



## 第二章

# 概 觀

本章的內容在於讓你瞭解 UML，以及如何使用 UML。當你讀完這一章之後，將會獲得以下的知識：

- 對於問題、解決方案及解決問題的概觀。
- 生命週期的概念，包括發展週期和階段，以及循環的週期和階段。
- 問題和解決方案的概念，包括領域或空間、系統、結構、模型、結構觀、圖型、語言。
- 解決問題的概念，包括範例、產物、活動、試探的程序。

本章將介紹一些在 UML 底層的關鍵概念。我將先介紹解決問題所用的所有表示法，這牽涉到如何將工作量分配到整個生命週期，以便解決一個問題。然後詳細說明如何瞭解問題與解決方案，同時解釋如何明瞭解決問題的過程。

## 問題、解決方案及解決問題

組織生產及交付產品及服務，以解決顧客的要求及需求。將需求視為問題。那麼用以滿足需求的產品或服務則可視為解決方案。為了產生有價值的解決方案（在最少時間內，使用最低的成本並產生最高的品質），這些組織必須獲得（學習到）、傳達（共享）和使用知識。解決方案的價值是由產品或服務的品質、花費、以及所需時間來決定。在專業的領域中，這種生產者 - 顧客的關係存在於所有的階層：組織為顧客解決問題，職員為老闆解決問題，醫生為病人解決問題，依此類推。在這些形態的關係當中，知識是決定成功與否的主要因素。

組織使用技術來傳播和交換資訊，讓它們的職員能夠解決商業問題。技術只是一種工具，如果沒有人知道如何使用或發揮它的潛能，就算有再好的技術也是枉然。商業與技術領域不斷地變化，我們必須設法掌控這個持續增高的複雜度，不過，我們必須記住一件事：技術本身並不是目標，只是達成目標的途徑。

計畫（Project）代表的是解決問題所做的努力。這牽涉到瞭解一個問題，或將此問題概念化、解決這個問題、並實踐或理解其解決方案。參與計劃的人員或組織稱為 stakeholder。有些 stakeholder 負責解決問題，他們貢獻自己的知識以瞭解問題，並確認解決方案的正確性；其他的 stakeholder 則負責利用他們的知識來產生此解決方案。一個計劃的結果稱為 deliverables 或 work products。它們是用以解決一個問題的產品或服務。基本上來說，計劃將“努力工作並期待最佳結果”這個策略型式化以解決問題。

UML 是一個讓你能輕易達成以下目標的語言：

- 將問題詳細說明、具體化、瞭解、並製作說明文件。
- 取得、傳遞及運用在解決問題所需的知識。
- 將解決方法詳細敘述、具體化、建構、並製作說明文件。

不過，UML 並非針對任何一個特定的問題的解決策略。相反地，它非常具有彈性，可適用於任何的策略。

---

Program 是解決問題所做的努力（計劃）之集合。它們之間必須彼此協調及交流互動，以便讓每個計劃可得益於同一個程序內其它的計劃。要達成這個目標的關鍵就是溝通。如果能夠進行跨越計劃的溝通，那麼就可以很輕易地分擔工作量和共享知識，並能降低工作的困難度。UML 的角色就是提供這樣的溝通管道，並讓整件事變得容易。

假設某個組織想以最理想的方式，將職員分配到不同的計劃。它需要一個資訊系統來保存職員和計劃的資訊，此系統應能依職員的能力將他們分配到最能發揮所長的計劃中。在處理這個問題時，必須對於如何瞭解或概念化此問題、推導此問題的答案、及實踐或獲得此解決方案，有一個整體性的策略。這個策略將決定我們如何在以瞭解此問題的前題下審視此問題，以及如何在以獲得答案的前題下審視此答案。我們必須應用此時所擁有的知識，以及由其它經驗所得到的啟發，來產生解決方案。我們的努力將被安排（生命週期）成一連串（可能是同時進行的）的步驟（活動），最後完成一個資訊系統。我們將在“解決問題”那一節分析這些概念。

這些問題出現在商業的情況（領域或空間）。解決方案必須符合此組織的資訊技術基礎建設（領域或空間）。必須全盤瞭解商業的問題（系統），才能符合商業的需求；同時必須完全瞭解資訊系統，才能符合其需求。最後的資訊系統也必須符合此組織的技術基本結構。當我們將此問題概念化，並設法獲得解決方案時，將會獲得關於商業問題和系統的知識，以及對如何處理不同的問題做下決定（以結構的觀點），而且應該能夠敘述及交換這些資訊（圖形）。我們將在“問題和解決方案”那一節探討這些概念。

方法（Method）說明如何引導所有對解決問題所做的努力。它們說了解決問題的策略之概觀及其要素、說明了如何以解決問題的策略來審視問題和解決方案，這稱為一個方法的描述觀點，因為它敘述了如何獲得及交換與一個問題和解決方案有關的知識。方法也詳述如何利用一個解決問題的策略，來解決問題並獲得答案，這稱為一個方法的規則觀點，因為它規定了如何利用知識以解決問題。方法說明了如何審視問題和解決方案，以及如何實現對解決問題所做的努力。

程序（Process）是方法實現的結果。它們使用方法的問題解決策略，來解決一個問題並獲得答案。程序是由方法所指定，這稱為一個程序的靜態觀點，因為它是一個關於問題是如何被解決的敘述；程序利用計劃內的方法來解決問題，這稱為一個程序的動態觀點。

因為計劃可能非常地複雜，因此，方法應該能提供一個解決問題之程序的基礎結構。它們應被視為組成解決問題之程序的忠告或建議，而不是視為限制解決問題之技巧的硬性規定。方法論 (methodology) 指的是相關方法的分類學或有良好組織的集合；因此，一個方法或許可視為方法論的一部分。

想要瞭解 UML 如何能簡化問題的解決，必須更深入地瞭解問題、解決方案、及解決問題的組成要素。經過這個理解的過程之後，你就能對一個問題進行分析，並決定 UML 的哪些部份最適用於此問題的解決方案。

