國立屏東大學 資訊工程學系 物件導向軟體工程

3. UML簡介

在開始本課程前，讓我們先總結一下 <note tip> 使用物件導向技術所開發的軟體，在執行時會建構出一個虛擬的物件導向世界，其中包含有對應於真實或虛擬世界中的人、事、時、地、物的抽象物件，透過物件與物件、物件與使用者的互動，來滿足使用者的需求 </note>

UML的全名為Unified Modeling Language (UML)可譯做統一模型語言，或一致性模型語言，是在物件導向軟體工程領域的一個標準的通用性(general purpose)模型語言UML是由Object Management Group(簡稱OMG)制定，目前最新版本為2.5.1[1)]UML就是由一組圖形表示法(graphic notation)其模型必須依據相關的圖示標記與規範來加以繪製，因此UML的模型又被稱為是視覺化模型(visual model)還要注意的是UML本身不含有任何建構模型的方法論，它只是視覺化的塑模語言(visual modeling language)UP才是方法論。

<note> General purpose是指不針對特定應用領域的意思，除譯做通用性以外，也常譯為一般用途 </note>

3.1 UML Diagrams

UML包含有許多種圖，請參考figure 1，本課程將分別加以簡介。要注意的是，在開發軟體時並不是每一種UML圖都是需要的，例如Scott W. Ambler曾說過：「沒見過有人用過複合結構圖(Composite structure diagram)交互概述圖(Interaction overview diagram) 或者通信圖(communication diagram)你可以視情況使用部份圖即可。

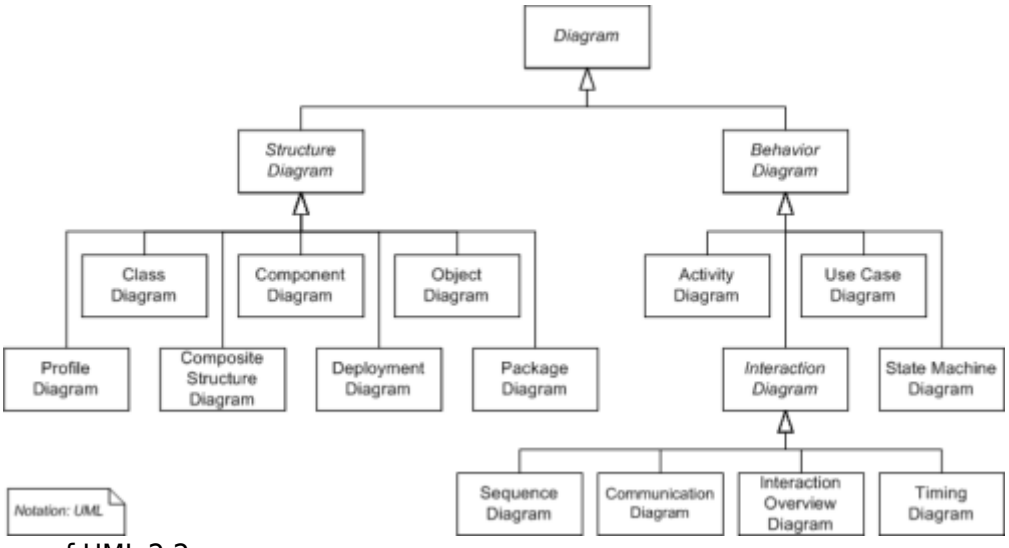

Fig. 1: Diagrams of UML 2.2

figure 1顯示UML的圖可分成兩大類：結構圖組(structure diagrams)與行為圖組(behavior diagrams)其中結構圖組是用以表示軟體靜態的結構，而行為圖組則是用以描述軟體動態的行為。在本課程後續將針對UML的各種圖進行簡介，同學先不需要在此階段就開始學習各種圖細節，只需要大致瞭解各種圖的作用即可。我們將在本課程的實務演練中，陸續針對如何繪製相關的UML圖加以詳細說明。

3.1.1 結構圖組(Structure Diagrams)

結構圖組(structure diagrams)所要描繪的是那些在開始進行軟體塑模時就要呈現的事，主要是軟體系統的結構，包含類別圖、元件圖、組合結構圖、部署圖、物件圖、套件圖與剖面圖。

3.1.1.1 類別圖(Class Diagram)

回顧本課程一開始時的總結：「使用物件導向技術所開發的軟體，在執行時會建構出一個虛擬的物件導向世界，其中包含有對應於真實或虛擬世界中的人、事、時、地、物的抽象物件，透過物件與物件、物件與使用者的互動，來滿足使用者的需求。」。從程式設計師的角度來看，在「虛擬的物件導向世界」裡的物件並不是憑空誕生的，我們必須使用new才能在記憶體來產生物件(實體)，而且在能夠new出來物件之前，還必須先要有「類別(class)」的存在 – 因為new是將特定類別的物件產生出來，例如：

```
Student amy = new Student();
```

在上述的程式敘述裡，使用new產生了一個Student類別的物件 – 當然，前提是我們必須要先有Studnet類別的定義。因此，在使用物件導向技術來開發軟體前，必須先決定在「虛擬的物件導向世界」裡需要存在哪些「類別」的物件，並且要將這些「類別」先加以定義。而UML的類別圖(class diagram)存在的目的，就是要進行這件工作 – 定義「虛擬物件導向世界」所需要的物件是屬於哪些類別，以及它們的屬性、操作方法與彼此間的關係。舉例來說，一個銷售系統軟體的類別圖可參考figure 2，由客戶、訂單、付款等類別所組成，我們應該要在開始進行軟體實作前(甚至是設計前)就先完成類別圖的繪製；或是至少先完成一部份，後續在開發的過程中，再進行所需的調整(增加新的類別或修改、刪除既有的類別)。

