Microsoft® .NET 的 Rational XDE™ 模型建構準則

目錄

1. 簡介

此文件提供有關如何在 Rational XDE Microsoft .NET Edition 代表及建構 RUP 模型構件的建議。當然，您是否決定要在 XDE 中爲這些 RUP 構件建模，是專案特定決策。在本文件中，我們加註 XDE 提供自動化支援的那些模型，以及不提供自動化支援的那些模型，這會影響您的決定。

由於所有 XDE 模型存在於 XDE 專案內，因此 XDE 專案結構這一節提供有關應建立哪些 XDE 專案以及應在那些專案中建立哪些 XDE 模型的建議。

RUP 和 XDE 都使用「模型」這個詞彙，而 RUP 模型和 XDE 模型之間的對映不一定是一對一。在 RUP 模型對 XDE 模型的對映這一節，有描述 RUP 模型到 XDE 模型的對映。

XDE 模型檔中每一個 RUP 模型構件的結構，會在它自己的區段中加以描述。

2. 範圍

本文件重點在描述建議的 XDE 模型檔結構，而不是在開發相關聯的 RUP 構件內容的流程上。本文件也不描述詳細啓發，它定義包含所描述之 XDE 模型的 XDE 專案。如需如何定義、開發及建模 RUP 構件內容的相關資訊，請參閱 RUP。如需專案的詳細資訊，請參閱 IDE 文件。

本文件不說明完整範例，而是使用已選取的範例來強調涵蓋的要點；然而，所有範例均彼此一致，而且是取自實際的 XDE 模型。

本文件所描述的專案和模型結構只是建議，您可以用任何同等有效的結構加以取代。

3. XDE 專案結構

本文件的焦點在於如何建構 XDE 模型。然而，由於所有 XDE 模型都存在於 XDE 專案內，因此，我們一定要提供專案結構的簡介，其中含有我們建議的模型結構。

VS.NET 解決方案是專案的集合，在每一個專案內可以有一或多個 XDE 模型檔。[1]因此，專案結構會影響建立的模型檔數目及其內容。

.NET Enterprise 應用程式可由多個專案組成，視應用程式如何架構而定。比方說，如果應用程式實作 XML Web 服務、Windows 和 Web 介面，則建議解決方案分別具有 Web 服務、Windows 應用程式和 Web 應用程式專案。如需不同 VS.NET 專案範本的相關資訊，請參閱 VS.NET 說明。

對於許多人正在開發的 .NET 企業應用程式而言，我們建議您建立下列 XDE 專案和模型

[1] XDE 定義兩種類型的模型檔 - 程式碼和非程式碼模型檔。程式碼模型檔可用來建立專案的 C# 語言特定元素的模型，非程式碼模型檔則不對映到實作語言，而是作爲分析和設計模型。與專案相關聯的程式碼模型檔只能有一個，而與專案相關聯的非程式碼模型檔可以有許多個。

XDE 專案	說明	XDE 模型 <建議的模型名稱> (<XDE 檔案類型：模型範本>]
應用程式專案（XDE 基本建模專案）	應用程式專案代表整個應用程式。包含說明全部應用程式的 XDE 模型檔	- 使用案例模型（Rational XDE：使用案例模型） - 分析模型（Rational XDE：分析模型） - 整體設計模型（Rational XDE：設計模型） - 整體實作模型（Rational XDE：空白模型） - 部署模型（.NET：部署模型）
資料建模專案（XDE 資料建模專案）	資料建模專案包含應用程式資料建模所需的資源，以及資料模型與資料庫之間的來回轉換工程。	- 邏輯資料模型（資料：邏輯資料模型） - 實體資料模型（資料：*供應商特定的實體資料模型檔*）[2] - 網域模型（資料：*供應商特定網域模型檔*）”
.Net 類別庫專案（XDE 類別庫建模專案）	類別庫專案是包含 C# 元素（例如類別、介面等等）的 # VS.NET 專案，這些是要實作類別庫所需要的（例如本範例中的拍賣管理程式）。內含的元素已套裝及部署爲 .NET 組合。 請注意，.NET 應用程式可包含多個這樣的類別庫專案。 如前所述，專案數量的選取是架構選項，可因不同應用程式而異。	- .Net 程式碼模型（.Net：.Net 程式碼模型） - .Net 部署模型（.Net：.Net 部署模型）
Web 專案（XDE Web 建模專案）	Web 專案代表應用程式的 Web 資源。內含的元素已套裝及部署在 .NET 組合中。 個別的 Web 專案可定義給呈現邏輯的特定領域。建議爲每一個需要產生的組合建立一個 Web 專案。如果有定義個別專案，則專案的名稱應反映其內容。	- .Net 程式碼模型（.Net：.Net 程式碼模型） - .Net 部署模型（.Net：.Net 部署模型）

同時使用「解決方案瀏覽器」和「XDE 模型瀏覽器」視圖的這個專案和模型組織的範例顯示在圖 1。「解決方案瀏覽器」視圖是顯示在圖的左邊，「模型瀏覽器」視圖顯示在右邊。