問題通常屬於 as-is，因為它們表示目前的狀態；解決方案則屬於 to-be，因為它們表示我們嘗試達到的目標狀態。請參考圖 2-1。

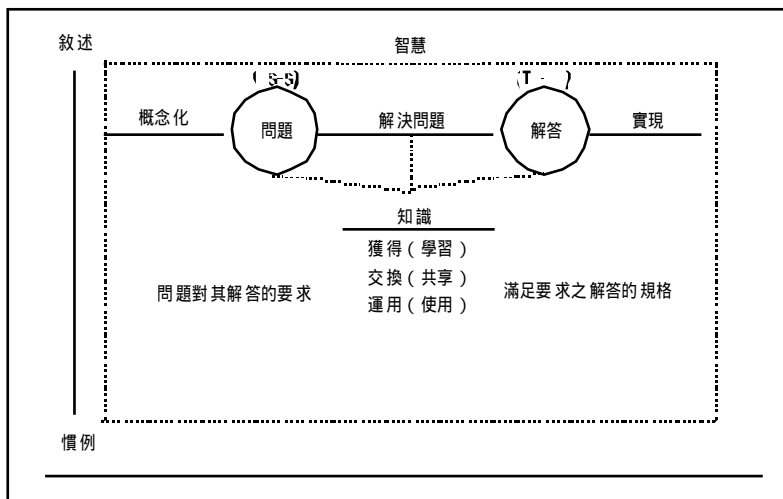


圖 2-1：問題解決的概觀

一個不會造成困難的狀態，仍會被視為問題，目的是為了問題的本質；儘管如此，解決方案只不過是對此狀態的瞭解而已，而結果可以決定這個狀態的特性 (characteristics)，而非需求 (requirements)，並提供一個純粹關於狀態的描述，而無關於任何的解決方案。這個過程通常發生在已經存在了問題的解決方案，以及必須瞭解和評估現存的解決方案時。

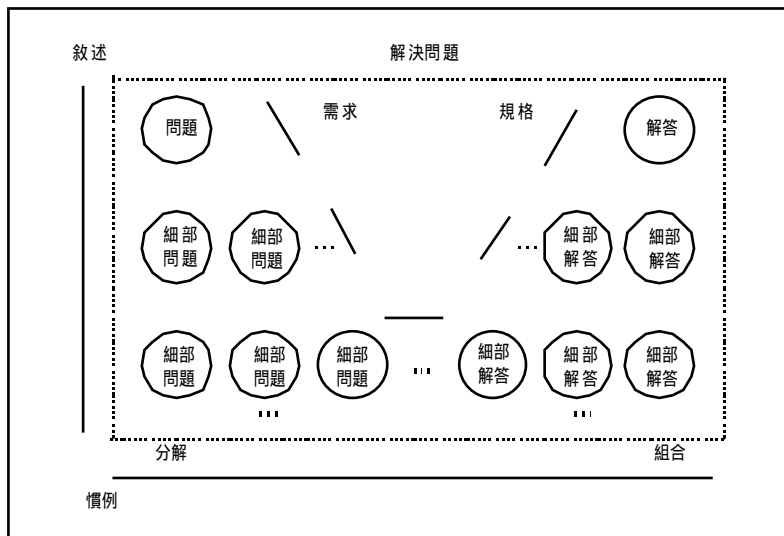


圖 2-2：問題解決的微觀

問題的解決（圖 2-2）牽涉到努力或程序的合作。像這樣的努力包括了：

- 將問題以某種可適應及溝通的型式，進行概念化及表示。讓整個問題能以某些語言表示，如此可與他人一起處理和溝通。
- 決定此問題之解決方案的內在需求，也就是說，此需求肇因於問題本身，而所獲得之解決方案必須滿足此需求。
- 藉由以下的詢問來描述此問題的特性：這個問題是什麼？這個問題有哪些需求是解決方案必須滿足的？在一個計劃中，如果這些問題能夠獲得最低限度的回答，也就獲得了關於此問題的敘述性觀點。
- 確認或說明滿足此問題之需求的解決方案。
- 藉由以下的詢問來描述此解決方案的特性：這個解決方案是什麼？這個解決方案有哪些規格符合問題的需求？在一個計劃中，如果這些問題能夠獲得最低限度的詳細回答，也就獲得了關於此解決方案的敘述性觀點。

---

如果無法確定某個解決方案是否能解決此問題，那麼可將此問題對解決方案的要求分類，並有系統的規劃為可同時解決的子問題。一個大的問題可不斷地分解成較小的子問題，子問題可以同時解決，而原始問題的解決方案，則可藉由子問題的解決方案組合而來。這個分析的過程可以持續不斷地進行，直到所有的子問題都找到答案；組合的過程也將持續不斷地進行，直到獲得原始問題的答案為止。因為這個過程是遞迴式的，因此敘述性的觀點將用以探尋規則性觀點。除此之外，利用分解可以解決許多即時及平行的問題，並讓整個過程進行的更加迅速。

解決問題的過程將繼續進行以下的事：

- 藉由回答以下的詢問，將可獲得解決此問題之解決方案的步驟規則：此解決方案如何滿足所有的要求（或子問題的要求）？在一個計劃中，如果這些問題能夠獲得最低限度的詳細回答，也就獲得了關於此解決方案的規則性觀點。
- 以某種具體且可用的格式，瞭解此解決方案所代表的意義。

為了使此過程變得更容易，知識是一個必須獲得、溝通（共享）及使用的重要元件。與整個工作有關的知識可分割成幾個群組，每個群組在某些時間與此工作的某些部份（概念化、問題、答案、解決問題、或達成目標）有關。這個關聯性表示這些知識元素何時會影響此工作；換句話說，在解決問題的過程中，何時要獲得、共用、及使用這些知識元素。