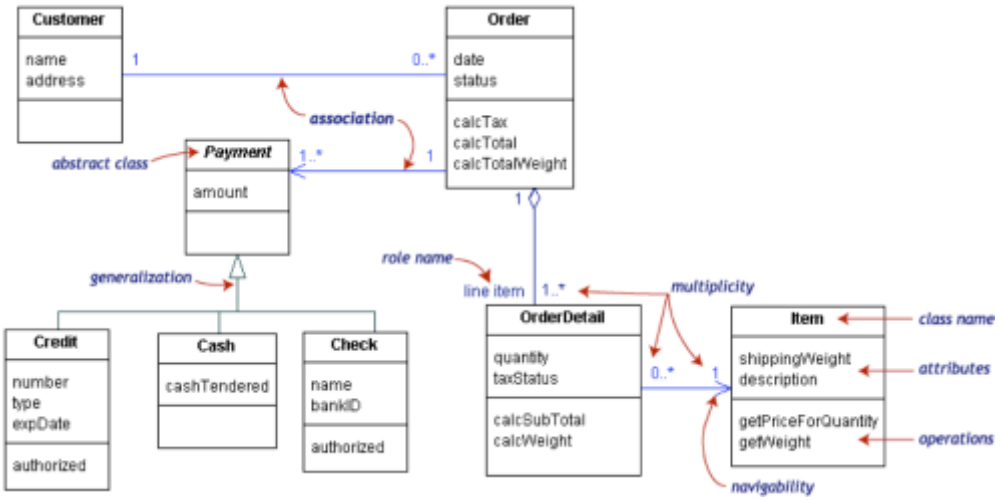

Fig. 2: An example class diagram

3.1.1.2 元件圖(Component Diagram)

元件圖(component diagram)描述的是軟體系統可以被分割成哪些元件(或是反過來說，是由哪些元件組成的軟體)，以及這些元件彼此間的關係。figure 3顯示的是一個網購平台軟體的元件圖，包含了瀏覽器端的介面、購物車及資料庫等元件所組成。

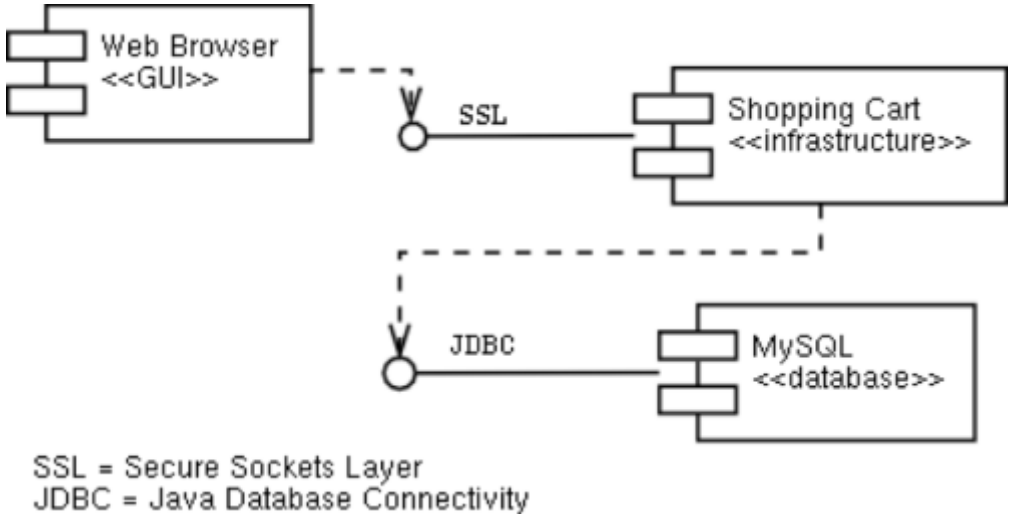

Fig. 3: An example component diagram

3.1.1.3 組合結構圖(Composite Structure Diagram)

組合結構圖(composite structure diagram)是用以描述一個類別的內部結構，例如說明它是如合與其它類別合作以完成工作。例如figure 4顯示了一個BrokeredSale類別(代理銷售)的內部，是從批發商類別的物件進行購買的操作，然後再對零售商類別的物件進行銷售的操作。

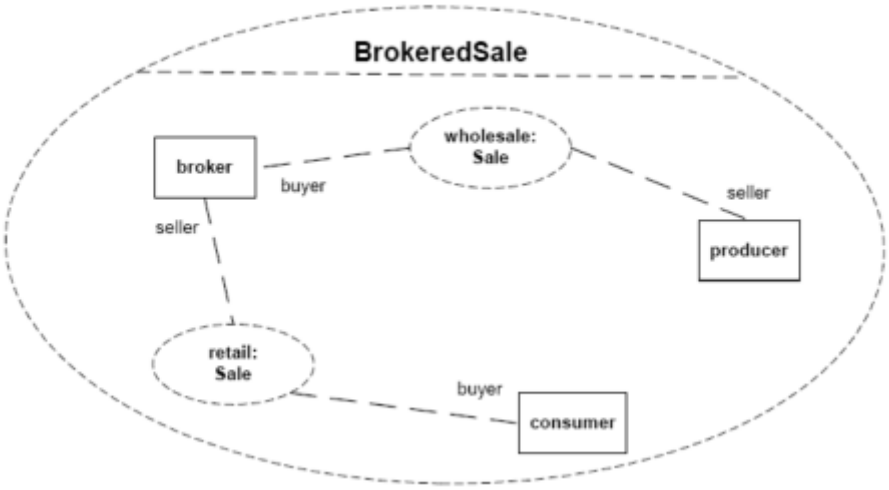

Fig. 4: an example composite structure diagram

3.1.1.4 部署圖(Deployment Diagram)

部署圖(Deployment Diagram又常被稱為配置圖)是用以顯示軟體所運行的硬體環境，更簡單的說法則是顯示軟體要在什麼硬體環境執行。請回顧一下figure 3網購系統的元件圖，figure 5的部署圖顯示了它所要運行的硬體環境，包含客戶端的電腦Web伺服器以及資料庫伺服器。

