入客製化之商業邏輯軟體，或是需要特殊的手段才有辦法達成開發者所需要的測試工作，在這個一來一往的試誤(try & error)工作之中，許多商業的產能便這樣耽誤了。

本論文中提出的「法則式代理人測試方法」，將應用 Java EE 之 Enterprise Java Beans 技術，著眼於提供購物網站一個適用的測試模組開發框架，讓開發者應用可再使用元件，建立適用於新系統的測試環境。另外，開發者與維護者可透過自定義之測試規則，其中可包含商業邏輯與軟體邏輯，使用代理人之技術，以達到無縫並且即時測試資訊，也可將測試策略存進一個知識庫，作為該系統或是將來升級系統後測試的依據。

3.1. 系統模型

作為一個具有可塑性與可再使用性的測試模組，開發者可藉由外部的修改，來擴張與應用其基本的功能。在本文中，將使用一個基礎的架構來驗證此模型的可行性。此基礎架構使用簡單的 MVC 架構，建構於網路伺服器上，對於多個資料庫做資訊的驗證。該架構的模型如圖 三。詳述如下：

1. 網頁基礎之操作介面

   以網路程式之介面為基礎，架構於程式伺服器。包含創造代理人實體，輸入監控之資料來源、監控與分析規則的介面，並可以存取代理人狀態與監控資訊。

2. 代理人工廠與管理單位

   代理人工廠可根據知識庫之規則，或是使用者自行輸入的規則，創造代理人的實體。管理單位可以將代理人所監控之資訊反應至介面供管理者及時查看。創造代理人時所輸入的管理規則，可存入知識庫中，供未來的代理人實體或是新系統使用。

3. 代理人實體集

   使用代理人工廠產生的代理人實體可根據指定的規則，紀錄或分析系統使用者或管理者需求的資料，並藉由網頁介面印出。不同的代理人可以根據不同的規則和不同的資料集同時做即時的監控，並以一定時間間隔回傳資訊。

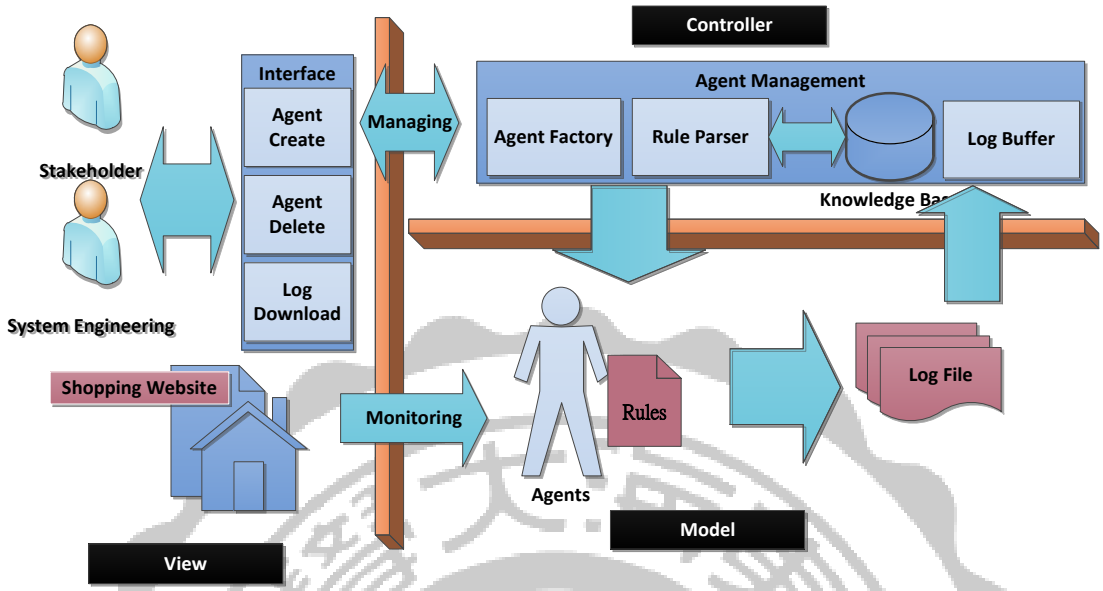

圖 三 系統架構

一個網路程式的管理者可藉由使用代理人連接網路服務之資料庫，並針對其中的資訊作分析，再根據自訂的產出監控資訊作研判。自定義的資料庫可以是牽涉商業智慧的報表、統計資訊，也可配合系統監控的資訊，使用 API 擴張成擁有自動與即時監控功能的中介軟體元件。

3.2. 智慧型代理人

智慧型代理人可以在不受使用者控制的情況下，自發性地完成指定的作業，法則式代理人(Rule-based agent)泛指可以特定的規則執行動作的代理人種類。在本論文中，代理人的執行法則(rules)透過預設