智慧是引導整個解決問題過程的能力，它將知識進行分類，並將這些知識應用於解決問題的過程以獲得一個解決方案。

## 生命週期（Life Cycle）

解決問題的策略必須有系統的安排，以提供一個管理的觀點及發展的觀點。這兩個觀點可供管理和執行整個工作。在第五章談到 UML 時，將會再討論生命週期。

生命週期（圖 2-3）可用下列方式分類。

- 生命週期是所有階段的集合，這些階段將整個工作分割為多個易於處理及控制的子工作。

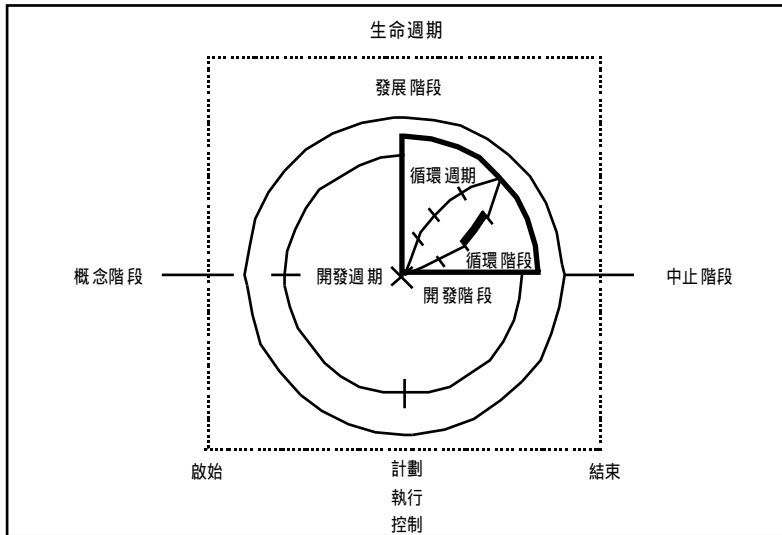


圖 2-3：生命週期

- 生命週期中的每個階段，都是解決一個問題的過程中獨立的部份。每個階段
  - 都有輸入或必須處理的資料項，這些都將在此階段中處理。
  - 都有啟始此階段的準則。
  - 需要原理技術和工具的應用程式以產生輸出，這解釋了某些工作是如何處理，以及誰負責進行這項處理。
  - 都有輸出或必須由此工作所產生的資料項。
  - 都有離開或結束此階段的準則。這將對工作的產品進行回顧，並（通常會）決定是否繼續或放棄整個工作。
- 概念的階段（phase）牽涉到瞭解一個問題或機會，並致力於解決此問題。
- 發展階段包含一個工作的啟始、計劃、執行、控制及結束等多重發展週期。
  - 啟始牽涉到瞭解：發展週期是處理一個新的或現存問題所必需的。

- 
- 計劃則是設計及維護一個設計方案，藉以達成解決問題的目的。
  - 執行則是在執行計劃和解決問題時，必須協調所擁有的資源。
  - 控制包括監控及測量進度，以便在必要時進行正確的評估。
  - 結束是將目前週期及週期開始時，問題解決方案的合理性概念化。

發展階段是由一個或多個開發週期（發展週期）所組成，其中每個開發週期都包含一個分配給此開發週期之成員的工作。因此它包括一個產品的發展過程。

#### ● 開發週期（Development Cycle）

- 由開發階段所組成。
- 將工作分配給開發階段。
- 在它的生命週期發展過程中，包含一個產品的發展過程。
- 第一次出現在回應一個問題時，並產生一個解決方案或產品。
- 此後將出現很多次，以改進一個現存的答案或產品設計。
- 觸發的原因是存在一個問題因而需要一個解決方案，或是改變了一個問題，而需要重新發展現存的解決方案。這些改變可能包括了對 stakeholder 的需求、基礎之技術、以及組織之競爭力的改變。
- 可能會稍微和前一個或後一個開發週期重疊。目前的開發週期開始時，可能會稍微和前一個開發週期的結束時期重疊；目前的開發週期結束時，可能會稍微和下一個開發週期的開始時期重疊。
- 產生一個產品的新世代。

#### ● 開發階段

- 由循環週期所組成。
- 將工作分配給循環週期（iteration cycle）。

- 
- 在它的開發週期之發展過程中，包含一個產品的發展階段。
  - 觸發的原因是一個新的開發循環的開始，或是在一個開發循環內前一個開發階段結束或即將結束。
  - 可能會稍微和前一個或後一個開發階段重疊。目前的開發階段開始時，可能會稍微和前一個開發階段的結束時期重疊；目前的開發階段結束時，可能會稍微和下一個開發階段的開始時期重疊。
  - 藉由某個重要事件將開發導向的元素和管理導向的元素同步化，所謂重要事件，就是在開發週期中的某個時間點，此時必須做關鍵性的決定及達成重要的目標。

#### ● 循環週期 ( Iteration Cycle )

- 由循環階段所組成。
- 將工作分配給循環階段。
- 在它的開發階段之發展過程中，包含一個產品的發展循環。
- 觸發的原因是一個新的開發階段的開始，或是在一個開發階段內前一個循環週期結束或即將結束。
- 可能會稍微和前一個或後一個循環週期重疊。目前的循環週期開始時，可能會稍微和前一個循環週期的結束時期重疊；目前的循環週期結束時，可能會稍微和下一個循環週期的開始時期重疊。
- 製造一個新版的產品。這是完整產品世代的子集合。

#### ● 循環階段 ( Iteration Phase )

- 由活動或職務所組成。
- 將工作分配給活動或職務。
- 在它的循環週期之發展過程中，包含一個產品的發展進度。
- 觸發的原因是一個新的循環週期的開始，或是在一個循環週期內前一個循環階段結束或即將結束。

- 可能會稍微和前一個或後一個循環階段重疊。目前的循環階段開始時，可能會稍微和前一個循環階段的結束時期重疊；目前的循環階段結束時，可能會稍微和下一個循環階段的開始時期重疊。
- 中止階段（Cessation Phase）則是因為瞭解已不再需要此解決方案，因此依序結束此解決方案的所有應用。
- 在生命週期中出現了許多並時性（concurrency）和平行性（parallelism），因此將工作或計劃分配給開發週期後，整個程序就能非常迅速地完成。

## 開發週期和階段

開發週期和階段（圖 2-4）擁有的特性：

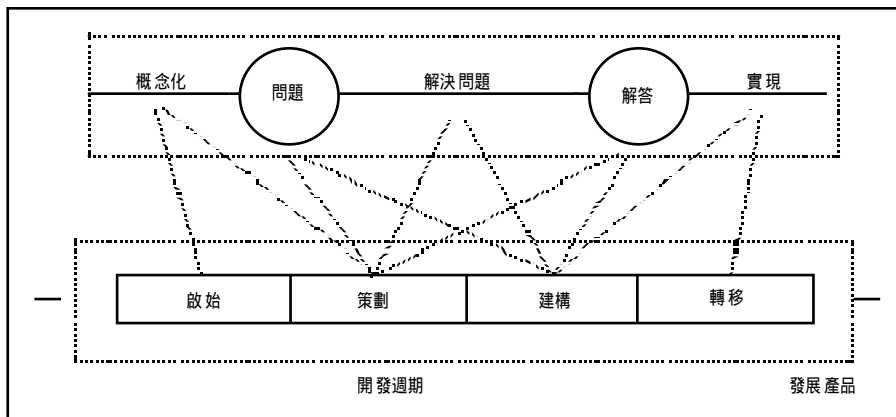
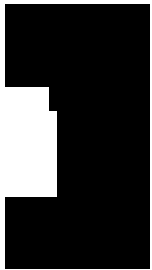


圖 2-4：開發週期中的各階段之重點

- 每個開發週期由四個階段（啟始、策劃、建構、轉移）所組成，並產生一個產品世代。由於它對計劃的進展提供一個高階的指標，所以這種對於生命週期的解釋常稱為管理觀點。
- 啟始（inception, initiation）開發階段通常使用一個循環週期，並進行以下的各種活動：

