

# **Software Engineering**

## **U-04 MDA**

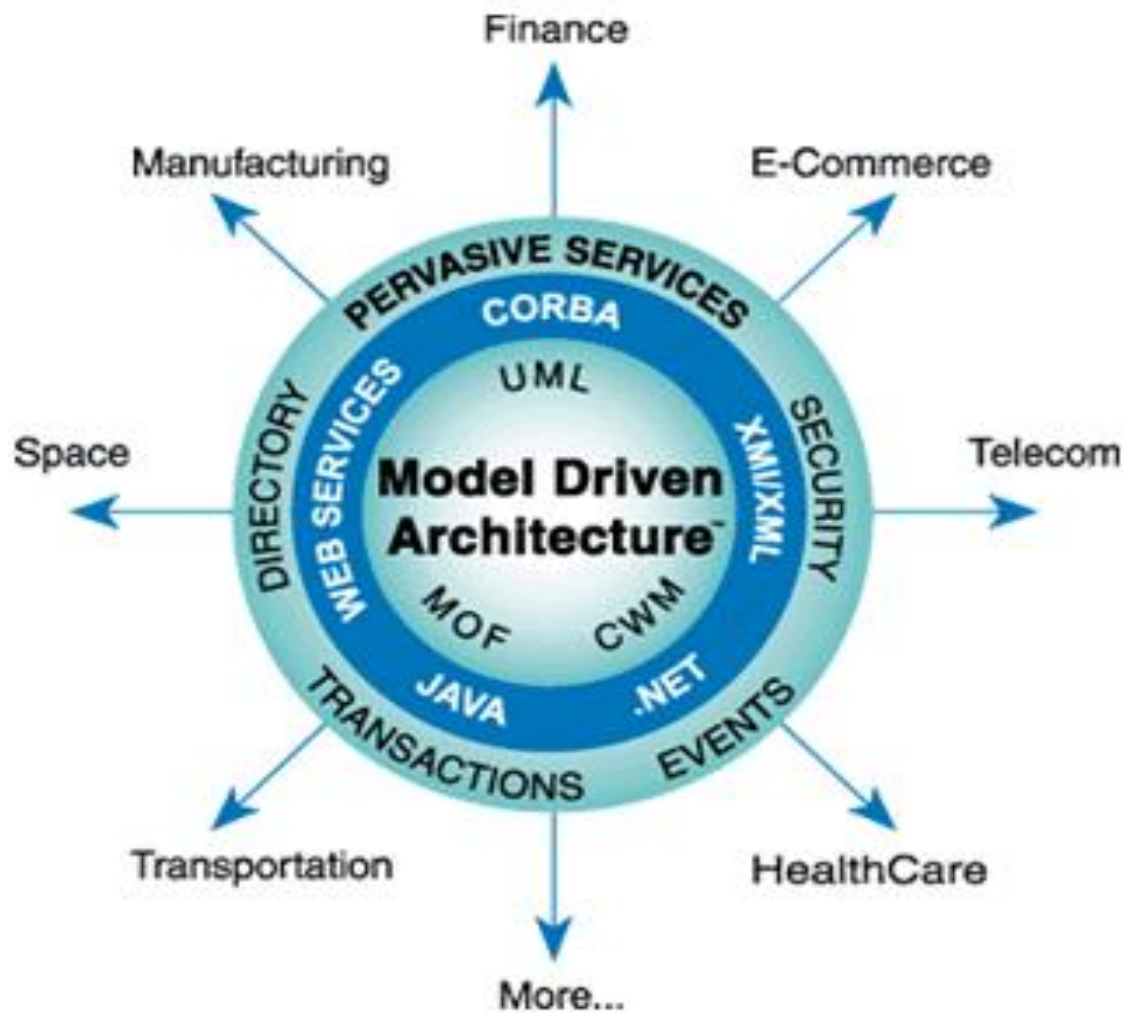
2024.04\_V1.4

# 內容大綱

## 學習目標

- MDA 架構
- MDA發展生命週期
- MDA轉換程序
- MDA轉換方法論
- MDA轉換案例
- 結論

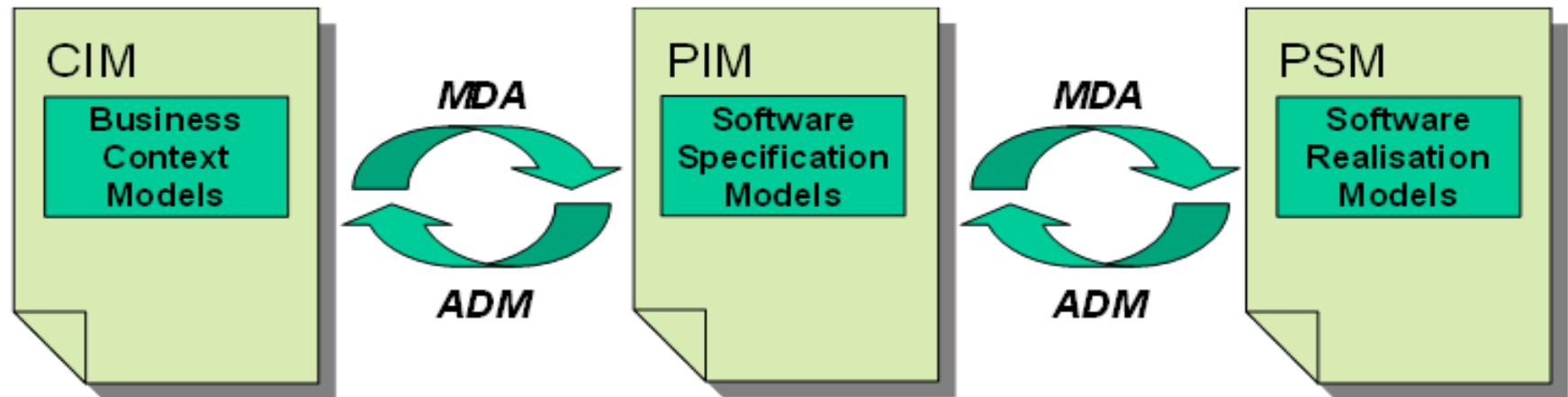
# MDA架構



參考網址: <http://www.omg.org/mda/>

# MDA發展生命週期(1/6)

- MDA (Model Driven Architecture)是由OMG定義的一種軟體開發架構，其關鍵是軟體開發過程中每個階段（或步驟）的產出均須建構出模式，且該模式產出是下一個階段的輸入。
- MDA的發展生命週期其實與其他系統開發模式主要的差別之一是在發展過程中步驟之產出，強調該產出是由電腦可理解的正規模式（Formal Model）表達。



