可快速與 Eclipse 環境整合的軟體之規範與輔助工具發展

Pluggable Software for the Eclipse Platform

吳昆澤
國立政治大學資訊科學系
g9109@cs.nccu.edu.tw

陳正佳
國立政治大學資訊科學系
chencc@cs.nccu.edu.tw

摘要

Eclipse 是目前最被普遍使用的開發環境與工具整合平台。已經有很多的軟體工具被建置或整合至其中；並且有越來越多的工具開發者嘗試將他們的工具以所謂外掛程式的方式整合至此平台。但是，由於 Eclipse 是一套新式且複雜的平台，大多數工具開發者並不知如何將其工具封裝為 Eclipse 外掛程式，而且學習 Eclipse 外掛程式開發也需冗長過程。基於此種因素，我們構想出一套可外掛軟體(Pluggable Software)的規範，對一般命令列工具的輸入、執行和輸出等部份提出限制條件。只要工具開發者建構的工具程式符合可外掛軟體的規範，就能利用我們提供的輔助工具，幫助工具開發者迅速將其工具整合到 Eclipse 平台。

關鍵詞：Eclipse、外掛程式、軟體整合開發環境、命令列介面、啟動

Abstract

Eclipse is a development environment and tool integration platform which is currently widely used. There has been many software tools built in the form of plug-ins and integrated into this platform, and more and more tool developers are trying to integrate their tools into this platform. But, since Eclipse is a relatively new and complicated system, most tool developers do not know how to develop Eclipse plug-ins, and it is a steep learning curve to get familiar with Eclipse plug-in development. Therefore, we propose the definition of pluggable softwares, which are general command line tools but must satisfy special restrictions imposed on their input, execution and output for ease of integration. As long as the tool developer can build a tool conforming to our definition, the aiding tool we provide can help the developer rapidly integrate his tool into the platform.

Keywords：Eclipse、Plug-in、IDE、Command Line Interface、Launch

一、導論

Eclipse[4] 是由以 IBM 為主導的 eclipse.org 組織所開發與推廣的開放原始碼(Open Source)平台，它是目前最被普遍採用的工具整合平台，本身既可當成開發環境(IDE)也可作為客戶端豐富平台(Rich Client Platform; RCP)。

Eclipse 相較於其他軟體開發環境的最大特色是它不僅是一個開發環境，而且是一個工具的整合平台。這些工具程式通常是以封裝成所謂的外掛程式(plug-ins)的方式而整合至 Eclipse 平台之中。一般的開發環境事實上也會提供外部工具的整合功能，然而這些工具通常只能以該環境預定的方式依附在該環境架構中，而工具本身則無法再被擴充。相對的，Eclipse 提供的是完整的可擴充性：除了可以用外掛程式擴充 Eclipse 平台以外，外掛程式本身也可以其他外掛程式擴充之。而實際上 Eclipse 平台除了極小的核心部分外，其他全部都是以外掛方式組成。

選用 Eclipse 為系統發展環境對軟體開發者而言具有相當多的好處。一方面由於 Eclipse 是一個免費的工具整合平台，開發者可從網路上找到其他開發者為 Eclipse 所建造的各式各樣商業化或非商業化的外掛軟體工具。利用 Eclipse 提供的簡單安裝機制，開發者能夠從中自由挑選需要的工具，只要下載安裝就可以建置一個適合自己的整合開發環境。另一方面，對於發展環境的使用者來說，由於工具程式都是外掛於 Eclipse 平台，具有相同模式的使用者介面與操作方式，因此對於不同的工具程式，使用者並不需重新摸索即能知道其中大部分的使用介面與操作方式。此外，另一個最吸引人的優點是 Eclipse 平台提供了工具程式間完善的整合環境，使得工具程式彼此得以在 Eclipse 平台內互相操控，而完成使用者所需的所有工作。這免除了傳統 IDE 使用者因所用工具間無法互相操控，而

必須在不同工具與介面間切換的麻煩。

Eclipse 的出現與強大吸引力，對工具程式發展者產生了一種新的重大需求：客戶要求工具程式廠商提供可在 Eclipse 平台操作的工具程式版本。為了迎合市場需求，工具程式開發者勢必要為其原有工具或新工具程式提供以 Eclipse 為寄居平台的版本。

但是要開發以 Eclipse 為寄居平台的外掛工具程式時，除了要瞭解該工具程式的功能外，同時也要清楚 Eclipse 外掛程式的機制，並遵循此機制才能開發外掛程式。雖然 Eclipse 提供不少文件，說明如何開發外掛程式，但是對於一般的工具程式開發者來說，要開發外掛程式並不容易。而且依分工原則，工具的功能應由工具程式開發者負責，但是如何建置成外掛程式，則應由熟悉 Eclipse 發展平台與外掛機制的設計師負責。然而目前這方面的人才目前仍屬稀少，這是因為 Eclipse 是一套新式而且複雜的平台，以致大多數工具開發者都還不知如何將其工具整合至該平台，而學習 Eclipse 外掛程式開發亦需冗長過程。基於此一因素，為了使工具程式能快速整合到 Eclipse 平台，我們因而構想出一套軟體規範，希望所有符合此規範的軟體工具，均能利用我們所發展的輔助工具，簡單而快速地將其整合到 Eclipse 平台。符合此種規範的軟體工具，我們稱之為可外掛軟體(Pluggable Software)。

此篇論文的主要目的是解決部分工具程式整合到 Eclipse 平台的問題。我們提出一套可外掛軟體的規範，要求工具程式的輸入、執行和輸出必須符合適當的限制條件。而所有符合此條件的工具程式均能利用我們提供的輔助工具將之整合至 Eclipse 平台。

二、相關背景介紹

2.1 Eclipse 工具整合平台

Eclipse 是一個以 Java 實做的整合開發環境，也是一個各種工具程式的整合平台，同時還是一個開放原始碼的社群，該組織的目標在於建立一個讓各種工具程式都可以外掛的單一整合平台。例如，藉由 Eclipse 的相容產品，程式設計師可以在同樣的一個視窗內，使用 Java 程式編輯器和各種其他的工具程式。

Eclipse 不僅只是一個整合開發環境而已，它同時還是一個各種工具程式的整合平台。Eclipse 的主要目的是建立一個多用途的整合開發環境和工具，讓應用程式開發者透過 Eclipse 的軟體開發套件，把各種不同的工具程式整合到 Eclipse 平台，達到一個共用使用者介面平台的目標。

