用於圖形化編輯器開發之專屬模型語言設計

A Modeling Language of Domain Specific Editors Based on Graphical Modeling Framework

呂宗龍
國立政治大學 資訊科學系
e-mail：g9223@cs.nccu.edu.tw

陳正佳
國立政治大學 資訊科學系
e-mail：chencc@cs.nccu.edu.tw

摘要

圖形化模型編輯器是模型編輯器開發長久以來一直努力的方向。Eclipse.org 已提供兩個功能強大的圖形化編輯器設計框架：GEF 圖形化編輯器設計框架，和將 EMF 與 GEF 結合並簡化其設計流程的 GMF 圖形化模型編輯器設計框架，來協助開發者進行圖形化模型編輯器開發。本篇研究主要是在探討 GMF 圖形化模型編輯器開發流程的簡化。在研究中希望藉由提供 GMF 圖形化模型編輯器設計框架的 GM3 專屬模型語言，讓開發者能夠以更為直觀的角度來開發圖形化模型編輯器的架構，藉以縮短 GMF 的開發流程。

關鍵詞：Eclipse、MDE、GMF、Graphical Editor、DSL

Abstract

The availability of a visual graphical editor for a target domain is the prerequisite of visual graphical modeling, which has been adopted by classical software development for decades and is especially emphasized in today's model-driven engineering. However, compared with traditional textual editors, developing a visual graphical editor from scratch is not an easy work. As a result, there were frameworks developed such as GEF and GMF aimed to simplify the construction of graphical editors. Even so, however, it is still though hard for an average programmer to construct a visual graphical editor by using these frameworks without a long time of learning. Our result is a modeling language of graphical editors called GM3, serving as a bridge between developers of graphical editors and the GMF. With GM3, We also proposed an approach to generate graphical model from domain model automatically. After the GM3 specification of an editor is produced, the GM3 transformation engine developed by us can be used to generate all files required of the GMF. And, finally, a subsequent application of the GMF can produce a complete graphical editor on Eclipse platform.

Keyword：Eclipse、MDE、GMF、Graphical Editor、DSL

1. 導論

MDA [4] [21] 是 OMG[24] 所提出來的新一代程式開發與系統整合的架構，藉由提升了系統開發的抽象層次，使開發人員能夠專注於發展業務邏輯。MDA 將模型定位為系統開發的主要核心，使用 UML[27] 、MOF [22] 等模型描述語言所描述與平台無關的模型（PIM：Platform Independent Model）來設計系統，將系統的模型設計和平台實作的程式碼分離開來，讓開發人員可以在更高的模型層次上發展系統，而無需擔心底層的實作細節。

在軟體模型的設計中，Eclipse[1] [3] [12] [13] 上的 EMF[1] (Eclipse Modeling Framework) 提供了許多方便的功能來協助開發者建立並維護程式模型，同時可產生相對應的程式碼和結構化編輯器來呈現模型實例 (model instance)，並對模型實例進行操作與儲存。但面對程式模型的龐大化，一般的文字或

結構化編輯器已無法幫助開發者理解與呈現程式模型的架構，因此便出現了許多圖形化編輯器的設計框架來幫助開發者建立模型的圖形化編輯器。

在眾多的圖形化編輯器設計框架中，Eclipse 平台上所發展的 GEF[2] [10] [17] 及 GMF[8] [9] [18] 是屬於功能性非常強大的工具。GEF 可提供基礎的圖形化編輯器設計框架，但由於 GEF 的設計過於低階，其生產流程過於繁雜且不易使用，因此便催生出 GMF 專案。GMF 在 GEF 中導入 EMF 的模型設計與儲存能力、改善 GEF 的設計流程並強化其基礎框架的功能。

雖然圖形編輯器設計框架已經愈來愈容易開發，但還是未能達到普遍接受的程度，一個好的工具如果不能夠做到容易理解與使用是很難被開發者所接受的，而在後續的維護上也會相當困難。

本研究便希望在 GMF 這樣一個良好的設計框架上，可以有一個更高層次的設計視野，以使用者的角度來設計一個專屬語言[1] [11]，並隱藏框架複雜的實作細節，使其在設計及維護上，都能夠有快速且易用的功能。因此我們設計一個專屬模型語言 GM3 來整合 GMF 設計裡的各個功能描述，並在語言工具中提供一個領域模型圖形化的自動產生器，以有效降低圖形化模型編輯器開發的困難度。

2. 圖形化編輯器的構成要素

在 GMF 的設計框架下，圖形化模型編輯器呈現給使用者的介面是一個空白的編輯區和擁有整排工具及模型項目的功能面板。圖形化模型編輯器的組成包含工具面板和模型編輯，工具面板提供的是可供操作的模型項目，編輯區上能夠繪製什麼樣的模型端視工具面板上有哪些模型可以提供。在模型編輯部分又可分為領域模型和圖形模型。領域模型代表著模型的組成架構和模型與模型間的關聯特性，而圖形模型代表著模型的外觀表現。由此觀之，領域模型、圖形模型與工具面板模型就是圖形化模型編輯器組成所必備的要件。