[Note] Architecture-driven modernisation (ADM)

參考網址: <http://www.modelbased.net/mdi/mda/mda.html>

# MDA發展生命週期(2/6)

- STEP-1. CIM (Computation Independent Module) 階段(無關運算的模式)
- The computational independent viewpoint is focused on **the environment of the system** and on **the specific requirements of the system**.
- A CIM represents the computational independent viewpoint.
- The CIM hides **the structural details** and, of course, **the details related to the targeted platform**.
- 重點在系統環境與需求,不涉及系統內部的結構與運作細節。

# MDA發展生命週期(3/6)

- STEP-2. PIM (Platform Independent Module) 階段(無關平台的模式)
- The platform independent viewpoint is focused on **the operation of the system, hiding the platform specific details**.
- A PIM exhibits **platform independence** and is suitable for use with a number of different platforms of similar types.
- The PIM gathers all the information needed to **describe the behavior of the system** in a platform independent way.
- 重點在系統內部細節,不涉及實作系統的實體平台。

# MDA發展生命週期(4/6)

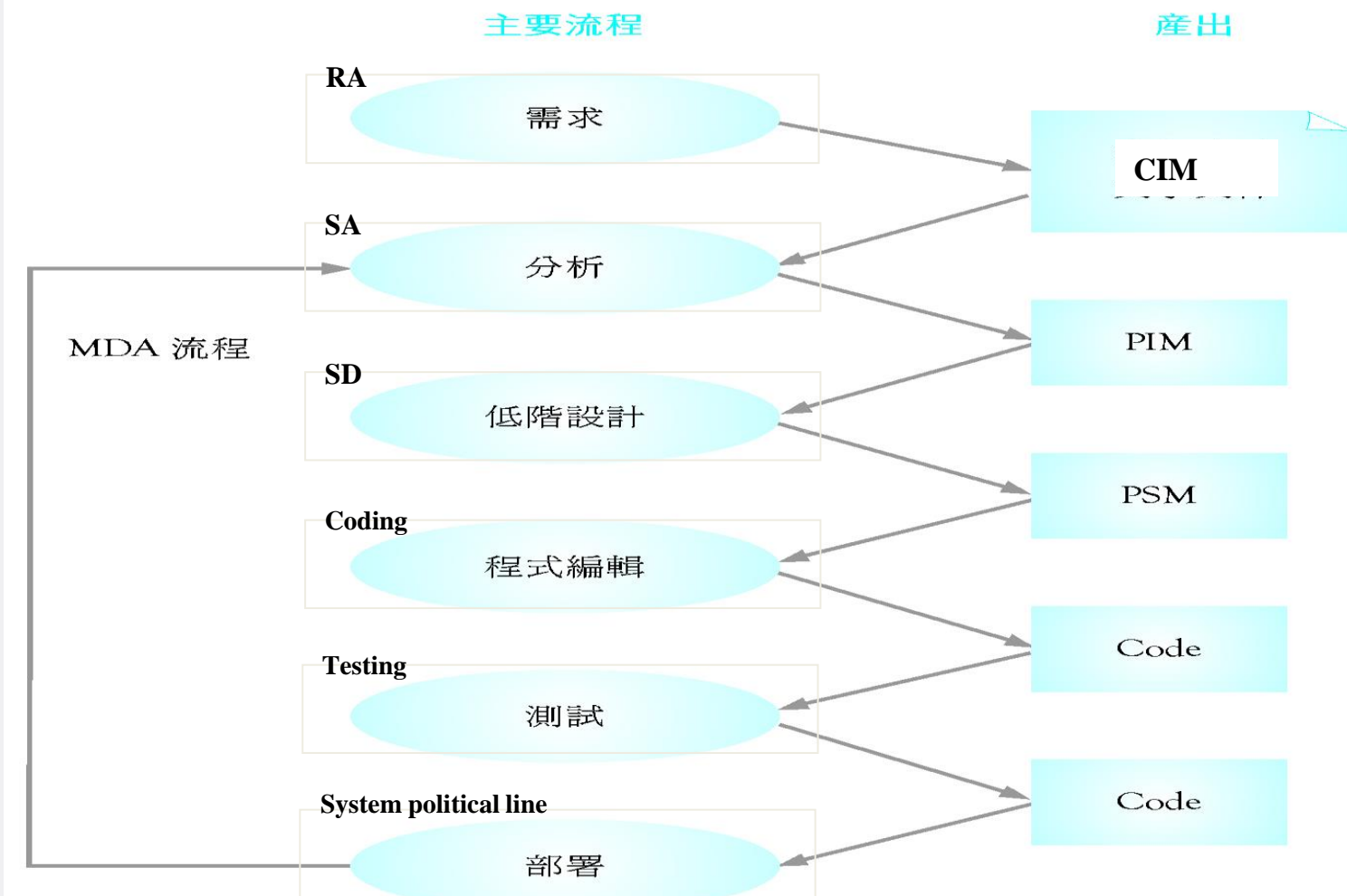
- STEP-3. PSM (Platform Specific Module) 階段 (特定平台模式)
- A platform specific model is a view of the system from the **platform specific** viewpoint.
- A PSM combines the specifications in the PIM with **the details that specify how the system uses a particular type of platform**.
- The PSM represents the PIM taking into account the specific platform characteristics.
- 重點在將所規劃的系統落實於特定實體平台的細節。例如:J2EE or .NET 平台。

# MDA發展生命週期(5/6)

- STEP-4. [Implementation Specific Model \(ISM\)](#)階段
- 重點每一個 PSM 需被轉成程式模式（或簡稱程式碼）。