語法的編譯,執行指定的命令,可以在管理者離線(off-line)的狀況下，達成監控系統的工作。

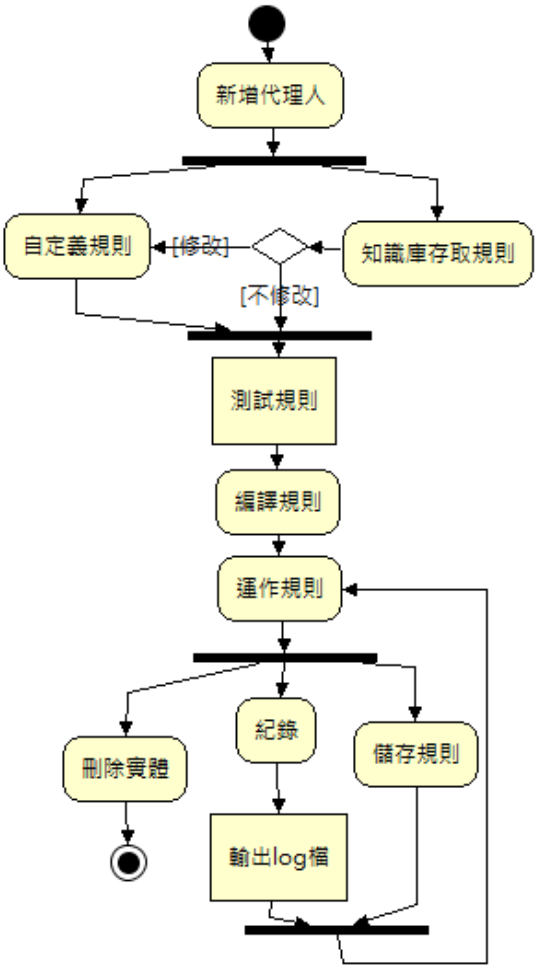

圖 四 代理人運作圖

一個測試代理人的運作如圖 四。在使用者定義了一個作為監控作用的測試代理人後,首先後台的編譯器就會根據使用者的測試步驟與產生之 log 編出一個代理人執行的步驟。接著這個編譯的結果，便

會成為代理人的實體運作之規則，而此代理人會在系統背景中執行不間斷的測試工作，達到監控的效果，直到管理者將實體刪除為止。在此實體執行的過程中，管理者可以下載這個規則中定義之紀錄成果，也可將此規則記錄到知識庫中。

代理人所使用的語法為一個元組(Tuple)，$r_x=\{S, s_0, V, l_x, \Sigma\}$，其中：

- S 代表所有可能的動作陳述(statement)集合。
- s_0 代表初始的動作陳述。
- V 代表其中擷取的參數集合。
- l_x 代表 r_x 規則輸出的 log 報表檔。
- Σ_x 代表 r_x 指令的序列，依序為$(s_0, s_1, \ldots, s_n)$。

其中 S 中的動作陳述包括：

1. 擷取網路資料庫的資訊

   在語法中，變數可透過指令 **ret()**取得資料庫的資訊。變數會以系統時間作為標籤儲存成資料集，時間間隔為代理人運作之時間間隔。

2. 將資料庫資訊作運算或比較

   運算可分為兩種：四則運算與邏輯運算。四則運算回傳數值，而邏輯運算回傳布林值。四則運算優先於邏輯運算。

3. 輸出至報表

透過 **statm()**指令，可將監控資訊輸出至一報表。其中的參數代表報表中的資訊 columns，可以依照需求輸出多行資訊。

每個代理人工作是獨立(independent)且是無狀態的(stateless)，彼此之間可共存並同時執行。這樣的好處是可以針對不同個案，進行不同的監控，並且不影響到系統的運作。另外，也可以根據情況增減監控的案例，讓監控的作業可以更加的有彈性。

表 一 規則語法範例

作業種類	**語法**
擷取資料	Var V = ret(data_block);
資料運算	Var V3 = V1 op V2;
輸出報表	statm(V1, V2, V3…);
運算子 1	+, -, *, /, %...
運算子 2	>, < , =, !=…
系統參數	SYS_TIME, SYS_SPACE…

3.3. 測試知識庫

由於使用過的監控規則或模組，未來很有機會被再次利用，不管是既存的系統或是經過升級的新系統，因為不同企業行號所保留下來的監控模組，都可能包含不同有價值的商業智慧，依存於不同的商業模式與企業管理效能。此時一個可以保存過去既有之規則的知識庫，對於企業的管理就顯得相當重要。在本系統中，使用者將透過管理介面，保留既有的監控規則。這些監控規則將透過建立時間，與管理者定義之名稱進行管理。

在本系統中，知識庫主要的應用在創造代理人的過程。管理者可根據知識庫中儲存的規則，選擇其中適用於目前測試目標之規則，或是根據其規則再加以修改。另外加增知識庫內容，做為未來再利用，系統提供兩種方式：保留的前三個規則，與目前運作中的規則。圖 五為知識庫於本系統應用的示意圖。

圖 五 知識庫應用示意圖

表 二 知識庫格式範例

Construct Time	Name	Rule	Comments
01/02/12	Cart	Var T = ret()…	Check Shopping Cart.
03/28/12	Cart_1	Var T = ret()…	Check Shopping Cart. Revised version of “Cart”.
04/02/12	Security	Var T = syss()…	Security Check.