Fig. 5: an example deployment diagram

要注意的是，我們也可以將兩張圖合併起來，請參考figure 6，我們可以直接將元件圖的內容繪製在特定的硬體環境裡。

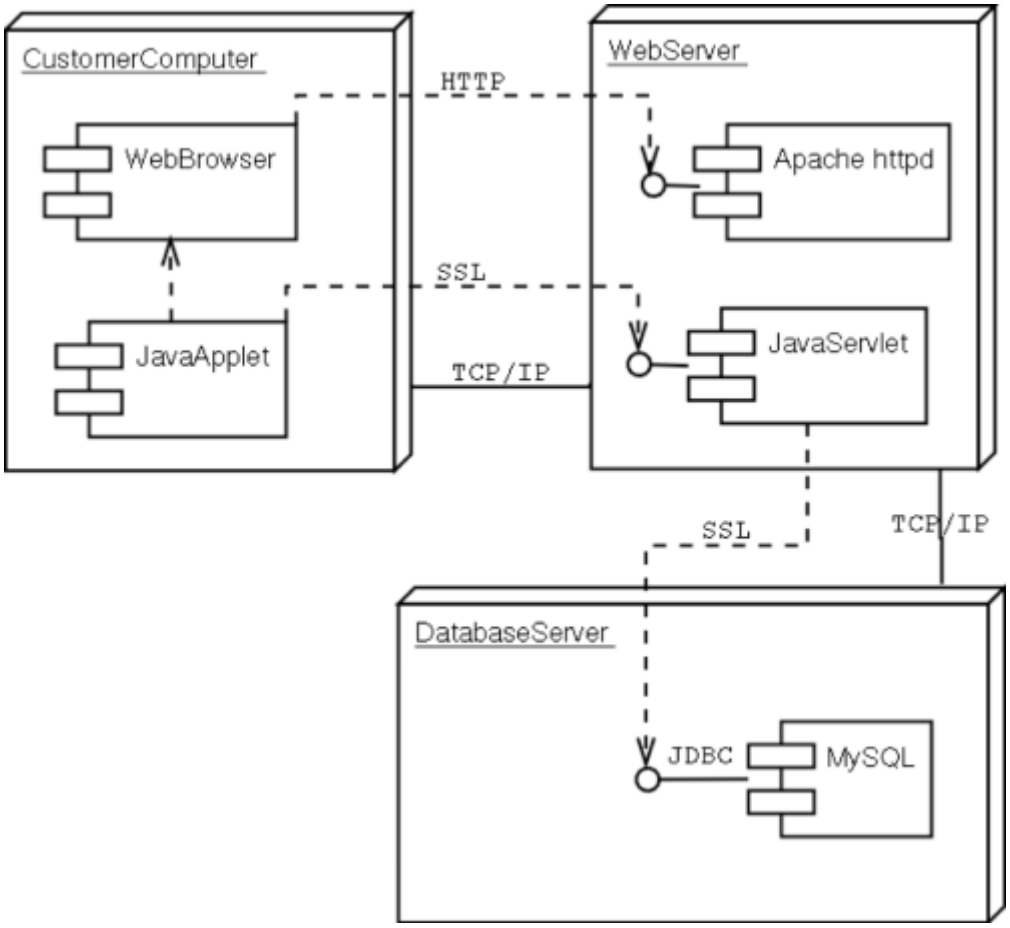

Fig. 6: an example deployment diagram with component diagrams

3.1.1.5 物件圖(Object Diagram)

物件圖(object diagram)顯示在某個特定時刻(不一定是時間，通常是指符合特定條件、特定情境的時刻)，系統內完整或部份的物件內容。更詳細來說，物件圖描述了在特定時刻時(或情境下)，系統內存在哪些物件？且其當下的屬性值或狀態為何？有助於讓使用者或開發人員，更容易、也更清處地瞭解系統某些功能運作的細節。請參考figure 7，它顯示了在系統執行的某個時刻，在「虛擬的物件導向世界」裡有一個名為c的物件，它是Company類別的物件，圖中還顯示了它擁有兩個Department類別的物件，名為d1與d2□d1物件還擁有一個同樣是Department類別的物件，名為d3□d3的經理以及其聯絡方式，則分別是Person類別與contactinformation類別的物件。

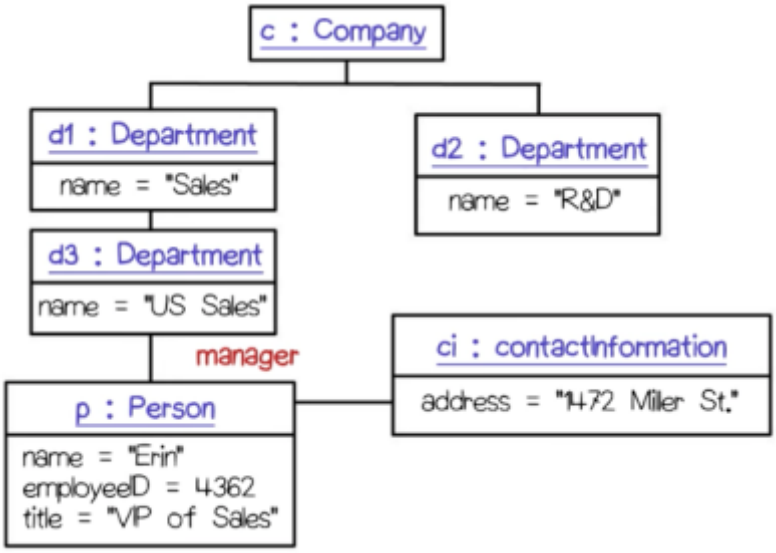

Fig. 7: an example object diagram

3.1.1.6 套件圖(Package Diagram)