[2] Rational XDE 爲多個資料庫供應商提供實體資料庫支援。 XDE 支援的每一個資料庫供應商有供應商特定的範本。

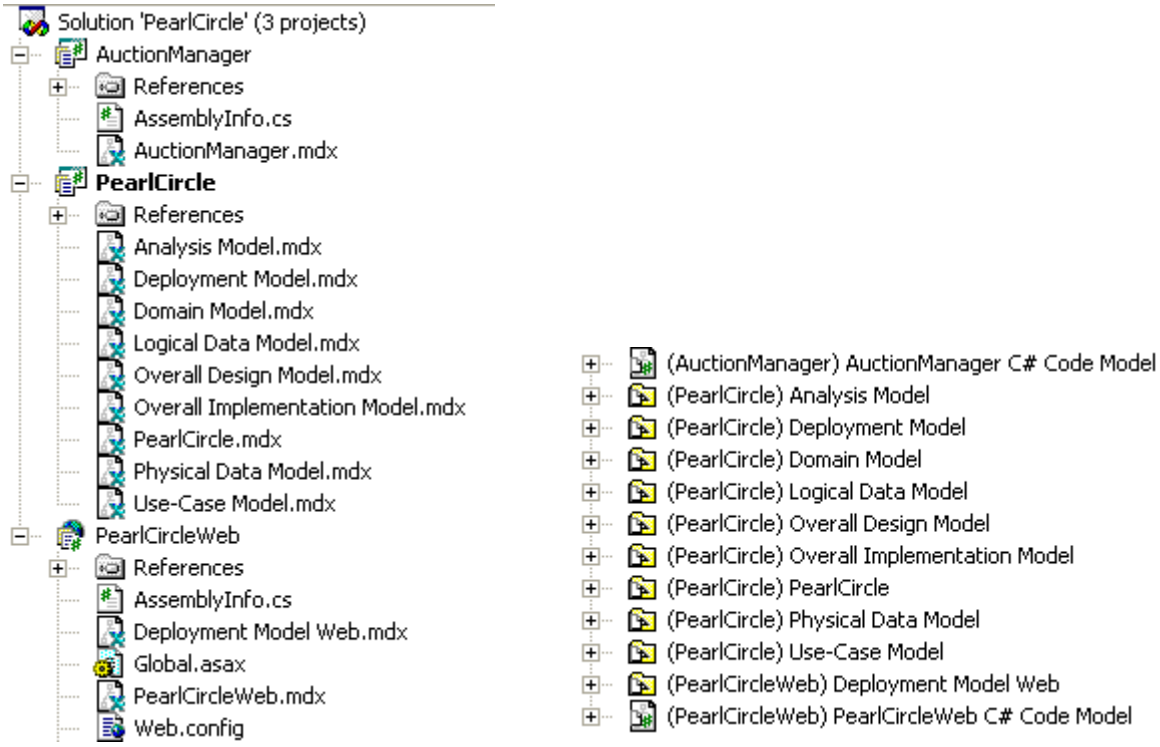

圖 1：專案模型和組織

另外，如果應用程式真的很小，而且只由一人開發，則上述專案結構可簡化成兩個專案，一個包含全應用程式和非 Web 元素，另一個包含 Web 元素。除了減少專案數之外，也可以減少模型數。例如，對於小型單人專案，可簡化如下：

- 不維護個別分析模型。分析與設計都是在 XDE 來回轉換模型中執行。
- 不維護「整體設計模型」和「整體實作模型」。專案夠小，故可直接查看 XDE 來回轉換模型而得到概觀。同時，使用案例實現化是在 .NET 程式碼模型中維護。
- 不維護個別邏輯資料模型。實體資料綱目直接在「實體資料模型」中開發。

下表是這種「小型專案結構」的總結。

XDE 專案	**說明**	**XDE 模型** **<建議的模型名稱> (<XDE 檔案類型：模型範本>]**
應用程式專案（XDE 類別庫建模專案）	應用程式專案代表應用程式的非 Web 層面。它包含描述整體應用程式的模型、資料模型和 .NET 特定模型。	- 使用案例模型（Rational XDE：使用案例模型） - 實體資料模型（資料：*供應商特定的實體資料模型檔*） - .Net 程式碼模型（.Net：.Net 程式碼模型） - .Net 部署模型（.Net：.Net 部署模型）
Web 專案（XDE Web 建模專案）	Web 專案代表應用程式的 Web 資源。內含的元素已套裝及部署在 .NET 組合中。	- .Net 程式碼模型（.Net：.Net 程式碼模型） - .Net 部署模型（.Net：.Net 部署模型）

小型專案和模型組織的一個範例顯示在圖 2

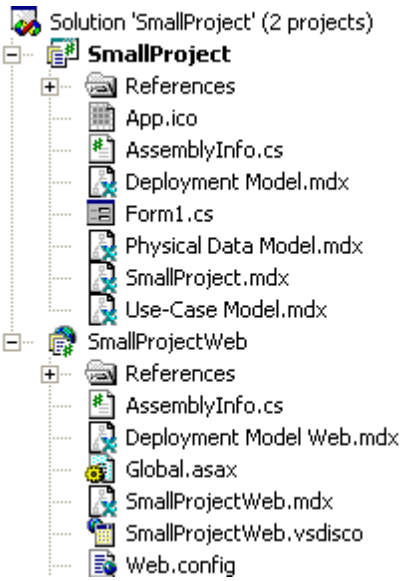

圖 2：小型 XDE 專案和模型組織範例

專案和個別模型檔數量的實際選取是架構選項，可因不同專案而異。然而，不論定義了多少個專案，每一個專案只能有一個 XDE .NET 程式碼模型檔。如需專案和它們可包含的 XDE 模型檔的相關資訊，請參閱 XDE 文件。

同時強烈建議，在所有 XDE 專案中，*XDE 模型名稱必須是唯一的*。在嘗試解決 XDE 模型之間的參照時，這點非常重要。如需交互模型參照及解決問題的相關資訊，請參閱 XDE 文件。

顯示在圖 1 中的 XDE 模型的結構是本文件剩餘內容的焦點。

4. RUP 模型到 XDE 模型的對映

在說明如何在 XDE 表示 RUP 模型構件之前，務必先解決「RUP 模型」與「XDE 模型」之間的混淆問題，因爲它們是不同的，而且從 RUP 模型對映到相關聯的 XDE 模型不一定一對一（接近，但不是一對一）。由於「模型」同時使用於 RUP 和 XDE 中，因此初步假設它們應該相同。然而，RUP 中的模型分隔流程重點（分析 vs. 設計 vs. 實作等等），而 XDE 中的模型則分隔開發重點（分隔程式碼模型以說明程式語言套裝結構與虛擬目錄結構，爲不同的程式語言和開發環境分隔程式碼模型等等）。爲了減輕混淆情形，在本白皮書的內容中，「模型」這個詞彙明確地限定爲 RUP 或 XDE。

下表彙總 RUP 模型到 XDE 模型的對映。 XDE 模型就是在 XDE 專案結構這一節所介紹的那些模型。每一個 XDE 模型的結構將在本白皮書的後面章節加以描述。

RUP 模型	<XDE 專案>: < XDE 模型名稱>
使用案例模型	應用程式專案：使用案例模型
分析模型	應用程式專案：分析模型
設計模型	應用程式專案：設計模型
資料模型	XDE 資料模型 資料建模專案：邏輯資料模型 資料建模專案：*供應商特定*實體資料模型 資料建模專案：*供應商特定*網域模型
實作模型	應用程式專案：實作模型
部署模型	應用程式專案：部署模型

