行政院國家科學委員會專題研究計畫 成果報告

現代化知識密集式編輯環境的生成系統之研究與設計
研究成果報告(精簡版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 99-2221-E-004-004-
執 行 期 間 ：99 年 08 月 01 日至 100 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

計畫參與人員：碩士班研究生-兼任助理人員：趙仁鋒
碩士班研究生-兼任助理人員：郭建宏
博士班研究生-兼任助理人員：詹亞騰
博士班研究生-兼任助理人員：高振益

公 開 資 訊 ：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 100 年 11 月 05 日

中文摘要：　一套高品質的編輯環境可以提供給標的語言各式各樣的輔助，讓使用者得以縮短編輯過程，並能有效降低錯誤。在以往，此種編輯環境並不易開發，但近年來隨著可擴充式的工具整合平台如 Eclipse、Netbeans 的出現，開發者可以重複利用這些平台提供的編輯功能框架與預先實做，因此不再需要從底層打造編輯環境。即使如此，實際經驗顯示，由於此類框架複雜難懂，要一般開發者利用這些平台實做高品質編輯環境，依然是困難重重。基於此種因素，最近幾年來，相繼有一些開放式編輯環境生成系統問世。這些系統可以使開發者避開工具整合平台的框架細節，而以更高階的標的語言定義為輸入，系統則利用模型驅動與程式碼生成等技術，自動生成可以在所屬平台執行的語言編輯環境。美中不足的是，這些系統所生成的編輯環境儘管有些已經非常優秀，其整體功能尚遜色於目前最先進的編輯環境。

本計畫的長遠目標，就是希望能增強既有生成系統，使其產出由基本編輯環境提升為高品質的整合式語言開發環境。在此環境中，不僅具有高品質的編輯功能，更具有整合式的程式/文件執行、測試、偵錯、編譯與佈建等功能。在本年度的具體目標上，我們集中在如何實現知識密集功能輔助，讓使用者在編輯時僅需出示簡單動作，系統即可從背後的龐大資訊來源中，找出與後續動作相關的輔助訊息與建議，從而幫助使用者快速輸入正確內容。另外，為了使產出的環境具有執行與偵錯能力，我們擴增了既有系統的語言規格定義範疇，使開發者得以加入標的語言之執行與偵錯語意描述，從而使得系統得以產生可在平台上執行的直譯器與偵錯程式。最後為使系統得以快速產生導引精靈，幫助開發者藉導引精靈完成資訊輸入，我們完成了一個多平台導引精靈程式開發系統，使用者只需輸入與平台無關的精靈描述，系統即能產生包含 Eclipse 平台在內的三種精靈程式實做。

英文摘要：　A high-quality editing environment is able to provide versatile editing aids for the users of the target language so that the whole editing process is shortened significantly and editing errors reduced effectively. In the past, such kind of systems is hard to onstruct. As the release of open and extensible tool integration platforms, such as Eclipse and NetBeans, the situation is changing due to the availability of frameworks and built-ins provided in these platforms. Even so, practical experience manifests that it is still rather difficult for an

ordinary developer to use these frameworks. To overcome such difficulty, there have been several generating systems released in recent years aiming to construct an editing environment on a platform without direct coding on the associated framework. Instead, given the definition of a target language, such systems can generate an editing environment for the target language on the associated platform. Compared with direct coding, this approach enable the developer to to generate an editing environment more easily. Unfortunately, the generated systems, even though some of which are excellent, are still inferior to most high-quality editors used for popular programming languages.

The long-term objective of this project is to enable the generation of high-quality language development environments by augmenting selected generating systems with additional capability to generate other longing features for the generated system. In such generated systems we will have not only high-quality editing utilities as found merely in top-level editors for most popular programming languages but also functionality for executing, testing, debugging, compiling and deploying edited programs/documents. Completed during this year are several tasks including the exploration of how to add to the selected generated systems knowledge intensive editing aids so as to enable users to get helpful messages and suggestions from tremendous information sources. Moreover, in order to have testing and debugging capability in the generated system, we enhance the language definition part of the selected generating system to allow the description of execution and/or debugging semantics for target languages so that the system can generate as well a corresponding executor and/or debugger. Finally, as a quick way to generate wizard utility for the generated systems, we develop also a wizard generating system by following OMG ‘s MDA approach.

