行政院國家科學委員會專題研究計畫 成果報告

Eclipse 外掛程式超模型之設計與相關工具開發
研究成果報告(精簡版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 97-2221-E-004-008-
執 行 期 間 ：97 年 08 月 01 日至 98 年 07 月 31 日
執 行 單 位 ：國立政治大學資訊科學系

計 畫 主 持 人 ：陳正佳

處 理 方 式 ：本計畫涉及專利或其他智慧財產權，2 年後可公開查詢

中 華 民 國 98 年 12 月 31 日

報告內容:

一、前言

Eclipse 是近年竄起的一個整合式軟體開發環境，由於策略的成功以及技術優勢，如今已成為市場上的主流軟體工具平台。Eclipse 事實上可同時或分別擔任軟體發展工具(software development tools)，軟體元件(software components)或客戶端應用程式(rich client applications)的整合平台，其於軟體工具平台之地位猶如現今作業系統下之 Linux 與 Windows。雖然有人直指 Eclipse 為下一世代的 Emacs，依作者個人觀點，Eclipse 之應用廣度與親善性實遠大於 Emacs。

除了在軟體架構上具有無限的可擴充性，允許其他軟體整合至此平台之特色外，Eclipse 還允許用戶與開發者免費衍生使用，是以開發者莫不期望將其軟體整合至此平台執行。擴充 Eclipse 平台功能的方式是將工具程式包裝成所謂的外掛程式(plug-ins)而整合至平台. 因此，若要獲得 Eclipse 平台的益處，首要條件是要具有 Eclipse 外掛程式的開發能力。美中不足的是，實際開發經驗顯示，發展 Eclipse 外掛程式並非易事。究其原因，一方面，Eclipse 平台本身是一極度複雜之系統，具有龐大而複雜的應用程式介面(API)，開發者必須學習如何使用、擴展與實做；另一方面，Eclipse 複雜的軟體框架往往使開發者陷入困惑，無所是從。因而就一般程式師而言，外掛程式開發的跨入門檻極高，需經歷冗長而陡峭的學習過程。而即或是已適格之程式師，亦將發現，因同一功能之實做往往需要散射至 plug-in 的許多不同位置(code scattering)，以現有工具與傳統模式開發 Eclipse 程式，仍然相當繁瑣與不便。

因應之道之一是發展支援工具以輔助開發者。例如 Eclipse SDK 上已經內建一個 plug-in 開發工具稱為 PDE，可有效的幫助開發者降低許多的煩瑣與相依性檢驗。然而此工具乃建立於 Java 程式設計之基礎上，開發者需以傳統程式設計模式設計 plug-ins，因而仍無法擺脫 API 呼叫與框架套用上的種種細節。除此外，市場上有些工具並不鼓勵開發者，一開始就以特定程式語言編輯與發展程式；相反地，它們支援並鼓勵客戶利用他們的工具進行軟體工程中所要求的寫碼前各階段工作如:需求分析、系統分析與系統設計。至於程式碼的產生部分則大部分可靠各種內建或客戶自訂的程式碼產生器，自動由系統設計階段所產出的系統模型轉換而成。因此當我們採用這些以模型為基礎的工具時，寫碼工作可以降底許多。而這種軟體系統開發方法推廣到極致，就是以 OMG 的模型驅動架構 MDA[12]為理論基礎的所謂的模型驅動軟體工程。這種開發方式的特性是:不以程式碼，而是以設計階段的正確系統模型為軟體開發的目標，程式碼及更下游的機器碼則應全由工具程式自動產出。換言之，MDA 希望將軟體開發的自動化程度提升一層:將現今由編譯器自動產生組合語言或機器碼的層次，提高到由模型轉換器自動產生 C 或 Java 等高階程式碼。因此，開發者所需的關注也就提升到如何獲取正確的模型以便轉換器據以自動產生程式碼。

我們因而計畫依據模型驅動之理念，提供一套以模型而非程式為核心的外掛程式開發架構，期使開發者得以運用 Eclipse 超模型發展外掛程式。依此構想，開發者將在 Eclipse 平台元件超模型的導引之下，創建其所需之外掛程式模型，接著再利用程式產生器產生外掛程式骨幹與部分實做，最後再內嵌必要之程式碼而得到完整之實做。此構想之特徵為開發者僅需關注專長之軟體邏輯設計，而無需知曉 Eclipse 平台框架與元件的呼叫細節，其好處猶如以視覺化之建構器而非以 API 開發 GUI 程式。