5. 使用案例模型

「使用案例模型」的建議結構顯示在圖3

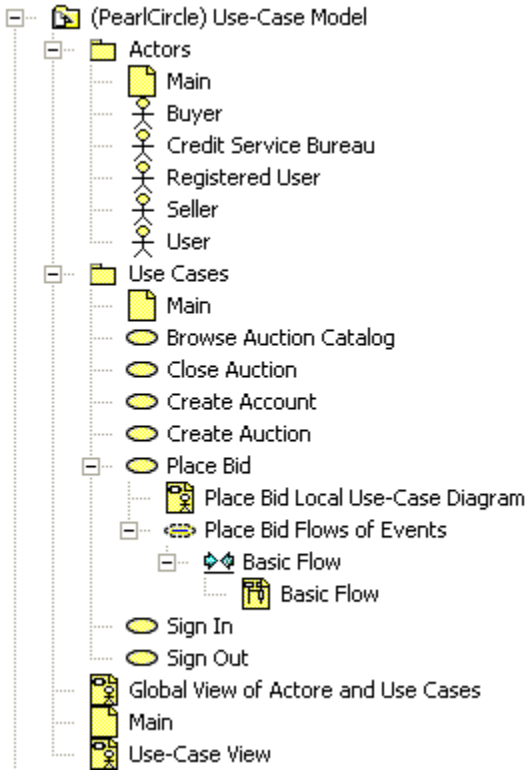

圖3：「使用案例模型」結構

「使用案例模型」分割爲兩個套件：參與者和使用案例。

除了包含**參與者**和**使用案例**的**使用案例模型**圖表之外，還可以使用其他圖表來闡明**使用案例**的不同層面。下列補充模型元素可併入到**使用案例模型**中的**使用案例**模型元素之下，如圖 3 所示

- 「競標區域使用案例圖表」圖表包含「競標」**使用案例**和參與該**使用案例**的**參與者**。
- 「事件競標流程」合作實例包含互動圖表，以圖形方式說明使用案例說明中所描述的事件流程（亦即，**參與者**和**使用案例**之間的互動）。此合作實例不應與**使用案例實現化**混淆不清，後者描述於分析模型這一節和設計使用案例實現化這一節，因爲「使用案例模型」中的合作實例完全爲「黑箱作業」，並不說明應用程式內各元素的互動。
- 「事件競標流程」活動圖形包含活動圖，以圖形方式說明使用案例說明中所描述的事件流程。

在圖 3 所顯示的範例中，圖 3 中的「參與者和使用案例的廣域視圖」圖表包含所有**使用案例**和**參與者**及其關係，這和「主要程式」圖表不同，「主要程式」圖表只包含有「主要程式」圖表的套件中的元素。如果有許多**參與者**和**使用案例**，可使用多個圖表來表達「參與者和使用案例的廣域視圖」圖表的資訊。

「使用案例視圖」圖表代表軟體架構的使用案例視圖。如需架構視圖的相關資訊，請參閱 RUP。

您可以在*參與者*和*使用案例*套件中建立其他套件，以進一步組織內含的模型元素，如圖 4 所示

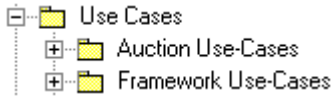

圖 4：其他的使用案例套件分割

6. 分析模型

分析模型是**分析類別**和分析**使用案例實現化**所在之處。

附註：是否應該維護個別的**分析模型**和**設計模型**，屬於專案特定決策。如果未維護個別的**分析模型**，則**分析類別**將移到適當的**設計模型**分割[3] 已修正。 另一個選項是在**設計模型**中建立**分析類別**和**性能原則實現化**，然後從該處將它們發展到設計表單中。請參閱設計模型這一節，以取得有關如何在 XDE 代表**設計模型**的相關資訊。

分析模型的建議結構顯示在圖 5

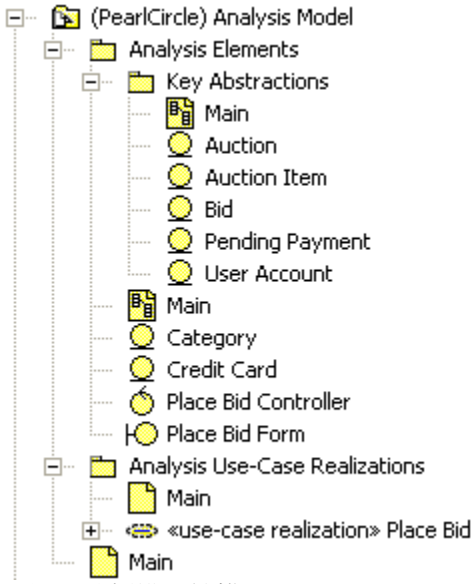

圖 5：分析模型結構

分析元素套件包含**分析類別**。**分析類別**的實例出現在「分析使用案例實現化」套件的圖表中。

除了**分析類別**之外，也可以在分析元素套件中定義套件，以進一步分割內含的**分析類別**（請參閱圖 5 中的「關鍵摘要」套件）。此額外分割是選用性的，尤其不維護個別 **分析模型**時，可選用它。在這些情況中，**分析類別**可視爲暫時性（亦即，它們會存在直到發展成設計元素爲止），因此其組織被認爲不重要。一個可能的異常狀況是關鍵摘要**分析類別**。

如圖 5 所示，「關鍵摘要」套件包含的**分析類別**，被認爲代表系統的關鍵摘要。如前所述，這個套件是選用性的。另一個替代方案是在「分析元素」套件中的類別圖上代表關鍵摘要。然而，建立個別套件能夠更明確地將**分析類別**分類爲關鍵摘要。事實上，即使未全面維護個別**分析模型**，有些專案仍選擇維護關鍵摘要 **分析類別**。在這些情況中，定義個別套件來包含所維護的**分析類別**會有幫助。
附註：關鍵摘要也會出現在「整體設計模型」的「邏輯視圖：關鍵摘要」圖表中。如需相關資訊，請參閱設計模型這一節。

「分析使用案例實現化」套件包含分析層次**使用案例實現化**，它們就分析元素套件中的**分析類別**而論，說明如何執行**使用案例**。每一個分析**使用案例實現化**會實現**使用案例模型**中的一個**使用案例**，其名稱與該

[3]稍後您會看到，正確的「整體設計模型分割」就是其中一個 XDE 來回轉換模型中的一個套件，因爲技術特定元素的設計是在來回轉換模型中執行。

使用案例相同，而且應具有圖 6 所顯示的結構：

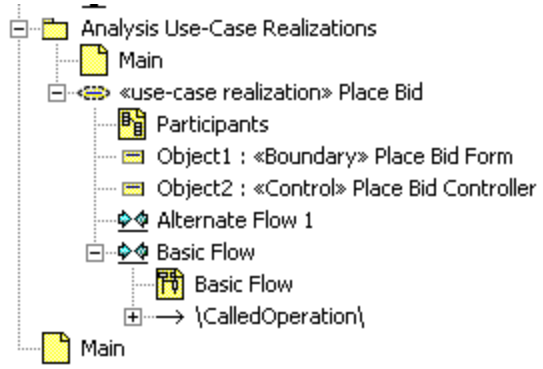

圖 6：分析使用案例實現化套件結構

「參與者」圖表顯示參與**使用案例實現化**的**分析類別**（來自分析元素套件），亦即，其實例出現在互動圖中的那些**分析類別**，以及支援互動圖所描述之協同作業的關係。