領域模型可對應至四種圖形結構，分別是節點、連線、標籤以及能夠包含這三種概念的容器區間。

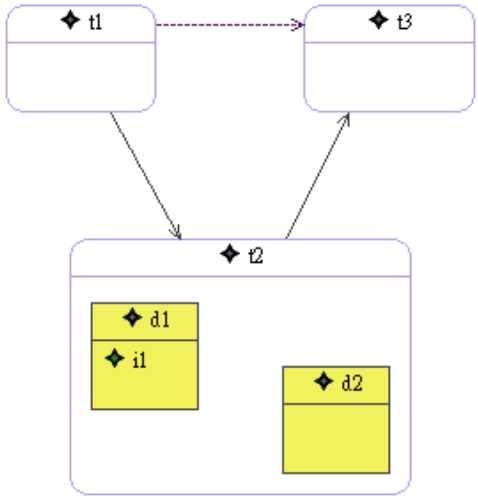

圖 1 圖形結構範例

其圖形結構如圖 1 所示，以下分別說明：

1. 節點：節點通常代表的是模型所呈現的圖形外觀。
2. 連線：連線是連接兩個關聯節點的連接線段。
3. 標籤：標籤是節點或連線上的文字或圖示說明。
4. 容器區間：容器區間是在節點圖形內的一個獨立區間，並可在該區間內加入上述的三種圖形概念。也就是說，容器區間可以當成是一個放置節點與連線的容器。把視野放大來看，我們也可以說整個圖形化模型編輯器就是一個根節點元件的容器區間。

除了圖形元件之外，圖形化模型編輯器內最重要的組件就是工具面板模型。工具面板是編輯器內主要的控制組件，它必須提供給使用者通用的編輯區模型選取工具、畫面縮放工具、節點與連線生成工具等等功能。如圖 2 所示，頂端三個項目分別是編輯區選取工具、縮放工具、註釋工具，其下則是節點項目工具群及連線項目工具群。可以說如果沒有工具面板，就無法在圖面上產生任何模型。

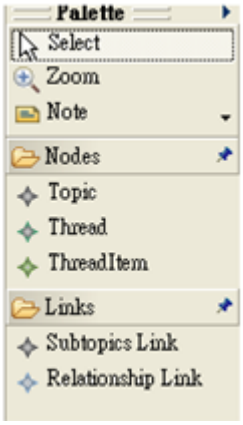

圖 2 工具面板範例

3. GMF 的設計架構

使用 GMF 開發編輯器的流程如圖 3 所示。首先開發人員需先設計好前三個模型檔案。領域模型、圖形模型以及工具面板模型。這三個模型本身的概念雖然是相依的，在設計階段時卻是以各自獨立的方式進行設計，也就是說領域模型裡完全沒有圖形模型的任何資訊，也不知道工具面板模型目前的狀況；在圖形模型與工具面板模型的情況亦同。準備好這三個模型後再需透過對應模型將這三組模型元素匹配起來，使得工具面板知道哪個工具項目該生成哪種模型，該模型又需要使用何種圖形來顯示等資訊。備好這四個模型之後再透過 GenModel 工具來產生程式碼，便可得到一個多功能的圖形化模型編輯器。

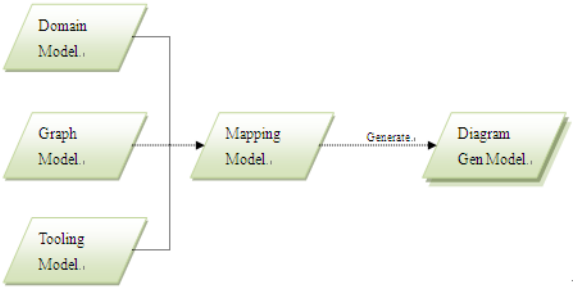

圖 3 GMF 模型開發流程圖

GMF 開發所需的四大模型為領域模型、圖形模型、工具面板模型及對應定義模型，其模型定義分別受 Ecore 超模型[5] 、gmfgraph 超模型、gmftool 超模型及 gmfmap 超模型所規範。其中領域模型使用的是 EMF 附帶的 Ecore 模型定義，因此除了 Ecore 模型之外的其餘三個模型皆是由 GMF 所架構出來的。

Ecore 模型是 UML [27] 類別圖的子集，Ecore 模型的設計便是以 EClass 為中心來思考。EClass 就是類別，其下有 EAttribute 跟 EReference，EAttribute 指的是 Java 原生的資料型別，EReference 則指稱到自定的 EClass 類別。

