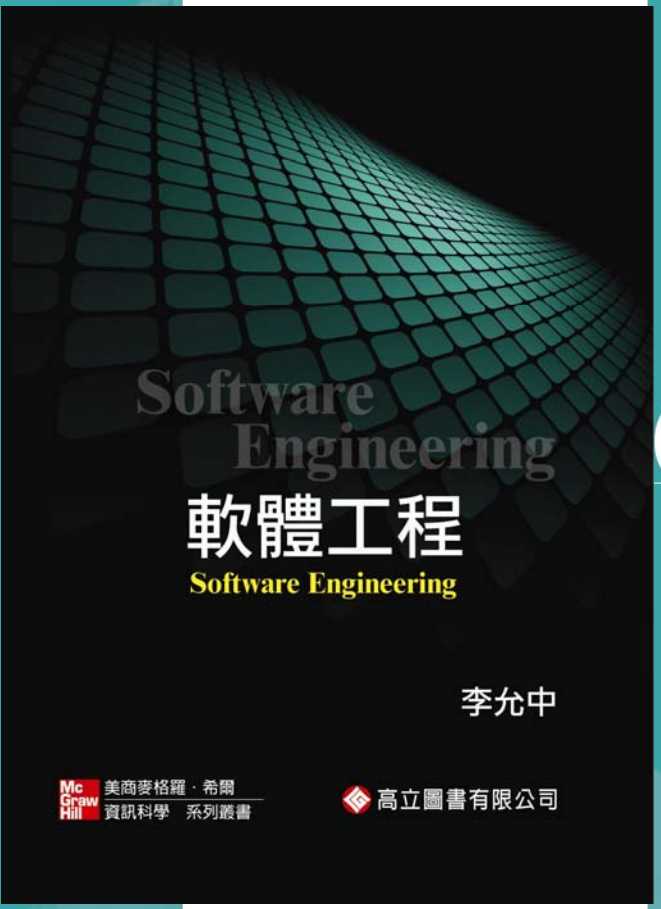


CHAPTER 4
軟體設計

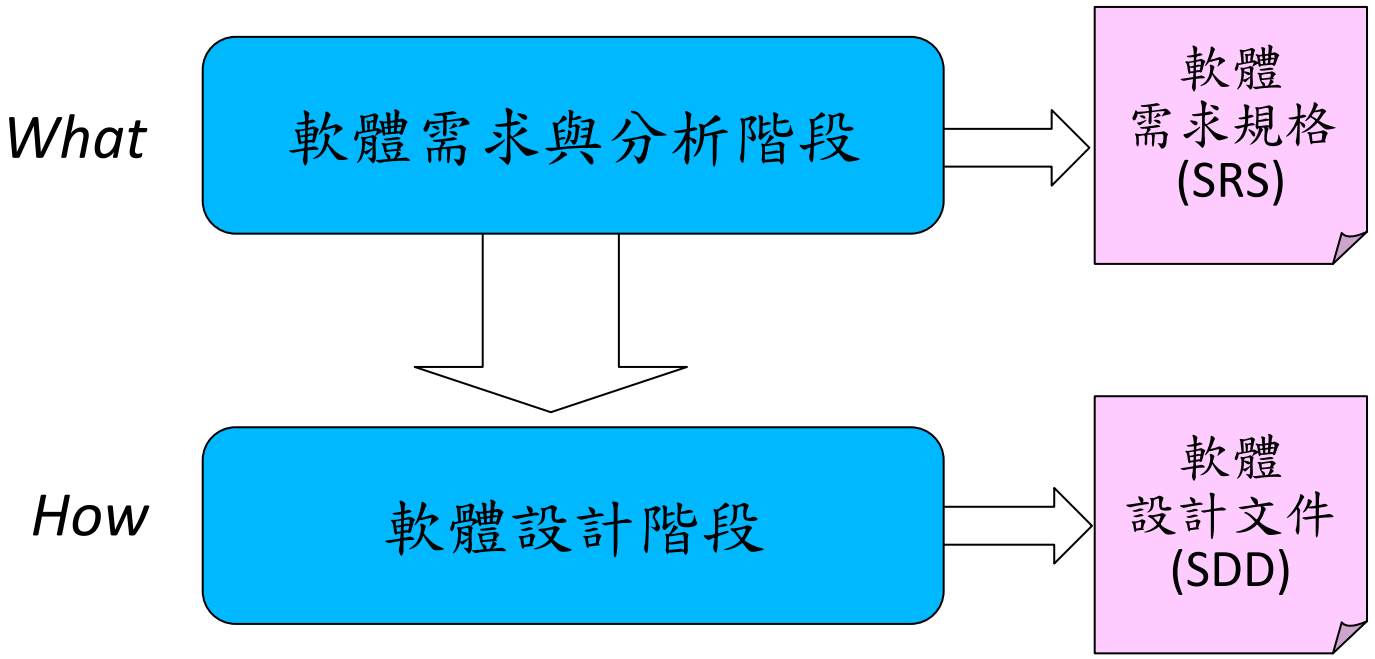
©2009, 李允中 教授, 國立中央大學資訊工程學系

4

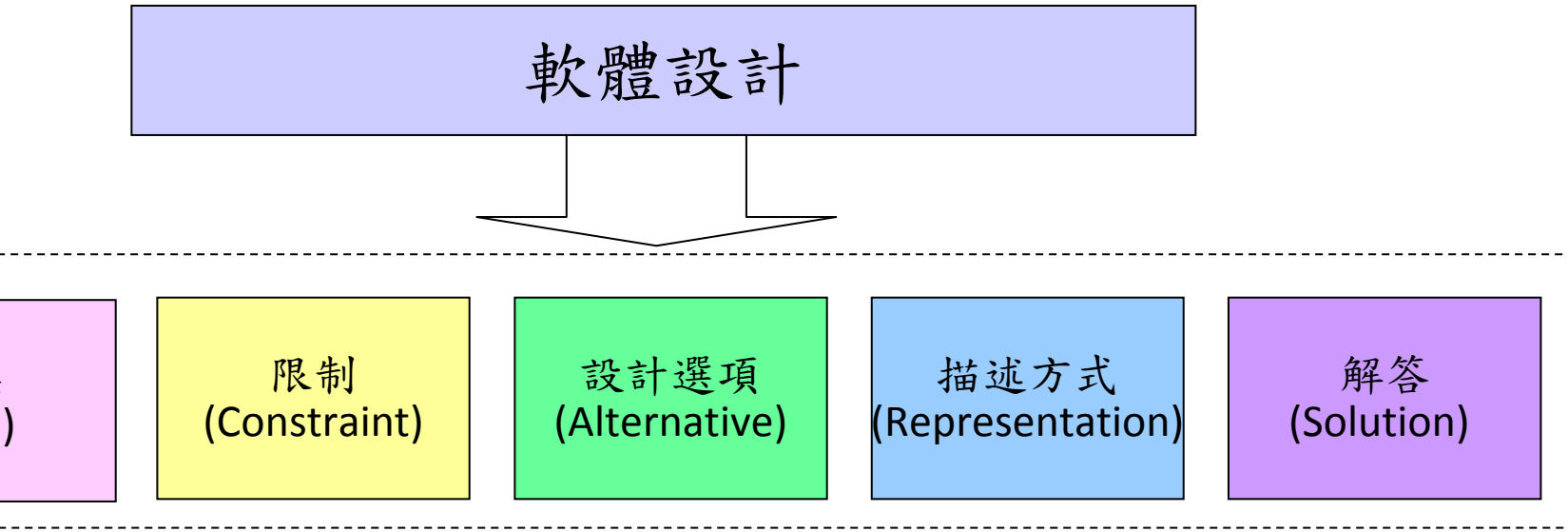
大綱

- 軟體設計概論
- 架構設計
- 軟體設計策略與方法
- 軟體設計規劃
- 進階軟體設計
- 本章總結
- 作業

軟體設計概論



軟體設計概論



可於道路上行駛的運輸工具

市場目標

經濟目標

銷售對象目標

採用何種引擎

油箱大小

預設行使的道路

安全的考量

預定銷售的對象

預定售價

汽車工程師透過腦力激盪，運用創造力以及相關的設計經驗，構思多個設計選項

選擇各種描述方式來塑模(Model)工程師心目中可以符合之前所設定之目標與限制的汽車模型

最後決定一個可行的解決方案(design solution)，稱之為設計解決方案，透過此解決方案描述如何建構這款汽車

抽象化1

- 抽象化(Abstraction)：由許多物件(Entities)中抽離出重要的特性(Features)來，而這些特性，足以讓被抽象化的物件，與別的物件分別開來。
  - 對一般汽車使用者而言，所感興趣的部分是車子的外觀、顏色、車子內裝、安全性、具備哪些功能等，而對於底盤的形式、油管的構造、引擎的設計等較無興趣。
  - 對於汽車研發製造者而言，其關注的焦點在於底盤、油箱、引擎的設計與結合上面，對於外觀、顏色等資訊則可能暫時不列在其關心之列。
- 抽象化階層(Levels of Abstraction)：對於不同的階層會提供不同的抽象化程度。

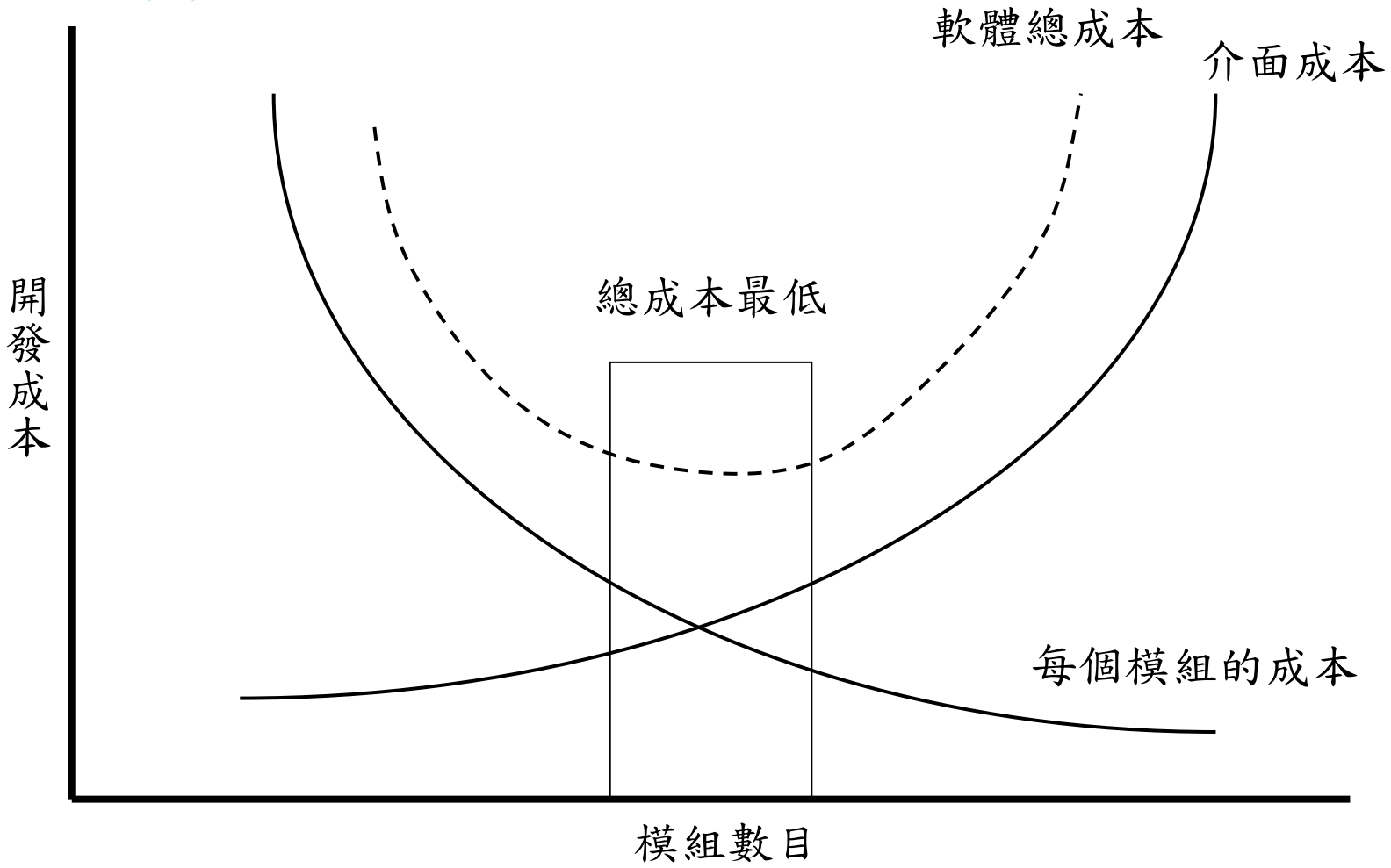
抽象化2

- 軟體系統發展的抽象化階層：
  - 軟體開發的初期（需求分析階段）：了解整個軟體系統會提供什麼樣的功能給使用者，而對於系統的內部運作則不是這階段的重點。