而欲建立此架構須完成幾項任務: 一是為 Eclipse 平台的主要常用框架建立超模型，以供開發者遵循據以產生其外掛程式模型，其次則是建構模型至 plug-in 實做的種種模型轉換與程式碼產生器。而我們在此一計畫的主要結果是為 Eclipse 平台的主要常用框

架與 API 建立超模型，以供開發者遵循據以產生其外掛程式模型;其次則是規畫建構相關的模型轉換與程式碼產生器，以供模型開發者產生對應的外掛程式骨架與部分實做。

二、研究目的

如前所述，Eclipse 是目前市場的主流軟體工具平台，其於軟體工具平台之地位猶如現今作業系統之 Linux 與 Windows。因此，開發者莫不期盼開發以 Eclipse 為承載平台之軟體或將其既有軟體整合至此平台執行。然而，由於系統之複雜，依據傳統模式，以程式為核心的開發方式，將使 Eclipse 外掛程式開發，變得極度的繁雜與困難。而依據 MDA 的理念，以模型為核心的方式，將可使開發者免於 Eclipse 軟體框架與 API 之細節，而可集中在專精的邏輯設計。然而以模型驅動方式開發程式的要件是必須具有對應領域的超模型與輔助的模型轉換與程式碼產生器。

因此，我們此一計畫的主要任務即是探討並實際為 Eclipse 平台的主要常用框架與 API 建立超模型，以供開發者遵循據以產生其外掛程式模型。此外，我們也需要提供必要的模型轉換工具，以便將高階之 PIM(Platform Independent Model) 轉為含有更多實作細節之 PSM(Platform Specific Model)。最後再經由程式碼產生器，將 PSM 轉換為包含程式骨架以及大部分實做的程式碼。

三、文獻探討

以手寫以外的方式自動產生 Eclipse plug-ins 的系統實際上是非常普遍存在的。在 Eclipse modeling 計畫群之內的許多計畫如: EMF, GMF, OAW, 以及我們自行發展的 EGOE[47]系統均允許開發者在後面階段自動產生 Eclipse plug-ins。然而與我們的計畫的最大的不同是，這些工具所產生的是 plug-ins 是為了實做特殊應用而由工具程式所產生的特殊用途 Plug-ins。舉例而言，EMF 可以為開發者輸入之 Ecore 超模型產生對應的樹狀結構式模型編輯器，其中包含各式選單(Menu & ToolBar)，偏好設定對話頁，編輯器與各種視圖(View)。然而這些外掛程式，都是為了支援模型編輯器而存在的，我們無法令 EMF 為我們產生除此之外的任何一個外掛程式。同樣的道理，GMF 可以為開發著產生圖型編輯器，EGOE 與 OAW 可以為開法者產生文字式編輯器，及各種功能支援外掛程式，但他們無法產生這些功能以外的任何外掛程式，例如產生一個可以為客戶產生新檔的對話精靈。

Eclipse Plug-ins 具有相當大的比例是屬於 GUI 設計，而在 GUI 設計上，目前有越來越多的趨勢是盡量改用 Declarative 的方式以取代傳統的 imperative 程式設計。最大好處是可以降低程式碼數量，同時又增加程式之可讀性與可維護性。此一方面的進展幾乎都是以 XML 為表達語言，最早出現的是 UIML，其後類似語言相繼出現，如 Mozilla 之 XUL, 以致最近之 XAML，Laszlo 以及 JavaFX 均屬之。至於應用於 Eclipse 的 SWT 與 JFface 等 GUI 工具的 XML 語言則有 XSWT 與 eFace.