Gmftool 超模型用來定義工具面板裡的工具項目。超模型主要的元件有工具面板容器區間及工具項目的規劃。

Gmfgraph 超模型用來定義編輯器所能建構的圖形模型，也就是模型的圖形化結構。定義圖形模型時要分別定義其圖庫區及其圖形索引。圖庫區內主要用來定義代表模型的節點結構、線段結構和文字標籤屬性。圖形索引則是用來指稱到圖庫區內的圖形結構，以便讓對應模型來指稱。若節點內擁有容器區間，也是在圖形索引區中定義之。

Gmfmap 超模型是擔任以上三大模型的媒合角色，也就是對應模型。GMF 模型開發的最後一步就是在對應模型裡定義工具面板的工具項目要產生出的領域模型，而該領域模型在編輯器中又是以什麼圖形結構來呈現。對應模型的結構分為節點定義區和線段定義區，在節點定義區內定義節點的文字標籤對應和容器區間內子節點的模型對應等；線段定義區則為線段的文字標籤對應及線段的模型對應等等。

4. GM3 架構規範

在 GMF 開發所需四大模型的模型規範中，Ecore 超模型是由 EMF 所提供，而其餘三個超模型則是由 GMF 所制訂。GMF 所訂定的模型規範相當複雜，其所提供的模型輔助編輯器在模型元件的變更操作上很不方便，規劃精靈也會產生預設的索引不匹配等問題，因此 GMF 所提供的模型開發工具並不是非常成熟易用。

本研究的想法是希望在編輯 GMF 開發所需的四大模型檔案之上，再建立一層專屬模型語言的設計，使得四大模型檔案不需用戶以互動的方式產生，而是透過模型轉換的功能，利用專屬模型語言設計好模型之後，自動轉換

成 GMF 所需的四大模型檔案。如此一來，不但可以節省模型建置的時間，更可以解決模型與模型間因匹配不一致所產生的調校等等問題，進而改善核心模型的規劃時程。

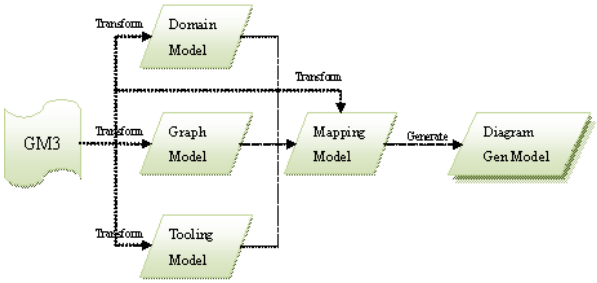

圖 4 GM3 專屬語言開發流程圖

圖 4 是加入 GM3 專屬模型語言後的圖形化模型編輯器開發流程圖。圖 5 則是簡化過的 GM3 專屬模型語言的超模型架構。

與 GMF 的各大模型比較，GM3 語言可以說是 GMF 模型規範的核心語言。在 GM3 語言中，我們將各大模型的核心元件保留下來，僅將圖形化模型編輯器的設計拆分成領域模型、工具面板模型和圖形模型三大部分。並利用兩條連結分別代表工具面板與領域模型的對應，以及領域模型與圖形模型的對應匹配關係。

4.1 領域模型

領域模型是圖形化模型編輯器設計的核心模型，可以說所有的模型都是圍繞著領域模型在打轉。領域模型主要依循 UML 的規範來架構，而圖形化模型編輯器主要用來呈現模型間的關聯性結構，因此在領域模型的規範中僅保留類別（Class）、屬性欄位（Attribute）及參考欄位 （Reference）元件的設計，而沒有將方法操作（Operation）等等其他元件納入考量。