  - 設計的階段：著重了解軟體內部的設計概念，包含會使用什麼樣的軟體架構，以什麼資料結構以及演算法來實作軟體系統的功能等。
  - 軟體實作(Coding)階段：處理所使用程式語言的細節。
- 軟體設計的抽象化方式
  - 程序抽象化(Procedure Abstraction)
    - 一個特定且具備限制功能的指令程序。
  - 資料抽象化(Data Abstraction)
    - 描述一個資料物件的集合。
    - 抽象資料型態(Abstract Data Type)：C語言中的struct或是Java語言中的class。

模組化1

- 模組化(Modularity)：將軟體系統依其系統功能性或資料的相依性，分成數個容易了解與處理，而且可以互相溝通的單元(Module)或元件。
  - 透過軟體系統的模組化可以降低系統的複雜度，提高系統的可維護性，因此，可增加系統的穩定性以及可再利用性(Reusability)。
- 模組的大小以及數量：
  - 系統模組愈小，設計就愈容易，但會增加系統模組的數量，增加溝通的成本。
  - 提高模組大小，可減少系統模組的數量與溝通成本，但會增加模組的設計困難度，也降低模組的再利用性。
  - 就系統設計而言，決定模組的大小以及數量是種權衡損益分析(Trade-off Analysis)的結果。

模組化2



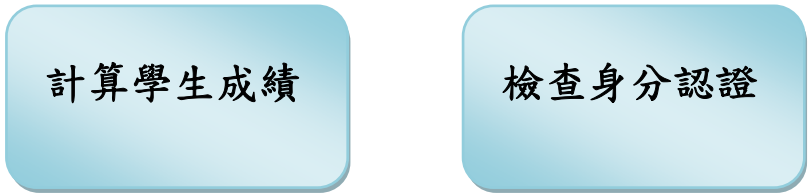
模組與成本關係圖

凝聚力1

- 凝聚力(Cohesion)：模組內部功能或資料的相依程度。
  - 凝聚力愈高，表示模組內部的功能或資料的相依程度愈高，通常較易維護並且適合再利用。
  - 凝聚力愈低，表示模組內部的功能或資料的相依程度愈低，將增加維護的成本，同時不利模組的再利用。
- 模組的凝聚力主要包含五種：
  - 功能凝聚力
  - 循序凝聚力
  - 溝通凝聚力
  - 程序凝聚力
  - 暫時凝聚力