以生成方式產生全部或部分程式，以替代人為手寫的研究，事實上，存在已久。此領域稱為 Generative Programming，每年都有一個名為 Generative Programming and Component Engineering(GPCE) 的國際會議針對此一領域與元件技術舉行。然而這些研究，主要是針對各式技術，理論與應用，而我們的計畫則可視為是此一領域的一個特殊領域應用。在 code generation 的技術上，我們所用的大部分是屬於範本套用的技巧。此類技術已普遍應用於各領域如: 網頁設計之 JSP, 一般用途範本引擎 Velocity 與 freeMarker， 文法程式分析器合成器(StringTemplate)，XML 文件轉換(XSLT)等。然而我們採用的 code generator 有兩大特性不見於前述工具:其一是我們的工具不能只

產生單一檔案，必須能快速同時產生多個檔案與目錄結構；其二是我們的工具能允許開發者在生成的檔案中插入額外資訊，同時又能允許新版合成檔案能保留來自舊檔的手寫資料。具備此一功能的工具，目前僅見於 EMF 之 JET 與 OAW 之 XPand2。

在模型轉換上，編譯器無疑是一個歷史最久同時功能也最複雜的模型轉換器。除此外，自 MDA 概念提出之後，發展出來的專用與一般用途模型轉換器已屢見不鮮，目前最普遍的包括 ATL，SmartQVT, Borland's Together QVT，IBM's MTF， Kermeta 等。有關各種 Model Transformation 的比較與分類，可見[50]。至於早於模型轉換的更一般理論則屬 Graph transformation[50]，相關工具與應用如 AGG 等則可見於同一網站。

四、研究方法

根據 MDA 指引[12]和主要參考書籍[01,02,38,39]，MDA 主要是建立在以下幾項概念:

1. 以高階語言原始碼為軟體發展標的的概念應予以提升至以模型為標的。
2. 一個模型若要能轉換成完整的原始碼實做，則其模型內容除了與標的平台無關的系統模型之外，亦應包含與標的平台相關的模型資訊。因為此種模型內含與特定平台有關資訊，故稱之為平台專屬模型(Platform Specific Model; PSM)。
3. 開發模型時應採用分離關注(Separation of Concerns)的原則，分別發展與標的平台無關的系統模型(稱作 PIM; Platform Independent Model)，以及發展所謂的模型轉換器(PIM2PSM mapping)，以便得以將 PIM 轉換成特定平台的 PSM。PIM 型塑的是企業或應用系統邏輯，因而通常是系統分析階段的結果，而所謂的系統設計，研究的是如何將系統分析結果實現在標的平台上，因此其目的就是要產生模型轉換器。
4. 通常一個平台可以有許多種方式實現同一種系統功能，反映到傳統系統設計的就是設計選擇與決策，同樣的，模型轉換器通常不是一個函數，而是一個關係，而使其變成函數而得以產生唯一 PSM 的就是設計偏好(DP; Design Preferences)。設計偏好在 S.J.Mellor[01]書中是以所謂的標記(Marks)來描述。有關 PIM, PSM, DP, PIM2PSM 之間的關係可以用以下等式描述:PSM = PIM2PSM(PIM, DP)，而在[01]中則使用以下等式: PSM = PIM2PSM(M(PIM))，其中 M(PIM)代表的是加註設計偏好之後的 PIM.

應用模型驅動方式開發軟體所需面臨的第一個問題是模型與超模型的表示方式。理論上，當然可以各自發展塑模語言以為表示方法，但是標準化卻是不同商家之工具與產品可互相操作的必要條件。在此項標準上，OMG 的 UML[14]，或者是利用 MOF[15]或 EMF[33]Ecore[11]自訂的塑模語言(或超模型)是可能的選擇。此部分我們決定使用 Ecore,一方面是因她較 UML 具有更大彈性,另一方面則是其支援資源工具以非常齊備。除此外，我們累積的 EMF 使用經驗，使我們可以快速脫離學習，進入實際任務。

了解 PSM 與 PIM 應如何表示之後，下一個主要問題是如何表達模型轉換。這一方面的語言與工具，相對於程式碼合成器，在 MDA 提出之時仍屬稀少，而 OMG 承諾制定的一套標準模型轉換語言 QVT[29,30,32]則已通過 OMG 的最後審核。目前市面上已經有少許實做[23,31,42]，可用以執行模型間的轉換；除此外 Eclipse 下的 Modeling 專案群提供了模型管理架構與各種轉換工具，我們使用其中之 ATL (ATLAS Transformation Language)做為模型轉換語言。

