

UML 系列之一

UML 答客問

第二輯

作者簡介

邱郁惠(271080@gmail.com)

畢業於東吳大學資訊科學系，研究 UML、OOA 十餘年，2007 年創辦 UML

Blog(<http://www.uml.tw>)推廣 UML 技術，並專職於企業內訓、專案輔導、自辦課程、專欄寫作。

擔任過 NEC、華夏、中科院、百通、MISOO 物件教室、大竝、中華汽車、HSDc(2007)、資策會(2008)、台灣大哥大(2008)、意藍科技(2008)、新鼎(2008)、博客來(2009)、網飛訊(2010)、文化大學推廣部(2010)、北區職訓局(2010)、PMI-TW 國際專案管理學會(2010)、巨鷗(2010)、三商電腦(2010)、秀傳醫療(2011)、翔生資訊(2011)、國際厚生(2011)、中華電信(2012)等公司的內訓講師及輔導顧問，也擔任過物件導向雜誌主編暨 UML/OOAD 專欄作家、RUN！PC 旗標資訊月刊(2008~2009)、以及 iThome 電腦報(2008~2012)專欄作家。

出版的繁體書有《寫給 SA 的 UML/MDA 實務手冊》(天瓏銷售第 1 名,已絕版)、《寫給 C++程式設計師的 UML 實務手冊》(天瓏銷售第 4 名,已絕版)、《UML-SystemC 晶片設計實務》(已絕版)、《OCUP/UML 初級認證攻略》(天瓏銷售第 14 名,已絕版)、《寫給 SA 的 UML/UseCase 實務手冊》(天瓏銷售第 10 名,已絕版)、《學會 UML/OOAD 這樣開始就對了》(金石堂預購第 1 名,已絕版)、《Visual Studio 2010/UML 黃金準則》、《SA 前進 UML 專案現場》、《IT 人，你如何表達？》(未出紙本版)。

同時，出版的簡體書有《系統分析員 UML 實務手冊》、《C++程式師 UML 實務手冊》、《SoC 設計實務手冊》、《UML 那些事兒》、《系統分析師 UML 用例實戰》、《UML 和 OOAD 快速入門》、《Visual Studio 2010 和 UML 黃金法則》、《系統分析師 UML 項目實戰》。

目前擁有 OCUP(OMG-Certified UML Professional)三級認證、PMP(Project Management Professional)認證、ITIL V3 Foundation 認證、IBM OOAD(Object Oriented Analysis and Design)認證、Scrum Master 認證，曾榮獲大陸頒予《優秀 IT 技術圖書原創作者》獎。

目錄



- Q11 ArgoUML 之簡介與操作示範 ---- 2
- Q12 Case Complete 之簡介與操作示範 ---- 34
- Q13 有關於 actor 與 extend 的疑問？ ---- 59
- Q14 開規格該用哪幾種 UML 圖呢？ ---- 64
- Q15 StarUML 只能針對類別圖產碼嗎？ ---- 65
- Q16 超商收銀機系統之使用案例圖提問 ---- 66
- Q17 IBM RSA 之簡介與操作示範 ---- 67
- Q18 CIM-1~3 並沒有絕對的順序？ ---- 129
- Q19 EA(Enterprise Architect)之操作示範 ---- 138
- Q20 雙向結合的圖示，到底有沒有箭頭？ ---- 180

目錄

Q11	ArgoUML 之簡介與操作示範 ----	2
Q11.1	免費工具—ArgoUML ----	2
Q11.2	開啟類別圖 ----	10
Q11.3	新增類別 ----	14
Q11.4	新增屬性 ----	18
Q11.5	新增操作 ----	21
Q11.6	依賴關係 ----	24
Q11.7	一般化關係 ----	27
Q11.8	組合關係 ----	30
Q11.9	綜合練習 ----	32
Q12	Case Complete 之簡介與操作示範 ----	34
Q12.1	試用工具—Case Complete ----	34
Q12.2	定義參與者 ----	40
Q12.3	定義參與者之目的 ----	43
Q12.4	定義使用案例 ----	48
Q12.5	增加欄位 ----	53
Q12.6	設定參照附件 ----	54
Q12.7	產生報表 ----	55
Q13	有關於 actor 與 extend 的疑問？ ----	59
Q14	開規格該用哪幾種 UML 圖呢？ ----	64

- Q15 StarUML 只能針對類別圖產碼嗎? ---- 65
- Q16 超商收銀機系統之使用案例圖提問 ---- 66
- Q17 IBM RSA 之簡介與操作示範 ---- 67
- Q17.1 MAD 開發工具—IBM RSA ---- 67
 - Q17.2 使用 RSA 產出 PIM ---- 70
 - Q17.3 使用 RSA 產出 PSM ---- 106
 - Q17.4 RSA 未來可能的發展 ---- 123
- Q18 CIM-1~3 並沒有絕對的順序? ---- 129
- Q19 EA(Enterprise Architect)之操作示範 ---- 138
- Q19.1 簡易的開發程序 ---- 138
 - Q19.2 新增專案 ---- 140
 - Q19.3 繪製使用案例圖 ---- 142
 - Q19.4 繪製類別圖 ---- 148
 - Q19.5 繪製循序圖 ---- 162
 - Q19.6 產生 Java 程式碼 ---- 169
 - Q19.7 產生文件 ---- 173
- Q20 雙向結合的圖示，到底有沒有箭頭? ---- 180

UML 答客問

2

※※

- Q11 ArgoUML 之簡介與操作示範
- Q12 Case Complete 之簡介與操作示範
- Q13 有關於 actor 與 extend 的疑問？
- Q14 開規格該用哪幾種 UML 圖呢？
- Q15 StarUML 只能針對類別圖產碼嗎？
- Q16 超商收銀機系統之使用案例圖提問
- Q17 IBM RSA 之簡介與操作示範
- Q18 CIM-1~3 並沒有絕對的順序？
- Q19 EA(Enterprise Architect)之操作示範
- Q20 雙向結合的圖示，到底有沒有箭頭？

Q11 ArgoUML 之簡介與操作示範

※※

Q11.1	免費工具—ArgoUML	----	2
Q11.2	開啟類別圖	----	10
Q11.3	新增類別	----	14
Q11.4	新增屬性	----	18
Q11.5	新增操作	----	21
Q11.6	依賴關係	----	24
Q11.7	一般化關係	----	27
Q11.8	組合關係	----	30
Q11.9	綜合練習	----	32

前陣子因為工作緣故，客戶詢問ArgoUML，所以編寫此文介紹ArgoUML的主要功能與操作步驟。

Q11.1 免費工具—ArgoUML

ArgoUML 由 Tigris 開放原碼社群所提供，可以直接到這個社群網站 (<http://argouml.tigris.org/>) 下載約莫 6.5MB 的 ArgoUML 壓縮檔。在解壓縮之後，不需要經過安裝步驟，直接執行 argouml 可執行 Jar 檔即可。在本文中，我們用以示範說明的版本為 0.21.3。

ArgoUML 需要用到 JDK 1.4 或 5.0 執行環境，要是發現無法執行，請

確認電腦裡已經安裝有 JDK 1.4 以上的版本，無法執行於 JDK 1.3 以下的版本。請看圖 Q11-1 是 ArgoUML 淡雅的載入畫面，左上角有 v0.21.3 字樣標示版本。



圖 Q11-1: ArgoUML 的載入畫面

再看圖 Q11-2，這是 Argo 的主畫面，它還提供了繁體中文的選單，有著國際化的視野。請看圖 Q11-3 類別圖的繪製區，我們繪製了一張簡單的類別圖。在這張類別圖裡，我們繪製了 Member、Order 類別，並且兩者之間有結合關係，個體數目為 1 對 0..*。

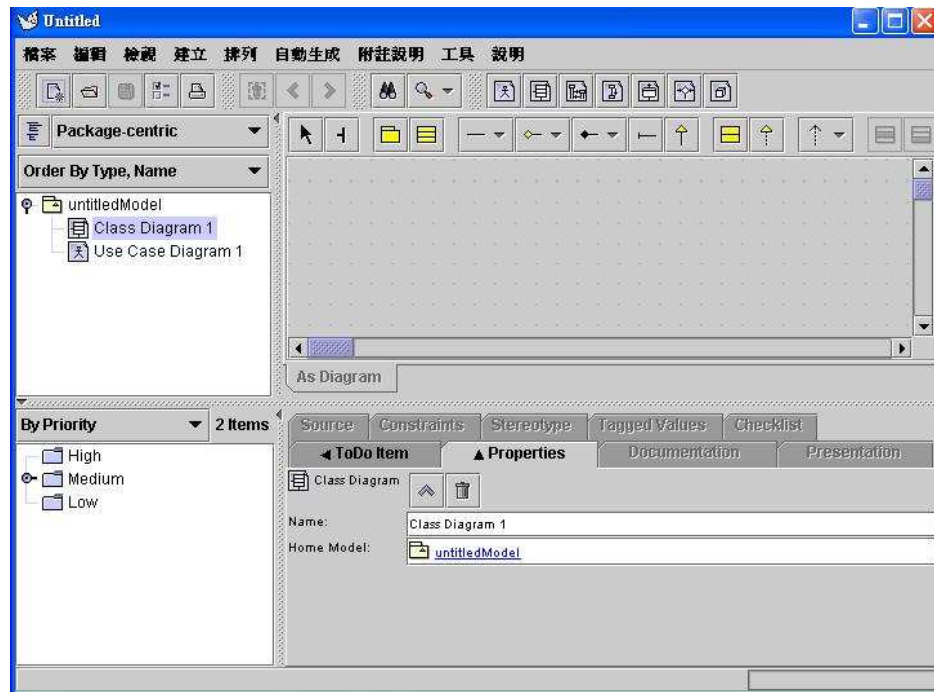


圖 Q11-2: Argouml 的主畫面

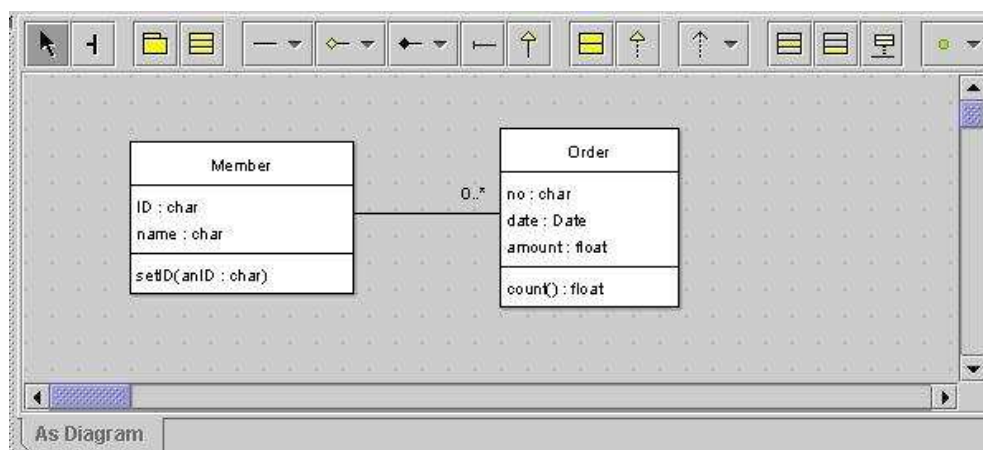


圖 Q11-3: Argouml 的類別圖繪製區

ArgoUML 支援多種格式的影像檔，像是 GIF、PNG、PS、EPS、SVG 等格式，匯出的 PNG 影像檔如圖 Q11-4 所示。

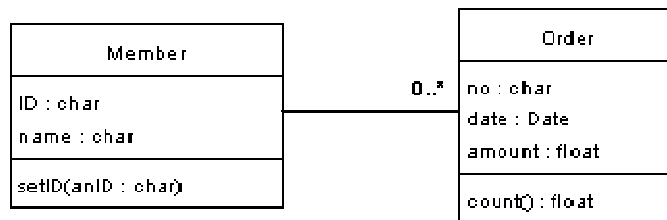


圖 Q11-4: ArgoUML 匯出的 PNG 影像檔

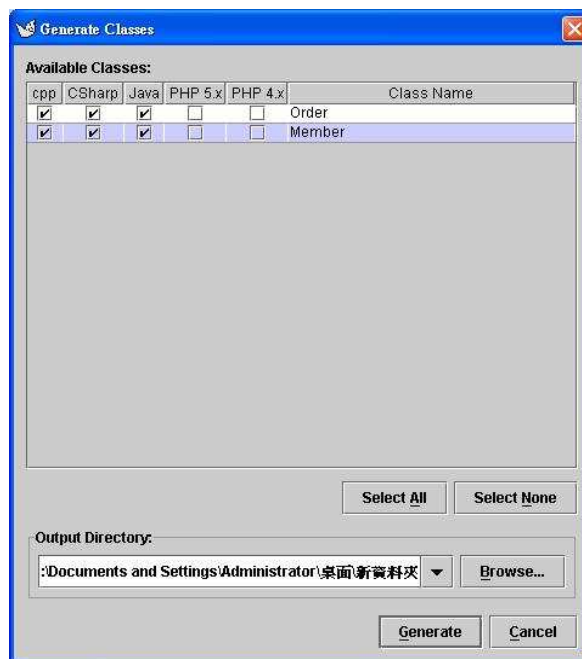


圖 Q11-5: ArgoUML 自動產碼的選項

對於許多使用者期盼的自動產碼功能，ArgoUML 也不含糊，提供了五種程式語言的選項，有 C++、C#、Java、PHP 5.x、PHP 4.x，如圖 Q11-5 所示。

針對圖 Q11-4 簡單的類別圖，我們選取了 C++、C#、Java 這三個常見的程式語言，讓 ArgoUML 來自動產碼。生成的程式碼中，有許多供 ArgoUML 讀取的註解，爲了節省篇幅，我們一一刪去了註解碼，僅留下有用的程式碼供您參考。由於 ArgoUML 生成的程式碼都很簡單，此處我們就不加以說明了。

C++的程式碼有.h 和.cpp 檔，如下圖 Q11-6~9 所列。

```
//// ArgoUML C++ Code
1.  #ifndef Member_h
2.  #define Member_h
3.  #include <vector>
4.  #include "Order.h"
5.  class Member
6.  {
7.  public:
8.      virtual setID(char anID);
9.  public:
10.     char ID;
11.     char name;
12.  public:
13.     std::vector< Order > myOrder;
14. };
15. #endif
//// ArgoUML C++ Code
```

圖 Q11-6: Member.h

```
//// ArgoUML C++ Code
1.  #include "Member.h"
2.  Member::setID(char anID)
```

```
3.  {
4.  }
//// ArgoUML C++ Code
```

圖 Q11-7: Member.cpp

```
//// ArgoUML C++ Code
1.  #ifndef Order_h
2.  #define Order_h
3.  #include "Member.h"
4.  #include "java/util/Date.h"
5.  class Order
6.  {
7.  public:
8.      virtual float count();
9.  public:
10.     char no;
11.     java::util::Date date;
12.     float amount;
13.  public:
14.     Member myMember;
15. };
16. #endif
//// ArgoUML C++ Code
```

圖 Q11-8: Order.h

```
//// ArgoUML C++ Code
1.  #include "Order.h"
2.  float Order::count()
3.  {
4.      return 0.0;
5.  }
//// ArgoUML C++ Code
```

圖 Q11-9: Order.cpp

生成的 C#的程式碼是.cs 檔，如下圖 Q11-10~11 所列。

```

//// ArgoUML C# Code
1.  public class Member
2.  {
3.      public char ID;
4.      public char name;
5.      public ArrayList  myOrder;
6.      public setID(char anID) { }
7.  }
//// ArgoUML C# Code

```

圖 Q11-10: Member.cs

```

//// ArgoUML C# Code
1.  using java.util;
2.  public class Order
3.  {
4.      public char no;
5.      public Date date;
6.      public float amount;
7.      public Member  myMember;
8.      public float count() {}
9.  }
//// ArgoUML C# Code

```

圖 Q11-11: Order.cs

生成的 Java 的程式碼是.java 檔，如下圖 Q11-12~13 所列。

```

//// ArgoUML Java Code
1.  import java.util.Vector;
2.  public class Member
3.  {
4.      public char ID;
5.      public char name;
6.      public Vector  myOrder;
7.      public setID(char anID) {}
8.  }
//// ArgoUML Java Code

```

圖 Q11-12: Member.java

```
//// ArgoUML Java Code
1.  import java.util.Date;
2.  public class Order
3.  {
4.      public char no;
5.      public Date date;
6.      public float amount;
7.      public Member myMember;
8.      public float count()
9.      {
10.         return 0.0;
11.     }
12. }
//// ArgoUML Java Code
```

圖 Q11-13: Order.java

其實，ArgoUML 具備相當多的特色，無法一一詳述，僅僅選擇說明關乎 UML 的部分。總歸來說，ArgoUML 具備下列多項特色：

- 可繪製 9 款 UML 圖—支援用例圖、類別圖、循序圖、狀態圖、物件圖、活動圖、通訊圖、元件圖和部署圖。不過，ArgoUML 目前還是僅支援到 UML 1.4 版，並未支援 UML 2.0 版裡的新圖款。
- 完全免費—ArgoUML 是一套開放原碼的軟體，不僅免費自由下載，連程式碼都免費開放。
- 跨平台—只要提供 JDK 1.4 或 5.0 執行環境的平台上，都可以執行 ArgoUML。
- 多國介面—ArgoUML 提供多國文字介面供使用者選擇，有美語、英文、法文、德文、西班牙文、俄文、挪威文、簡體中文、繁體中文、葡萄牙文等等，降低因外文造成的學習及使用門檻。

- 多種格式影像檔—可匯出 GIF、PNG、PS、EPS 和 SVG 等等格式的影像檔。
- 自動產碼—本文所採用的 ArgoUML v0.21.3 版本，目前可以依據類別圖的內容產出 Java、C++、C#、PHP 4.x、PHP 5.x 程式碼。
- 反向工程—目前 ArgoUML 可以讀取 .java、.jar 和 .class 反向解析出其內的類別、元件等等的模式元素，自動建出類別圖。反向工程有兩個主要用途，其一是舊有的原始程式碼反轉成圖之後，可以建構 UML 模式的方式繼續將新的設計添加上去；另一項用途是，想要解析原始程式碼時，可以透過反轉的類別圖來理解，不再需要觀看一行又一行的程式碼，這將節省大量的時間和精力。
- 語法檢驗—ArgoUML 遵守 UML 1.4 版的語法規則，不支援違反語法的繪圖動作。
- 支援 XMI—XMI 是一種以 XML 為基礎的交換格式，用以交換不同開發工具所產出的 UML 模式。ArgoUML 接受匯入 XMI 1.1 和 1.2 版，但是匯出的 XMI 版本為 1.2。

Q11.2 開啓類別圖

只要一執行 ArgoUML，就可以發現，ArgoUML 已經自動備妥一張名為 Class Diagram 1 的新類別圖。

開啓類別圖的步驟，如下所述：

1. 滑鼠左鍵雙擊瀏覽器裡的 untitledModel 之後，打開了其內的類別圖及類別圖項目，如圖 Q11-14 所示。

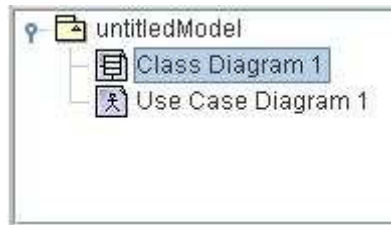


圖 Q11-14: 瀏覽器

2. 在 Argouml 裡，使用了如圖 Q11-15 的小圖示代表類別圖款。



圖 Q11-15: 類別圖款的圖示

3. 滑鼠左鍵雙擊類別圖項目之後，螢幕畫面的右上半部換上了類別圖繪製區，如圖 Q11-16 所示。

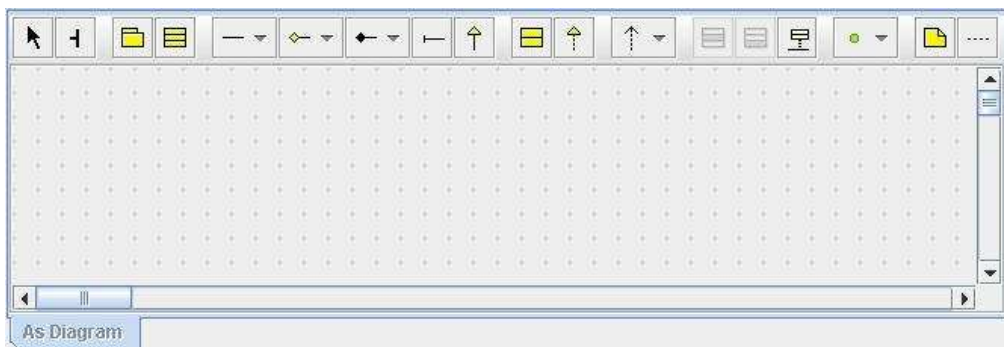


圖 Q11-16: 類別圖繪製區

當然，我們也可以新增自己的類別圖，步驟如下所述：

1. 在第二列選單裡，有一排 UML 圖款的按鍵，如圖 Q11-17 所示。



圖 Q11-17: UML 圖款

2. 滑鼠左鍵按下類別圖款按鍵之後，您會發現螢幕畫面左上半部的瀏覽器裡，多出了一張類別圖。
3. 另一種新增類別圖的方式，點選第一列選單裡的【建立►新 Class 圖】，即可。
4. 第三種新增類別圖的方式，滑鼠右鍵單擊選取瀏覽器裡的任一項目，於出現的選單中選取【Create Diagram►新 Class 圖】，即可。

瀏覽器裡出現的任何類別圖都可以刪除，包括我們自己建立的，或是 ArgoUML 一開始自動備妥的類別圖。一旦執行了刪除動作之後，是無法復原的。

不過，刪除類別圖並不代表刪除類別圖裡的類別或關係。比方說，我們去遊玩時照了很多張相片，回家觀看之後，可能刪除一些照的不好的相片，但並不會因此刪除照片裡的景物。每一張類別圖都只是某些類別或關係的照片，刪除類別圖並不會因此刪除被照入的類別圖像或關係圖像。

刪除類別圖的步驟，如下所述：

1. 滑鼠左鍵單擊選取瀏覽器裡的類別圖項目之後，螢幕畫面右下半部會出現 Properties 頁區。

2. 在 ArgoUML 裡，使用了垃圾桶按鍵代表從模型中刪除的功能。
3. 滑鼠左鍵按下類別圖 Properties 頁區裡的垃圾桶按鍵，即可由模型中刪除類別圖，如圖 Q11-18 所示。

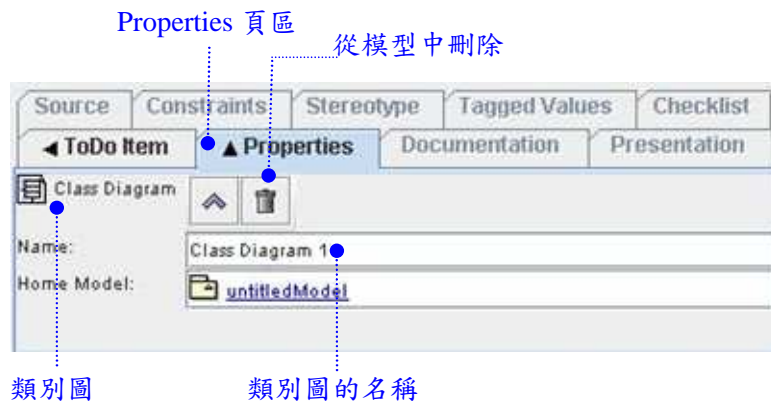


圖 Q11-18: 類別圖的 Properties 頁區

4. 另一種刪除類別圖的方式，滑鼠左鍵單擊選取瀏覽器裡，欲刪除的類別圖項目，再點選第一列選單裡的【編輯 ► 由模型中刪除】，即可。
5. 第三種刪除類別圖的方式，滑鼠右鍵單擊選取瀏覽器裡的類別圖項目，於出現的選單中選取【由模型中刪除】，即可。

最後，我們來看類別圖的工具箱，如圖 Q11-19 所示。開啓任何一張類別圖，繪製區的上方就出現工具箱，列出可以放置在類別圖面上的相關圖示。本文裡，我們會用到類別、屬性、操作、結合、聚合、組合和一般化關係。

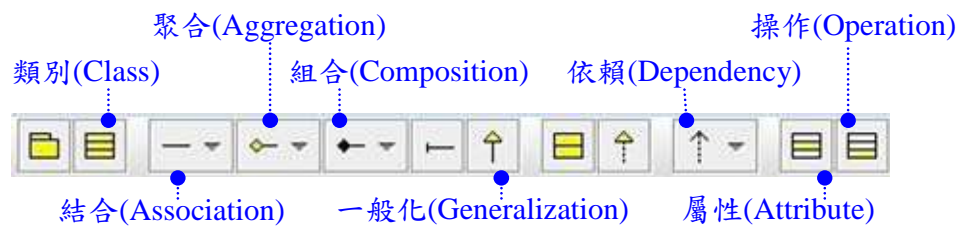


圖 Q11-19: 類別圖的工具箱

Q11.3 新增類別

首先，我們來新增類別，步驟如下所述：

1. 滑鼠左鍵按下工具箱裡的類別按鍵之後，會出現十字型游標。
2. 於類別圖任一空白處，再一次按下滑鼠左鍵，便新增了一個無名的類別，如圖 Q11-20 所示。

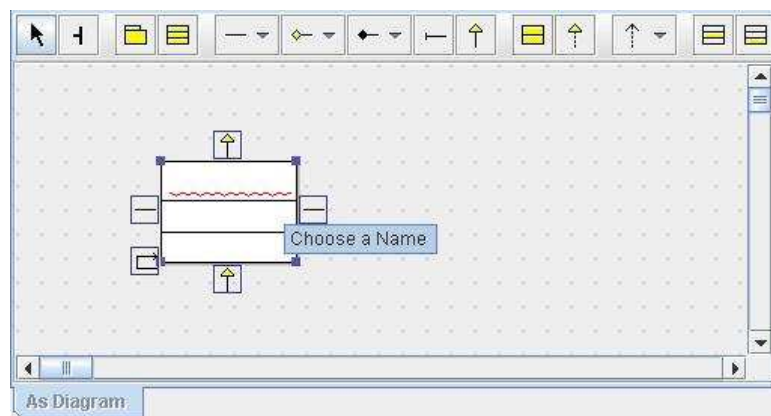


圖 Q11-20: 無名的類別

- 滑鼠左鍵點選圖內的類別圖示，並於三格矩形的第一格處，雙擊滑鼠左鍵，填寫類別名稱，如圖 Q11-21 所示。

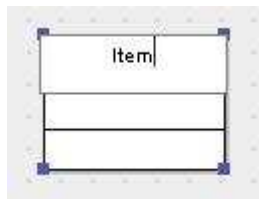


圖 Q11-21: 填入類別的名稱

- 另一種填寫類別名稱的方式，於類別 Properties 頁區裡的 Name 空格，鍵入類別的名稱即可，如圖 Q11-22 所示。



圖 Q11-22: 類別的 Properties 頁區

- 如果，我們要將這個類別指定為抽象類別的話，請於類別

Properties 頁區裡的 Modifiers 選區，勾選第一個代表抽象的 *abs...* 選項即可。勾取為抽象類別之後，會發現類別圖示處的名稱，自動變成斜體字型，如圖 Q11-23 所示。

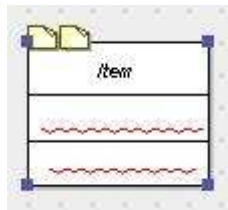


圖 Q11-23: 抽象類別

6. 另一種新增類別的方式，滑鼠左鍵按下類別 Properties 頁區裡的類別按鍵，即可。不過，此處新增的類別並不會出現在類別圖面上，僅會出現於瀏覽器裡，如圖 Q11-24 所示。

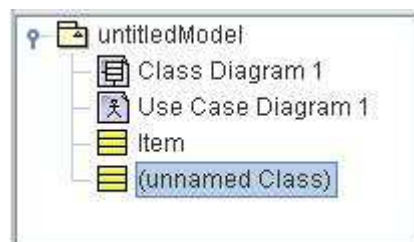


圖 Q11-24: 瀏覽器裡增加了一個無名的類別

7. 回到瀏覽器裡，滑鼠左鍵點選無名的類別，拖曳至類別圖面空白處，放開滑鼠左鍵，圖面上就多了一個無名類別的圖像了。

刪除類別有兩種模式，其一是從類別圖面上移除類別的圖像，另一種是從模型裡刪除類別的實體，如圖 Q11-25 所示。

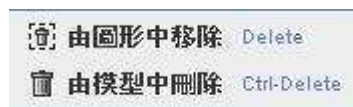


圖 Q11-25: 移除或刪除

由圖形中移除類別圖像的步驟，如下所述：

1. 滑鼠右鍵點選圖內的類別圖示，並於出現的選單中選取【由圖形中移除】，即可。
2. 另一種由圖形中移除類別圖像的方式，滑鼠左鍵點選圖內的類別圖示，再將游標移至第二列選單裡的移除鍵處，按下滑鼠左鍵即可，如圖 Q11-26 所示。



圖 Q11-26: 由圖形中移除的圖示

3. 第三種由圖形中移除類別圖像的方式，滑鼠左鍵點選圖內的類別圖示，再將游標移至第一列選單處，點選【編輯 ► 由圖形中移除】即可。

由模型中刪除類別實體的步驟，如下所述：

1. 滑鼠右鍵點選圖內的類別圖示，並於出現的選單中選取【由模型中刪除】，即可。
2. 另一種由模型中刪除類別實體的方式，滑鼠左鍵點選圖內的類別圖示，再將游標移至第一列選單處，點選【編輯 ► 由模型中刪除】，即可。
3. 第三種由模型中刪除類別實體的方式，滑鼠右鍵單擊選取瀏覽器裡的類別項目，於出現的選單中選取【由模型中刪除】，即可。

Q11.4 新增屬性

現在，我們來新增屬性，操作步驟如下：

1. 點選類別圖面上的類別圖示，將滑鼠游標移動至工具箱處，滑鼠左鍵按下的屬性按鍵後，便新增了一個屬性，如圖 Q11-27 所示。

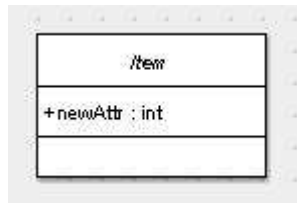


圖 Q11-27: 屬性

2. 於屬性名稱處，雙擊滑鼠左鍵，以便修改屬性能見度、名稱及型別，如圖 Q11-28 所示。ArgoUML 提供四種能見度，分別為：私有(-)、保護(#)、封包(~)和公開(+)

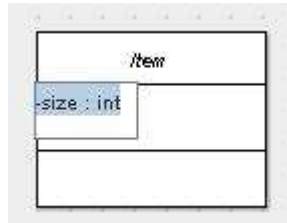


圖 Q11-28: 修改屬性

3. 另一種新增屬性的方式，將滑鼠游標移至類別圖示的第二格屬性區，雙擊滑鼠左鍵就可以新增一項屬性。
4. 第三種新增屬性資料的方式，滑鼠左鍵按下類別 **Properties** 頁區裡的新增屬性按鈕即可，如圖 Q11-29 所示。

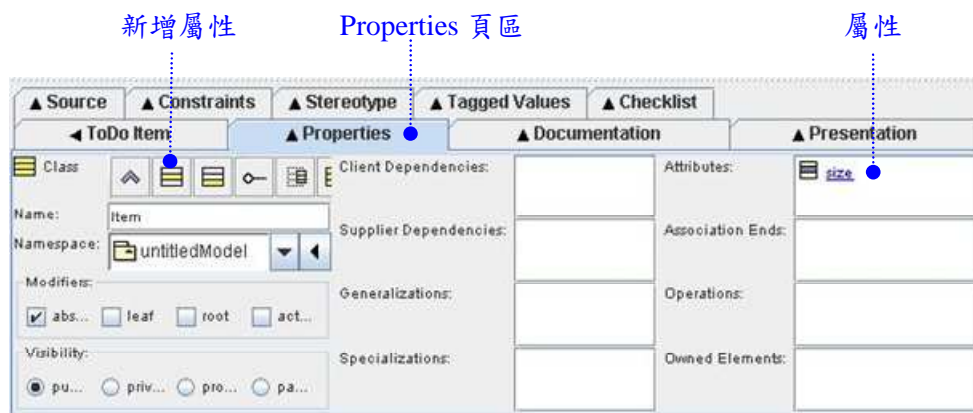


圖 Q11-29: 類別的 Properties 頁區

5. 開啟屬性 **Properties** 頁區也可以修改屬性的所有相關資料，如圖 Q11-30 所示。



圖 Q11-30: 屬性的 Properties 頁區

6. 開啓屬性的 Properties 頁區有三種方法：其一，滑鼠左鍵點選類別圖示中的屬性項目。其二，於類別 Properties 頁區的 Attributes 格處，雙擊滑鼠左鍵。其三，滑鼠左鍵點選瀏覽器裡的類別項目之後，再點選其內的屬性項目，即可開啓屬性的 Properties 頁區，如圖 Q11-31 所示。

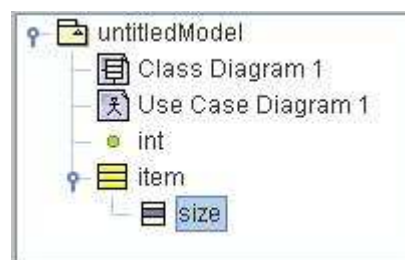


圖 Q11-31: 瀏覽器裡的屬性項目

Q11.5 新增操作

新增操作的步驟，如下所述：

1. 點選類別圖面上的類別圖示，將滑鼠游標移動至工具箱處，滑鼠左鍵按下的操作按鍵後，便新增了一個操作，如圖 Q11-32 所示。

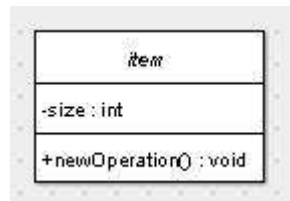


圖 Q11-32: 操作

2. 於操作名稱處，雙擊滑鼠左鍵，以便修改操作能見度、名稱及參數，如圖 Q11-33 所示。ArgoUML 提供四種能見度，分別為：私有(-)、保護(#)、封包(~)和公開(+).