類別是領域模型的核心元件。類別有一個指稱到自己的 super 連結，此連結即表示類別與類別之間的繼承關係。

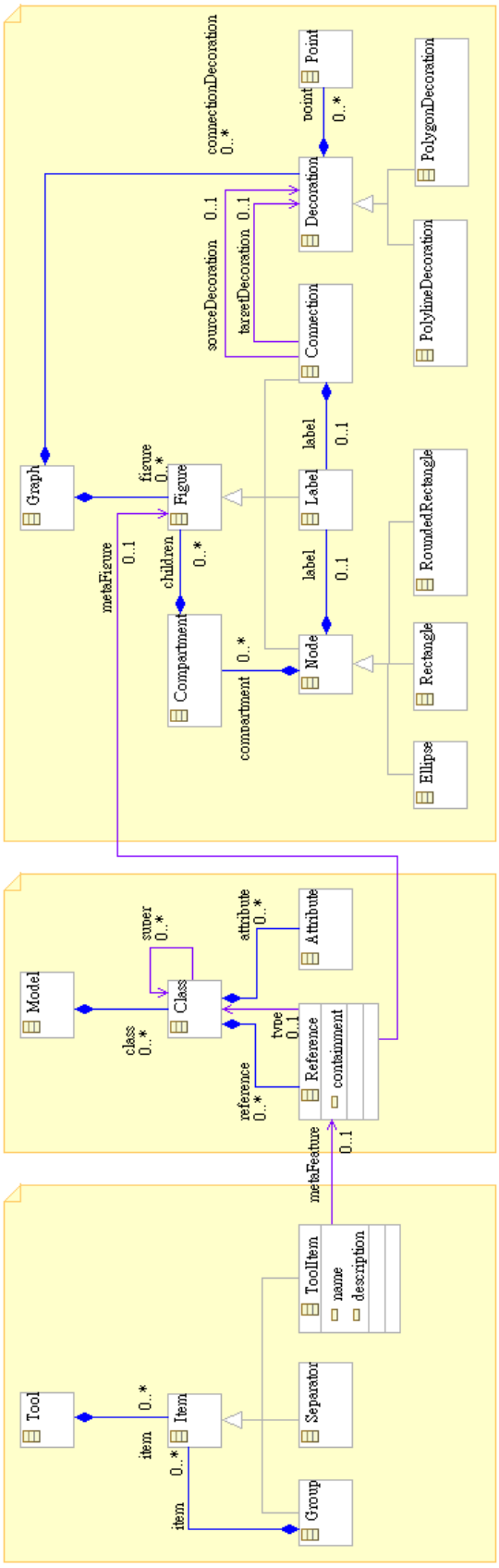

圖 5 GM3 超模型架構（簡化版）

類別下可建立屬性欄位和參考欄位。屬性欄位可用來包含原始資料型態，儲存主要的程式資料。參考欄位則是一個指稱到其他類別物件的變數，當類別物件需要參考到其他物件時，便使用參考欄位來做指稱。參考欄位內有一個 containment 的 boolean 設定，用來表示該參考欄位是屬於組合參考或是關聯參考。

參考欄位是 GM3 三大模型對應關係中的核心元件。領域模型便是透過參考欄位的 metaFeature 及 metaFigure 連結分別與工具面板模型和圖形模型做匹配。

4.2 工具面板模型

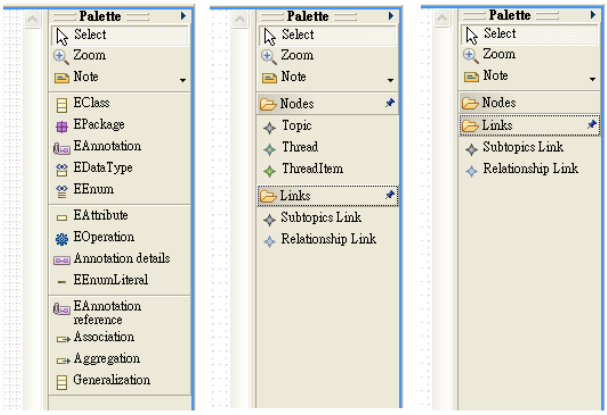

圖 6 工具面板範例集

圖形化編輯器裡，工具面板的組成大致可分為兩大部分，一個是通用工具部分，另一個則是模型生成工具部分。通用工具部分屬於制式化形式，其工具項目主要是模型選擇游標，畫布的縮放及文字註釋工具集。面板的最上方預設一定會擺上這些工具項目，而我們真正的設計核心，則是在其下的模型生成工具部分。

在工具面板模型規範中，主要的設定有工具群 (Group)，分隔板 (Separator) 及工具項目 (ToolItem)。工具項目是主要的作用元件，其餘的元件大多以面板佈局為目的。

工具群用來容納工具項目，將工具項目依照定義的先後順序依次擺放到工具群內。工具群具有區間收合的功能，當面板上的工具項目過多時，工具群也會自動收合以排出空位出來。

分隔板則以一個細橫條的形式出現在面板上，純粹當作工具項目間的區隔線之用，不具任何操作，也沒有名稱屬性。

工具項目提供工具名稱及說明文字的屬性設定。工具項目為對應關係中主要的匹配元件，並透過 metaFeature 連結來與領域模型參考元件（Reference）進行配對。

4.3 圖形模型

圖形模型是整個圖形化模型編輯器的設計重點，代表的是編輯器的表達能力，因此圖形模型便是以第二節所述的構成要素為重心來進行規劃，其元件相對比較複雜。

圖形化模型編輯器主要的構成要素是節點、線段、標籤以及容器區間。圖形模型的主要元件則是由節點模型（Node）、線段模型（Connection）和標籤模型（Label）所組成。容器區間模型（Compartment）則為節點模型的子元件。