一、前言

現今軟體系統的開發過程均極度依賴良善的軟體工具輔助，主因為這些工具能為開發者大大提升生產力。其中尤其以整合式開發環境(IDE)，能為程式設計師在軟體發展上提供各種輔助；可有效加速程式開發、簡化系統維護與管理等複雜的工作。在程式碼的編輯上，現今的 IDE 提供了各式各樣，多采多姿的強大輔助功能。除了見諸傳統編輯器的基本功能，現今 IDE 的最大特色之一便是能為開發者提供各式各樣的有效即時幫助。例如 IDE 可針對編輯中的程式進行即時自動編譯，因此一旦程式有語法或語意錯誤時，IDE 可及時測知，並能經由多重親善畫面，告知開發者錯誤的原因，以及各項修補建議，供開發者快速修正。其次，現今編輯器通常兼具標的語言的語法知識，因此能提供程式碼輔助/補全(Code assistant/completion)、與程式段落範本(Templates)等功能，可有效的幫助開發者快速修正或事先預防可能的輸入錯誤。

除此外，現今 IDE 編輯軟體如 Eclipse[10]的 JDT[11]的最大特色就是所謂的知識密集式編輯輔助。通常此種編輯器可以連結並善用程式設計所需的背後巨量程式庫及其相關說明文件等。其具體應用就是，可以透過不同的畫面(views)、隨手提示(HandOver)以及輸入建議，讓開發者可以快速找到相關之各種類別(class)、欄位(field)與函數(methods)，可使用戶即時了解其用法並助其快速輸入。我們認為這是一種最值得喝采的功能，因為現今的程式開發，通常必須依賴他人事先提供的巨量程式庫，而這些資料如 JDK 通常極度龐大複雜，以致即使是領域專家也難完全了解並熟記每一細節。因此，若無知識密集式編輯環境的幫助，程式開發所需的程式庫功能與用法查詢，將大大降低開發者的生產力。

因此當發展一個程式或領域語言時，使用者最期盼的當然就是一個針對該語言，具有所有高功能編輯特徵，而能為開發者大大提升效率的現代化編輯環境。遺憾的是一個全功能的現代化知識密集式編輯環境至今依然不易建造。現行系統中，具有類似特徵的整合開發環境，多半只針對具有廣大用戶群的一般用途語言如 C++，Java,等提供此類功能。只有相對少數環境如 ANTLR Studio[28]、XMLSpy[27]等則針對單一特定領域語言如 ANTLR[24]，XML 等提供現代化編輯與輔助功能。至於通用文字編輯器如 UltraEdit、EditPlus 等，雖能以純文字方式編輯多數語言，但提供的輔助卻相當有限。

本計畫的目的就是希望能針對編輯環境不易開發的現狀，提出一些可加快編輯環境開發的生成工具與方法。我們希望藉由增強現有生成工具，使得開發者不但可快速產生編輯環境，而且所生成的編輯環境提供給標的語言的是一種具備知識密集輔助特性，以及可提供多樣編輯輔助的高品質編輯環境，因而能幫助使用者快速決定內容、快速產生輸入，並能預防及降低錯誤，進而能有效地縮短編輯過程。

如前所述，在以往此種編輯環境並不易開發，因此大多數特定領域語言(DSL)並未見有高功能的專屬編輯器。然而此種狀況正在改變之中，因為隨著開放、可擴充式的工具整合平台如 Eclipse、Netbeans[25]的出現，開發者可以利用這些平台提供的人機互動與編輯功能框架與預先實做，因此不再需要重頭打造編輯環境。即使如此，實際的使用經驗告訴我們，由於此類框架的複雜難學，要一般的開發者利用這些框架實做高品質編輯環境，依然是困難重重。

基於此種因素，我們早在 2006 年即提出一套稱做 EGOE[21,30]的編輯環境生成系統，可以使開發者避開框架細節，而以更高階抽象的標的語言定義為輸入，系統則利用模型驅動的技術，自動生成出可以在 Eclipse 平台上執行的高品質多頁面編輯環境。EGOE 所產生的編輯環境能同時為標的語言提供結構化與純文字等不同的編輯頁面，並且於文字編輯時提供醒目提示、內容輔助與文字喜好設定等編輯相關的輔助功能。隨後的 EGOE II[31]除