凝聚力2

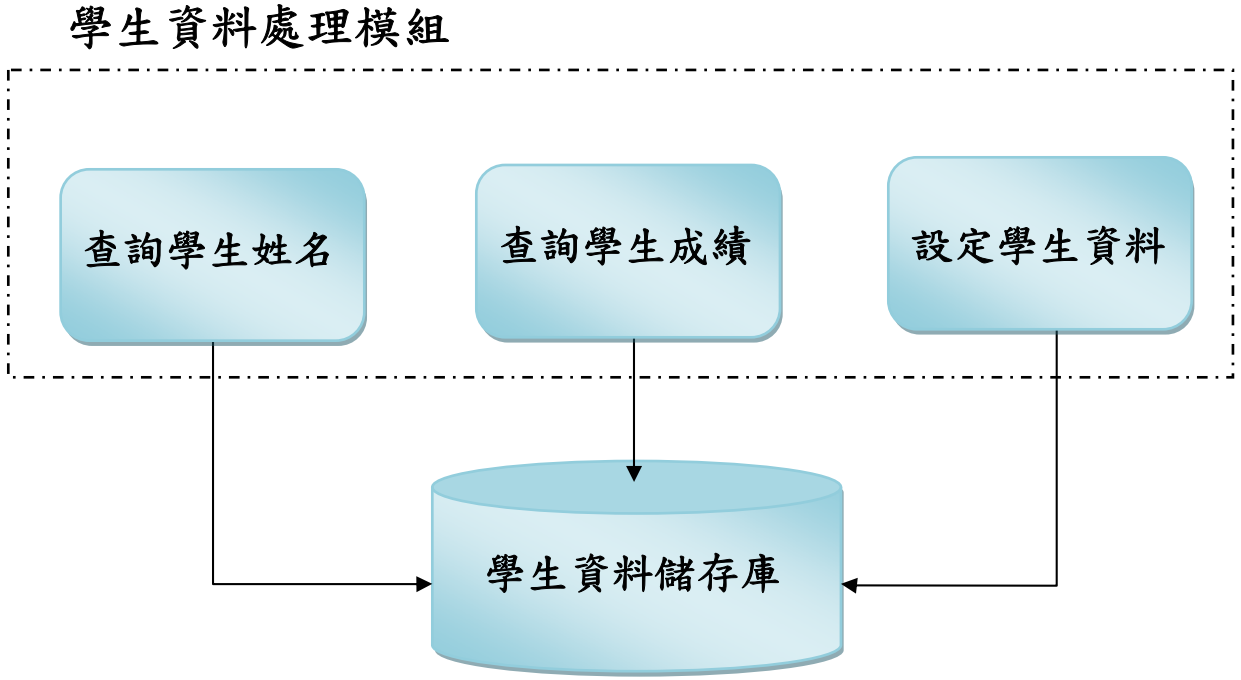
- 功能凝聚力(Functional Cohesion)：一個模組內部只完成某一項功能。



- 循序凝聚力(Sequential Cohesion)：模組內部具有多個功能，同時這些功能之間的執行具有順序性。
  - 功能之間會使用共用的資料

凝聚力3

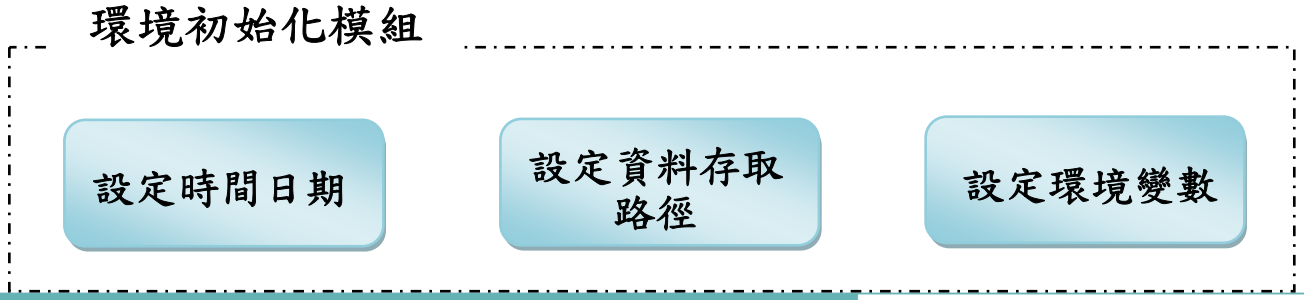
- 溝通凝聚力(Communication Cohesion)：模組內具有多個功能，且這些功能都會存取或處理相同的資料做為輸出與輸入資料。



凝聚力4

- 程序凝聚力(Procedural Cohesion)：模組內具有多個不同的功能，這些功能之間按照一定的順序執行。
  - 功能之間不需共用資料。

- 暫時凝聚力(Temporal Cohesion)：模組內之多個功能僅僅在時序上有關聯，卻不一定會有執行的順序或是共用資料的特性存在，



凝聚力5

低

模組化程度

高

等級	所代表的意義
暫時凝聚力	在同一時間處理
程序凝聚力	依特定順序處理（無相關資料）
溝通凝聚力	處理同一份資料
循序凝聚力	依特定順序處理相關資料
功能凝聚力	只處理特定功能

低

凝聚力程度

高

各種凝聚力的比較分析

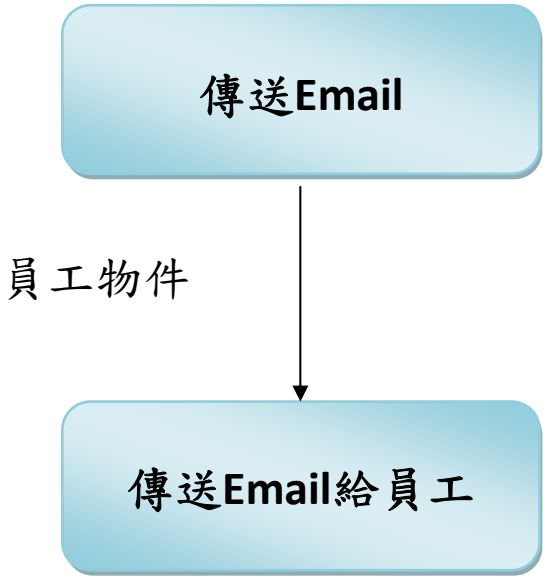
耦合力1

- 耦合力(Coupling)：模組之間的獨立程度。
  - 耦合力愈低，表示模組間的獨立程度愈高，較易維護並且適合再利用。
  - 耦合力愈高，表示模組間的獨立程度愈低，易增加系統溝通的錯誤，提高維護的成本。
- 耦合力可以區分為四種：
  - 資料耦合力
  - 資料結構耦合力
  - 控制耦合力
  - 全域耦合力

耦合力2

- 資料耦合力(Data Coupling)：模組之間使用一些簡單的資料型別做為模組之間的參數傳遞。
- 資料結構耦合力(Data Structure Coupling)：若傳遞的資料不是基本資料型態，而是複雜的資料結構（或物件）。