套件圖(package diagram)將系統分為群以及子群，並且顯示它們間的關係，請參考figure 8的例子。

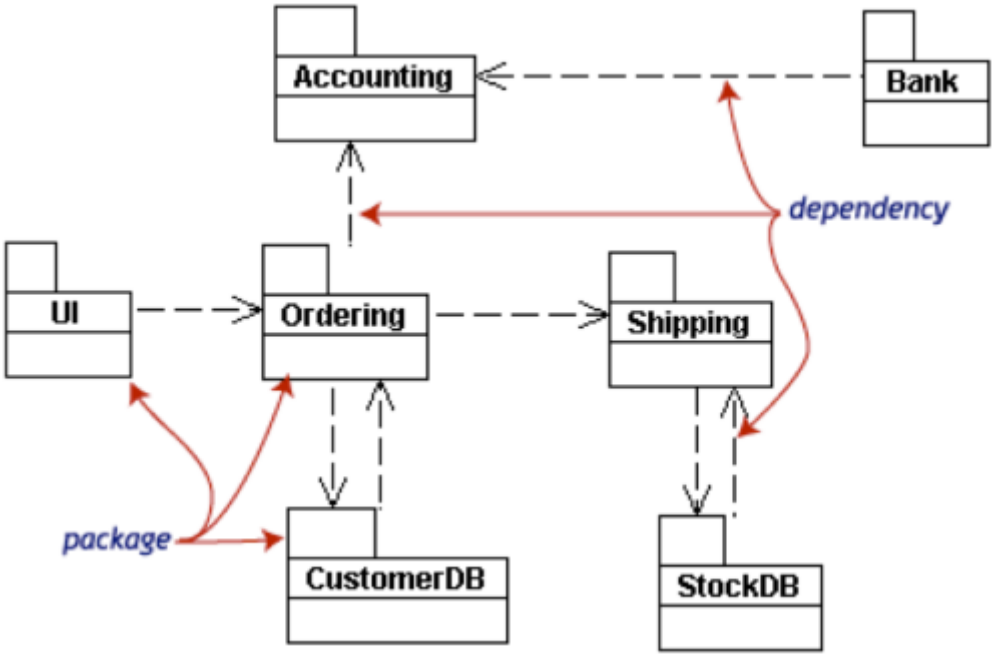

Fig. 8: an example package diagram

3.1.1.7 剖面圖(Profile Diagram)

剖面圖(profile diagram)是一種可擴充的機制，讓我們可以為特定的應用領域客製化UML模型，換句話說是用以制定模型的模型。請參考figure 9的例子。

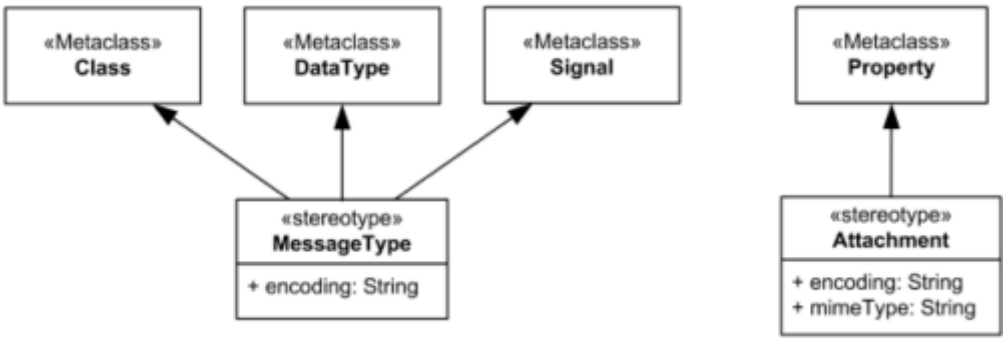

Fig. 9: an example profile diagram

3.1.2 行為圖組(Behavior Diagrams)

行為圖組(behavior diagrams)是描述系統在執行時的行為表現，通常用以呈現系統的各個功能。相關的圖包含活動圖、狀態圖、使用案例圖、通訊圖、互動概述圖、循序圖、與時序圖。

3.1.2.1 活動圖(Activity Diagram)

活動圖(activity diagram)是用以描述系統功能的具體流程。還記得我們說過「使用物件導向技術所開發的軟體，在執行時會建構出一個虛擬的物件導向世界，其中包含有對應於真實或虛擬世界中的人、事、時、地、物的抽象物件，透過物件與物件、物件與使用者的互動，來滿足使用者的需求。」，此處的透過物件與物件、物件與使用者的互動，來滿足使用者的需求就是要呈現在活動圖裡的內容。請參考figure 10，一個客戶透過ATM提款的功能，是透過系統中的Customer、ATM與Bank的互動來完成的，請參考figure 10的活動圖。