- 
- 設定範圍、目標和需求的界限。
  - 建立一個商業的案例和基本原則，包括說明未來的理想、詳列成功的條件、評估危險性、估計所耗用的時間及成本、以及詳述一個計劃。
  - 專注於瞭解或組織此問題的表示式，以及解決它（範圍和商業案例）的理論基礎。
- 策劃（elaboration, planning）開發階段通常使用一個循環週期，並進行以下的各種活動：
- 藉由前一個階段詳細製作工作的明細表。
  - 完成範圍、目標、和需求之界限及底限的設定。
  - 藉由維繫遠景、鞏固成功的標準、減少最高的危險性，來設定商業案例的底限。利用預定的時程設定計劃的底限。
  - 在建構階段中，將需求分配到多個循環週期。
  - 專注於瞭解或組織此問題的表示式，以決定此問題要求其解決方案所要具備的需求（需求的獲取的及需求的分析），建立及確認整體解決方案的基礎（結構設計），並將需求分散於此建構開發階段（計畫）的循環週期中。
- 建構（construction）（執行）開發階段通常使用一個以上的循環週期，並進行以下的各種活動：
- 藉由控制目前的風險及維護計畫和時程來更新商業案例。
  - 藉由執行多個循環週期，建立及發展此項工作的產品。
  - 專注於下列事項：瞭解或設計此問題要求其解決方案所要具備的需求，使得可以詳細說明此問題及解決方案、詳述此解決方案的規格（分析）、更新所有解決方案的基礎（結構設計）、針對循環週期的要求而更新為支援某個特定解決方案所需的基礎（細部設計），針對循環週期的要求而確認其解決方案（證明及整合）、以及提供或整合（或交付）此解決方案（deployment）或子集合（subset）。



- 轉移 (transition) (結束, closing) 開發階段通常使用一個循環週期，並進行以下的各種活動：

- 將最後的產品或解決方案分散佈署至問題中。
- 對過渡時期的產品進行評估和分類。
- 主要關注的焦點，在於提供及整合或交付此解決方案。

## 整合週期和階段

整合週期和階段 (圖 2-5) 擁有的特性：

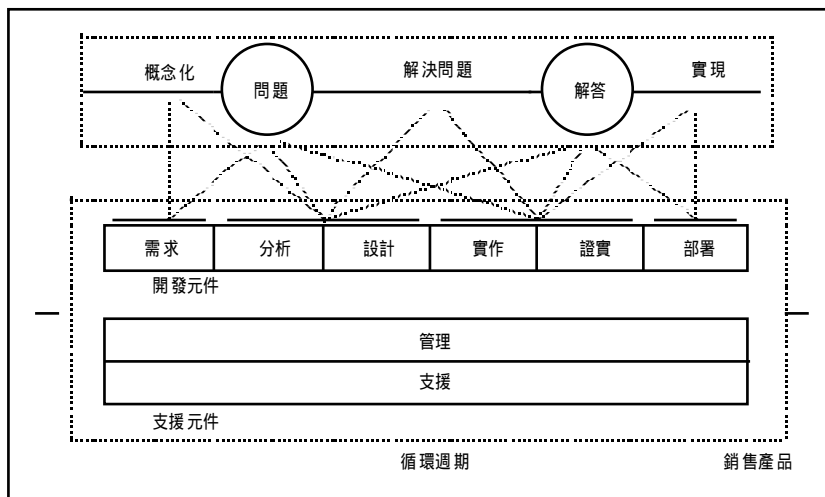


圖 2-5：一個循環週期內的循環階段

- 每個循環週期由兩個支援元件 (管理、支援) 和六個開發元件 (需求、分析、設計、實作、證實、出售) 所組成。最後會產生一個版本的產品。由於它對計劃的進展提供一個低階的指標，所以此種生命週期通常稱為開發觀點 (development perspective)。
- 管理 (控制) 階段牽涉到對整個循環週期的管理。

- 
- 支援（控制）階段牽涉到在此循環週期中，對 stakeholder 參與者之需要的滿足；這包括開發環境的支援、設定、標準定義、及指導方向。
  - 在計劃發展階段中，將進行以下的循環階段：
    - 需求（需求獲取）和分析（需求分析）階段包括瞭解或組織此問題的表示式，以決定解決方案的需求，並將需求分散到建構發展階段的循環週期（計劃）。
    - 設計（結構設計）階段包含建立及確認整個解決方案的基礎。
  - 在建構（執行）發展階段中，將進行以下的循環階段：
    - 需求階段牽涉到瞭解及說明問題對其解決方案的要求，以詳述問題及解決方案。
    - 分析階段包括對解決方案規格的詳細說明。這個階段的關鍵在於知識的應用。
    - 設計階段則包含更新所有解決方案的基礎（結構設計），以及針對循環週期的要求而更新為支援某個特定解決方案所需的基礎（細部設計）。這個階段的關鍵在於知識的應用。
    - 實踐階段則是為符合循環週期的要求而產生解決方案。
    - 確認（及整合）階段是為符合循環週期的要求而確認其解決方案
    - 出售階段則為提供及整合或交付此解決方案或其子集合。

## 問題和答案

問題和答案（圖 2-6）是一種具有因果關係的情況（領域或空間）。為了解決問題（系統），必須先瞭解問題；為了建構及使用此問題的答案（系統），也必須先瞭解這個答案。為了簡化實踐的過程並遵守各種的限制，此解決方案必須經過系統化的安排（結構）；為了解決問題，必須獲取關於此問題及解決方案的正確知識（模型）、將有關問題及解決方案的決定系統化（結構性的觀點）、以及使用某種語言（此語言必須能在解決問題的過程中，提供溝通的能力）繪製圖形。