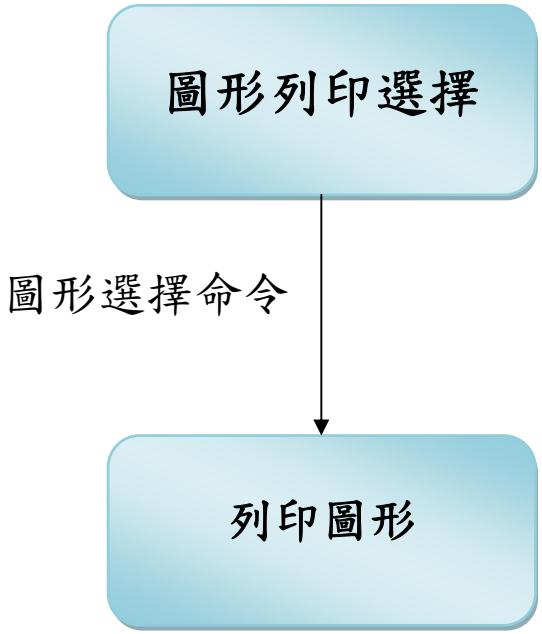
```
public class Emailer {
  public void sendEmail(Employee e, String text)
  {...}
  ...
}
```



耦合力3

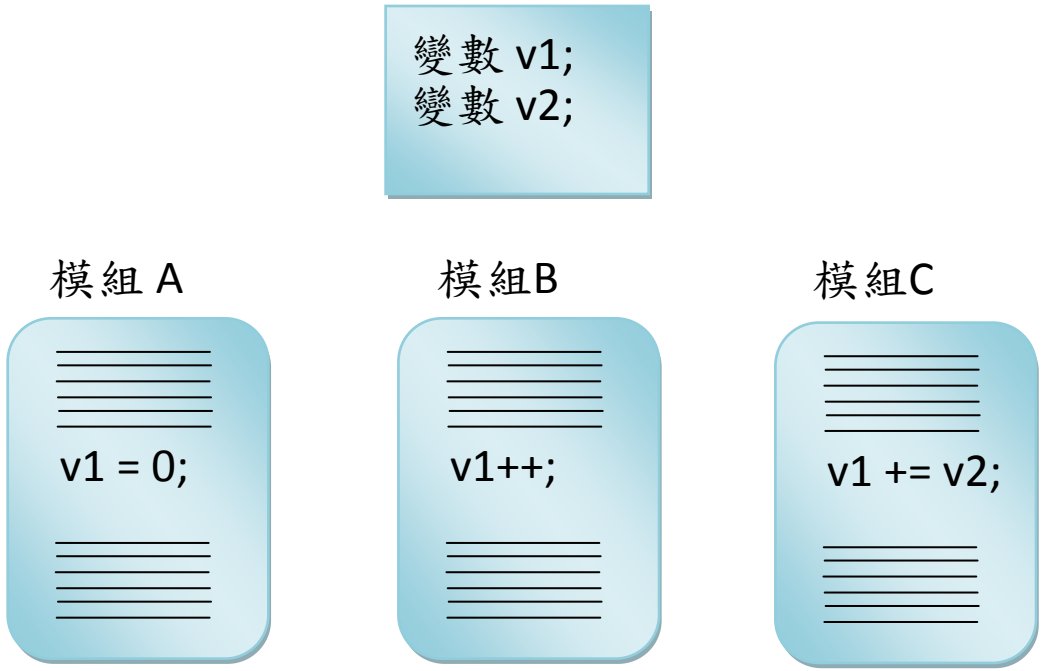
- 控制耦合力(Control Coupling)：模組之間傳遞的是旗標(Flag)或是控制訊號。

```
public void routineX(String command) {
  if (command.equals("drawCircle")
    drawCircle();
  else if (command.equals("drawRectangle")
    drawRectangle();
  else
    drawTriangle();
}
```

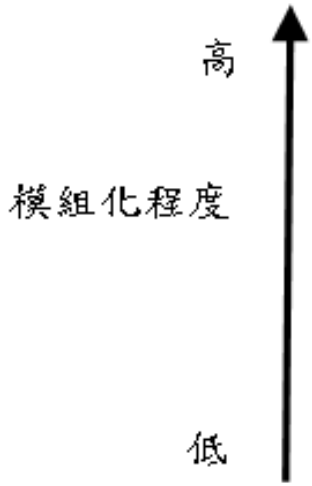


耦合力4

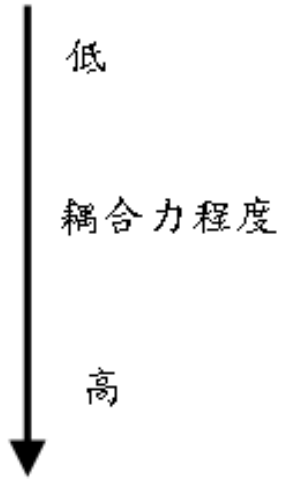
- 全域耦合力(Global Coupling)：模組之間使用相同的資料區，並且各個模組皆可以對此資料區內的資料進行讀寫的動作。



耦合力5



等級	所代表的意義
資料耦合力	彼此傳遞基本資料型態
資料結構耦合力	彼此傳遞複雜資料結構型態
控制耦合力	傳遞特定控制訊號
全域耦合力	共用同一資料區

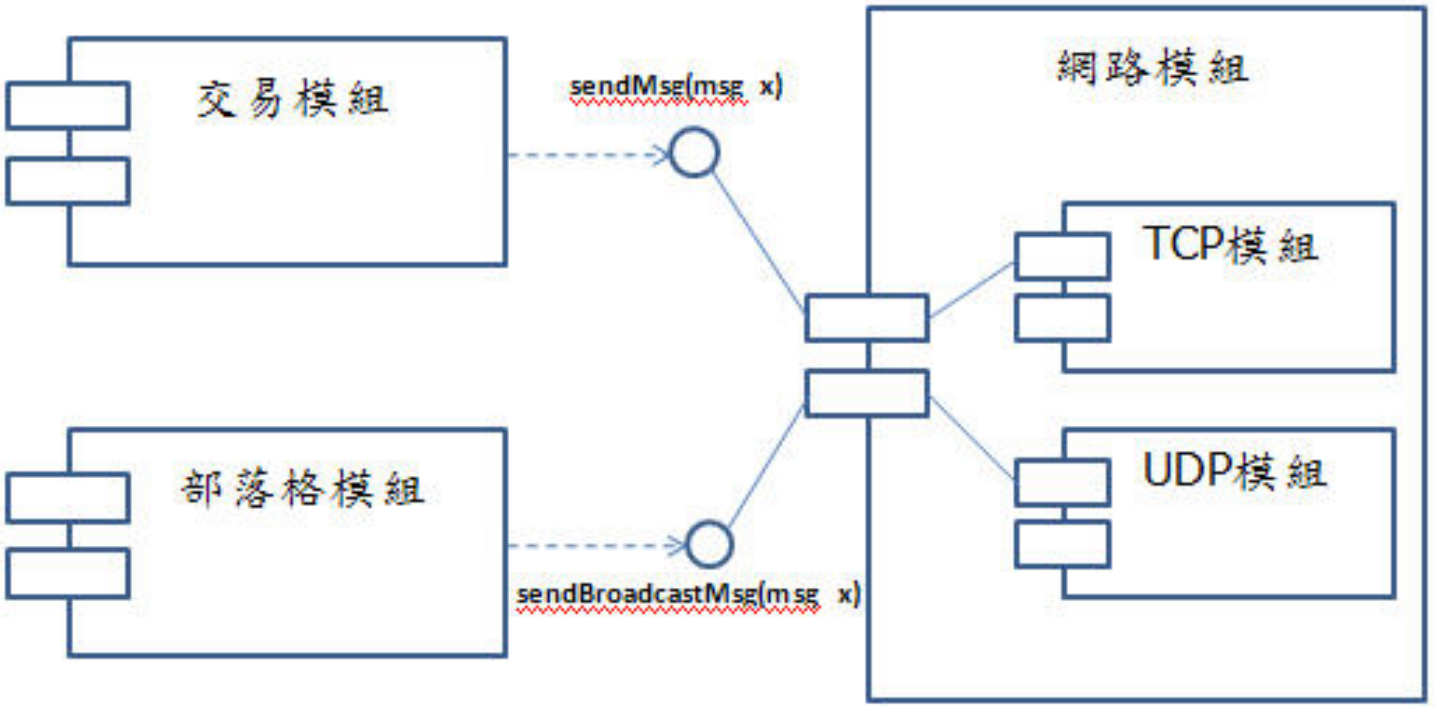


各種耦合力之評量

架構設計1

- 軟體架構設計是架構整體系統組織的過程，包含以下幾個活動：
  - (1)將系統分割成數個子系統
  - (2)決定這些子系統如何互動
  - (3)決定這些子系統的介面
- 軟體架構的構面包含靜態面與動態面：
  - 架構的靜態面：架構設計展現系統切割成不同的子系統，以及各個子系統的介面連結。
  - 架構的動態面：架構設計可以描述子系統之間的動態互動行為，資料如何在各個子系統之間傳遞與處理，以及這些子系統執行的動態行為等。