圖 2-1 是 Eclipse 系統開發套件(SDK)的架構圖。其中平台執行核心(Platform Runtime)負責管理延伸點(Extension Point)和外掛程式的模型，可動態地定義和維持外掛程式的註冊資訊。工作區(Workspace) 是負責資源管理 (Resource management)，透過檔案系統管理各種資源(專案、檔案和資料夾等)。工作台(Workbench)是使用者介面的實作，包含 SWT(Standard Widget Toolkit)和 JFace。

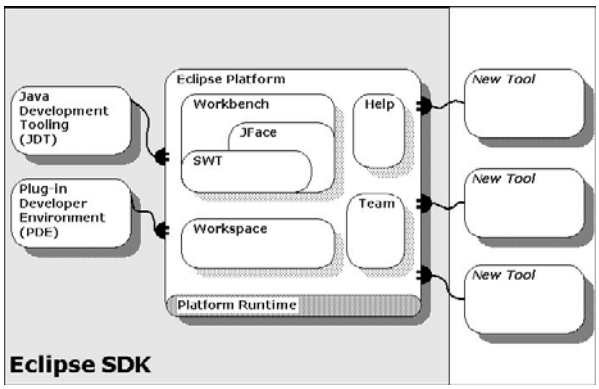

圖 2-1 Eclipse SDK

Eclipse 平台的結構圍繞著所謂的外掛程式的概念。外掛程式是提供功能給系統的結構化程式與資料組件。功能的提供形式可以是程式碼庫(具有公用 API 的 Java 類別)、平台延伸(Extension)、甚至是文件。外掛程式可以定義延伸點，讓其他的外掛程式能夠新增功能，這是 Eclipse 外掛程式與一般外掛程式最主要的差異。

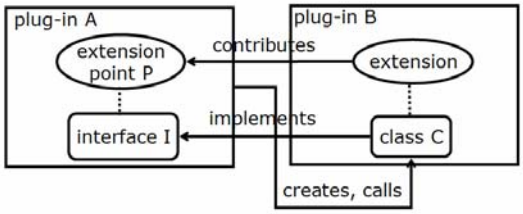

圖 2-2 延伸與延伸點

圖 2-2 說明延伸與延伸點之間的關係。外掛程式 A 提供一個延伸點 P，由於外掛程式 B 要擴充外掛程式 A 的功能，因此使用延伸點 P 來擴充功能，並且在外掛程式 B 的類別 C 中實作延伸點 P 所定義的介面 I，以提供延伸點 P 所要求的功能。由於延伸與延伸點的關係，因此提供外掛程式相當好的擴充性，這也是 Eclipse 外掛程式與其他平台外掛程式間最大的差異，可以透過延伸點在既有的外掛程式上擴充新的功能，更可以定義延伸點以供其他的外掛程式擴充。

工具程式與 Eclipse 平台整合的方式有二：平台內整合與平台外整合。所謂平台內整合是指遵循 Eclipse 的標準機制把工具程式封裝部署為外掛程式，其中重點任務包含：為工具程式建立使用者圖形介面以便利使用者叫用工具，改寫或包裝工具程式的入口類別，使其變成平台上某些延伸點的延伸

實做，以便平台可依需要自動叫用。此外當無適當延伸點可用時，我們可能尚須為工具程式自定延伸點。這種整合的特點是工具程式與 Eclipse 平台以及其他外掛程式均共用同一歷程(process) ，在 Amsden[11] 的工具整合程度四級分類裡是屬於最高二級：API 與 UI 整合。

至於平台外整合，則是讓工具程式依作業系統下的原來方式，以獨立的歷程執行。Eclipse 平台部份則負責利用自己的使用者介面收集工具程式執行時所需之參數與選項，再據以產生命令列指令，最後再透過Java的標準方法產生獨立於Eclipse平台的歷程以啟動並執行該工具程式。這樣的整合在[11] 的分類裡是屬於第一級的叫用整合(invocation integration)。叫用整合最顯著的缺陷是 Eclipse 平台會忽略工具程式執行時對系統產生的改變，以致無法與作業系統保持同步。倘若我們能提供功能，使 Eclipse 平台得以感知工具程式執行後在檔案資源上所產生的變化，使其具有進一步觸發連動其他工具的可能，則此種整合方式稱為資源層級整合，在[11] 中是屬於第二級。

平台內整合通常比平台外整合更為完善，其優點包括：具有較快的執行效率、提供使用者一致的 Eclipse 風格使用介面、以及工具間易於利用平台機制互相操控連動。所以理想上應該以平台內整合為優先考量。然而實務上面臨的問題是平台內整合意謂著：工具程式必須是 Java 程式(或可用 JNI 包裝的非 Java 程式)、整合時工具程式的部份程式碼必須變動、以及整合碼通常相依於個別工具程式，不具規律，不易自動合成。由於這些要求實在限制過大，因此我們的輔助工具提供的只是平台外的工具整合。

2.2 命令列介面(Command Line Interface)

在使用者圖形介面(Graphical User Interface, GUI)出現前，使用者與電腦之間的溝通是透過命令列(Command Line)[7] 執行。除了主命令(command)外，我們還可以使用輸入選項與參數等附加敘述，說明所需的相關訊息，例如輸入或輸出的檔案名稱、是否進行除錯等等。

使用命令列的好處在於有統一的格式，所有的互動皆是以文字表示，不會因為不同的作業系統而有不同的圖形介面。另一個好處是，複雜的任務可能需要依結果連續啟動不同的工具，這些複雜任務可以寫成指令稿(shell script)，以批次方式執行，而不必像 GUI 需要等待使用者與其互動。一些我們常用的工具程式，例如作業系統的大部分內建指令，都是以命令列的介面執行。我們發展可外掛軟體的第一個想法，即是希望能幫助工具發展者將其命令列介面轉換為符合 Eclipse 風格的圖形式使用介面。然而前提是這些工具必須以命令列方式執行，且命令列的輸出入參數與選項描述必須符合適當的規範。