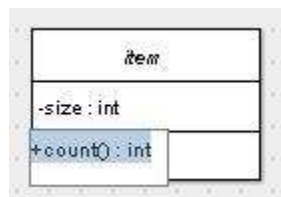


圖 Q11-33: 修改操作

3. 另一種新增操作的方式，將滑鼠游標移至類別圖示的第三格操作區，雙擊滑鼠左鍵就可以新增一項操作。

4. 第三種新增操作資料的方式，滑鼠左鍵按下類別 Properties 頁區裡的新增操作按鈕即可，如圖 Q11-34 所示。

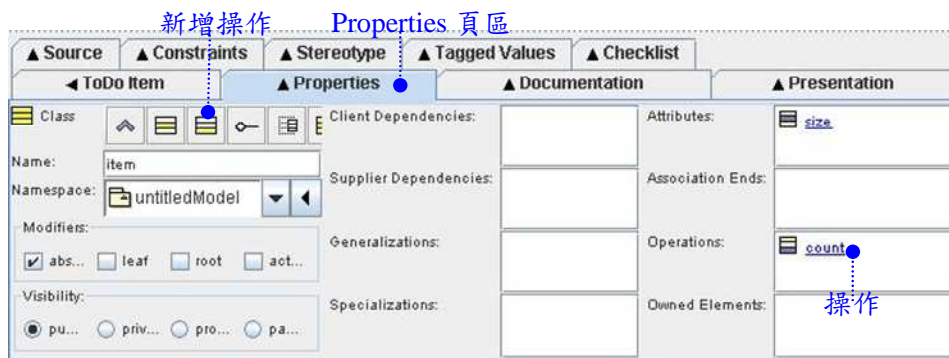


圖 Q11-34: 類別的 Properties 頁區

5. 開啓操作 Properties 頁區也可以修改操作的所有相關資料，如圖 Q11-35 所示。

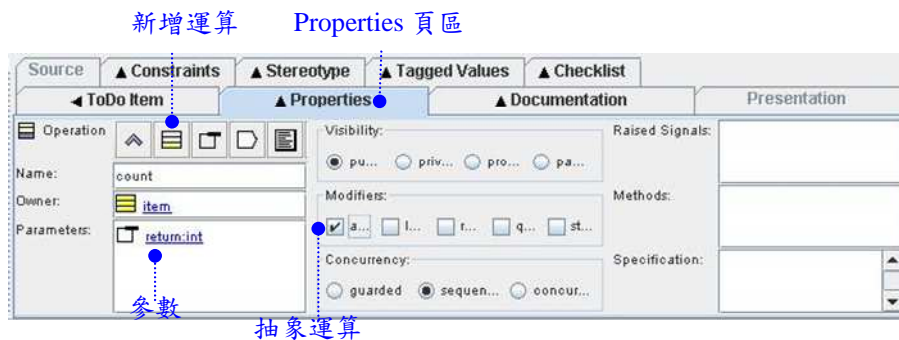


圖 Q11-35: 操作的 Properties 頁區

6. 如果，我們要將這個操作指定為抽象操作的話，請於操作 Properties 頁區裡的 Modifiers 選區，勾選第一個代表抽象的 *abs...* 選項即可。勾取為抽象操作之後，會發現類別圖示處的操作名稱，自動變成斜體字型，如圖 Q11-36 所示。

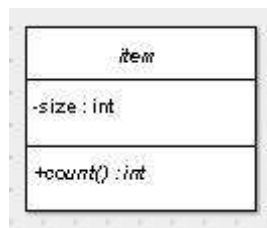


圖 Q11-36: 抽象操作

7. 此外，在操作 Properties 頁區 Parameters 格裡，我們也可以使用滑鼠左鍵雙擊開啓參數 Properties 頁區，修改參數的所有相關資料，如圖 Q11-37 所示。



圖 Q11-37: 操作的 Properties 頁區

8. 開啓操作的 **Properties** 頁區有三種方法：其一，滑鼠左鍵點選類別圖示中的操作項目。其二，於類別 **Properties** 頁區的 **Attributes** 格處，雙擊滑鼠左鍵。其三，滑鼠左鍵點開瀏覽器裡的類別項目之後，再點選其內的操作項目，即可開啓操作的 **Properties** 頁區，如圖 Q11-38 所示。

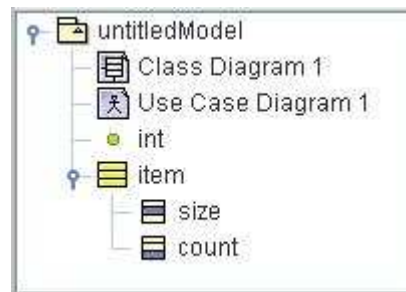


圖 Q11-38: 瀏覽器裡的操作項目

Q11.6 依賴關係

新增依賴關係的步驟，如下所述：

1. 滑鼠左鍵按下工具箱裡的依賴關係按鍵之後，會出現十字型游標。
2. 於類別圖內依賴端類別圖示處，再一次按下滑鼠左鍵不放，然後拖曳至被依賴端類別圖示處放開，便新增了一個依賴關係，如圖 Q11-39 所示。

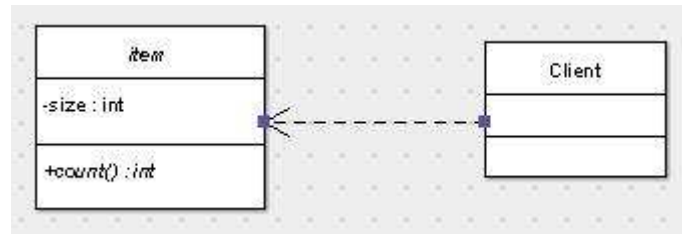


圖 Q11-39: 依賴關係

- 滑鼠左鍵按下依賴關係 Properties 頁區 Stereotype 按鈕，如圖 Q11-40 所示。

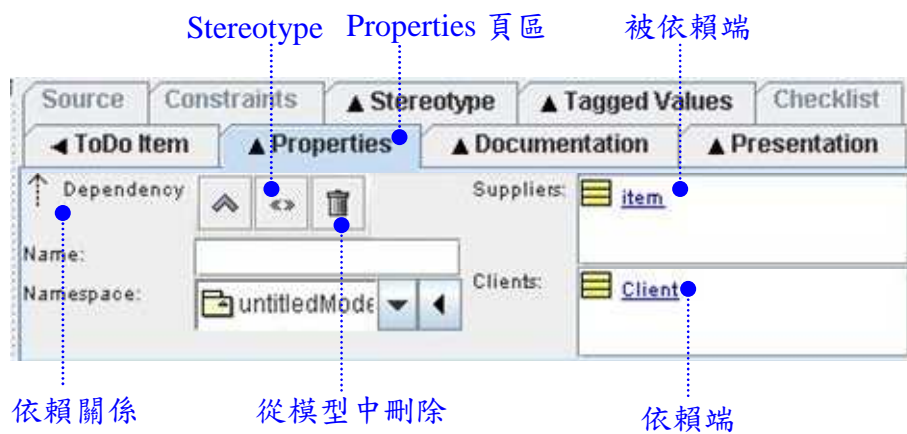


圖 Q11-40: 依賴關係的 Properties 頁區

- 此時，StarUML 會新增一個等待命名的 Stereotype，如圖 Q11-41 所示。

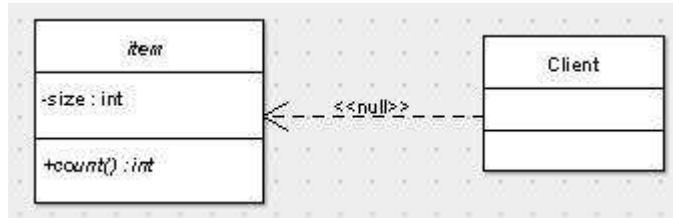


圖 Q11-41: 等待命名的 Stereotype

5. 同時，StarUML 會開啓 Stereotype 的 Properties 頁區，請於 name 格裡填上 call，如圖 Q11-42 所示。

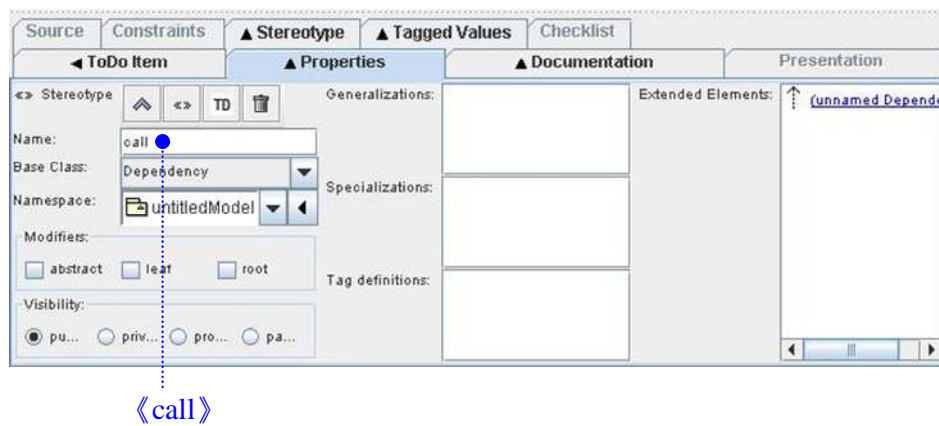


圖 Q11-42: Stereotype 的 Properties 頁區

6. 依賴關係旁標示《call》，代表依賴端物件調用被依賴端物件的操作。當我們新增了一個名為 call 的 Stereotype 之後，依賴關係旁原先標示《null》之處，變成《call》，如圖 Q11-43 所示。

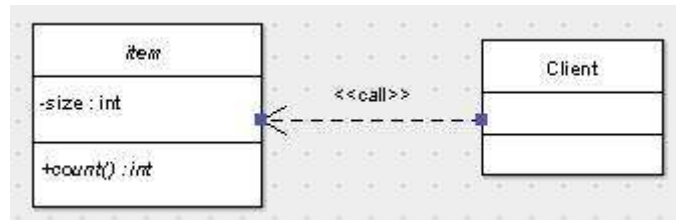


圖 Q11-43: 《call》依賴關係

7. 之後，如果再次用到《call》依賴關係時，使用滑鼠右鍵點取依賴關係的虛線，並於出現的選單中勾取【Apply Stereotypes►《call》】。

Q11.7 一般化關係

新增一般化關係的步驟，如下所述：

1. 滑鼠左鍵按下工具箱裡的一般化關係按鍵之後，會出現十字型游標。
2. 於類別圖內子類別圖示處，再一次按下滑鼠左鍵不放，然後拖曳至父類別圖示處放開，便新增了一個一般化關係，如圖 Q11-44 所示。

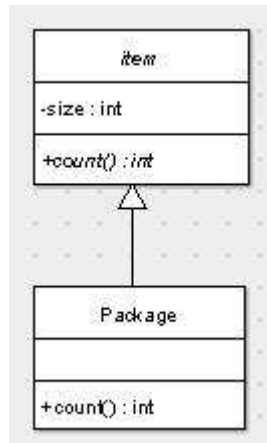
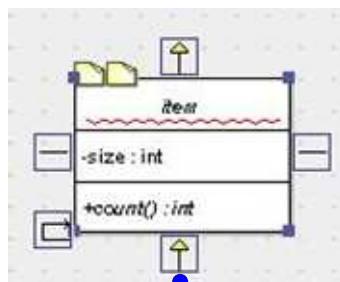


圖 Q11-44: 一般化關係

在新增父類別之後，可以立即新增一般化及子類別，步驟如下：

1. 滑鼠游標移至一父類別圖示處時，圖示下方會出現一個小方框，其內有代表一般化關係的帶三角形實線圖示，如圖 Q11-45 所示。



新增一般化關係與子類別

圖 Q11-45: 小方框裡有代表一般化的圖示

2. 於類別下方的一般化小方框內，按下滑鼠左鍵，便自動於父類別下方新增一般化關係與子類別，如圖 Q11-46 所示。

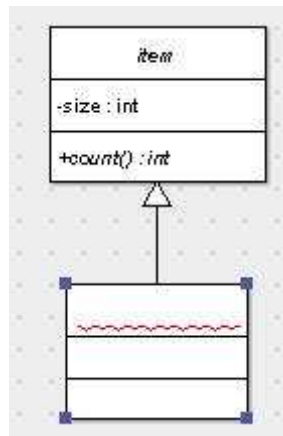


圖 Q11-46: 自動新增一般化關係與子類別

3. 同理，也可以在新增子類別之後，立即新增一般化及父類別。於類別上方的一般化小方框內，按下滑鼠左鍵，便自動於子類別上方新增一般化關係與父類別，如圖 Q11-47 所示。

新增一般化關係與父類別

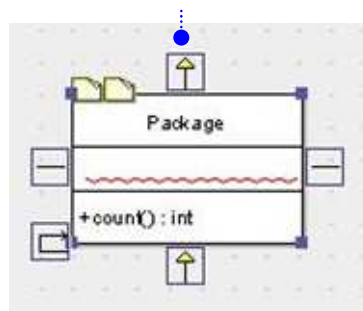


圖 Q11-47: 自動新增一般化關係與父類別

Q11.8 組合關係

新增組合關係的步驟，如下所述：

1. 滑鼠左鍵按下工具箱裡的組合關係按鍵之後，會出現十字型游標。於類別圖內整體類別端圖示處，再一次按下滑鼠左鍵不放，然後拖曳至部分類別端圖示處放開，便新增了一個組合關係，如圖 Q11-48 所示。

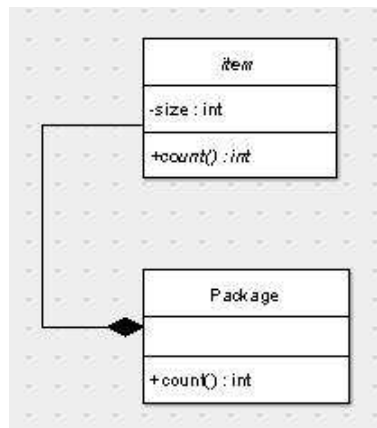


圖 Q11-48: 組合關係

2. 滑鼠右鍵點選任一結合端，會出現【Multiplicity】及【Aggregation】選項，可供選擇。個體數目有四個預設值可選，1、0..1、1..*、0..*。實線的結合端或部分端選【Aggregation►none】，空心菱形聚合端選【Aggregation►aggregate】，實心菱形組合端選【Aggregation►composite】。

3. 如果，需要自訂個體數目時，可以開啓結合端 Properties 頁區，以便進行修改。請先回到結合關係的 Properties 頁區，兩個結合端項目放置於 Connections 格裡，如圖 Q11-49 所示。

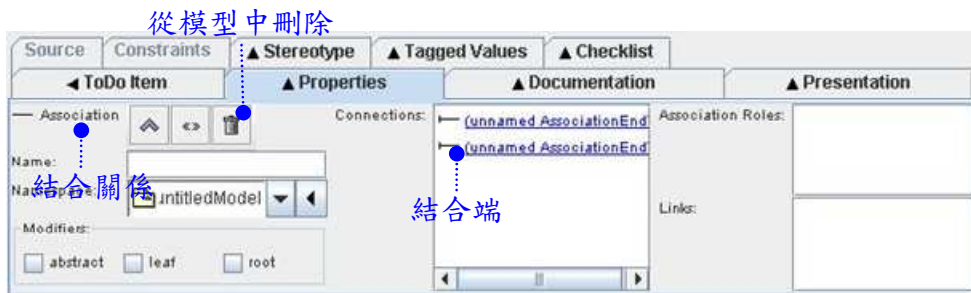


圖 Q11-49: 結合關係的 Properties 頁區

4. 滑鼠左鍵雙擊任一結合端項目，即可開啓結合端的 Properties 頁區，如圖 Q11-50 所示。



圖 Q11-50: 結合端的 Properties 頁區

Q11.9 綜合練習

最後，我們綜合所有的步驟，練習繪製一張類別圖，如下：

1. 打開瀏覽器裡的 `untitledModel` 之後，滑鼠左鍵雙擊其內的類別圖項目，開啓類別圖。
2. 滑鼠左鍵按下工具箱裡的類別按鍵之後，於類別圖任一空白處，再一次按下滑鼠左鍵，新增一個類別。滑鼠左鍵點選圖內的類別圖示，並於三格矩形的第一格處，雙擊滑鼠左鍵，填寫類別名稱 – **Item**。
3. **Item** 類別是個抽象類別，請於類別 **Properties** 頁區裡的 **Modifiers** 選區，勾選第一個代表抽象的 **abs...** 選項。勾取為抽象類別之後，會發現類別圖示處的名稱，自動變成斜體字型。
4. 點選類別圖面上的類別圖示，將滑鼠游標移動至工具箱處，滑鼠左鍵按下的屬性按鍵後，便新增了一個屬性。於屬性名稱處，雙擊滑鼠左鍵，以便修改屬性能見度、名稱及型別。**ArgoUML** 提供四種能見度，分別為：私有(-)、保護(#)、封包(~)和公開(+).
5. 點選類別圖面上的類別圖示，將滑鼠游標移動至工具箱處，滑鼠左鍵按下的操作按鍵後，便新增了一個操作。於操作名稱處，雙擊滑鼠左鍵，以便修改操作的能見度、名稱及參數型別。
6. 我們要將這個操作指定為抽象操作，所以請於操作 **Properties** 頁區裡的 **Modifiers** 選區，勾選第一個代表抽象的 **abs...** 選項。勾取為抽象操作之後，會發現類別圖示處的操作名稱，自動變成斜體字型。
7. 滑鼠游標移至 **Item** 類別圖示處時，圖示下方會出現一個小方框，其內有代表一般化關係的帶三角形實線圖示。

8. 於類別下方的一般化小方框內，按下滑鼠左鍵，便自動於 **Item** 類別下方新增一般化關係與子類別。請為子類別命名為 **Package**，並且新增一項重新定義繼承而來的 **count** 操作。
9. 再新增另一個名為 **File** 的子類別，並且使用滑鼠左鍵按下工具箱裡的一般化關係按鍵之後，於類別圖內 **File** 子類別圖示處，再一次按下滑鼠左鍵不放，然後拖曳至 **Item** 父類別圖示處放開，再度新增一個一般化關係。
10. 再新增另一個名為 **Client** 的類別，並且使用滑鼠左鍵按下工具箱裡的依賴關係按鍵之後，於類別圖內 **File** 子類別圖示處，再一次按下滑鼠左鍵不放，然後拖曳至 **Item** 類別圖示處放開，新增一個依賴關係。
11. 滑鼠左鍵按下依賴關係 **Properties** 頁區 **Stereotype** 按鍵，StarUML 會開啓 **Stereotype** 的 **Properties** 頁區，請於 **name** 格裡填上 **call**。當我們新增了一個名為 **call** 的 **Stereotype** 之後，依賴關係旁就標示了《**call**》。
12. 滑鼠左鍵按下工具箱裡的組合關係按鍵之後，會出現十字型游標。於類別圖內組合端的 **Package** 類別圖示處，再一次按下滑鼠左鍵不放，然後拖曳至部分端的 **Item** 類別圖示處放開，便新增了一個組合關係。
13. 滑鼠右鍵點選任一結合端，會出現【**Multiplicity**】選項。選擇了個體數目之後，整張圖就完成了。
14. 如果想要單獨匯出這張類別圖，可將游標移至第一列選單處，點選【檔案►儲存圖像】，即可匯出 GIF 格式的影像檔。

Q12 Case Complete 之簡介與操作示範



Q12.1	試用工具—Case Complete ----	34
Q12.2	定義參與者 ----	40
Q12.3	定義參與者之目的 ----	43
Q12.4	定義使用案例 ----	48
Q12.5	增加欄位 ----	53
Q12.6	設定參照附件 ----	54
Q12.7	產生報表 ----	55

在撰寫「UML入門」一書時，上網搜尋了許多UML工具；其中這一款工具相當獨特，因它專門支援使用案例模式。美中不足的是，它既無免費的版本，而且我所使用的試用版也沒有支援中文。但是，還是想為您介紹這款工具，感受一下它對使用案例模式的專業支援。

Q12.1 試用工具—Case Complete

Case Complete 是一款使用案例模式(Use Case Model)專用的開發工具，由美國公司 Serlio 所提供，可直接連上這家公司的網站 (<http://www.serlio.com/>) 下載 30 天試用版。Case Complete 需要用到 Microsoft .NET Framework 1.1 或 2.0 版，安裝前請先行確認您的主機上頭已經備妥所需環境，請看圖 Q12-1 是 Case Complete 2.5 試用版的起始畫面。

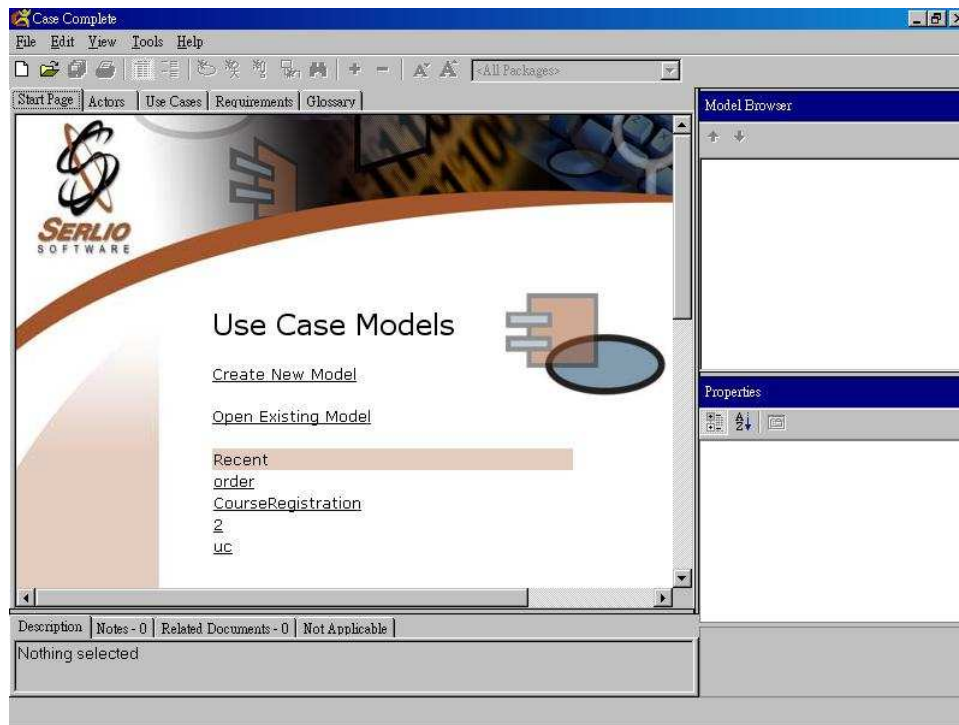


圖 Q12-1: Case Complete 的起始畫面

由於，Case Complete 是一款使用案例模式專用的開發工具，對使用案例模式的支援相當細緻，這是其它支援 UML 工具所無法比擬的。美中不足的是，Case Complete 沒有免費的公眾版或個人版，同時也無法支援中文，不過我們還是想要展示它的試用版，藉此讓您看到工具如何貼切地支援我們建立使用案例模式。

一旦，我們選擇建立一個新的使用案例模式之後，Case Complete 自動開啓起始頁(Start Page)，如圖 Q12-2 所示。

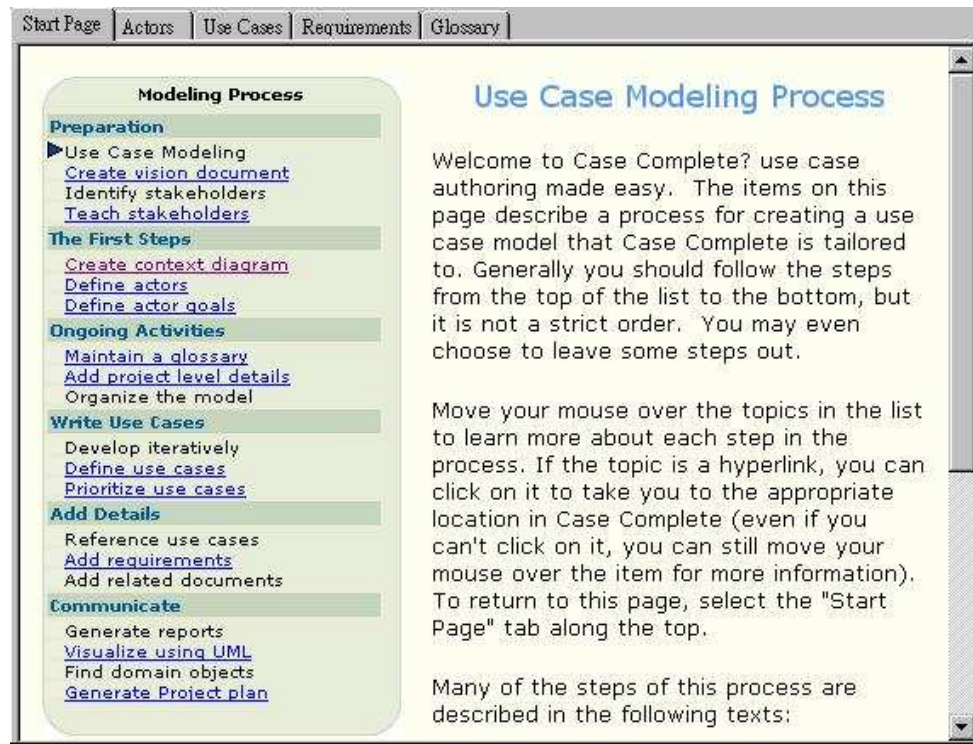


圖 Q12-2: Case Complete 的建議程序

起始頁是一項非常貼心的設計，其內提出了建立使用案例模式的建議程序，不僅讓使用者可以按部就班地使用 Case Complete，同時也讓初學者有機會學習一套開發程序。在起始頁裡，除了條列每道開發步驟且詳加說明之外，還可以使用滑鼠左鍵雙擊任何一道步驟，Case Complete 會自動連結到該步驟相關的頁面，方便使用者作業。

請看圖 Q12-3，Case Complete 提供詳盡的使用案例細頁，讓我們可以盡情填寫使用案例敘述。

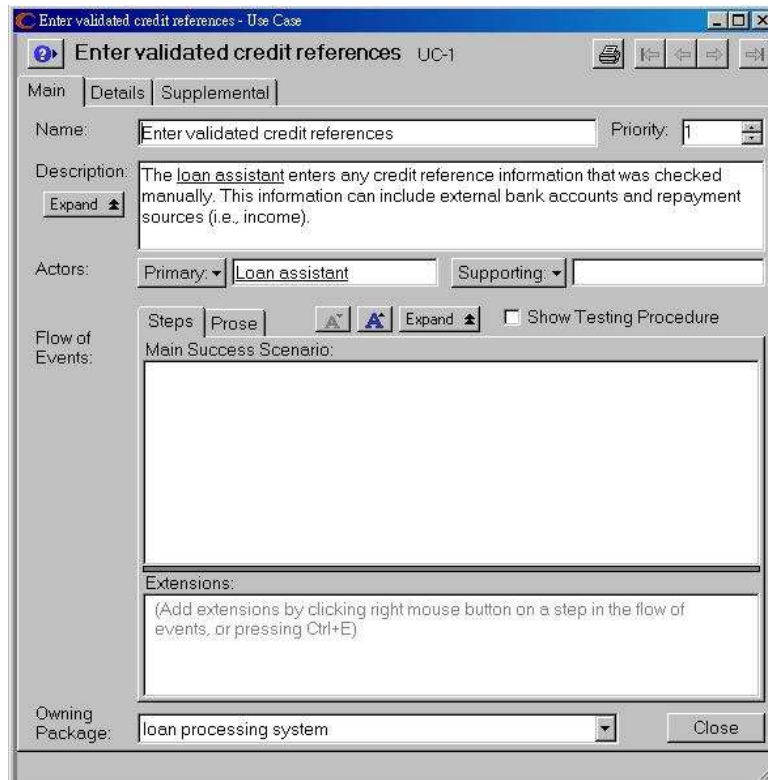


圖 Q12-3: 使用案例細頁

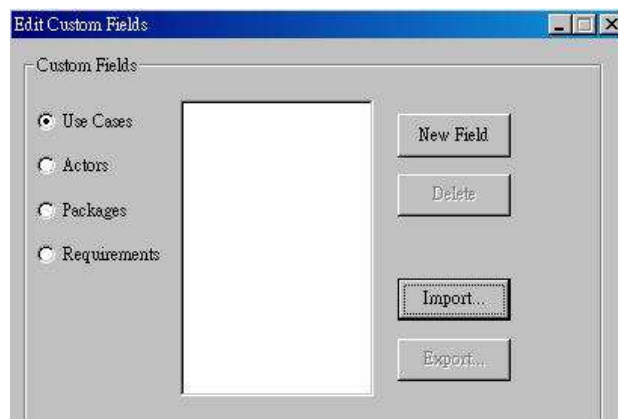


圖 Q12-4: 編輯客製欄位

此外，Case Complete 還能夠讓我們自行增加欄位，以便填寫特殊的使用案例敘述資料，如圖 Q12-4 所示。最後，Case Complete 還提供自動產生 HTML 或 Word 兩種格式報表的功能，如圖 Q12-5 所示。

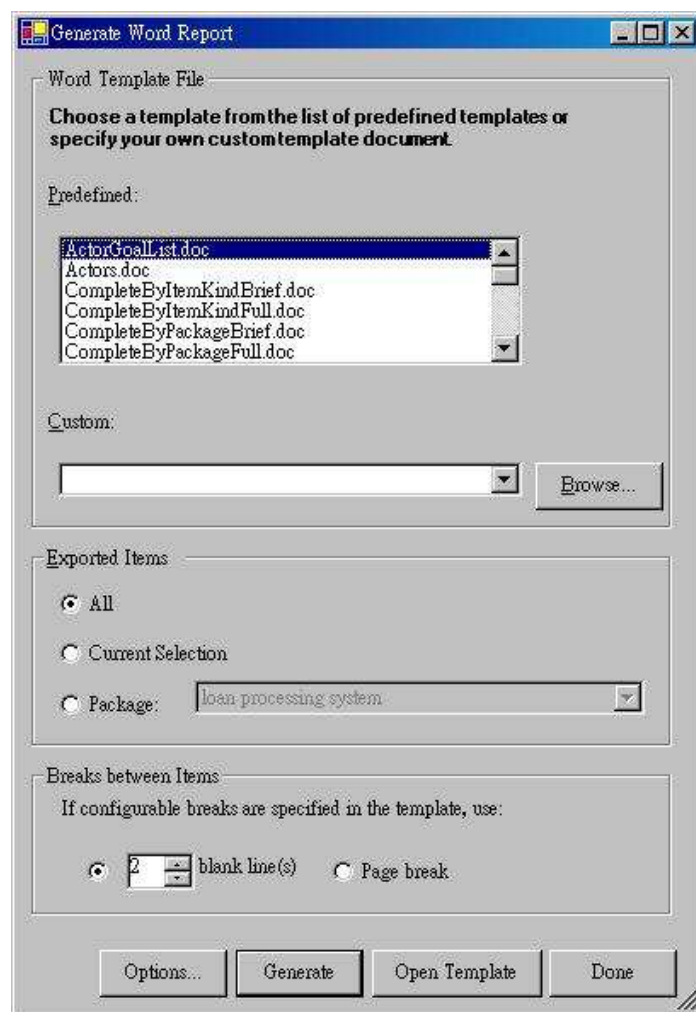
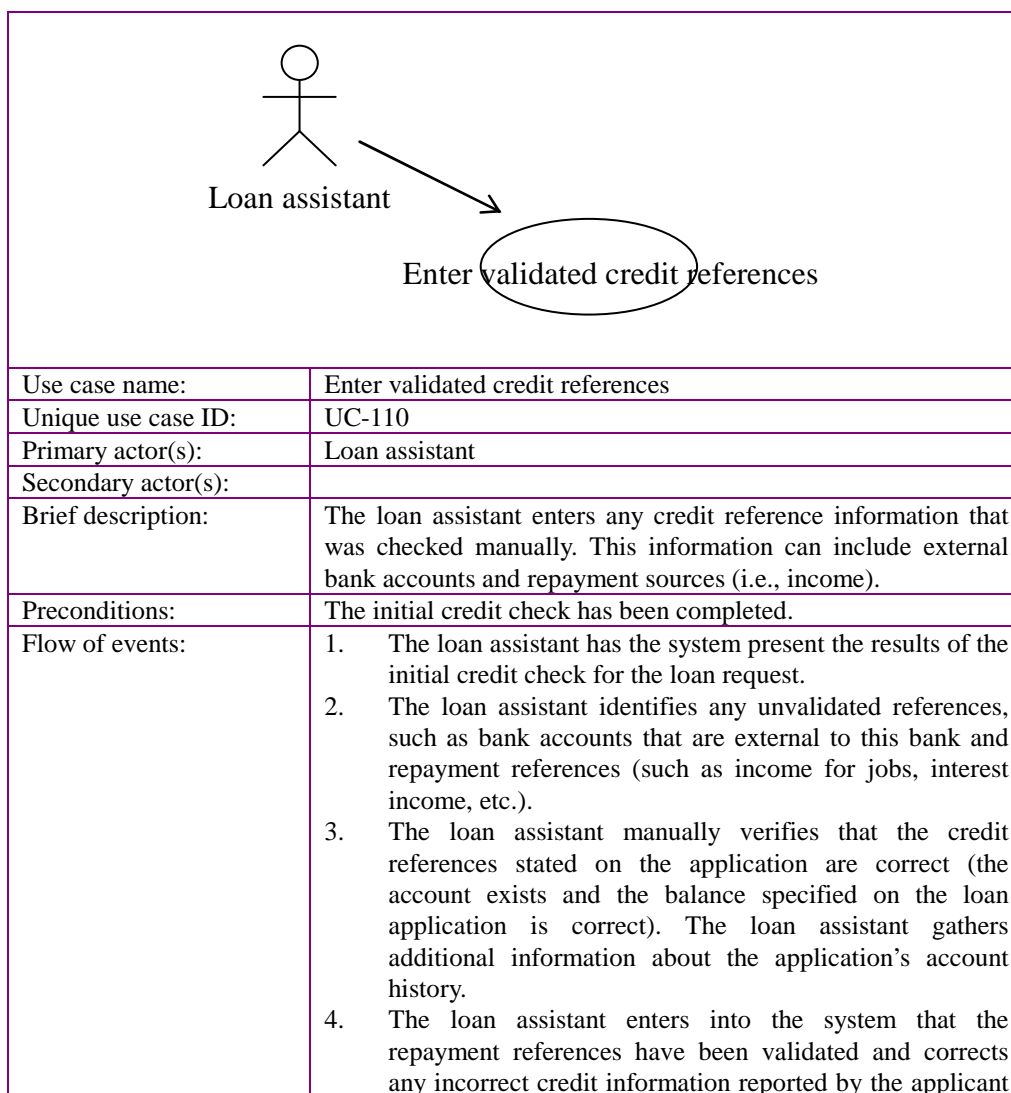


圖 Q12-5: 自動產生報表

後續內容裡，我們會挑選幾道重要的步驟介紹給您。由於，Case Complete 不支援中文，所以我們節錄“Advanced Use Case Modeling”書中 Loan Processing System 裡的 Enter validated credit references 用例來做示範。



	<p>on the application.</p> <ol style="list-style-type: none"> 5. The loan assistant enters into the system that the repayment references stated on the application are correct and that the amount of repayment income and time on the job have been accurately reported. 6. The loan assistant enters into the system that the repayment references have validated and corrects any incorrect repayment information on the loan request. 7. The loan assistant marks the loan request “Loan references validated.” 8. The system routes the loan request to a loan officer for final evaluation.
Postconditions:	The loan references have been validated.
Priority:	High
Alternative flows and exceptions:	<ul style="list-style-type: none"> ● The loan references cannot be validated and the loan request processing is suspended until the applicant can be informed of this fact and provides correct references. ● The loan references provided significantly contradict information on credit accounts or repayment sources and the automated credit score needs to be recalculated by the system.
Nonbehavioral requirements:	<ul style="list-style-type: none"> ● Only the loan officer and loan assistant should have access to credit information. ● The system should support 10 concurrent loan assistants entering reference information at one time.
Assumptions:	None
Issues:	Differences between the reported references and actual references trigger a new calculation of the credit score.
Source:	Meeting with loan assistants, interview memo 2348.

Q12.2 定義參與者

我們跳過了幾道步驟，直接進行定義參與者的步驟，如下：

1. 滑鼠左鍵選取起始頁裡的 Define actors 之後，連結進入參與者頁面(Actor Page)，如圖 Q12-6 所示。



圖 Q12-6: 選取 Define actors

2. 如果不清楚從何處下手尋找參與者，可回到起始頁，滑鼠左鍵單擊起始頁裡 Define Actors 說明細節的 Click here 處，如圖 Q12-7 所示。



圖 Q12-7: Define Actors 說明細節

3. Case Complete 貼心備妥了參與者尋找指南，如圖 Q12-8 所示。

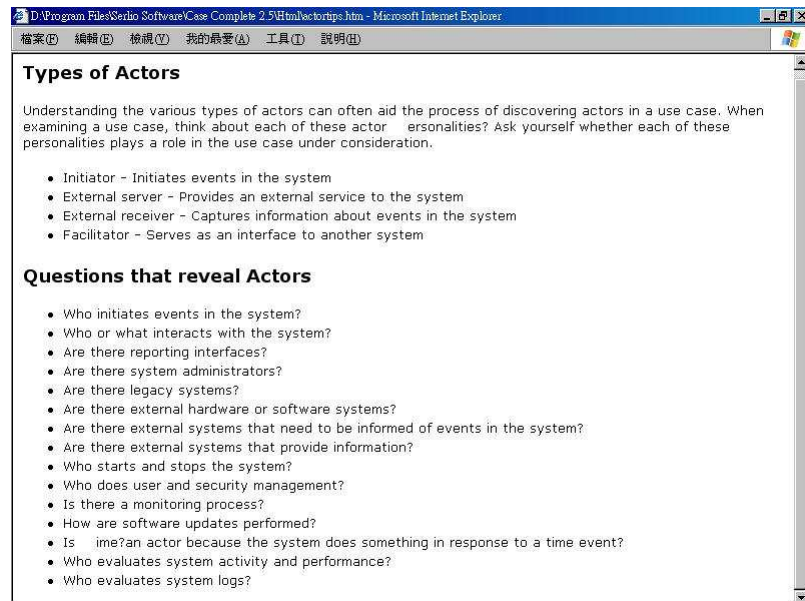



圖 Q12-8: 參與者尋找指南

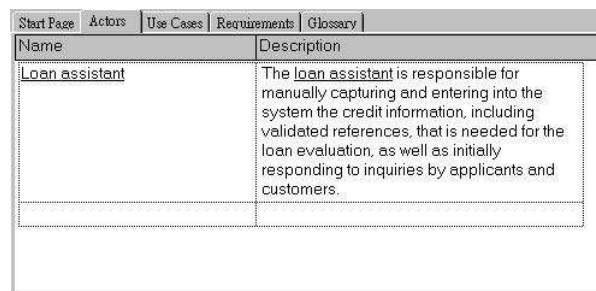
4. 連結進入參與者頁區(Actor Page)，如圖 Q12-9 所示。



Name	Description
Click here to add new items	

圖 Q12-9: 參與者頁區

5. 滑鼠左鍵點選參與者頁區裡的指示文字處，便可以填寫參與者名稱及說明了，如圖 Q12-10 所示。



Name	Description
Loan assistant	The loan assistant is responsible for manually capturing and entering into the system the credit information, including validated references, that is needed for the loan evaluation, as well as initially responding to inquiries by applicants and customers.