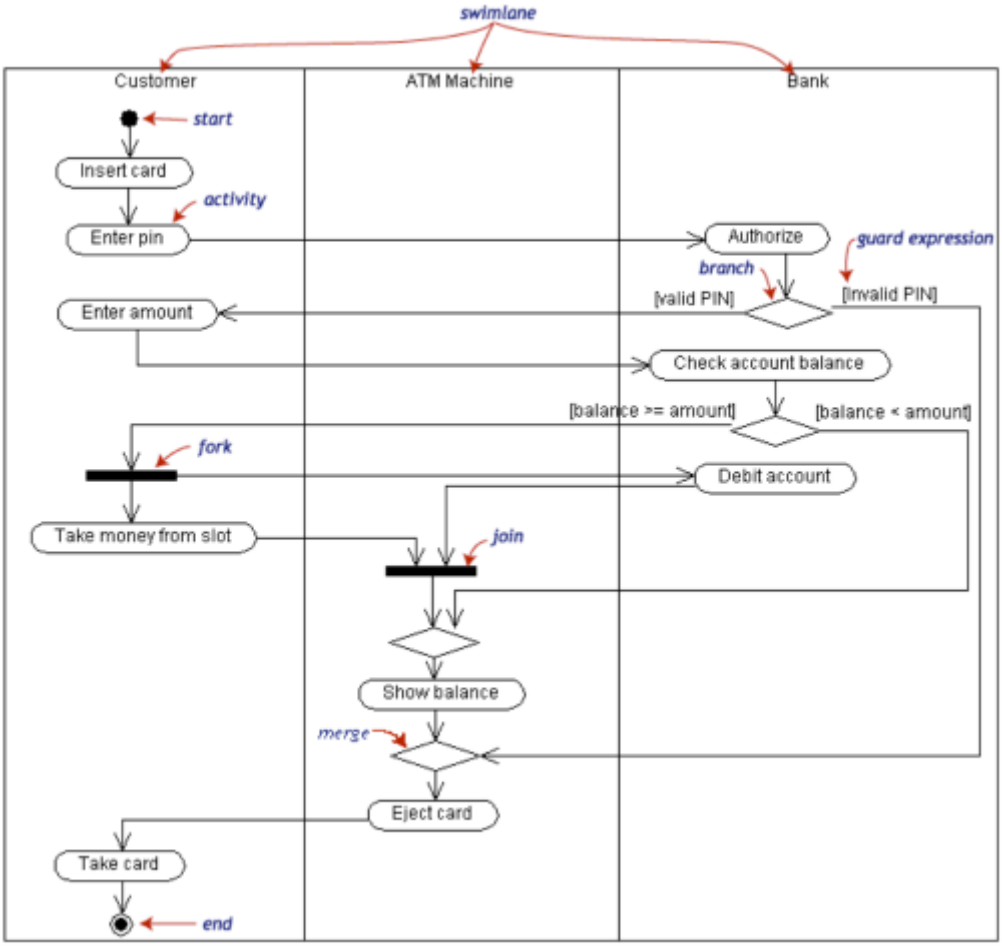

Fig. 10: an example activity diagram

3.1.2.2 狀態圖(State Machine Diagram)

狀態圖(state machine diagram又常稱為statechart diagram)在「虛擬的物件導向世界」裡的物件，從其誕生(也就是被new出來後)開始，到其不再被使用或是刪除掉(也就是被free或delete掉)為止，其所具有的狀態以及狀態間的轉移皆定義於狀態圖裡，請參考figure 11

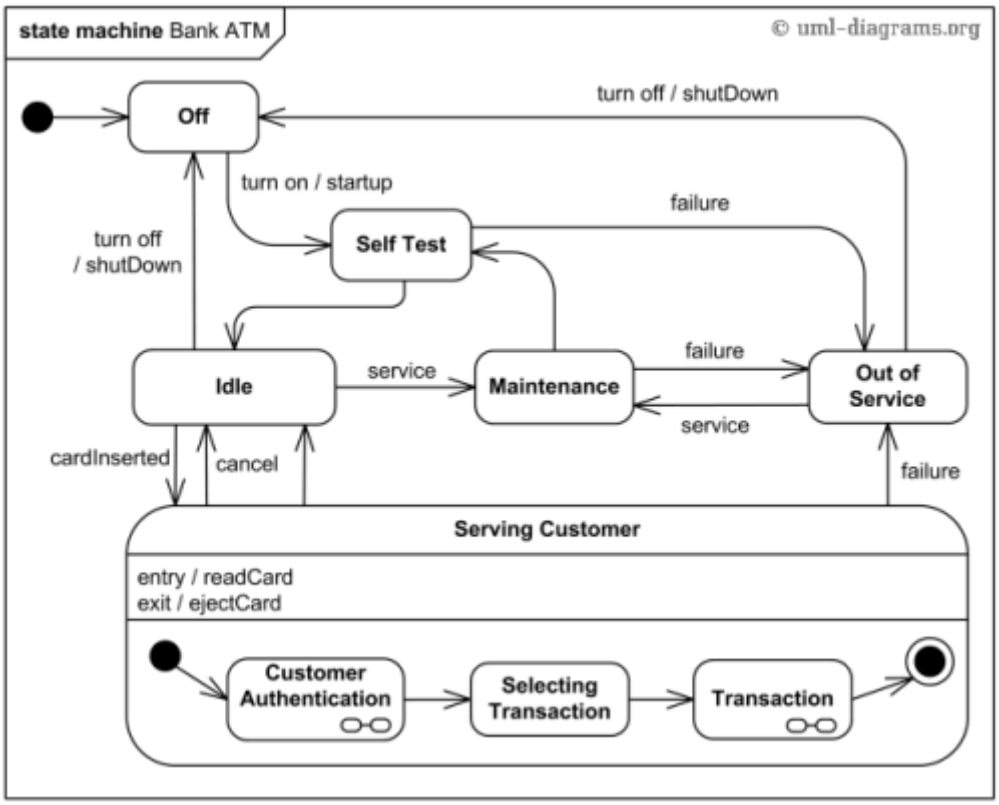

Fig. 11: an example state chart diagram

3.1.2.3 使用案例圖(Use Case Diagram)

使用案例圖(use case diagram)透過系統的使用者的觀點，將系統所要提供的功能加以呈現。具體來說，每個系統所應提供的功能都被視為是使用者會使用到的功能，我們將它們表達為所謂的使用案例(use case) – 使用案例圖所呈現的是使用案例以及它們間的關係。