改善原有架構之外，更大大的增加原有的功能，提供的功能有：程式碼摺疊、美化、錯誤標記、範本精靈，程式碼自動補全以及變數宣告與引用之自動連結等。

本計畫是延續我們這幾年來，在相關議題已經擁有的經驗，設法增強既有工具的能力，使其生成之編輯器具有高程度的知識密集編輯功能輔助。我們希望使用者在編輯時僅示簡單動作，系統即可從使用者編輯所需的龐大相關資訊來源中，找出精確、即時，以及與後續動作相關的輔助訊息與建議，從而幫助使用者快速輸入正確內容。除此外，我們也為現有工具擴增其表達與生成範疇，使其不僅可以表達一般編輯器所需之標的語言的詞彙與語法資訊，還可進一步的讓客戶提供標的語言的執行與偵錯語意，我們的工具可據此為標的語言產生整合式的直譯與偵錯程式，如此則可方便客戶對編輯文件或程式，進行測試與偵錯。最後為加快導引精靈程式之生成，我們也為自己開發建立了一個多平台導引精靈程式開發系統。用戶只需依照系統要求，提供自己所需之導引精靈的高階描述，系統即能為用戶產生在三個不同平台的導引精靈實做。

二、研究目的

我們希望能針對語言編輯環境不易開發的現狀，提出一些可加快語言編輯環境開發的工具與方法。執行方式則是希望能增強既有生成系統，使其產出由基本編輯環境提升為高品質的整合式語言開發環境。在此環境中，不僅具有高品質的編輯功能，更具有整合式的程式/文件執行、測試、偵錯、編譯與佈建等功能。在具體目標上，我們集中在如何實現知識密集式功能輔助，讓使用者在編輯時僅需出示簡單動作，系統即可從背後的龐大資訊來源中，找出與後續動作相關的輔助訊息與建議，從而幫助使用者快速輸入正確內容。另外，為了使產出的環境具有執行與偵錯能力，我們計畫擴增既有系統的語言規格定義範疇，使開發者得以加入標的語言之執行與偵錯語意描述，從而使得系統得以產生可在平台上執行的直譯器與偵錯程式。

三、文獻探討

本計畫的目的是希望使開發者能快速為各式領域專屬語言建構整合於工具平台的高品質的語言編輯與執行環境。我們採用的工具平台是 Eclipse;我們達成快速建構語言編輯與執行環境的辦法是採用模型驅動的方式，由開發者輸入極高階的標的語言定義，再經由模型轉換與程式碼合成等過程，自動合成標的語言的編輯與執行環境。

早期的多功能編輯器實做通常是直接呼叫作業或視窗視窗系統下的 API。這方面的基本做法可參見 Jame Brown[34]與 Nikolai Weibull[33]。由於這些傳統實做實在過於低階，也缺乏適當層次的框架與預先實做，因而需要耗費龐大的的時間與人力，實做軟體也難以重複使用。其後逐漸的有支援編輯器實做的 GUI 組件問世，例如 Java swing 的 javax.swing.text、.NET 也提供功能更強的 TextBox 或 RichTextBox 均可讓開發者得以較以往便利的產生編輯器，然而若目標是更高功能編輯環境，則上述框架仍然顯得簡陋甚或全無支援。

好在自 2001 年後，隨著可擴充式的整合式軟體工具開發平台如 Eclipse[09,10] 09,10] 與 NetBeans[25]的發佈而讓編輯環境開發者有了更好的選擇。這些平台不但提供給開發者功能更強的 GUI 介面以及編輯軟體框架支援，使開發者可以更快速建造高功能的編輯環境，產出的編輯程式還可以充分的和平台其它工具完全連結整合，而不像以往只是一個孤立的編輯器。而本計畫的所有軟體也均建構在 ECLIPSE 平台之上。

Eclipse平台不但是一個整合式軟體開發環境(IDE)，更是一個開放式的軟體工具整合平台，其最大特徵就是其無限的可擴充性。Eclipse的外掛程式(plug-ins)軟體架構[35,10]