3.4. 應用元件介面

在本論文中，代理人的開發介面將採用 Java EE 中的 Enterprise Java Beans 技術。使用開放的介面不但能減少系統開發的難度與增加雲端 Server 的相容性，也可以提供未來雲端系統開發者的可再利用性(reusability)，讓該代理人資源不僅僅適用於本系統，也可以提供未來將本監控模型建立於其他 Java EE 基礎的雲端服務的彈性空間，或是透過重構的工作改進並適用於市面上既存的系統。圖 六為代理人之介面與產生之實例示意圖。

系統開發者可依該 EJB 介面增加代理人在系統中之功效，或是在新系統中創造專用的監控代理人。由於 EJB 本身的擴充性，新的開

發者可以根據本系統之代理人框架，加上新的測試邏輯自行定義出本身的使用方法，以適用於新的商業邏輯系統。.

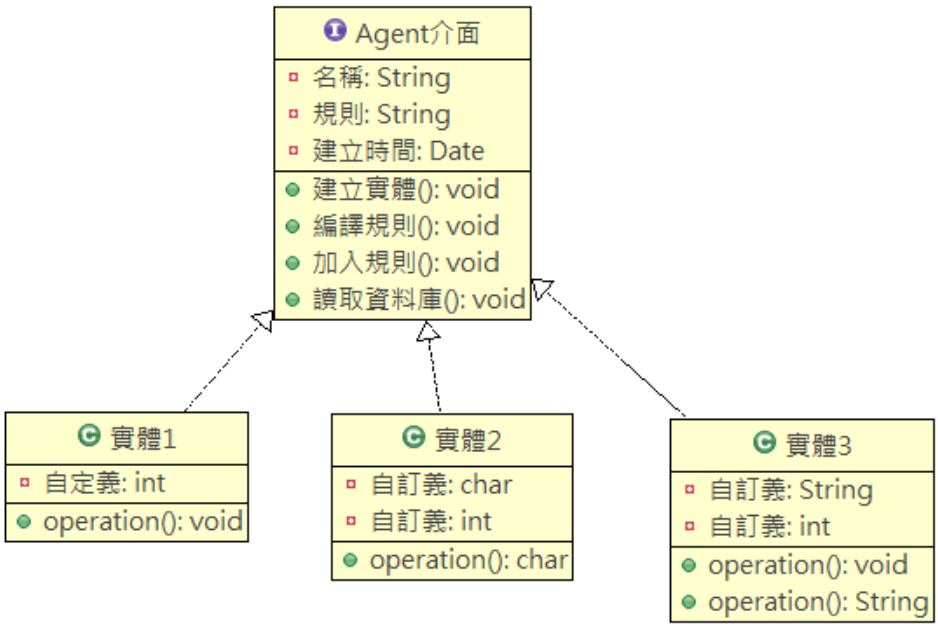

圖 六 代理人介面示意圖

第四章　系統的設計與實作

在本章中，將描述依照前述之系統模型，應用雲端中的 Web 介面與網路服務 Server，實作出之系統概要與實際運作原理。

4.1. 系統規格

本節將簡述系統開發的大致環境與應用的系統環境。

4.1.1. 系統環境

以下是本系統架構之環境：

- Fedora 16
- Virtual Box 4.1.14
- JBoss Application Server 7.1
- MySQL Server 5.5

4.1.2. 開發環境

本系統將使用以下環境開發。

- J2EE 1.7：包括動態的網路技術如 Servlet 等。
- Eclipse Indigo for Java EE
- Plug-in for JBoss Application Server
- Hibernate Framework

4.2. 系統流程

本系統基本流程如圖 七所示。以下解釋其中細則：

1. 首先登入管理頁面
2. 此時可以選擇新增代理人功能、儲存代理人規則功能。
3. 代理人創造可以自行填寫或從知識庫取出，系統產生根據此規則產生實例執行。
4. 透過管理頁面，可以將執行中之代理人實例刪除。
5. 也可透過管理頁面下載監控 log。
6. 實例使用之規則可以留存於知識庫，包括執行中規則或是前三次使用之規則。