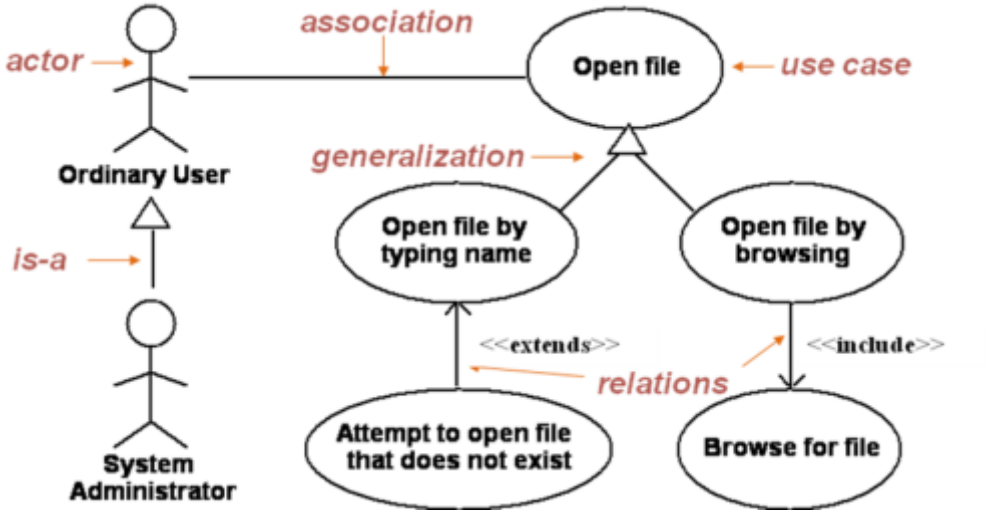

Fig. 12: an example use case diagram

3.1.2.4 互動圖組(Interaction diagrams)

此圖組包含多種圖，用以表示系統中的功能相關的控制流程與資料流。常見的圖包含通訊圖、互動概述圖、循序圖與時序圖。

3.1.2.4.1 通訊圖(Communication Diagram)

此圖透過在物件間的訊息溝通來表示功能。

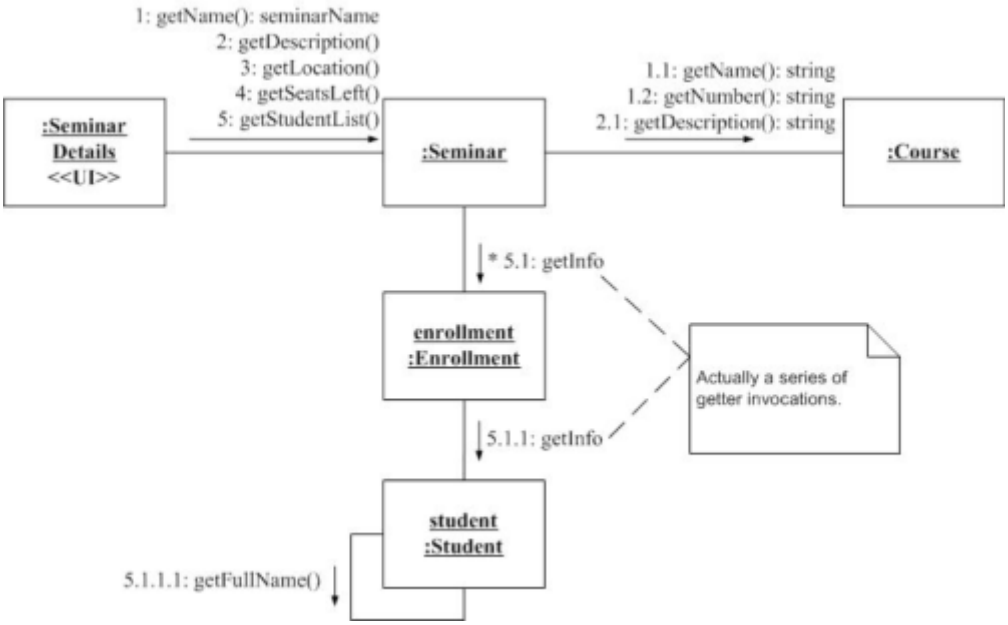

Fig. 13: an example communication diagram

3.1.2.4.2 互動概述圖(Interaction overview diagram)