命令列介面是由使用者輸入文字指令與電腦溝通，但是電腦要如何處理這些輸入的選項與參數，對於工具程式開發者來說，是一件繁瑣的工作。由於命令列處理是所有命令列程式都必須處理的前端模組，本身既繁瑣又具有高度可重用性，因此市場上也針對不同語言發展了許多所謂的命令列處理器(Command Line Processor)。其中 Jakarta Commons 的一項子計畫 CLI[5] ，就提供一個簡單而容易使用的 Java 應用程式介面，幫助開發者處理命令列輸入的選項與參數。開發工具程式的時候，利用 CLI 可快速完成複雜的輸入選項與參數處理模組。我們對可外掛軟體的輸入要求即是它們必需滿足 CLI 對命令列各種類型選項的規範。據此我們利用 XML 定義了一套可描述工具程式中選項與參數格式的語言。當需要將工具程式整合至 Eclipse 時，只要輸入符合此語言格式的工具輸入介面描述，我們提供的輔助工具即可據此產生所需的 Eclipse 圖形介面程式碼，可在執行時收集工具程式所需之輸入資訊。

2.3 日誌記錄(Log)與統一的訊息格式

工具程式執行過程中，會產生各種不同的例外或錯誤訊息。這些說明訊息通常是以不同的格式(依工具程式而異)出現在主控台或特定輸出檔案上。使用這些訊息的方式，通常是由使用者解讀其內容，再經由使用者的知識與認知，而判斷是輸入或執行程式的哪一環節出現問題。對於非專精的使用者而言，例外或錯誤訊息通常代表的是真正麻煩的開始，因為他們通常無法由此找出程式的問題所在。

而整合開發環境架構的最大附加價值之一即是，系統可幫使用者由錯誤訊息，自動追溯並顯示問題的源頭。但是要俱備此種功能的前提是，必須具有統一的訊息格式以及相關的追蹤資訊。所以當我們將工具轉為外掛程式時，必須要求這些工具程式具有明確的訊息表達格式並且內含問題根源追蹤訊息。因此我們在可外掛軟體的輸出部分引入標準日誌(Log)的做法，要求工具程式的輸出訊息均以統一的日誌格式的方式表達。

2.4 輸出處理與訊息連結

當工具程式執行時，產生的說明訊息可以直接顯示在主控台端，而檔案等資源的輸出與變動則可要求 Eclipse 平台管理。因此當工具程式變動檔案資源時，Eclipse 應被告知並進行平台狀態更新，以保持與作業系統之同步，同時也因此使平台使用者能即時感知工具程式造成的資源變化。

除了可以用文句之外，理論上亦可用樹狀或圖表的方式在 Eclipse 平台顯示說明訊息(Robinson[10])。文句式的說明或許不夠親善，但

是卻可以採用訊息與來源連結軟體如 ErrorLink[9] 等外掛程式來進行分析、建立與相關檔案及位置的連結、進而產生其他更親善的輸出訊息。

2.5 後續資源瀏覽與編輯

訊息連結的結果使得Eclipse平台能夠由連結資訊找到相關的輸出或輸入檔案資源，以便進行瀏覽或編輯。而平台的內建機制則會自動或被動地找到一個適用的編輯器以開啟該檔案資源，唯其前提是使用者的 Eclipse 平台必須依檔案型態或名稱之不同，事先登錄與安裝對應的編輯或瀏覽程式。

對大多數檔案資源，尤其是文字型態檔案而言，Eclipse 平台利用其預設機制，必能為使用者找到一個適用的編輯器。只是此編輯器可能功能過於簡單而未能滿足使用者之需求。因此，對於一些無法由市場上找到高品質編輯器，卻又不滿意於預設編輯器之過分簡單的需求者而言，市場上也出現一些為 Eclipse 而造的萬用編輯器如 Colorer[1] 等。這類編輯器可以同時支援多種預設或自行設定的不同型態的文字型檔案，它可以提供到一般文字或程式碼編輯所需的中上等級功能。利用 Colorer 這類編輯器，我們只要依其說明產生描述標的類型檔的語言定義，即能在 Colorer 編輯器上產生對應的檔案類型支援，因此可以免除掉我們必須自行發展編輯器以支援各種後續檔案編輯的巨大難題。

我們發展可外掛軟體的輔助工具所預定的想法是：除非無法找到可自由使用的開放原始碼程式，否則盡量使用現有工具或程式庫，而不要重複發展已經存在的部份。基於此項原則，我們的系統需要使用兩個外掛程式：ErrorLink 和 Colorer。由於其使用與系統實作較為密切，因而留待第四章系統實作時，再予以詳述。

三、可外掛軟體

為了幫助開發者快速將工具程式整合至 Eclipse 平台，我們提出可外掛軟體的構想；符合此規範的工具程式，可利用我們的輔助工具整合至 Eclipse 平台使用。而所謂的可外掛軟體，必須符合以下的規範：

1. 可外掛軟體是以命令列方式執行的工具程式。
2. 可外掛軟體所用的參數與選項格式，可用我們定義的 XML 語言予以描述。
3. 可外掛軟體所用的輸入與輸出檔必需清楚明確，且程式所產生的錯誤或例外訊息，須包含問題根源的追蹤訊息且可被後續程式處理。

我們定義可外掛軟體的原則是希望開發者能在不更動工具程式原始碼的前提下，得以將工具整合到 Eclipse 平台使用。

3.1 可外掛軟體的運作模式

符合可外掛軟體規範的工具程式的執行方式，通常是由使用者於作業系統的命令列中輸入指令以執行該工具程式。指令內容除了工具的路徑、名稱之外，也必須包含執行該程式所需的選項與參數，例如輸入檔、輸出檔、錯誤訊息檔及其他工具執行時所需之設定資訊。當啟動後，工具程式首先會解析出命令列上的選項與參數，再據以決定輸出入檔以及其他影響工具行為的設定。所有資訊備齊之後，即可由輸入檔讀入資料而進行其處理邏輯，而在執行過程中，可能會產出資料至輸出檔，期間若有錯誤，例外或特殊事件時亦可能產生說明訊息至日誌檔或錯誤訊息檔。值得一提的是，異常輸出資訊之中通常含有許多與輸入檔或輸出檔有關的說明訊息。若這些工具程式是內建在 IDE 上，系統通常會連結相關檔案位置，並自動或快速的讓使用者得以用適當的編輯器或瀏覽器開啟相關檔案至對應位置以便編輯審閱。而我們將工具程式整合至 Eclipse 的目的之一就是希望使之具備此一功能。