是一種微核心的結構:除了極少數的核心程式之外，所有其他程式都必須包裝成所謂的Bundle或plug-ins，方能安裝到平台之中。而一旦程式安裝之後，大家的角色都是平等的，彼此之間可以互相叫用。當然的也就可以使用Eclipse 平台預建的許多Plug-ins，如各種GUI widgets 與工作台(Workbench)以及功能如同增強版的File manager 的工作區(Workspace)資源(Resource)管理系統。而最後的效果就是所有工具程式均可在Eclipse 環境中使用相同的介面和操作方式執行。有關ECLISPE平台以及平台插件開發的參考資料實在汗牛充棟[05,35,36]，其中最重要的當屬其完整的線上參考文件以及應用程式介面參考說明。

OMG在2002年所提出模型驅動架構(MDA)[08]，主張以更高階、更抽象、更貼近開發者的模型取代程式碼，成為系統開發的主體，有如現今的高階語言早期取代組合語言或機器語言一般。此外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。此種模型因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)。MDA主張開發系統實，應採用分離關注的原則，分別發展與標的平台無關的系統模型(稱作PIM; Platform Independent Model)，接著發展所謂的模型轉換器(PIM2PSM mapping)，以便將PIM轉換成特定平台的PSM，最後再經由模型轉程式(M2T)的轉換而產生最後實做。有關這方面的資訊可參覽OMG之MDA網站[08]或維基網站，參考文獻可見[01,02,03,04, 07,19,20]。

連結 Eclipse 與模型驅動架構的最有用工具當屬 EMF(Eclipse Modeling Framework)[06,13,26]。EMF 是一套支援模型驅動系統設計的框架系統。其功能是讓用戶可以輸入一個符合其 Ecore 超模型的資料模型，EMF 即能利用為該資料模型產生功能強大的模型實做、對應的測試程式以及架構於 EClpse 平台的結構化編輯程式。因此如果利用 EMF 的 Ecore 格式來定義程式語言的抽像語法，我們即可快速獲得標的語言的抽想語法 API 以及結構化編輯器。有關 EMF2 的應用與介紹，除 EMF 網站以及線上文件之外，最重要得當 EMF 創建者所著之 EMF 著作[06,26]。

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。而在自動產生編輯程式的研究，相關的系統至少有: Eclipse IMP[16,29]、SmartTools[32]、DLTK[14]、xText[17]、Meta-Environment[22]，MetaEdit+[23]，Rascal[37,38]等。而自 2011 年起，每年也都有一次的國際性相關系統競賽舉辦[44]。在這些系統中，屬於開放系統且生成的編輯器具有最多最強的功能的當屬 Xtext，因此我們也以之為擴增之對象。Xtext 是 Eclipse Modeling 計畫群中的一個子計畫，其前身是生 OAW[15]，目前則是屬於 TMF 子計畫群。Xtext 事實上包含許多工具，除了可以讓開發者產生飄的語言的編輯環境之外，更提供後續標的語言的許多應用支援。其中包含可做模型轉換(M2M)與模型展程式碼(M2T)的 Xtend，提供標的語言執行能力支援的 Xbase 以及提供可快速組合物件、產生流程定義(Workflow Definition)並具以執行的流程執行引擎 MWE2。有關 XText 與相關工具的介紹，主要參見[17]。除此外，XText 使用了相依注入(Dependency INjection)的方式，合成編輯器實做，為此他採用了 Google Guice 在這方面的 API[39]。

為了使產生的編輯環境具有程式執行與偵錯能力，我們需要為標的語言定義有關執行與偵錯的語意描述，其後方能以模型驅動方式產生對應的直譯器與偵錯器。有關語言的語意描述方式，文獻有許多的探討[40,41]。因為我們需依此產生 Java 實做，因此必須是具有可執行性且易於 Java 實做。我們目前採用的以自然語意[41]為基礎的描述方式，且參考了 RML[42,43]的做法，為語意描述提供具體與抽象語法，最即可經由程式碼生成產生

直譯器與偵錯器。我們產生的直譯與偵錯程式，是架構在Eclipse的啟動與偵錯框架之上，因此可以充分的使用 Eclipse 的啟動與框架的各種好處。有關 Eclipse 的啟動與框架相關資訊，除了參考 Eclipse 線上文件之外，最有效的介紹是來自 Eclipse Conference 的 tutorials.