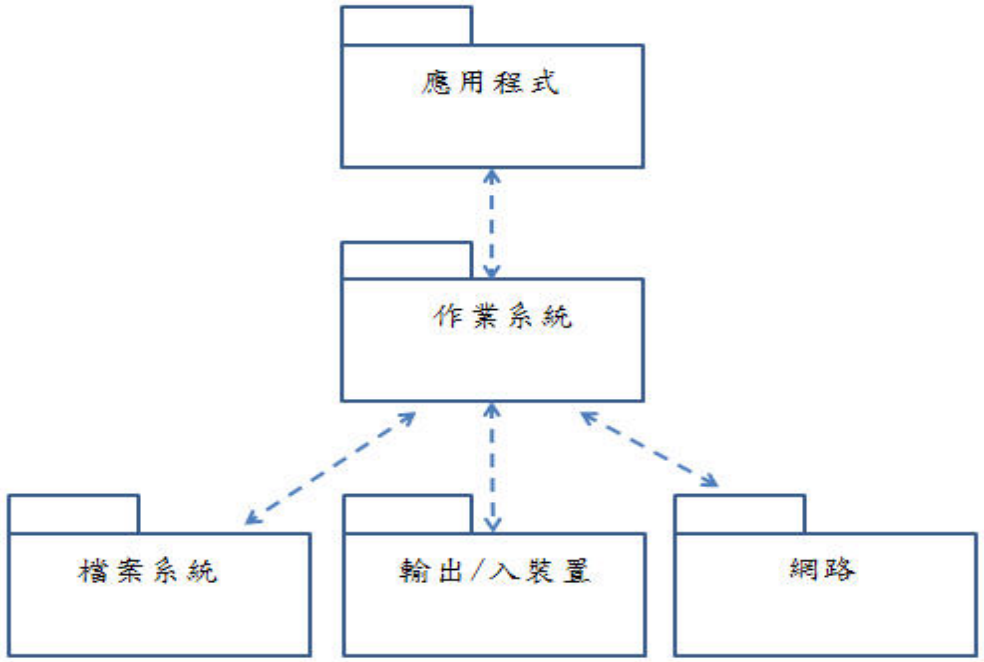
架構設計2



架構設計範例

架構樣式1

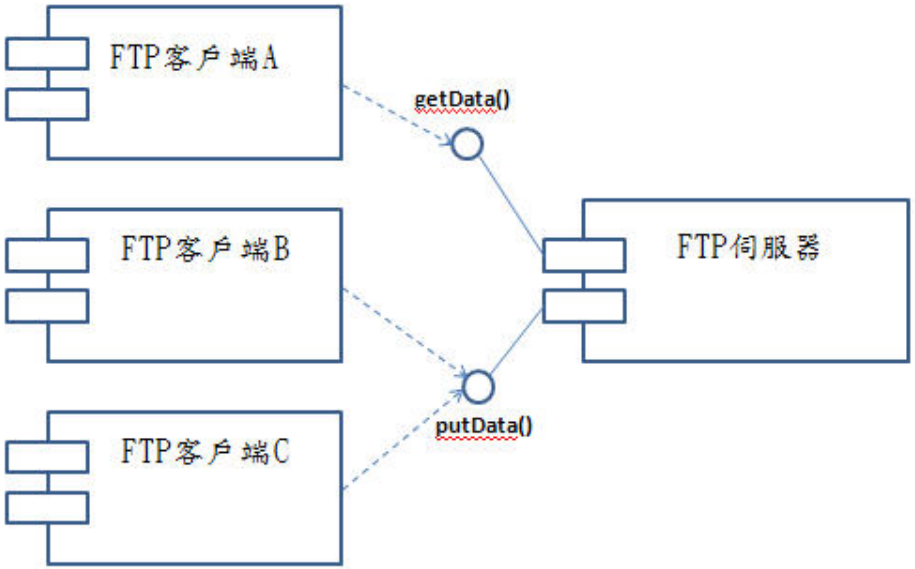
- 階層架構樣式(Layer Architecture Pattern)
  - 子系統之間區分成不同的階層。
  - 每一階層都會開放一組嚴謹定義的介面(Well-Defined Interface)，讓上層元件利用。



架構樣式2

- 分散式架構樣式(Distributed Architecture Pattern)：架構在網際網路的環境當中，其系統各個模組是分別部署於網路上多台主機，以提升系統的可擴充性及運算能力。
- 常見的分散式架構樣式：
  - 客戶－伺服器架構(Client-Server Architecture)
    - 至少有一個子系統提供相關的服務（稱之為伺服器端），而數個客戶端子系統可同時存取伺服器端所提供的服務。
  - 多層式架構(N-Tier或稱Multi-Tier Architecture)
    - 軟體系統的各功能模組間有既定之”客戶-伺服器”關係，例如三層架構包含使用介面呈現模組、系統邏輯運算模組、資料處理模組，前者為後者之客戶端，後者為前者的伺服器端。
  - 點對點架構(Peer-to-Peer Architecture)
    - 沒有嚴格的客戶端與伺服器端的限制，亦即，一個子系統元件可以是提供服務的伺服器端，同時又是要求服務的客戶端。