QVT 以及 ATL 基本上均適用在模型至模型之間的轉換，然而當我們獲致專屬於 Eclipse 平台的 PSM 時，我們還需將之轉換為由 Java 原始碼檔以及相關資源與描述檔所構成的外掛程式專案，以便後續的 PDE[19]接手進行建構與封裝。因此為使我們的工作能達成，通常 PSM 至程式實做間我們必須進行程式碼合成。這部份我們則決定使用 JET[41]主因是因為它是 Ecore 工具 EMF 的一部份，可以和 EMF 完全整合。有關模型至

程式的轉換工具,現有工具如[22,23]均提供各自的完整的語言供客戶使用。Open source 的部份則可使用 AndroMDA[43]、EMF[33]上的 JET[41]範本語言、或使用 MODELWARE[34] 計畫下的 MOFScript[35],此 Eclipse 外掛程式事實上實做的是 OMG 正在推動的通用的模型至本文轉換語言[36]的原型。

確立模型表達與轉換工具之後，下一考量，即是PSM, PIM, PIM2PSM, PSM2Code (模型至程式碼)等實體的內涵為何。簡單的分析之後，馬上就可獲致以下結論: 外掛程式的組成是由兩部分組成:一者是與Eclipse平台無關的工具程式邏輯；一者則是用於支援工具程式的Eclipse元件或框架。因此，PIM表示的應該是與Eclipse平台無關的軟體工具部分；PIM2PSM轉換則負責陳述PIM上的各元素如何與Eclipse元件編織而成為具有完整Eclipse平台外掛程式功能的一個PSM。至於PIM與程式碼之間，基本上只是視為同一語意的兩種不同表法，彼此間應該是同構的(Isomorphic)。因此只要PSM的內容完整無矛盾，套用Visitor pattern或應用Template即能產生對應程式碼。唯若要能夠設計程式合成器，必須對採用工具，外掛程式專案的組成結構、以及PIM的模型架構有充分的熟悉與掌握。綜合前述，我們確認PSM的模型結構(PSM MetaModel 或 PSM描述語言)是最容易分析定義的，其主因乃是因為PSM與完整外掛程式碼同構，而外掛程式碼專案的完整結構在Eclipse平台上是具有清楚定義而可以分析得到的。

一旦得知PSM的模型結構之後,接下來進行的就是模型分離的工作。我們需要針對PSM模型內的所有可能型態的元素分析其組成來源以及組成方式。組成來源依其是與平台相關或與工具相關而分別歸屬為PIM元素或PDM(Platform Description Model)[12,49]元素。至於組成方式則歸類為PIM2PSM轉換器的一部份。因此我們的研究方法是Bottom-up的逆向分析。先由閱讀學習,了解 Eclipse的主要元件與外掛結構,接著利用同構概念，反向訂出PSM的模型結構，然後再經模型分離分析，解構出PIM和PDM的必要結構，以及PIM2PSM 轉換器的必要內涵。值得一提的是，此種逆向分析所形構出來的metamodel通常不會完備，我們應該還需要以自訂或內插方式補足其欠缺。

五、結果與討論

本計劃申請時的主要目標有二。一為從現有 Eclipse 平台上的各種框架與 API，取最常被開發者引用的部分，提供更高層次的模型描述，稱之為超模型。有了超模型之後，開發者即可利用這些超模型，在 EMF 編輯器之下，編輯產生超模型實例，即是傳統所謂的模型。其次則是規畫建構相關的模型轉換與程式碼產生器，以供模型開發者產生對應的外掛程式骨架與部分實做。

在超模型的設計上，我們規畫了:

1. 與 Eclipse 功能觸發有關之超模型。此部分的主要概念或框架包括:選單區(Menu bar)、選單(Menu)、彈跳式選單(Context Menu)、選項(Menu Item)、工具箱(Tool Bar)、動作(Action)、動作集(Action Set)、指令(Command)、指令處理器(handler)、鍵結(Key Binding)等。除了為上述基本概念定義超模型之外，我們也定義了以下超模型:視景動作(View Action)、編輯器動作(Editor Action)、彈跳選單動作(Context Menu Action)，全域動作(Global Action)、指令組(Command),等。
2. 與資訊閱覽相關之超模型。此部分的主要概念是視圖(View)，它是 Eclipse 用以輔助使用者瀏覽相關資訊的各種內嵌輔助視窗。Eclipse 平台實際上，已經預建許多視圖,例如: Resource Nevigator,Console,Task View, Problem View, Content Outline View, Property View.此部分的結果包含為視圖(View)定義一個通用的超模型，以供程式師開發其他視圖。而另一個重大的任務則是，針對上述特殊視圖，提供延伸點超模型，原因是因為這些視圖本身就是一種接受其他外掛程式奉獻的可延伸平台服務，因此開發者需要有一實用超模型以供其產生對應之延伸。