四、研究方法

本計畫所要研究的是如何使開發者能快速為各式領域專屬語言建構整合於 Eclipse 工具平台的高品質語言編輯與執行環境。我們達成快速建構語言編輯與執行環境的辦法是避免直接在 Eclipse 編輯與偵錯框架上，進行編程；相對的我們採用模型驅動的方式，由開發者輸入與實做少有關連的極高階標的語言定義(其中包含具體語法，抽像語法，執行與偵錯語意描述，以及其他與各式編輯功能相關的的敘述等)，再經由模型轉換與程式碼合成等過程，自動合成標的語言的編輯與執行環境。

在具體實現上，我們原本在計畫書裡是規劃繼續增強我們既有的自建工具 EGOE II[21,30]，使其產生的系統具有更多高功能的編輯與執行能力。然而由於此領域的高度競爭與迅速成長[44]，我們發現原先規劃的許多待達成功能，事實上已經可以利用這些工具例如 Xtext[17]的最新版本產生。為了快速達成目標，避免無謂重複，因此我們決定先擱置 EGOE II 的功能增強，相反的，我們改以 Xtext 及其相關工具群為基礎，建造可以和這些工具整合的補強組件，透過 Xtext 和我們的補件，可以使得產生的系統不僅具有規畫中的其他高功能編輯能力,更也可以具備目前 Xtext 尚不能產生的語言執行與偵錯能力。

Xtext 是 Eclipse Modeling 計畫群中的一個子計畫，其前身是生 OAW，目前則是屬於 TMF(Textual Modeling framework)子計畫群。使用 Xtext 時，我們需先在 Eclipse 上產生一個 Xtext 專案，接著只需開檔輸入標的語言的具體語法，並說明如何連結或同時產生對應的抽像語法的文法描述，然後啟動系統事先為專案產生的工作流程檔，即能由此自動為開發者產生極高功能的標的語言編輯環境實做。此實做會在 Eclipse 專案區裡產生三個 plug-in 專案，這些 plug-ins 一旦安裝佈建，即是可在 Eclipse 平台上操作的高功能的標的語言編輯環境。

Xtext 產生的編輯程式除了傳統編輯器的基本功能之外，其他可實現的功能還包括:語法醒目提示、個人喜好設定、超連結、美化器(Formatter)、內容即時輸入輔助(包含詞彙補全、範本提示、以及內外部可用功能建議等)、不同層次的錯誤或警告訊息之顯示與更正建議、文件大綱顯示、跨語言與跨檔案編輯時期程式或文件內容連結，以及各式程式靜態檢驗等。除此之外，Xtext 還能結合 EMF 為抽象語法產生對等之 Ecore 模型，再經由 EMF 工具連結即可產生模型 API 可供程式師叫用，另外，Xtext 也提供 EMF Resouce 的實做 XTextResource，允許將符合具體語法的文件載入為模型物件或將符合抽像語法的模型物件序列化為具體語法格式的資料而永續存放。

一套實用的編輯環境，必須實做許許多多的編輯功能，這些功能如何整合一起且不衝突，是軟體架構設計的一大考驗。 Xtext 在此部分是使用 Goole Guice 提供的 API，採用所謂的分離關注(Separation of Concerns)與相依注入(Dependency Injection) 技術。要言之，其做法是將功能服務物件的獲取與功能服務物件的叫用分離，讓需要叫用功能服務之程式，只負責宣告其所需之功能服務種類，即可假設這些物件已經備妥可供叫用，但卻並不負責產生或抓取任何實做物件。而真正的物件生成與連結動作是當系統啟動之時，會先透過組態規劃，設定每一功能服務之實做，接著以此產生注入器。有了注入器之後，所有新產生的物件，使用前均需經注入器的檢視而將相依功能注入，因而每一物件程式實行任務前均可得到所需之服務。在此架構下，我們實做其他編輯功能服務的方式，即變得

相對簡單。其概略如下:1.確認功能服務的客戶與啟動時機，2. 在功能客戶內增加叫用該服務之程式碼，使其得以在適當時機叫用服務程式，3.實做功能服務程式，其中包含註記所有需要相依注入的其他功能服務,4.修改系統起動設定，登錄實做的功能服務，以便後續注入器用以將該服務注入至需要的客戶物件之中。