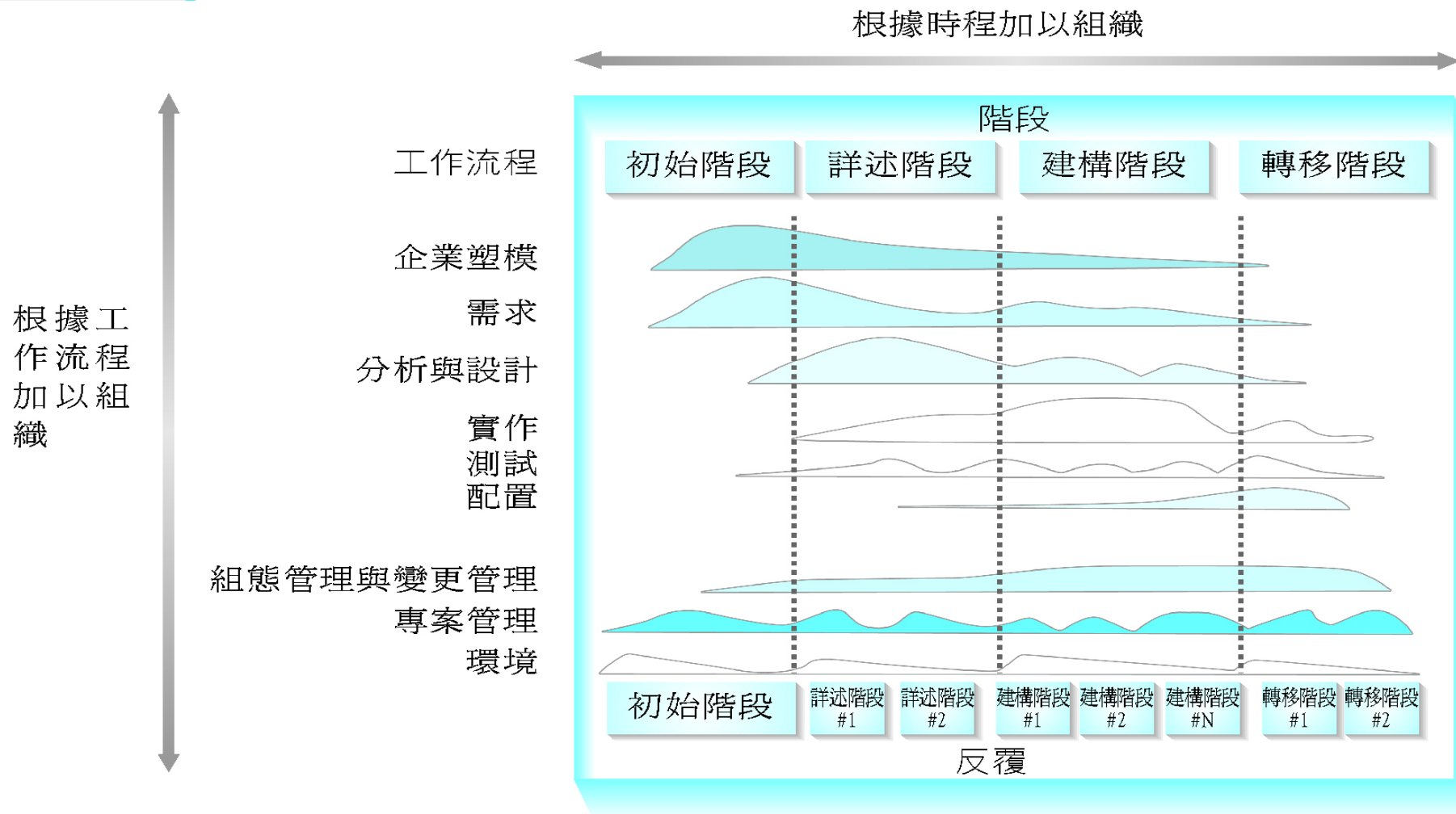
# MDA軟體發展生命週期 (6/6)



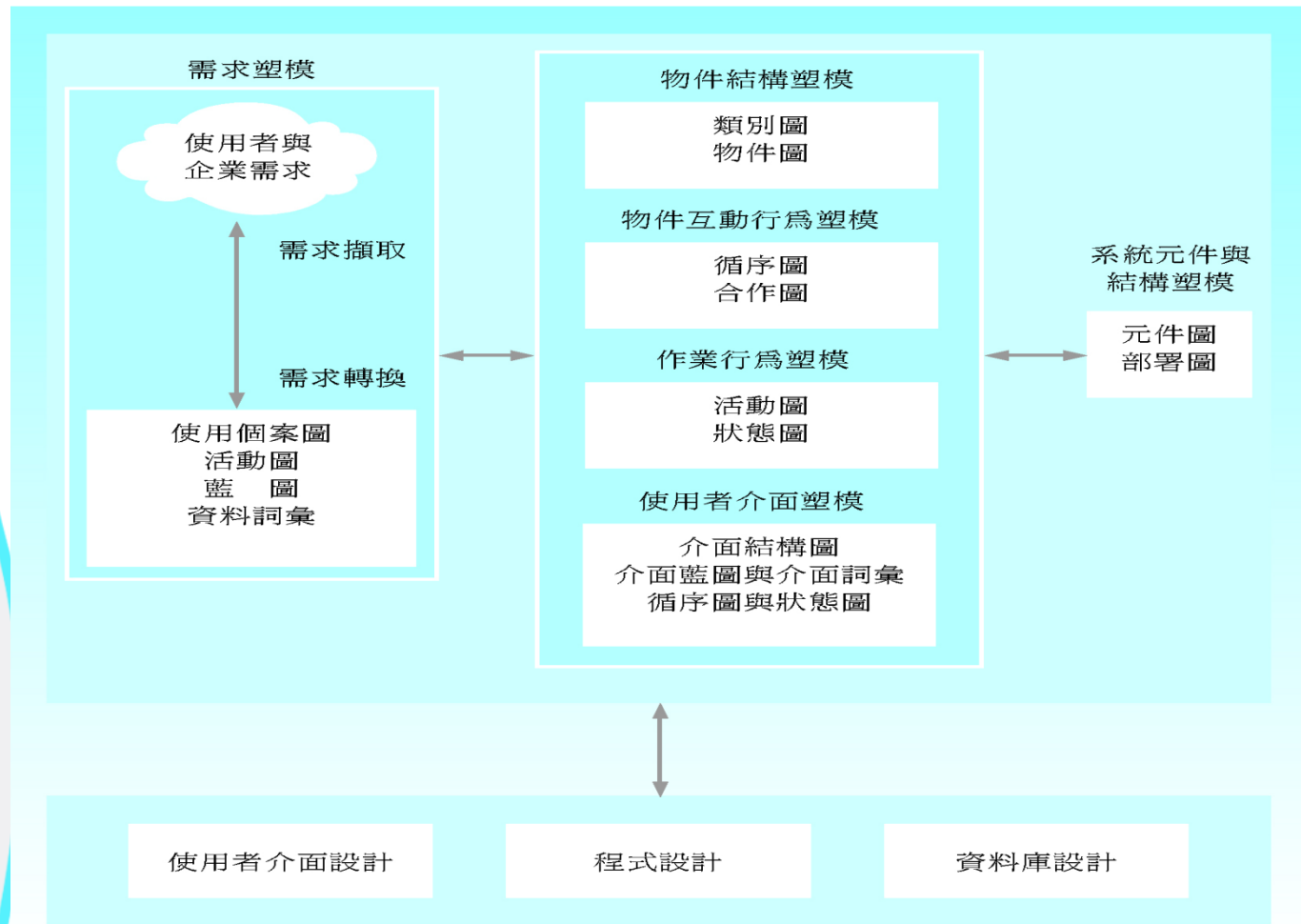
# MDA 開發程序 (修正)