3. 與資訊編輯相關的編輯器(Editor)超模型。Eclipse 有一個功能及強但也極其複雜的編輯器框架，因此如何為其定義一個完整的超模型，事實上仍有相當困難，而我們目前的方式是僅為其基礎功能部分建立其超模型，至於更進階部分，我們在[46]部分也已有相關探討。
4. 對話盒(Dialog)與導引精靈(Wizard)超模型。Eclipse 具有為數眾多的對話與精靈，可以提供給用戶互動式資料輸入，因此也是開發外掛程式，亟需具備的組件。在導引精靈部分，我們也規畫設計一通用精靈語言，利用此語言，可將平台無關的精靈描述，實做在許多不同平台上。
5. 喜好與設定(preference and dialog settings)超模型。所有軟體工具均具有各式各樣供使用者自行決定的喜好設定。當工具程式整合至 Eclipse 平台時，當然這些設定亦必依平台之喜好設定框架隨之整合至平台。我們需要為喜好設定框架定義超模型，已公開發者快速描述其喜好設定。
6. 插件(Plug-in)超模型。前述各項超模型描述的 Eclipse 平台的功能組件，然而這些組件並無法獨立存在，而必須依附在所謂的 Plug-in 實體專案之下。因此，為了能描述組件位置，我們必須為 Plug-in 專案提供一超模型描述。

除了上述開發者最常使用的功能之外，事實上還有許多其他框架需要研擬其超模型。例如，Eclipse 的線上輔助說明系統，Cheat Sheet，Tutorial 等，而更複雜的還包括外部程式呼叫系統，測試與偵錯系統，乃至 JDT 與 PDE 的相關超模型等。而實際上，這個未完成事項是不可能成空的，因為 Eclipse 是一個為微核心的可擴充平台，不同的工具均可納入此平台，而接受其他工具的整合要求，因此理論上我們都應為其建立超模型。當然，實際上是否有此需要，乃取決於需求者是否夠多以及超模型是否容易定義。

另外值得一提的是，前述的各種超模型，依據分離關注原則，實際上每一項目均包含兩部分:一者為 MPIM，一者為 MPSM，MPIM 是做為 PIM 的超模型，MPSM 則是做為 PSM 的超模型。MPIM 提供的是與框架/API 引用細節無關的精簡模型建構指引，可以提供一般開發者快速產生外掛程式模型;而 MPSM 提供的則是與框架/API 引用有關的完整模型定義，可供程式產生器據以產生預期的外掛程式骨架與部分實做。

在模型轉換以及模型至程式碼的轉換實作上，由於我們是採用 EMF 的 Ecore 的關係，EMF 為我們提供了許多的便利。在 PIM2PSM 的模型轉換部分，我們目前的做法，基本上是仿效 EMF 由 Ecore model 變為 GenModel 的做法:首先要求每一 PSM 必須能指稱至對應的 PIM，因而得以將 PIM 視為是 PSM 的一部分，而 PIM 的任何變化也自動反應至 PSM，如此可大為降底 PIM/PSM 之同步與一致性問題。其次則是，PSM 所必須提供的許多程式碼必要資訊，往往並不出現於 PIM，因而如何取得這些資訊，實為一重要研究課題。我們目前的做法則是採用類似套表之做法，針對 PSM 的某些特定結構，設定內容產生規則，利用觸發這些規則或方法即可取的對應內容。而規則裡最常出現的是所謂的預設規則，她允許系統設計者為某些屬性或結構提供常設值。實做這類模型轉換的工具，採用的是 Eclipse Modeling 計畫群下的 ATL，我們的模型轉換功能，可以以之實現。未來我們則計畫，使這些規則可以被覆蓋，精煉，因此同一 PIM，開發者可以產生不同版本的 PSM。