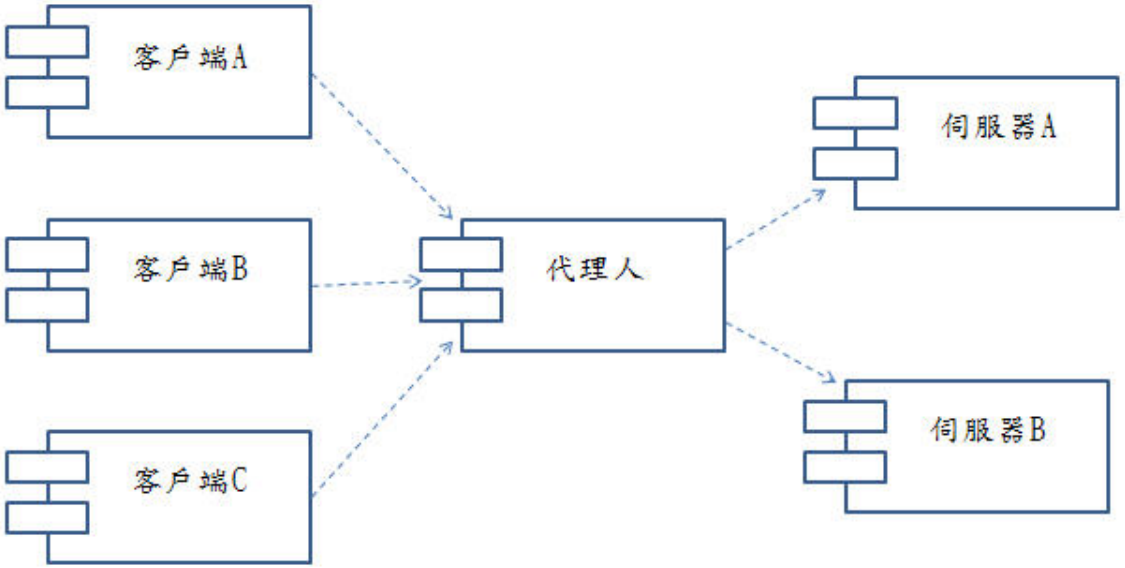
架構樣式3



客戶－伺服器架構

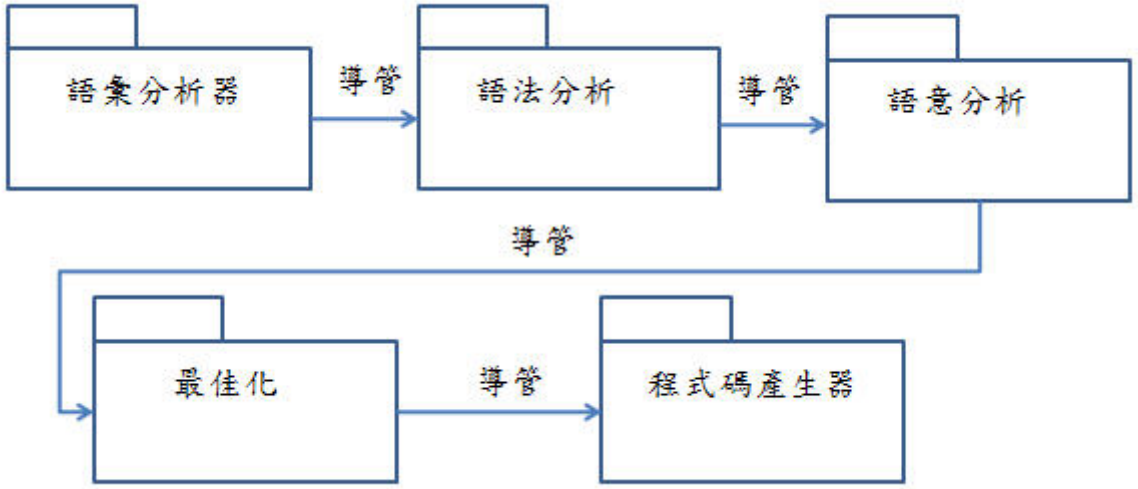
架構樣式4

- 代理人架構樣式(Broker Architecture Pattern)：透過一個中介的代理人子系統(Broker)將客戶端與伺服器端分隔開來，客戶端不必知道伺服器端是誰，只要向代理人要求服務即可。
  - 可有多個伺服器同時來提供相同的服務，增加系統運作時的彈性，甚至可達到負擔平衡(Load Balance)的作用。
  - CORBA即是種典型的代理人架構。



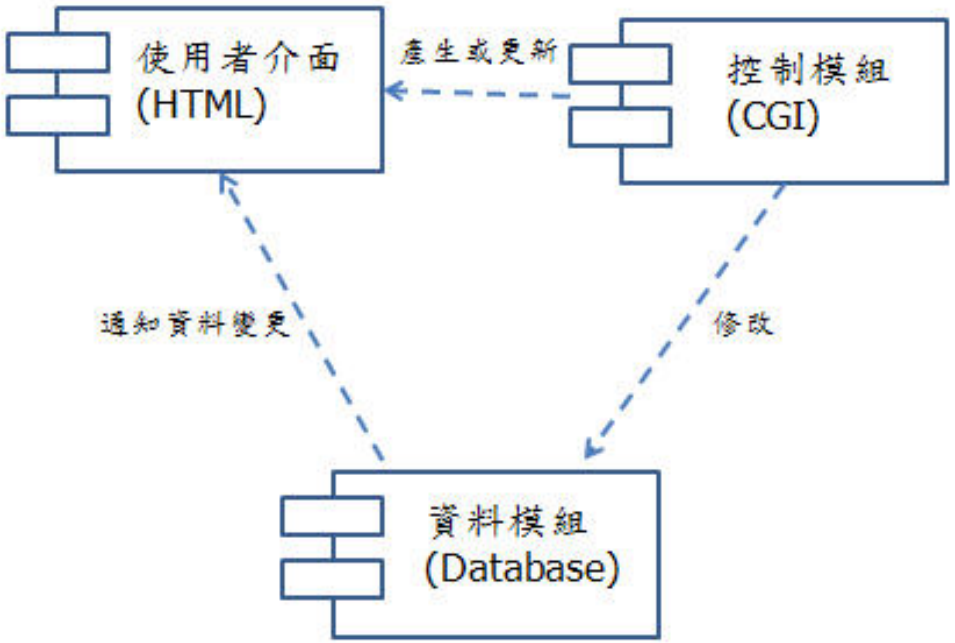
架構樣式5

- 管狀架構(Pipe and Filter Architecture Pattern：描述資料流的架構，也是一種循序處理的架構。
  - 把各子系統視為獨立運行不同功能的過濾器(Filter)，而資料透過具方向性的導管(Pipe)傳送到下一個過濾器進行運算處理。
  - 過濾器的輸入或是輸出導管可以連接多個，並不限定在只能有一個輸出或是輸入導管連接。



架構樣式6

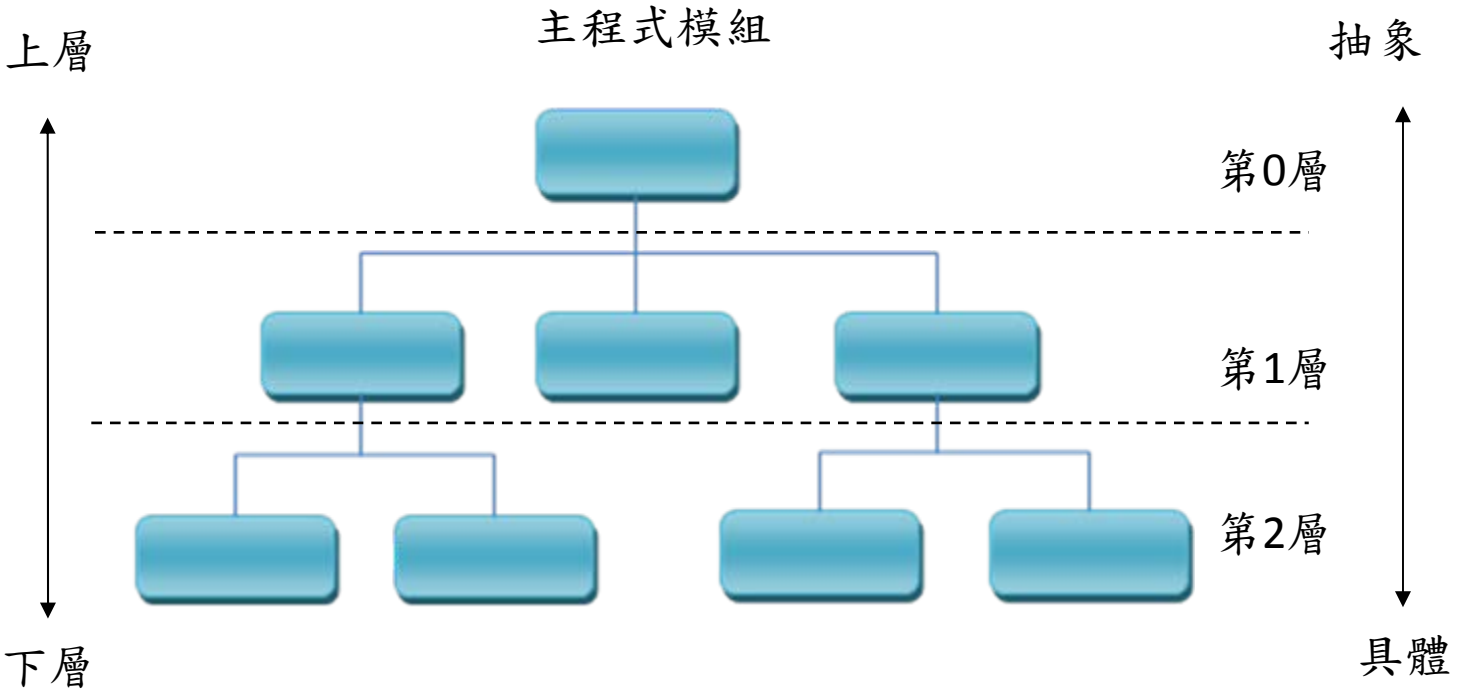
- MVC架構(MVC Architecture Pattern)：將整個系統切割成三種不同型態的子系統，分別稱為資料(Model)、顯示(View)、以及控制(Control) 。