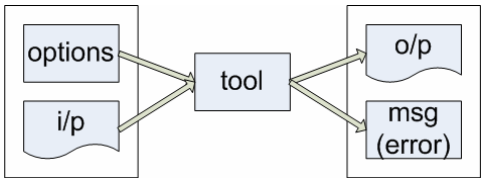

圖 3-1 可外掛軟體的運作模式

我們以圖 3-1 總結一般工具程式的操作模式。圖中 tool 代表工具程式，i/p 與 o/p 分別是工具程式的輸入檔與輸出檔，options 是其他相關的選項，至於 msg 則是異常輸出資訊，用以表達錯誤、例外或特殊事件。其中值得注意的是，雖然 i/p、o/p 與 msg 代表的是輸出入檔案，其名稱是在命令列中的選項或參數中給定，是屬於輸入選項或參數。

3.2 整合後的系統架構

工具程式的命令列資訊格式、啟動方式、輸出入檔位置、和異常輸出資訊的位置與格式是操控此工具程式的基本要素。只要能提供這些要素所需之相關訊息，我們便能在任何非該工具程式預設的環境下，以類似遠端代理人(remote proxy)[2] 的模式執行該工具程式。因此，對於任何符合可外掛軟體規範的工具程式，只要我們在執行時期(Runtime)能透過 Eclipse 平台介面取得這些要素所需之資料，便能在 Eclipse 平台內啟動並執行該工具程式，而達到整合之效果。基於此種構想，我們很快的就推導出整合後的工具程式與 Eclipse 平台之間所構成個整體系統架構(見圖 3-2)。

整合後的系統是由兩大組件組成：一個是在作業系統下執行的原工具程式(tool)，另一個則是

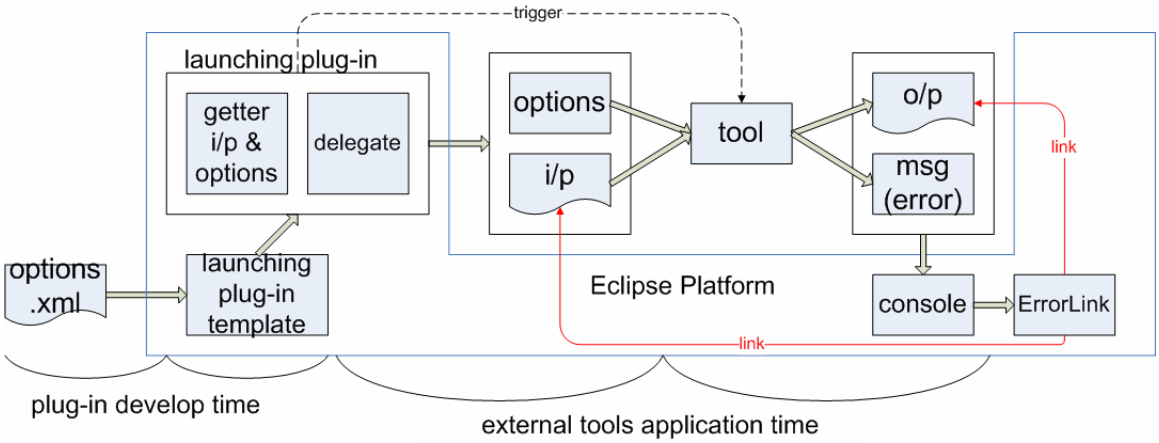

圖 3-2 整合後的系統架構

Eclipse 平台(即凹形區塊)。工具程式的啟動、以及執行時所需的命令列指令、選項與參數，是由平台內的對應啟動器外掛程式(launching plug-in)所觸發與提供。啟動器是在整合時期利用我們提供的啟動器範本與精靈(即 Launching plug-in template 方塊)所自動生成。生成過程所需之各項資訊，例如工具程式之位置路徑、命令列的選項與參數格式、以及採用的對應圖形輸入元件描述(大都事先已放在 options.xml 檔)等，則是透過與整合人員的互動，由精靈在生成之前先予收集。

工具程式執行之後會產生或變動檔案資源，也會產生表示錯誤、警告或意外的說明訊息。這些輸出入檔、變動檔、以及訊息的去處以及格式，同樣地已經事先利用精靈收集完畢。因此當工具執行完畢時，啟動器一方面會依事先要求，進行資源更新；另一方面，則會自行啟動訊息連結程式，以抽取說明訊息中的相關檔案位置資訊並建立連結資訊(即 Errorlink 區塊)。最後工具使用者可由說明訊息的去處(即圖 3-2 的 console)看到執行的結果。若需要追溯或編輯問題根源則可經由雙擊該說明訊息以開啟對應檔案並自動移位至對應位置。

四、系統實作

在系統實作這一部份中，我們將簡要說明如何在 Eclipse 啟動外部的工具程式，然後定義規範選項格式的 XML Schema，接著再介紹如何以 PDE(Eclipse SDK 內附之外掛軟體開發套件)的外掛程式範本框架，自動建立啟動工具的外掛程式。我們接著闡述如何對於輸出在主控台的說明訊息，使用 ErrorLink 外掛程式以建立檔案的連結。最後，如果需要提供編輯器，我們說明如何利用 Colorer 外掛程式，以及如何產生定義檔案格式的 HRC[13] 檔。

4.1 啟動工具程式

啟動作業(launching)是指在 Eclipse 內執行(run)或除錯(debug)程式的動作。啟動器(launcher)則是一系列存在於 Eclipse 中可執行啟動作業的 Java 類別。因為 Eclipse 沒有內建啟動作業的功能，所以基本的 Eclipse 工作台(workbench)是不能啟動(launch)任何程式，只能透過外掛程式使 Eclipse 具有啟動的能力。Eclipse 的軟體開發套件(SDK)裡有提供一些實用的啟動器可用於 Java Applet、Java 應用程式、JUnit 或平台外掛程式等類型程式於測試或執行模式時的啟動。但是如果這些都不能滿足需求，又無法找到某個可執行此工作的外掛程式時，那麼就需要撰寫自己的啟動器。利用針對不同工具程式所設計的啟動器，使用者可以透過 Eclipse 的啟動架構來執行工具程式，並在主控台輸出執行後的結果。