圖 Q12-10: 填寫參與者名稱及說明

Q12.3 定義參與者之目的

一般而言，參與者使用系統有其目的，所以我們可以在定義參與者的目的之後，自動轉出相對應的使用案例，步驟如下：

1. 滑鼠左鍵選取起始頁裡的 Define actor goals 之後，連結進入參與者頁區。Case Complete 改變了參與者頁區的觀看模式，不同於

定義參與者時所呈現的資料，如圖 Q12-11 所示。

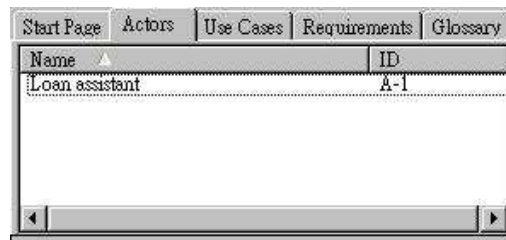


圖 Q12-11: 參與者頁區

2. 滑鼠左鍵雙擊參與者頁區裡的參與者名稱處，便可以順利開啓參與者細頁，如圖 Q12-12 所示。

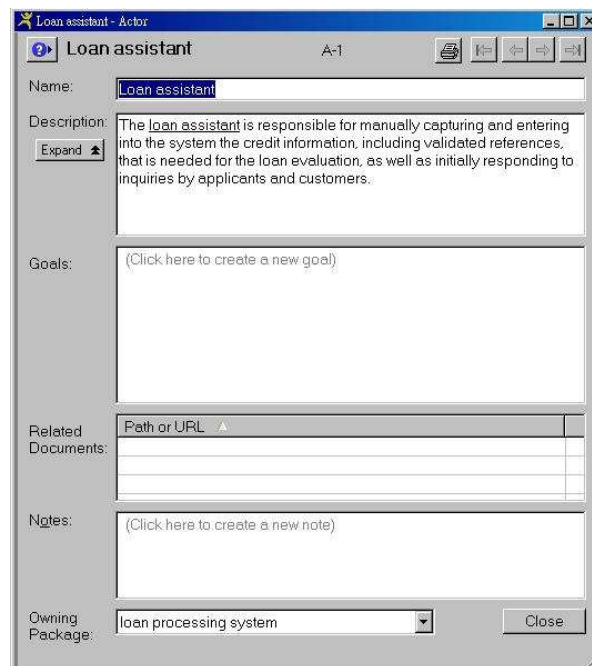


圖 Q12-12: 參與者細頁

3. 滑鼠左鍵點選參與者細頁 Goal 空格裡的指示文字處，便可以填寫參與者的目的了，如圖 Q12-13 所示。



圖 Q12-13: 參與者的目的

4. 填寫完畢，請按下 Close 鍵，關閉參與者細頁。回到起始頁，滑鼠左鍵單擊起始頁裡 Define Actor Goals 說明細節的 generate your initial use cases directly from those goals 處，如圖 Q12-14 所示。

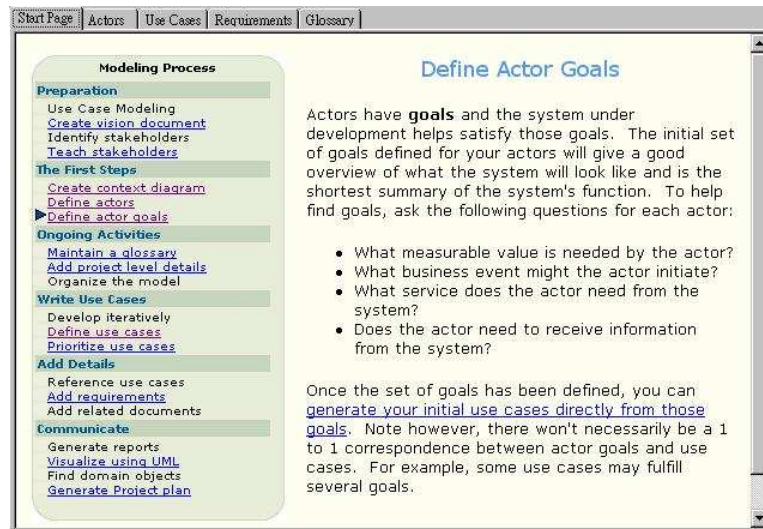


圖 Q12-14: 目的轉使用案例

5. 此時，Case Complete 會自動為每個目的對應一個使用案例，並且開啓如圖 Q12-15 的選取視窗。請取消 inquire applicants 和 inquire customers 的目的選項，僅留下 enter credit information 目的，並且將欲轉出之使用案例名稱更動為 - Enter validated credit references。

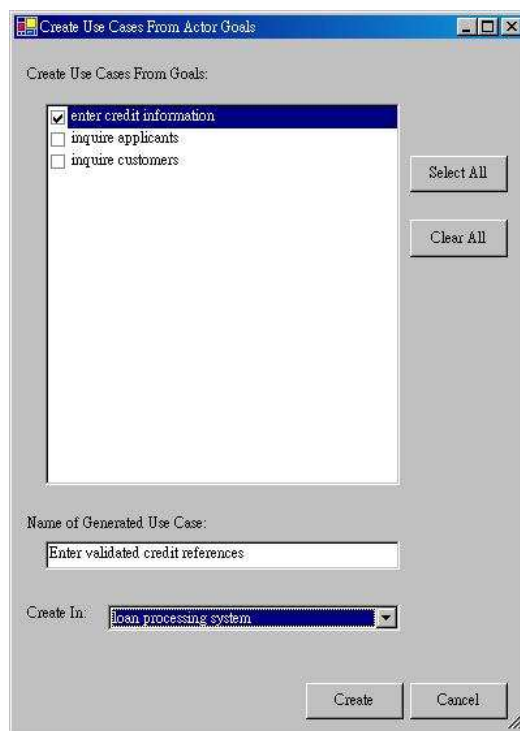


圖 Q12-15: 目的轉使用案例的選取視窗

6. 選取並設定好欲轉出的使用案例之後，請按下 Create 鍵。這時圖 Q12-15 會變成圖 Q12-16 的樣子，原先的 enter credit information 目的旁已經被編號為 UC-1 了。

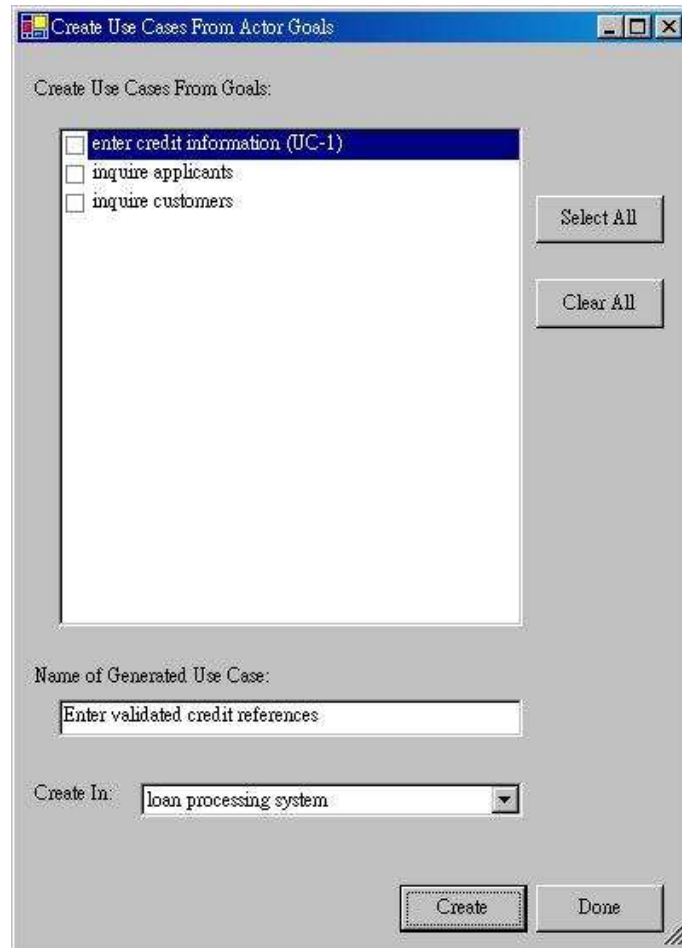
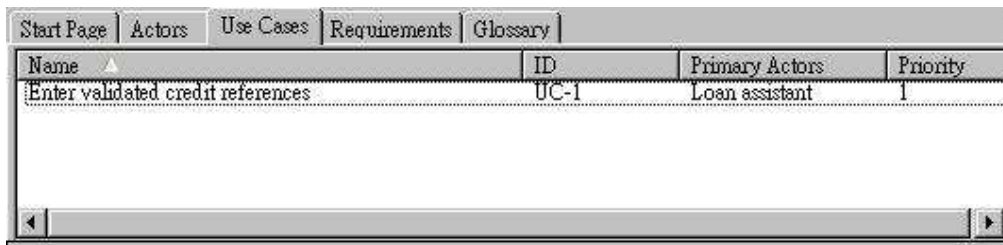


圖 Q12-16: 目的已經被編號為 UC-1 了

7. 按下圖 Q12-16 裡的 Done 鍵之後，自動開啓的使用案例頁區裡已經新增了一個名為 Enter validated credit references 的使用案例，並且編號為 UC-1，如圖 Q12-17 所示。



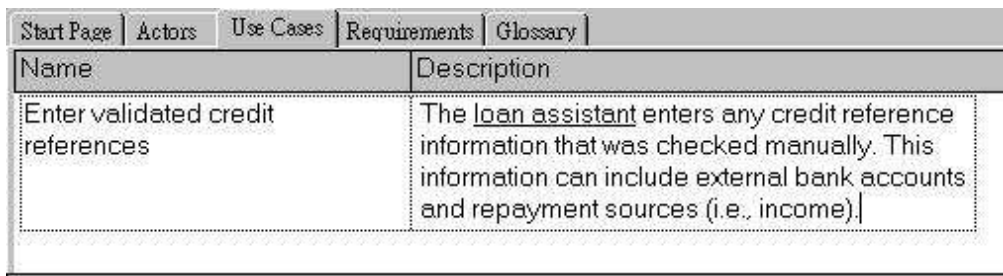
Name	ID	Primary Actors	Priority
Enter validated credit references	UC-1	Loan assistant	1

圖 Q12-17: 自動產生編號 UC-1 的使用案例

Q12.4 定義使用案例

自動轉出使用案例之後，我們就可以開始填寫使用案例敘述了，步驟如下：

1. 滑鼠左鍵選取起始頁裡的 Define use cases 之後，連結進入使用案例頁區。Case Complete 已經備妥了 Enter validated credit references 使用案例，留下 Description 空格讓我們填寫使用案例簡述，如圖 Q12-18 所示。



Name	Description
Enter validated credit references	The <u>loan assistant</u> enters any credit reference information that was checked manually. This information can include external bank accounts and repayment sources (i.e., income).

圖 Q12-18: 填寫使用案例簡述

2. 滑鼠左鍵雙擊模式瀏覽器(Model Browser)裡的 Enter validated credit references 使用案例，如圖 Q12-19 所示。

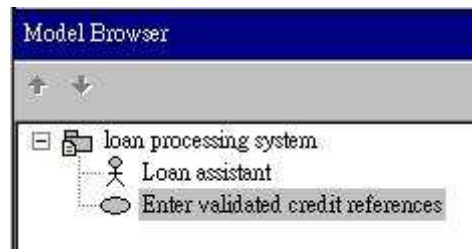


圖 Q12-19: 模式瀏覽器

3. 此時，便可以順利開啓使用案例細頁，盡情填寫使用案例敘述了，如圖 Q12-20 所示。

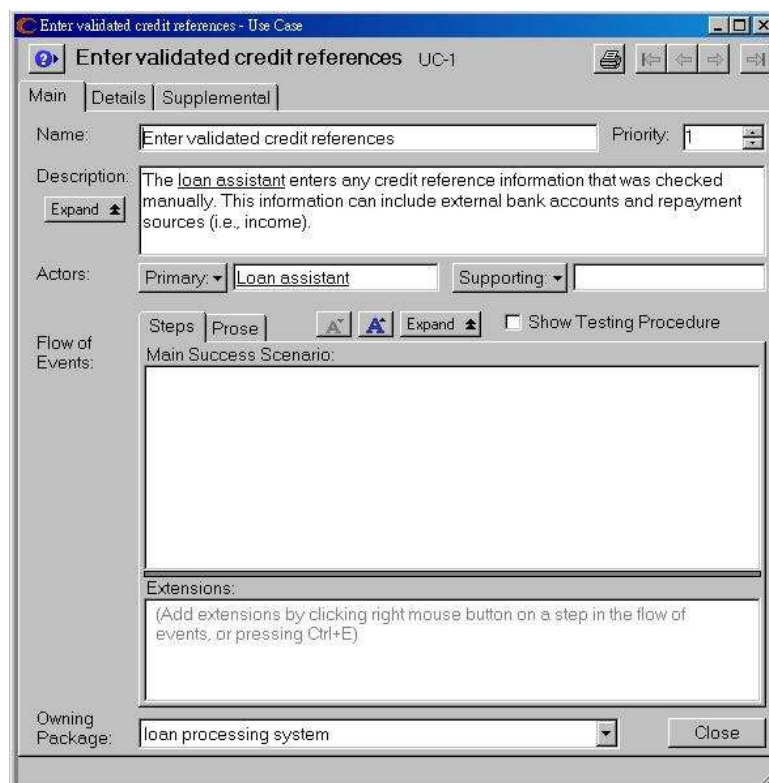


圖 Q12-20: 使用案例細頁

4. 請於 Flow of Events 的 Steps 頁區填寫使用案例敘述的主要程序，如圖 Q12-21 所示。

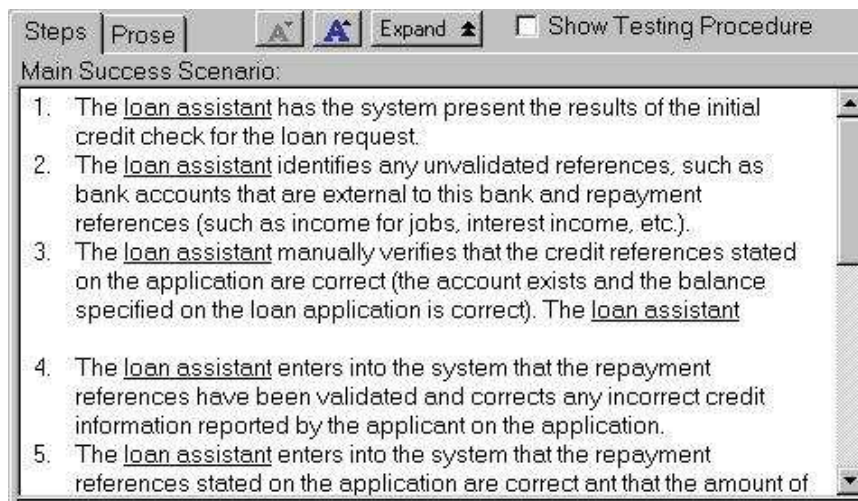


圖 Q12-21: 填寫主要程序

5. 填寫完使用案例敘述的主要程序之後，請於 Steps 頁區的空白處按下滑鼠右鍵，並於出現的選單中選取 Add Anytime Extension 選項，以便於 Extension 空格處新增替代程序，如圖 Q12-22 所示。

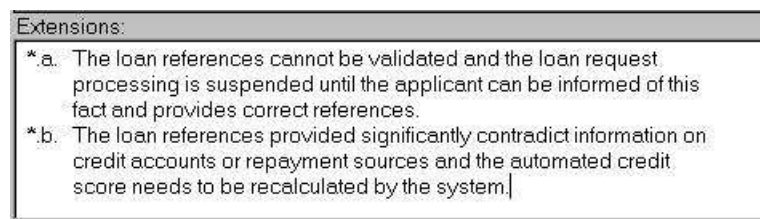


圖 Q12-22: 填寫替代程序

6. 滑鼠左鍵點選開啓使用案例細頁 Detail 次頁，如圖 Q12-23 所示。

Enter validated credit references - Use Case UC-1

Enter validated credit references UC-1

Main Details Supplemental

Preconditions:

Success Guarantees:

Level:

Complexity:

Use Case Status:

Implementation Status:

Assigned To:

Release:

Referenced Requirements:

Requirement	ID	Type	Priority
-------------	----	------	----------

Select...

(Double click on a requirement to view its details)

Close

圖 Q12-23: Detail 次頁

7. 請於 Preconditions 空格處填寫使用案例敘述的執行前要件，如圖 Q12-24 所示。

Preconditions: The initial credit check has been completed.

圖 Q12-24: 填寫執行前要件

8. 滑鼠左鍵點選開啓使用案例細頁 Supplemental 次頁，如圖 Q12-25 所示。

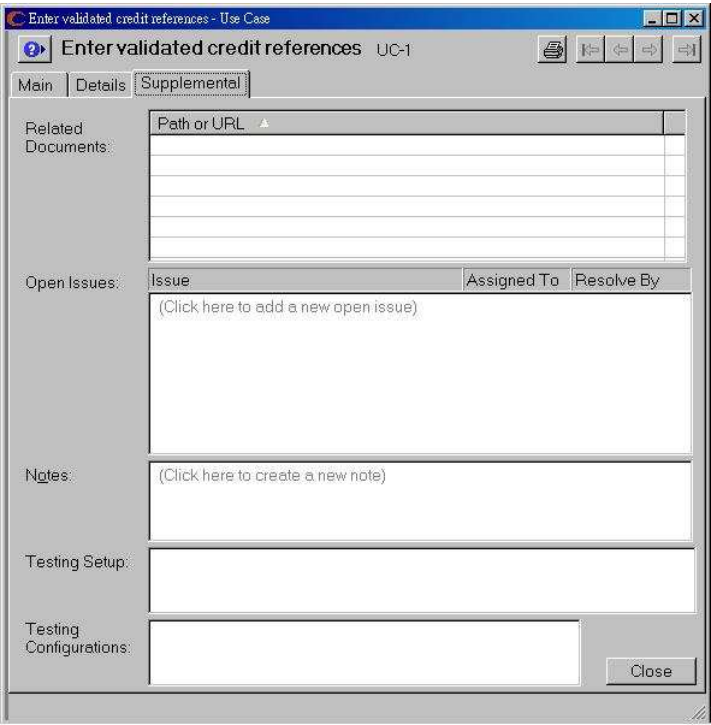


圖 Q12-25: Supplemental 次頁

9. 請於 Open Issues 空格處填寫使用案例敘述的待解議題，如圖 Q12-26 所示。

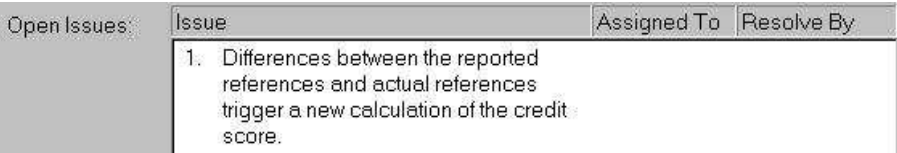


圖 Q12-26: 填寫待解議題

Q12.5 增加欄位

在 Enter validated credit references 使用案例敘述裡，還有 Postcoditons、Nonbehavioral Requirements、Assumptions、Source 這四個欄位待填。雖然，Case Complete 沒有提供相對應的欄位，所以我們得自行增加欄位，其步驟如下：

1. 滑鼠右鍵點選第一列選單【Tools►Custom Fields】，開啓如圖 Q12-27 的編輯客製欄位(Edit Custom Fields)視窗。

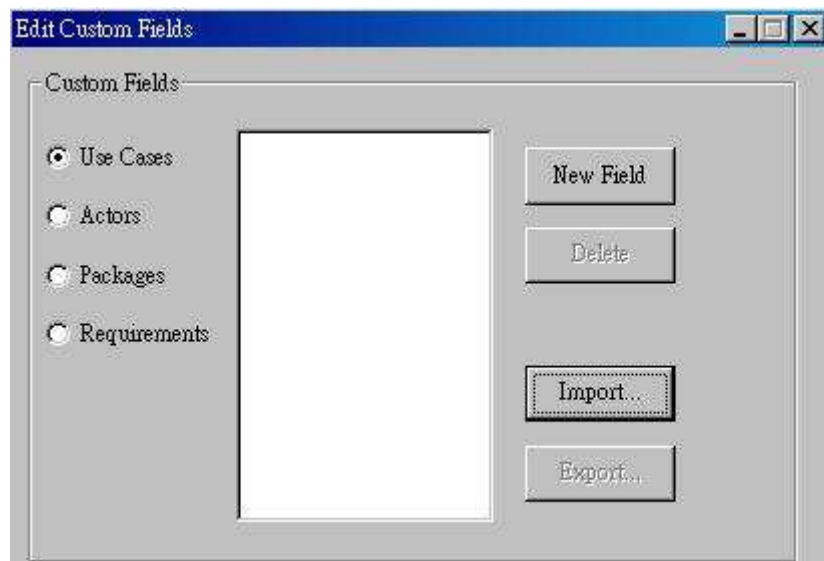


圖 Q12-27: 編輯客製欄位

2. 按下 New Field 鍵，並且更名，新增四個欄位，如圖 Q12-28 所示。

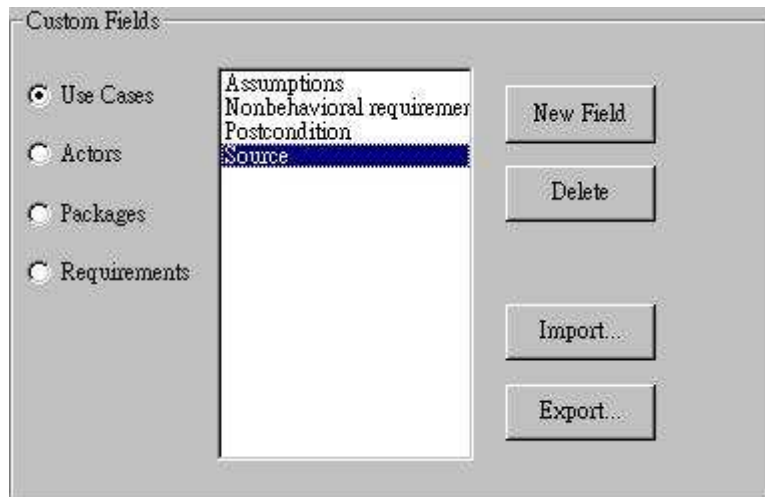


圖 Q12-28: 新增欄位

- 滑鼠左鍵雙擊模式瀏覽器裡的 Enter validated credit references 使用案例，開啓使用案例細頁 Detail 次頁，可以發現其內多了 Custom Fields 區，並且新增了四個欄位，如圖 Q12-29 所示。

Custom Fields	
Assumptions	
Nonbehavioral requirements	
Postcondition	
Source	

圖 Q12-29: Custom Fields 區

Q12.6 設定參照附件

使用案例可以做爲組織中心，於使用案例敘述中連結相關的附件，比方說 UML 圖檔、會議紀錄、XSD 文檔、表格設計等等。一旦，流程或需求有所變動時，可以快速搜尋出以使用案例爲首的一連串參照附件，進行

修改。設定參照附件的步驟如下：

1. 滑鼠左鍵雙擊模式瀏覽器裡的 Enter validated credit references 使用案例，開啓使用案例細頁 Supplemental 次頁，看到裡面有一個 Related Documents 區。
2. 請於 Related Documents 區空白處按下滑鼠右鍵，點選【Add Related Document】選項，加入參照附件，如圖 Q12-30 所示。



圖 Q12-30: Related Documents 區

3. 隨後，於附件項目處雙擊滑鼠左鍵，即可開啓參照附件。

Q12.7 產生報表

最後，Case Complete 還提供自動產生 HTML 或 Word 兩種格式報表的功能，如下：

1. 滑鼠右鍵點選第一列選單【Tools►Generate Word Report】，開啓如圖 Q12-31 的報表產生視窗。

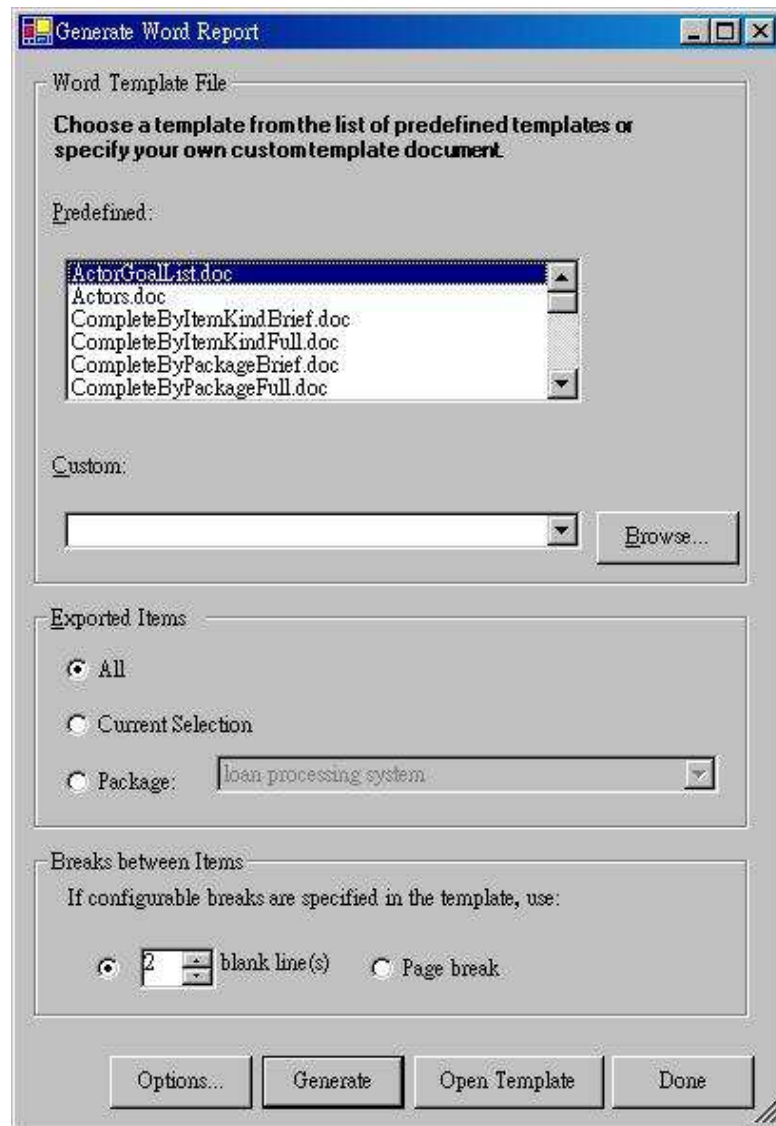


圖 Q12-31: 自動產生報表

2. 選擇產生其中的 UseCases.doc，並且按下 Generate 鍵之後，自動產出 Word 報表，主要內容如下：

Details	
Parent: loan processing system Primary Actors: Loan assistant Preconditions: The initial credit check has been completed. Level: Use Case Status: Assigned To:	
Supporting Actors: Success Guarantee: Complexity: Implementation Status: Release:	
Custom Fields	Value
Assumptions	None
Nonbehavioral requirements	Only the loan officer and loan assistant should have access to credit information. The system should support 10 concurrent loan assistants entering reference information at one time.
Postcondition	The loan references have been validated.
Source	Meeting with loan assistants, interview memo 2348.
Flow of Events	
Main Success Scenario: <ol style="list-style-type: none"> 1. The loan assistant has the system present the results of the initial credit check for the loan request. 2. The loan assistant identifies any unvalidated references, such as bank accounts that are external to this bank and repayment references (such as income for jobs, interest income, etc.). 3. The loan assistant manually verifies that the credit references stated on the application are correct (the account exists and the balance specified on the loan application is correct). The loan assistant gathers additional information about the application's account history. 4. The loan assistant enters into the system that the repayment references have been validated and corrects any incorrect credit information reported by the applicant on the application. 5. The loan assistant enters into the system that the repayment references stated on the application are correct and that the amount of repayment income and time on the job have been accurately reported. 6. The loan assistant enters into the system that the repayment references have validated and corrects any incorrect repayment information on the loan request. 7. The loan assistant marks the loan request "Loan references validated." 8. The system routes the loan request to a loan officer for final evaluation. 	

Flow of Events		
Extensions: *.a The loan references cannot be validated and the loan request processing is suspended until the applicant can be informed of this fact and provides correct references. *.b The loan references provided significantly contradict information on credit accounts or repayment sources and the automated credit score needs to be recalculated by the system.		
Open Issues	Assigned To	Resolve By
Differences between the reported references and actual references trigger a new calculation of the credit score.		
Related Documents		
C:\Documents and Settings\Administrator\CHIU-PC\桌面\Temp\loan processing system.uml		

Q13 有關於 actor 與 extend 的疑問？

有關於actor與extend的疑問？

(Jerry於UML Blog之提問)

邱老師：

有個關於use case的小問題想請教妳一下。我畫的圖裡頭描述的大概是「所有人都能查詢和瀏覽CD，只有會員才能購買CD」。

圖Q13-1裡的「買CD」有畫上《extend》，說明會員需要登入後才可以購買CD。

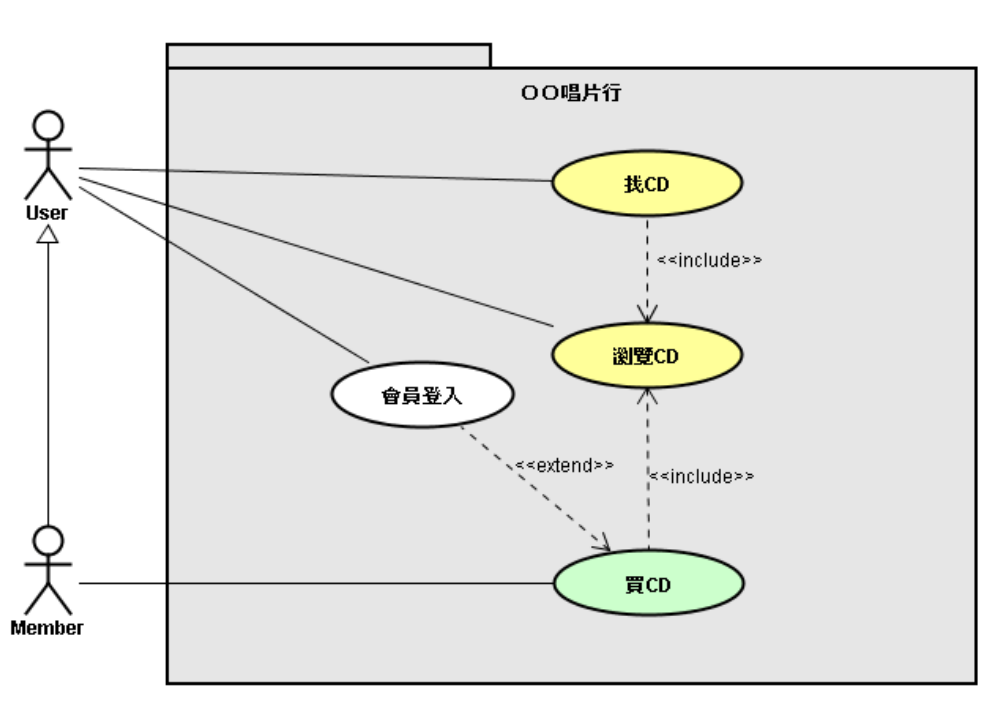


圖 Q13-1

圖Q13-2裡單純把會員actor只畫向「買CD」，說明的是當一般User登入後角色的狀態就已經變成會員，可以操作「買CD」的use case了。

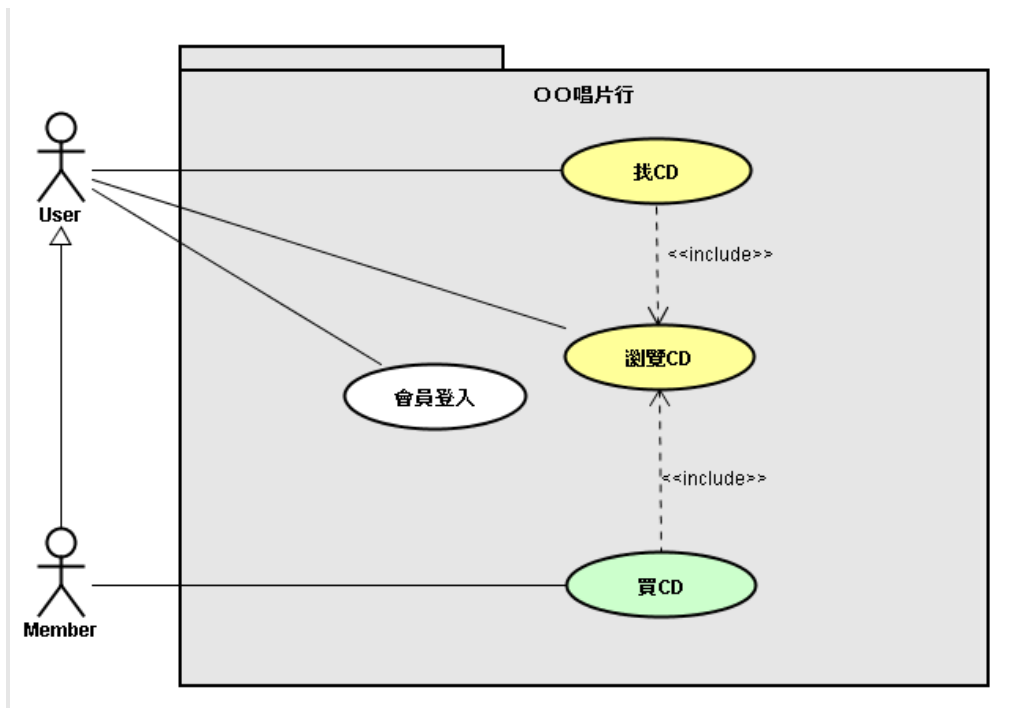


圖 Q13-2

在這邊想要問的是圖Q13-2的這種畫法是合理的嗎？還是圖Q13-1的畫法比較適當？

二擇一的話，我會偏好圖 Q13-2 的畫法，因為我信奉「少即是多」；如果圖面上少一個圖示也不影響溝通的話，那我會寧可把圖面空間留給非要突顯不可的重點。

除非您是要透過這個例子學習擴充關係及包含關係，否則實務上，我會更建議採用圖 Q13-3 的畫法，去掉關係線。不知您是否有發現，即便是圖 Q13-3 的畫法，溝通上的效果應該跟圖 Q13-2 是一樣的，但是較為簡潔易懂。當然，我得再次強調，除非您是特意使用關係線的，那當然另當別

論。

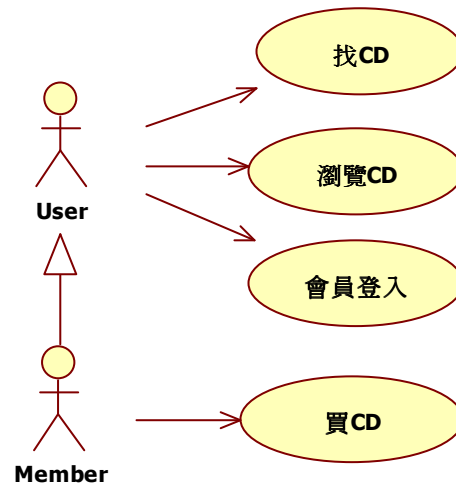


圖 Q13-3: 去掉使用案例之間的關係線

同理，除非是特別想突顯參與者之間的一般化關係，否則的話，我會更樂意採用圖 Q13-4 的，理由同上，少即是多，簡潔至上啊！

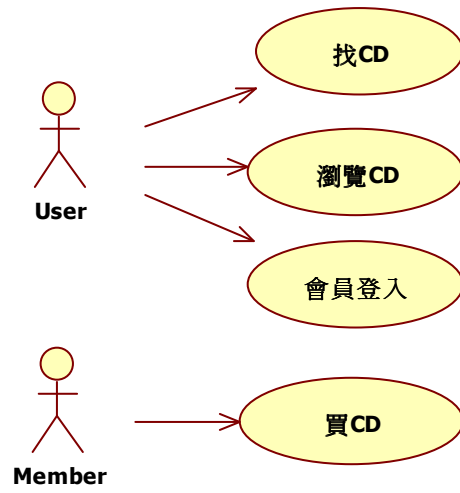


圖 Q13-4: 去掉參與者之間的一般化關係

Q14 開規格該用哪幾種 UML 圖來表示呢？

開規格該用哪幾種UML圖來表示呢？

(Jason&Phoebe於UML Blog之提問)

我是個初學者，想請教幾個問題；如果要開規格該用哪幾種圖來表示呢，而使用這些圖示的理由是什麼

實務上的經驗，使用案例圖和敘述 (Use Case Diagram and Description)、類別圖(Class Diagram)及循序圖(Sequence Diagram)這三款圖通常是少不了的。它們依序可呈現系統對外提供的服務，系統內部的程式結構，以及系統執行服務期間所需要用到的物件運作情況。

至於，活動圖(Activity Diagram)和狀態圖(State Machine Diagram)則視情況使用。若企業流程或系統流程過於複雜，覺得有需要透過 UML 圖來記錄與理解的話，可以使用活動圖。狀態圖通常在商用系統中較少使用到，我的使用時機一般是遇到較複雜且重要的企業物件(Business Object)時，才會搭配狀態圖來協助分析設計該物件。

關於這幾款 UML 圖的使用範例，可以購買 UML 答客問第一輯，參照文件中第 3 題的詳細說明。再者，2007 年 11 月中旬即將由上奇出版的「寫給 SA 的 UML/MDA 實務手冊」新書中，更加詳盡說明了這幾款圖的使用，屆時歡迎到書局翻閱及選購。

Q15 StarUML 只能針對類別圖產碼嗎？

StarUML 只能針對類別圖產碼嗎？

(匿名於UML Blog之提問)

本人是做OOAD的新進工程師最近做一個專案有用到StarUML這軟體，我想問一下在sequence, activity, collaboration裡畫的圖也可以generate出來嗎？因為我看到只能generate class diagram的code。

StarUML 確實只能自動產出類別圖(Class Diagram)的原始程式碼。我所測試過免費的 UML 工具中，都只能依據類別圖自動產碼。

至於付費的 UML 工具中，有些應用於嵌入式系統的 UML 工具，則有依據狀態圖(State Machine Diagram)自動產碼的功能，不過針對商用系統而言，這個功能似乎用不到。

如果有知道那個 UML 工具有針對其他 UML 圖產碼的功能，還盼告知，我也在找這樣的 UML 工具，先謝過。StarUML 使用上有任何的問題和經驗，我也有興趣知道，還請多分享，再次言謝。

Q16 超商收銀機系統之使用案例圖提問

超商收銀機系統之使用案例圖提問

(小婉於UML Blog之提問)

我們做的系統是類似超商收銀機的系統，我們不知道要不要把「顧客」加在使用案例圖(Use Case Diagram)當中，不知道要不要把店員跟店長的「登入」加入圖當中，不知道需不需要有系統維護人員？

只要是系統的使用者，就是使用案例圖中的參與者(Actor)。

以您的系統而言，如果「顧客」有直接使用超商收銀機系統的話，就可以把它加入途中，若沒有，那就不需要加入。我在想若以一般的超商收銀機來說，顧客應該不會是參與者，所以不需要加入。同理，也可以用是否有直接使用到系統來判斷「系統維護人員」需不需要加入。

此外，關於店員跟店長的「登入」服務，既然是系統所提供的服務項目之一，當然可以為它產生一個使用案例(Use Case)，並且加入使用案例圖中。

Q17 IBM RSA 之簡介與操作示範

※※

Q17.1 MAD 開發工具—IBM RSA ---- 67

Q17.2 使用 RSA 產出 PIM ---- 70

Q17.3 使用 RSA 產出 PSM ---- 106

Q17.4 RSA 未來可能的發展 ---- 123

前陣子因為工作緣故，有與IBM談論關於MDA/RSA主題的出版事宜，此文即節錄自當時所撰寫之文章，簡單介紹了RSA對於MDA專案的支援和操作步驟。

Q17.1 MDA 開發工具—IBM RSA

UML 是 MDA 中極為重要的核心要件，所以可以說，所有的 UML 開發工具其實都具備有支援 MDA 專案的能力。不過，MDA 概念方興未艾，所以工具廠商現在邊跑邊學習，可望很快就會推出更多的 MDA 開發工具。

開發 MDA 專案時，除了 UML 之外，開發工具是另一項不可或缺的關鍵。因此，本文將示範如何使用 IBM RSA(Rational Software Architect)來建構 MDA 專案。IBM RSA 具備有 UML 模式的建模功能，並且能夠讀取 PIM 自動產出特定平台的 PSM，以及讀取 PSM 自動產出特定語言的程式碼。請看圖 Q17-1，這是 IBM RSA 的建模畫面，選用區裡可讓開發人員選用 UML 圖示。

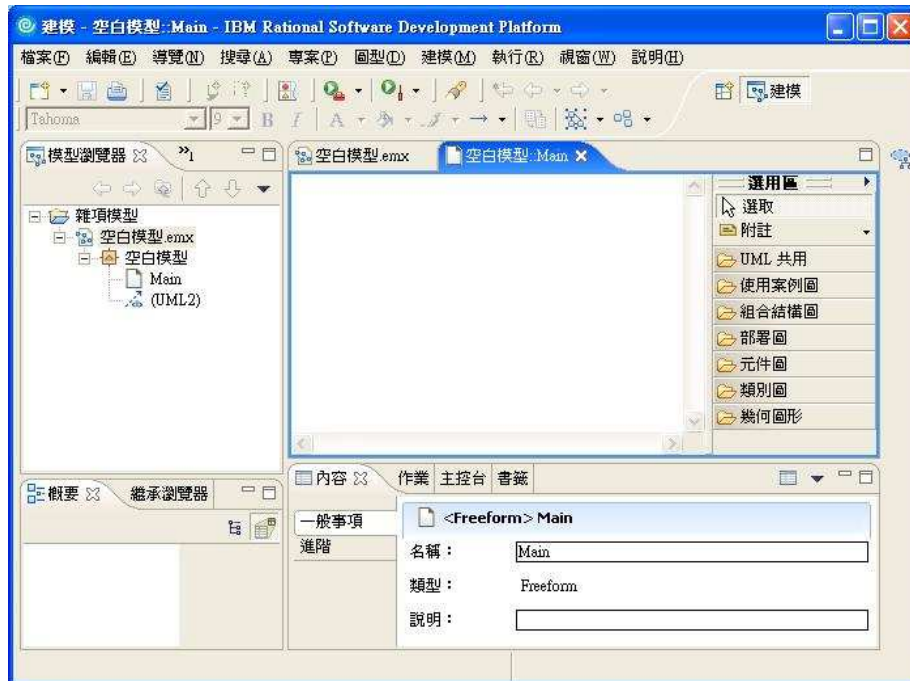


圖 Q17-1: IBM RSA 的建模畫面

最新的 UML 2.0 版定義了 13 款圖，不過 IBM RSA 並沒有全部支援。請看圖 Q17-2，這是在 IBM RSA 這套支援 UML 的軟體工具裡，所提供的 UML 圖款。再看圖 Q17.4-2 中右，由左至右依序為這是使用案例圖(Use Case Diagram)、類別圖(Class Diagram)和循序圖(Sequence Diagram)裡可用的小圖示。