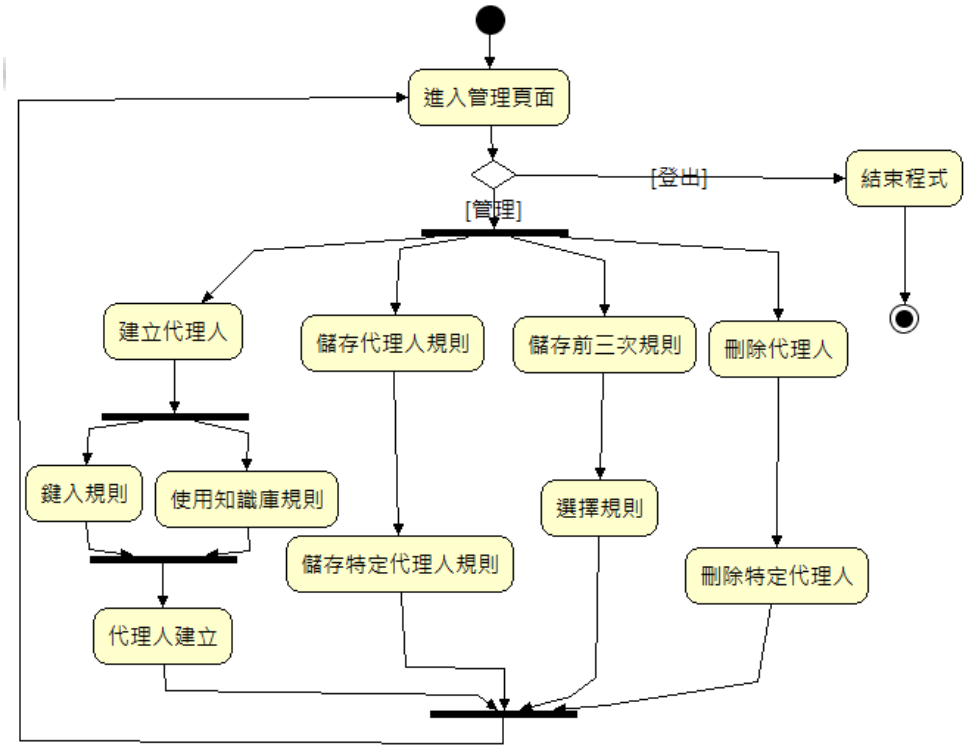

圖 七 流程圖

4.3. 使用者介面

使用者介面包含創建與管理現存代理人的功能。其中的細項包括：

- 運作中代理人列表：列出現在運作中的代理人，與建立時間。
- 新增代理人功能：進入新增頁面，包含規則輸入功能，可選擇知識庫庫存規則或是自行新增規則。
- 下載代理人監控紀錄：可由管理頁面選擇代理人之監控記錄檔。
- 刪除運作中代理人：選擇欲刪除的代理人，系統會儲存期監控期間之記錄檔。
- 將測試規則存至知識庫：有兩種選擇，一是直接由運作代理人選單，另一個是由自動保存的三項規則清單選取。
- 重新讀取運作中代理人清單
- 登出管理頁面

在本系統的實作中，介面的部分使用 Java Servlet 加上 JSP 實作，建立在 JBoss AS 7.0 的容器(container)之中，架設在同樣由 Red Hat 維護的 Linux 版本 Fedora 16 作業系統。

圖 八 管理介面圖

4.4. 代理人工廠

代理人的實作使用了EJB的做法，同樣建立在JBoss的容器之下，使用一個統一的 interface，實作成一個 Session bean。此 Session bean 包含的測試規則將會透過 Message-driven bean 來產生執行的實例。此實例即為做即時測試的代理人。當代理人被選擇建立的時候，會給使用者一個輸入框，以鍵入其監控的規則。使用者也可選擇已記錄至知識庫的規則，由清單選擇後，會自動填入輸入框，以供使用者作更改。

接著使用者可以按照需求選擇需要測試的資料庫表單，將該表單

之登錄資訊與資料組指定給代理人之參數。而後則能依照指定的參數寫入規則，供給執行規則語法的代理人攫取其中資料。在實作中，可擷取的資料設定為兩組表單中的資料組。當規則被建立，並通過伺服器端語法檢查後，會由後方的一個編譯器，編譯出該代理人應該執行的動作。系統將按照規則組合成代理人的實體，進行定義的測試工作。