在節點模型元件的配置方面，目前有橢圓形（Ellipse）、矩形（Rectangle）和圓角矩形（RoundedRectangle）供開發者選用，節點內部可再配置標籤模型元件及容器區間模型元件。

容器區間是依附在節點的內部，因此容器區間模型元件便成為節點模型元件內的一個子元件。容器區間模型元件內還需配置其他的圖形元件以符合容器區間的編輯需求。

線段模型元件可以設置標籤元件及箭頭樣式。箭頭樣式可供所有線段模型元件共用，因此在設計上是將其配置在圖形模型的根元件之下。線段模型元件利用 sourceDecoration 參考及 targetDecoration 參考分別指稱來源端與目標端的樣式元件。箭頭樣式分為空心樣式（PolylineDecoration）及實心樣式（PolygonDecoration）兩種。可利用座標元件(Point)來設計其箭頭樣式。若不設定的話，則是使用 V 字型及實心三角形來做為預設的箭頭樣式。

最後看到 Figure 這個模型元件。它是圖形模型主要元件的抽象模型，所以我們便透過它來與領域模型做匹配對應。

透過以上的模型配置我們即可轉換出如圖 4 中第二階段的各個模型來達到 GMF 圖形化模型編輯器的設計需求。開發者不需要額

外設定即可得到一個功能完整且符合領域模型設計規範的圖形化模型編輯器。

在 GMF 圖形模型的設計上還有很多可客製化的功能。而 GM3 專屬模型語言的設計之初是以簡潔易用為主要的設計宗旨，將核心元件以外的功能皆排除在外，難免會損失 GMF 的部分功能。因此在我們的開發流程中還保留 GMF 的設計階段，讓設計者能夠將其餘的功能留到該階段時再行擴充，以保留 GMF 完整的塑模能力。

5. 圖形化自動建構規則

我們除了提供 GM3 專屬語言來做核心模型元件初步的規劃工作之外，在語言工具中還提供一個預設的圖形化模型自動建置功能。以下就是我們的圖形化模型建構策略的探討。

圖形化編輯器最主要的構成要素是領域模型、圖形模型和工具面板模型。當領域模型架構好之後，後續的工作就是完成圖形模型與工具面板模型的設計。

首先我們來探討領域模型的組成與圖形模型的對應關係。領域模型能夠對應到何種圖形結構，端看領域模型中模型與模型之間是形成什麼樣的關聯性而定。領域模型中與圖形相關的關聯性有五種，分別是關聯關係 (has-a)、組合關係 (contain-a)、多重性 (multiplicity)、泛化關係 (is-a) 以及抽象性 (abstraction)。

1. 關聯關係 (Aggregation)

關聯關係在 UML 中是以一個端點為空心菱形的連線來顯示，指的是獨立模型之間的參考使用。如 A 類別關聯 B 類別，代表 A 模型中有一個參考到 B 模型的變數，而 A 模型與 B 模型並不屬於同一整體，可以各自獨立存在。

2. 組合關係 (Composition)

組合關係又稱為組合聚合關係，在 UML 中是以端點為實心菱形的連線或容器區間來表示。當一個模型是另外一個模型的其中一部分時使用。例如 A 類別組合 B 類別，則 B 類別是 A 類別其中一部分，因此在 A 類別建立後，B 類別才會跟著被建立出來。同理，當 A 類別被消滅時，B 類別也會隨著被銷毀。

3. 多重性 (Multiplicity)

多重性是應用在關聯關係和組合關係連線上的數字，指的是所參考類別的物件個數。

4. 泛化關係 (Generation)

泛化關係又稱作繼承關係，指的是通用類別與特殊化類別之間的關係，並以端點為空心三角形的連線表示。其中該通用類別與特殊化類別屬於相同種類。也就是說，特殊化類別是通用類別的擴充版本，在執行時期特殊化類別可以視為通用類別來使用。

5. 抽象化 (Abstraction)

抽象化應用在泛化關係的父節點上，以 abstract 這個限定詞來表示。被標註為抽象性的類別並不能夠在執行時期產生實體物件，該類別僅供子類別定義共用結構之用，或是在使用多型時作為通用的參考類別。若要產生實體物件則需使用其未抽象化的子類別才行。