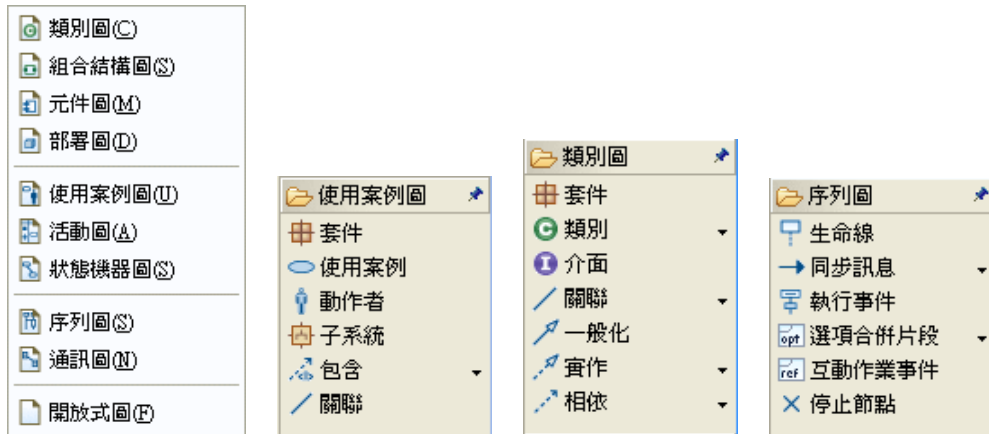


圖 Q17-2: UML 圖款

至於，依據 UML 模式轉出特定實體平台的 PSM 模式，以及程式碼的部份，IBM RSA 目前則支援了轉出 CORBA、EJB、XSD、C++ 及 Java，如圖 Q17-3 所示。



圖 Q17-3: PIM 至 PSM 及程式碼的選項

此外，IBM RS 還支援了 UML Profile 機制，讓開發人員可以自行定義 UML 方言。請看圖 Q17-4，開發人員只要新建 UML 設定檔專案，就可以自訂 UML 方言。



圖 Q17-4: UML 設定檔專案

在看圖 Q17-5 左，透過這個延伸設定，我們可以定義出 EJB2 實體平台專屬的 Remote Interface 元素。此外，再透過圖 Q17-5 右裡的圖示設定，還可以為 Remote Interface 指定一個獨特的小圖示。



圖 Q17-5: Remote Interface

Q17.2 使用 RSA 產出 PIM

此處，我們省略了 CIM 階段的企業目標分析和企業實體分析，僅以線上購物的企業流程分析為起始，接著平行進行系統功能設計和系統類別設計，最後則進行結帳使用案例的系統流程設計，如圖 Q17-6 所示。

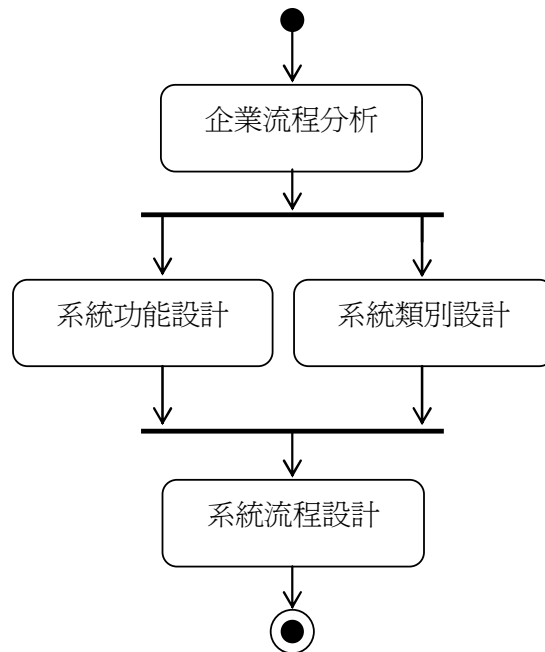


圖 Q17-6: MDA 專案開發程序

這個開發流程剛好用到 UML 的活動圖、使用案例圖、類別圖和循序圖，所以在稍後的次小節中，我們也透過線上購物的範例來說明 RSA 的操作步驟。

Q17.2.1 線上購物

我們參考了一般線上購物的流程，簡化設計出如圖 Q17-7 的流程範例。會員查看並選定商品，加入購物車後，就可以準備結帳了。結帳過程會確認購物資料，並輸入收貨人及信用卡資料，於連線獲取信用卡授權碼且產生交易序號後，發送電郵給供應商及會員，完成結帳手續。

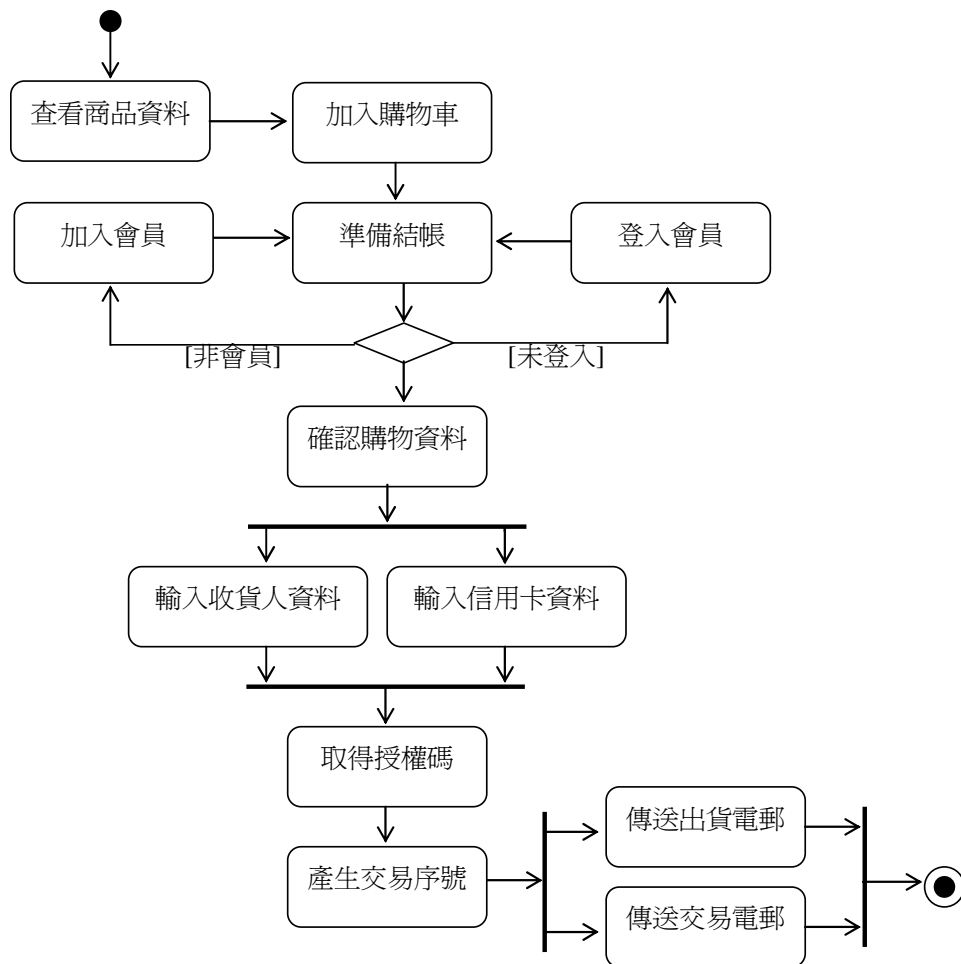


圖 Q17-7: 線上購物的流程分析

接著，我們參考 CIM 階段的企業流程分析，定義出如圖 Q17-8 的系統功能圖。再者，我們還為結帳撰寫使用案例敘述，以文字描述出結帳的內容細節。

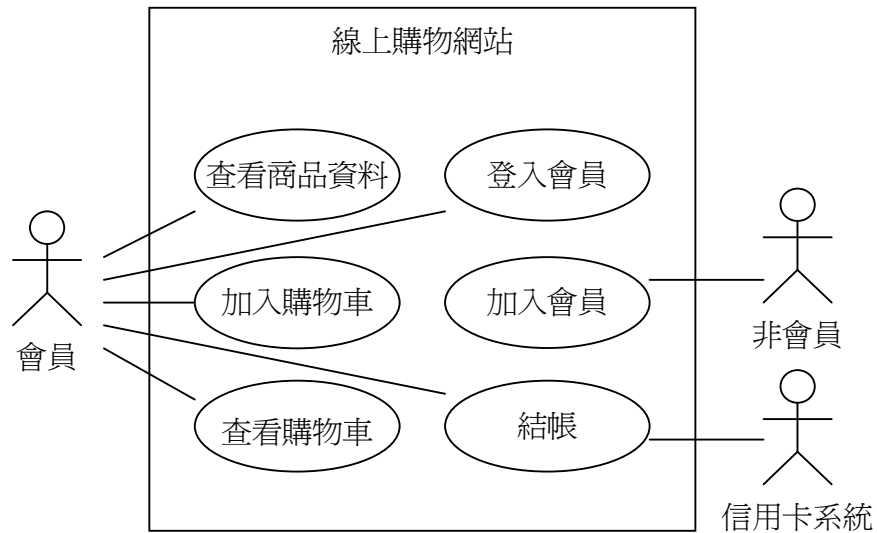


圖 Q17-8: 線上購物網站的功能設計

使用案例名稱：	結帳
主要程序：	
參與者的動作	系統的反應
1. 會員決定進入結帳程序。	2. 系統請會員確認購物資料。
3. 確認購物資料。	4. 請會員輸入收貨人及信用卡資料。
5. 輸入收貨人及信用卡資料。	6. 連線至信用卡系統，獲取信用卡授權碼。
	7. 產生交易序號。
	8. 傳送出貨電郵給供應商。
	9. 傳送交易電郵給會員。
	10. 出現交易完成的畫面。

實務上，系統功能設計與系統類別設計是並行且循環的，也就是先定義出一部份重要的系統功能，再設計一部份相關的系統類別。然後，再定義一些系統功能，也再設計一部份系統類別。請看圖 Q17-9，我們針對線上購物定義出幾個重要的系統類別。

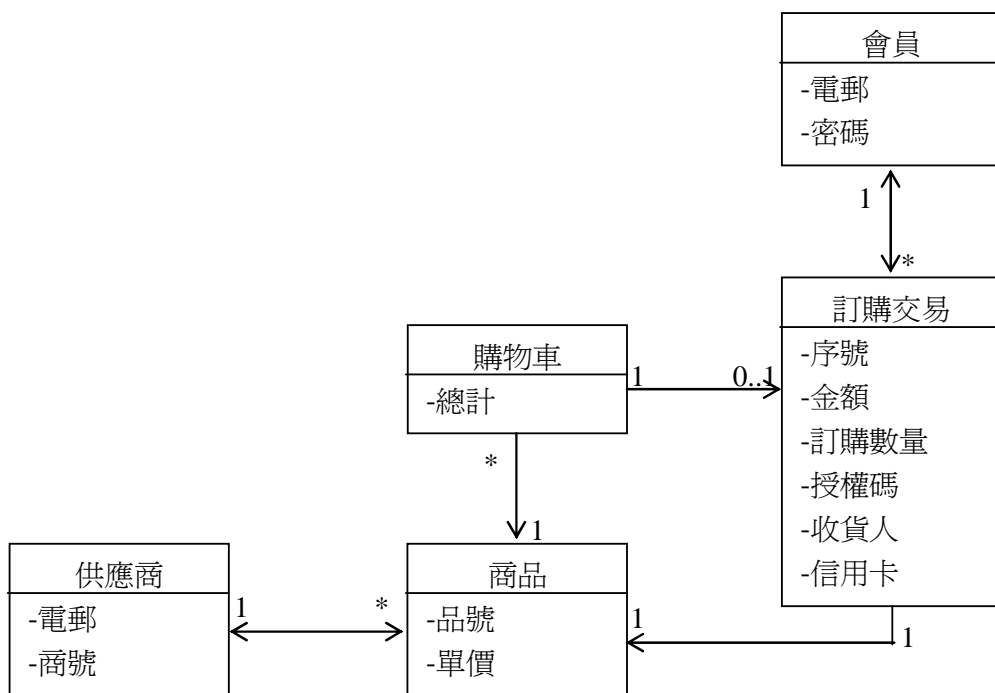


圖 Q17-9: 線上購物網站的類別設計

最後，參考系統功能設計與系統類別設計，為結帳使用案例設計出運作流程並繪製成循序圖，如圖 Q17-10 所示。

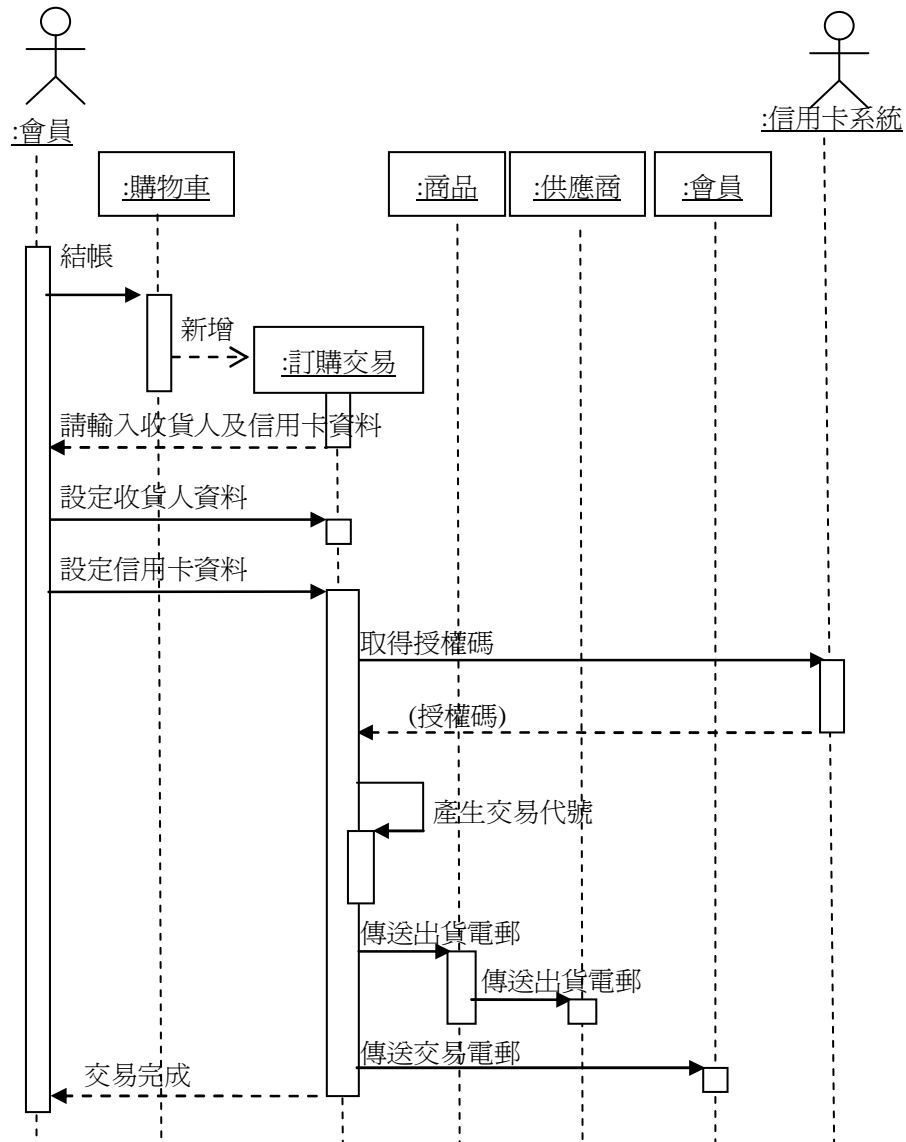


圖 Q17-10: 結帳使用案例的流程設計

由於，系統流程設計將整合了系統功能設計與系統類別設計，所以假使我們將圖 Q17-9 的類別圖改成如圖 Q17-11 的話，剛才的圖 Q17-10 循序圖

恐怕就不能運作而必須跟著修改了。請看圖 Q17-11 的系統類別設計，我們將原先放置於訂購交易類別裡的收貨人與信用卡，獨立出來成為兩個新的類別。

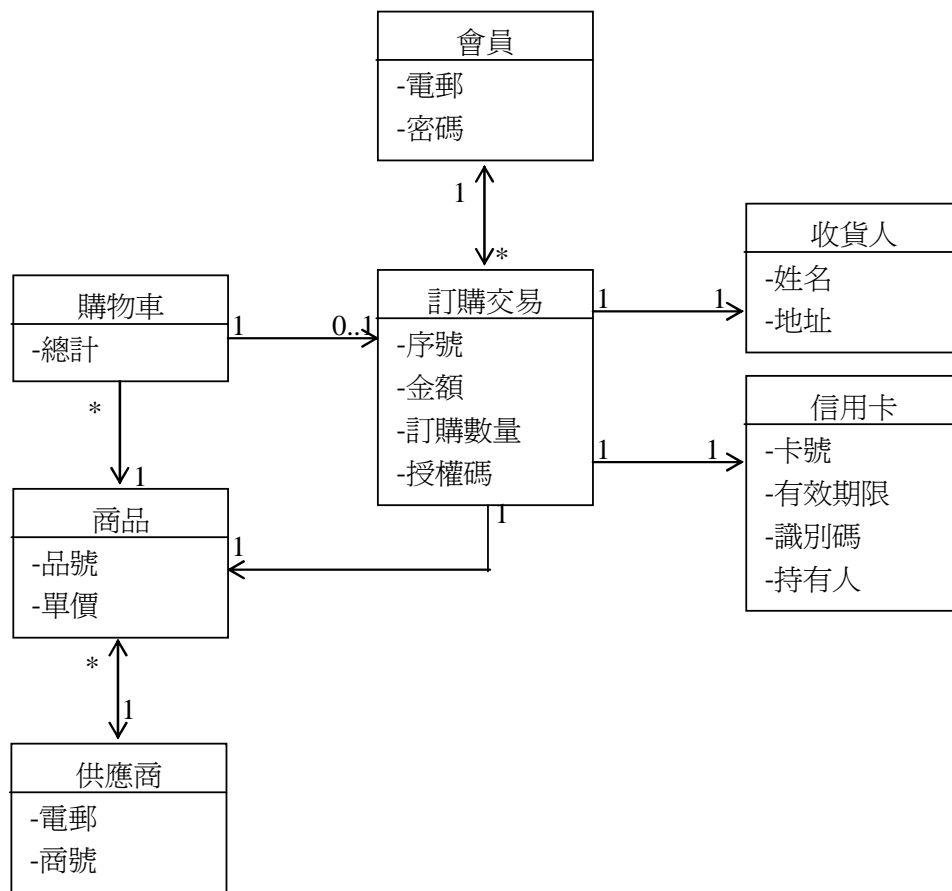


圖 Q17-11: 線上購物網站的類別設計

Q17.2.2 新建 UML 專案與模型

開啓 RSA 並設定過工作區之後，將會進入如圖 Q17-12 的主畫面。接

著，我們得新建一個 UML 專案才能夠開始建立 CIM 與 PIM 模型。

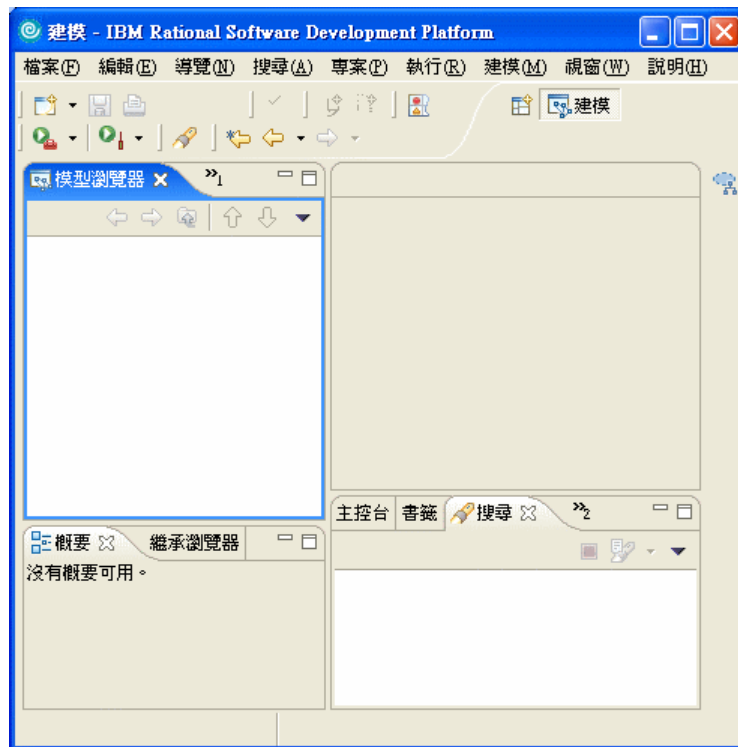


圖 Q17-12: RSA 的主畫面

新建 UML 專案，以及 CIM 和 PIM 模型的步驟如下：

1. 選取主選單中的【檔案►新建►專案】，如果 UML 專案精靈已經安裝，將會出現如圖 Q17-13 的選取精靈。

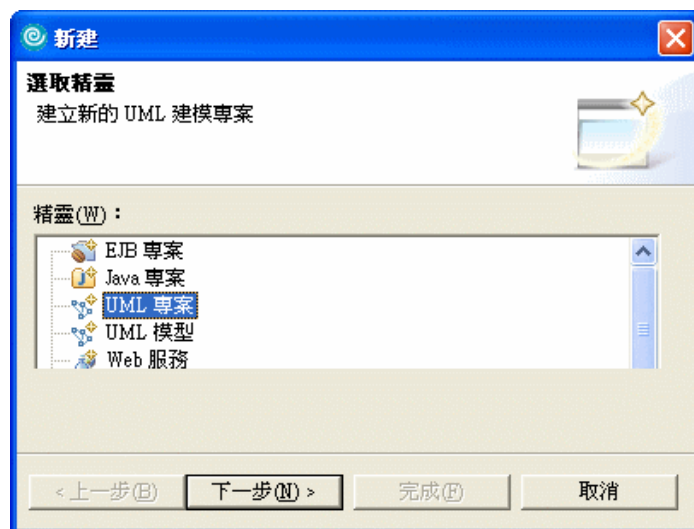


圖 Q17-13: 選取精靈

2. 選取 UML 專案並執行下一步之後，將出現如圖 Q17-14 的對話窗。

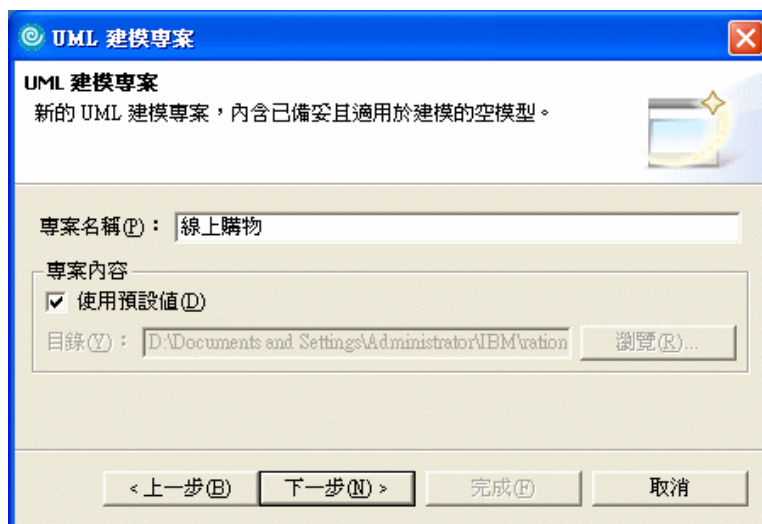


圖 Q17-14: UML 建模專案

3. 輸入「線上購物」做為專案名稱並執行下一步之後，將出現如圖 Q17-15 新建 UML 模型的對話窗。選取【空白模型】範本並更名為「CIM 模型」後，按下完成鍵。

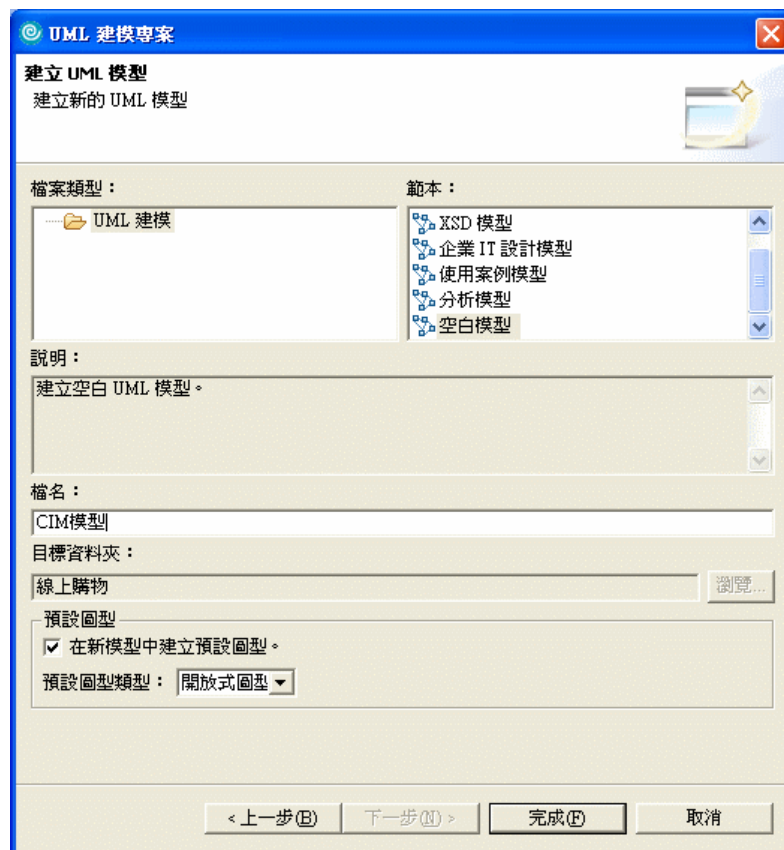


圖 Q17-15: 建立 CIM 模型

4. 隨即，RSA 將自動開啓空白的 CIM 模型，並進入建模環境中，如圖 Q17-16 所示。

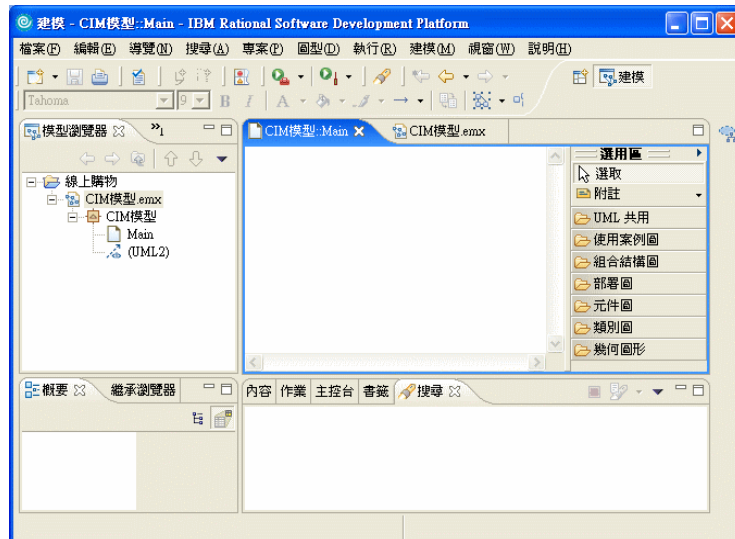


圖 Q17-16: 空白的 CIM 模型建模環境

- 接著，我們還要新增另一個 PIM 模型。選取主選單中的【檔案►新建►UML 模型】，於新建 UML 模型的對話窗中，選取【空白模型】範本並更名為「PIM 模型」後，按下完成鍵。
- 請看 RSA 左上方的模型瀏覽器中，已經新增了 CIM 與 PIM 兩項模型，如圖 Q17-17 所示。

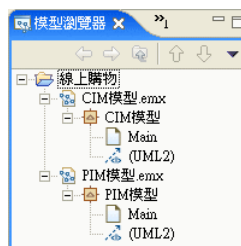


圖 Q17-17: 模型瀏覽器

Q17.2.3 企業流程分析

在此小節中，我們會說明如何使用 RSA 建立線上購物的企業流程分析模式，最後會建立出如圖 Q17-18 的 UML 活動圖。

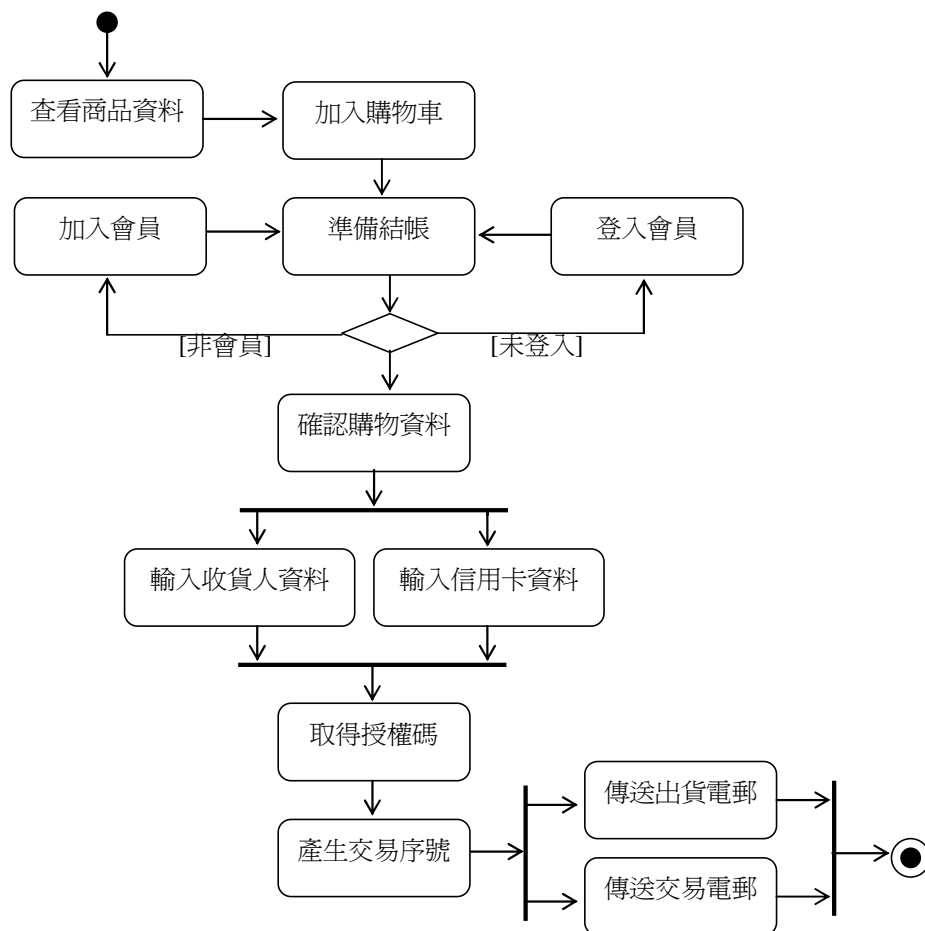


圖 Q17-18: 線上購物流程的活動圖

新增活動圖

首先，我們要新增一張活動圖，其步驟如下：

1. 右擊選取模型瀏覽器中的【CIM 模型】，並於選單中選取【新增圖型▶活動圖】，如圖 Q17-19 所示。

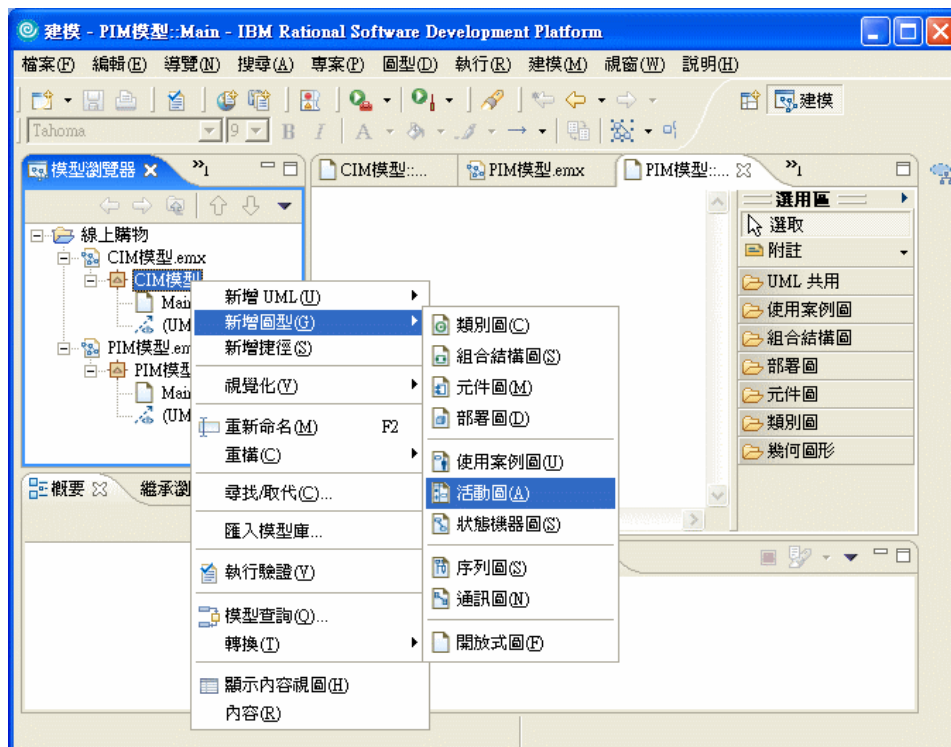


圖 Q17-19: 新增活動圖

2. 請看 RSA 左上方的模型瀏覽器中，已經新增了名為「活動 1」的活動圖，如圖 Q17-20 所示。

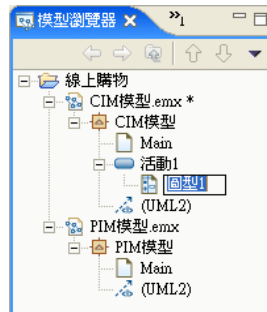


圖 Q17-20: 活動 1

3. 雙擊活動 1 圖面左上角名稱處，並將「活動 1」更名為「線上購物企業流程」，如圖 Q17-21 所示。

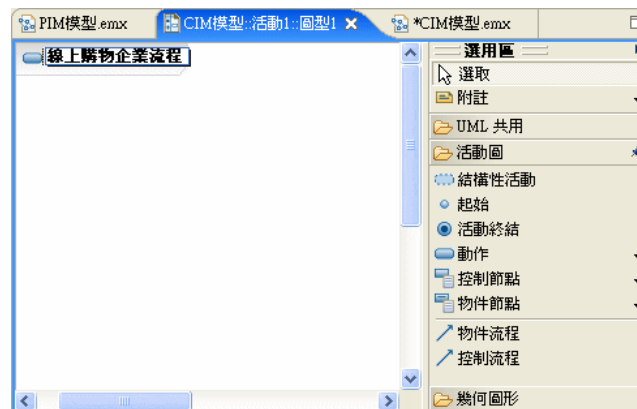


圖 Q17-21: 線上購物企業流程

建立活動圖

現在，我們就可以開始建立活動圖的內容了，其步驟如下：

1. 選取活動圖選用區裡的【起始】，如圖 Q17-22 所示。



圖 Q17-22: 活動圖選用區

2. 接著，將游標移至活動圖面空白處，再按一次滑鼠左鍵，隨即可見圖面上多了一個起始的小圓心，如圖 Q17-23 所示。

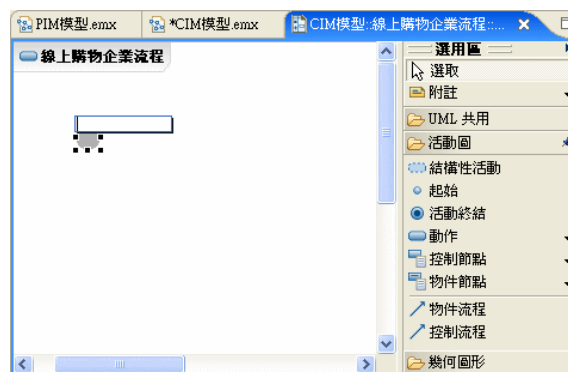


圖 Q17-23: 新增起始

3. 依照上述步驟，點選活動圖選用區裡的【動作】放置於圖面上，並更名為「查看商品資料」，如圖 Q17-24 所示。

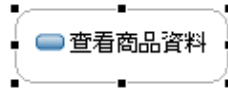


圖 Q17-24: 新增動作

- 點選活動圖選用區裡的【控制流程】，並由起始圖示拖曳至查看商品資料動作圖示，如圖 Q17-25 所示。請依照前述步驟，建立出其餘的動作與控制流程，直到建立決策圖示。



圖 Q17-25: 新增控制流程

- 點選活動圖選用區裡的【控制節點▶決策】，如圖 Q17-26 所示。



圖 Q17-26: 新增決策

6. 接著，將決策置於圖面上，並建立起「準備結帳」和「登入會員」與決策之間的控制流程，如圖 Q17-27 所示。



圖 Q17-27: 新增控制流程

7. 隨後，點選決策與「登入會員」之間的控制流程，並於內容頁籤的保護主體，填入「未登入」，如圖 Q17-28 所示。

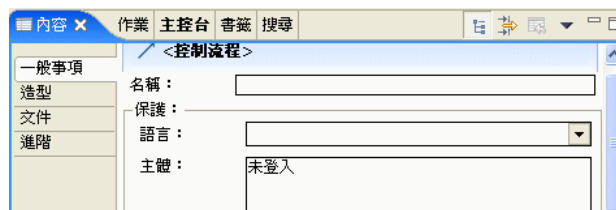


圖 Q17-28: 新增保護

8. 圖面上馬上就出現未登入的限制條件了，如圖 Q17-29 所示。請依照前述步驟，建立起其餘的動作與控制流程，直到建立分出圖示。

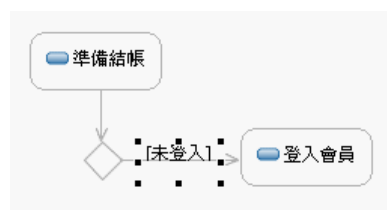


圖 Q17-29: 未登入

9. 點選活動圖選用區裡的【控制節點▶分出】，放置於圖面上，接著建立起「確認購物資料」與分出之間的控制流程，如圖 Q17-30 所示。

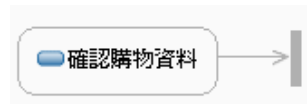


圖 Q17-30: 分出

10. 請將游標移至分出圖示旁，會出現兩個箭頭的小符號，如圖 Q17-31 所示。

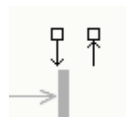


圖 Q17-31: 兩個箭頭

11. 請按照「請按一下拖曳來建立關係」的指示字眼，按一下拖曳，並點選【建立控制流程至新動作】來新增動作，如圖 Q17-32 所示。



圖 Q17-32: 建立控制流程至新動作

12. 圖面上馬上就出現了新動作，請更名為「輸入收貨人資料」，如圖 Q17-33 所示。

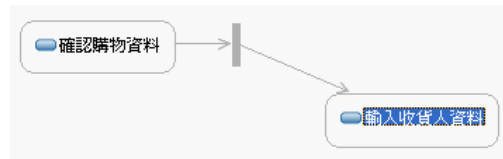


圖 Q17-33: 輸入收貨人資料

13. 請將游標移至「輸入收貨人資料」圖示旁，會出現兩個箭頭的小符號。請按照「請按一下拖曳來建立關係」的指示字眼，按一下拖曳，並點選【建立新控制流程至►新結合】來建立關係，如圖 Q17-34 所示。請依照前述步驟，建立出其餘的動作與控制流程，直到建立活動終結圖示。