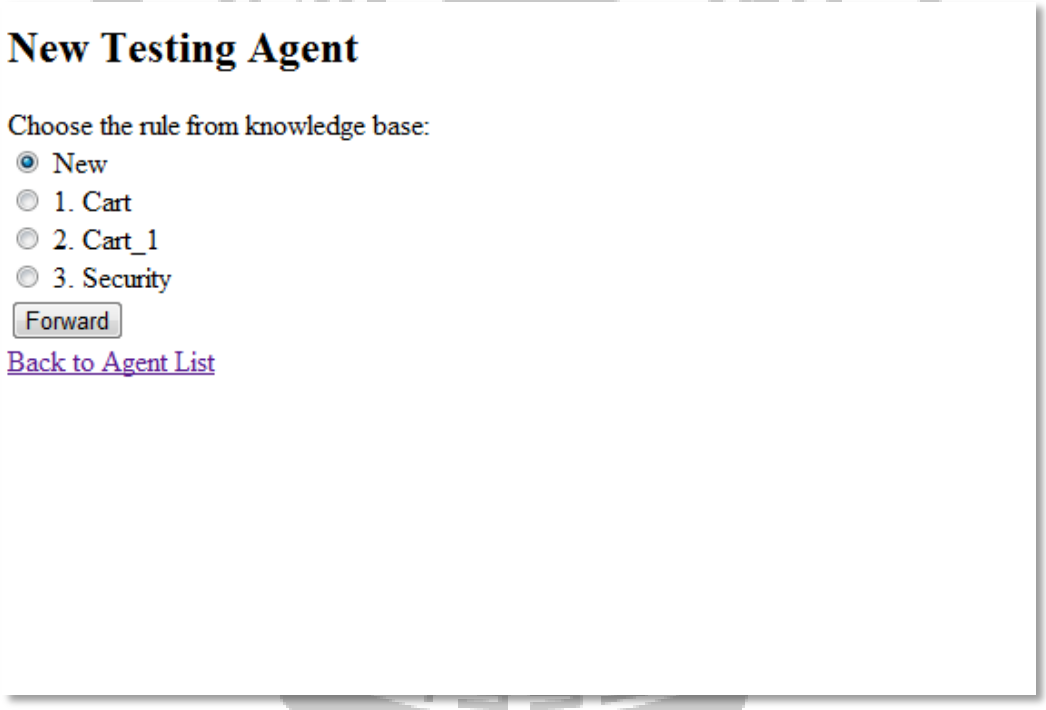

圖 九 選擇新創 rule 或使用知識庫

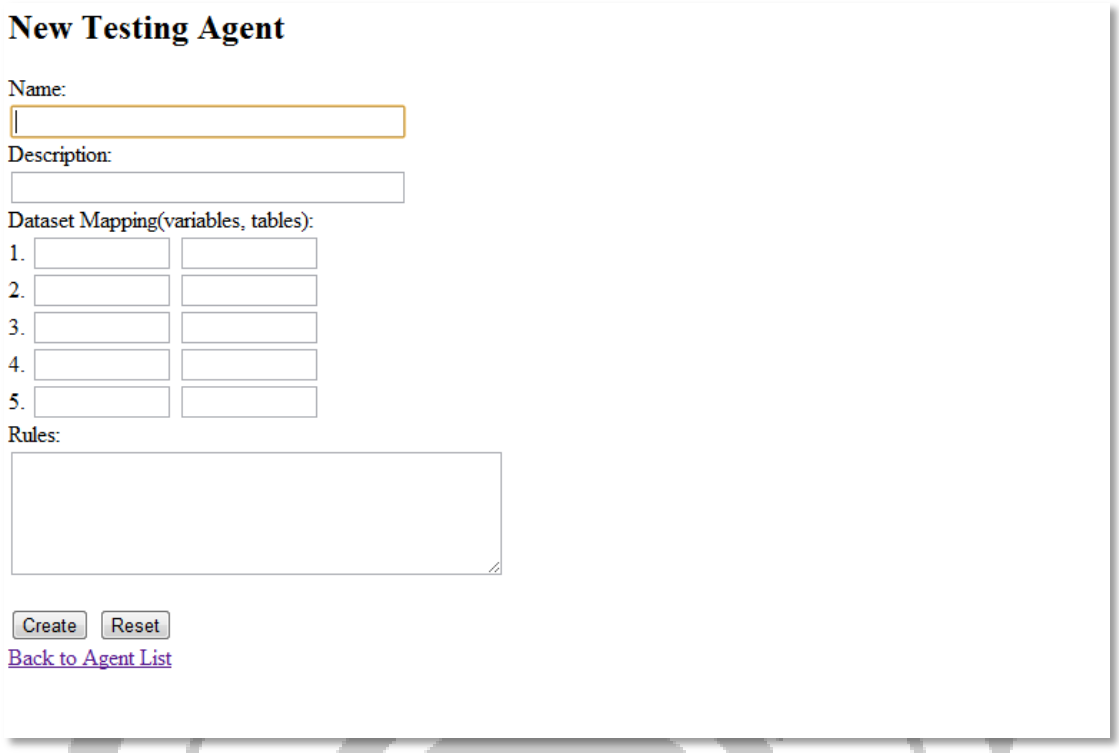

圖 十 創造代理人之介面

4.5. 知識庫實作

代理人執行的規則在通過編譯後，會自動存在一個 buffer 中。此 buffer 容許最近 3 項創造的測試規則。使用者可以透過介面將規則存入知識庫中，並賦予一個名稱與簡介，以做為未來當使用者再創建測試代理人時，可以選用知識庫中已存之規則，而不需再由人親手輸入，讓同質性高的測試作業可以增加效率節省開發時間與資源。

在本系統中，知識庫的實作採取了 JDBC 加上 MySQL 的方式，另外結合了 Hibernate 框架以實現物件導向的資料庫程式設計。物件導向框架的好處在於：結合程式開發相關的物件，將資料庫內涵對應

物件中的屬性。並且能透過既定的物件屬性，實作查詢功能，避免參雜過多的應用介面的操作，造成程式邏輯上的複雜難以維護。

4.6. Enterprise Java Beans 與再利用元件

代理人的實體將包括一連串獨立的 EJB 實體，而此實體是透過統一的介面所產生。因此網站開發者可以透過這個特性，將此測試代理人模組移植到可相容的系統，實作只需做些微的調整即可。

在此基礎下，開發者可依需求增加代理人實際應用的範圍，並可擴展到不同的商業邏輯上。最終可依此架構之雛形，發展出一個購物網站開發之框架，其中此測試代理人的系統，便可由開發者自行發展為依需求在背後進行自動黑箱式測試的模組。代理人介面將提供的主要介面如下表：

表 三 介面架構表

名稱	**作用**	**主要邏輯**
CreateInstance()	創造代理人個體	代理人工廠主體，創造實體。
ReadRules()	匯入運作規則	讀進規則，檢查語法。
ParseRules()	運作規則編譯	解讀出規則，排定執行動作步驟。

ReadDB()	讀取資料庫	從表單讀取欲分析的資料集。

第五章　案例研究與評估

本章中建構一個虛擬的購物網站平台，包括一般購物網站可能包含的功能與介面，如：使用者登入、購物車與商品欄等等。並且以特定的代理人與運行規則，來驗證本論文之架構可行性。