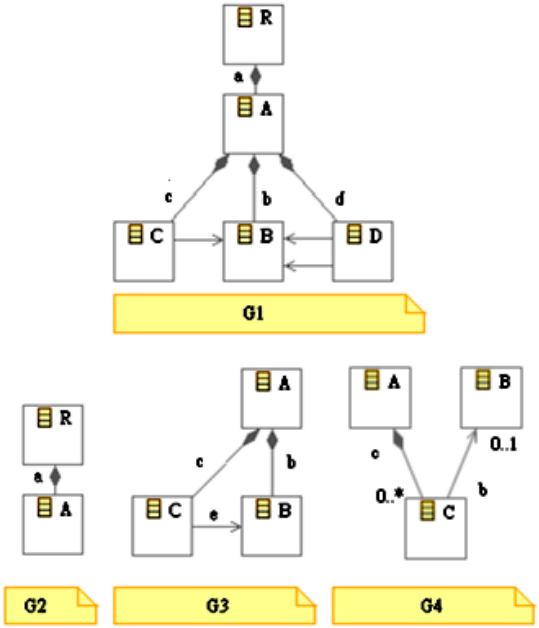

圖 7 領域模型範例

領域模型與圖形模型的對應具有多樣性。在考量各種搭配與物件間的依存關係後，我們提出以下的自動建構規則。以下利用圖 7 來說明將領域模型建構成圖形模型的自動化建構規則：

1. 被組合的類別 -> 節點 

如圖 7 的 G2 部分為例，R 類別透過 a 參考變數來組合 A 類別，因此在類別 R 中，每個 a 參考到的類別物件都可以轉成一個節點圖形。

2. 類別中含有組合參考 -> 容器節點

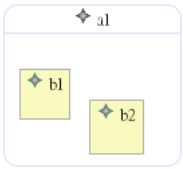

圖 8 容器節點

當類別中存在組合關係 (contain-a) 的參考時，如圖 7 的 G1 中 A 類別透過 b 參考變數來組合 B 類別，則可將 B 類別視為 A 類別整體的元件之一，因此在圖形的呈現上我們可以在 A 類別的節點中加入一個含有 B 類別集合的容器區間，如圖 8 所示。

換句話說，若要將節點類別建立在某個容器節點內，則在領域模型中，必須將該節點類別組合在某容器節點的類別內。

3. 關聯參考 -> 線段

當類別含有關聯關係 (has-a) 的參考時，此關聯參考可以在該類別物件與關聯目標類別物件之間形成一個連線。以圖 7 的 G3 為例，C 類別與 B 類別有一個 e 的關聯參考，則可建立一個從 C 類別物件節點連接到 B 類別物件節點的 e 連線。

4. 多重性

多重性可以定義在組合關係或關聯關係上。如果定義在組合關係上，其多重性的值則表示在該容器區間內能夠產生子模型的數量。如果是在關聯關係上，其值則代表該模型與目標模型能夠產生的連線個數。

5. 屬性欄位 -> 圖示 (icon) 標籤 

在類別中若有字串變數或其它屬性值時，可將字串化後的屬性值以圖示或文字標籤的形式放置在節點或線段中。

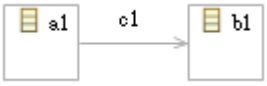

圖 9 屬性線段

6. 擁有關聯參考的被組合類別 -> 具有屬性設定之線段

純粹利用 EReference 建置出來的線段其領域模型只是節點類別內一個對外的關聯參考，關聯參考本身並不具有任何的屬性設定。如果我們希望在線段中也能加入文字標籤等等的屬性設定時，便可以利用類別的模型物件來建立線段結構。

當一個被組合的類別中擁有一個關聯參考時，我們可以把該類別物件視為一個連接線段。如圖 7 中 G4 領域模型中的 C 類別所示：A 類別透過 C 類別來與 B 類別產生關聯。其中 C 類別被 A 類別所組合，並擁有對 B 類別的關聯參考，則在產生模型實例時，可以利用一個 c1 類別物件來將 a1 類別物件與 b1 類別物件連接起來，即圖 9 所示。其中 C 類別的容器 （A 類別）和 C 類別的關聯參考 （指向 B 類別）則分別指稱到連線的來源模型和目標模型。

由於此結構規範亦可在產生模型實例時建構成 A 容器區間包含C 節點和 C 節點與 B 節點的線段組合，所以此種轉換法可以算是通則中的特例。若需要使用此這項建置規則，則必須由使用者自行指定之。

6. 使用範例

程式語言的實際語法主要分為文字型語法和圖表型語法。文字型語法能夠記錄比較複雜的表達式，但比較不好理解也不容易管理。圖表型語法優點在於能夠大量的表達概念與其關聯性，而缺點則是能夠表達的細節有限。

GM3 專屬模型語言為結合文字型語言可詳盡描述的優點，及圖表型（結構式）語言有簡潔概念的優點，因此在設計語法時，採用精修過的文字型語法做為語言描述。在這領域中，比較有名的例子就是將 EMF 的 Ecore 模型文字化的 KM3 語言。