除了以上提到的各式啟動器，Eclipse 平台使用者介面亦提供了通用的外部工具箱(External Tools)，讓我們可以配置和執行任何 OS 下的程式、批次檔、或 Ant 建置檔。平台也會自動儲存這些外部工具的配置(configuration)以便再次執行時使用。透過外部工具箱可以執行 OS 下的其它工具程式，產生的輸出會顯示在主控台視圖(console view)中。外部工具箱的最大缺點是所有外部工具均需共用工具箱提供的簡化的單一輸入介面，而無法像啟動器一樣提供專屬某一工具軟體的客製化親善介面。

4.2 選項格式

在執行工具程式之前，通常需要使用者輸入相關資訊，用以設定執行工具程式的選項，並讓系統知道該執行哪些動作或提供何種功能。但是在工具程式的開發過程中，如果要開發者自行處理命令列輸入的參數，分析輸入的字串再根據選項設定執行的功能，是一件十分瑣碎繁雜的工作。而且不同

的工具程式開發者，其所設計的選項與參數格式也都不盡相同，沒有一個統一的格式可以規範。

不同的工具程式理論上都可以有自訂的參數與選項格式，但這將使得輔助工具難以自動產生對應的使用者介面。基於此，我們參照 POSIX[8] 與 CLI 命令列處理器對輸入選項與參數的規範，利用 XML Schema 定義一個選項描述語言。

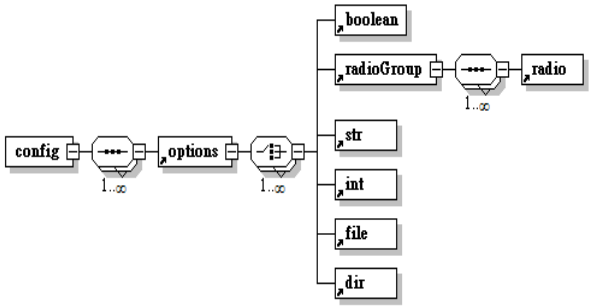

圖 4-1 選項格式

圖 4-1 是以 Altava XMLSpy 所製的此一語言的概略顯示(詳細細節請參考[14])。圖中 config 是根元素，其屬性內含工具程式執行檔的位置，並且以多個 options 子元素提供不同的選項群組。每一選項群組可有多個選項。選項的類型包括 boolean、radioGroup、str、int、file 和 dir。boolean 可以是不含參數的選項，以是否出現表示是否為真，也可以是帶有 true 或 false 兩種參數的輸入。radioGroup 則表示只能在此選項群組內選擇單一的 radio。輸入字串可以使用 str，或是輸入數字可以使用 int，檔案與資料夾則是以 file 與 dir 表示。除了 boolean 選項外，其他的選項是以名稱、連接號與值的方式輸入工具程式。它們的差異在於，針對不同的類型，我們可以產生不同的使用者圖形介面，以處理不同型態的輸入資訊。

啟動器配置介面的對話頁面可用來顯示及儲存來自於啟動配置屬性的相關資料。對話頁面以各種不同的 SWT 物件顯示各式各樣的選項格式，例如有勾選框、字串輸入、數字輸入、檔案輸入和目錄輸入等。透過這些不同的 SWT 物件，可以與使用者的互動，從而收集啟動工具程式時所需的參數。

利用此選項格式語言，工具整合者得以描述該工具程式的選項，再以我們設計的外掛程式剖析此 XML 檔案，終而產生可互動的圖形式對話頁，讓使用者得以輸入工具程式的選項與參數。

4.3 外掛程式範本

PDE(Eclipse 上的外掛程式開發環境)提供數個外掛程式的範本(Template)，讓用戶能以互動的方式，快速地以這些範本為基礎來建立外掛程式。我們應用 PDE 此一基礎架構的功能，在 PDE 上設計了一件外掛程式範本，可用以在整合時期為整合者合成可啟動工具的外掛程式。有此範本之後，若需將外部的工具程式整合到 Eclipse 平台，只需要撰寫定義該工具程式選項格式的 XML 檔案，就可以使用此外掛程式範本，產生工具使用時期所需之外掛程式。此外掛程式不僅能夠啟動工具並且也提供收集選項資訊所需的對話頁面，因此外部的工具程式可以透過此外掛程式的使用，達到與 Eclipse 整合的目的。

4.4 利用 ErrorLink 進行錯誤訊息連結

Errorlink[9] 主要是用以將主控台的錯誤或例外等說明訊息連結至相關檔案。此外掛程式可讓我們對主控台的輸出做比對，藉以從中抽取相關檔案的路徑、名稱、位置與行數資訊，並進而建立連結。之後只要雙擊錯誤訊息，即可開啟瀏覽產生錯誤的來源檔案。雖然 Eclipse 不知道怎麼處理錯誤訊息的文字字串，但 ErrorLink 可以讓我們由此得到檔案的連結。

為了由主控台的輸出文字訊息建立檔案的連結，ErrorLink 使用正規表達式(Regular expression)比對主控台上的輸出文字。ErrorLink 假設建立連結所需之檔名與位置訊息均可由輸出訊息取得，並且假設所有相關資訊都會出現在同一行內。因此用戶可提供數組正規表達式，用以決定哪些輸出行需要建立連結，此外對於每一需要連結的輸出行，我們尚須由中萃取相關的連結資訊。進階的正規表達式工具允許我們在表達式中利用括號建立子表達式: 當表達式與輸出行匹配成功時，子表達式對應的文字字串即可同時取得。因此 ErrorLink 允許我們在表達式中指定最多三個子表達式，分別對應至建立連結所需之檔案名稱，列號，與行號。其中最重要的部份是檔案的名稱，如果輸出訊息內沒有該檔案的完整位置，那就沒有辦法建立檔案的連結。很顯然的，ErrorLink 對於輸出訊息預設了太多的限制條件，因此許多軟體工具的輸出訊息都將不適用。我們目前使用 ErrorLink 只是權宜措施，未來將計畫擴充其功能，並增加其適用範圍。

4.5 用 Colorer 產生合適編輯器

Colorer 是一套可以提供語法標記(Syntax Highlighting)和本文剖析(Text Parsing)的程式，本身可以提供服務給平台的編輯系統做即時本文剖析，再將剖析的結果轉換成彩色的本文。剖析的結果還可允許搜尋和建立綱要視圖，此外也提供縮排(Indent)以及配對標籤(如左右括號)的快速匹配。Colorer 使用純粹的 C++和 XML，它是完全可攜的並且可在 win32 / unix / mac 平台上運作，也提供高層的 Java 應用程式介面。