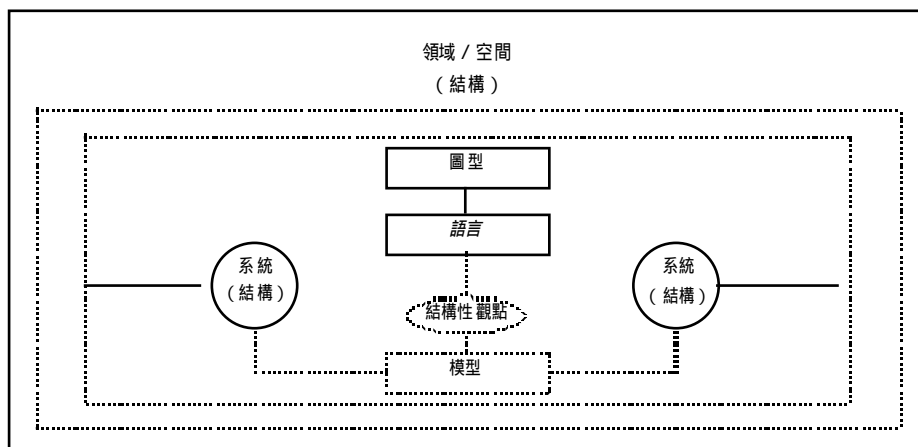


圖 2-6：問題和答案

## 領域或空間 ( Domain or Space )

領域或空間 (圖 2-7) 是在各自獨立的狀態及有興趣的範圍中，相關元件有系統化的集合。這些元件稱為領域元素或空間元素 (elements)：

- 它們也稱為內容 (contexts)。
- 由一組專用術語組成特定的詞彙，藉以簡化此領域內元件的溝通。
- 由一組概念組成關於此領域內元件的原則或表示式，並為此領域內的元件溝通提供一個目標。
- 它們對於本身領域的範圍有一個明確的界限，每個在此界限內的元件即處於此種狀態，也就是內部範圍；而在此界限之外的每個元件，即處於外部範圍或其他的狀態。
- 它們沒有什麼目的。也就是說，這些構成一個領域的元件並未經過有系統的安排，藉以達成特定的目標，而是不經意或因為其它理由或基準而結合在一起。一般而言，一個領域的元件會被集合在一起，通常是因為它們與此領域內相同的元件有所關聯。

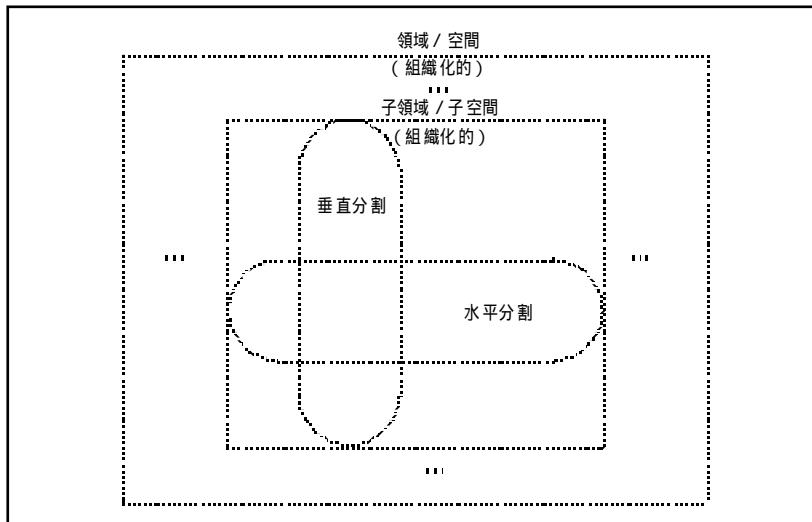


圖 2-7：領域或空間

- 可以表示真實或抽象的狀態。
- 可以不斷地分解而成為很多個次領域 (subordinate domains)。這些次領域可以視為此領域的主題中有興趣的題目。
- 可以垂直的方式分析、分割、審視，或以垂直的方式劃分這些元件，當作多個關注的領域。
- 可以水平的方式分析、分割、審視，或將水平跨越多個領域的方式劃分這些元件。這些分割而成的部份將能更深入地顯示詳細資訊。
- 可以是許多知識本體的主題，即使是由相同的領域所衍生而來的，這些知識的本題仍可能有不同的內容。
- 表示問題及解決方案存在及被解決之情況與結構。

UML 與所處的領域無關。它可以使用於領域之內，以簡化問題解決的過程；也可以跨領域使用，讓知識能重覆使用。在開發資訊系統時，領域所代表的就是問題之存在與解決方案被實踐的前後關係。

## 系統

系統（圖 2-8）就是有交互作用及相關聯的元件之系統化集合，這些元件會合作以達成一個目標。這些元件稱為系統元素。系統：

- 是擁有一個目標、目的或責任的狀態。如果系統的目標無法利用或被忽略，那麼可將系統視為一種狀態；若在狀態中加入了一個目的，則可將狀態視為一個系統。狀態和系統基本上是相同的，只有當以其目的為基礎而進行操作時才會產生差異。

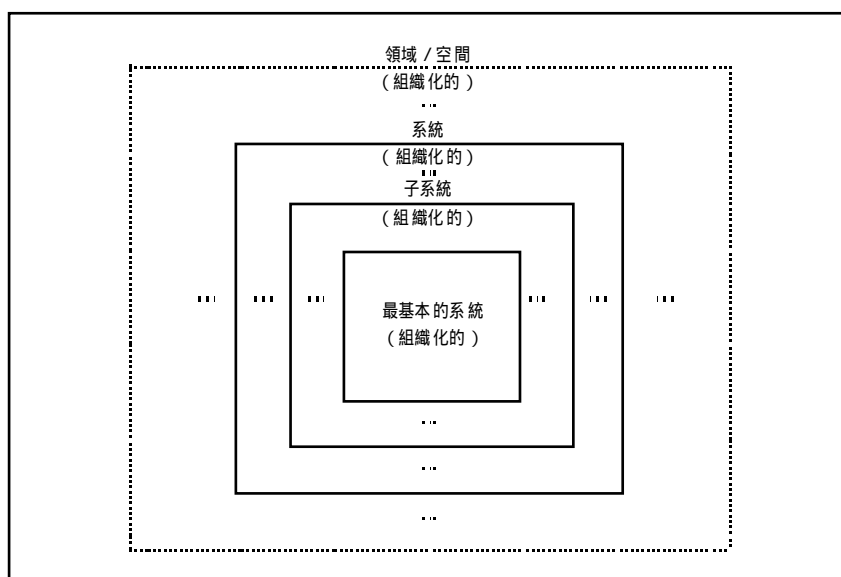


圖 2-8：系統

- 必須擁有一個或一組目標。也就是說，它們經過有系統的規劃以完成一組目標，而此規劃是以其目標的特質為基礎。
- 可以表示實際或抽象的目標實體 (purposeful entity)。
- 可以不斷地分解而成為很多個次系統 (或子系統)。
- 當完全分解之後，會產生最基本的系統元素，這些元素無法再被進一步分解。