- CIM-1:定義企業流程(Business Process),產出企業 Use Case圖
- CIM-2:分析企業流程,產出水道圖(活動圖+swimlane)
- CIM-3:定義系統範圍,產出系統 Use Case圖+活動圖=>系統功能層級結構 (\*若無法以活動圖表達企業規則,可使用狀態圖分析)
- PIM-1:分析系統流程,產出系統 強韌分析圖
- PIM-2:進行資料分析(定義屬性),產生DFD圖,ERD圖,並建置資料庫
- PIM-3:進行流程分析(定義操作與方法),產生循序圖, (\*)狀態圖
- PIM-4:進行UI分析,產生介面架構總圖, 介面詞彙與藍圖
- PIM-5:定義靜態結構,產生靜態類別圖(設計階段)
- PSM: 進行開發平台轉換
- Code:進行系統時作(程式開發與測試)

# UML方法論 - RUP 模式(1/3)

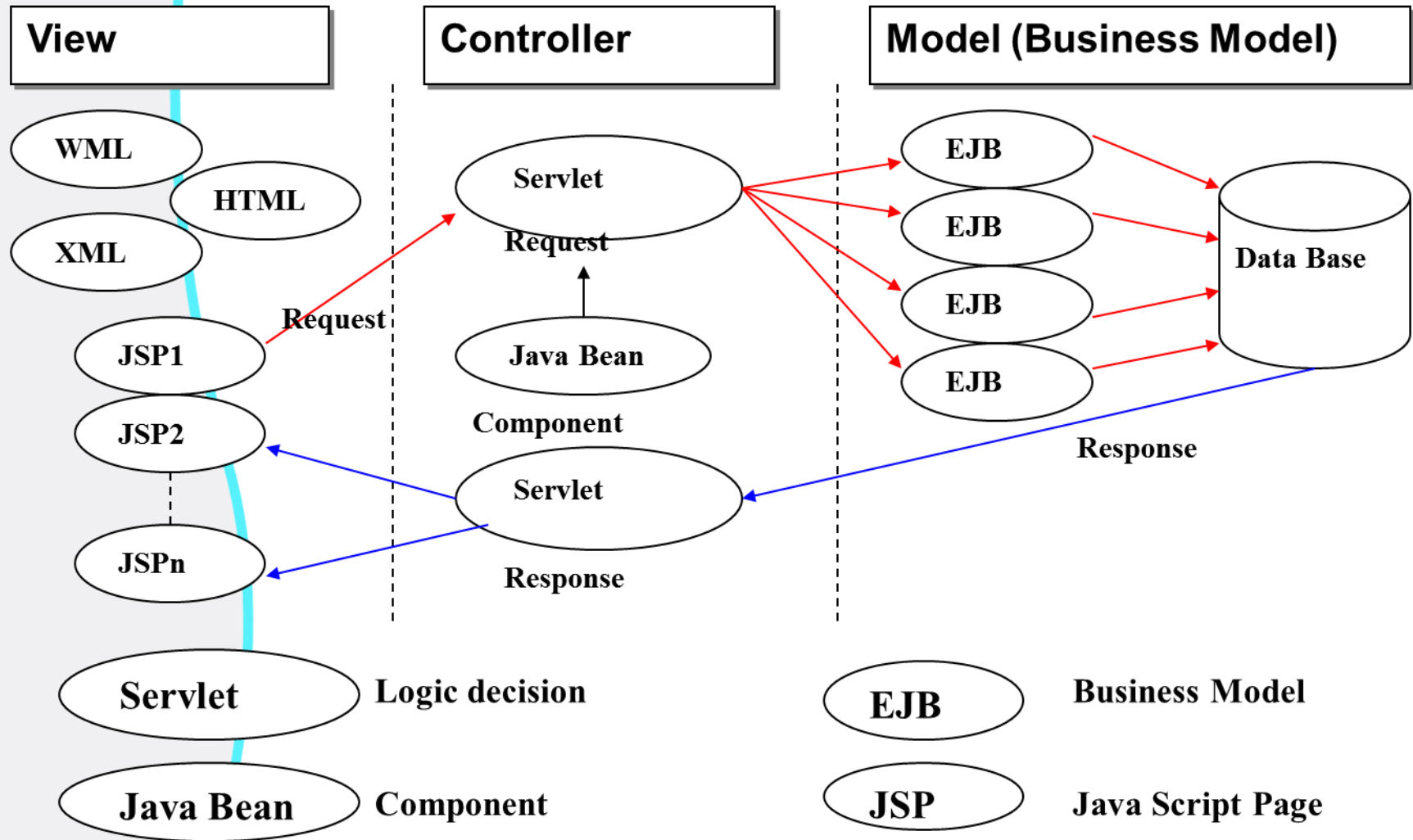


# UML方法論 - 塑模流程(2/3)



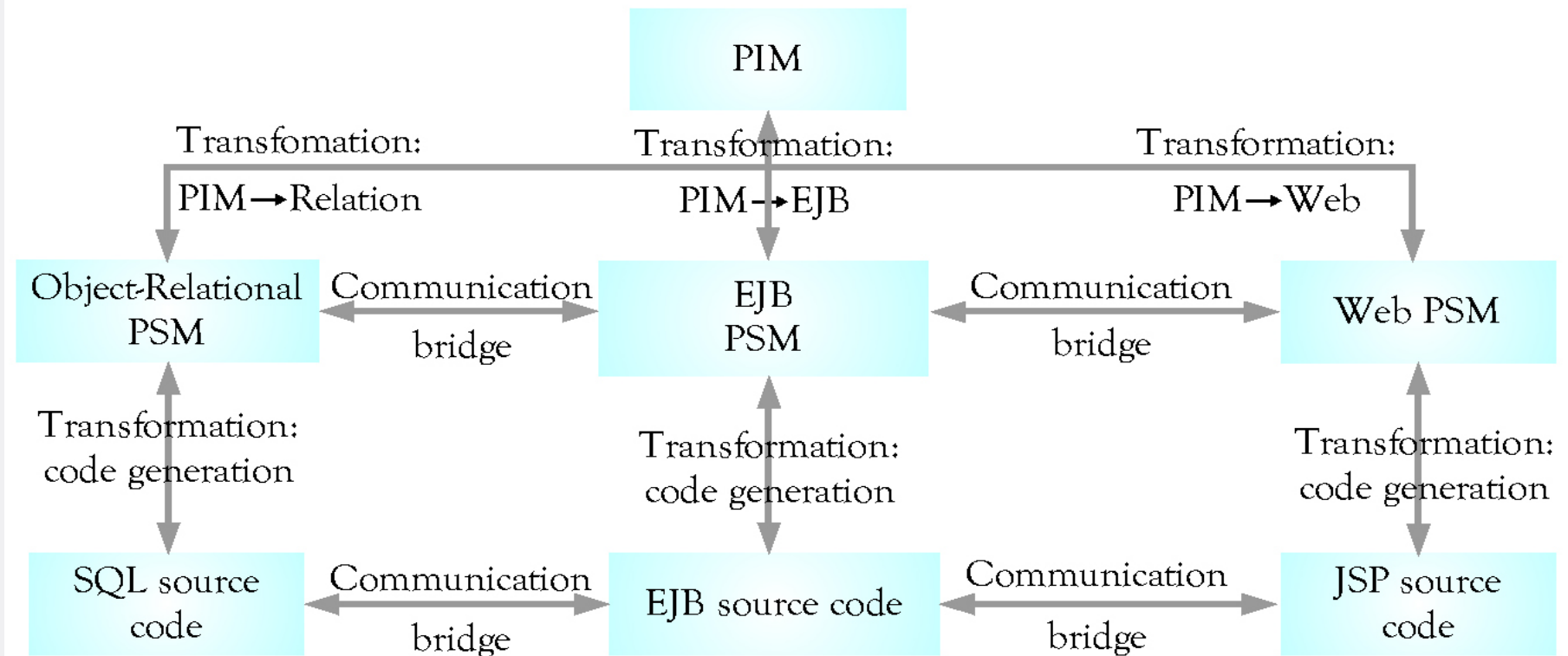
[illegible]