EclipseColorer 是 Colorer 的 Eclipse 外掛程式。安裝之後，我們即可在 Eclipse 上使用 Colorer。然而 Colorer 雖然可立即支援上百種語言，卻不可

能包含所有語言。因此 Colorer 提供給用戶自定語言格式的功能：我們只需依其規定，擴充其 HRC[13] 檔案，以描述標的語言的語法和詞彙結構，即能提供編輯器剖析和分色顯示的能力。

Colorer 把標的語言的描述分為兩部份：prototype 與 type：type 部份提供的是標的語言的語法和詞彙結構描述；prototype 部份則提供標的語言的名稱、群組、對應檔案類型、首行特性以及對應的 type 檔案位置等資訊。所有標的語言的 prototype 資訊是共同放在外掛程式的 proto.hrc 檔，而每一標的語言的 type 則是以獨立的 HRC 檔案存放。因此為了支援新標的語言，我們必須修改此外掛程式的 proto.hrc 檔，以提供 prototype 部分的資訊，接著我們還要撰寫標的語言的 HRC 檔案，提供 type 部分的資訊，用以表示此語言的固定語彙、標記和語法。當 Colorer 剖析該類型檔案時，就依據此 HRC 檔案所設定的結構處理此類型的檔案。

五、範例：JavaCC

JavaCC[6] 是 Sun Micro 公司提供的一個文法剖析程式產生器，可以讀入一個定義好的文法規則，然後產生以 Java 實做的對應文法剖析程式。接下來我們分成幾個步驟，說明如何將 JavaCC 整合至 Eclipse 平台。