- 可以是許多知識本體的主題，即使是針對相同的系統所衍生而來的，這些知識的本體仍可能有不同的內容。
- 表示在某種狀態下，問題及解決方案存在及被解決之情況。

UML 與所用的系統無關。它可用於處理各種不同的系統，包括硬體系統、軟體系統、以及與使用者交談的系統。在開發資訊系統時，系統所代表的就是解決問題後所獲得的答案。

## 架構 (Architecture)

架構 (圖 2-9) 是關於系統的結構和行為之設計。結構：

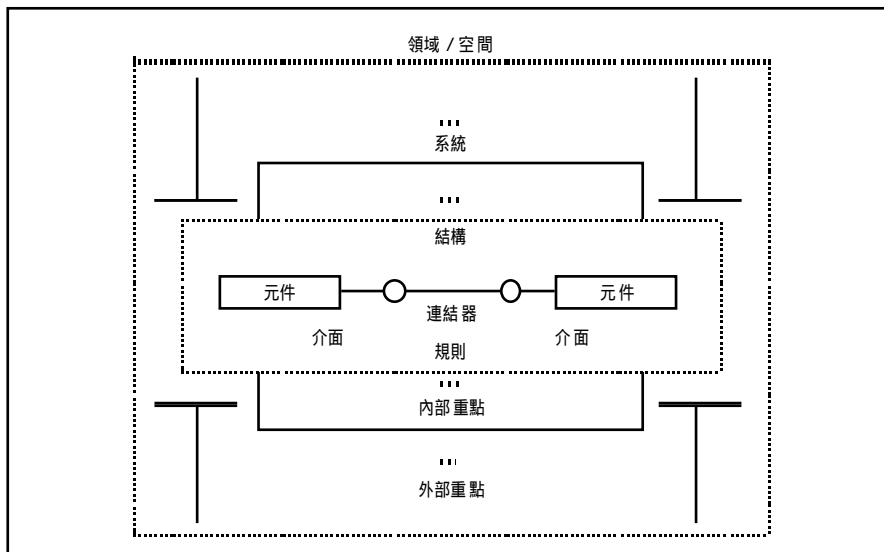


圖 2-9：結構

- 著重於系統的內部。
- 著重於狀態的外部。
- 包括元件及其介面，這些屬於系統和子系統。這些元件稱為結構元素 (architectural elements)。

- 包括經由介面整合及連接其它元件的連結器 ( connector ) , 這些連結器也稱為結構元素。
- 包括管理元件和連結器之組合的規則及限制。
- 包括管理元件和連結器之間互動的規則及限制。
- 為了滿足某些目標, 牽涉到經由連結器的元件連接之規劃。
- 可不斷地分解而成為許多個結構, 這些結構稱為次結構 ( subordinate architectures )。每個元件和連結器也可以不斷地分解而成為較小的元件和連結器。
- 可將元件的垂直部分當作子結構的重點部份, 對其進行分析、分割、審視。
- 可將元件的水平部分當作跨越子結構的重點部份, 對其進行分析、分割、審視。
- 當完全分解之後, 會產生基本的系統元素和基本的連結器, 這些元素和連結器無法再進一步的分解。基本的系統元素和連結器就相當於最基本的系統元素。
- 可歸類為形式 ( form ) 或現成的樣版 ( pattern ), 用以解決重覆出現的結構化問題。這些樣版能保留可重覆使用的重要知識。
- 可歸類為參考一群形式或樣版的結構化型態。這些型態對結構的選擇加上了某種程度的統一化限制。
- 包含問題和答案的組織系統。它們非常適用於控制大量提高的複雜性。
- 當問題、答案及狀態存在及被處理時, 結構表示其內部的組織架構。

UML 是以結構為核心的。它支援領域內的系統組織。組織化的機制可在系統發生改變時, 處理並試著降低其所帶來的複雜度。在開發資訊系統時, 相較於沒有組織化的系統而言, 有良好架構的系統在面對系統的改變, 耗用較少的時間以及較少的成本。

## 模型 ( Model )

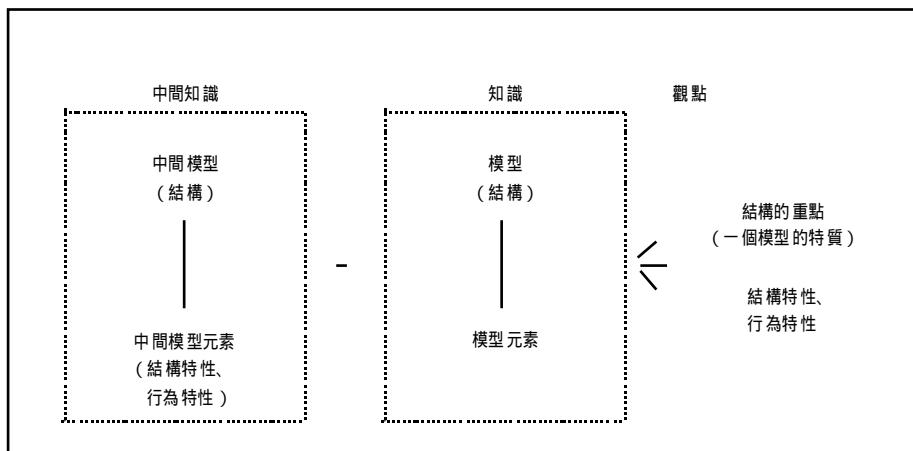


圖 2-10：模型

模型 (圖 2-10) 是系統或狀態之完整抽象概念。

- 模型是完整的，或說其語意是封閉性的；也就是說，對一個模型的內容進行理解或解釋時，並不需要引用此模型之外的知識。
- 模型是抽象的構造，用以針對某個給定的主題表示系統或狀態。
- 藉由抽象化技巧，以專注於系統中相關聯的細節，並忽略不重要的部分。這對系統和觀察者的角度定義了一個界限。
- 模型是系統的藍圖，藉以進行系統的建構及更新修補。
- 用以瞭解及控制系統內部的複雜度。
- 用以溝通及確認系統的可靠性。
- 可不斷地分解而成為許多個次模型 (subordinate models)。
- 當完全分解之後，會產生基本的模型元素，這些元素無法再進一步的分解。

- 
- 模型擁有基本成份的元素。模型元素是系統元素或狀態元素的完整抽象化概念。它們：
    - 是由模式元素所定義。中間模型是定義模式元素（中間模型元素）的模型，而模式元素可用於衍生的模型之中。模型是包括模型元素的中間模型之實踐，模型元素則是模式元素之實踐。
    - 可保存系統元素的特性或特質。
    - 可保存系統元素的結構化特性，也就是系統元素的靜態特質。
    - 可保存系統元素的行為特性，也就是系統元素的動態特質，以及系統元素間的互動或合作。
    - 可能與此模型中其它的元素具有關聯性。
  - 可經由一組整體、但幾乎獨立且不重疊的觀點進行審視。觀點是模式方面的因素，強調一個模型某個特定的性質。以下的觀點存在於任何一個模型：
    - 結構化的模型觀點強調模型元素的靜態（或結構性）特質。
    - 行為的模型觀點強調模型元素的動態（或行為性）特質，或是這些模型元素在模型內的合作關係。
  - 可以不同程度的精確性來表達。
  - 一個模型可能是參考模型，亦即描述其模型元素所有可能設定的模型。結構是參考模型的子集合。實踐是實作某個特定結構的產品。
  - 當將模型視為系統，而此系統的目的是為獲取其它系統之完整抽象化概念，則此模型擁有一個結構。
  - 獲取一個系統或狀態的知識。
  - 當問題、答案及狀態存在及處理時，模型可用以表示其知識。