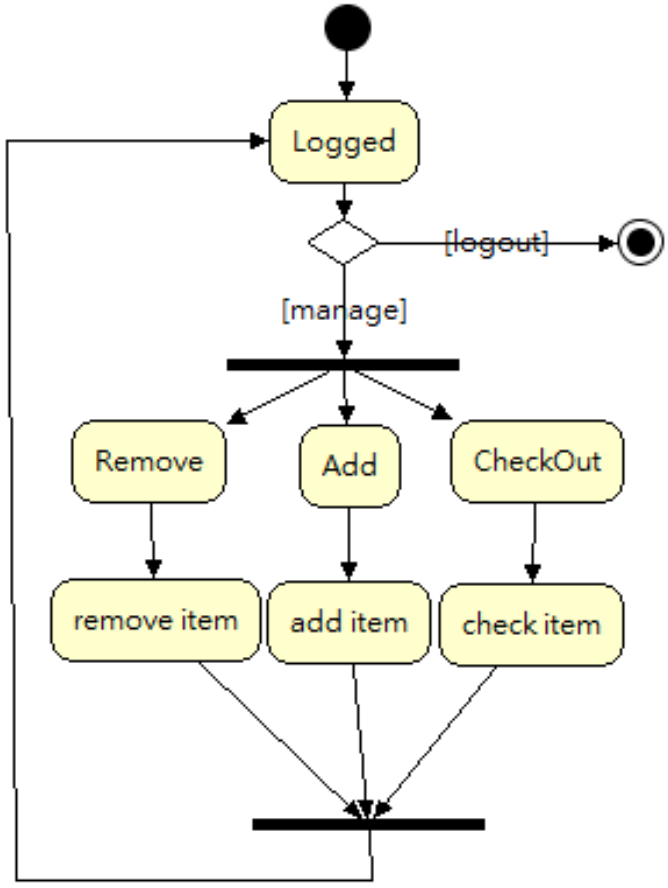

圖 一一 購物網站實例

5.1. 購物網站實例

圖 一一為一個簡單的購物網站實例。本章將依據這個例子，配合本論文之測試系統架構，實際驗證這些操作與原理。

由圖可見，使用者輸入帳號密碼<uid, pswd>登入購物系統，接著可以對購物車(cart)裡的物品(product)的種類與數量(quantity)做操作。其中的動作包括有移除物件(remove)、增加物件(add)與結帳(checkOut)。

5.2. 初始化代理人

在初始化代理人的作業中，必須要提供以下資訊：

- 名稱：作為識別用。
- 規則：實際執行的規則。
- 註解：作為未來選取規則與了解該代理人實作規則用。

接著，便可選欲作為規則種定義的資料組的表單資料。代理人會根據規則對測試的資料作運算，並輸出 log 檔案。

在本例中，將分為三個部分做測試，分別是：購物車之成本檢查，避免標錯價的狀況、購物車中物品與存貨檢查、與檢視登入者的動作，以保護帳號安全。以下分別以三者的狀況創造測試資源：

5.2.1 成本檢查代理人

在本測試中，將比較購物車中，在固定獲利 10%的狀態下，是否跟貨物成本計算符合，以初步判定標價之正確性。price 為價錢，cost 為成本。參考語法如下：

表 四 成本規則

```
Var V1 = ret(price);
Var V2 = ret(cost);
Var V3 = (V1 >= (V2 * 1.1)
statm(V1, V2, V3);
```

5.2.2 存貨檢查代理人

在本測試中，將比較庫存是否足以應付購物車訂單數量。quantity 為各貨品在購物車的數量，stock 為各貨品在庫存的數量。參考語法如下：

表 五 庫存規則

```
Var V1 = ret(quantity);
Var V2 = ret(stock);
Var V3 = (V1 <= V2);
statm(V1, V2, V3);
```

5.2.3 登入動作檢查代理人

在本測試中，將觀察使用者登出入是否過於頻繁，以找出不適當的操作情形。name.logintime 為名為 name 之使用者名稱的登入時間列表，name.logouttime 則為登出時間列表。比較登入登出的時間是否小於 0.5 分鐘(30 秒)。

表 六 登入動作規則

```
Var V1 = ret(name.logintime);
Var V2 = ret(name.logouttime);
Var V3 = (V1 - V2 < 0.5);
statm(V1, V2, V3);
```

在測試一段時間後，可由管理頁面下載依照測試規則跑出的 log 紀錄檔案，由其中內容可得知使用者欲分析的結果。以下是參考的運算結果：

表 七 成本運算結果

quantity	cost	result
190	177	FALSE
490	444	TRUE
450	411	FALSE
930	849	FALSE

970	878	TRUE
210	187	TRUE
560	508	TRUE
930	848	FALSE
690	625	TRUE

表 八 庫存運算結果

quantity	stock	result
14	11	FALSE
13	17	TRUE
16	18	TRUE
20	15	FALSE
7	6	FALSE
6	11	TRUE
13	10	FALSE
15	18	TRUE
9	10	TRUE

表 九 登入監控結果

name.logintime	name.logouttime	result
下午 11:31:00	下午 11:33:15	FALSE
下午 11:39:15	下午 11:39:21	TRUE

下午 11:41:52	下午 11:46:25	FALSE
下午 11:48:17	下午 11:48:32	TRUE
下午 11:54:10	下午 11:56:47	FALSE
下午 11:54:44	下午 11:56:43	FALSE
上午 12:00:50	上午 12:05:11	FALSE
上午 12:06:11	上午 12:11:03	FALSE
上午 12:07:36	上午 12:12:32	FALSE