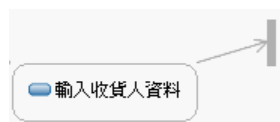


圖 Q17-34: 結合

14. 請將游標移至結合圖示旁，會出現兩個箭頭的小符號。請按照「請按一下拖曳來建立關係」的指示字眼，按一下拖曳，並點選【建立新控制流程至新活動終結】來結束整條流程，如圖 Q17-35 所示。



圖 Q17-35: 活動終結

Q17.2.4 系統功能設計

在此小節中，我們會說明如何使用 RSA 建立線上購物網站的系統功能模式，最後會建立出如圖 Q17-36 的 UML 使用案例圖。

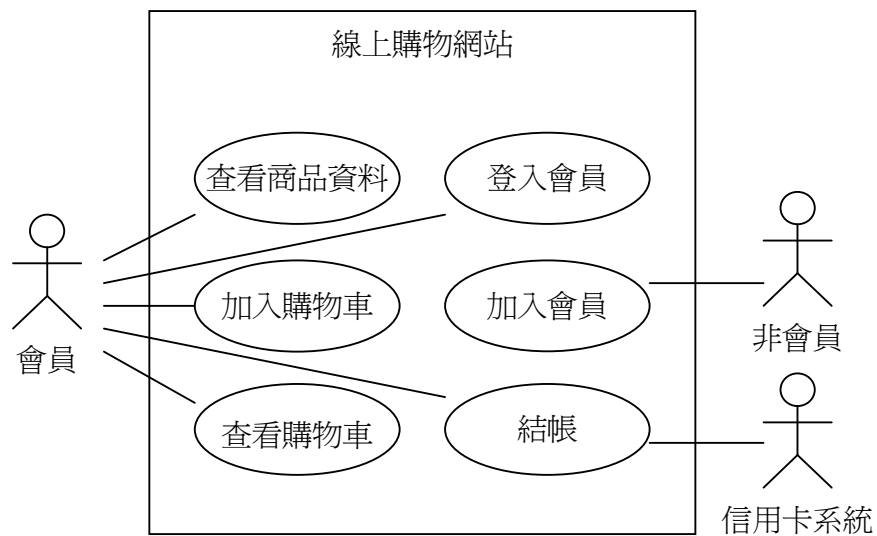


圖 Q17-36: 線上購物網站的使用案例圖

由於，RSA 中譯 “Use Case” 為「使用案例」，中譯 “Actor” 為「動作者」。而本文則中譯為「使用案例」與「參與者」。所以，此處會有混用這幾個中譯詞的情況。

新增使用案例圖

首先，我們要新增一張使用案例圖，其步驟如下：

1. 右擊選取模型瀏覽器中的【PIM 模型】，並於選單中選取【新增圖型►使用案例圖】，如圖 Q17-37 所示。

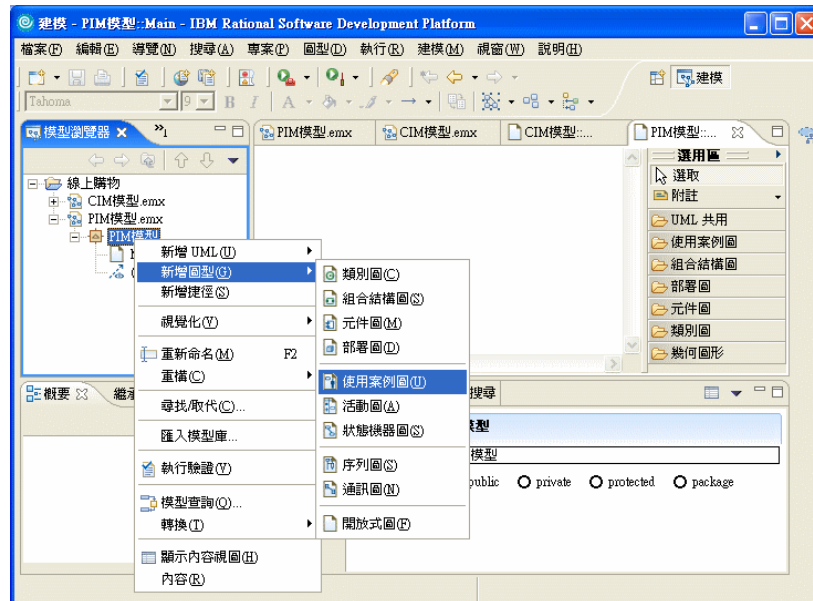


圖 Q17-37: 新增使用案例圖

2. 請看 RSA 左上方的模型瀏覽器中，已經新增了名為「圖型 1」的使用案例圖，如圖 Q17-38 所示。

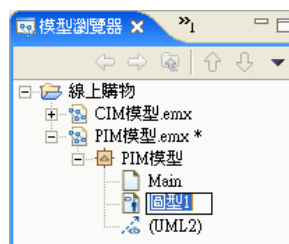


圖 Q17-38: 圖型 1

- 點選模型瀏覽器中的「圖型 1」，並於內容頁籤的名稱處，更名為「線上購物網站」，如圖 Q17-39 所示。

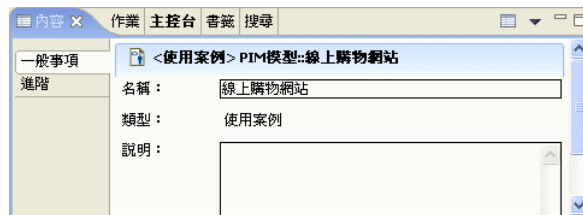


圖 Q17-39: 線上購物網站

建立使用案例圖

現在，我們就可以開始建立使用案例圖的內容了，其步驟如下：

- 選取使用案例圖選用區裡的【使用案例】，如圖 Q17-40 所示。



圖 Q17-40: 使用案例圖選用區

- 接著，將游標移至使用案例圖面空白處，再按一次滑鼠左鍵，隨即可見圖面上多了一個名為「使用個案 1」的橢圓，如圖 Q17-41 所示。



圖 Q17-41: 新增使用案例

3. 請為新增的使用案例更名為「結帳」，如圖 Q17-42 所示。



圖 Q17-42: 結帳

4. 依照前述步驟，點選使用案例圖選用區裡的【動作者】放置於圖面上，並更名為「會員」，如圖 Q17-43 所示。



圖 Q17-43: 新增參與者

5. 點選使用案例圖選用區裡的【關聯】，並由會員圖示拖曳至結帳圖示，如圖 Q17-44 所示。

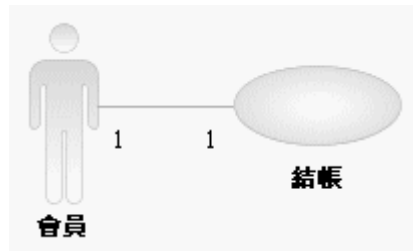


圖 Q17-44: 新增關聯

6. 請將游標移至使用案例圖示旁，會出現兩個箭頭的小符號，如圖 Q17-45 所示。

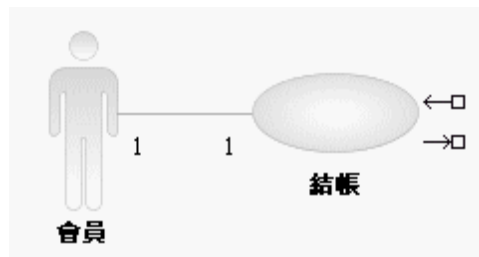


圖 Q17-45: 兩個箭頭

7. 請按照「請按一下拖曳來建立關係」的指示字眼，按一下拖曳，並點選【建立新關聯至►新動作者】，如圖 Q17-46 所示。

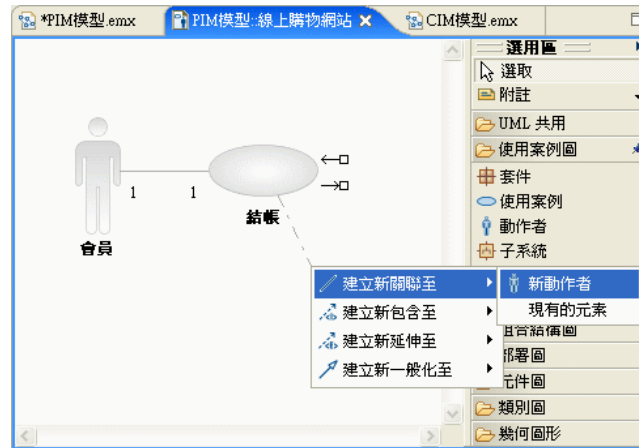


圖 Q17-46: 建立新關聯至新動作者

8. 圖面上馬上就出現了新的參與者，請更名為「信用卡系統」，如圖 Q17-47 所示。

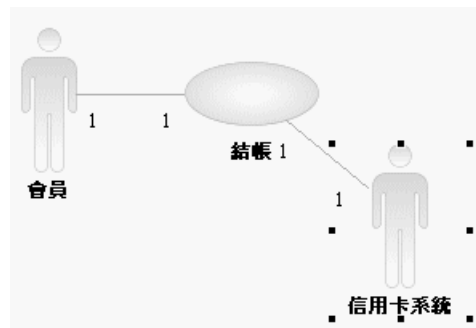


圖 Q17-47: 信用卡系統

9. 請依照前述步驟，建立出其餘的使用案例、參與者與其間之關聯，直到完成整張使用案例圖。

Q17.2.5 系統類別設計

在此小節中，我們會說明如何使用 RSA 建立線上購物網站的系統類別模式，最後會建立出如圖 Q17-48 的 UML 類別圖。

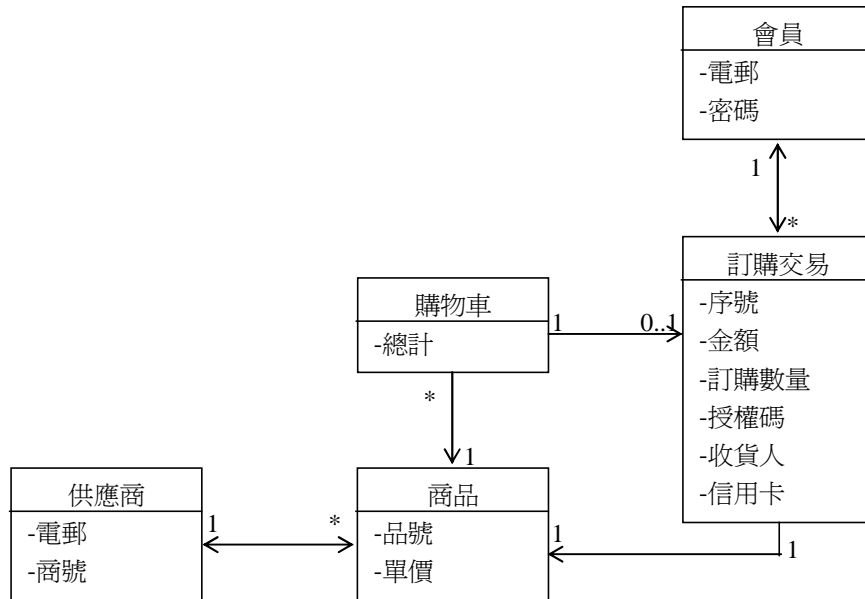


圖 Q17-48: 線上購物網站的類別圖

新增類別圖

首先，我們要新增一張類別圖，其步驟如下：

1. 右擊選取模型瀏覽器中的【PIM 模型】，並於選單中選取【新增圖型►類別圖】，如圖 Q17-49 所示。

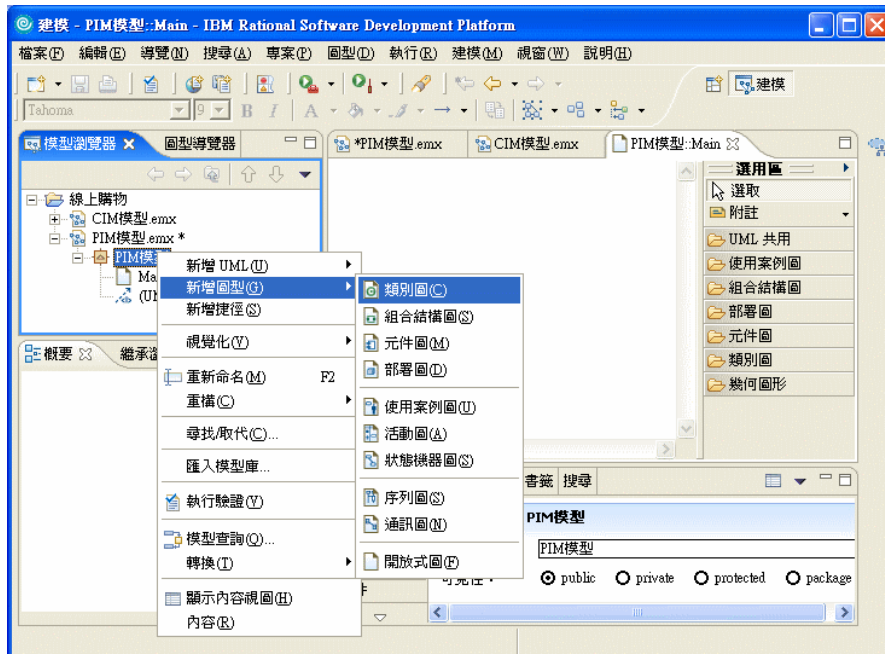


圖 Q17-49: 新增類別圖

2. 請看 RSA 左上方的模型瀏覽器中，已經新增了名為「圖型 1」的類別圖，如圖 Q17-50 所示。

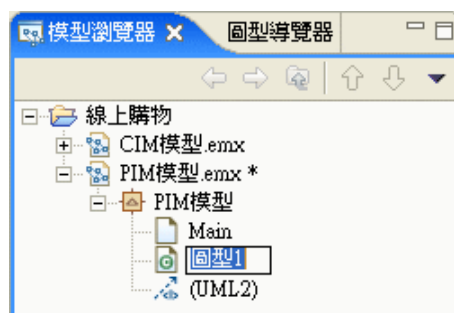


圖 Q17-50: 圖型 1

- 點選模型瀏覽器中的「圖型 1」，並於內容頁籤的名稱處，更名為「線上購物系統類別圖」，如圖 Q17-51 所示。

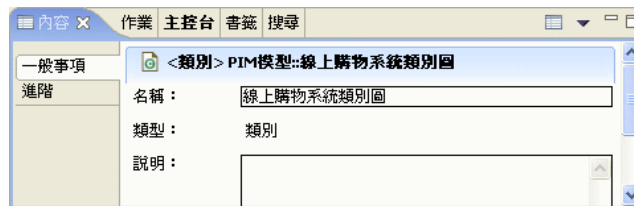


圖 Q17-51: 線上購物系統類別圖

建立類別圖

現在，我們就可以開始建立類別圖的內容了，其步驟如下：

- 選取類別圖選用區裡的【類別】，如圖 Q17-52 所示。



圖 Q17-52: 類別圖選用區

- 接著，將游標移至類別圖面空白處，再按一次滑鼠左鍵，隨即可見圖面上多了一個類別的矩形。請為新增的類別更名為「購物車」，如圖 Q17-53 所示。



圖 Q17-53: 新增類別

3. 請將游標移至類別圖示旁，會出現新增新屬性的小紅方格符號，如圖 Q17-54 所示。



圖 Q17-54: 小紅方格符號

4. 按下小紅方格符號之後，即刻新增了一個屬性，如圖 Q17-55 所示。請為新增的屬性更名為「總計」。



圖 Q17-55: 新增屬性

5. 請將游標移至類別圖示旁，會出現兩個箭頭的小符號，如圖 Q17-56 所示。



圖 Q17-56: 兩個箭頭

6. 請按照「請按一下拖曳來建立關係」的指示字眼，按一下拖曳，並點選【建立新關聯至►新類別】，如圖 Q17-57 所示。

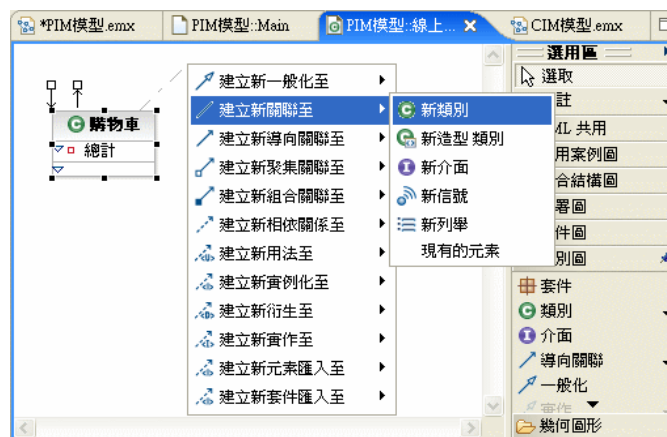


圖 Q17-57: 建立新關聯至新類別

7. 圖面上馬上就出現了新類別，請更名為「商品」，如圖 Q17-58 所示。

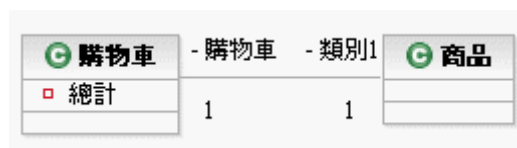


圖 Q17-58: 商品

8. 點選「購物車」與「商品」之間的關聯，請將其內容頁籤的一般事項更

改為圖 Q17-59 的設定。

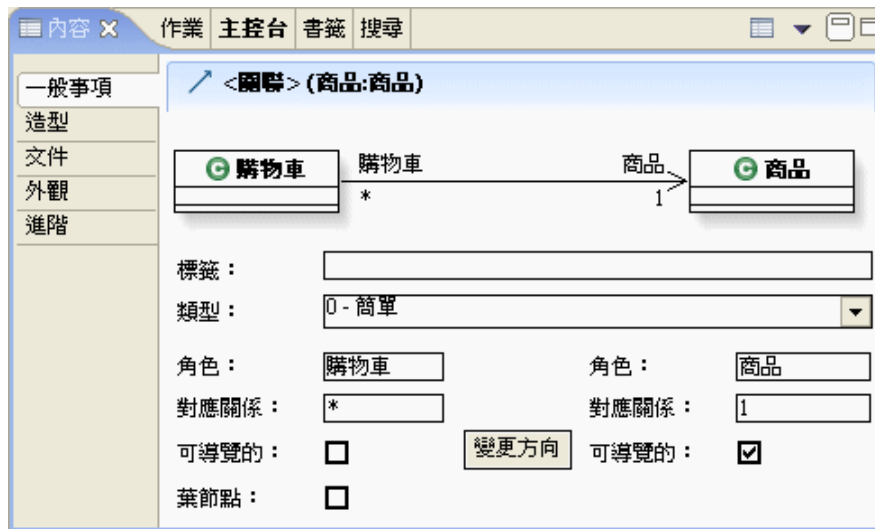


圖 Q17-59: 更改設定

9. RSA 立即同步更新圖面，如圖 Q17-60 所示。

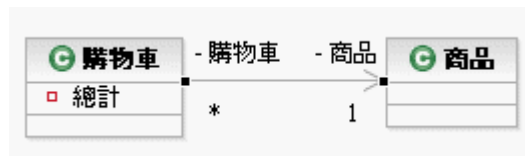


圖 Q17-60: 更新圖面

10. 請依照前述步驟，建立出其餘的類別與其間之關聯，直到完成整張類別圖。

Q17.2.6 系統流程設計

在此小節中，我們會說明如何使用 RSA 建立結帳使用案例的運作流程模式，最後會建立出如圖 Q17-61 的 UML 循序圖。

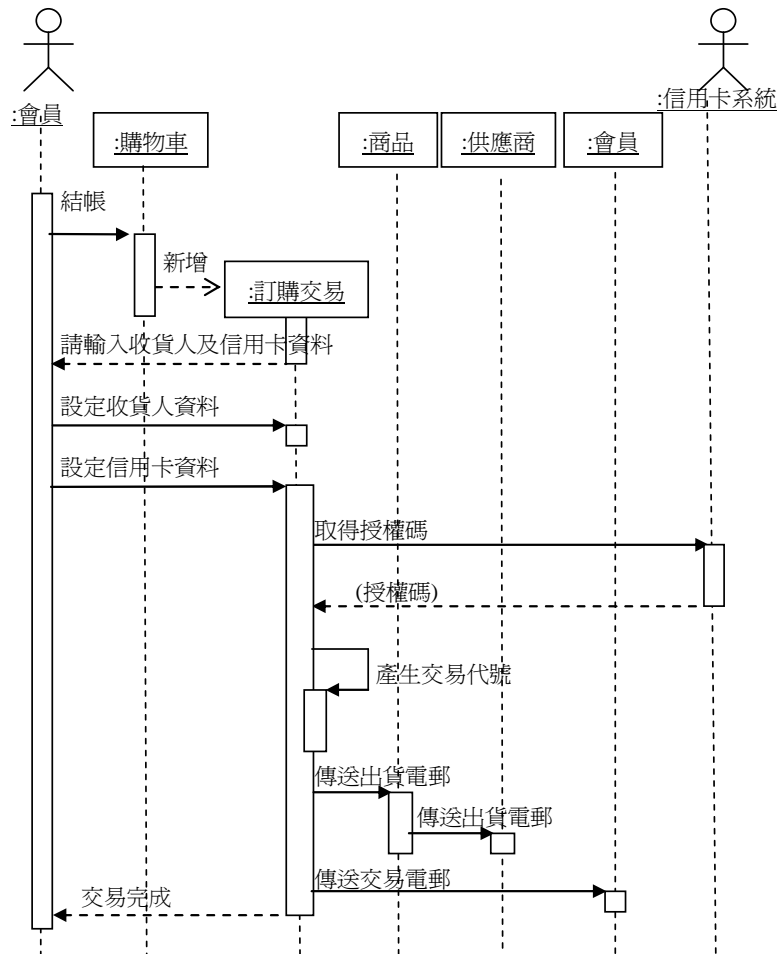


圖 Q17-61: 結帳使用案例的循序圖

新增循序圖

首先，我們要新增一張循序圖，其步驟如下：

1. 右擊選取模型瀏覽器中的【PIM 模型】，並於選單中選取【新增圖型►序列圖】，如圖 Q17-62 所示。

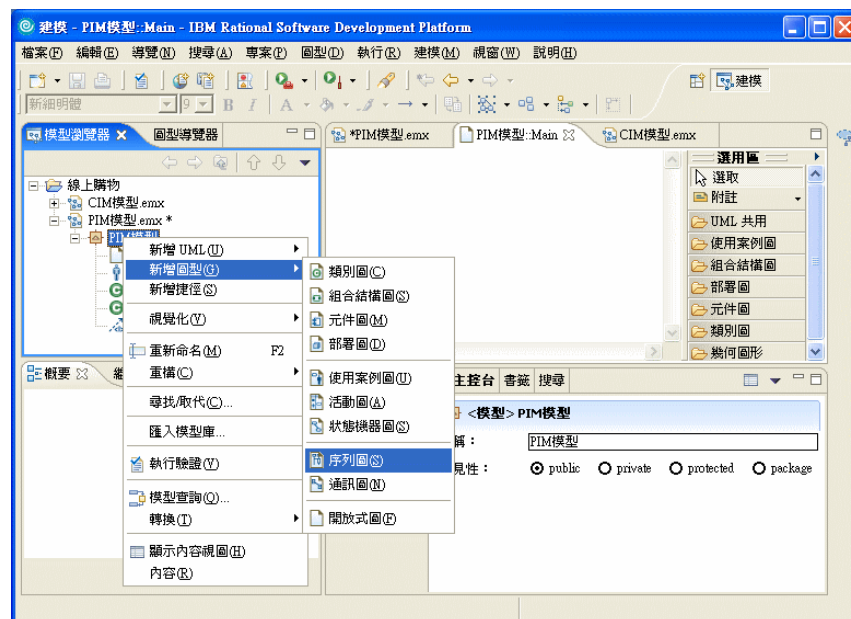


圖 Q17-62: 新增循序圖

2. 請看 RSA 左上方的模型瀏覽器中，已經新增了名為「圖型 1」的循序圖。
3. 雙擊圖面左上角名稱處，並將「交談作業 1」更名為「結帳流程」，如圖 Q17-63 所示。

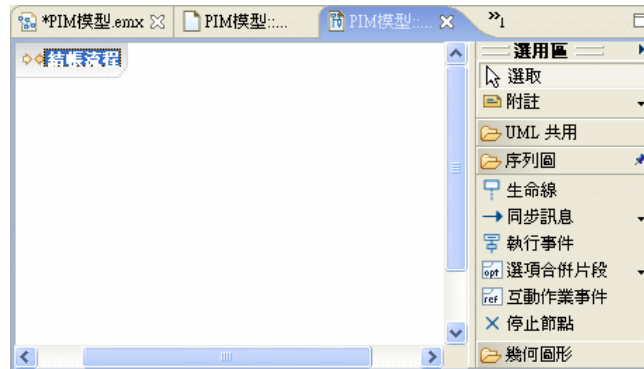


圖 Q17-63: 結帳流程

建立循序圖

現在，我們就可以開始建立循序圖的內容了，其步驟如下：

1. 選取循序圖選用區裡的【生命線】，如圖 Q17-64 所示。

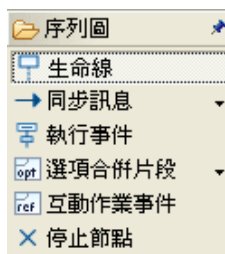


圖 Q17-64: 循序圖選用區

2. 接著，將游標移至循序圖面空白處，再按一次滑鼠左鍵，並點選【選取現有的元素】，如圖 Q17-65 所示。

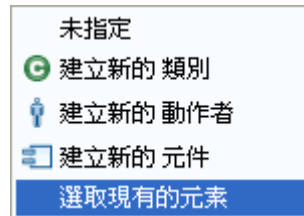


圖 Q17-65: 選取現有的元素

3. 在選取「會員」參與者之後，隨即可見圖面上多了一個會員物件，如圖 Q17-66 所示。請依照前述步驟，建立出其餘的物件，直到建立訊息圖示。



圖 Q17-66: 會員物件

4. 點選循序圖選用區裡的【同步訊息】，並由會員虛線處拖曳至購物車虛線處，並於出現的對話窗中輸入「結帳」做為訊息名稱，如圖 Q17-67 所示。

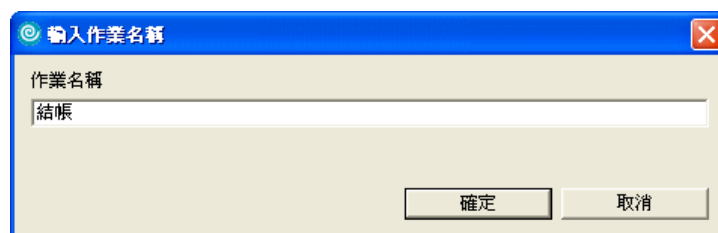


圖 Q17-67: 新增訊息

5. 圖面上馬上就同步更新，出現結帳訊息，如圖 Q17-68 所示。



圖 Q17-68: 結帳訊息

6. 請將游標移至長條矩形旁，會出現兩個箭頭的小符號，如圖 Q17-69 所示。

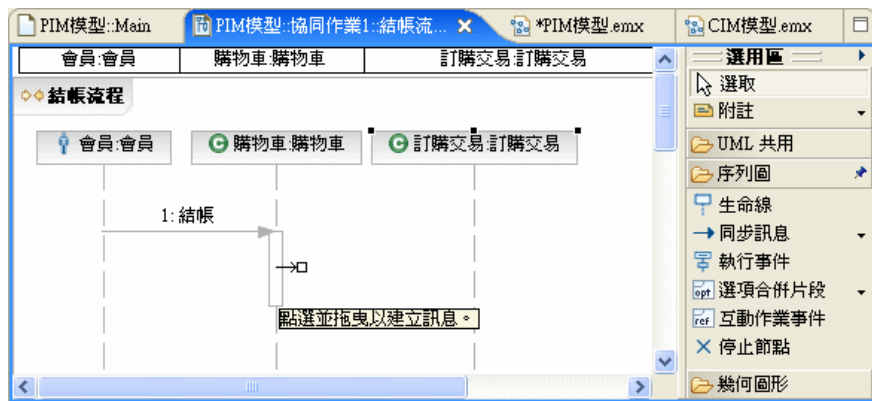


圖 Q17-69: 兩個箭頭

7. 請按照「點選並拖曳以建立訊息」的指示字眼，按一下拖曳至訂購交易虛線處，並點選【建立新的建立訊息】且更名為「新增」，如圖 Q17-70 所示。

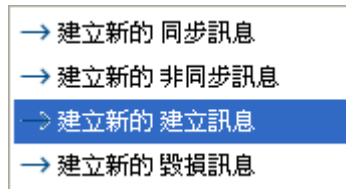


圖 Q17-70: 建立新的建立訊息

8. 圖面上馬上就同步更新，出現新增訊息，如圖 Q17-71 所示。

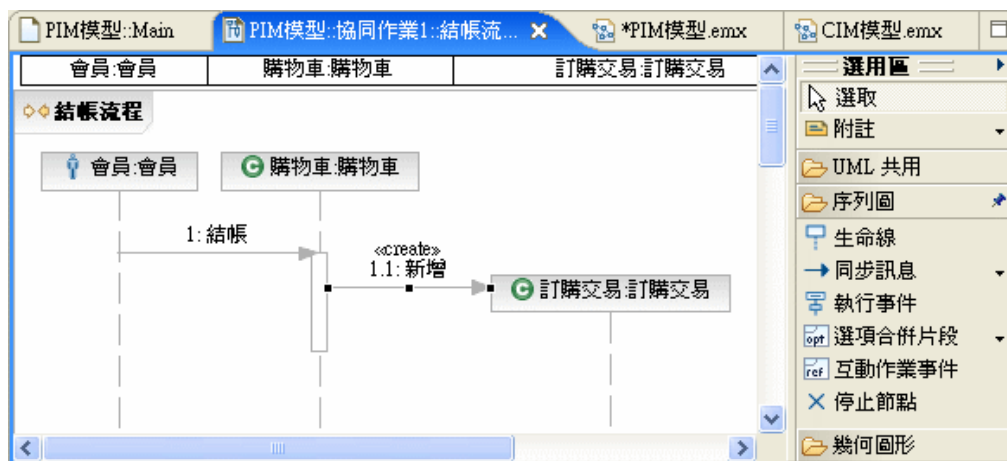


圖 Q17-71: 新增訊息

9. 請依照前述步驟，建立出其餘訊息，直到完成整張循序圖。

Q17.3 使用 RSA 產出 PSM

在此小節中，我們會說明如何操作 IBM 的 RSA 來轉出 PSM 模式，以及 Java 程式碼。因為，考量 RSA 轉出 Java 程式碼不適合使用中文，所以此處我們重新提出一個英文簡化版的 PIM 設計。然後，才示範如何操作 RSA

轉出 Java 程式碼。再者，由於 RSA 目前還未提供 EJB3 的支援，所以我們僅示範由 PIM 轉出 EJB2 的操作步驟。

這是一個超級簡化版的 PIM 設計，主要在示範 RSA 轉出 EJB 設計的操作步驟，如圖 Q17-72~74 依序為 PIM 系統功能圖、PIM 系統類別圖和 PIM 系統流程圖。

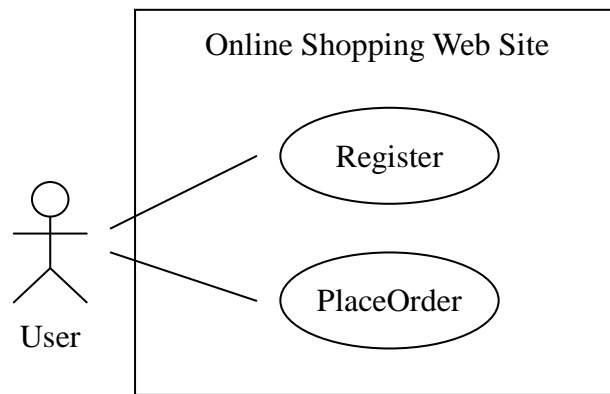


圖 Q17-72: PIM 的系統功能圖

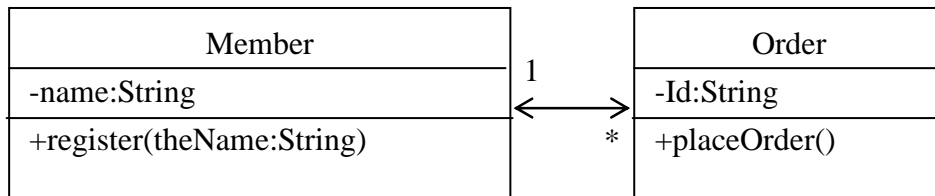


圖 Q17-73: PIM 的系統類別圖

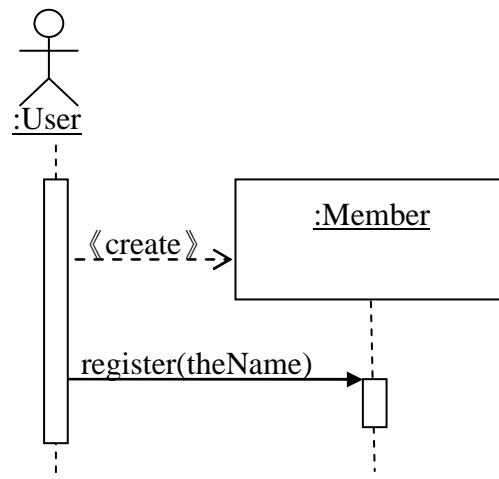


圖 Q17-74: PIM 的加入會員用例流程圖

Q17.3.1 PIM 轉 PSM

新建完 UML 專案及 PIM 模型之後，我們就可以開始依據 UML 專案的 PIM 內容轉出 EJB 專案的 PSM 內容了。步驟如下：

1. 左擊選取模型瀏覽器中的【PIM 模型】，並於選取主選單中的【建模►轉換►UML 至 EJB】，如圖 Q17-75 所示。

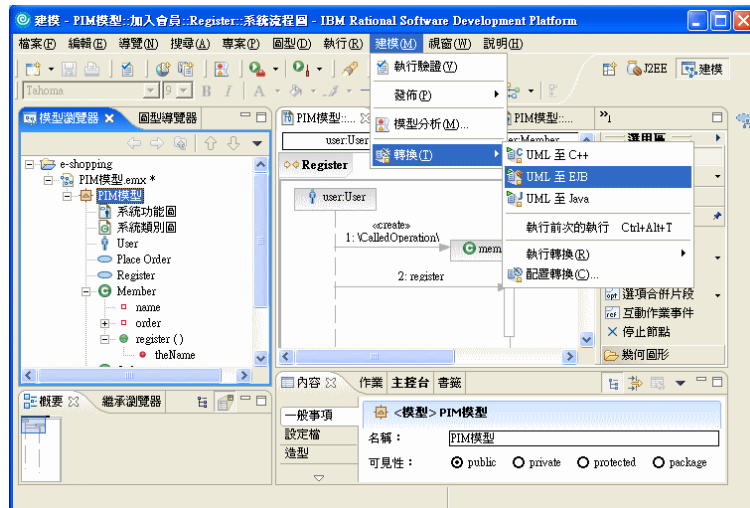


圖 Q17-75: 建模轉換 UML 至 EJB

2. 執行 UML 至 EJB 的轉換之後，會出現如圖 Q17-76 的對話窗。

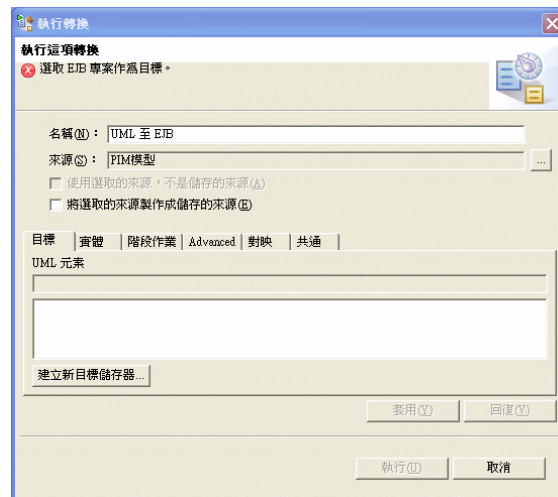


圖 Q17-76: 執行轉換

3. 按下【建立新目標儲存器】的按鍵之後，RSA 會出現新建 EJB 專案的對話窗。請輸入專案名稱，並且按下【顯示進階】的按鍵，更改設定成如圖 Q17-77 的樣子。

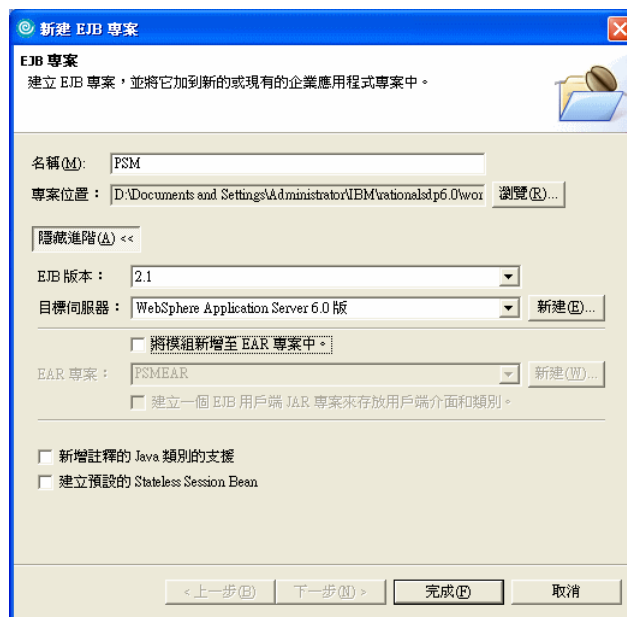


圖 Q17-77: 進階設定

4. 按下【完成】鍵之後，將回到執行轉換的對話窗。不同的是，此時在目標頁籤中，已經出現了一個剛才新建的 EJB 專案了，如圖 Q17-78 所示。

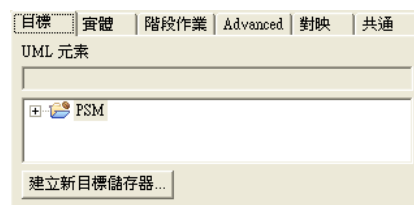


圖 Q17-78: 目標頁籤

5. 同時，可以在實體頁籤與階段作業頁籤處，設定 Entity Bean 與 Session Bean 的本端界面(Local Interface)與遠端界面(Remote Interface)，如圖 Q17-79 所示。



圖 Q17-79: 實體頁籤與階段作業頁籤

6. 最後，請按下執行轉換視窗裡的【執行】鍵，RSA 隨即自動產出 EJB 專案及 Java 程式碼。
7. 請看 RSA 模型瀏覽器中，已經新增了 EJB 專案，如圖 Q17-80 所示。