為了加快導引精靈程式之開發，我們因而建立了一個多平台導引精靈程式開發系統。此系統是依照 OMG 之模型驅動架構(MDA)而設計的。根據 MDA 理論[01,02,08,19,20]，我們需為精靈程式的開發，定義了一個與平台無關的精靈程式超模型(WMM)，以及為三個實做平台:Eclipse 平台、Java Widard API 平台，以及 Web 平台定義了三個平台相關的精靈程式超模型。接下來，我們必需實做模型轉換程(M2M)式，以便將使用者提供，符合 WMM 規範的精靈模型轉換為符合個別平台要求的的精靈模型，最後再經所謂的模型轉程式碼(M2T)工具，系統即能產生三個平台的精靈程式實做。最後，為了方便使用者提供符合 WMM 規範的精靈模型，又能藉此熟悉並善用 xText 快速生成領域專屬語言環境的強大功能，我們必須為 WMM 提供一個對應的精靈描述領域專屬語言 WDL。利用此領域專屬語言，以及我們利用 xText 產生的 WDL 編輯環境，使用者即可在高品質的編輯環境下，產生自己的精靈程式模型，並可輕易的利用整合的模型轉換，以及模型轉程式實作等系統內建的功能鏈，迅速產生各種平台實做。

為了彌補 Xtext 現今無法產生執行與偵錯程式的不足，我們也希望讓開發人員可以額外的為標的語言定義有關執行與偵錯的語意描述，之後系統即能依照這些規則，產生對應的直譯器與偵錯器。有關語言的語意描述方式有許多機制，而我們所要求的是必須具有可執行性且易於 Java 實做。我們目前採用的以自然語意為基礎的描述方式，我們參考了 RML 的做法，為這些描述提供具體與抽象語法，最後並經由程式碼生成的過程，產生直譯器與偵錯器。我們產生的直譯與偵錯程式，因為是架構在 Eclipse 的啟動與偵錯框架[]之上，因此可以充分的使用 Eclipse 的啟動與框架的各種好處,其中包括有一個友善的 GUI 介面，讓開發者彈指之間即可啟動測試或偵錯，而進行偵錯時，也可利用 Eclips 的整合偵錯介面來逐步追蹤程式的運作狀況。

五、結果與討論

本計劃完成了數項任務,以下分別說明。首先我們學習了 Xtext 生成系統,瞭解利用 Xtext 生成的語言環境所具有的各種功能，並研究如何擴充 xText,俾能利用擴充後的 xText 產生標的語言環境，使其具有原先 xText 尚所不能生成的功能。

為了加快導引精靈程式之開發，我們建立了一個利用模型驅動架構(MDA)為理論基礎的多平台導引精靈程式開發系統。在此系統中，我們為精靈程式的開發，定義了一個與平台無關的精靈程式超模型(WMM)，以及為三個實做平台:Eclipse 平台、Java Widard API 平台，以及 Web 平台定義了三個平台相關的精靈程式超模型。使用者只要提供符合 WMM 規範的精靈模型，系統即能由此產生三個平台的精靈程式實做。為了方便使用者提供符合 WMM 規範的精靈模型，又能藉此熟悉並善用 xText 快速生成領域專屬語言環境的強大功能，因此我們也為 WMM 提供一個對應的精靈描述語言 WDL。利用此領域專屬語言，以及我們利用 xText 產生的 WDL 編輯環境，使用者可以在高品質的編輯環境下，產生自己的精靈程式模型，並可輕易的利用整合的模型轉換，以及模型轉程式實作等系統內建的功能鏈，迅速產生各種平台實做。

Eclipse 平台的 Java 編輯工具 JDT 功能非常強大，其中有兩種伴隨視圖(View)與知識密集編輯輔助極有相關,但 Xtext 目前尚無法產生,它們分別是 SourceView 與 JavaDocView。當用戶編輯 JAVA 程式時，只要點選到任何 JAVA 元素，例如類別、方法、欄位或建構式時，

兩個視圖即能收到通知，而其任務則是即時的找出對應的程式碼與說明文件，並將之呈現於各自的視圖區內。這兩種視圖使客戶得以不用離開編輯對像，即能瀏覽周遭程式碼的定義與說明，因而能增進程式編輯效率。我們在此部分的任務，就是使這兩種視圖也能出現在 xText 產生的語言環境中。SourceView 的實做較為直接，只需將自己登錄為 SelectionListener,收到編輯器傳來的選擇改變事件之後,找出對應的內或外部程式片段，再將之顯示於自己的視圖區即可。DocView 的實做，則還牽涉到如何定義標的語言元素的說明區塊，以及如何由語言元素抓取對應的說明區塊的方法。我們目前的做法是仿效 JAVA 的方式，以特別的註解標示說明區塊，而 DocView 則從標的元素的前面抓取最近的說明區塊，做為該元素的說明。未來我們還計畫增強 Xtext 功能，使其具有類似 JDK 的 javadoc 功能，能將標的語言文件內嵌的說明區塊資訊萃取出來，成為獨立之說明文件。屆時我們的 DocView 將也能從這些說明文件來源及時找出對應的說明資料。