依據 MDA 理論，模型至程式碼的轉換反而是 MDA 過程中最簡單的部分，因為依據 MDA，所有與程式碼產生有關的資訊，均必須明示於前端 PSM，因而程式產生器將有足夠資訊產生對應程式。然而就外掛程式而言，由於程式碼分散於不同檔案(code scattering)的關係，我們必須同時產生多個不同輸出，更麻煩的是，這些輸出必須與其他功能共用同一檔案而無法獨立擁有，因此我們必須將這些產出織入至不同標的檔(Code Weaving)。

參考文獻：

[01] Stephen J. Mellor, Kendall Scott, Axel Uhl, Dirk Weise, MDA Distilled: Principles of Model-Driven Architecture, Addison Wesley, 2004.

[02] Anneke Kleppe, Jos Warmer, Wim Bast, MDA Explained: The Model Driven Architecture: Practice and Promise. Addison Wesley, 2003.

[03] Jos Warmer, Anneke Kleppe, Object Constraint Language: Getting Your Models Ready for MDA, 2nd Edition, Addison Wesley, 2003.

[04] Grady Booch, James Rumbaugh, Ivar Jacobson , The Unified Modeling Language User Guide, 2nd Edition, Addison Wesley, 2005.

[05] Frankel, David, Model driven architecture : applying MDA to enterprise computing, Wiley, 2003.

[06] Daniel J. Duffy, Domain Architectures: Models and Architectures for UML Applications, Wiley, 2004.

[07] Jeff McAffer, Jean-Michel Lemieux, Eclipse Rich Client Platform: Designing, Coding and Packaging Java Applications, Addison Wesley, 2005,

[08] Jim D'Anjou, Scott Fairbrother, et al. The Java Developer's Guide to Eclipse, 2nd Edition, Addison Wesley, October 2004.

[09] John Arthorne and Chris Laffra, Official Eclipse 3.0 FAQ, Addison Wesley, June 2004.

[10] Eric Clayberg and Dan Rubel,Eclipse: Building Commercial-Quality Plug-Ins, 2nd Edition, Addison Wesley, 2006.

[11] Frank Budinsky, Timothy Grose, et al. Eclipse Modeling Framework, Addison Wesley, August 2003.

[12] Object Management Group, MDA Guide V1.0.1. http://www.omg.org/cgi-bin/doc?omg/03-06-01

[13] Object Management Group, MDA web site. http://www.omg.org/mda

[14] UML web site. http://www.uml.org

[15] Object Management Group, MOF web site. http://www.omg.org/mof/

[16] Eclipse web site. http://www.eclipse.org.

[17] Eclipse platform project. http://wwwe.clipse.org/ecipse/

[18] Eclipse Java development Kit. http://wwwe.clipse.org/jdt/

[19] Eclipse plug-in development environment. http://wwwe.clipse.org/pde/

[20] Eclipse GMT project. http://www.eclipse.org/gmt/

[21] Eclipse ATL subproject. http://www.eclipse.org/gmt/atl

[22] Sparx systems, Enterprise Architect. http://www.sparxsystems.com.au/

[23] Borland, Together Architect. http://www.borland.com/us/products/together/index.html

[24] No Magic, MagicDraw UML. http://www.magicdraw.com/

[25] Jack Greenfield, Keith Short et al. Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. John Wiley & Sons, 2004.

[26] Erich Gamma, Richard Helm et al. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.

[27] Ian Sommerville, Software engineering. Addison Wesley, 2005.

[28] OMG, Catalog of OMG Domain Specifications. http://www.omg.org/technology/documents/domain_spec_catalog.htm

[29] OMG, Revised submission to the MOF 2 Q/V/T RFP. http://www.omg.org/cgi-bin/doc?ad/2005-07-01

[30] QVT-Partners, QVT Partners revised submission to QVT. http://qvtp.org/downloads/1.1/qvtpartners1.1.pdf

[31] QVT-Partners, qvtp-eclipse. http://qvtp.org/downloads/qvtp-eclipse/

[32] Interactive ObjectsSoftware GmbH, Inc Project Technology, and Unisys Corp : Revised Submission to MOF Query / View / Transformation RFP. http://www.omg.org/ docs/ad/03-08-11.pdf

[33] Eclipse EMF project. http://www.eclispe.org/emf/

[34] MODELWARE project. http://www.modelware-ist.org/

[35] MOFScript project. http://umt-qvt.sourceforge.net/mofscript/

[36] OMG, RFP on MOF Model to Text Transformation. http://www.omg.org/cgi-bin/apps/doc?ad/05-11-03.pdf

[37] Eclipse MDDI project. http://www.eclipse.org/mddi/

[38] Markus Völter, Thomas Stahl, et. al., Model-Driven Software Development: Technology, Engineering, Management. John Wiley & Sons, 2006.