- 資料子系統元件：儲存系統的常駐資料，這些資料可以被相關的控制模組功能來進行處理，並由顯示模組來呈現。
- 顯示子系統元件：包含所有的使用者介面模組，負責提供各式顯示介面供系統使用者讀取資訊或進行操作。
- 控制子系統元件：包含所有的控制模組，負責處裡來自使用者介面的事件，並進行各所屬程式邏輯運算(Business Logic)。

4 軟體設計策略與方法：通用策略1

- 各個擊破的方法：以抽象化為核心概念，將問題的解決方法透過由上而下的分解，將複雜的問題拆解為數個較小並且容易解決的模組，經由小模組的解決，進而解決複雜的問題。

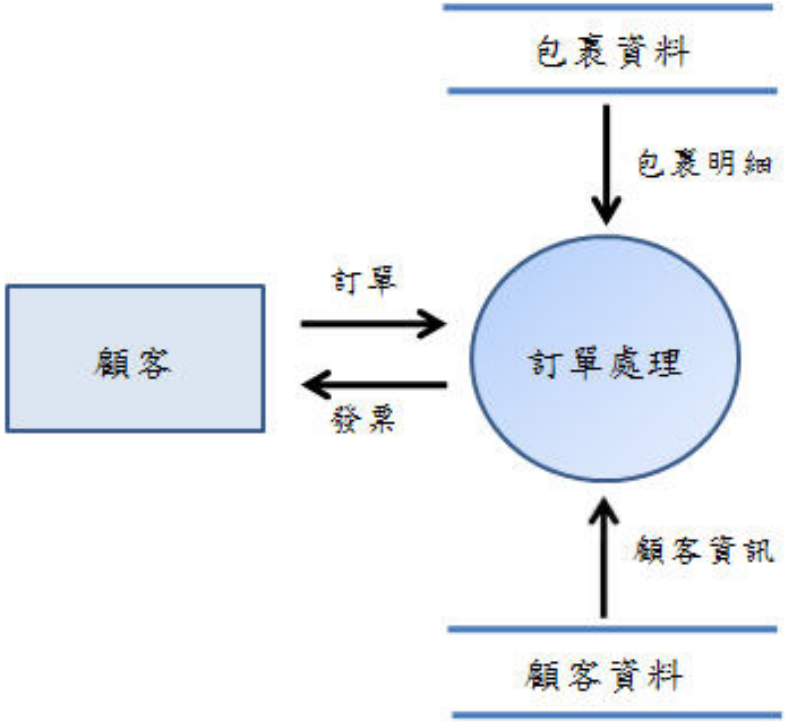


軟體設計策略與方法：通用策略2

- 由上而下的方法：
  - 把一個大的問題先分解成幾個次小問題，再把各個次小的問題再分解成更小的幾個問題，直到可以讓每個問題都能單獨的處理。
  - 以功能分解(Function Decomposition)為中心，透過功能的切割，可以區分成不同的階層
- 由下而上的方法是以物件導向設計為主流，透過物件以及物件之間的互動來完成相關的功能。

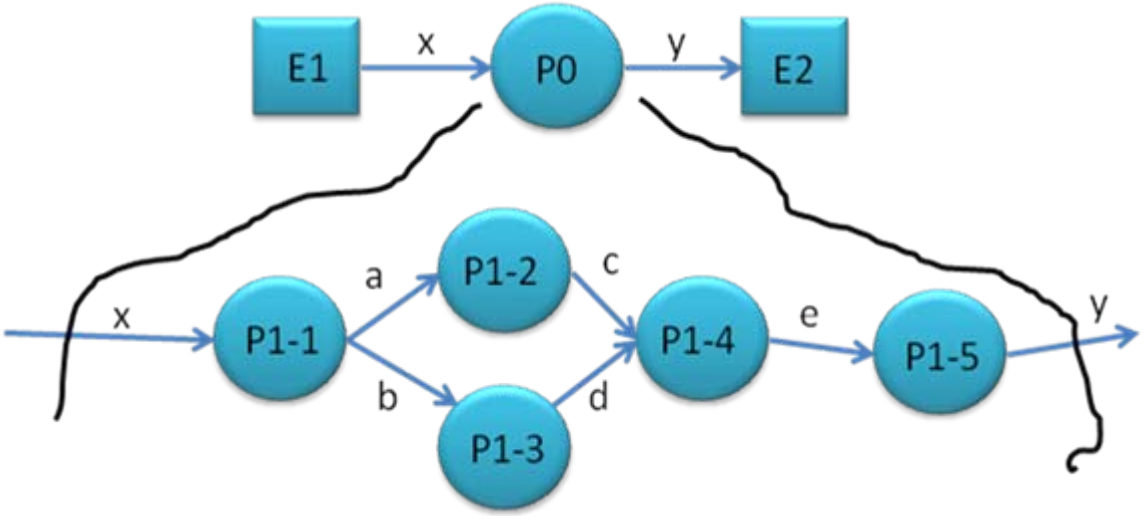
4 軟體設計策略與方法：功能導向設計1

- 功能導向設計(Function-Oriented Design)：是傳統結構化分析設計方法，以系統所提供的功能(Function)為主要思考模式。
- 資料流設計(Data Flow Diagram, DFD)：
  - 將軟體系統描述為資訊在系統中傳遞與轉換的過程。
  - 圓形表示系統或是處理流程(Process)，方形代表外部的實體(Entity)，箭號代表資料流(Data Flow)，箭號上的文字表示資料，雙線代表資料儲存區(Data Store)。