首先我們定義選項的格式，其中包含 JavaCC 的啟動位置和提供的選項。圖 5-1 是 JavaCC 選項格式的範例(詳細之實例請見[14])。

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="config.xsd" command="c:\javacc-3.2\bin\javacc.bat">
  <options columns="3">
    <int name="-LOOKAHEAD" label="-LOOKAHEAD" value="1" sep="=" desc="LOOKAHEAD        (default 1)"/>
    <int name="-CHOICE_AMBIGUITY_CHECK" label="-CHOICE_AMBIGUITY_CHECK" value="2" sep="=" desc="CHOICE_AMBIGUITY_CHECK (default 2)"/>
    <int name="-OTHER_AMBIGUITY_CHECK" label="-OTHER_AMBIGUITY_CHECK" value="1" sep="=" desc="OTHER_AMBIGUITY_CHECK (default 1)"/>
  </options>
  <options columns="2">
    <boolean name="-STATIC" label="-STATIC" value="true" sep="=" arg="true" desc="STATIC              (default true)"/>
    <boolean name="-DEBUG_PARSER" label="-DEBUG_PARSER" sep="=" arg="true" desc="DEBUG_PARSER          (default false)"/>
    <boolean name="-DEBUG_LOOKAHEAD" label="-DEBUG_LOOKAHEAD" sep="=" arg="true" desc="DEBUG_LOOKAHEAD      (default false)"/>
    <boolean name="-DEBUG_TOKEN_MANAGER" label="-DEBUG_TOKEN_MANAGER" sep="=" arg="true" desc="DEBUG_TOKEN_MANAGER   (default false)"/>
    <boolean name="-OPTIMIZE_TOKEN_MANAGER" label="-OPTIMIZE_TOKEN_MANAGER" value="true" sep="=" arg="true" desc="OPTIMIZE_TOKEN_MANAGER (default true)"/>
    <boolean name="-ERROR_REPORTING" label="-ERROR_REPORTING" value="true" sep="=" arg="true" desc="ERROR_REPORTING       (default true)"/>
    <boolean name="-JAVA_UNICODE_ESCAPE" label="-JAVA_UNICODE_ESCAPE" sep="=" arg="true" desc="JAVA_UNICODE_ESCAPE   (default false)"/>
    <boolean name="-UNICODE_INPUT" label="-UNICODE_INPUT" sep="=" arg="true" desc="UNICODE_INPUT         (default false)"/>
    <boolean name="-IGNORE_CASE" label="-IGNORE_CASE" sep="=" arg="true" desc="IGNORE_CASE           (default false)"/>
    <boolean name="-COMMON_TOKEN_ACTION" label="-COMMON_TOKEN_ACTION" sep="=" arg="true" desc="COMMON_TOKEN_ACTION  (default false)"/>
    <boolean name="-USER_TOKEN_MANAGER" label="-USER_TOKEN_MANAGER" sep="=" arg="true" desc="USER_TOKEN_MANAGER    (default false)"/>
    <boolean name="-USER_CHAR_STREAM" label="-USER_CHAR_STREAM" sep="=" arg="true" desc="USER_CHAR_STREAM      (default false)"/>
    <boolean name="-BUILD_PARSER" label="-BUILD_PARSER" value="true" sep="=" arg="true" desc="BUILD_PARSER         (default true)"/>
    <boolean name="-BUILD_TOKEN_MANAGER" label="-BUILD_TOKEN_MANAGER" value="true" sep="=" arg="true" desc="BUILD_TOKEN_MANAGER  (default true)"/>
    <boolean name="-SANITY_CHECK" label="-SANITY_CHECK" value="true" sep="=" arg="true" desc="SANITY_CHECK         (default true)"/>
    <boolean name="-FORCE_LA_CHECK" label="-FORCE_LA_CHECK" sep="=" arg="true" desc="FORCE_LA_CHECK        (default false)"/>
    <boolean name="-CACHE_TOKENS" label="-CACHE_TOKENS" sep="=" arg="true" desc="CACHE_TOKENS          (default false)"/>
    <boolean name="-KEEP_LINE_COLUMN" label="-KEEP_LINE_COLUMN" value="true" sep="=" arg="true" desc="KEEP_LINE_COLUMN     (default true)"/>
  </options>
  <options columns="3">
    <dir name="-OUTPUT_DIRECTORY" label="Output Directory:" sep="=" desc="OUTPUT_DIRECTORY    (default Current Directory)"/>
    <file name="" label="Input File:" sep="" desc="Please input your grammar file."/>
  </options>
</config>
```

圖 5-1 JavaCC.xml

定義選項的格式之後，接著使用我們先前所設計的範本，以建立可以在 Eclipse 啟動 JavaCC 的外掛程式。以下是使用該外掛程式範本的過程，首先我們以圖 5-2 所示的方式建立一個新的外掛程式專案：

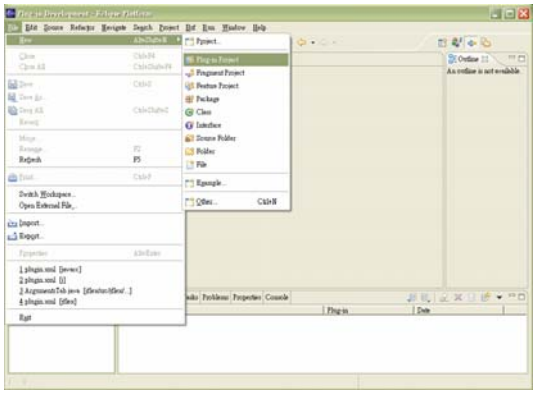

圖 5-2 建立新的外掛程式專案

在設定外掛程式的相關內容後，使用我們所提供的範本(圖 5-3)，並輸入定義 JavaCC 選項格式的 XML 文件(圖 5-4)。

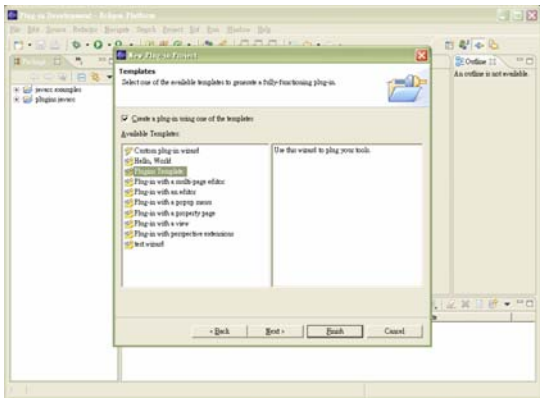

圖 5-3 可外掛軟體的範本

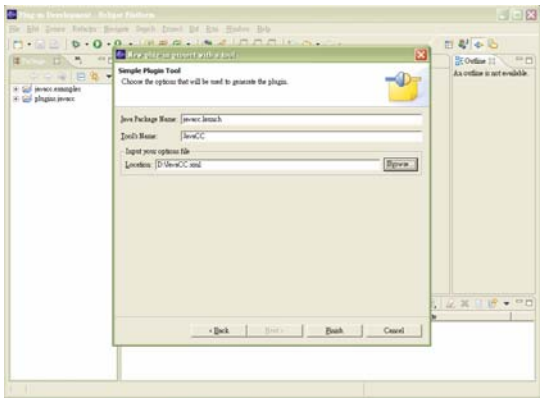

圖 5-4 輸入 JavaCC 套件名稱、工具名稱與定義選項格式的 XML 檔案

最後按下“Finish”鍵之後，分析此選項格式的 XML 檔案，以產生處理選項的標籤頁面，我們可在 Eclipse 建立用於啟動 JavaCC 的外掛程式專案(圖 5-5)。

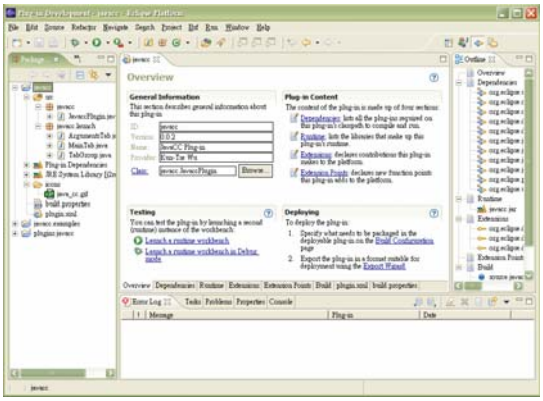

圖 5-5 產生可以啟動 JavaCC 的外掛程式專案

圖 5-6 是安裝此外掛程式後，所提供的啟動配置類型，我們可以在標籤群組設定選項參數和更新資源等標籤頁面，並以“Run”鍵啟動 JavaCC 的工具程式。