簡單來說，此種圖將活動圖視為基礎結點，進一步描述各個節點間的通訊，也就是繪製活動圖間的訊息溝通。

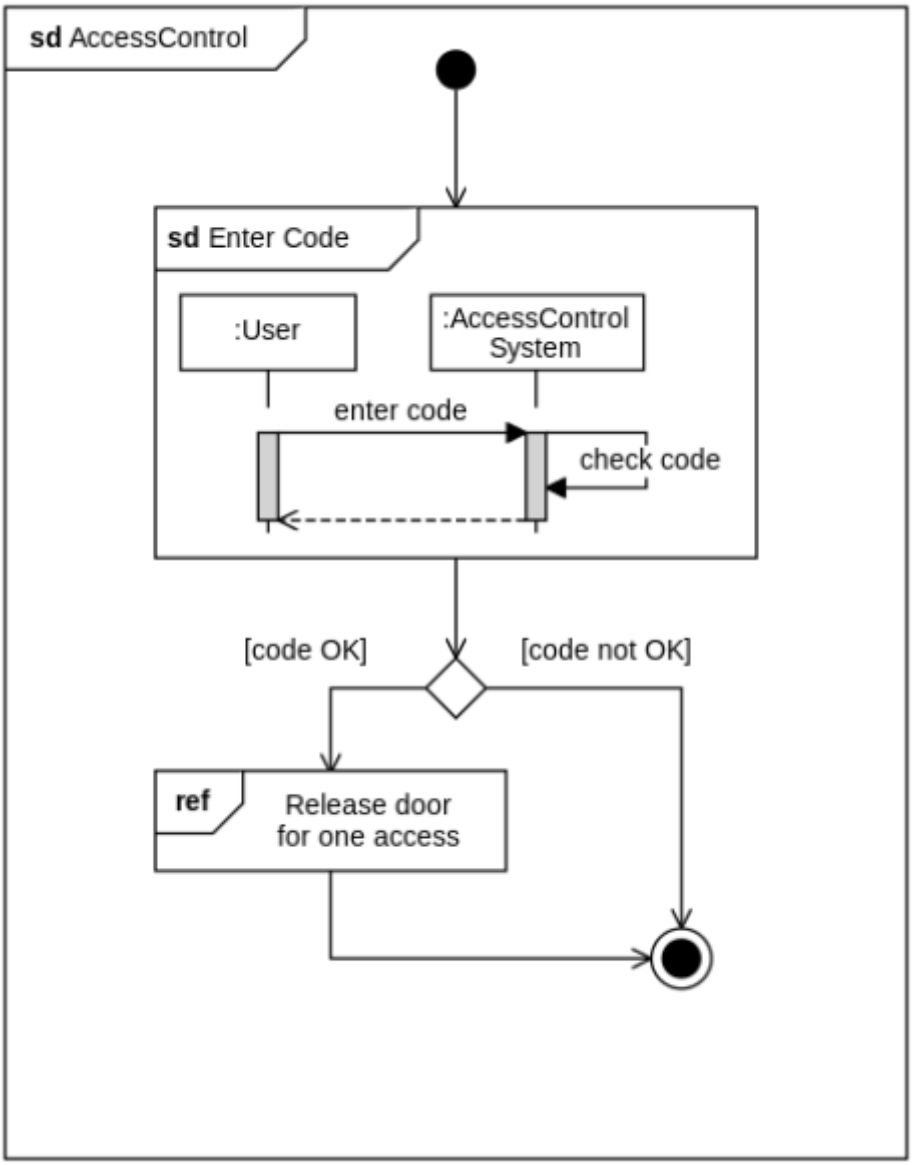

Fig. 14: an example interaction overview diagram

3.1.2.4.3 循序圖(Sequence Diagram)

循序圖(sequence diagram)顯示物件與物件或是物件與使用者間如何透過訊息溝通完成特定的功能，其中也顯示了每個物件在訊息溝通過程中的作用區間。另外，也有人使用等價的合作圖(collaboration diagram)做為替代。

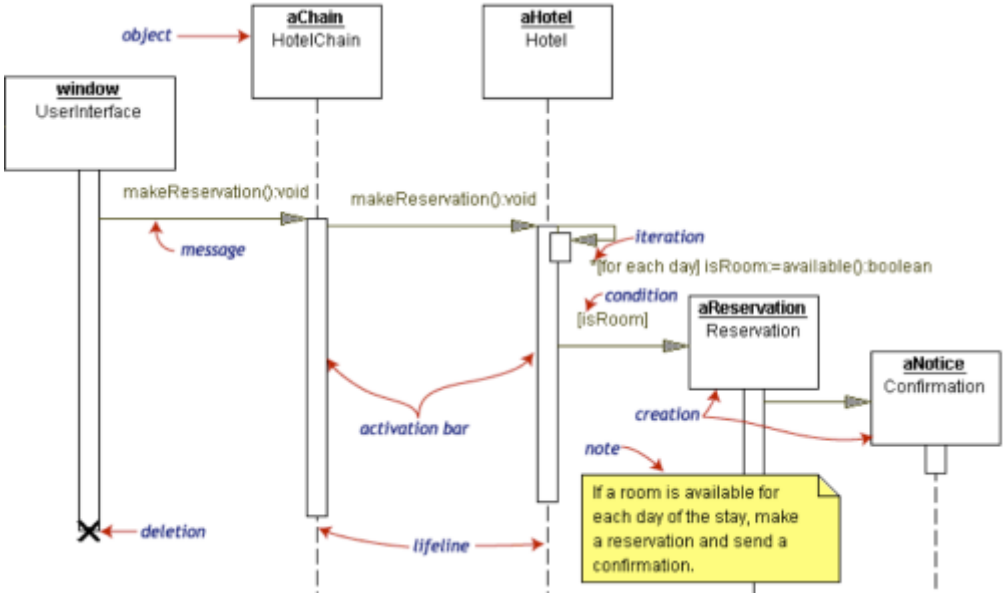

Fig. 15: an example sequence diagram

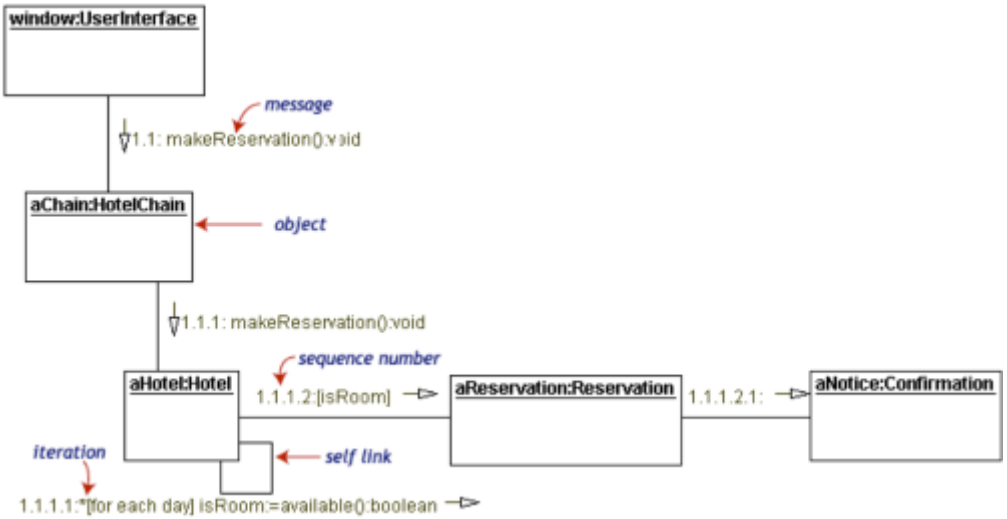

Fig. 16: an example collaboration diagram

3.1.2.4.4 時序圖(Timing Diagram)