# MVC 架構



# MDA轉換程序(1/2)

- 先製作PIM，再將PIM轉成一個或多個PSM，接著再將PSM轉成Code。

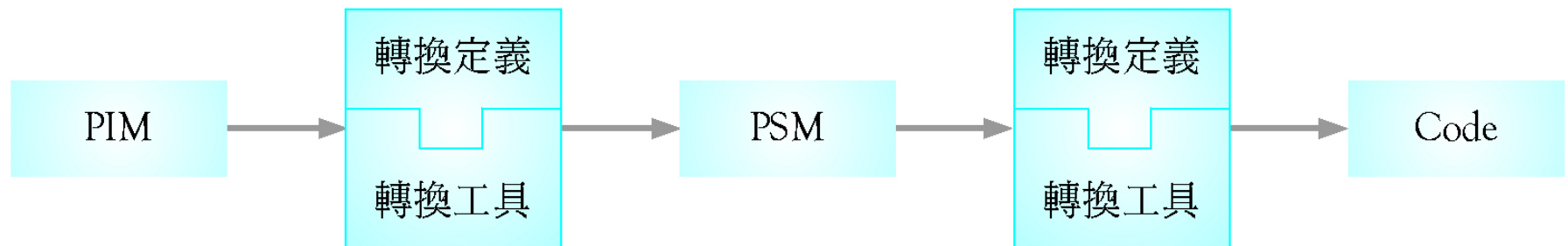


# MDA轉換程序(2/2)

- PIM可藉由CASE工具轉換成PSM，再轉換成Code

圖 12-3

MDA 三個主要模式與轉換步驟





# MDA轉換方法論 – PIM轉Object-Relational PSM

- PIM轉Object-Relational PSM包括三項工作：
  1. 資料型態對應
  2. 類別轉換
  3. 關係轉換。

# 資料型態對應

- PIM中之每一個屬性在PSM中應有一個資料型態與之對應，資料型態對應主要是從類別圖中之屬性型態與特定Object-Relational資料庫的資料型態間找出對應，可分為簡單與複雜的屬性，簡單的屬性如String、Float與Date等，可以直接對應到資料庫的某一個資料型態。
- 基本上一個永存性的類別，可以對應到資料庫的一個資料表(Table)，此時該資料表須額外加入一個欄位(OID, Object Identity)以作為主鍵(Primary Key)。

# 資料型態對應範例

Attribute	Data type in Oracle 9i
String	VARCHAR2
Float	FLOAT
Double	FLOAT
Byte	NUMBER(3,0)
Integer	NUMBER(10,0)
Single	FLOAT
Long	NUMBER(20,0)
Boolean	NUMBER(1,0)
Date	DATE
Currency	FLOAT

# 類別轉換

- 類別圖中有永存類別 ( Persistent Class ) 與暫存類別 ( Transient Class ) ，其中暫存類別之資料不需被儲存，而僅永存類別之資料需被儲存在資料庫中。
- 轉換步驟如下：
  - 將每一個永存類別轉換成一個資料表。
  - 將永存類別中之每一個屬性轉換成該資料表中的一個欄位。
    - 若資料庫使用關聯式資料庫，將鍵屬性轉換成資料表中之主鍵，並加入資料型態。
    - 若資料庫使用物件-關聯式資料庫，將新增OID欄位當成資料表中之主鍵。
    - 將每一個非鍵屬性轉換成資料表中的一個欄位，並加入資料型態。

# 關係轉換(1/5)

- 類別圖中有四種關係：
  1. 相依關係
  2. 實現化關係
  3. 關聯關係
  4. 一般化關係
- 其中僅關聯關係與一般化關係需要轉換成關聯表，而相依與實現化關係會轉換到使用者介面或程式碼。

# 關係轉換(2/5)

- 關聯關係之轉換
  - 關聯關係之轉換包括屬性之基數 ( Multiplicity )、聚集與組合關係之轉換。
  - 關聯關係之屬性基數有一對一、一對多、多對多之情況，其轉換步驟與規則如下：

# 一對一轉換範例



Table A	主鍵	外鍵
A_OID	✓	
.....		

Table B	主鍵	外鍵
B_OID	✓	
.....		
A_OID		✓

# 一對多轉換範例



Table A	主鍵	外鍵
A_OID	✓	
.....		

Table B	主鍵	外鍵
B_OID	✓	
.....		
A_OID		✓



# 多對多關聯類別之轉換範例

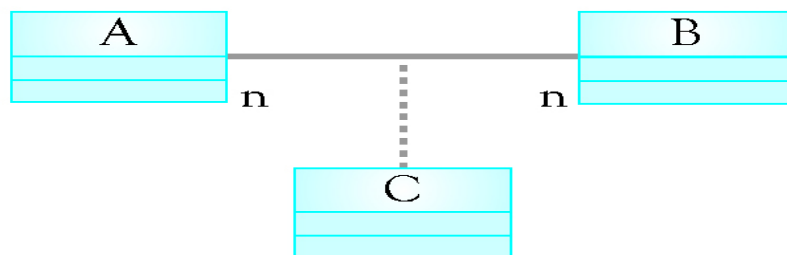


Table A	主鍵	外鍵
A_OID	✓	
.....		

