Chapter 12

生成式 AI x MDA 轉換之案例實作

本章學習重點

詳讀本章，你至少能瞭解：

- 生成式 AI 結合 MDA 與物件導向系統分析與設計的實作概念。
- 生成式 AI 結合 MDA 如何提升高階設計文件的應用。
- 何謂「分析與設計的文件就是規格，也是系統」的概念。
- 如何以生成式 AI 實作 MDA 三個核心模式的轉換。

12.1 導　論

在軟體系統開發過程中，若將物件導向分析與設計(OOAD)結合模型驅動架構(Model-Driven Architecture, MDA)的概念，則在完成需求塑模與平台獨立模型(Platform-Independent Model, PIM)之後，接下來的關鍵工作即為選定系統的開發工具與執行平台，並將 PIM 轉換為可執行的程式碼(Code)。本節以「便當王網路訂購系統」為案例，說明如何在完成 PIM 之後依第 11 章之方法運用生成式 AI（以下以 ChatGPT 為例），直接將 PIM 轉 Code 的實作，包括展示轉換過程中需輸入 PIM 的圖形以及與生成式 AI 互動的提示語(Prompt)設計。

在完成便當王網路訂購系統之 PIM 塑模後，其彙整之類別圖共包含四個實體類別，分別為「客戶資料」、「訂單資料」、「便當袋資料」與「便當資料」（如圖 10-5 所示）。其中，客戶資料與訂單資料之間為一對多關聯關係；訂單資料與便當資料之間為多對多關聯關係，並透過「便當袋資料」作為關聯類別。此外，系統包含「客戶登入」、「便當型錄」、「細部說明」與「便當袋」四個介面類別，並相依於「登入」、「顯示」與「訂購」三個控制類別。

為了教學說明之便利性，第 9 與第 10 章採用中文類別圖進行說明；然而，由於本章之 PIM 轉 Code 係透過 ChatGPT 來完成，實務上建議讀者於 PIM 階段即採用英文類別圖與命名方式，以利後續自動化轉換作業。此外，PIM 轉 Code 的方法高度相依於所採用之開發工具與技術，因此在進行轉換前，必須明確說明使用之技術環境、開發框架、程式語言與資料庫等。

本章實作採用之「DB→AP→UI」轉換順序（詳細內容請參閱本書 11 章），清楚呈現資料結構、系統運作邏輯與介面呈現三者之間的層次關係。ChatGPT 在此流程中扮演關鍵角色，使模型得以一致、可驗證且可重複的方式，自動化轉換為程式碼，並確保各層之間的結構與邏輯維持高度一致性。本章旨在協助讀者理解如何結合生成式 AI 工具（如 ChatGPT）與 MDA，系統性地完成由 PIM 轉換至 Code 的完整軟體開發流程。

12.2 PIM 轉 DB

PIM 轉 Code 的過程主要涵蓋四項關鍵要素：資料型態對應、實體類別轉換（欄位名稱）、資料長度定義，以及實體類別間關係的轉換。當 PIM 階段之類別圖完成後，後續即需依據目標關聯式資料庫平台（例如 MySQL）的特性，運用

ChatGPT 將類別圖轉換為符合該平台規範的資料表結構。在資料型態對應方面，由於本案例採用之關聯式資料庫為 MySQL，各實體類別中屬性的資料型態，須轉換為 MySQL 所支援之對應資料型態，並同時定義其資料長度與必要限制條件。相關之資料型態對應關係彙整如表 12-1 所示。

Table
表 12-1 MySQL 之資料型態

資料型態	Data type in MySQL
Character(Char)	VARCHAR()
Integer(Int)	INT()
Date	DATE()

在實體類別轉換方面，於便當王網路訂購系統之彙總類別圖中，「便當袋資料」原本屬於暫存類別，其餘實體類別則屬於永存類別。然而，當客戶完成訂購並執行確認動作後，系統需將訂單編號與訂購日期記錄於「訂單資料」中，並將客戶所購買之便當數量儲存在「便當袋資料」內，最終形成一筆完整的訂單紀錄。由於上述資訊皆需長期保存，因此必須實際存入資料庫。基於此需求，在進行資料表轉換前，須先進行必要的前置處理，將原先定義為暫存之實體類別，一併視為永存類別進行轉換。如此一來，「便當袋資料」中之屬性（例如購買數量）方能被正確保留於資料庫中，避免在資料持久化過程中遭到遺失。同時，於後續進行 PIM 轉應用程式 Code 時，亦能確保相關屬性可順利轉換為對應之程式變數與資料結構，維持模型與程式碼之間的一致性。