[39] Sami Beydeda, Mattias Book and Volker Gruhn (eds.) Model-Driven Software Development. Springer, 2005.

[40] Tony Clark, Andy Evans et.al. Applied MetaModeling : A Foundation for Language Driven Development. xactium.com 2005.

[41] Eclipse.org. Java Emitter Templates (JET) . available from: http://www.eclipse.org/emft/projects/jet/

[42] SmartQVT :An open source model transformation tool implementing the MOF 2.0 QVT-Operational language. available from: http://smartqvt.elibel.tm.fr/

[43] AndroMDA, Available from: http://www.andromda.org.

[44] Clemens Szyperski. Component Software, Addison-Wesley, 2002.

[45] Andy Ju An Wang and Kai Qian. Component-oriented Programming, John Wiley & Sons, Inc., 2005.

[46] 詹亞騰，以模型導向技術發展的一套高質語言編輯環境生成系統，政大資科系碩士論文，2008..

[47] Cheng-Chia Chen. EGOE: a Generating System of Multi-View Editing Environments on Eclipse Platform. Proceedings of the International Computer Symposium, Taipei, Taiwan, Volume I, 298-303, 2006.

[48] Ting-Huei Li and Cheng-Chia Chen. From Ontology to Semantic Web Service via Model-Driven System Development. Proceedings of National Computer Symposium (NCS'07), Volume 3, 858-867, Taichung, Taiwan, 2007. (best paper award)

[49] Colin Atkinson and Thomas Kühne. A Generalized Notion of Platforms for Model-Driven Development. In Model-Driven Software Development. Springer, p119-139, 2005. Springer.

[50] K. Czarnecki and S. Helsen. Classification of Model Transformation Approaches. In online proceedings of the 2nd OOPSLA'03 Workshop on Generative Techniques in the Context of MDA. Anaheim, October, 2003.

[51] Graph Transformation and Graph Grammar. http://www.gratra.org.

[52] Ting-wei Lin，Enhance Eclipse Platform with Remote Plug-ins，Master thesis, Department of Computer Science, NCCU, 2009.

計畫成果自評:

本計劃申請時的主要目標有二。一為從現有 Eclipse 平台上的各種框架與 API，取最常被開發者引用的部分，提供更高層次的模型描述，稱之為超模型。有了超模型之後，開發者即可利用這些超模型，在 EMF 編輯器之下，編輯產生超模型實例，也即是開發者所要的所謂外掛程式模型。其次則是規畫建構相關的模型轉換與程式碼產生器，以供模型開發者產生對應的外掛程式骨架與部分實做。

在超模型的設計上，我們主要是針對以下幾項最常使用的框架/API 規畫出對應之超模型。這些重要項目包括:1.與 Eclipse 功能觸發有關之超模型。2.視圖(View)與視景(perspective)超模型，3.編輯器(Editor)超模型，4.插件(Plug-in)超模型 5.導引精靈與對話盒(Wizard and Dialog)超模型，6.喜好設定超模型(Preference and dialog settings)。這些部份的完成，基本上符合我們當初的規劃建議。

在 PIM2PSM 的模型轉換部分，我們利用 Eclipse Modeling 計畫群下的 ATL 為轉換語言，參考 EMF 由 Ecore 至 GenModel 的做法，實做了必須的模型轉換。在 PSM2Text 的部分，由於外掛程式每一功能單元具有 Code Scattering 與 file sharing 的特性，我們必須同時產生不同輸出，並且必須將之織入至專案裡的不同檔案中。綜合上述總結，我們相信已完成當成提出的計畫目標。至於論文發表方面，此部分成果的書寫仍在進行之中，而在人才培養方面，則造就了兩位碩士[46,52]。