UML 提供獲取及運用知識的能力，以簡化問題的解決過程。知識無法清楚的檢閱，但可以用實例解說。在知識的應用中，知識的實例出現的目的，是為了衍生出一個問題的答案，這就是所謂的智慧。

## 結構性觀點

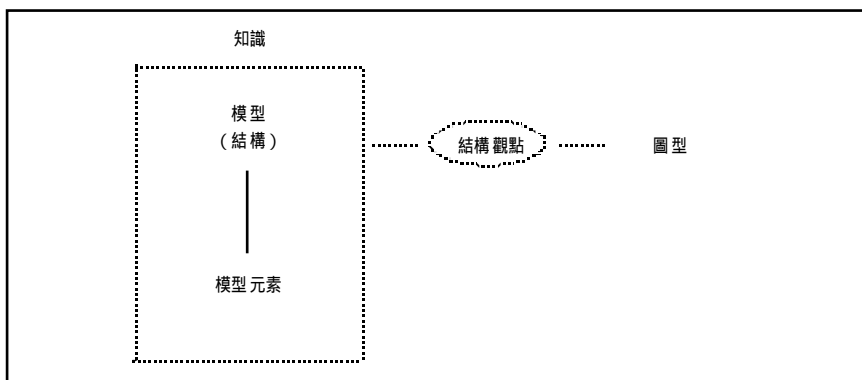


圖 2-11：結構性觀點

結構性觀點（圖 2-11）是模型的抽象化概念。

- 藉由結構性觀點，可將模型表示或設計成圖形。
- 對參與解決問題的工作的各個 stakeholder，對特定幾組概念規劃模型的觀點。一組概念建立一個結構化的焦點。
- 藉由不同的觀點，提供一個模型中重要的結構化元素之萃取能力。一個觀點中不重要的結構化元素將被忽略。對於處理結構化焦點的模型，此模型的一組概念將會建立一個觀點。

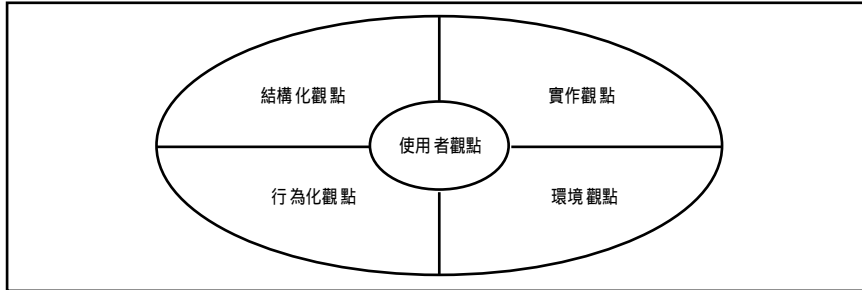


圖 2-12：模型的觀點

- 對於問題和答案的模型，包含以下的模型觀點（圖 2-12）：
  - 使用者模型觀點包含一個問題和解決方案，其來自於處理問題及答案者的觀念。這個觀點表達了問題所有者的目標及對答案的要求。
  - 結構化模型觀點包含一個問題和解決方案的靜態（或結構化）觀點。
  - 行為化模型觀點包含一個問題和解決方案的動態（或行為化）觀點，以及問題和解決方案元素之間的互動或合作。
  - 實踐模型觀點包含解決方案實現之結果的結構化和行為化觀點。
  - 環境模型觀點包含一個領域的結構化和行為化觀點，而解決方案必須在此領域中求得。
- 可擴充以含括其它模型的觀點（例如介面、安全性、資料）。
- 時常參考到模型，因為它們最終會對應到整個模型的一個子集合。
- 將模型對應成各種圖形的類型。
- 以關於知識和主題的決定為核心，將一個模型內的知識組織化，這些決定將會進一步的改變知識和未來的決定。

UML 提供使用者模型觀點、結構化模型觀點、行為化模型觀點、實踐模型觀點、及環境模型觀點，以簡化問題解決過程之知識組織。

## 圖解 ( Diagrams )

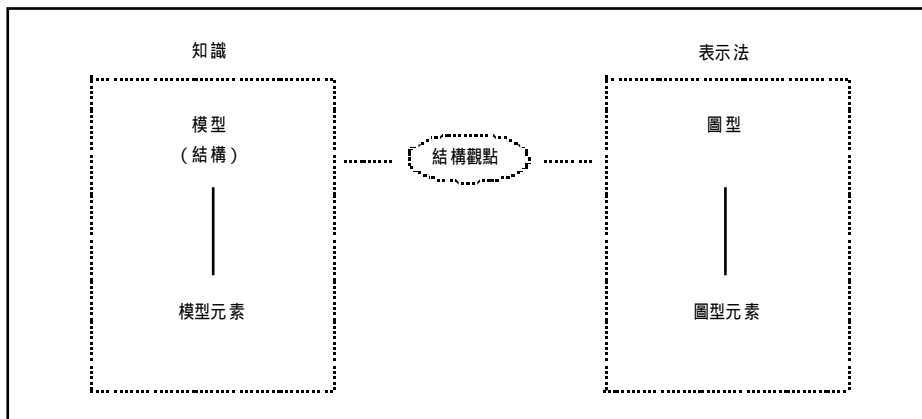


圖 2-13 : 圖解

圖解 (圖 2-13) 是模型元素的圖形化投影。

- 圖解是由弧線 (或路徑) 和頂點 (或節點) 的連接圖。
- 以圖解的方式, 描繪一個模型的結構化觀點或其子集合。
- 圖解是有視覺化關係之圖形化結構的結合。
- 圖解是用以描述模型的實體結構。
- 是藉由結構化觀點所敘述之模型的藍圖。
- 擁有審視元素, 其為模型元素之集合的圖形化投影。
- 用來傳遞溝通模型的內容或知識。
- 以一種可傳遞的格式, 描繪一個模型內的知識。
- 圖解也可稱為模型, 因為它們能描繪模型、或是作為模型的視覺化表示法。