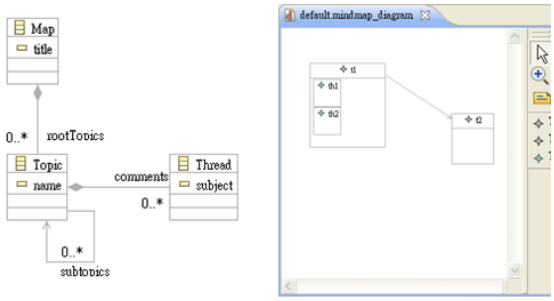

圖 10 心智圖領域模型及其範例

我們透過設計一個圖形化模型編輯器的例子來說明如何使用我們的 GM3 語言進行開發。 圖 10 是一個心智圖（Mindmap）的領域模型。整個領域模型以 Map 類別為根節點，也就是說 Map 是包含整個編輯區的根容器類別，在編輯區內可產生其所包含的節點及連線模型元件。我們希望所設計出來的編輯器如同圖 10 右方範例所示，能夠在編輯區內產生主標題節點 (Topic 類別)，在主標題節點內可以放置多個子標題 (Thread 類別)，並在主標題與主標題之間可以用關聯線段來連結 (subtopics 參考)。其中 Topic 類別中的 name 屬性和 Thread 類別內的 subject 屬性皆是用來產生文字標籤的字串型別。

表 1 心智圖之領域模型 - GM3 語言

```
model mindmap {
  class Map {
    attribute title[0-1] : String;
    reference rootTopics[*] container : Topic; (TopicNode)
  }
  class Topic {
    attribute name[0-1] : String; (TopicName)
    reference subtopics[*] : Topic; (TopicLink)
    reference comments[*] container : Thread; (ThreadNode)
  }
  class Thread {
    attribute subject[0-1] : String; (ThreadSubject)
  }}
```

由以上敘述我們得到一個如表 1 的領域模型語言描述。

參考圖 5 的 GM3 語言規範，我們看到表 1 第 3 行中，GM3 語法利用超類別（mataClass）的小寫名稱做為描述 Class 模型元件文字型語法的關鍵字。同理，其它模型元件的文字型語法也都是依照此結構來描述。接著關鍵字之後的是該模型的名稱定義、多重性定義及參考型別定義。最後括號內描述的是指稱到圖形模型元件名稱的 metaFigure 參考。

表 2 心智圖之工具面板模型 - GM3 語言

```
tool mindmapPalette {
  group nodeGroup {
    toolItem Topic = Map/rootTopics     : dsc="New Topic";
    toolItem Thread = Topic/comments : dsc="New Thread";
  }
  group linkGroup {
    toolItem Subtopic = Topic/subtopics: dsc="New Subtopic";
  }}
```

表 2 是我們定義的工具面板模型。如同前述的語法規則，在工具面板模型的定義上也是以小寫名稱作為模型元件的關鍵字。比較特別的是工具項目元件名稱定義之後的等號敘述，我們刻意將工具項目連往領域模型的對應參考設定在名稱定義之後，以加強檢閱的效率。該對應參考的定義方式是以 *類別/參考* 的型式來描述。舉例來說，表 2 的第 3 行定義了一個名為 Topic 的工具項目，並以領域模型 Map 類別中的 rootTopics 參考為其所要產生的模型元件之參考連結，並定義該項目的詳細說明為 “New Topic”。

表 3 心智圖之圖形模型 – GM3 語言

```
graph mindmapGraph {
  node TopicNode[*]    : roundedRectangle {
    label TopicName;
    compartment ThreadCompartment {
      node ThreadNode[*] : rectangle {
        label ThreadSubject;
      }
    }
  }

  connection TopicLink[*] {
    targetDecoration: ArrowDecoration;
  }

  polylineDecoration ArrowDecoration;
}
```

最後表 3 是我們的圖形模型定義。在領域模型中，Thread 模型是定義為 Topic 模型

的子元件，因此在定義圖形模型時，該結構就會如同上述 compartment 裡的巢狀敘述。

透過以上的定義，我們便可得到如圖 11 所示的 Mindmap 圖形化模型編輯器，並擁有圖中右側的縮圖大綱視圖等等完整功能。

如果開發者不希望做太多的設計工作，也可以採用我們的工具來執行 GM3 預設的圖形化自動建構功能。以心智圖為例，開發者僅需提供如表 1 所述的領域模型定義，GM3 語言工具即會採用第五節所描述的建構策略來產生與圖 11 相同規格的圖形化模型編輯器。

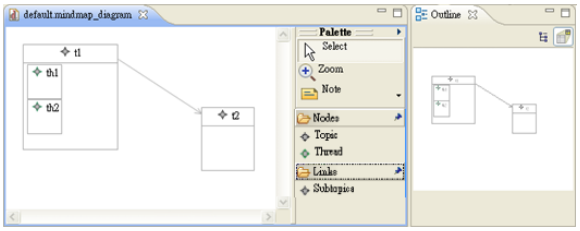

圖 11 心智圖之圖形化模型編輯器

7. 結論

程式模型一般是儲存資料的主要結構，而模型的圖形化可以依照開發者的意願任意組合。如類別內的屬性欄位可以顯示成節點內的屬性（圖示標籤），也可以顯示為某個關聯參考連線的屬性設定，端看開發者如何安排其顯示結構。