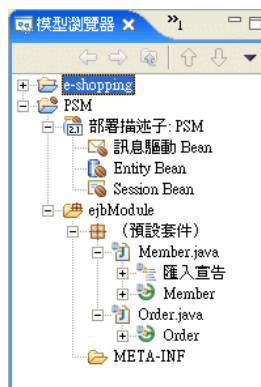


圖 Q17-80: 新增了 EJB 專案

8. 點開其中的 Member.java 以及 Order.java 可以見到如下列所示的 Java 程式碼。

```
//// RSA Java Code
1.  /**
2.     * @author Administrator
3.     *
4.     * TODO 如果要變更這個產生的類別註解的範本，請移至
5.     * 視窗 - 喜好設定 - Java - 程式碼樣式 - 程式碼範本
6.     * @generated "UML 至 EJB (uml2.ejb.transformation)"
7.     */
8.  public class Member {
9.      /**
10.         * <code>name</code> 的註解
11.         * @generated "UML 至 EJB (uml2.ejb.transformation)"
12.         */
13.         private String name;
14.
15.         /**
16.         * @return 傳回 name。
17.         * @generated "UML 至 EJB (uml2.ejb.transformation)"
18.         */
18.         public String getName() {
19.             return name;
20.         }
21.
22.         /**
23.         * @param thename 要設定的 name。
24.         * @generated "UML 至 EJB (uml2.ejb.transformation)"
25.         */
25.         public void setName(String thename) {
26.             name = thename;
27.         }
28.
29.         /**
30.         * <code>order</code> 的註解
31.         * @generated "UML 至 EJB (uml2.ejb.transformation)"
32.         */
32.         private Set order;
```

```

33.      /**
34.       * @return 傳回 order。
35.       * @generated "UML 至 EJB (uml2.ejb.transformation)"
36.       */
37.      public Set getOrder() {
38.          return order;
39.      }

40.      /**
41.       * @param theorder 要設定的 order。
42.       * @generated "UML 至 EJB (uml2.ejb.transformation)"
43.       */
44.      public void setOrder(Set theorder) {
45.          order = theorder;
46.      }

47.      /**
48.       * @param theName
49.       * @generated "UML 至 EJB (uml2.ejb.transformation)"
50.       */
51.      public void register(String theName) {
52.          // TODO 自動產生方法 Stub
53.      }
54.  }
//// RSA Java Code

```

```

//// RSA Java Code
1.  /**
2.   *
3.   *
4.   * TODO 如果要變更這個產生的檔案的範本，請移至
5.   * 視窗 - 喜好設定 - Java - 程式碼樣式 - 程式碼範本
6.   */

7.  /**
8.   * @author Administrator
9.   *
10.  * TODO 如果要變更這個產生的類別註解的範本，請移至
11.  * 視窗 - 喜好設定 - Java - 程式碼樣式 - 程式碼範本
12.  * @generated "UML 至 EJB (uml2.ejb.transformation)"
13.  */

```

```
14. public class Order {
15.     /**
16.      * <code>member</code> 的註解
17.      * @generated "UML 至 EJB (uml2.ejb.transformation)"
18.      */
19.     private Member member;

20.     /**
21.      * @return 傳回 member。
22.      * @generated "UML 至 EJB (uml2.ejb.transformation)"
23.      */
24.     public Member getMember() {
25.         return member;
26.     }

27.     /**
28.      * @param themember 要設定的 member。
29.      * @generated "UML 至 EJB (uml2.ejb.transformation)"
30.      */
31.     public void setMember(Member themember) {
32.         member = themember;
33.     }

34.     /**
35.      * <code>Id</code> 的註解
36.      * @generated "UML 至 EJB (uml2.ejb.transformation)"
37.      */
38.     private String Id;

39.     /**
40.      * @return 傳回 Id。
41.      * @generated "UML 至 EJB (uml2.ejb.transformation)"
42.      */
43.     public String getId() {
44.         return Id;
45.     }

46.     /**
47.      * @param theId 要設定的 Id。
48.      * @generated "UML 至 EJB (uml2.ejb.transformation)"
49.      */
50.     public void setId(String theId) {
```



```

51.         Id = theId;
52.     }

53.     /**
54.      * @param theId
55.      * @generated "UML 至 EJB (uml2.ejb.transformation)"
56.      */
57.     public void placeOrder(String theId) {
58.         // TODO 自動產生方法 Stub
59.     }
60. }
//// RSA Java Code

```

Q17.3.2 重現類別圖

現在，我們來重現轉成 EJB 專案之後的類別圖，步驟如下：

1. 右擊選取模型瀏覽器中的【Member.java】，並於選單中選取【視覺化 ► 新增至新圖型檔 ► 類別圖】，如圖 Q17-81 所示。

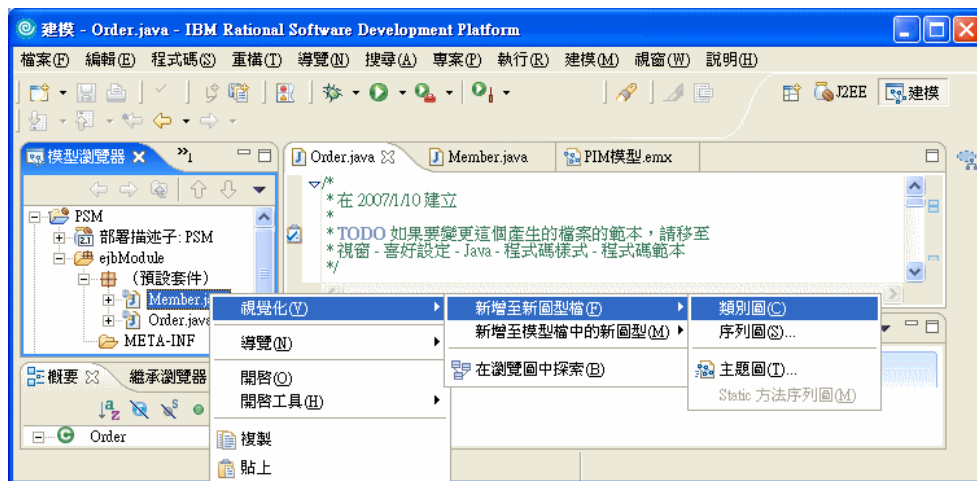


圖 Q17-81: 視覺化

2. 請看 RSA 左上方的模型瀏覽器中，已經新增了一張類別圖，且自動開啓如圖 Q17-82 所示。

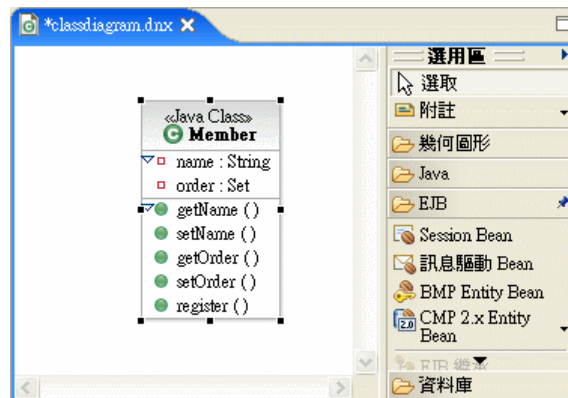


圖 Q17-82: 新建類別圖

3. 再一次右擊選取模型瀏覽器中的另一個【Order.java】，並於選單中選取【視覺化►新增至現行的圖型】，RSA 隨即更新類別圖，如圖 Q17-83 所示。

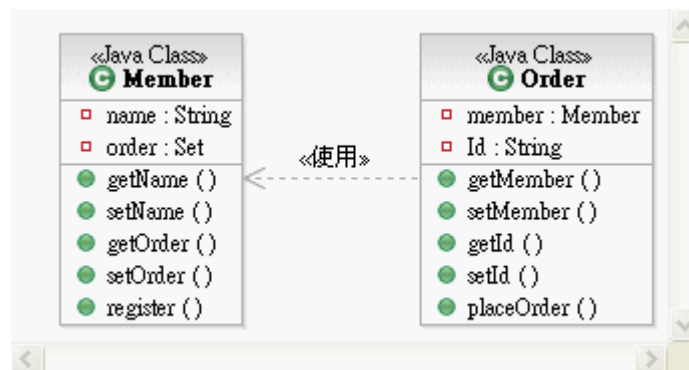


圖 Q17-83: 更新類別圖

4. 再回過頭來開啓原先 PIM 的系統類別圖，相較一下，有很大的不同，如圖 Q17-84 所示。

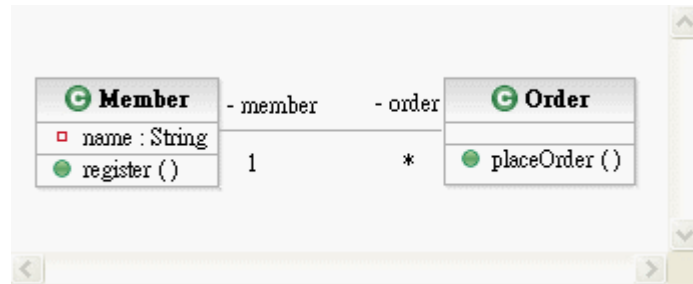


圖 Q17-84: PIM 的系統類別圖

Q17.3.3 新建 PSM 系統類別圖

從重現的類別圖中可以發現，RSA 將 PIM 的類別轉成一般的 Java 類別。所以，現在我們得手動新建 PSM 類別圖，其步驟如下：

1. 右擊選取模型瀏覽器中的【ejbModule】，並於選單中選取【新建►其他】，如圖 Q17-85 所示。

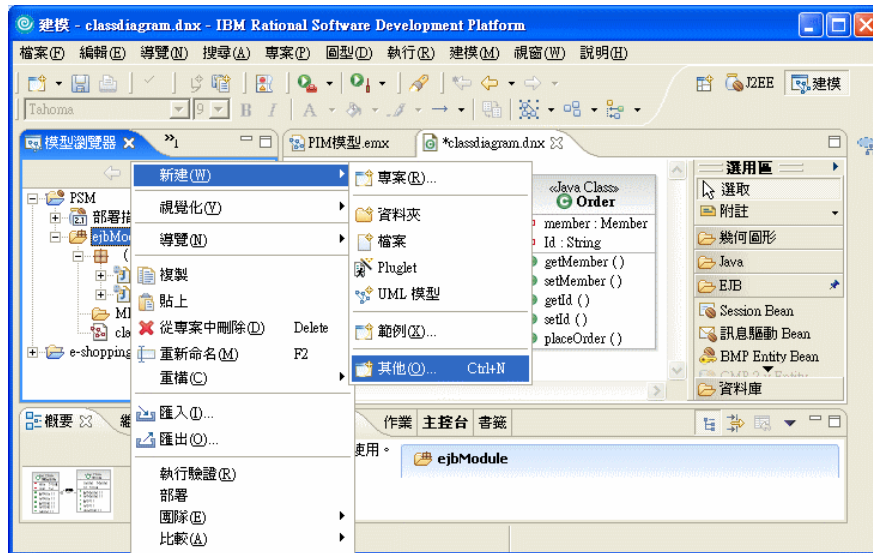


圖 Q17-85: 新建類別圖

2. 請於出現的對話窗中，選取【建模►類別圖】並執行下一步，如圖 Q17-86 所示。

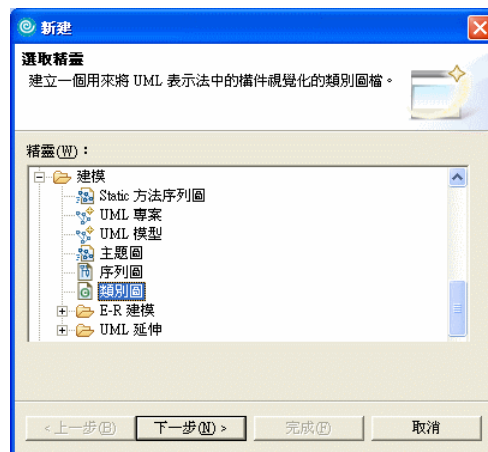


圖 Q17-86: 建模

3. 隨後，請依照對話窗的指示為新建的類別圖命名，並按下【完成】鍵。
4. 現在，我們就可以正式開始建立 PSM 類別圖的內容了。請選取 EJB 選用區裡的【CMP 2.x Entity Bean】，如圖 Q17-87 所示。



圖 Q17-87: EJB 選用區

5. 接著，將游標移至類別圖面空白處，再按一次滑鼠左鍵，隨即出現一建立 Bean 的對話窗。請輸入 Bean 名稱爲【Member】，並按下【完成】鍵，如圖 Q17-88 所示。



圖 Q17-88: 建立 Enterprise Bean

6. 隨後，RSA 立即於類別圖面上，新增了一個如圖 Q17-89 的 Member Entity Bean。

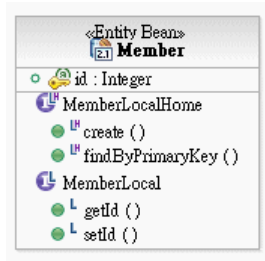


圖 Q17-89: Member Entity Bean

7. 點開模型瀏覽器中的 Java 檔案，依序可見 MemberBean.java、MemberLocal.java 及 MemberLocalHome.java 的程式碼，如下所列。

```

//// RSA Java Code
1. package ejbs;
2. /**
3.  * Bean implementation class for Enterprise Bean: Member
4.  */
5. public abstract class MemberBean implements javax.ejb.EntityBean {
6.     private javax.ejb.EntityContext myEntityCtx;
7.     /**
8.      * setEntityContext
9.      */
10.    public void setEntityContext(javax.ejb.EntityContext ctx) {
11.        myEntityCtx = ctx;
12.    }
13.    /**
14.     * getEntityContext
15.     */
16.    public javax.ejb.EntityContext getEntityContext() {
17.        return myEntityCtx;
18.    }
19.    /**
20.     * unsetEntityContext
21.     */
22.    public void unsetEntityContext() {

```

```
23.         myEntityCtx = null;
24.     }
25.     /**
26.      * ejbCreate
27.      */
28.     public java.lang.Integer ejbCreate(java.lang.Integer id)
29.         throws javax.ejb.CreateException {
30.         setId(id);
31.         return null;
32.     }
33.     /**
34.      * ejbPostCreate
35.      */
36.     public void ejbPostCreate(java.lang.Integer id)
37.         throws javax.ejb.CreateException {
38.     }
39.     /**
40.      * ejbActivate
41.      */
42.     public void ejbActivate() {
43.     }
44.     /**
45.      * ejbLoad
46.      */
47.     public void ejbLoad() {
48.     }
49.     /**
50.      * ejbPassivate
51.      */
52.     public void ejbPassivate() {
53.     }
54.     /**
55.      * ejbRemove
56.      */
57.     public void ejbRemove() throws javax.ejb.RemoveException {
58.     }
59.     /**
60.      * ejbStore
61.      */
62.     public void ejbStore() {
63.     }
64.     /**
65.      * Get accessor for persistent attribute: id
```

```
66.      */
67.      public abstract java.lang.Integer getId();
68.      /**
69.       * Set accessor for persistent attribute: id
70.       */
71.      public abstract void setId(java.lang.Integer newId);
72.  }
//// RSA Java Code
```

```
//// RSA Java Code
1.  package ejbs;
2.  /**
3.   * Local interface for Enterprise Bean: Member
4.   */
5.  public interface MemberLocal extends javax.ejb.EJBLocalObject {
6.      /**
7.       * Get accessor for persistent attribute: id
8.       */
9.      public java.lang.Integer getId();
10.     /**
11.      * Set accessor for persistent attribute: id
12.      */
13.     public void setId(java.lang.Integer newId);
14.  }
//// RSA Java Code
```

```
//// RSA Java Code
1.  package ejbs;
2.  /**
3.   * Local Home interface for Enterprise Bean: Member
4.   */
5.  public interface MemberLocalHome extends javax.ejb.EJBLocalHome {
6.      /**
7.       * Creates an instance from a key for Entity Bean: Member
8.       */
9.      public ejbs.MemberLocal create(java.lang.Integer id)
10.         throws javax.ejb.CreateException;
11.     /**
12.      * Finds an instance using a key for Entity Bean: Member
13.      */
14.     public ejbs.MemberLocal findByPrimaryKey(java.lang.Integer primaryKey)
15.         throws javax.ejb.FinderException;
```



```
16.  }  
//// RSA Java Code
```

Q17.4 RSA 未來可能發展的功能

針對 UML 轉出 EJB3 程式碼的部份，筆者推想，RSA 未來可能發展的功能，如下所述：

- Message-Driven Bean – 雖然，RSA 在 EJB 專案中，有支援 Message-Driven Bean 的圖示及自動產出相關的 Java 程式碼。不過，如果我們打算從 UML 專案轉出 EJB 專案的話，僅能自動轉出 Entity Bean、Stateless Session Bean 及 Stateful Session Bean，並不支援轉出 Message-Driven Bean。
- Entity Bean – 目前，RSA 僅支援到 EJB2，還未支援 EJB3，想當然在 Entity Bean 的相關設定上，都還停留在對於 Home、Local 或 Remote 界面(Interface)的部份。日後，當 RSA 開始支援 EJB3 時，在 Entity Bean 處的設定上頭，極可能會增添跟永續(Persist)機制相關的設定。
- Java 程式碼 – 由於，EJB3 採用 Java5 的@(Annotation)特性，可在 Java 程式碼中註解額外的資訊，簡化了關於 EJB3 實體平台的編碼。所以，RSA 支援 EJB3 之後，自動產出的 Java 程式碼將比現在支援 EJB2 的程式碼要簡單多了。

Q17.4.1 Message-Driven Bean

雖然，RSA 在 EJB 專案中，有支援 Message-Driven Bean 的圖示及自動產出相關的 Java 程式碼。不過，如果我們打算從 UML 專案轉出 EJB 專案的話，僅能自動轉出 Entity Bean、Stateless Session Bean 及 Stateful Session

Bean，並不支援轉出 Message-Driven Bean。

請看圖 Q17-90，一旦我們開啓 EJB 專案之後，可在它的 EJB 選用區中，選用訊息驅動 Bean(Message-Driven Bean)。隨後，再經歷一連串如圖 Q17-91 的對話窗設定之後，RSA 便會自動產出相關的 Java 程式碼。



圖 Q17-90: EJB 選用區



圖 Q17-91: 選擇傳訊服務類型

換言之，從我們先前使用 UML 所建出的 PIM 模式中，RSA 是無法從其中萃取有助產出 Message-Driven Bean 的資訊，所以無法自動轉出 Message-Driven Bean 以及相關的 Java 程式碼。

因此，從 UML 的 PIM 設計轉出 EJB2 的 PSM 設計，且進入到 EJB2 專案的設計環境之後，我們就能夠繼續添加 EJB 實體平台相關的細部設計，隨即自動產出更多且細緻的 Java 程式碼。

Q17.4.2 Entity Bean

目前，RSA 僅支援到 EJB2，還未支援 EJB3，想當然在 Entity Bean 的相關設定上，都還停留在對於 Home、Local 或 Remote 界面(Interface)的部份。請看圖 Q17-92 的視窗，這是 RSA 裡用來指示使用者輸入 Entity Bean 相關設定的對話窗。

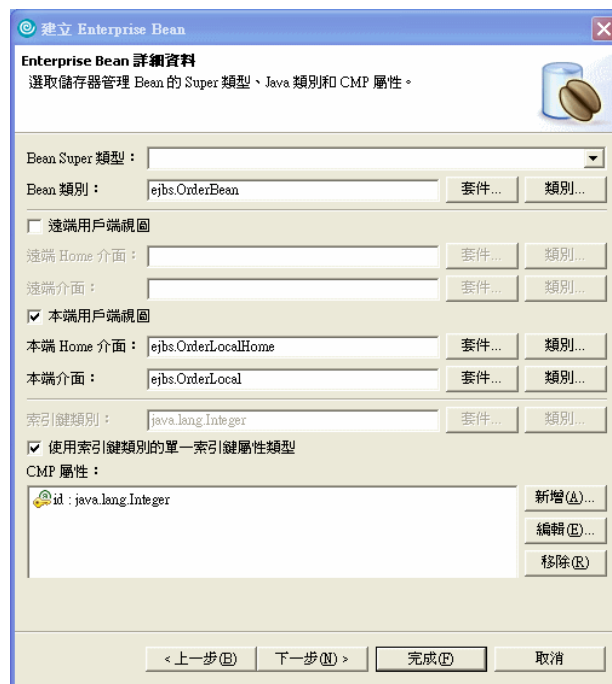


圖 Q17-92: RSA 裡的 Entity Bean 設定

由於，EJB3 已經不再需要應用程式實作繁瑣的 EJB3 界面，僅需透過簡單的 @Local、@Remote 或 @WebService 關鍵字的宣告，即可獲得相關的平台服務。因此，日後 RSA 開始支援 EJB3 時，在 Entity Bean 處的設定上頭，極可能會增添跟永續(Persist)機制相關的設定。

比方說，Taylor MDA 在支援 EJB3 的 Entity Bean 時，便提供了如圖 Q17-93 的設定，讓開發人員可以透過此處的設定，自動產出瑣碎的 Java 程式碼。可以想見，RSA 日後也應該會提供諸如此類的設定機制。

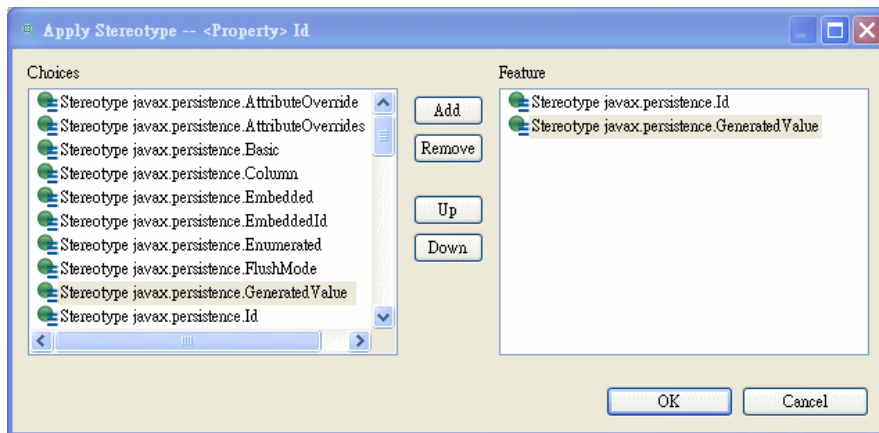


圖 Q17-93: Taylor MDA 裡的 Entity Bean 設定

Taylor MDA 是一套開放源碼(Open Source)的 Eclipse 插件，目前支援 UML 的類別圖及用例圖，且可以讀取類別圖自動產出 EJB3 的程式碼。有興趣的讀友可以上 Taylor MDA 的網站(<http://taylor.sourceforge.net/>)下載安裝，試試看它最新的 EJB3 支援。

Q17.4.3 Java 程式碼

由於，EJB3 採用 Java5 的@(Annotation)特性，可在 Java 程式碼中註解

額外的資訊，簡化了關於 EJB3 實體平台的編碼。所以，RSA 支援 EJB3 之後，自動產出的 Java 程式碼將比現在支援 EJB2 的程式碼要簡單多了。

比方說，在 EJB2 實體平台裡，宣告一個提供 Remote 界面的 Stateless SB，至少得產出 Stateless SB 的 Bean 實體、以及 Home 界面與 Remote 界面總共三部份的 Java 程式碼，如下列程式碼所示。

```
// Stateless
public class RegisterUCBean implements SessionBean{
    public RegisterUCBean() {
        // ...
    };
    public setMemberInfo(MemberInfo memberInfo) {
        // ...
    };

    // EJB2
    public void ejbCreate() throws CreateException {
        // ...
    };
    public void ejbRemove() {
        // ...
    };
    public void ejbActivate() {
        // ...
    };
    public void ejbPassivate() {
        // ...
    };
    public void setSessionContext(SessionContext ctx) {
        // ...
    };
};
}
```

```
// Home
public interface RegisterUCHome extends EJBHome{
    // ...
}
```

```
// Remote
public interface RegisterUC extends EJBObject{
    // ...
}
```

然而，EJB3 因為採用 Java5 的@特性，以及實體平台的改善之後，不僅 Stateless Session Bean 的 Java 程式碼大幅簡化，同時也僅需編寫 RegisterUC Stateless SB 的 Bean 實體以及 Remote 界面的 Java 程式碼。如下表所示：

```
@Stateless
public class RegisterUCBean implements RegisterUC{
    public RegisterUCBean() {
        // ...
    };
    public setMemberInfo(MemberInfo memberInfo) {
        // ...
    };
}
```

```
@Remote
public interface RegisterUC{
    void setMemberInfo(MemberInfo memberInfo);
}
```

Q18 CIM-1、2、3 並沒有絕對的順序？

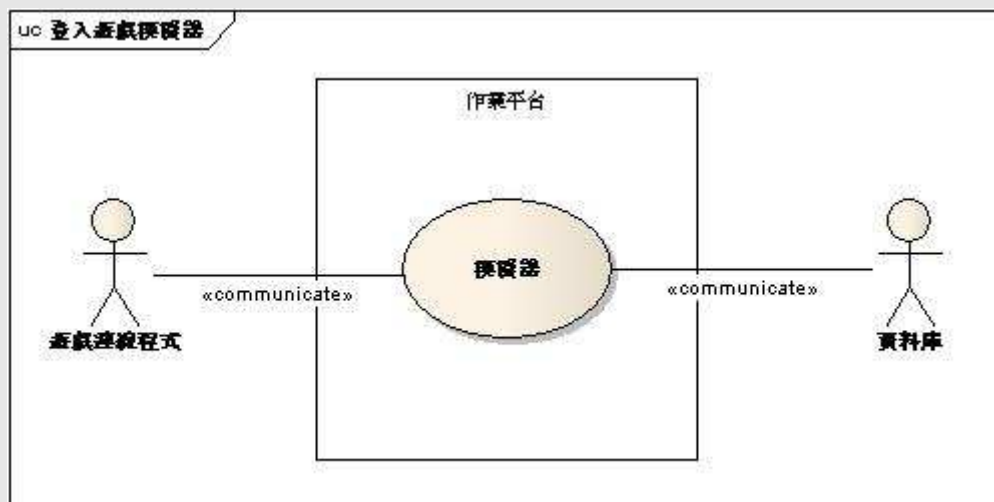
CIM-1、2、3並沒有絕對的順序？

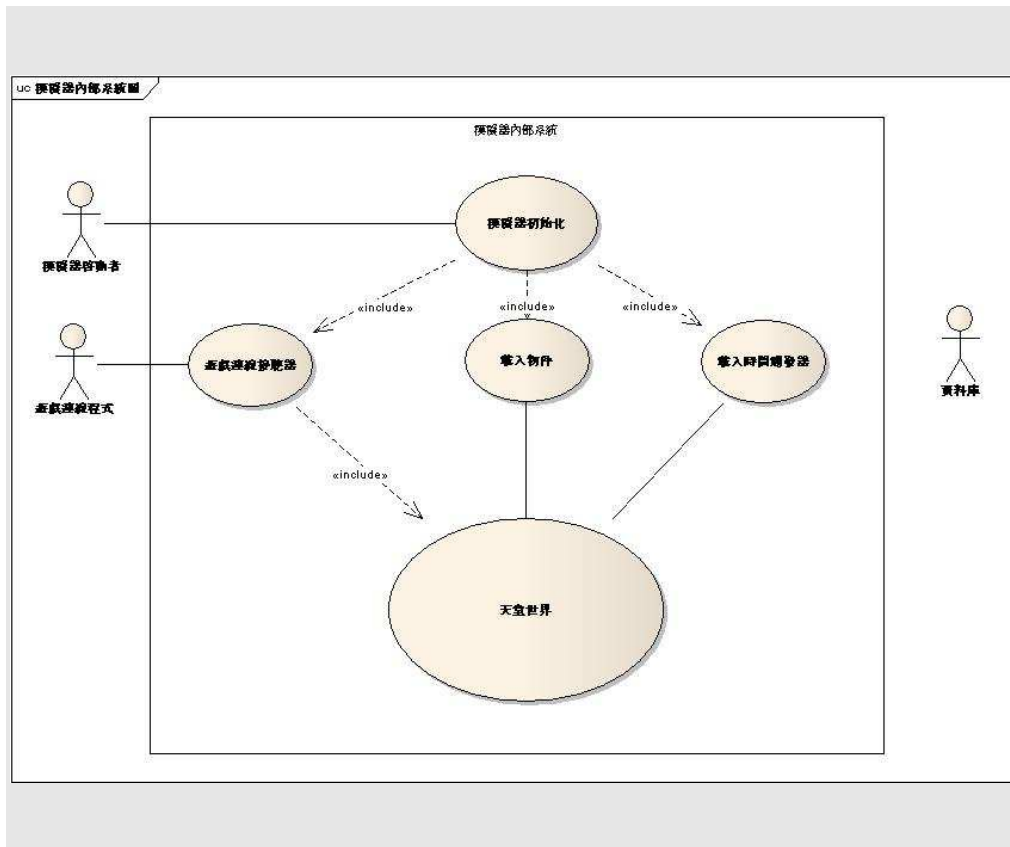
(mailliw對於「寫給SA的UML/MDA實務手冊」一書的提問)

我最近正在學習畫圖，我一直以為，要先把系統的範圍給畫出來。例如：我想做一個遊戲伺服器，那我的做法會是先畫出模擬器的範圍，這應該算是CIM-3吧，然後再分析模擬器內部的企業流程CIM-1與企業規則CIM-2。

還是說其實 CIM-1、2、3並沒有絕對的順序？這應該是要問的問題。

請參考下圖，依序為：登入遊戲模擬器、模擬器內部系統圖。





此處，我先重製了 mailiw 寄來的原圖，因為原圖看起來模糊，所以重製兩圖如下：

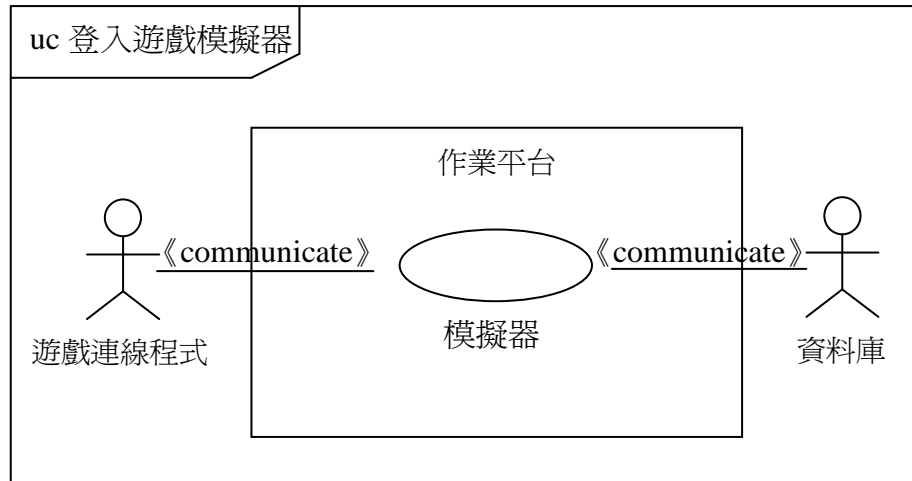


圖 Q18-1: uc 登入遊戲模擬器

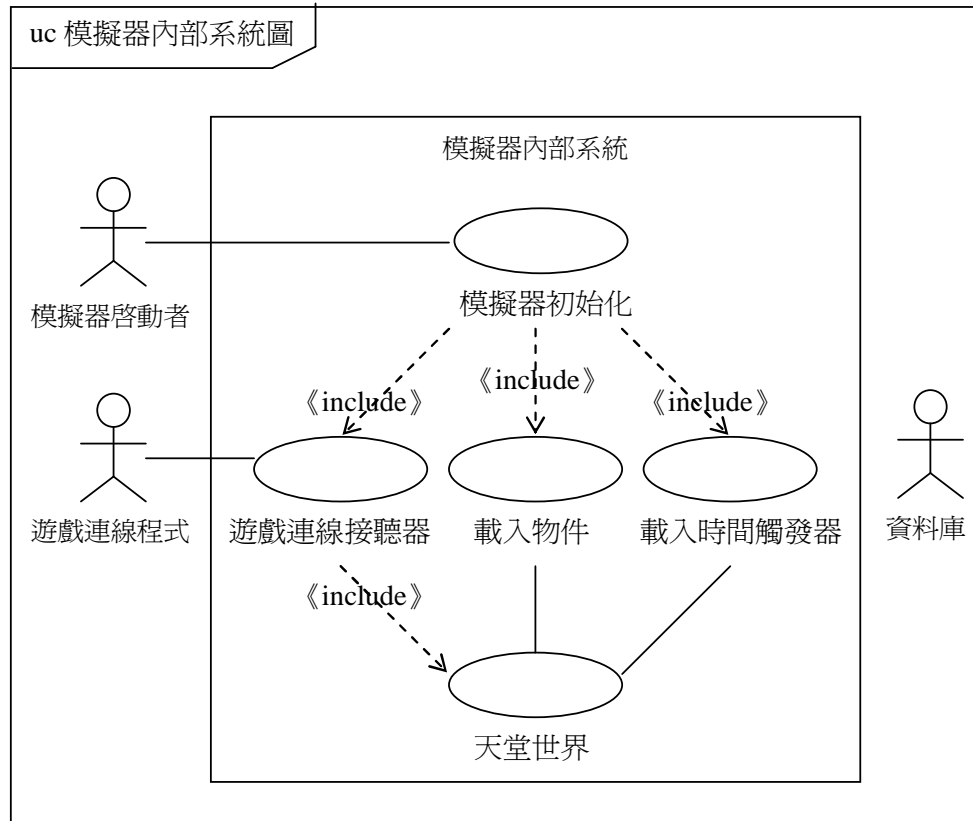


圖 Q18-2: uc 模擬器內部系統圖

CIM-1~3 以及 PIM-1~4，這七個步驟確實沒有依序的關係。

我將分析步驟編號成 CIM-1~3、PIM-1~4，一共七個分析步驟，這樣的編號是我從研究 DoD AF(美國國防部系統架構框架)所得來的靈感。而我這樣做，有一個主因是，在應用 UML 圖時，同一款 UML 圖可以有不同的用途，而初學者經常會有所混淆。

最常見的例子是，使用案例圖(Use Case Diagram)可以用來表達企業

流程，也可以用來表達系統服務；如果在專案中，有進行企業流程的分析，也做了系統服務的分析的話，我發現 UML 的初學者經常會對此有困擾，往往會混淆為什麼同樣是使用案例圖，但是一張叫做企業用例圖 (Business Use Case Diagram)，但另一張卻又叫做系統用例圖 (System Use Case Diagram)。

爲了降低這樣的困擾，我試著不以 UML 圖爲主，而是以分析設計步驟爲主，教導成員每一個步驟的重點爲何，以及採用的 UML 圖爲何？以此來降低 UML 初學者的學習門檻，同時大幅節省了專案成員教育訓練的時程。而且有了這樣的思維，專案成員也可以認知到，UML 圖並不是非用不可的關鍵，每一個分析步驟所要呈現的觀點才是真正的重點，所以日後有更好的技術時，當然可以將任何一款 UML 圖取而代之。

此外，在進行專案時，一開始就可以評估哪幾個分析步驟一定要做，哪幾個分析步驟可以視情況添加，這樣有助於時程的評估，也有助於經費的預算。對於專案成員的心理，也有極佳的安定性，專案成員會很清楚知道現在做到哪個步驟了，接下來會進行哪個步驟，以及步驟之間的相關性爲何。再者，可以將每一個分析步驟視爲一個元件，嘗試不同的排列組合，當然這個部份還是要顧問的配合，才能爲專案打造出剪裁合宜的開發流程。

對於沒有明確企業範圍的資訊系統，如此處的遊戲模擬器、嵌入式系統等，其實可以省略 CIM1~2，直接進行 CIM-3 以及後續的 PIM-1~4，請參考「寫給 SA 的 UML/MDA 實務手冊」的第 11 章，有類似的範例。

再回頭來看遊戲模擬器的例子，如果省略 CIM1~2、直接進行 CIM-3 的話，我會先繪製出初步的系統 UC 圖(CIM-3)，步驟如下：

1. 先繪出系統外的系統參與者。此處，我們關切的資訊系統為「遊戲模擬器」，因此位於此系統外部，且會使用系統的人類使用者，或者是會與系統連線的其他系統，皆為「系統參與者」(System Actor)。

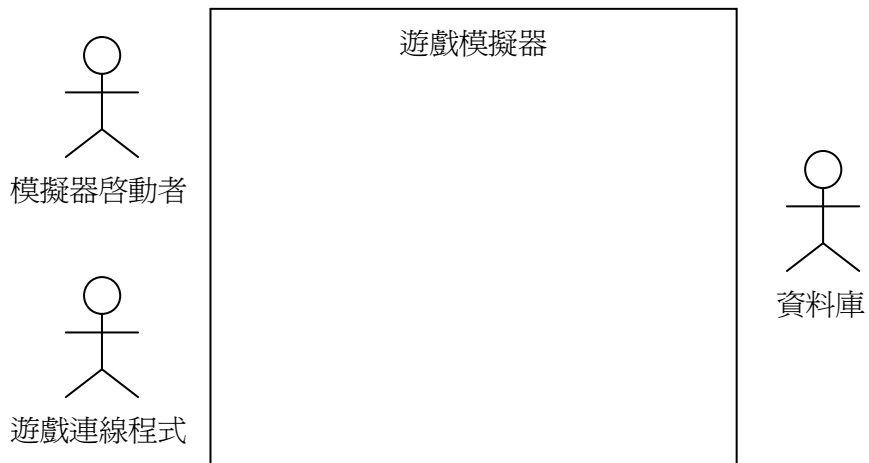


圖 Q18-3: 遊戲模擬器

2. 加上重要的系統 UC，並且連接系統參與者。啟動 UC 的參與者，特稱為「啟動者」(initiator)，其餘不具有啟動特質的參與者，可稱之為「支援者」(support)。直接操作電腦的使用者，通常就是系統 UC 的啟動者。而且在系統 UC 執行期間，有時會需要連線其他系統以取得協助，這些連線系統就是支援者。SA 在繪圖時，可以採用帶箭頭關係線，讓啟動者連線指向 UC，UC 連線指向支援者。這樣一來，從圖面上就可以明確分辨出啟動者與支援者。

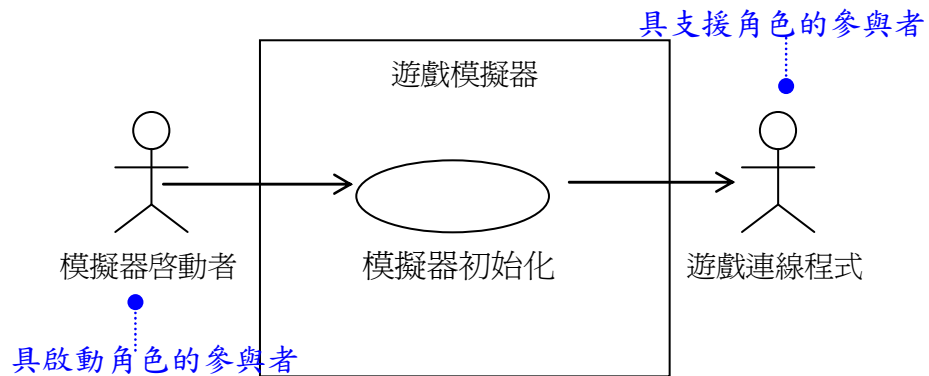


圖 Q18-4: Actor

3. 如果有重用 UC 對話的情況發生，可以考慮使用「包含」(include)或「擴充」(extend)關係。兩個 UC 之間的「包含」關係，用以表示某一個 UC 的對話流程中，包含著另一個 UC 的對話流程。兩個 UC 之間的「擴充」關係，用以表示某一個 UC 的對話流程，可能會依條件臨時插入另一個 UC 的對話流程中。