UML 提供傳遞溝通知識的能力, 藉由提供許多環繞結構化觀點的圖形, 簡化問題解決的過程。

## 語言

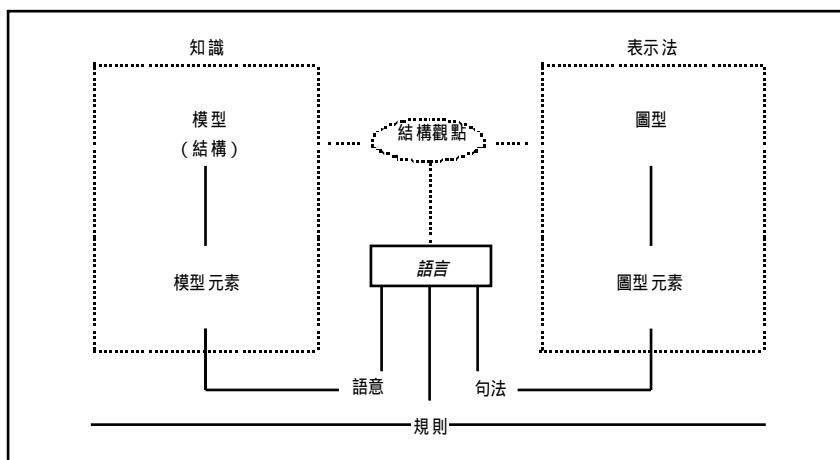


圖 2-14：語言

語言（圖 2-14）是用來表達及溝通內容或資訊的方法。語言：

- 擁有藉以溝通的概念和語意。在 UML 中，模型元素定義了基本的模式概念及語法，這些概念及語法建立溝通的方法或內容。
- 擁有用來表示溝通所用之語意的語法或表示式。語法元素表示語意元素。在 UML 中，模型元素的表示式或視覺化表示法，提供了語法，並為溝通的語意建立了表示式。
- 擁有表示慣用法的規則，以及如何結合語法結構，來組成溝通之內容的規則。UML 獲得了各式使用與最佳實踐的規則和慣用法。
- 將一個或多個模型元素對應到一個或多個圖形元素。
- 像 UML 這樣的語言，可以控制如何描繪一個模型內的知識，以便溝通這些知識。

## 解決問題

解決問題（圖 2-15）必須先以瞭解為前題對其進行審視，並以實現為前題審視其解決方案。解決問題的過程包含了在一連串（可能是同時進行）的程序（活動）中，利用知識以獲得此問題的答案（產品）。在這些程序中，會應用知識並使用各種不同的工具。除此之外，也可能使用由其它問題解決過程所獲得的知識及規則。

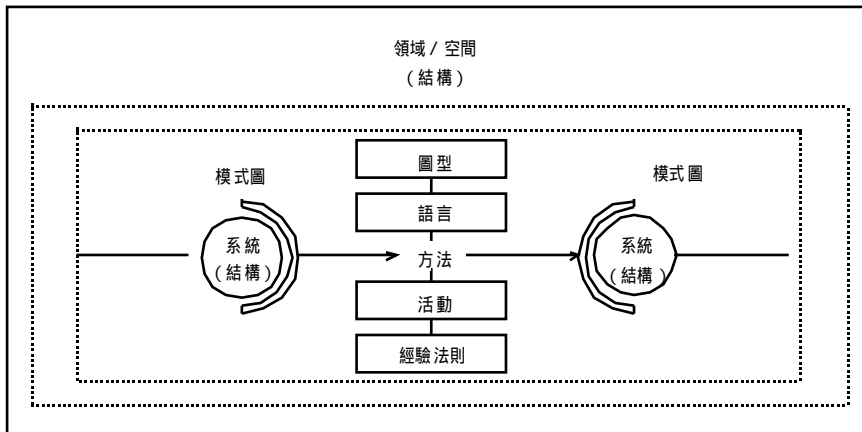


圖 2-15：解決問題

## 模式圖 ( Paradigm )

模式圖是有系統地安排、各自獨立的相關元件集合，而這些元件形成模型的基礎。這些元件稱為模式圖元素 ( paradigm element )。

- 模式圖定義一組構成特定的術語。
- 模式圖定義一組構成與主題有關之原則的概念。
- 藉由定義模式元素，建立中間模型的基礎。模式圖定義了可用於模型中的專門術語及概念。
- 模式圖構成了語言和方法的基礎。UML 的基礎建立在物件導向的模式圖之上，而且可用於不同的方法。

- 
- 建立概念及相關的專門術語，它們在模式圖中是基礎的、而且被普遍地接受為真理。這些概念建立了一個基礎，在此之上其它的概念才得以建立，而系統也藉此而建立模型、進行審視（察覺、瞭解、解釋）及操作。
  - 當問題、答案、狀態存在和被處理時，模式圖可表達一個方法如何開始著手、審視以及建立模型。

UML 的基礎建立於物件導向的模式圖之上。因此 UML 能擁有足夠的彈性，讓跨越各種不同系統的領域之方法都能使用它。物件導向的模式圖也對變化及複雜度的掌控提供了不少幫助。

## 製品 (Artifact)

製品是工作的產品或是由工作所得可交付的成果。製品：

- 由方法所記述或敘述。
- 在程序中實踐或建構。
- 必須有一些 stakeholder 負責製造它們，其它人則負責接受（或拒絕）它們。
- 以語言（例如 UML）來表達，此表示式提供了完成此問題之解決方案的能力。
- 如果在相同的狀態，但在不同的系統或是跨越不同的狀態和系統時，這些製品會被重新使用，它們將成為資源。
- 代表過渡時期及最終的產品（解決方案），這些產品由是一個方法所指定，並由一個程序所實踐。

UML 的圖示是簡化問題解決過程的產品。它們有系統地安排及描述要瞭解問題之答案所需的知識。

---

## 活動 ( Activity )

活動是由製造或發展製品所主導的工作之集合。活動：

- 是部分或暫時由方法所掌控。
- 在程序中實踐。
- 必須有 stakeholder 參與，這些 stakeholder 負責指揮或執行工作。
- 必須有由活動所開發（建立、修改等等）的產品。
- 必須處理關於一件工作的技術性及管理方面的問題。
- 必須視為一組建議，而且必須為其所應用的狀態重新設定。
- 為了達到目的而建立或建議工作及使用的技術。工作所指的是思考、執行、及回顧整個過程，以符合目標。技術則提供了執行工作的意義。它們皆以原理為基礎，並有工具支援。
- 活動代表的是在計劃內的子工作，由一個方法所指定，並由一個程序來達成。

因為 UML 並未規定某一個方法或程序，因此它也並沒有指定某個特定的活動。

## 經驗法則 ( Heuristic )

經驗法則是由經驗得來的規則。經驗法則：

- 可將方法的應用簡化為程序。
- 包含最佳的實踐以及由經驗所學到的知識。
- 代表由其它經驗所獲得的知識，以及由以前的方法之應用所獲得的知識。

UML 包含了工業界最佳的問題解決策略。這個策略是由整個 UML 的系統，以及它所提供的各種圖解所提供的。