此圖特別強調的是物件互動的時間性限制。

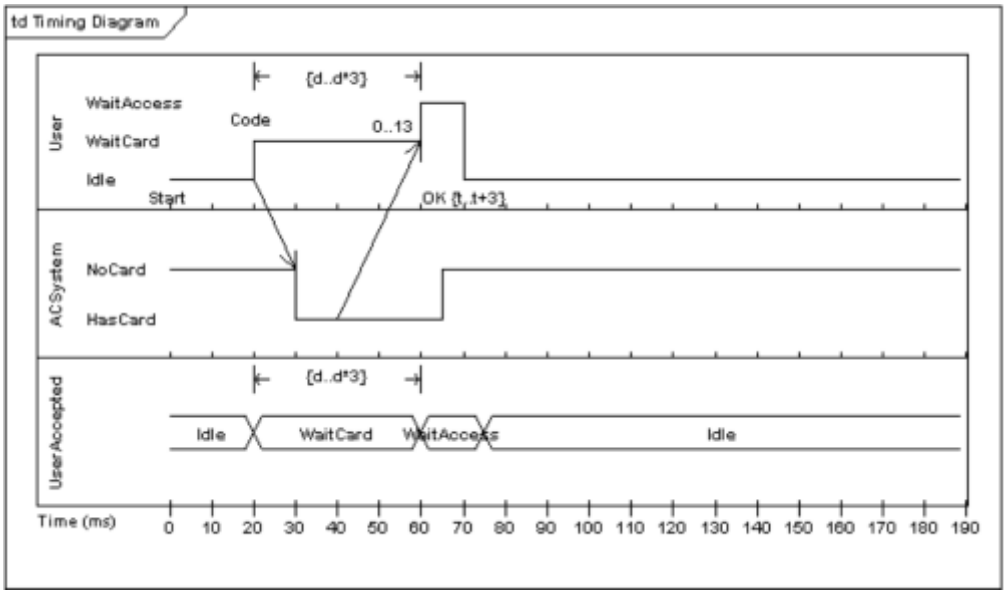

Fig. 17: an example timing diagram

3.2 4+1 觀點模型(4+1 Views Model)

由於一個軟體系統通常會讓人覺得過於複雜，因此透過4+1(也就是5個觀點)將軟體分割為從　不同的軟體專案參與人員的角度出發的5個部份就是4+1觀點模型的概念。此模型方法確保你必須考慮從5個重要的觀點來思考系統，請參考figure 18所顯示的4+1觀點：

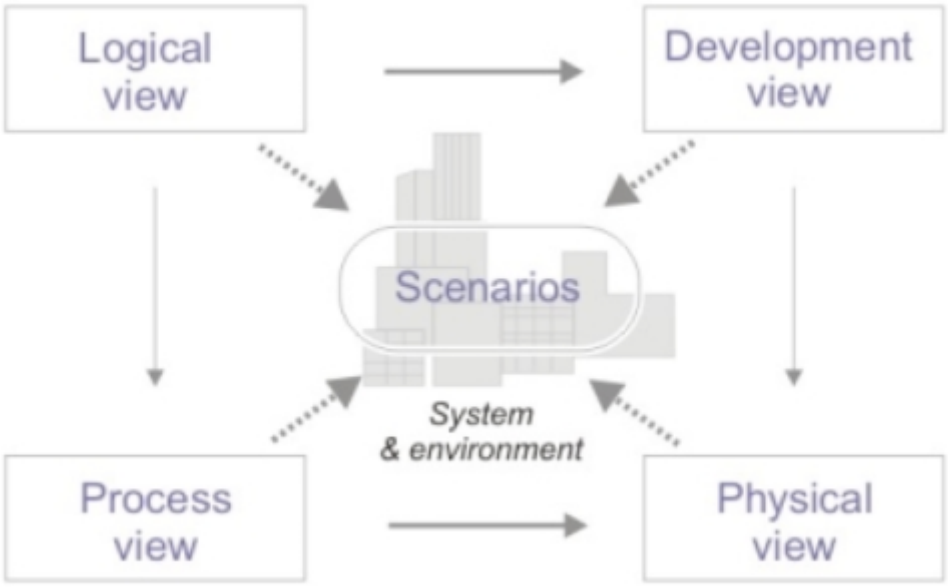

Fig. 18: 4+1 Views of UML

此外，4+1觀點還可以搭配UML做為模型來進行軟體的開發，以下我們分別說明這些不同的觀點：

3.2.1 邏輯觀點(Logical View)

邏輯觀點(logical view)主要是從系統應該提供什麼樣的功能給使用者的角度，來考慮的是系統應由哪些元

件、類別與物件來組成，白話一點來說，也就是考量在「虛擬的物件導向世界」裡該具有哪些東西? 常見相關的UML圖包含：

- 類別圖(class diagram)
- 物件圖(object diagram)
- 狀態圖(state machine diagram)

3.2.2 程序觀點(Process View)

程序觀點(process view)考量的是前述邏輯觀點所考慮到的系統物件相關的操作程序，包含它們相互之間的互動溝通等流程與規範。白話一點來說，也就是考量在「虛擬的物件導向世界」裡該發生些什麼事？程序觀點偏重的是動態的系統觀點，關注的是系統內的處理程序、功能，以及執行階段的行為表現。常用的相關UML圖包含：

- 活動圖(activity diagram)
- 循訊圖(sequence diagram)

3.2.3 實體觀點(Physical View/Deployment view)

實體觀點(physical view)是從系統開發者的角度來考量的是系統運行的軟硬體環境，常用的UML圖包含：

- 部署圖(Deployment Diagram)

3.2.4 開發觀點(Development View/Implementation view)

開發觀點(development view)是從程式設計師的角度來考量的是系統開發的相關議題，包含如何管理大量的類別、元件、執行的環境、所使用的類別庫等議題。常用的UML圖包含：

- 元件圖(component diagram)
- 套件圖(package diagram)

3.2.5 +1觀點 使用案例觀點(use cases view)

使用案例觀點(use cases view)顧名思義就是從使用者的角度出發，考慮系統應該提供什麼樣的功能給使用者。常用的UML模型為：

- 使用案例圖(use case diagram)

1)
Object Management Group, Unified Modeling Language, version 2.5r.1, December 2017, http://www.omg.org/spec/UML/Current