與一般的圖形化編輯器設計框架（如 Grace[19] , Tiger[26] ）比較起來，GMF 已經非常強大且容易開發。其原始模型規範相當龐大，提供許多模型元件的客製化能力。然而其複雜的設計理念也造成初期核心模型規劃與編製上的困難，使得利用 GMF 來開發圖形化編輯器仍然具有一定的學習門檻，並且其塑模引導精靈的自動化能力也未盡完善。

本研究便針對 GMF 塑模機制上的弱點加以改善，提供 GM3 專屬語言做為加速開發流程的設計語言，保留其原本的設計能力以進行各種客製化的設計，並提出一個將領域模型自動轉換為圖形結構的圖形化建構策略，以彌補 GMF 塑模引導精靈功能上的不足。

GMF 的各大模型極其複雜，編修起來相當辛苦。使用我們的 GM3 專屬語言作為初步的編輯器規劃，既可以達到快速編修的功能，又可以免去模型間對應性錯亂而不易察覺的困擾，再加上我們圖形化建構策略的自動化工具，讓開發者能夠更快速地進入 GMF 的設計領域。

參考文獻

[1] Sherry Shavor, Jim D'Anjou, Scott Fairbrother, Dan Kehn, John Kellerman, Pat McCarthy , The Java Developer's Guide to Eclipse, Addison-Wesley, 2007.

[2] Bill Moore, David Dean, Anna Gerber, Gunnar Wagenknecht, Philippe Vander-heyden, Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework, Feb 2004. http://www.redbooks.ibm.com/redbooks/pdfs/sg246302.pdf

[3] David Gallardo, Ed Burnette, Robert McGovern, Eclipse in Action, Manning, 2003.

[4] Tony Clark, Andy Evans, Paul Sammut, James Willans. “Language Driven Development and MDA”. MDA Journal, Ocober 2004, http://www.bptrends.com/publicationfiles/10-04 COL MDA - Frankel - Xactium.pdf

[5] Frank Budinsky, David Steinberg, Ed Merks, Raymond Ellersick, Timothy J. Grose , Eclipse Modeling Framework, Addison-Wesley, 2003.

[6] Applied Metamodelling, 2007. http://albini.xactium.com/web/downloads/b1a35960appliedMetamodelling.pdf

[7] ATLAS Transformation Language, 2007. http://www.eclipse.org/gmt/atl/

[8] Creating your own Domain Specific Modeler using GMF, 2007. http://eclipsezilla.eclipsecon.org/attachment.cgi?id=175

[9] Developer Guide to the GMF Runtime Framework, 2007. http://help.eclipse.org/help32/topic/org.eclipse.gmf.doc/prog-guide/runtime/Developer%20Guide%20to%20Diagram%20Runtime.html

[10] Display a UML Diagram using Draw2D, 2007. http://www.eclipse.org/articles/Article-GEF-Draw2d/GEF-Draw2d.html

[11] Domain Specific Language and MDA, 2007.

http://albini.xactium.com/web/downloads/XactiumDomainSpecificModellingWP.pdf

[12] Eclipse.org, 2007. http://www.eclipse.org/

[13] Eclipse Platform Technical Overview, 2007. http://www.eclipse.org/whitepapers/eclipse-overview.pdf

[14] Eclipse Modeling Framework, 2007. http://www.eclipse.org/emf/

[15] Eclipse Modeling Framework Technology project, 2007. http://www.eclipse.org/emft/projects/

[16] Generative Modeling Technologies Project, 2007. http://www.eclipse.org /gmt/

[17] Graphical Editing Framework, 2007. http://www.eclipse.org/gef/

[18] Graphical Modeling Framework, 2007. http://www.eclipse.org/gmf/

[19] Grace, 2006. http://www.doclsf.de /grace/

[20] Human-Usable Textual Notation Specification, 2007. http://www.omg.org/docs/ptc/04-01-10.pdf

[21] MDA, 2007. http://www.omg.org/mda/

[22] Meta Object Facility Specification (MOF), 2007. http://www.omg.org/docs/formal/00-04-03.pdf

[23] Object Constraint Language Specification, 2007. http://www.omg.org/docs/formal/03-03-13.pdf

[24] Object Management Group, Inc., 2007. http://www.omg.org

[25] The Design Patterns Java Companion, 2007. http://www.patterndepot.com /put/8/JavaPatterns.htm

[26] Tiger Project, 2006. http://tfs.cs.tu-berlin.de/~tigerprj/

[27] UML, 2007. http://www.uml.org/

[28] XML Metadata Interchange, 2007. http://www.omg.org/technology/documents/formal/xmi.htm