5.3. 測試知識庫學習

在建立一個代理人實體之後，有兩種將既有規則存入知識庫的做法。其一為由管理頁面選擇正在執行的測試代理人實體名單，直接選擇需要保留之規則。另外一個存入的方式，為系統自動保留之近期使用的規則列表。該列表最大的保留數為 3 個。

使用者可選擇執行過的測試規則，包括已刪除之實體運行之規則，永久保留至知識庫中。保存的規則可在日後的創造代理人實體的頁面選擇使用，或是做更改成為不同的規則。

圖 十二 選擇存入之規則

5.4. 評估

由實作成果，比較目前的工具測試方法(Tool-assist testing)，至少有以下幾點優勢：

- 開發者可在系統內部將此系統與本身的商業邏輯做整合，不需要配合工具操作。
- 管理者可以輸出自定義的監控 log 檔案。
- 可以移植過去成功的工具內含的測試演算法。

另外，較之於過去方法，也有幾項技術性上的缺點：

- 須對 Java EE 的開發有一定認知。

- 測試規則涵蓋性稍嫌不足。
- 對執行環境(Runtime environment)有依賴性。

然而，本文仍演示了以代理人的方式，在系統背後執行自定義規則式之測試的可能性。在這個架構之下，如能引進過去各種成功的測試作業執行經驗，並且改善執行的環境與技術，相信能夠創造出一個彈性化的測試機制，適用於各種以 Java EE 為基礎的網路服務。

第六章　結論與未來工作

代理人技術是人工智慧中，相對接近於實際應用的部分。雖然對於代理人的應用已經五花八門，但是卻很少將其運用於雲端程式的維護上。本論文討論了一個使用代理人的自動化測試，建構於網路上可以不間斷的黑箱測試系統，達成網路程式監控的效果。在理想的狀況下，管理者可以透過自訂義的規則與輸出紀錄格式，配合該組織自有的商業邏輯做管理。曾經使用過的測試規則，更可以存在一個知識庫，給予未來作為相似作業的資源。而這個系統的核心，將以 Java EE 中的 Enterprise Java Beans 元件實作，以其分散且獨立的特點，達到可再用(Reusable)與可移植(Portable)，不但可以適用於其他系統，只要經過些微的更改，就能做未來的應用。該系統架構於一個 JBoss 的 Application Server 中，並應用其中的 framework 工具與其對應 Java EE 中各種層面的包容性，以減少開發的障礙。最後，本論文將實作應用於一個虛擬的網路購物程式，示範了其操作的步驟與驗證了系統的可行性。證明本系統可以應用於特定的網路程式架構，也證實了其自動監控的程序之實用性。

在未來的工作，能在本論文之架構下，擴充代理人之規則，將其標準化，讓代理人能操作實際網站之架構，或是擷取網站之程式碼，分析其漏洞，自動構成可程序化的黑箱測試方法，並可藉由參考過去

成功的方法，幫助提升在測試方面的涵蓋度。另外，預計將本論文之代理人架構，應用在雲端硬體監控項目中，以作為彈性運用的判斷依據。最終配合此監控架構，構成一個雲端系統中介軟體開發的框架。

參考文獻

[1] Bigne, E., Ruiz, C., & Sanz, S. (2005). The Impact of Internet User Shopping Patterns and Demographics on Consumer Mobile Buying Behavior. *Journal of Electronic Commerce Research*. 6 (3).

[2] Jarvenpaa, S. L., & Todd, P. A. (1997). Consumer reactions to electronic shopping on the World Wide Web. *International Journal of Electronic Commerce*. 1, pp. 59–88.

[3] Peterson, R. A., Balasubramanian, S., & Bronnenberg, B. J. (1997). Exploring the implications of the Internet for consumer marketing. *Journal of the Academy of Marketing Science*. 25, pp. 329–346.

[4] Huang, M. (2000). Information load: its relationship to online exploratory and shopping behavior. *International Journal of Information Management*. 20, pp. 337–347.

[5] iThome. (2008). 線上購物網站的安全到底出了什麼問題. Retrieved from http://www.ithome.com.tw/itadm/article.php?c=47677

[6] 李允中. (2009).。*軟體工程 Software Engineering.* 美商麥格羅.希爾, Taiwan

[7] Perry, W. E. (1999). *Effective Method for Software Testing*. 2nd Ed. Danvers, MA: Wiley.

[8] Beizer, B. (1995). *Black-Box Testing: Techniques for Functional Testing of Software and Systems*. Danvers, MA: Wiley.

[9] Edwards, S. H. (2001). A framework for practical, automated black-box testing of component-based software. *Software Testing, Verification and Reliability*, 11(2), pp. 97-111.

[10] Mutz, D., Vigna, G., & Kemmerer, R. (2003). An Experience Developing an IDS Stimulator for the Black-Box Testing of Network

Intrusion Detection Systems. *Proceedings of the 19th Annual Computer Security Applications Conference*. Las Vegas, NV, USA

[11] Krichen, M., & Tripakis, S. (2004) Black-Box Conformance Testing for Real-Time Systems. *Lecture Notes in Computer Science*, 2004(2989), pp. 109-126.

[12] Booch, G. (1994). *Object-oriented analysis and design with applications* 2nd ed. Redwood City, Calif: Benjamin/Cummings Pub. Co.