「流程」互動實例（「基本流程」和「替代流程 1」）包含序列圖，說明事件的**使用案例**流程。每一個事件重要使用案例流程應該有一個互動實例。在相關聯的**使用案例**的執行期間，互動實例中的序列圖說明參與的**分析類別**之間的流程。

7. 設計模型

RUP **設計模型**由多個 XDE 模型表示 –「整體設計模型」和位於個別 XDE 來回轉換模型中的來回轉換設計元素（來回轉換設計元素是參與來回轉換工程的詳細設計元素）。如此一來，即可運用個別來回轉換模型中可用的自動化。

「整體設計模型」說明整體應用程式的設計，並包含跨越多個來回轉換模型的元素。它包含邏輯分割區 (LP) 來啓發個別來回轉換模型的組織，以及**使用案例實現化**來連結一切（**使用案例實現化**說明不同來回轉換模型的設計元素之間的協同作業）。「整體設計模型」包含了參照來回轉換設計元素的圖表。如需個別 XDE 來回轉換模型的相關資訊，請參閱實作模型 這一節。

另一個可能性是在相同 XDE 程式碼模型中代表**設計模型**和**實作模型**。唯有當您只有一個目標實作語言，而且團隊規模很小時，這才行得通。

維護「整體設計模型」是選用性的，但卻是組織圖表、提升摘要層次及提供設計元素場所的好辦法，而且還能得知要應用的實作機制。

「整體設計模型」的建議結構顯示在圖 7

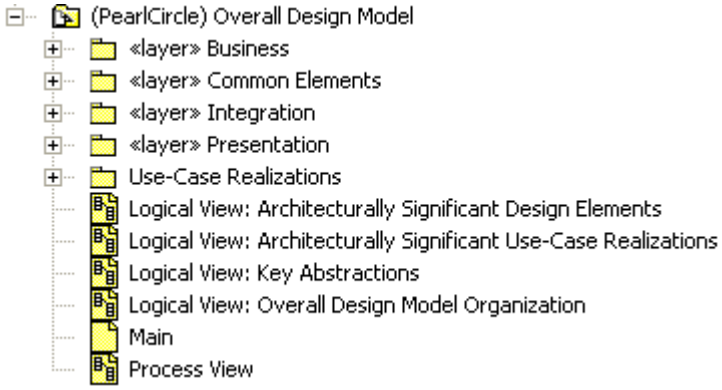

圖 7：整體設計模型結構

此「整體設計模型」包含下列套件：

- <<layer>> 套件包含（或包含參照的圖表）系統的設計元素（**設計類別**、 **介面**和**設計子系統**）。 此結構代表特定分割策略，請參閱 設計層 這一節的描述。
- 使用案例實現化套件包含設計層次使用案例實現化。使用案例實現化的內部結構在 設計使用案例實現化這一節有更詳細的討論。

代表架構視圖的圖表，其圖表名稱中有包含 “View”。如需架構視圖的相關資訊，請參閱 RUP。

「邏輯視圖：關鍵摘要」圖表包含系統的關鍵摘要。有數個選項可維護這些關鍵摘要：

- 有維護完整的**分析模型**。在該情況下，「邏輯視圖：關鍵摘要」圖表包含來自**分析模型**的**分析類別**，它們代表系統的關鍵摘要。
- 只維護部分**分析模型**，亦即，只有關鍵摘要。在該情況下，「邏輯視圖：關鍵摘要」圖表包含來自**分析模型**的**分析類別**，它們代表系統的關鍵摘要。
- 不維護**分析模型**的任一部分。在該情況下，代表關鍵摘要的**分析類別**可維護在**設計模型**的一個套件中，它叫作關鍵摘要

如需**分析模型**的相關資訊，請參閱分析模型」這一節。

7.1 設計層

<<layer>> 套件包含系統的設計元素（例如**設計類別**、**介面**和**設計子系統**），它們是從**分析類別**發展而成的。 <<layer>> 套件可包含任何數量的子套件，它們進一步分割內含的設計元素。設計**使用案例實現化**（包含在「整體設計模型」的使用案例實現化套件中，並於設計使用案例實現化這一節的標題下加以討論）是從這些套件包含的設計元素方面來撰寫的。

設計模型可遵循任何數量的分割策略。這一節所描述的分割策略顯示在圖 8

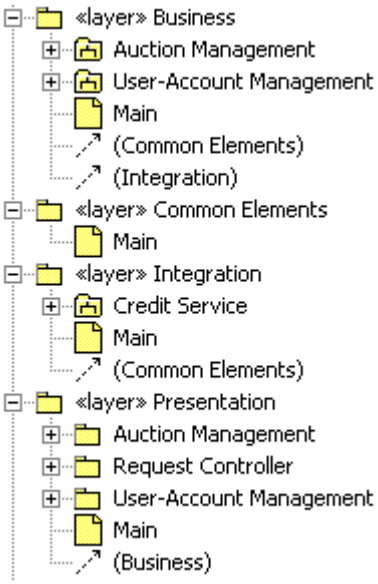

圖 8：設計套件分割範例

在這個範例中，第一層套件被視爲層，其中每一層有特定責任。第二層套件依商業功能進一步分割層套件元件。

呈現層套件負責處理與一般使用者的互動。在 .Net 應用程式中，可能位於呈現層套件的設計元素包括 Active Server Pages (ASP.NET)。您可以進一步將呈現層套件分割成子套件，將屬於同一組相關**使用案例**的元素加以分組；例如圖 8 的拍賣管理套件：

商業層套件負責執行任何商業流程。在本文件所呈現的「整體設計模型」結構中，商業層套件包含一組設計子系統套件，每一個主要商業功能各一個（例如圖 8 的拍賣管理和使用者帳號管理子系統套件）。**設計子系統**套件在設計子系統這一節的標題下有更詳細的描述。

整合層套件負責提供後端系統資源的存取，包括資料庫和外部系統。在本文件所呈現的**在設計模型**結構中，整合層套件也包含設計子系統套件，每一個外部系統各一個（例如圖 8 的信貸服務子系統套件）。**設計子系統**套件在設計子系統這一節的標題下有更詳細的描述。