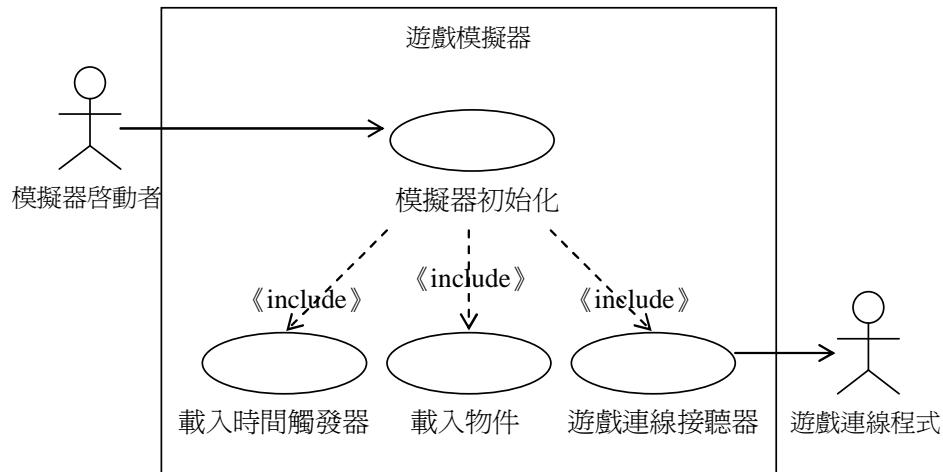


圖 Q18-5: 使用案例之間的關係

下述是我的想法及疑問：

1. UC圖不宜過分複雜化，如果不畫出UC之間的關係，也能達成相同的溝通結果，那不一定得將UC之間的關係線畫出。畢竟UC圖是最靠近使用者的一款圖，或許也是UML圖中最容易看懂的圖，所以可以的話，我會盡量保持UC圖的簡潔性。至於，許多繁複的細節，其實可以記錄到UC敘述，藉此保持UC圖的單純性，同時將細節封裝到UC敘述中。
2. 第一張「登入遊戲模擬器」的圖，其實我不大看得懂，為什麼「模擬器」會是個UC呢？所以，我的回答沒處理這張圖，直接跳過了。Sorry！
3. 第二張「模擬器內部系統圖」，我也有些困惑。第一個困惑是「遊戲連線接聽器」，這不太像是UC。另外，「載入物件」和「載入時間觸發器」這兩個UC，就沒什麼問題；因為UC是使用的過程，有動作

的，所以適合以動詞為起始來命名。

4. 再者，第二張圖中，「載入物件」和「載入時間觸發器」UC 之間有使用到「結合關係」(association)，這也讓我有困惑。一般而言，UC 之間的關係線有：一般化關係(generalization)、包含關係和擴充關係。
5. 總之，關於第一張圖的「模擬器」UC 以及第二張圖的「遊戲連線接聽器」和「天堂世界」UC，以及結合關係線的使用，因為不知道設計者的意涵，我也不便胡亂猜測，所以也就沒有動手修圖了。

Q19 EA 之操作示範

※※

- Q19.1 簡易的開發程序 ---- 138
- Q19.2 新增專案 ---- 140
- Q19.3 繪製使用案例圖 ---- 142
- Q19.4 繪製類別圖 ---- 148
- Q19.5 繪製循序圖 ---- 162
- Q19.6 產生 Java 程式碼 ---- 169
- Q19.7 產生文件 ---- 173

Q19.1 簡易的開發程序

實務上，使用案例圖及敘述、類別圖與循序圖三者之搭配，幾乎是 UML 專案的基本型，所以在分工或外包的設計文檔中，通常少不了這三款 UML 圖。常見的開發程序是，並行建構使用案例圖文與類別圖，接著才建構循序圖以及按圖編碼，如圖 Q19-1 所示。

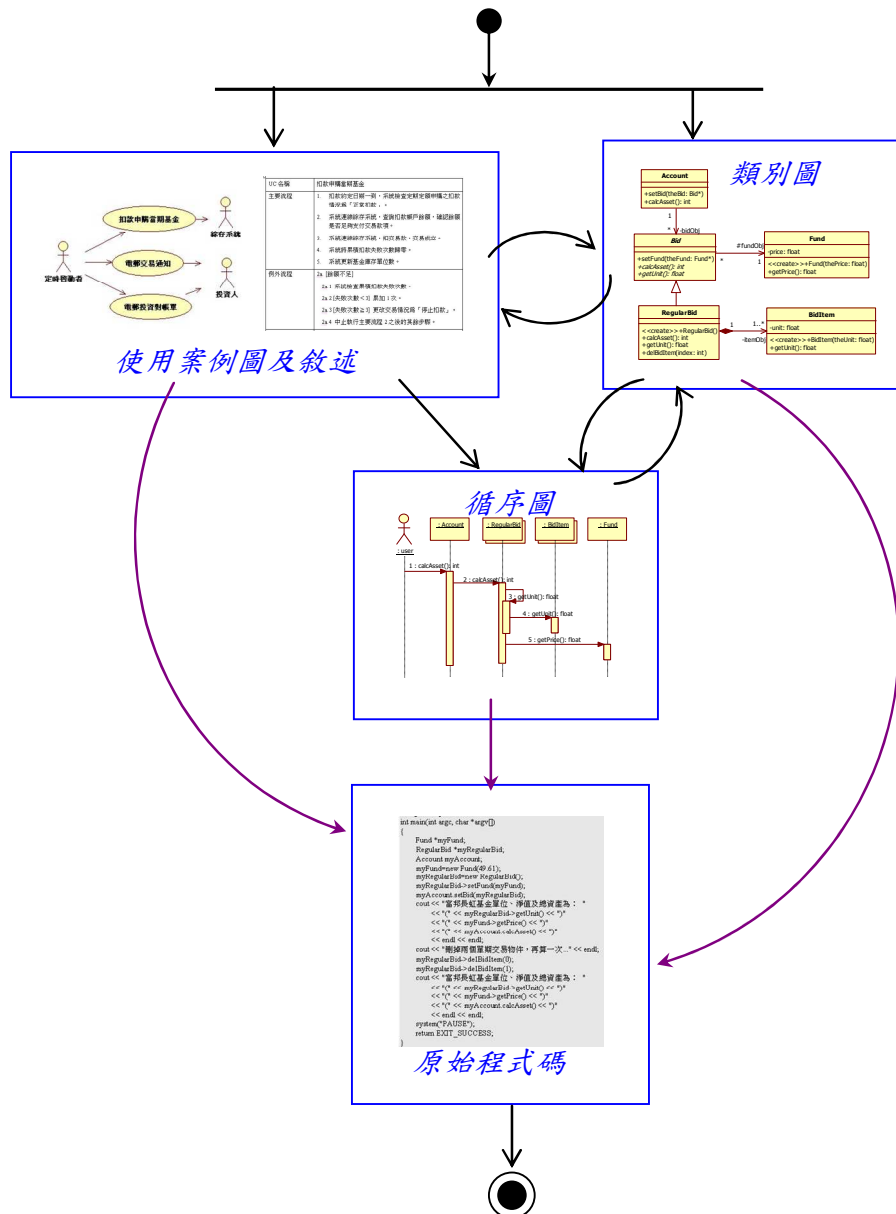


圖 Q19-1: 簡易的開發程序

一個系統只有一個內部結構，而且系統對外提供的所有的服務，都僅依賴這個穩定的內部結構所支撐。然而，每一項服務的運作方式皆不同，所以雖然系統僅有一個靜態結構，可以卻有很多個動態行為。

因此，透過 UML 圖來呈現系統的狀況時，一個系統僅有一張呈現系統內部結構的類別圖，而且無論使用案例圖中有多少個使用案例。但是，每一個使用案例至少對應一張循序圖，呈現出系統執行使用案例期間，其內部的一群物件互動的運作情況。

再者，類別圖通常不是一次就能夠設計完全，而是透過一個又一個的使用案例，以及一張又一張的循序圖，三者經過多次循環更新的歷程後，類別圖才逐步成形且穩定下來。

在接下來的小節中，我們會以一個極簡單的「計算總資產」使用案例為範例，展示如何使用 EA(Enterprise Architect)建立 UML 模式，並且依此 UML 模式自動產出 Java 的程式碼。

Q19.2 新增專案

一執行 EA 時，會出現如圖 Q19-2 的起始頁(start page)，此時，您可以於管理專案(manage projects)區塊底下，選擇新增專案(create a new project)。

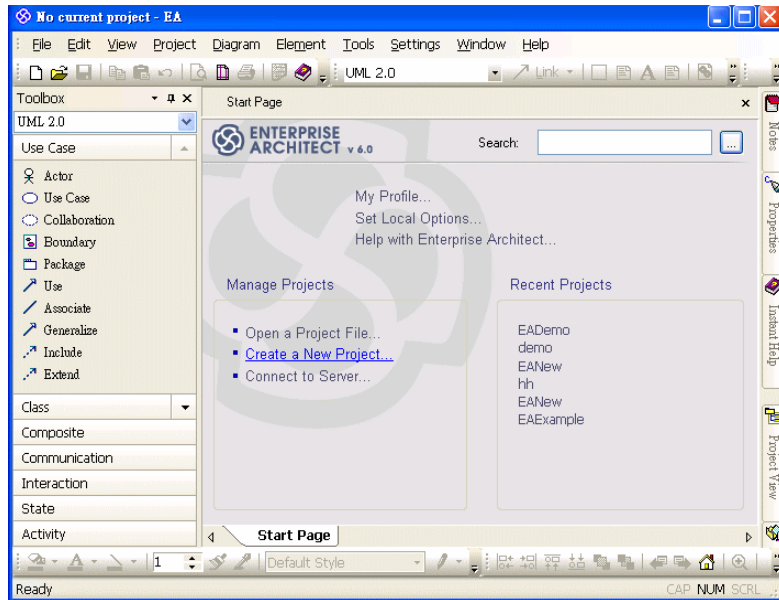


圖 Q19-2: 新增專案

接著，請您輸入專案名稱及指定檔案路徑，如圖 Q19-3 所示。此範例的專案名稱爲 FundSys(基金系統)。

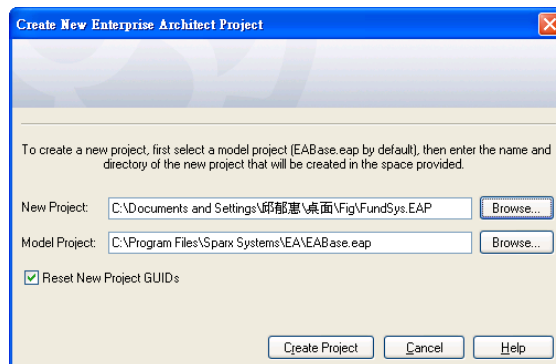


圖 Q19-3: 專案名稱及路徑

一旦，新增專案成功後，再回到起始頁，您會發最新專案(recent projects)區塊底下，已經新增了一個名為 FundSys(基金系統)的專案項目了，如圖 Q19-4 所示。



圖 Q19-4: 新增基金系統(FundSys)專案

Q19.3 繪製使用案例圖

由於，EA 會自動備妥一張空白的使用案例圖，所以您只要依照下述步驟開啓使用案例圖(use case diagram)即可。操作步驟如下：

1. 在專案視區(project view)中，雙擊【Views►Use Case View►Use Case Model►Use Case Model】，如圖 Q19-5 所示。

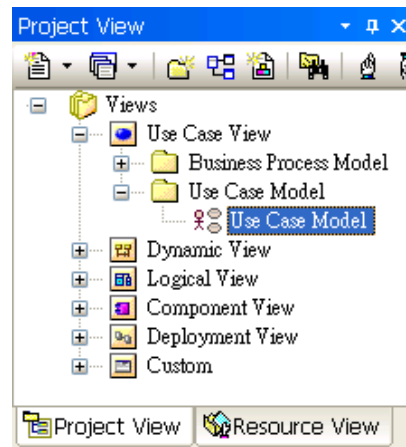


圖 Q19-5: Use Case Model

2. Use Case Model 是 EA 自動備妥的使用案例圖，雙擊開啓如圖 Q19-6 的空白使用案例圖。

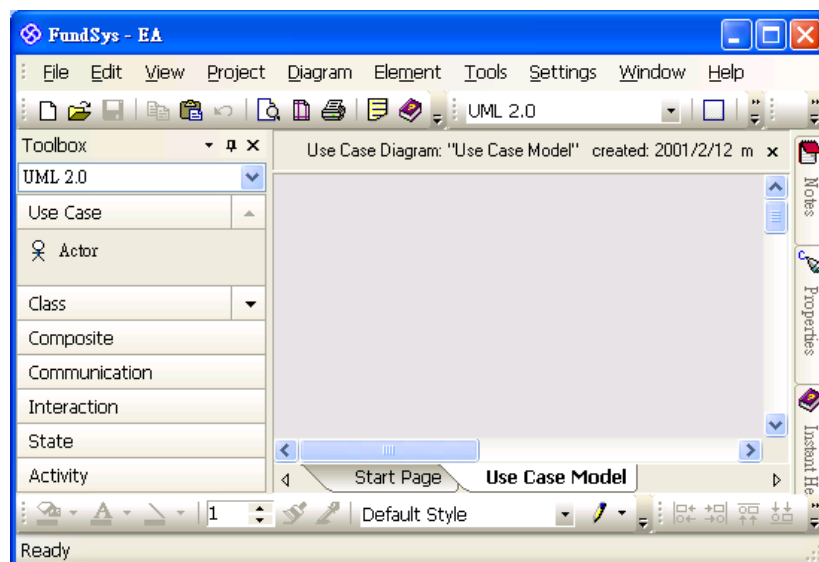


圖 Q19-6: 開啓使用案例圖

3. 點選工具箱裡的橢圓形 Use Case(使用案例)圖示，如圖 Q19-7 所示。

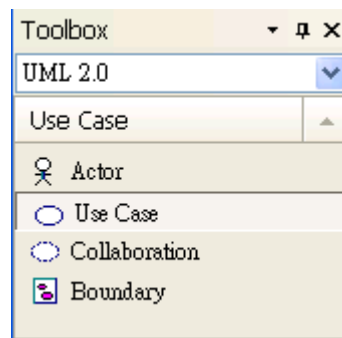


圖 Q19-7: 點選使用案例(use case)圖示

4. 隨後，在圖面空白處再點一次，此時 EA 將出現填寫使用案例細節的性質表(properties)。請於 Name 處填入使用案例名稱「計算總金額」，而且可以於下方的 Nodes 處填入簡單的使用案例敘述，如圖 Q19-8 所示。

UseCase : 計算總資產

General Responsibilities Constraints Link Scenario Files

Name: 計算總資產

Stereotype: ☐ Abstract ☐

Author: Status: Proposed

Scope: Public Complexity: Easy

Alias: Language: <none>

Keywords:

Phase: 1.0 Version: 1.0

Notes:

<使用案例敘述>

<簡述>投資人以基金目前淨值估算總資產

<主要流程>

1. 螢幕上秀出基金單位數及淨值。
2. 計算出總資產。
3. 螢幕上秀出總資產。

<企業規則>

1. 資產=基金淨值*單位數

圖 Q19-8：填寫使用案例敘述

5. 關閉使用案例的性質表後，EA 會自動更新使用案例圖面，為使用案例標示出更新後的名稱，如圖 Q19-9 所示。

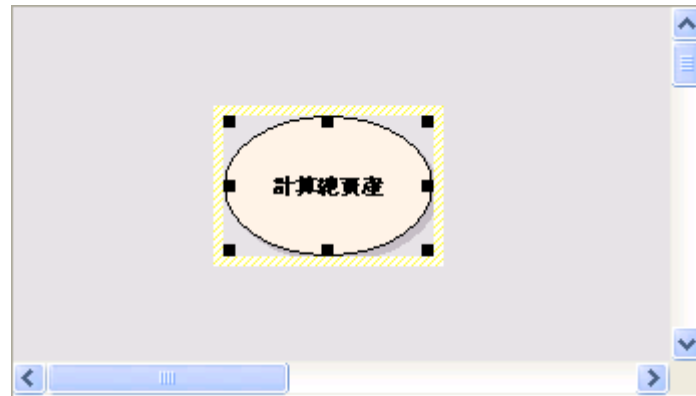


圖 Q19-9: 計算總資產之使用案例

- 點選工具箱(toolbox)裡的人形 Actor(參與者)圖示，如圖 Q19-10 所示。

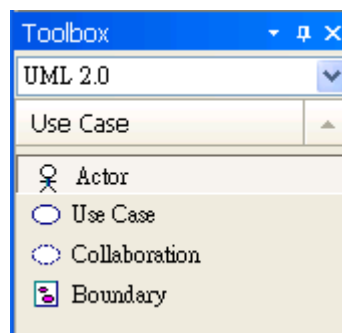


圖 Q19-10: 點選參與者(actor)圖示

- 請為於參與者性質表的 Name 空格中，填入「投資人」。關閉性質表後，EA 將更新畫面，如圖 Q19-11 所示。

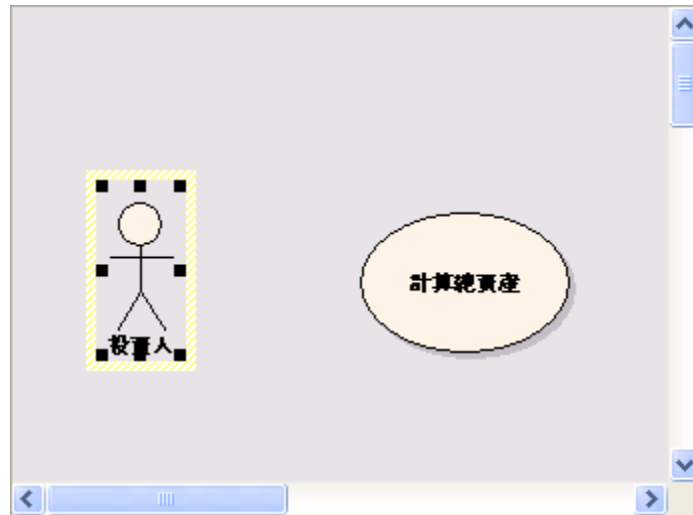


圖 Q19-11: 新增投資人參與者

8. 最後，請點選工具箱裡的實線 Association(結合關係)圖示，我們
要來建立參與者與使用案例之間的關係線，如圖 Q19-12 所示。

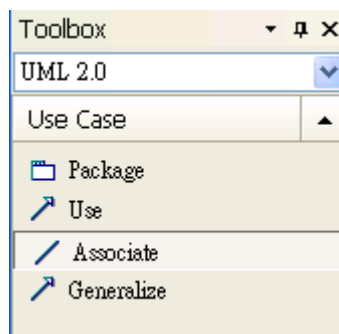


圖 Q19-12: 點選結合關係圖示

9. 隨後，點選參與者並拖曳至使用案例處放開，建立出兩者之間的

關係線，如圖 Q19-13 所示。

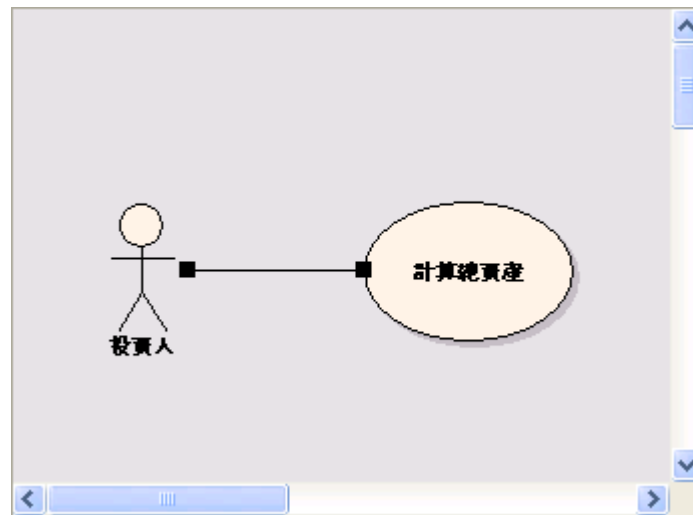


圖 Q19-13: 新增結合關係

Q19.4 繪製類別圖

同樣地，EA 也是會自動備妥一張空白的類別圖，所以您只要依照下述步驟開啓類別圖(class diagram)即可。操作步驟如下：

1. 在專案視區中，雙擊【Views►Logical View►Logical Model►Logical Model】，如圖 Q19-14 所示，就可以開啓 EA 備妥的空白類別圖了。

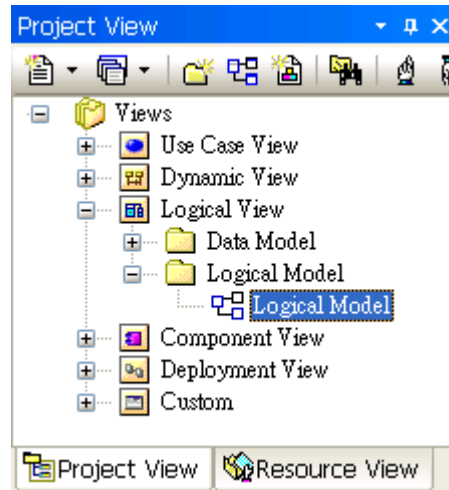


圖 Q19-14: Logical Model

2. 點選工具箱裡的三格矩形 Class(類別)圖示，如圖 Q19-15 所示。

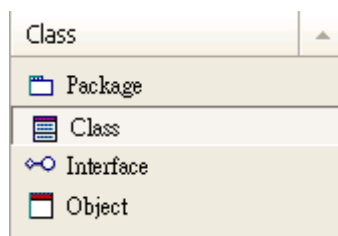


圖 Q19-15: 點選類別(class)圖示

3. 隨後，在圖面空白處再點一次，請於類別性質表的 Name 處填入類別名稱「Fund」(基金)，如圖 Q19-16 所示。

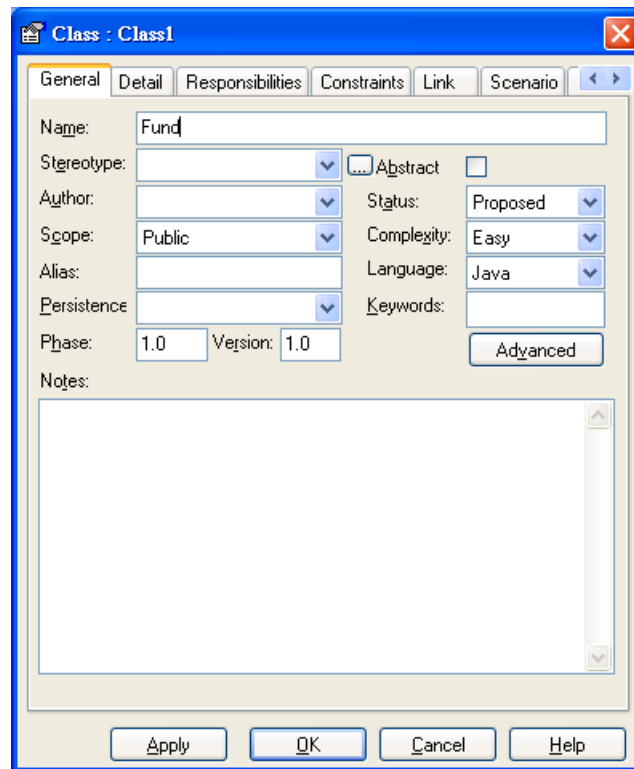


圖 Q19-16: 填寫類別細節

4. 接著，點選 Detail 頁籤，按下 Attributes(屬性)按鍵，準備新增屬性，如圖 Q19-17 所示。

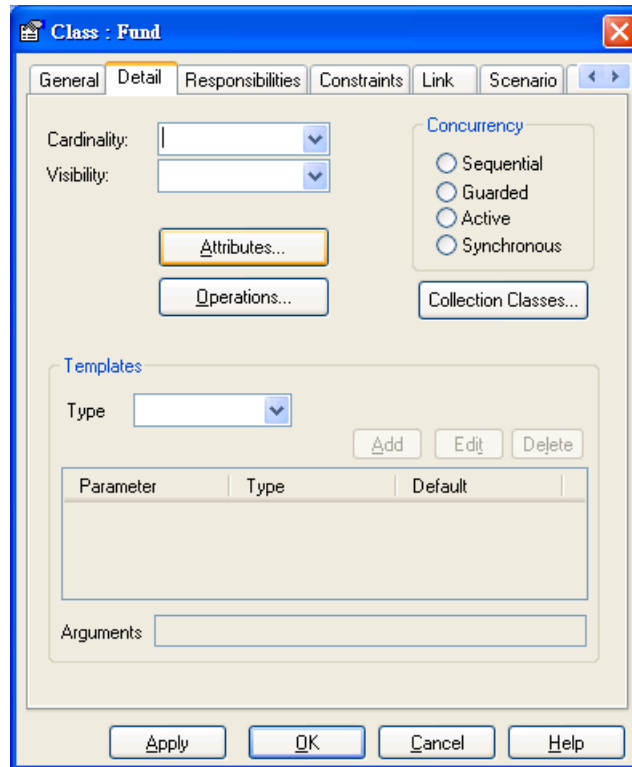


圖 Q19-17: 按下 Attributes 按鍵

5. 請於屬性性質表的 Name 處填入屬性名稱「price」(基金淨值)，並且選取資料型態為「float」，如圖 Q19-18 所示。

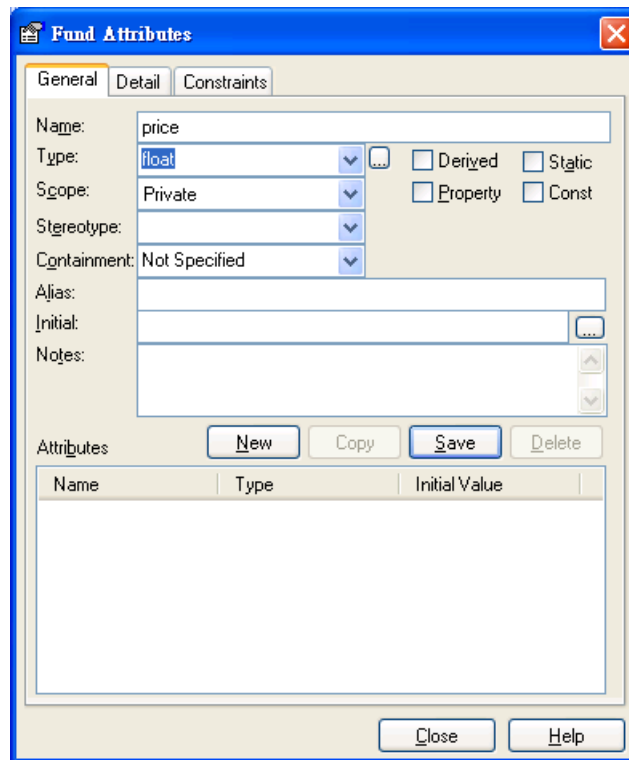


圖 Q19-18: 填入屬性名稱及資料型態

6. 記得按下 Save 鍵，EA 才會將剛才新增的屬性存入，如圖 Q19-19 所示。

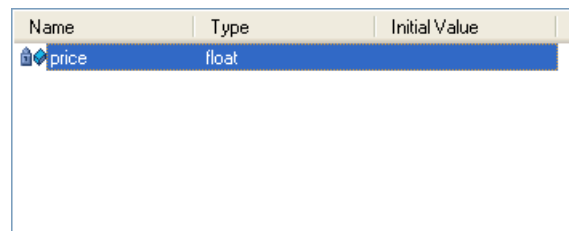


圖 Q19-19: 新增一項屬性了

7. 接著按下 Close 鍵，回到類別圖面，可以發現 EA 已經自動更新圖面了，如圖 Q19-20 所示。

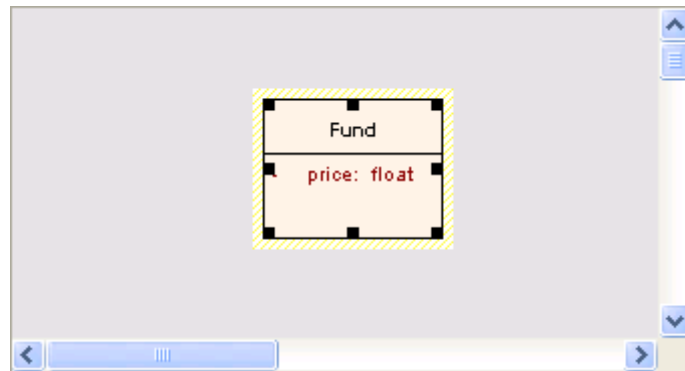


圖 Q19-20: Fund 類別

8. 雙擊 Fund 類別，再度開啓性質表，並按下類別性質表 Detail 頁籤的 Operations(操作)按鍵，來開啓操作性質表。
9. 請於操作性質表的 Name 處填入操作名稱「getPrice」，並且選取回傳型態(return type)為「float」，接著按下 Save 鍵，新增了一項操作，如圖 Q19-21 所示。

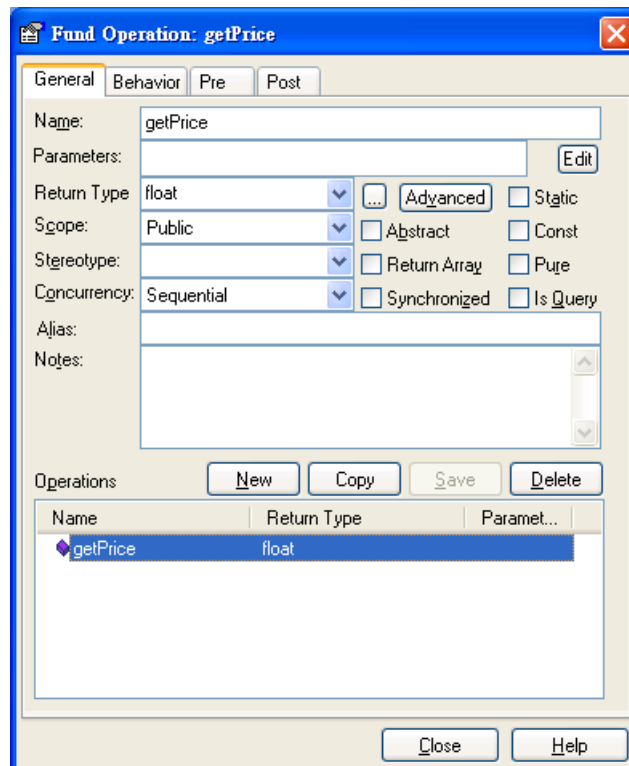


圖 Q19-21: 操作性質表

10. 接著，我們還要新增一個建構元(constructor)。所以，請按下操作性質表中的 **New** 按鈕，並執行如下細節：
- 填寫 Name 為「Fund」。
 - 填寫 Parameters(參數)為「thePrice:float」
 - 選取 Stereotype 為「create」，代表此操作為建構元。
 - 最後，按下 **Save** 鍵，新增了一項建構元，如圖 Q19-22 所示。

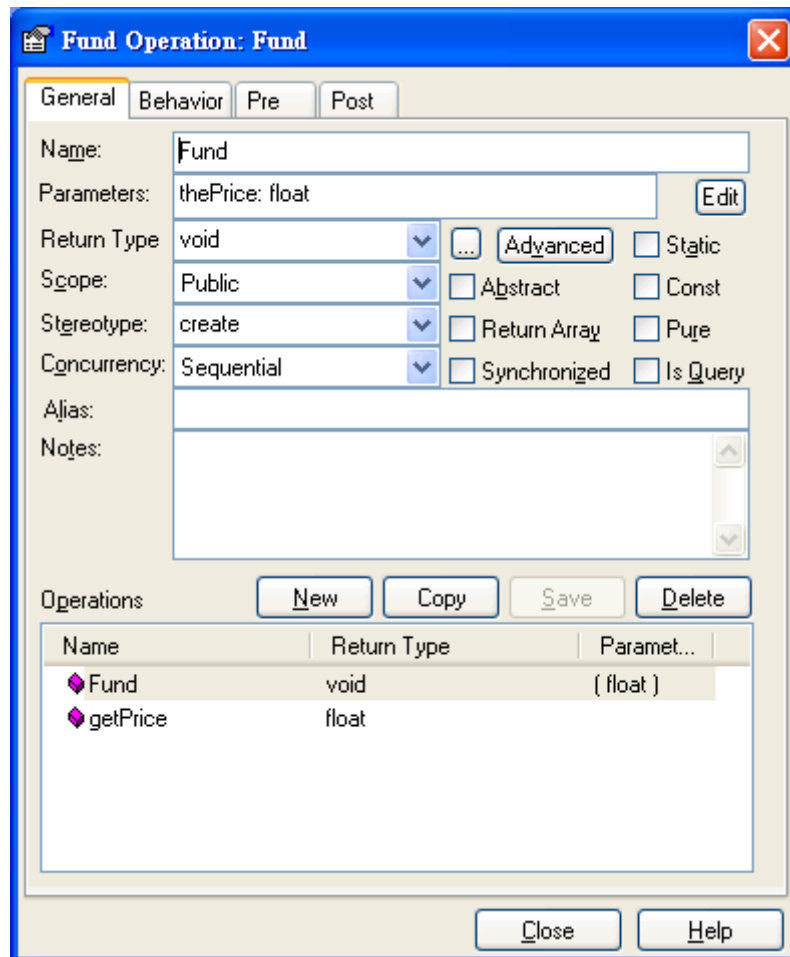


圖 Q19-22: 新增建構元

11. 按下 Close 鍵，回到類別圖面，可以發現 EA 已經自動更新圖面了，如圖 Q19-23 所示。

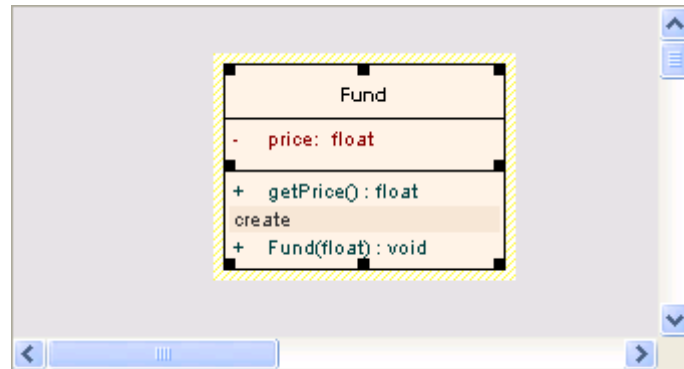


圖 Q19-23: 更新圖面

12. 現在，請您依照上述步驟新增另一個 Bid(申購)類別，如圖 Q19-24 所示。

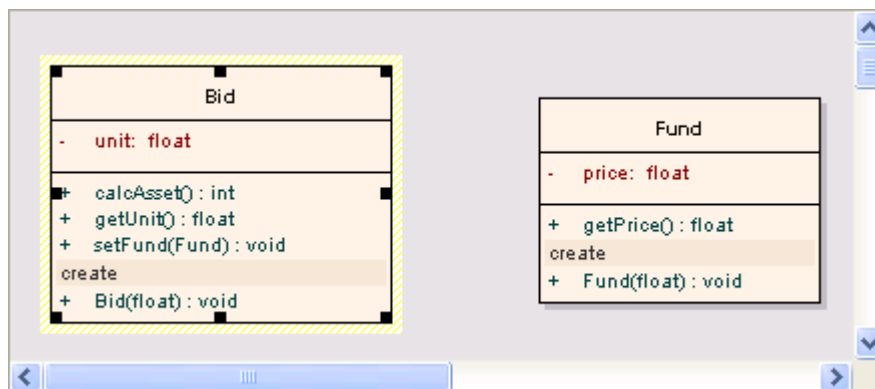


圖 Q19-24: 新增 Bid(申購)類別

13. 接著，我們要繼續來建立類別之間的結合關係(association)。請點選工具箱裡的實線 Associate(結合關係)圖示，如圖 Q19-25 所示。

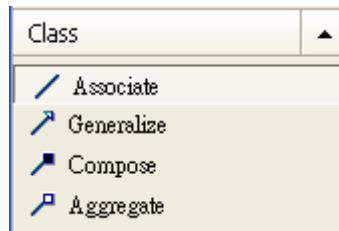


圖 Q19-25: 點選 Associate

14. 隨後，點選 Bid 抽象類別並拖曳至 Fund 類別處放開，建立出兩者之間的結合關係，如圖 Q19-26 所示。

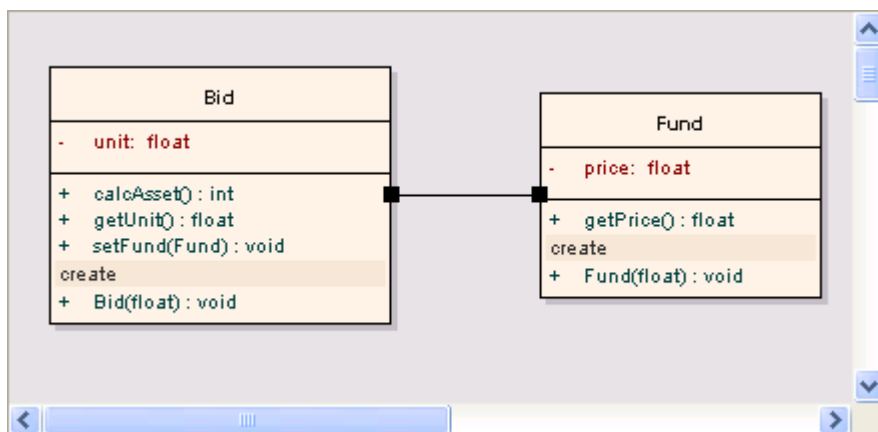


圖 Q19-26: 新增結合關係

15. 雙擊結合關係以便開啓其性質表，並於 Direction 處挑選 Source->Destination，建立起由 Bid 指向 Fund 的結合關係線，如圖 Q19-27 所示。