12.2.1 ChatGPT 轉 DB 方法

1. 準備資料

如前所述，PIM 轉 Code 之轉換流程高度依賴所採用的開發工具與技術架構等。因此，在應用 ChatGPT 進行 PIM 至資料庫(DB)程式碼轉換之前，必須先明確界定相關的技術前提與執行環境。具體而言，需清楚說明資料庫轉換所需之輸入文件、系統執行環境，以及所使用的轉換工具與開發框架等關鍵要素。本案例所採用之技術設定與輸入資料類型彙整如表 12-2 所示，並於下文中加以簡要說明。

Table
表 12-2 準備資料

1.檔案資料	2.環境	3.工具
(1)PIM 檔案（類別圖、介面詞彙） (2)提示語 (Prompt)	(1)DB：MySQL (2)AP：Laravel 框架	(1)ChatGPT (2)PlantText

◎說明

- PlantUML：是一個通用的開源圖表工具，透過簡潔的腳本(script)描述即可快速生成 UML 圖或其他圖表。使用者可利用腳本精確定義並繪製 UML 的類別圖、循序圖等。
- **類別圖**：用於描述系統中類別(Class)的靜態結構及其相互關係。其內容通常包含：類別名稱、屬性名稱與資料型態、操作方法，以及類別之間的關聯（如繼承、聚合、關聯等）。在資料庫設計中，通常僅「實體類別」會轉換為資料表，並在轉換時標示主鍵與外鍵，其中主鍵以《PK》表示，外鍵以《FK》表示。
- **介面詞彙**：介面藍圖會搭配一個介面詞彙以表達該介面之代號、名稱、說明、介面中各元件的名稱、類型與功能及概念說明，若該元件可用於顯示或儲存資料庫中的資料，須說明其代表的資料表欄位、資料型態和長度。
- **提示語**(Prompt)（如表 12-3 所示）。

2. 應用 ChatGPT 將 UML 類別圖轉 PHP Laravel Migration 與 Model

理論上，雖然可以直接使用 ChatGPT 將 UML 類別圖一次性轉換為 PHP Laravel 的 Migration 與 Model，但在實務上更建議分為三個步驟進行（如表 12-4 所示）。如此不僅能逐步檢視各階段的產出是否正確，也有助於在問題發生時更容易追蹤並進行除錯。

步驟一：產出符合 PlantUML 語法的類別圖腳本

將原始類別圖檔與 Prompt1（如表 12-5 所示）一併輸入 ChatGPT，讓模型生成符合 PlantUML 語法的類別圖腳本。此步驟的目的在於將圖形化的類別結構轉換為符合 PlantUML 語法的類別圖腳本，以確認 ChatGPT 正確理解圖意，為後續的資料庫遷移檔(Migration)與 Model 產出奠定基礎。

Table
表 12-3 轉 DB 提示語設計

(1) PIM 之類別（主要是實體類別與關係）＋介面詞彙		
(2) Prompt 設計需要考量的元素：		
輸入	角色	資料庫管理師／網頁設計後端工程師
	任務	依輸入類別圖與介面詞彙，產出對應資料庫之遷移檔(Migration)、Model
	產出內容與格式	a. Migration：是以程式碼方式管理資料庫綱要的機制。透過遷移命令 (php artisan migrate)，可依照 Migration 檔中所定義的資料表結構，自動建立或更新實際的資料庫表格。此機制能確保資料庫綱要與系統版本同步演進，並提供安全、可追蹤的版本控管流程。其中，資料表欄位名稱、型態、長度、主鍵、外鍵（資訊來自類別圖提供的資料表欄位名稱與關聯＋介面詞彙定義的欄位名稱與型態、長度）
		b. Model：為應用程式 (AP)的一部分，透過 Laravel 的 Eloquent ORM 映射至資料庫中的實體資料表。Model 除了管理資料表欄位外，也封裝了一對一、一對多、多對多等正反向資料查詢的方法，使開發者能以物件操作的方式處理資料存取，並確保程式邏輯與資料庫結構保持一致（資訊來自類別圖）
輸出	符合 PHP Laravel 框架的 migration 與 model 程式碼	

Table
表 12-4 ChatGPT 轉 DB 實作三步驟

步驟	I	P	O	驗證
一	PIM 類別圖圖檔 +Prompt1	生成式 AI	PlantUML 類別圖腳本	完整性 正確性
二	PlantUML 類別圖腳本 +介面詞彙+Prompt2	生成式 AI	PHP 語法 ● Migration	完整性 正確性
三	PHP 語法 Migration +Prompt3	生成式 AI	PHP 語法 ● Model	完整性 正確性