共用元素層套件包含各層共用的元素。

同樣地，這一節所描述的結構可以用不同的結構來取代，以反映不同的分割策略。

7.2 設計子系統

設計子系統由「整體設計模型」中的子系統套件來表示。每一個設計子系統套件應該具有相同的結構。結構的細節因所擷取的**設計子系統**的詳細程度而異。

更正式而嚴格的**設計子系統**結構的範例顯示在圖 9

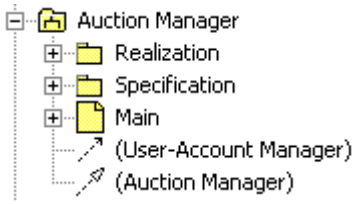

圖 9：設計子系統結構

此設計子系統套件結構支援在設計子系統套件內定義個別的規格和實現化套件。此結構受到 *UML Components: A Simple Process for Specifying Component-Based Software*（作者：J. Cheesman 和 J. Daniels）這本書的影響。您可以使用不包含這些分割的簡化設計子系統套件結構，這並不影響本文件所定義的其他模型檔案結構。下列章節將討論每一個規格和實現化套件。

7.2.1 子系統規格

規格套件包含**設計子系統**介面的說明。[4]子系統規格的一個範例顯示在圖 10

圖 10：設計子系統規格範例

[4]在這個簡單的範例中，您可能想知道針對該介面的個別套件需求。然而，在真實專案中，該套件是值得維護的，因爲它可包含對於說明該子系統的參照，尤其，它包含介面限制，例如作業的先決條件和後續條件。

7.2.2 子系統實現化

實現化套件包含**設計子系統**規格如何實現的說明。設計子系統套件的實現化套件的範例顯示在圖 11

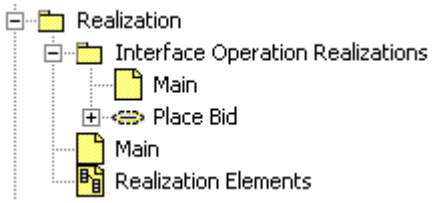

圖 11：設計子系統實現化範例

「實現化元素」圖表包含對於實現子系統的設計元素的參照。設計元素本身可位於 .Net 程式碼模型中，它們在其中參與來回轉換工程。如果需要詳細資訊，請參閱實作模型」這一節。

介面作業實現化套件包含合作實例，說明子系統元素如何實現**設計子系統** 介面的重要作業（在規格套件中）。每一個重要子系統介面作業有一個合作實例。[5]介面作業實現化套件的一個範例顯示在圖 12

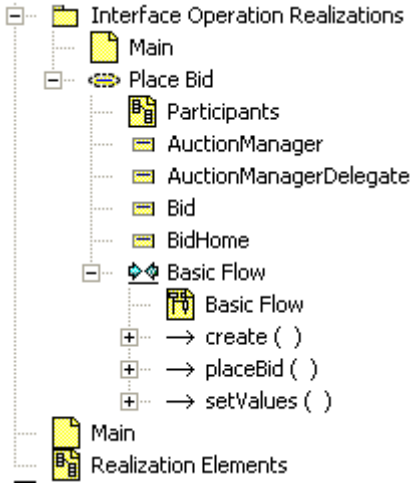

圖 12：介面作業實現化套件範例

就像分析層次**使用案例實現化**（先前已在 分析模型 這一節討論過）和設計層次**使用案例實現化**（稍後在設計使用案例實現化中會加以討論），每一個介面作業實現化都包含類別圖，其中有參與實現化的子系統元素（圖 12 中的「參與者」圖表），以及一個互動圖，它說明那些參與者如何分工合作來執行子系統介面作業（圖 12 中的「基本流程」圖）。

[5]並非所有作業都需要定義在這個層次。有些較簡單的作業不需要個別的合作。

7.3 設計使用案例實現化

使用案例實現化套件包含設計層次**使用案例實現化**。每一個**使用案例實現化**與**使用案例模型**中的一個**使用案例**相關聯，其名稱與該**使用案例**相同，而且應具有「圖 13」所顯示的結構。

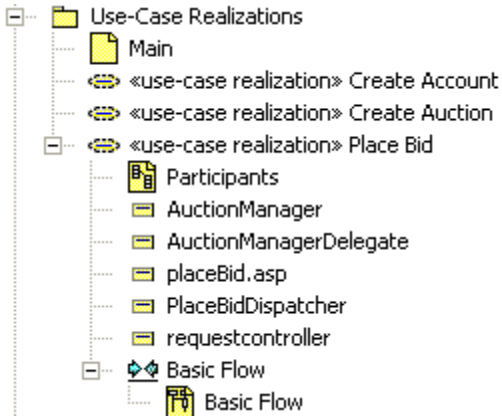

圖 13：設計使用案例實現化結構

使用案例實現化「參與者」圖表顯示參與**使用案例實現化**的設計元素（亦即，其實例出現在**使用案例實現化**互動圖中的那些設計元素），以及支援互動圖所描述之協同作業的關係。

「基本流程」圖是互動圖的一個範例，它說明在相關聯的**使用案例**的執行期間，參與的設計元素之間的流程。**使用案例**中的每一個事件流程應該有一個互動實例。

請注意，**使用案例實現化**圖表可能（而且經常會）包含對於設計元素的參照，這些設計元素實際上是位於個別 XDE 來回轉換模型中。**使用案例實現化**示範個別的來回轉換模型中的元素之間的合作關係。

8. 資料模型

RUP **資料模型**是由多個 XDE 模型檔來表示：

- **邏輯資料模型**（選用性）。代表邏輯資料模型，是資料庫的邏輯設計之應用程式獨立視圖。
- **實體資料模型**。代表資料庫供應商特定的實體資料模型。它包含詳細模型元素來定義資料庫表格的特定性質。「實體資料模型」XDE 模型檔也包含資料庫特定實作構件，來於供應商特定資料庫中實作這些表格。
- **網域模型**（選用性）。代表可用來定義「實體資料模型」之一致資料類型的資料庫供應商特定資料類型。

XDE 模型檔的分隔提供了最佳彈性來支援 整體設計模型、**資料模型**和實體資料庫之間的自動化。

以下對每一個 XDE 模型檔有更詳細的說明。

8.1 邏輯資料模型（選用性）

當專案需要建立對資料庫設計很重要的主要實體和關係的獨立式邏輯資料表示法時，「邏輯資料模型」可使用於這類情況。建立 XDE 邏輯資料模型是選用性的，因爲資料庫設計團隊可以將**設計模型**中的持續性**設計類別**轉換成**資料模型**中的表格，直接在 XDE 實體資料模型中建立起始實體資料庫設計結構（請參閱 Error! Reference source not found.一節）。

XDE 邏輯資料模型可依需要分割成主題區套件。主題區套件定義實體類別的邏輯群組。 XDE 邏輯資料模型也可包含「共用元素」套件，內含跨越主題區的模型元素。

名稱中有 “View” 的圖表是用來記載架構的資料視圖。「資料視圖：整體邏輯資料模型組織」圖表是用來

記載邏輯資料模型的高階資料組織，如 XDE 邏輯資料模型的主要分割（亦即套件）中所表示。「資料視圖：主要邏輯資料元素」是用來記載**資料模型**的主要邏輯元素。如果有維護邏輯資料模型（亦即，有個別的「邏輯資料模型」），則這個圖表將包含 XDE 邏輯資料模型的元素。如需架構視圖的相關資訊，請參閱 RUP。

XDE 邏輯資料模型的建議結構的一個範例顯示在Error! Reference source not found.