Xtext 產生的系統只能稱之為編輯環境而非語言環境，因為目前產生的系統並無法對編輯的文件或程式進行執行與偵錯。主要的原因是目前的 Xtext 尚能讓開發者提供標的語言的執行或偵錯語意描述，因此也無法生成所需的直譯器與偵錯器。為了彌補 Xtext 現今的不足，我們因此提供了這一方面的補件。在此補件架構之下，開發人員可以額外的為標的語言定義執行與偵錯語意描述，系統即能依照這些規則，產生標的語言對應的直譯器與偵錯器。此系統產生的執行與偵錯環境具有以下三項特性: 1.採用自然語義來描述標的語言之執行與偵錯語義而非直接實做，因而能夠讓使用者專注在更抽象的層面上來描述一個語言如何在機器上運作，而不用去顧慮其他實作的細節。而相較於 small-step 語義而言，自然語義隱藏了更多執行細節，使用者能夠比較容易完成語言的語義定義。2.抽象化偵錯定義：除了執行語義以外，本機制也將偵錯定義抽象化，使用者不用一一實作每個偵錯元件或是功能，只要宣告語言在執行時的程序結構、變數與暫停點，即可自動生成基本的偵錯功能與變數顯示。3.提供人性化偵錯環境：本機制生成的直譯器與偵錯器均整合至 Eclipse 平台上,提供一個圖形化且友善的介面輔助程式開發者撰寫程式。而進行偵錯時，也可利用 Eclipse 的偵錯介面來逐步追蹤程式的運作狀況。

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

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

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

[06] Dave Steinberg, Frank Budinsky, et al. Eclipse Modeling Framework, 2nd Edition, Addison Wesley, 2009.

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

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

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

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

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

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

[13] Eclipse Modeling Framework Project (EMF). http://www.eclipse.org/modeling/emf/

[14] Eclipse Dynamic Languages Toolkit project. http://www.eclipse.org/dltk/

[15] Eclipse openArchitectureWare project. http: //www. eclipse. org/gmt/ oaw/

[16] Eclipse IDE Meta-tooling Platform (IMP) project. http://eclipse-imp.sourceforge.net/imp.html

[17] Eclipse xText project. http://www.xtext.org/

[18] XML Textual Modeling Framework. http://www.eclipse.org/modeling/tmf

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

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

[21] 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.

[22] Brand, M.G.J. van den, A. van Deursen, et.al., Industrial Applications of Asf+Sdf. Algebraic Methodology and Software Technology , 9-18, 1996, Springer-Verlag.

[23] MetaEdit+ Workbench, http://www.metacase.com/mwb/.

[24] ANTLR Parser Generator. http://www.antlr.org/

[25] NetBeans. http://www.netbeans.org/

[26] Budinsky, David Steinberg, Ed Merks, et.al. Eclipse Modeling Framework: A Developer's Guide, 2003, Addison Wesley.

[27] Altova, XMLSpy XML Editor. http://www.altova.com/products/xmlspy/xml_editor.html

[28] Antlr studio for Eclipse. http://www.placidsystems.com/antlrstudio.aspx

[29] Philippe Charles, et.al.. SAFARI: A Meta-Tooling Platform for Creating Language-Specific IDEs. Companion to the 21st ACM SIGPLAN conference on OOPSLA, 2006.

[30] 黃立昇，在 Eclipse 平台上發展的一套編輯環境生成系統，政大資科系碩士論文，2006.

[31] 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.

[32] SmartTools Software Factories, http://www-sop.inria.fr/smartool/

[33] Nikolai Weibull, ned- Text Editor of the Future. http://ned.rubyforge.org/. 2003.

[34] James Brown, Design and Implementation of a Win32 Text Editor. http://catch22.net/tuts/neatpad, 2008.

[35] Jeff McAffer, P. VanderLei and S. Archer, OSGi and Equinox, Creating Highly Modulare Java Systems, 2010, Addison Wesley.