[13] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns. Elements of Reusable Object-Oriented Software*. Boston, MA: Addison-Wesley.

[14] Johnson, R.E., & Foote, B. (1988). Designing Reusable Class. *Journal of Object-Oriented Programming*, 1(2): pp. 22-35.

[15] Meyer, B. (1990). Tools for the New Culture: Lessons from the Design the Eiffel Libraries. *Communications of the ACM*, 33(9): pp. 68-88.

[16] Holland, I. M. (1993). The Design and Representation of Object-Oriented Components. (Doctoral thesis). Northeastern University, Boston, MA. Retrieved from http://www.ccs.neu.edu/home/lieber/theses-index.html

[17] Chu, C. W., Lu, C. W., Yang, H., & He, X. (2000). A Formal Approach for Component Retrieval and Integration Analysis. *Journal of Software Maintenance*, 12(5): pp. 325–342.

[18] Chu, C. W., & Yang, S. (1999). A Formal Approach to Software Design Process with Reuse. Proceedings of *the Fourth World Conference on Integrated Design & Process Technology*, Kusadasi, Turkey.

[19] Chu, W.C., Hsu, C.P., Lu, C. W., & He, X. (1999). A Semi-Formal

Approach to Assist Software Design with Reuse. Proceedings of *ICSM'99: IEEE International Conference on Software Maintenance*, Oxford, England, pp. 256-264.

[20] Yang, J. T., Huang, J. L., Wang, F. J., & Chu, C. W. (2002). Constructing an Object-Oriented Environment for Web Application Testing. *Journal of Information Science and Engineering*, 18(1): pp. 59-84.

[21] Chu, C.W., Lu, C.W., Chang, C.H., & Chung, Y.C. (2001). Pattern Based Software Re-engineering. *Handbook of Software Engineering and Knowledge Engineering*, (1), pp.767-786. River Edge, NJ: World Scientific Publishing.

[22] Bradshaw, J. E. (1997), *Software Agents*, MIT Press.

[23] Hayes, C. C. (1999). Agents in a Nutshell-A Very Brief Introduction. *IEEE Transactions on Knowledge and Data Engineering*, 11(1), pp. 127-132, Jan.-Feb. 1999, doi:10.1109/69.755621

[24] Zulkernine, F. H., Powley, W., & Martin, P. (2009). Aut-onomic Management of Networked Web Services-Based Processes. *Autonomic Computing and Networking*, Eds. by Denko, M. K., Yang, L. T. & Zhang, Y., New York, NY: Springer.

[25] Zulkernine, F. H., & Martin, P. (2011). An Adaptive and Intelligent SLA Negotiation System for Web Services. *IEEE Transactions on Services Computing*, 4(1), pp. 31-43.

[26] Zulkernine, F., Patrick Martin, P., Craddock, K., & Wilson, K. (2009) Policy-based Middleware for Web Services SLA Negotiation. Proceedings of *IEEE International Conference on Web Services.* Los Angeles, CA.

[27] Green, D. (2002) Enterprise Beans. *J2EE Tutorial*. Retrieved from http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/EJBConcepts.html

[28] Rubinger, A. L., & Burke, B. (2010). *Enterprise JavaBeans 3.1*, 6th Ed. Sebastopol, CA: O'Reilly Media.

[29] Debu Panda, Reza Rahman, Derek Lane. (2007). *EJB 3 in Action*. Greenwich, CT: Manning.

[30] Java Persistence 2.0 Expert Group. (2009). *JSR 317: Java™ Persistence API, Version 2.0*. Retrieved from http://download.oracle.com/otndocs/jcp/persistence-2.0-fr-oth-JSpec/

[31] King, G., Bauer, C., Andersen, M. R., Bernard, E., Ebersole, S. & Ferentschik, H. (2004). *Hibernate Reference Documentation*. Raleigh, NC:Red Hat. Retrieved from http://docs.jboss.org/hibernate/stable/core/manual/en-US/html/

[32] Bernard, E., Ebersole, S., & King, G. (2005). *Hibernate Entity Manager*. Raleigh, NC:Red Hat. Retrieved from http://docs.jboss.org/hibernate/entitymanager/3.5/reference/en/html_single/

[33] JBoss Enterprise Middleware. Retrieved from http://www.redhat.com/jboss/

[34] In, H. P., Kim, C. H., Yun, U., & Yau, S. S. (2003). Q-MAR: A QoS Resource Conflict Identification Model for Situation-Aware Middleware. Proceedings of *FTDCS'03*: *The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*. San Juan, Puerto Rico.

[35] Wohlstadter, E., Tai, S., Mikalsen, T., Diament, J., & Rouvellou, I. (2006). A Service-oriented Middleware for Runtime Web Services Interoperability. Proceedings of *ICWS '06: International Conference on Web Services*. Rosemont, IL.

[36] Tambe, S., Dabholkar, A., & Gokhale, A. (2009). Fault-Tolerance for Component-Based Systems - An Automated Middleware

Specialization Approach. Proceedings of *ISORC '09: IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, 2009.

[37] Coulson, G., Grace, P., Blair, G., Mathy, L., Duce, D., Cooper, C. . . . Cai, W. (2004) towards a component-based middleware framework for configurable and reconfigurable grid computing. Proceedings of *WETICE 2004: The 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. Modena, Italy.