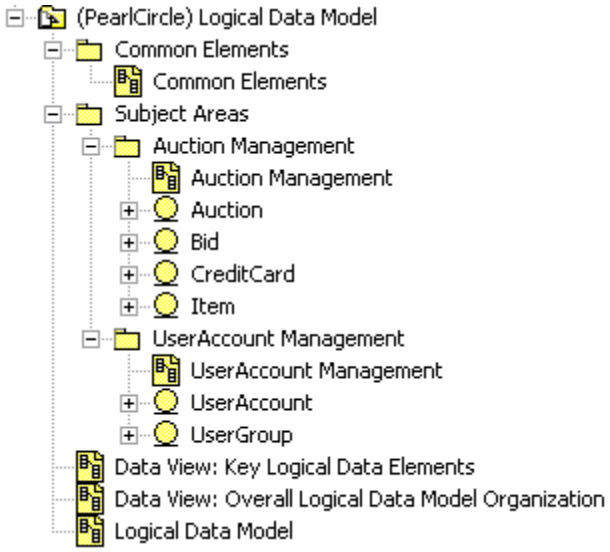

圖 14：XDE 邏輯資料模型結構

在這個範例中，有兩個主題區套件：拍賣管理和使用者帳號管理。每一個主題區套件包含構成邏輯資料模型的實體類別。但並沒有直接對映到**設計模型**中的套件結構，只是有些類似而已。

8.2 實體資料模型

實體資料模型包含詳細的資料庫表格和預存程序設計，它們透過 XDE Data Modeler 正向工程機能來實作資料庫。實體資料模型也包含用來定義資料庫實體儲存體配置的模型元素。一般而言，模型元素包含資料庫和表格空間，它們再構成目標儲存媒體上的資料庫表格的實際佈置。

在建立 XDE 實體資料模型時，資料庫設計師必須選取適當的目標資料庫。支援的資料庫包括：DB2 MVS、DB2 UDB、Oracle、Sybase 和 SQL Server。 XDE 將 XDE 模型檔名稱預設爲已選取的資料庫。在本文件的「實體資料模型」範例中，XDE 模型檔名稱已更新爲「實體資料模型」。在建立「實體資料模型」時，資料庫設計師可選擇接受預設名稱。

XDE 實體資料模型的建議結構的一個範例顯示在Error! Reference source not found.

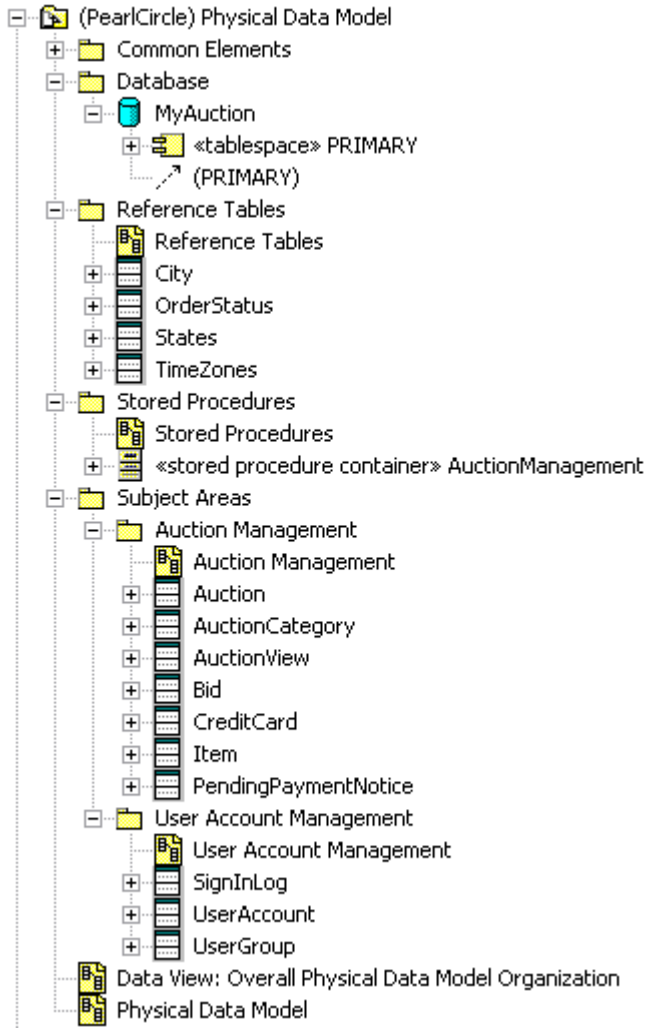

圖 15：XDE 實體資料模型結構

「共用元素」套件包含資料庫表格和跨越主題區的視圖。

資料庫套件包含定義資料庫實體儲存體配置的模型元素。它包含資料庫和表格空間，它們構成目標儲存媒體上的資料庫表格的實際佈置。表格空間是用來邏輯地群組資料庫內的表格。如需定義表格空間的準則，請參閱 RUP。資料庫套件可依需要分割爲較低階的套件，視應用程式的複雜度而定。

在Error! Reference source not found. 所顯示的範例中，資料庫套件包含單一資料庫，MyAuction，其相關聯的表格空間 PRIMARY 和表格實現化關係。表格空間可命名爲任何適合資料庫專案的名稱。對於 MyAuction 資料庫，只定義一個叫作 PRIMARY 的表格空間。執行正向工程時，會建立透過與資料庫表格空間的實現化關係而鏈結到資料庫的表格（在資料庫或 DDL 中）。

參照表格套件包含靜態資料表格，它們保留應用程式所需的「固定」資料資訊。

預存程序套件包含代表資料庫預存程序的所有類別（<<stored procedure container>> 類別和相關聯的 <<stored procedure>> 作業）。與單一表格相關的預存程序可連同預存程序所參照的表格，一起套裝在預存