[36] Eric Clayberg and Dan Rubel, Eclipse Plug-ins, 3rd edition, 2009, Addison Wesley.

[37] Paul Klint, et.al., RASCAL: A Domain Specific Language for Source Code Analysis and Manipulation. In Proceedings of SCAM'09, vol. 0, pp. 168–177., 2009

[38] Paul Klint, Tijs van der Storm, and Jurgen Vinju. EASY Meta-Programming with RASCAL. In Proceedings of GTTSE'09, pp. 185–238, 2009.

[39] Google Guice : , http://code.google.com/p/google-guice/.

[40] G. Rosu and T. F. Serb˘anut˘a. An overview of the K semantic framework. Journal of Logic and Algebraic Programming, 79(6):397–434, 2010.

[41] Hanne Riis Nielson and Flemming Nielson. Semantics with Applications: An Appetizer. Springer-Verlag, 2007.

[42] Peter Fritzson: Efficient Language Implementation by Natural Semantics, Book Draft, 2010.

[43] Mikael Pettersson: Compiling Natural Semantics. Lecture Notes in Computer Science, 1549. 1999, Springer-Verlag.

[44] Language Workbench Competition 2011, http://www.languageworkbenches.net/.

國科會補助計畫衍生研發成果推廣資料表

日期:2011/10/21

國科會補助計畫	計畫名稱: 現代化知識密集式編輯環境的生成系統之研究與設計
	計畫主持人: 陳正佳
	計畫編號: 99-2221-E-004-004- 學門領域: 程式語言與軟體工程

無研發成果推廣資料

99 年度專題研究計畫研究成果彙整表

計畫主持人：陳正佳	計畫編號：99-2221-E-004-004-
計畫名稱：現代化知識密集式編輯環境的生成系統之研究與設計	

成果項目			量化			單位	備註（質化說明：如數個計畫共同成果、成果列為該期刊之封面故事...等）
			實際已達成數（被接受或已發表）	預期總達成數(含實際已達成數)	本計畫實際貢獻百分比		
國內	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	0	0	100%		
		專書	0	0	100%		
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（本國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		
國外	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	0	0	100%		
		專書	0	0	100%	章/本	
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（外國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		

其他成果 (無法以量化表達之成果如辦理學術活動、獲得獎項、重要國際合作、研究成果國際影響力及其他協助產業技術發展之具體效益事項等，請以文字敘述填列。)	無

	成果項目	量化	名稱或內容性質簡述
科教處計畫加填項目	測驗工具(含質性與量性)	0	
	課程/模組	0	
	電腦及網路系統或工具	0	
	教材	0	
	舉辦之活動/競賽	0	
	研討會/工作坊	0	
	電子報、網站	0	
	計畫成果推廣之參與（閱聽）人數	0	

國科會補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

■達成目標

□未達成目標（請說明，以 100 字為限）

□實驗失敗

□因故實驗中斷

□其他原因

說明：

2. 研究成果在學術期刊發表或申請專利等情形：

論文：□已發表 □未發表之文稿 ■撰寫中 □無

專利：□已獲得 □申請中 ■無

技轉：□已技轉 □洽談中 ■無

其他：(以 100 字為限)

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）(以 500 字為限)

本計畫所實做之導引精靈開發系統，乃是極具應用價值之軟體工具，因為利用此一系統一般客戶即能快速開發其所需之導引精靈，不但可以節省許多人力與時間，使用上也毋需具備背後之複雜平台程式設計基礎。為了配合碩士助理之畢業進度，我們的論文將待下年度發表。此外我們也計畫待更為完善成熟之後，將此一系統發佈為開放源碼系統供大家使用。

目前的編輯環境生成工具，大部分集中於編輯功能的增進，而少有提供執行或偵錯語意的描述，因而也無法產生對應的直譯與偵錯程式。因此我們在此部分的努力，讓開發者可以在語言定義中提供執行與偵錯語義的描述，系統再據以產生具執行與偵錯能力的語言編輯環境，無疑地對於有此種需求的開發者而言，是具有極大吸引力。而一旦這種生成方式成熟，將再也不會有人願意從頭打造直譯與偵錯程式。在未來規劃上，我們計畫提升我們的工具，使得語意表達範圍、抽象層次與完整性都能更一步增進，而方便開發者快速生成語言環境。