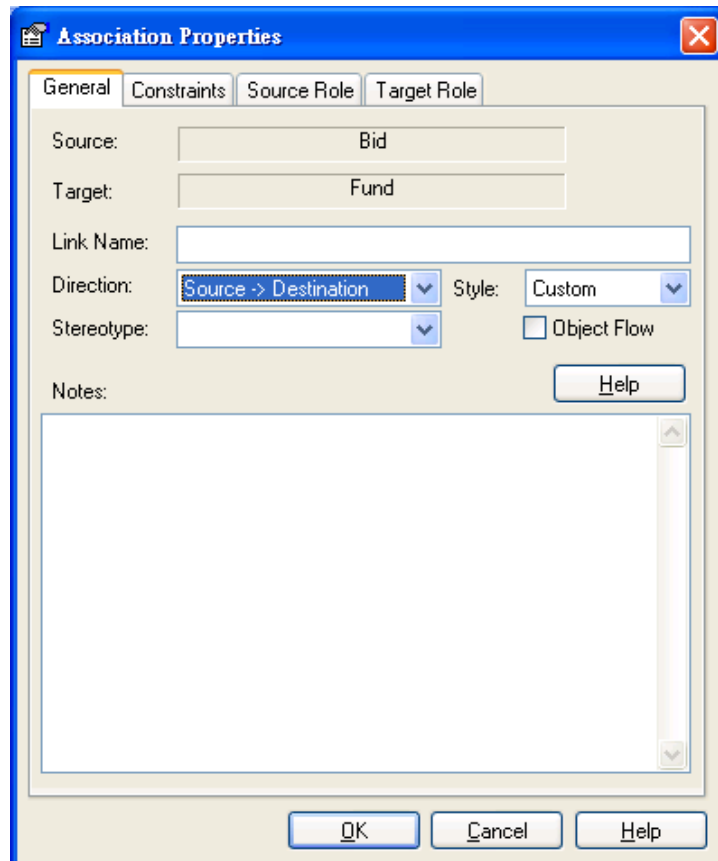


圖 Q19-27: 建立結合關係的方向性

16. 結合關係的設定還沒結束，請開啓 Source Role 頁籤，並於 Multiplicity(個體數目)處挑選*(星號)，如圖 Q19-28 所示。

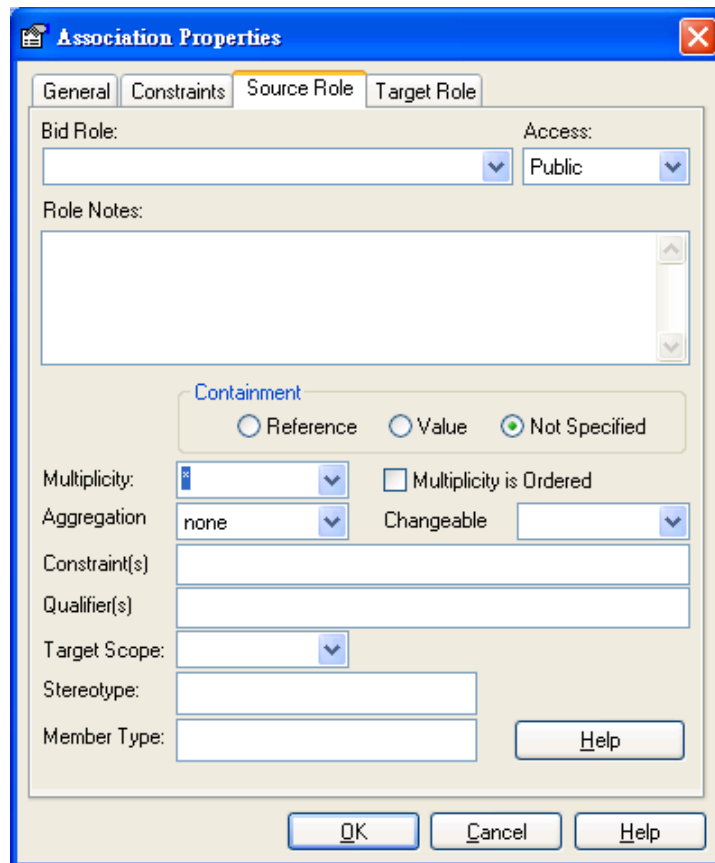


圖 Q19-28: Source Role

17. 再開啓 Target Role 頁籤，並於名稱處填寫 fundObj，Access(存取)處選取 Private(私有)，最後在 Multiplicity 處挑選 1，如圖 Q19-29 所示。

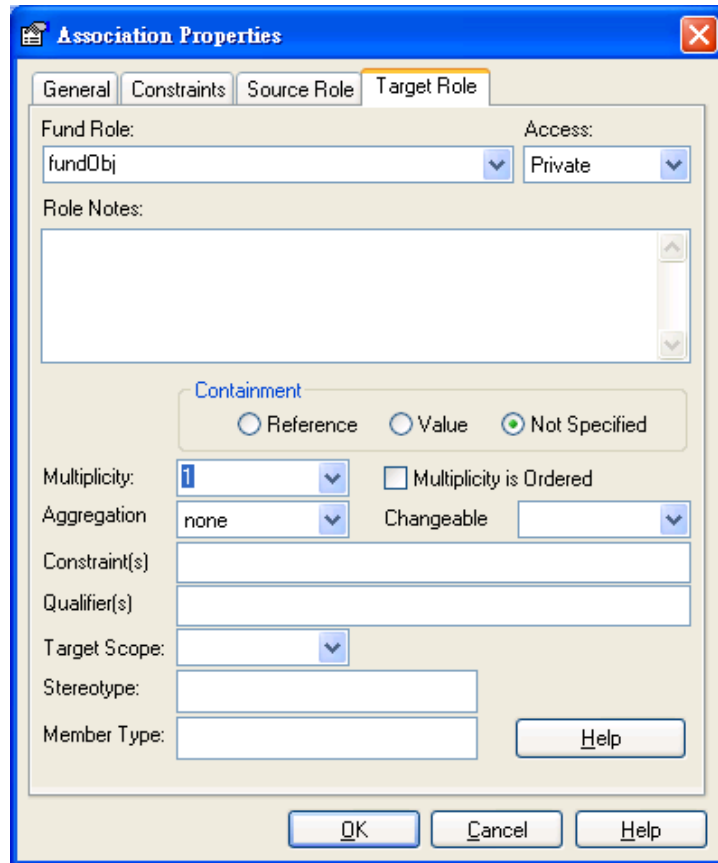


圖 Q19-29: Target Role

18. 按下 OK 鍵，回到類別圖面，可以發現 EA 已經自動更新圖面了，如圖 Q19-30 所示。

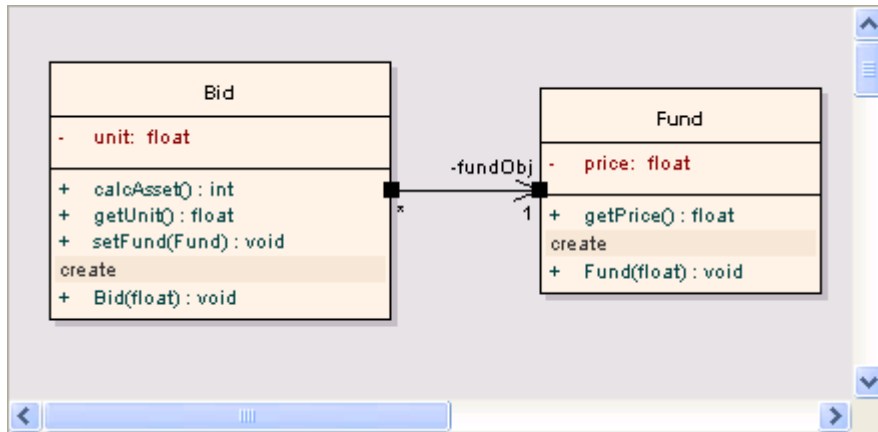


圖 Q19-30: 更新圖面

19. 請您依照上述步驟新增 Account(帳戶)類別，並且建立起 Account 與 Bid 之間的結合關係，完成如圖 Q19-31 的類別圖。

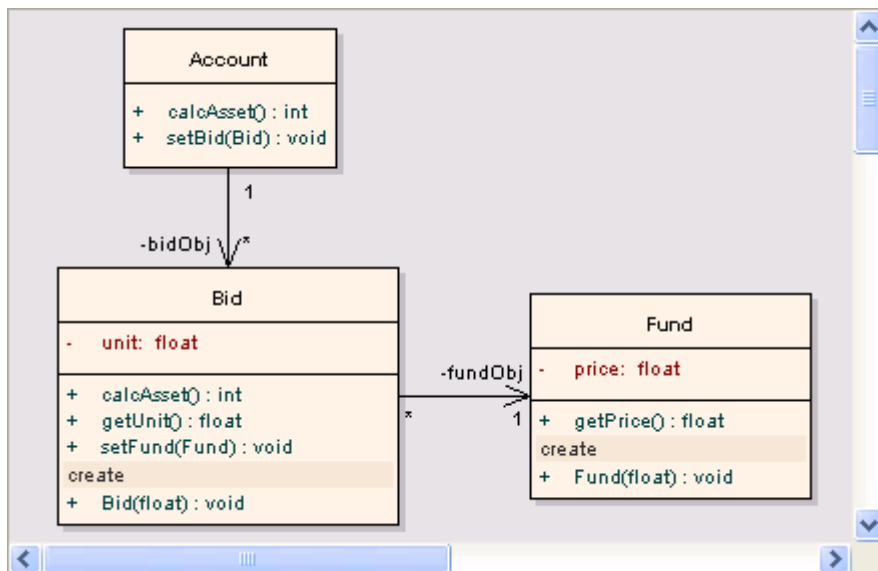


圖 Q19-31: 類別圖

Q19.5 繪製循序圖

EA 已經自動備妥一張空白的循序圖，所以您只要依照下述步驟開啓循序圖(sequence diagram)即可。操作步驟如下：

1. 在專案視區中，點選【Views►Dynamic View►Interactions►Interactions】並更名為「計算總資產 UC 之主要流程」，如圖 Q19-32 所示。

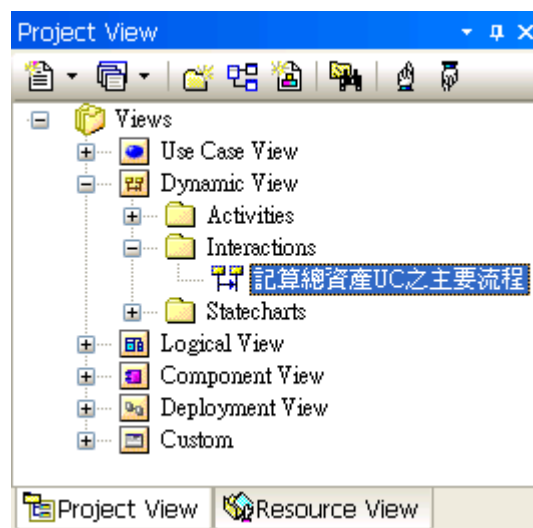


圖 Q19-32: 為循序圖更名

2. 接著，開啓【Views►Use Case View►Use Case Model】，點選「投資人」參與者類別，並拖曳至圖面空白處，放開滑鼠左鍵，如圖 Q19-33 所示。

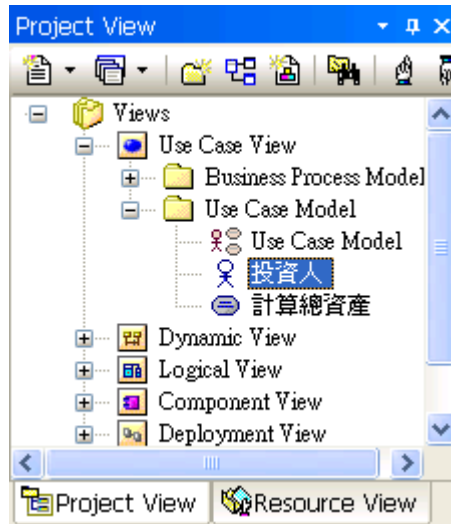


圖 Q19-33: 點選投資人

3. 此時，EA 將出現如圖 Q19-34 的視窗，請選擇「as Instance of Element (Object)」，才能在循序圖內新增一個物件。

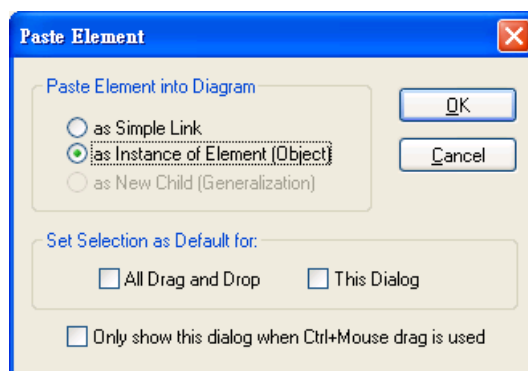


圖 Q19-34: 點選 as Instance of Element (Object)

4. 按下 OK 鍵之後，EA 自動更新圖面，出現了一個投資人物件，如圖 Q19-35 所示。

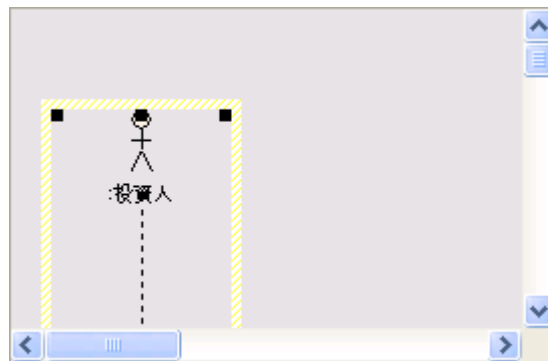


圖 Q19-35: 新增投資人物件

5. 依照上述步驟，開啓專案視區中的【Views►Logical View►Logical Model】，將底下的 Account、Bid 和 Fund 類別依序拖曳到圖面空白處，新增出三個物件，如圖 Q19-36 所示。

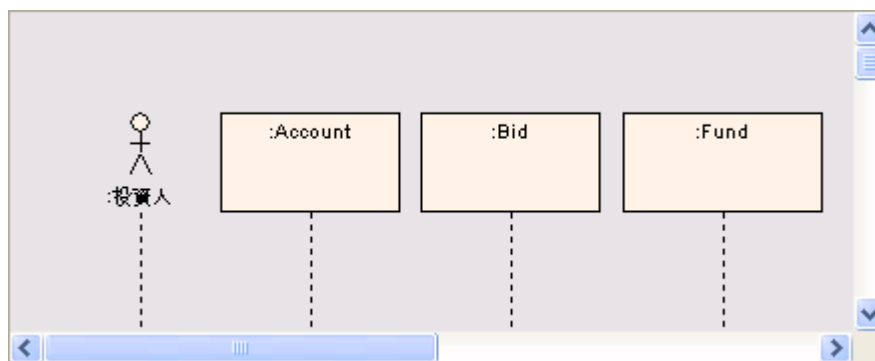


圖 Q19-36: 新增另外三個物件

6. 點選工具箱裡的箭頭實線 Call(呼叫訊息)圖示，如圖 Q19-37 所示。

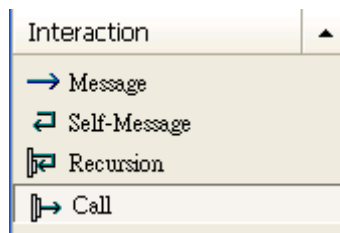


圖 Q19-37: Call

7. 隨後，點選投資人物件存活線並拖曳至 Account 物件存活線處放開。此時，EA 將出現如圖 Q19-38 的視窗，請填入訊息名稱爲「//create」，並且選取 Lifecycle 爲「New」。我們可以在訊息名稱前加上//(雙斜線)，標記此訊息並未實際對應操作。

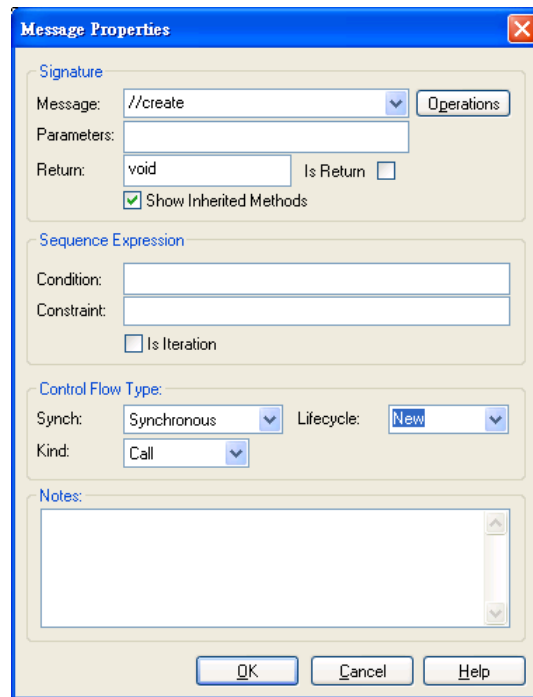


圖 Q19-38: //create

8. 按下 OK 鍵之後，EA 會自動更新圖面，如圖 Q19-39 所示。

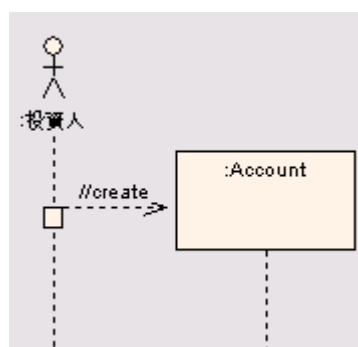


圖 Q19-39: 誕生訊息

9. 接著，再建立投資人物件與 Fund 物件之間的誕生訊息。請於 Message 處選取 Fund(float)，並且選取 Lifecycle 為「New」，如圖 Q19-40 所示。

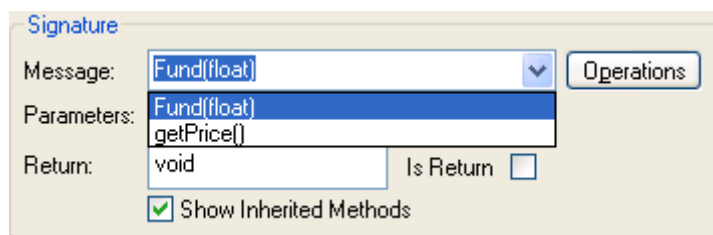


圖 Q19-40: 選取 Fund(float)

10. 按下 OK 鍵之後，EA 會自動更新圖面，如圖 Q19-41 所示。

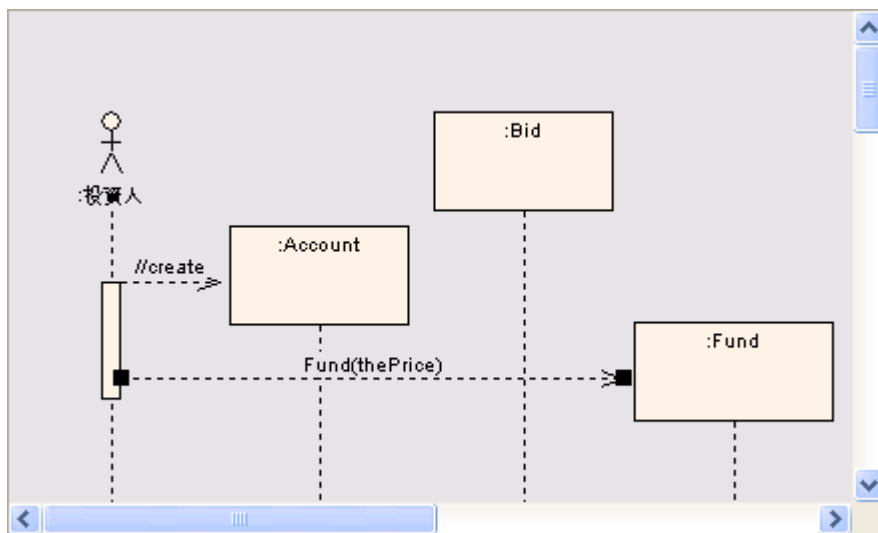


圖 Q19-41: Fund(float)

11. 依照上述步驟產生投資人與 Bid 物件之間的誕生訊息，以及 setFund 訊息。特別注意的是，setFund 是一般的呼叫訊息，也就是對應一般的操作，而非對應建構元，所以在性質表的 Lifecycle 處保持空白，不需選取「New」。這樣一來，setFund 訊息線就會出現實心箭頭，如圖 Q19-42 所示。

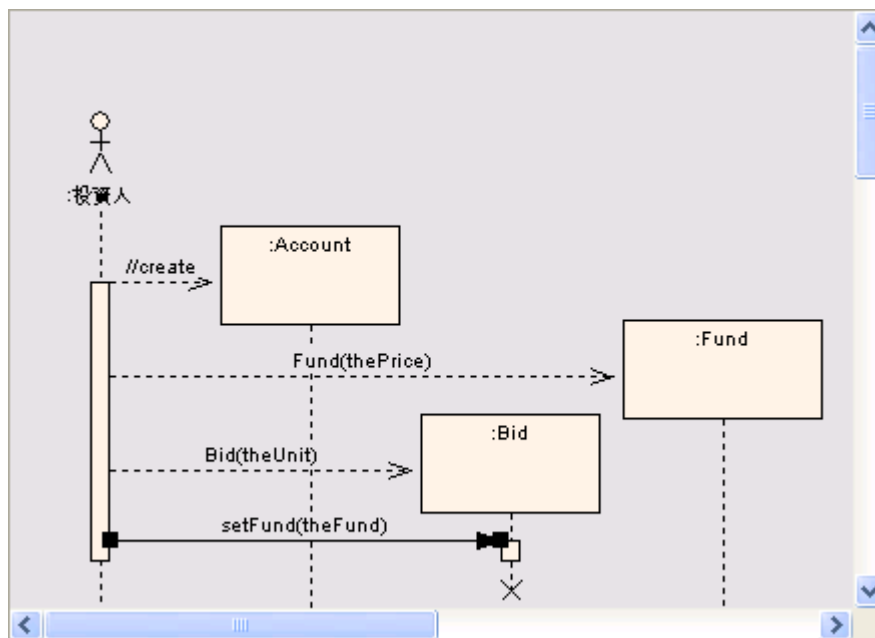


圖 Q19-42: 呼叫訊息

12. 依照上述步驟完成整張循序圖，如圖 Q19-43 所示。

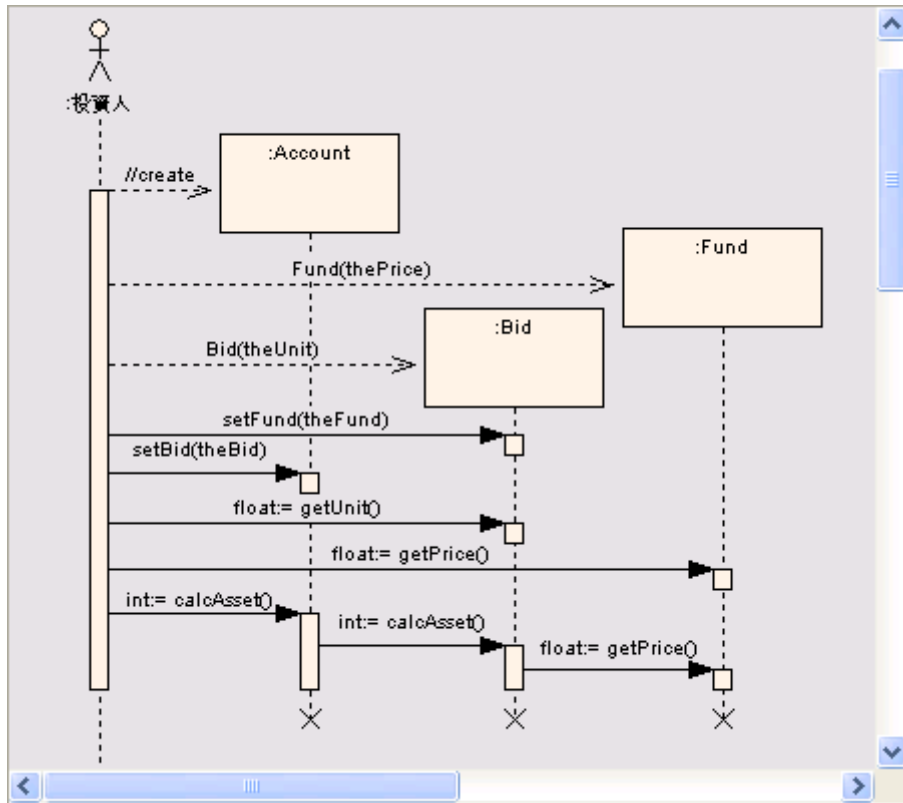


圖 Q19-43: 循序圖

Q19.6 產生 Java 程式碼

EA 提供 Java、C++、C#等多項程式語言的產碼器，可讀取類別圖，轉出原始程式碼。所以，針對前面的類別圖，您可以執行下述步驟，自動產出 Java 程式碼：

1. 回到專案視區，開啓【Views►Logical View►Logical Model►Logical Model】底下的類別圖。
2. 執行主選單的【Project►Source Code Engineering►Generate

Package Source Code】後，將出現如圖 Q19-44 的視窗，請按下 Generate 按鈕。

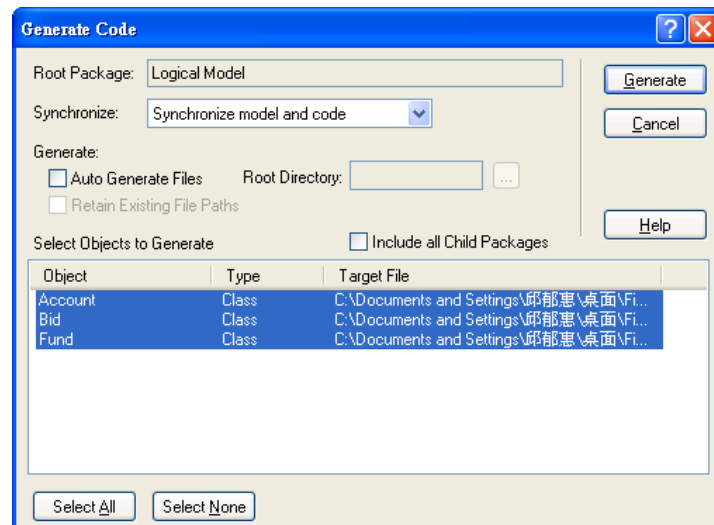


圖 Q19-44: 選取類別

3. 最後，EA 將自動產出如圖 Q19-45~47 的 Java 程式碼。

```
package Logical Model;
```

```
/**
```

```
 * @created 17-十二月-2007 下午 10:24:49
```

```
 * @version 1.0
```

```
 */
```

```
public class Account {
```

```
    private Bid bidObj;
```

```
    public Account(){
```

```
    }
```



```
public void finalize() throws Throwable {  
  
    }  
  
    public int calcAsset(){  
        return 0;  
    }  
  
    /**  
     *  
     * @param theBid    theBid  
     */  
    public void setBid(Bid theBid){  
  
    }  
  
}
```

圖 Q19-45: Account.java

```
package Logical Model;  
  
/**  
 * @version 1.0  
 * @created 18-十二月-2007 上午 12:19:29  
 */  
public class Bid {  
  
    private float unit;  
    private Fund fundObj;  
  
    public Bid(){  
  
    }  
  
    public void finalize() throws Throwable {  
  
    }  
  
    /**  
     *  
     * @param theUnit
```

```
    */  
    public void Bid(float theUnit){  
  
    }  
  
    public int calcAsset(){  
        return 0;  
    }  
  
    public float getUnit(){  
        return null;  
    }  
  
    /**  
     *  
     * @param theFund  
     */  
    public void setFund(Fund theFund){  
  
    }  
  
}
```

圖 Q19-46: Bid.java

```
package Logical Model;  
  
/**  
 * @version 1.0  
 * @created 18-十二月-2007 上午 12:19:29  
 */  
public class Fund {  
  
    private float price;  
    public Fund(){  
  
    }  
  
    public void finalize() throws Throwable {  
  
    }  
  
}
```

```
/**
 *
 * @param thePrice
 */
public void Fund(float thePrice){

}

public float getPrice(){
    return null;
}
}
```

圖 Q19-47: Fund.java

Q19.7 產生文件

EA 擁有強大的文件產生器，可以支援我們產出多樣的文件。此處，我們以常見的 HTML 文件為例，說明產出步驟：

1. 回到專案視區，點選您想要產出文件的目錄，此處以最上層的 View 目錄為例，如圖 Q19-48 所示。

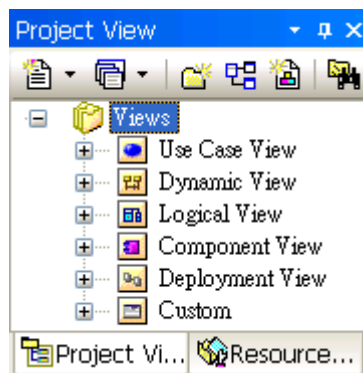


圖 Q19-48: View 目錄

2. 執行主選單的【Project►Documentation►HTML Report】後，將出現如圖 Q19-49 的視窗。請指定輸出路徑，並按下 Generate 按鈕。

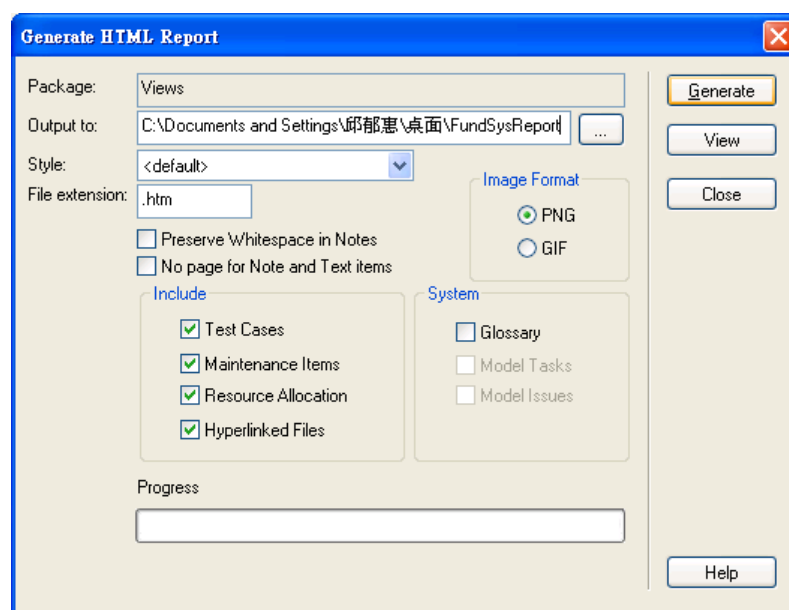


圖 Q19-49: 產生 HTML 文件

3. 隨後，在我們指定的路徑下，EA 幫我們建立了兩個子目錄，如圖 Q19-50 所示。

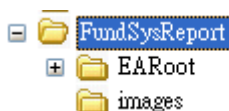


圖 Q19-50: 檔案結構

4. 進入 EARoot 目錄，雙擊 contents.htm 檔案之後，開啓如圖 Q19-51 的網頁。您可以對應 EA 專案視區底下的所有目錄，都出現在這個網頁中。



圖 Q19-51: contents.htm

5. 循著連結開啓【View►Logical View►Logical Model►Logical Model】後，將出現如圖 Q19-52 詳盡的類別圖資料。

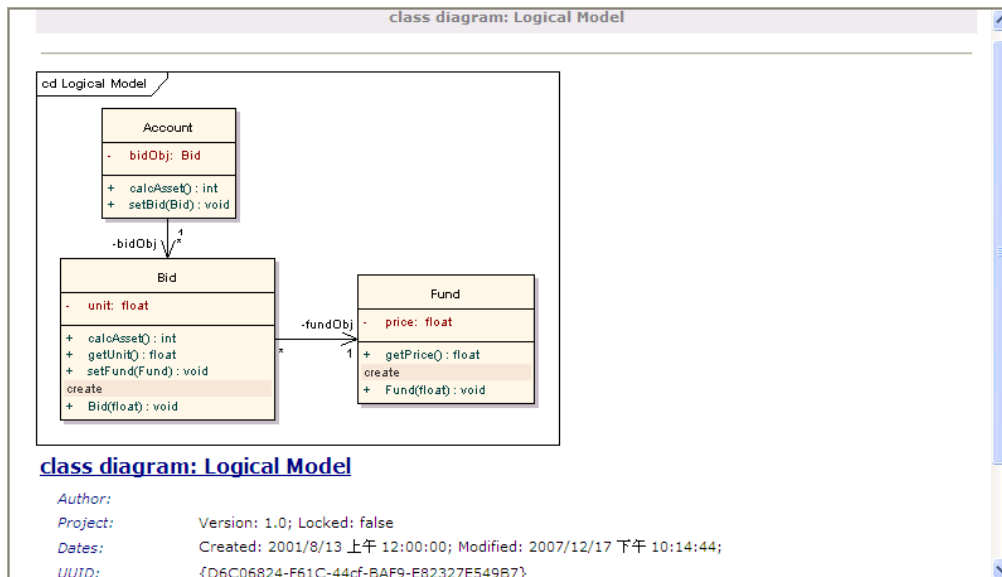


圖 Q19-52: 類別圖

6. 也可以循著連結開啓【View►Logical View►Logical Model►Account】後，將出現如圖 Q19-53 詳盡的 Account 類別資料。



圖 Q19-53: Account 類別

7. 接著，我們再開啓【View►Use Case View►Use Case Model►Use Case Model】，看到如圖 Q19-54 的使用案例圖。

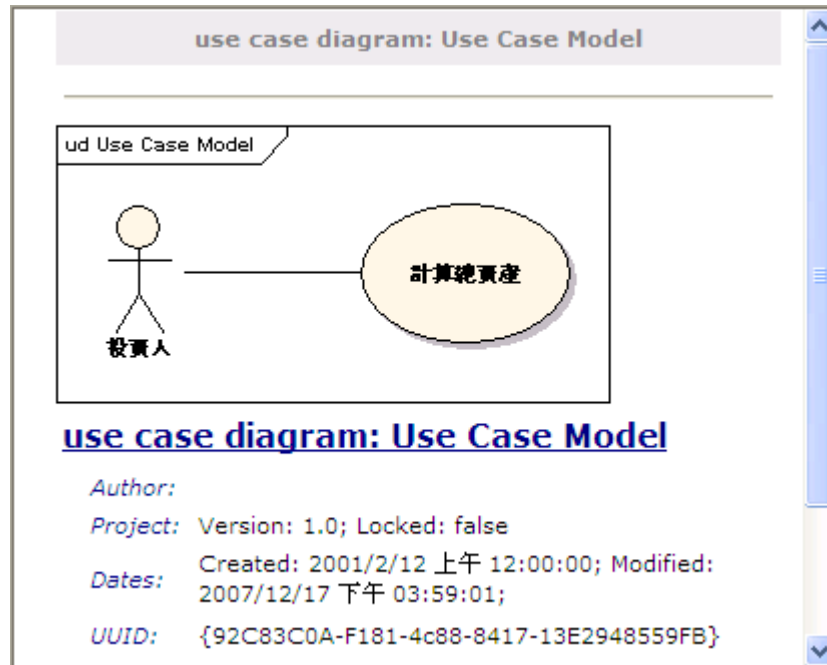


圖 Q19-54: 使用案例圖

8. 最後，再開啓【View►Dynamic View►Interactions►計算總資產 UC 之主要流程】，看到如圖 Q19-55 的循序圖。

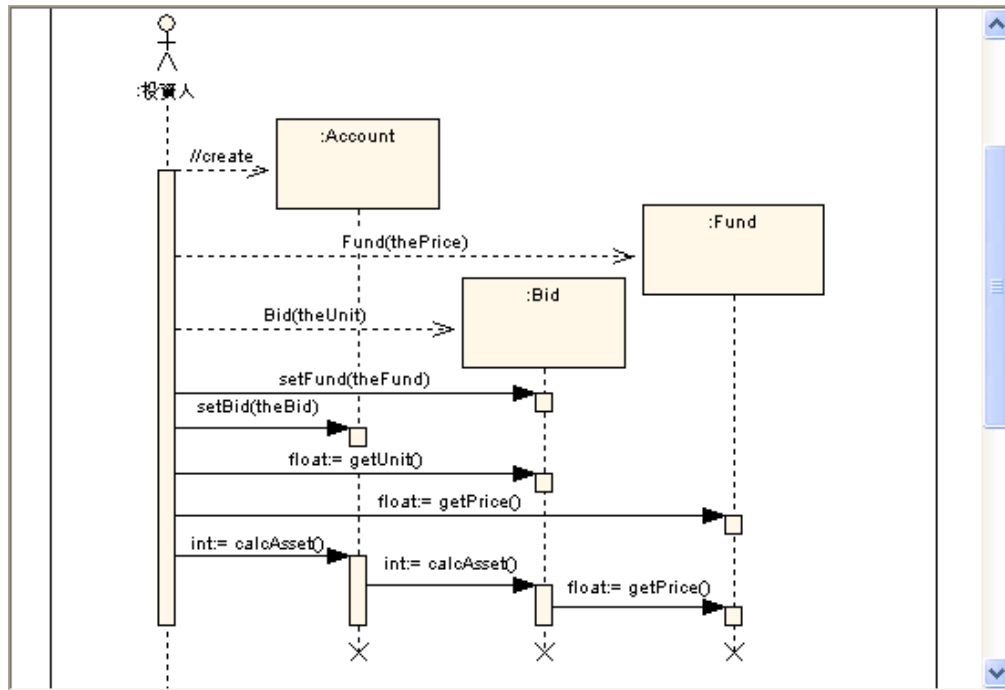


圖 Q19-55: 循序圖

Q20 雙向結合的圖示，到底是沒箭頭，還是雙箭頭？

(Jay於UML Blog之提問)

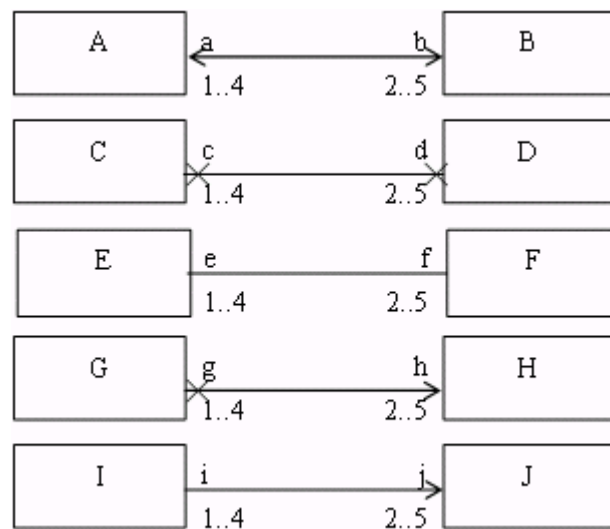


圖 Q20-1: 結合圖示

上面這張圖，是 UML2.x 規格書裡頭的範例圖。從 UML1.x 到 2.x 版，「可航性」(navigation)的表示法，其實是有變動過的。

先談 UML2.x，結合關係的箭頭端，代表具有可航性；打個小叉叉，就是不可航；單純直線，代表還未指定可航或不可航。所以，回到上圖的例子：

1. AB 是雙向結合。
2. CD 是兩端都不具有可航性。(別問我這樣的設計用在什麼地方，我也不知道，等著誰來告訴我。)
3. EF 還未指定兩端是否具可航性，還是不可航性。
4. GH 為單向結合，可由 G 航行到 H。
5. I 端還未決定可否航行，可以確定的是，可由 I 航向 J。

UML2.x 這樣定義之後，其實就很清楚了。不過，由於 UML1.x 版不是這樣定義的，所以實務上混淆著 UML1.x 和 2.x 的情況下，就容易造成誤解。

我們來看原先 UML1.x 的定義，只有兩種情況：有箭頭或沒箭頭。有箭頭就是可航行，沒箭頭就不可航行。所以，如果以 UML1.x 版來解釋上圖的話：

1. AB 是雙向結合。
2. CD 無法解釋，小叉叉是 UML2.x 的新符號。
3. EF 也是雙向結合。
4. G 端無法解釋，H 端具有可航性。
5. IJ 是單向結合，由 I 端航行到 J 端。

當年 UML1.x 給了兩套表示法，只是實務上都採用第一套表示法：

1. 直線代表雙向結合；單箭頭代表單向結合。

2. 雙箭頭代表雙向結合；單箭頭代表單向結合。

回到實務上來說，畫雙箭頭或小叉叉太繁瑣，所以現在都還是沿用單箭頭表示單向結合，直線表示雙向結合的表示法。也因此，到目前為止，多數的 UML 工具都還是採用直線代表雙向結合、單箭頭代表單向結合。

再者，UML 規格文件中，全都使用直線代表雙向結合，也沒見它用了小叉叉或雙箭頭。嘿嘿，這也難怪雙箭頭和小叉叉不見流行。