Table B	主鍵	外鍵
B_OID	✓	
.....		

Table C	主鍵	外鍵
A_OID	✓	✓
B_OID	✓	✓
.....		

# 關係轉換(3/5)

- 聚集與組合關聯關係之轉換
  - 聚集與組合均是關聯關係的一種特定變異，兩者很相似，相同點均是描述整體與其組件之關係，也就是均表達一個「較大」類別之物件（整體）是由另一些「較小」類別之物件（組件）所組成。
  - 組合關聯關係的組件不能單獨存在，而聚集關聯關係的組件則可以單獨存在。
  - 聚集關聯關係之轉換與關聯關係之一對一、一對多之轉換相似，也就是整體端的主鍵需放入組件端作為其外鍵。

# 聚集關聯關係之轉換範例

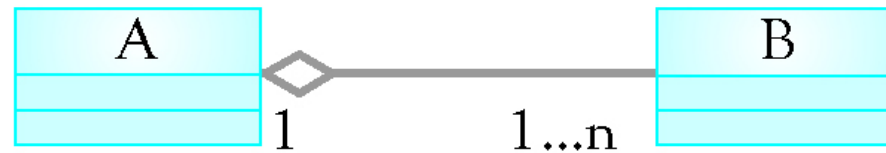


Table A	主鍵	外鍵
A_OID	✓	
.....		

Table B	主鍵	外鍵
B_OID	✓	
.....		
A_OID		✓

# 關係轉換(4/5)

- 組合關聯關係之基數也有一對一與一對多兩種，其轉換方式需將整體端的主鍵需放入組件端作為其主鍵與外鍵。



Table A	主鍵	外鍵
A_OID	✓	
.....		

Table B	主鍵	外鍵
A_OID	✓	✓
B_OID	✓	
.....		

# 關係轉換(5/5)

- 一般化關係之轉換
  - 一般化關係中之類別有父類別與子類別之分，其關係之轉換是將父類別之主鍵放入子類別中作為子類別之主鍵與外鍵。

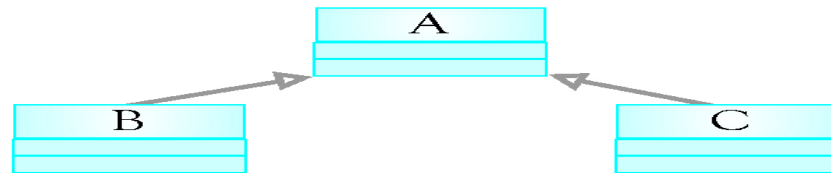


Table A	主鍵	外鍵
A_OID	✓	
.....		
Table B	主鍵	外鍵
A_OID	✓	✓
.....		
Table C	主鍵	外鍵
A_OID	✓	✓
.....		

# PIM轉應用程式 PSM

- PIM轉應用程式PSM的主要工作有二：
  - （1）將PIM中的每一個類別（關聯類別除外）轉換成一個應用程式的樣版類別(Template Class)。
  - （2）每個樣版類別應產生一個溝通橋樑與該類別在PIM轉資料表PSM中所產出的資料表溝通，例如進行資料之存取。
- PIM轉Java應用程式PSM的主要步驟如下：
  - 將類別名稱轉換成樣版類別名稱。
  - 將類別的每一個屬性轉換成該樣版類別之 “private”屬性，且產生 “public”的get operation與set operation。
  - 將類別的每一個操作轉換成該樣版類別之操作(或稱方法)。
  - 若該類別有關聯關係，且其另一端之類別的基數為1，則須在該樣版類別產生一個 public的屬性；若另一端基數為多，則須在該樣版類別產生一個 public且資料型態為陣列的屬性。

# PIM轉Java應用程式PSM之範例



A.java	B.java
<pre>public class A {     private String attr_a1;     private Date attr_a2;     private Float attr_a3;     public B theB[];     public setattr_a1(): String;     public getattr_a1(a1: String);     public setattr_a2(): Date;     public getattr_a2(a2: Date);     public setattr_a3(): Float;     public getattr_a3(a3: Float); }</pre>	<pre>public class B {     private Integer attr_b1;     private Currency attr_b2;     private Boolean attr_b3;     public A theA;     public setattr_b1(): Integer;     public getattr_b1(b1: Integer);     public setattr_b2(): Currency;     public getattr_b2(b2: Currency);     public setattr_b3(): Boolean;     public getattr_b3(b3: Boolean); }</pre>

# PIM轉使用者介面PSM

- PIM轉使用者介面PSM，包括三項工作：類別屬性對應網頁元件、使用者介面及控制類別轉換、關係轉換，分別介紹如下。



# 類別屬性對應網頁元件

- PIM中之每一個屬性在Web PSM中應有一個網頁元件與之對應，網頁元件對應主要是從類別圖中介面類別之屬性型態與特定Web系統開發環境的網頁元件之間找出對應。

Attribute	Web Component in HTML
Textbox	<<HTML Input>>text
Password	<<HTML Input>>password
Checkbox	<<HTML Input>>checkbox
RadioButton	<<HTML Input>>radio
Submit	<<HTML Input>>submit
Hidden	<<HTML Input>>hidden
Image	<<HTML Input>>image
Button	<<HTML Input>>button
Fieldset	<<HTML Select>>
Textarea	<<HTML Textarea>>
ListBox	<<HTML Textarea>>

# 使用者介面及控制類別轉換(1/2)

- 從類別圖轉換至Web PSM之前必須先做前置處理，才能將類別圖中的使用者介面類別依照不同的應用類型分別轉換為適當的使用者介面類別。
- 為符合實際系統軟體的運作，UML的延伸機制提出網頁應用程式的類別符號，將使用者介面類別依照性質的不同歸類為五種：
  - 伺服器端網頁類別(Server-side Class)
  - 客戶端網頁類別(Client-side Class)
  - 表單類別(Form Class)
  - 框架類別(Frameset Class)
  - 目標類別(Target Class)。



# 使用者介面及控制類別轉換(2/2)

- 介面類別的轉換須依照介面在網路應用系統中不同的性質加以歸類。轉換步驟如下：
  - 每個使用個案對應產生一個客戶端網頁類別。
  - 分析每個使用個案之「活動圖」中所有輸出 / 入註記，並搭配「PAC系統架構圖」整理出該使用個案的使用者介面類別滙總表，依滙總表中的介面類別轉換為表單類別，且必須載入至該使用個案的客戶端網頁類別中執行操作。
  - 每個使用個案對應產生的控制類別則被轉換為伺服器端網頁類別。