◎驗證

採逆向工程檢核：將步驟一產出的 PlantUML 類別圖腳本匯入 PlantText 等工具，檢視重建後的類別圖是否與原始類別圖在資料結構上完全一致，確認類別與關聯的完整性與正確性。

Table
表 12-5 Prompt1

角色：資料庫管理師／網頁設計後端工程師 **任務**：將 UML 類別圖轉成 PlantUML 類別圖腳本 **產出內容與格式**：類別圖腳本

步驟二：產出 Migration(Database Schema)

將步驟一產出的完整且正確的 PlantUML 類別圖腳本加上介面詞彙，連同 Prompt2（如表 12-6 所示）一併輸入 ChatGPT，讓模型生成適用於 PHP Laravel 框架的 Migration 檔案。Migration 檔應包含 Schema::create()的方法，明確描述資料表欄位名稱、資料型態、長度、主鍵設定、外鍵約束與關聯欄位。 Migration 檔案所需之欄位名稱、型態、長度、主鍵、外鍵等資訊來自類別圖與介面詞彙提供的資料表欄位名稱與關聯，在 Laravel 專案中執行遷移命令(php artisan migrate)生成 MySQL 中的資料表。

Table
表 12-6 Prompt2

任務：根據輸入的 PlantUML 類別圖腳本與介面詞彙，生成符合 Laravel 框架要求的 migration 檔案。 **產出內容與格式**：PlantUML 類別圖腳本提供資料表欄位名稱與關聯並提供一對多、多對一、多對多等關係、介面詞彙定義欄位名稱與型態。每個實體類別產出對應的 Laravel migration 檔案，內容需包含：各資料表名稱與 Schema::create()方法；所有欄位名稱、型態與主鍵、外來鍵；外來鍵定義與關聯欄位設置（含 belongsTo、hasMany 等關聯關係）；以確保資料表間結構完整、可支援後續 Model 關聯與資料操作。

◎驗證

實際建立資料表後，檢查欄位名稱、型態、長度、主鍵與外鍵約束是否如預期被正確建立，確認資料表結構與類別圖（欄位名稱與關聯）、介面詞彙（欄位名稱與型態、長度）一致。

步驟三：產出 AP 中的 Model

將步驟二產出的完整且正確的 Migration，連同 Prompt3（如表 12-7 所示）一併輸入 ChatGPT，生成後端 PHP Laravel 框架下的 Model。Model 為應用程式(AP)的一部分，透過 ORM 映射到實際的資料表，並且封裝一對一、一對多、多對多等正反向關係。

Table
表 12-7 Prompt3

任務：根據輸入的 Migration 檔案，生成符合 PHP Laravel 框架要求的 Model。 **產出內容與格式**：每個實體類別產出對應的 Laravel Model 類別，其中$table 屬性指定實際映射的資料表名稱，$primaryKey與$keyType分別指定主鍵欄位與主鍵資料型態，$fillable屬性定義可以批次操作（例如 create）的欄位，建立 hasOne、hasMany、belongsTo 方法對應一對一與一對多等關聯。並採用 Eloquent ORM 規範撰寫；以確保 Model 層能正確對應資料表結構並支援資料操作。

◎驗證

1. 檢視 Model 中定義的資料表名稱、主鍵與關聯設定是否與 Migration 生成在 MySQL 中的資料表一致。
2. 建立測試資料以驗證外鍵與關聯關係的正確性。可透過執行 Laravel Tinker 命令(php artisan tinker)進行互動式測試。例如，可建立兩張具關聯的測試資料（如每位 customer 對應至少兩筆 order）驗證關聯與外鍵設定的正確性。

12.2.2 便當王系統轉 DB 之實作

步驟一（如表 12-8 所示）：產出符合 PlantUML 語法的類別圖腳本

Table
表 12-8 步驟一

步驟	I	P	O	驗證
一	PIM 類別圖圖檔 +Prompt1	生成式 AI	PlantUML 類別圖腳本	完整性 正確性

輸入：PIM檔案（便當王系統類別圖） **輸出**：便當王系統PlantUML類別圖腳本

將便當王系統的類別圖圖檔（實體類別的名稱與屬性）（如圖 12-1 所示）與 Prompt1（如表 12-5 所示）一併輸入 ChatGPT，生成符合 PlantUML 語法的類別圖腳本（如表 12-9 所示）。

◎驗證

將生成的 PlantUML 類別圖腳本匯入 PlantText 轉回圖形（如圖 12-2 所示），並與便當王系統類別圖（如圖 12-1 所示）逐項比對，確認並修正至一致。主要驗證項目包括：各類別名稱與數量是否一致、各類別屬性名稱（使用英文類別圖可