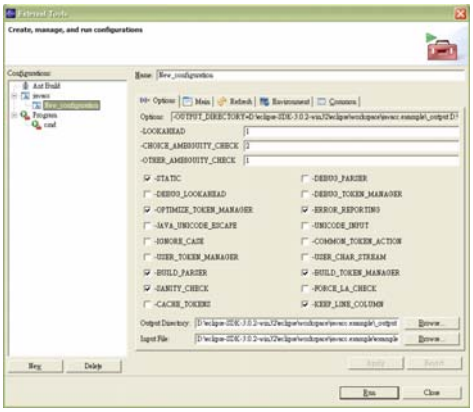

圖 5-6 JavaCC 的啟動配置類型，於標籤群組中提供設定選項參數的標籤頁面

為了在輸出執行結果的主控台端提供更多的支援，例如當 JavaCC 的文法檔案有錯誤時，我們希望直接在主控台端建立檔案的連結。因此我們使用 ErrorLink 分析主控台的文字訊息，並且建立檔案的連結。使用 ErrorLink，我們要安裝 SunShade 外掛程式，並在 Eclipse 的喜好設定(Perference)內，於 SunShade 的 ErrorLink 加入“Reading\sfrom\sfile\s(.*\.jj).*”的描述。

接著我們希望在 Eclipse 編輯 JavaCC 的文法檔案，因此要對.jj 與.jjt 的檔案提供文字編輯器。文字編輯器可以由 Colorer 實作，而在 EclipseColorer-take5_0.7.0 的版本即提供 JavaCC 的文法檔案，所以只需要安裝 EclipseColorer-take5_0.7.0 的外掛程式，並在喜好設定 > 工作台 > 檔案關聯內，設定開啟.jj 與.jjt 副檔名的編輯器類型，即可在 Eclipse 編輯該副檔名的檔案。圖 5-7 是 JavaCC 整合至 Eclipse 的範例：

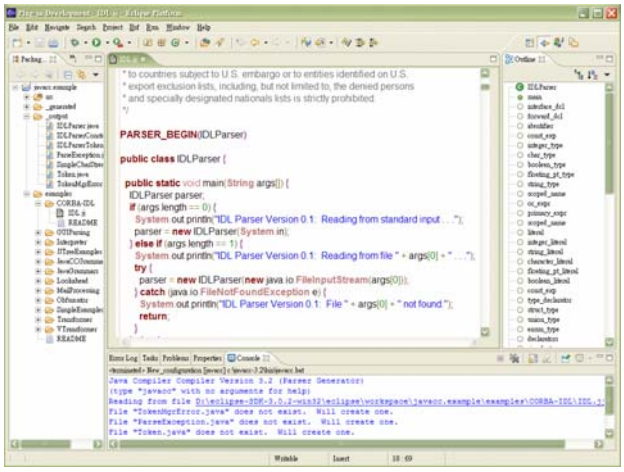

圖 5-7 提供編輯器與主控台的檔案連結

以上是我們透過可外掛軟體的規範，將 JavaCC 整合至 Eclipse 環境的過程，接著我們將比較此種方式與 Eclipse 現有的 JavaCC 外掛程式。在 Eclipse 外掛程式清單[2] 的網站內，我們找到兩個 JavaCC 的外掛工具程式，分別是 com.subx.eclipse.javacc_0.9.5 和 rk.eclipse.javacc_1.1，表 5-1 說明三者之間的差異。

表 5-1 JavaCC 外掛程式比較表

	com.subx.eclipse.javacc_0.9.5	rk.eclipse.javacc_1.1	Pluggable Software
提供選項	X	O	O
編輯器	X	O	O
檔案連結	X	X	O

六、結論與未來研究方向

6.1 結論

當需要將工具程式整合至 Eclipse 平台時，可以在 Eclipse 設計一啟動器以啟動該外部工具程式，並且利用現有的外掛程式，在主控台建立檔案連結與提供檔案編輯器。這種做法可快速將既有或新開發軟體工具整合至 Eclipse。

為能達到上述便利，工具程式必須滿足所謂可外掛軟體的規範，其目地是希望能在不更動工具程式碼的前提下即能將該工具整合至 Eclipse。

為方便所有可外掛軟體整合至 Eclipse，我們提供以下幾項輔助：第一，統一工具程式的輸入：在啟動工具程式之前，使用者可以使用經由輔助工具產生的圖形介面以收集工具所需之輸入選項與參數。第二，提供啟動外掛程式的範本：為了啟動

外部的工具程式，因此我們根據 Eclipse 的啟動架構，延伸 PDE 外掛程式的範本，建立可產生啟動外部工具程式的外掛程式範本。第三，工具程式執行後會在主控台輸出結果，為更進一步提供其它的功能，我們以 ErrorLink 在主控台建立檔案的連結，當開啟特定的類型檔案時，可能需要提供合適的編輯器，我們以 Colorer 提供編輯器的簡易做法。

6.2 未來研究方向

目前整合可外掛軟體的方式是透過啟動一個外部歷程(external process)的方式以執行該軟體工具，其整合程度並不如共享一個 Eclipse 歷程的其他外掛程式。因此如何將工具程式完全轉為外掛程式，讓工具程式與 Eclipse 平台更完整地結合，是未來值得研究的一個議題。

本文僅對以命令列方式執行的工具程式提供快速整合至 Eclipse 平台的輔助工具。因此下一階段的一個重要研究議題，毋寧是如何擴充這些工具的適用範圍至具備使用者圖形介面的工具程式。

對具備使用者圖形介面的工具程式而言，要將之整合到 Eclipse 平台，並非易事。此乃因為 Eclipse 具有自己的使用者圖形介面 SWT 和 JFace，與大多數工具程式採用的 AWT 或 Swing 等主流使用者圖形介面並不相同。整合使用者圖形介面的方式有很多種，像是 Eclipse 在 Windows 平台就提供將 AWT 和 Swing 元件嵌入 SWT 內的做法[12] 。另一種做法是在 Eclipse 平台的外部啟動使用者圖形介面[11] ，然後在 Eclipse 平台的工作區(workspace)與檔案系統間建立相連性。後者做法與我們現在對命令列工具的做法相類似，因此目標應較易達成。然而這種方式並非優良整合，主因是共存的不一致使用者圖形介面將容易造成使用者混淆。

我們在延伸 PDE 範本外掛程式時，發現到可以使用範本建立外掛程式，將開發過程所需的各個部份設計成通用的範本，並由開發者輸入建構工具程式所需的資訊，即可利用範本產生外掛程式。PDE 目前既有的範本有動作集(ActionSet)、內容頁面(Property Page)、編輯器、蹦現功能表(PopupMenu)和視景(Perspective)等。如果我們可以設計更多不同的範本，將工具程式的各部份都寫成範本，這麼一來，開發工具程式時只要根據所需的部份，由開發者填入所需的資訊，使用範本就可以輕鬆地建構出來。

市場上目前有很多類似於外掛程式的元件技術，例如 CORBA、DCOM、Java Bean、Web Service 等。這些不同架構的元件有無可能經適當轉換，直接變為 Eclipse 的外掛程式，是值得進一步研究的方向。

參考文獻

[1] The Colorer Library Project. *http://colorer.sourceforge.net/*

[2] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, *Addison Wesley*, 1994.

[3] Eclipse Plug-ins. *http://www.eclipse-plugins.info/eclipse/index.jsp*

[4] Eclipse.org. *http://www.eclipse.org/*

[5] Jakarta Commons CLI. *http://jakarta.apache.org/commons/cli/*

[6] Java Compiler Compiler. *https://javacc.dev.java.net/*

[7] Neal Stephenson. In the Beginning was the Command Line. *http://www.spack.org/index.cgi/InTheBeginningWasTheCommandLine*

[8] POSIX Conventions for Command Line Arguments. *http://java.sun.com/docs/books/tutorial/essential/attributes/_posix.html*

[9] The SunShade project. *http://sourceforge.net/projects/sunshade*

[10] Will Robinson, Ben D'Angelo: Integrating software productivity tools into Eclipse. *OOPSLA Workshop on Eclipse Technology eXchange* 2003: 40-44.

[11] Jim Amsden, Levels Of Integration, *http://www.eclipse.org/articles/Article-Levels-Of-Integration/Levels%20Of%20Integration.html*, 2001.

[12] Eclipse documentation for the SWT AWT embedding support, *http://help.eclipse.org/help31/index.jsp?topic=/org.eclipse.platform.doc.isv/reference/api/org/eclipse/swt/awt/package-summary.html*, 2005.

[13] HRC language reference, *http://colorer.sourceforge.net/hrc-ref/index.html*.

[14] Kun-Tse Wu, The Design and Construction of Pluggable Software Architecture for the Eclipse Platform, *Master's thesis, Department of Computer Science, National Chengchi University*, 2005.