# 關係轉換

- 有別於實體類別關係，在網頁應用模式中，使用者介面類別之間有四種關係：超連結(Link)、建立(Build)、重導(Redirect)、送出(Submit)。這些關係是隸屬於實體類別關聯關係之延伸機制。

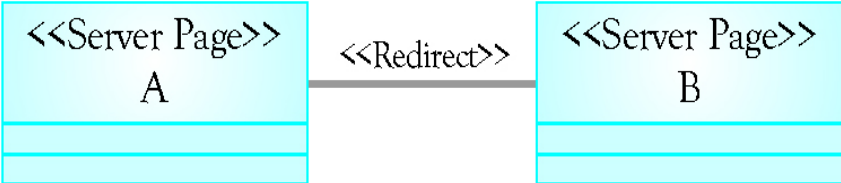

# 超連結關係轉換之範例

Situation	範例
1. 兩個物件皆是客戶端網頁。	 <p>The diagram shows two light blue rectangular boxes representing web pages. The left box is labeled '&lt;&lt;Client Page&gt;&gt;' and 'A' below it. The right box is labeled '&lt;&lt;Client Page&gt;&gt;' and 'B' below it. A horizontal line connects the right side of box A to the left side of box B, with the text '&lt;&lt;Link&gt;&gt;' centered above the line.</p>
2. 某個物件是客戶端網頁，另一個物件是伺服器端網頁。	 <p>The diagram shows two light blue rectangular boxes representing web pages. The left box is labeled '&lt;&lt;Client Page&gt;&gt;' and 'A' below it. The right box is labeled '&lt;&lt;Server Page&gt;&gt;' and 'B' below it. A horizontal line connects the right side of box A to the left side of box B, with the text '&lt;&lt;Link&gt;&gt;' centered above the line.</p>


# 建立關係轉換之範例

Situation	範例
1. 某個物件是伺服器端網頁建立另一個物件是客戶端網頁。	<pre>graph LR; A["&lt;&lt;Server Page&gt;&gt; A"] -- "&lt;&lt;Build&gt;&gt;" --&gt; B["&lt;&lt;Client Page&gt;&gt; B"]</pre>

# 重導關係轉換之範例

Situation	範例
1. 某個伺服器端網頁轉向另一個伺服器端頁面。	 <pre>graph LR; A["&lt;&lt;Server Page&gt;&gt; A"] -- "&lt;&lt;Redirect&gt;&gt;" --&gt; B["&lt;&lt;Server Page&gt;&gt; B"]</pre>
2. 某個伺服器端網頁轉向另一個客戶端頁面。	 <pre>graph LR; A["&lt;&lt;Server Page&gt;&gt; A"] -- "&lt;&lt;Redirect&gt;&gt;" --&gt; B["&lt;&lt;Client Page&gt;&gt; B"]</pre>

# 送出關係轉換之範例

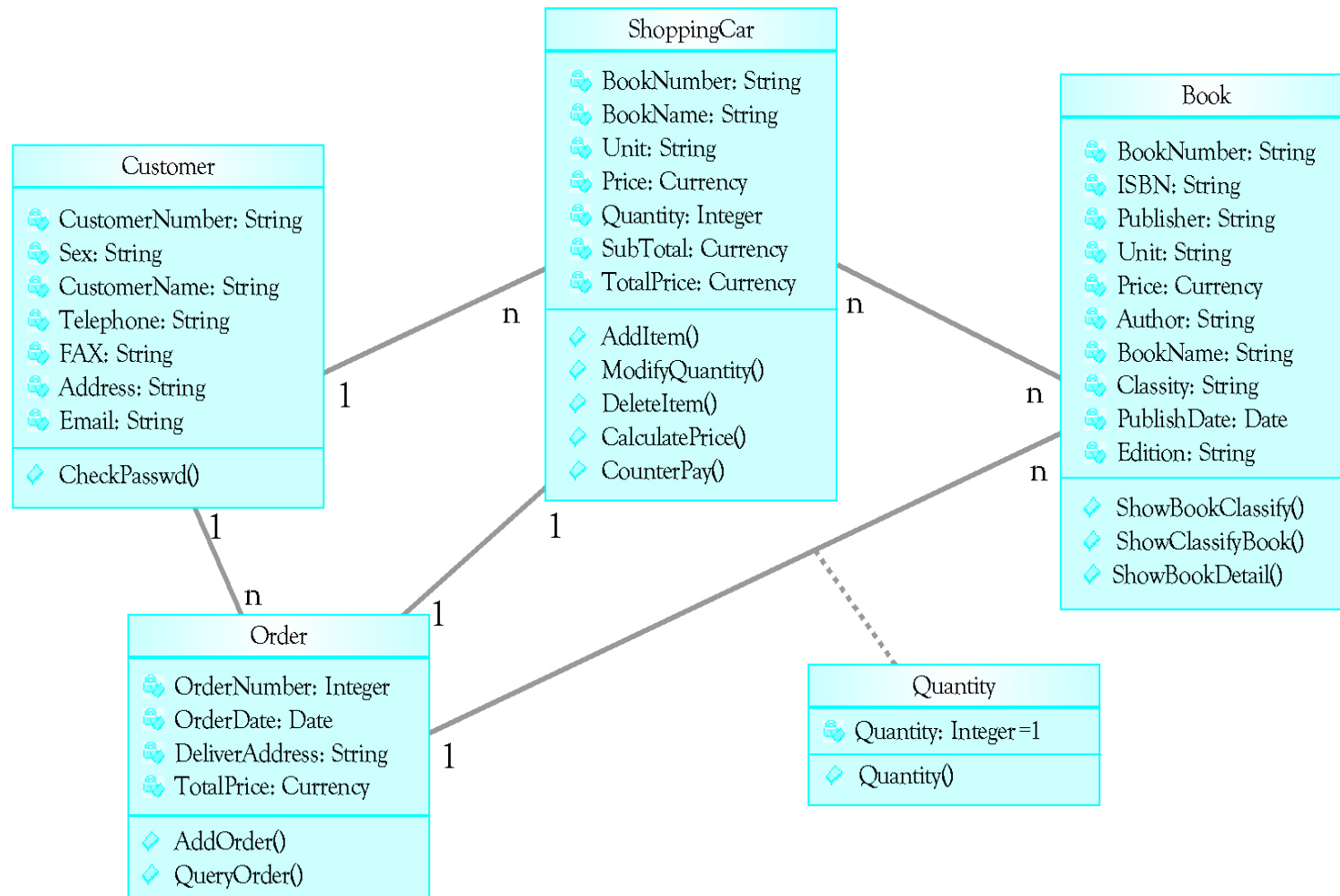
Situation	範例
1. 表單類別輸入資訊後送到另一個物件為伺服器網頁。	 <pre>graph LR; A["&lt;&lt;HTML Form&gt;&gt; A"] -- "&lt;&lt;Submit&gt;&gt;" --&gt; B["&lt;&lt;Server Page&gt;&gt; B"]</pre> <p>The diagram illustrates a transition between two objects. On the left is a light blue box labeled '&lt;&lt;HTML Form&gt;&gt;' with 'A' below it. On the right is a similar box labeled '&lt;&lt;Server Page&gt;&gt;' with 'B' below it. A horizontal line connects the two boxes, with the text '&lt;&lt;Submit&gt;&gt;' centered above it. Both boxes have two empty horizontal lines at the bottom.</p>