程序套件或主題區套件中，視您要呈現一個「以預存程序爲中心」或「以表格爲中心」的視圖而定[6]

「主題區」套件包含邏輯地分組相關表格和視圖集合的套件[7]建議在主題區套件中建立視圖及表格。此建議純粹是爲了組織而已。在有使用視圖的主題區中保存視圖很有幫助，這樣可以把它們放在與表格一樣的主題區內。在Error! Reference source not found. 所顯示的範例中，有兩個主題區套件：拍賣管理和使用者帳號管理。主題區套件的數量視應用程式的複雜度而定。然而，一般而言，「邏輯資料模型」中的主題區套件會「啓發」實體資料模型中的主題區套件。邏輯資料模型中的主題區是實體資料模型中的主題區的摘要。

主題區套件中的表格包含對表格所定義的直欄和觸發程式。表格是透過下列其中一項而建立

- XDE 類別到表格轉換功能。
- XDE 對現有的資料庫功能進行反向工程[8]
- 由資料庫設計師手動建立。

對現有的資料庫進行反向工程時，會在 XDE 實體資料模型中建立綱目套件。這些套件的名稱是以資料庫擁有者爲基礎[9] 進行反向工程的資料庫。建議反向工程的表格移到主題區套件內的主題區套件中，並刪除反向工程的綱目套件。將表格移到主題區套件的作用是組織表格，使資料庫設計師可以依需要來更新表格。”

名稱中有 “View” 的圖表是用來記載架構的資料視圖。「資料視圖：整體實體資料模型組織」圖表是用來記載實體資料模型的高階資料組織，如 XDE 實體資料模型的主要分割（亦即套件）中所表示。如需架構視圖的相關資訊，請參閱 RUP。

8.3 網域模型（選用性）

網域模型是選用性的 XDE 模型，用來儲存資料庫的使用者定義資料類型。網域可讓資料庫設計師在資料庫設計上重複使用元素內容。資料庫設計師使用網域，在資料庫各處一貫地記載直欄內容。直欄的名稱是定義在表格中；網域是用來定義直欄的 *TypeExpression*。

[6]以表格爲中心的視圖可讓您只靠一個視圖就可以更瞭解資料庫設計/作業。以預存程序爲中心的視圖可簡化尋找和變更/維護預存程序的工作。

[7]有些人不瞭解如何在實體資料模型中使用主題區套件，因爲它需要額外的維護工作，來維護邏輯和實體資料庫主題區套件。實體資料模型中的主題區是爲了與邏輯資料模型（如果有使用它的話）一致，這對於「大型」實體資料模型以及沒有邏輯資料模型的情況更是如此。在這種情況下，主題區套件可用來管理從「類別到表格」轉換而產生的表格。

[8]通常資料庫會進行反向工程一次，然後使用 XDE 的 Compare 和 Sync 功能，使所有未來的更新同步。

[9]在 XDE 內，會擷取資料庫擁有者作爲 <<database>> 元件的一項內容。在作爲連線字串的「位置」內容內，有一個綱目屬性。對資料庫進行反向工程時，這通常是指資料庫擁有者。

XDE 網域模型的建議結構的一個範例[10] 顯示在Error! Reference source not found.

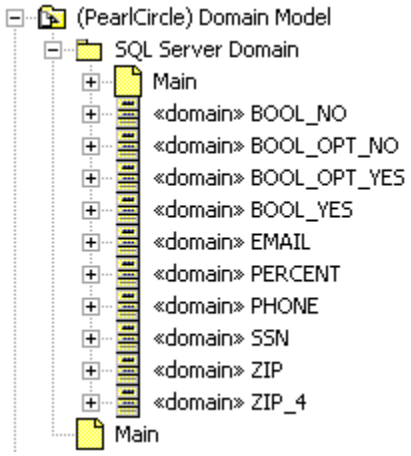

圖 16：XDE 網域模型結構

這個範例示範在「SQL Server 網域」套件內組織的 SQL Server 網域值。當資料庫設計師定義大量網域時，資料庫設計師可能需要使用「SQL Server 網域」套件下的套件來組織網域。

9. 實作模型

如 RUP 中所定義，**實作模型**包含的實作元素有視覺化表示法和實體表示法兩種（亦即，代表實作元素的 UML 元素，以及檔案系統中的實體檔）。關於**實作模型**的 XDE 價值在於其透過來回轉換工程自動同步化這些個別表示法的能力。

在 XDE 內，**實作模型**是在多個 XDE 模型內表示，其中一個範例顯示在圖 20:

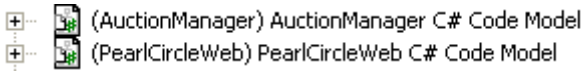

圖 20：實作 XDE 模型

在這個範例中，我們定義下列 XDE 模型檔來代表**實作模型**:

- 「AuctionManager C# 程式碼模型」是包含 Microsoft Visual C# 程式碼元素的 XDE 程式碼模型檔，這些程式碼元素構成拍賣管理程式**實作子系統**。此模型中的元素參與 XDE 來回轉換工程。
-
- 「PearlCircleWeb C# 程式碼模型」XDE 模型檔包含 ASP.NET C# 程式碼元素（Web Forms、Web 控制項和 HTTP 處理常式），它們構成 PearlCircle Web **實作子系統**。此模型對應到一個 VS.NET Web 應用程式專案。此模型中的元素參與 XDE 來回轉換工程。

請記住，每一個 VS.NET 專案只能有一個 XDE 程式碼模型檔。專案和個別模型檔數量的選取是架構選項，可因不同專案而異。如果需要詳細資訊，請參閱 XDE 線上說明。

每一個模型在後面章節有更詳細的描述。

9.1 類別庫專案的 C# 程式碼模型

「C# 程式碼實作模型」包含使用 C# 實作的元素。

[10]在 XDE 內，支援數個供應商資料庫，包括 DB2、Oracle、Sybase 和 SQL Server。建立網域 XDE 資料模型時，資料庫設計師將選取適當的供應商資料庫來建立網域 XDE 資料模型。 XDE 將建立已選取的資料庫供應商的預設網域清單。

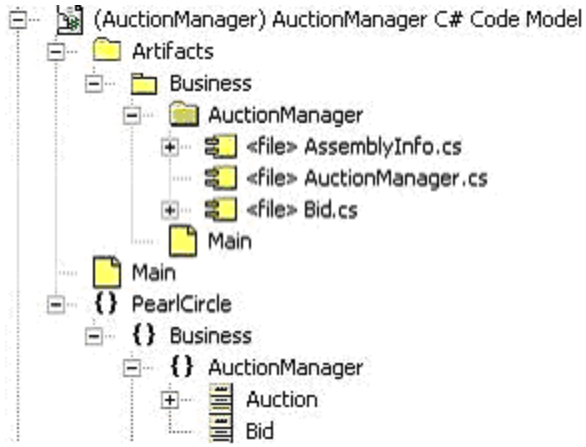

圖 21：「拍賣管理程式 C# 程式碼模型」結構

在這個範例中，「拍賣管理程式 C# 程式碼模型」的結構反映「整體設計模型」的結構（在 7這一節會加以討論）。每一個「整體設計模型」套件有一個套件（代表 .NET 命名空間），其內容將以 C# 實作（這包括 Serviced Components 和其他支援的 C# 類別）。 XDE 中的 .NET 名稱空間建模成爲具有 *namespace* 造型的套件（括住的大括號 '**{ }**'圖示）。由於 C# 程式語言不允許名稱空間名稱中有空白，所以 C# 名稱空間名稱與相等的「整體設計模型」模型元素的名稱可能不同。

如 圖 21所顯示，「C# 程式碼模型」包含程式碼檔案的視覺化表示法（.cs 元素）。 XDE 爲每一個它包含程式碼檔案的程式碼模型檔建立 *Artifacts* 套件。這些程式碼檔案對映到「整體設計模型」已定義的類別（請參閱 7這一節），這些類別已逐步形成並發展到可以實作的程度（以 XDE 而言，則爲可以進行來回轉換工程的程度）。

如 圖 21所示，「C# 程式碼模型」結構遵循的慣例是使用公司名稱作爲起始 C# 名稱空間名稱。範例應用程式的公司名稱爲 Pearl Circle。因此，包含實作元素的套件是放置在*PearlCircle* 名稱空間內。所以，*PearlCircle* 名稱空間內的所有 C# 元素會有一個完整名稱，其字首爲PearlCircle。例如，*AuctionManager* 名稱空間的完整名稱爲 *PearlCircleBusiness.AuctionManager*。使用公司名稱作爲起始　C# 名稱空間名稱的慣例可保證 C# 類別名稱爲唯一的，即使納入第三方 C# 類別庫亦然。

9.2 Web 應用程式專案的 C# 程式碼模型

XDE Web 模型包含對應於 Active Server Page .NET (ASP.NET) Web 應用程式專案的元素。圖 22 提供 PearlCircleWeb XDE 程式碼模型檔的範例，呈現 ASP.NET 專案的元素。

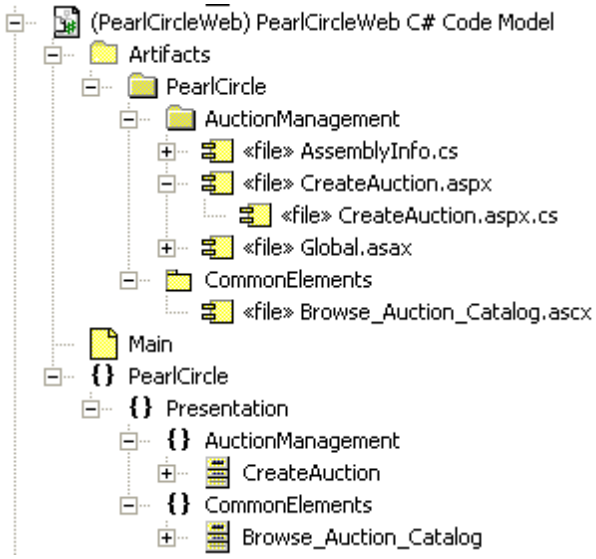

圖 22：「PearlCircle Web C# 模型」結構

在這個範例中，「PearlCircle Web C# 模型」的結構反映「整體設計模型」的結構（在 7這一節會加以討論）。「整體設計模型」中的每一個套件有一個名稱空間，其內容將以 C# 實作（這包括 code-behind 類別和其他支援的 C# 類別）。 XDE 中的 .NET 名稱空間建模成爲具有 *namespace* 造型的套件（括住的大括號 '**{}**'圖示）。由於 C# 程式語言不允許名稱空間中有空白，所以 C# 名稱空間名稱與相等的「整體設計模型」模型元素的名稱可能不同。

與 Web Forms 和 Web Controls 相關聯的 C# code-behind 類別（.aspx.cs 或 .ascx.cs）及其他支援的 C# 類別（例如 HttpHandlers）可使用 XDE 進行來回轉換工程。請注意，目前 XDE 對 .aspx 或 .ascx 檔案不支援 RTE。[11]*CreateAuction* 類別（顯示在 圖 22 中）將位於內部檔案 *CreateAuction.aspx.cs* 中的 code-behind 類別建模。在 Artifacts 套件中，*CreateAuctiont.aspx.cs* 檔案顯示在 *CreateAuction.aspx* 檔案之下。

如果所有重要的架構功能都包含在 code-behind 類別內，則唯一需要的就是自動產生的 XDE 程式碼模型。然而，萬一重要的架構功能是在 Web 控制檔 (.ascx) 內實作，則對應到這個檔案的類別可手動新增至此模型。範例之一爲 *Browse_Auction_Catalog* 類別，它顯示在 圖 22請注意，此類別是以手動方式新增至此圖表中。

10. 部署模型

部署模型是以 XDE 空白模型檔案名稱爲Deployment Model 代表。

「部署模型」包含節點及其連接，它們代表部署環境的網路配置。它也識別將部署在這些節點上的實作元素。

「部署模型」的一個範例顯示在圖 25

[11] Web 表單的一個單元包含兩個個別檔案：code-behind 檔案和 .aspx 檔案。如需相關資訊，請參閱 VS.NET 文件。

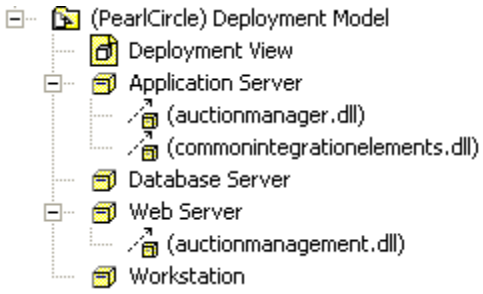

圖 25：部署模型結構

在這個範例中，所識別的節點爲「資料庫伺服器」、「應用程式伺服器」和「Web 伺服器」。auctionmanagement.dll 部署在 Web 伺服器上。auctionmanager.dll 和 commonintegrationelements.dll 是部署在應用程式伺服器上。

「部署模型」模型檔也可包含架構視圖圖表。在圖 25 中，「部署視圖」圖表代表架構的部署視圖。如需架構視圖的相關資訊，請參閱 RUP。