# MDA轉換 - PIM轉PSM案例

- 雖然PIM轉PSM之工作可以自動化的執行，但瞭解自動化背後的轉換原理或方法論，有助於瞭解PIM之建構。
- 假設網路購書系統最終之類別圖可表達成實體類別圖以及UI與控制類別圖；實體類別圖共有五個實體類別（含一個關聯類別），類別間之關係均是關聯關係

# 網路購書系統之實體類別圖



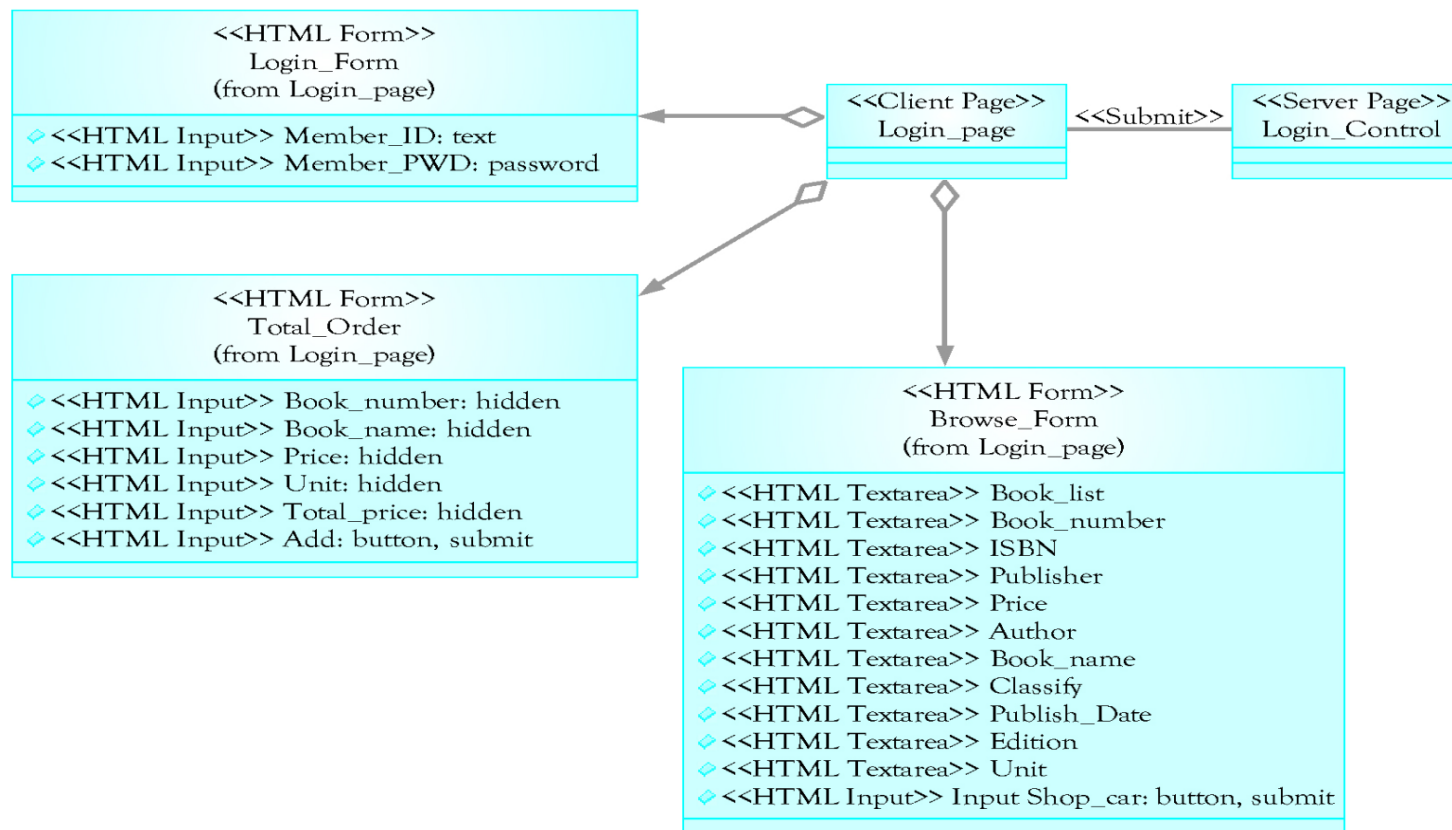
# PIM轉Web PSM(1/3)

- PIM轉Web PSM包括網頁元件對應、介面及控制類別轉換、關係轉換。
  - 網頁元件對應

Attribute	Web Component in HTML
Textbox	<<HTML Input>>text
Password	<<HTML Input>>password
Hidden	<<HTML Input>>hidden
Button	<<HTML Input>>button
Checkbox	<<HTML Input>>checkbox
Submit	<<HTML Input>>submit
Textarea	<<HTML Textarea>>

# PIM轉Web PSM(2/3)

## – 介面及控制類別轉換








# PIM轉Web PSM(3/3)

## — 關係轉換

- 介面類別若轉換為表單類別會和其所屬的使用個案之客戶端網頁類別之間存在「聚集」的關係。由客戶端的網頁或表單輸入的資料，會傳送至伺服器端網頁類別再經由後端的實體類別進行處理，故客戶端網頁類別和伺服器端網頁之間存在送出(Submit)關係。

# Web PSM轉系統畫面及其程式碼

- Web PSM轉系統畫面及程式碼包括網頁元件對應、介面及控制類別轉換、關係轉換三個部分。
  - 系統畫面之網頁元件對應

Web Component in HTML	Picture	HTML Code
<<HTML Input>>text		<input type="text" name="textbox" size="10">
<<HTML Input>>hidden		
<<HTML Input>>button		<input type="submit" value="送出" name="B1">
<<HTML Input>>submit		
<<HTML Input>>password		<input type="password" value=" " name="P1">
<<HTML Input>>checkbox		<input type="checkbox" value="3" name="C1">>
<<HTML Textarea>>		<textarea rows= "4" name= "textarea"cols="30"> </textarea>