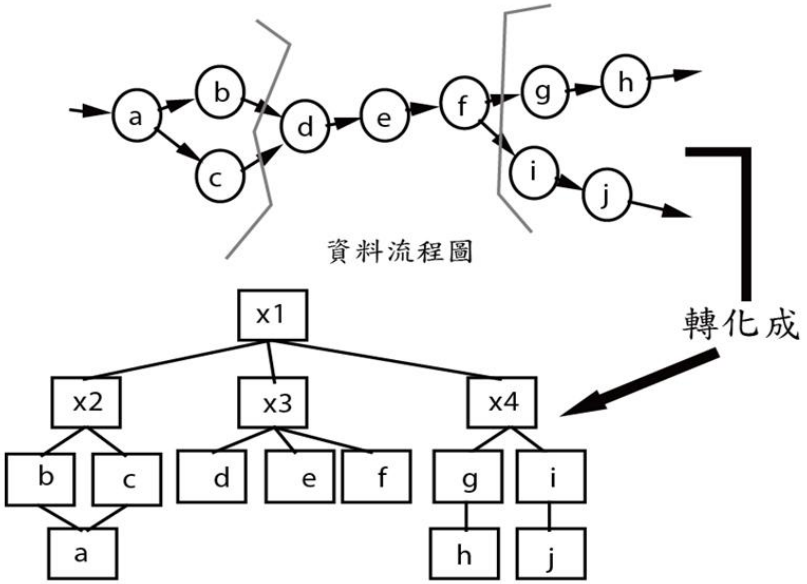
4 軟體設計策略與方法：功能導向設計2

- 資料流程圖的細化：
  - 一般而言會將每個流程利用由上而下的方法進行細化(Refinement)
  - 第0層細化到第1層時，流程的輸入與輸出必須要一致
    - 例如，圖中的x和y。
  - 新的細化流程中，可以產生新的資料以及資料流
    - 例如，圖中的a到e。



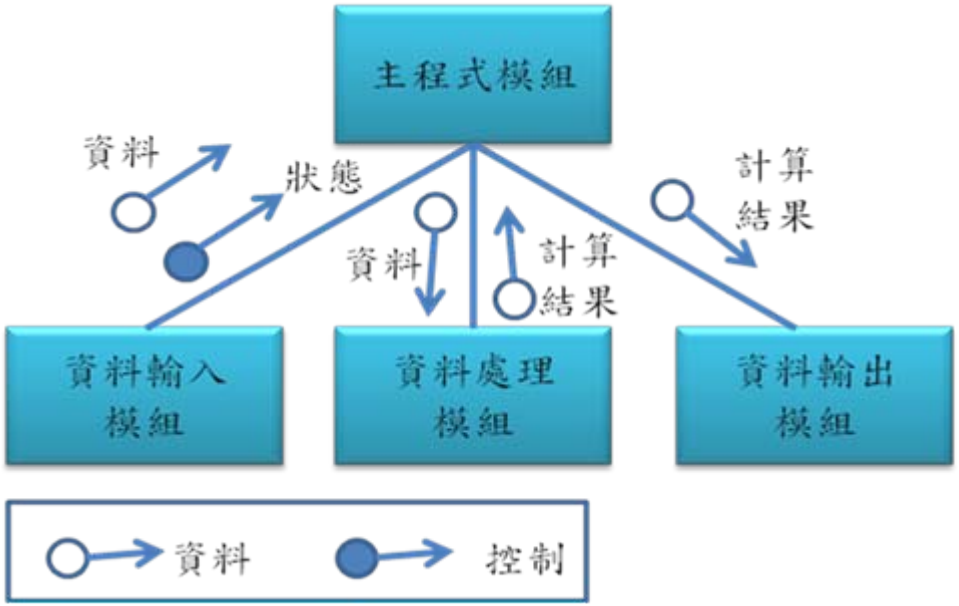
4 軟體設計策略與方法：功能導向設計3

- 結構圖(Structured Chart)：是由Ed Yourdon所提出的一項進行系統設計時的工具，以樹狀或層次的表示方法來表現系統模組的結構與組合關係。
  - 可將資料流程圖(DFD)對應到設計的結構圖。



4 軟體設計策略與方法：功能導向設計4

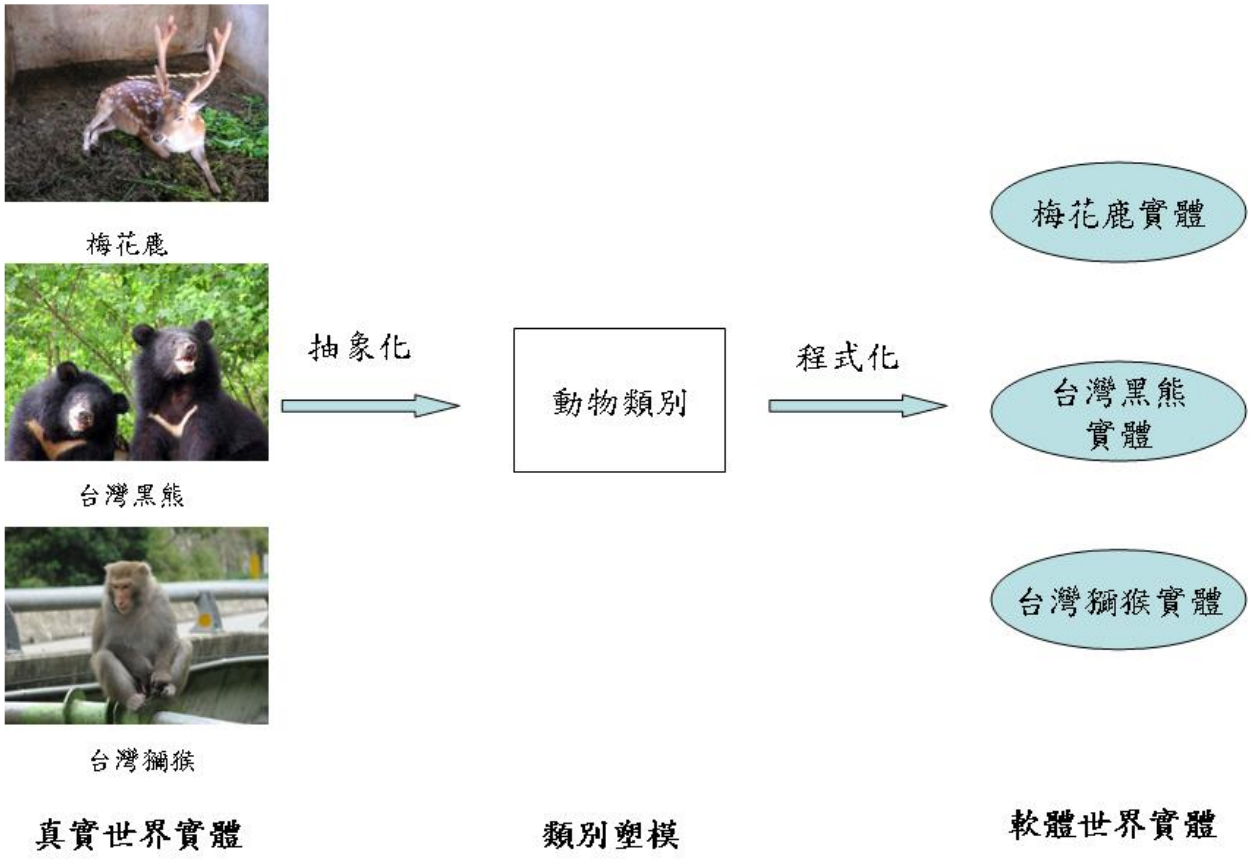
- 資料流程圖到結構圖之轉換
  - 資料流程圖對應到結構圖並沒有嚴格的方式。
  - 常見的方法是將資料流程圖簡單區分為三個部分，資料輸入、資料處理、資料輸出等三部分，每一個部分對應到結構圖中的一個模組。



4 軟體設計策略與方法：物件導向設計1

- 物件導向設計(Object-Oriented Design)方法：以類別(Class)與物件(Object)為軟體元件的設計方法。
  - 主要的目的是分析軟體系統的需求，並以物件塑模(Modeling)以及物件之間的互動(Interaction)來完成軟體系統的需求。
  - 基本的設計模式有三個步驟：
    - (1)至問題領域的真實世界中，界定出可能的物件。
    - (2)根據問題領域所找出來的物件，以抽象化的方式群組成類別。
    - (3)根據每個類別決定該類別與其他類別互動的相關責任(Responsibility)。

4 軟體設計策略與方法：物件導向設計2



物件導向設計概念圖

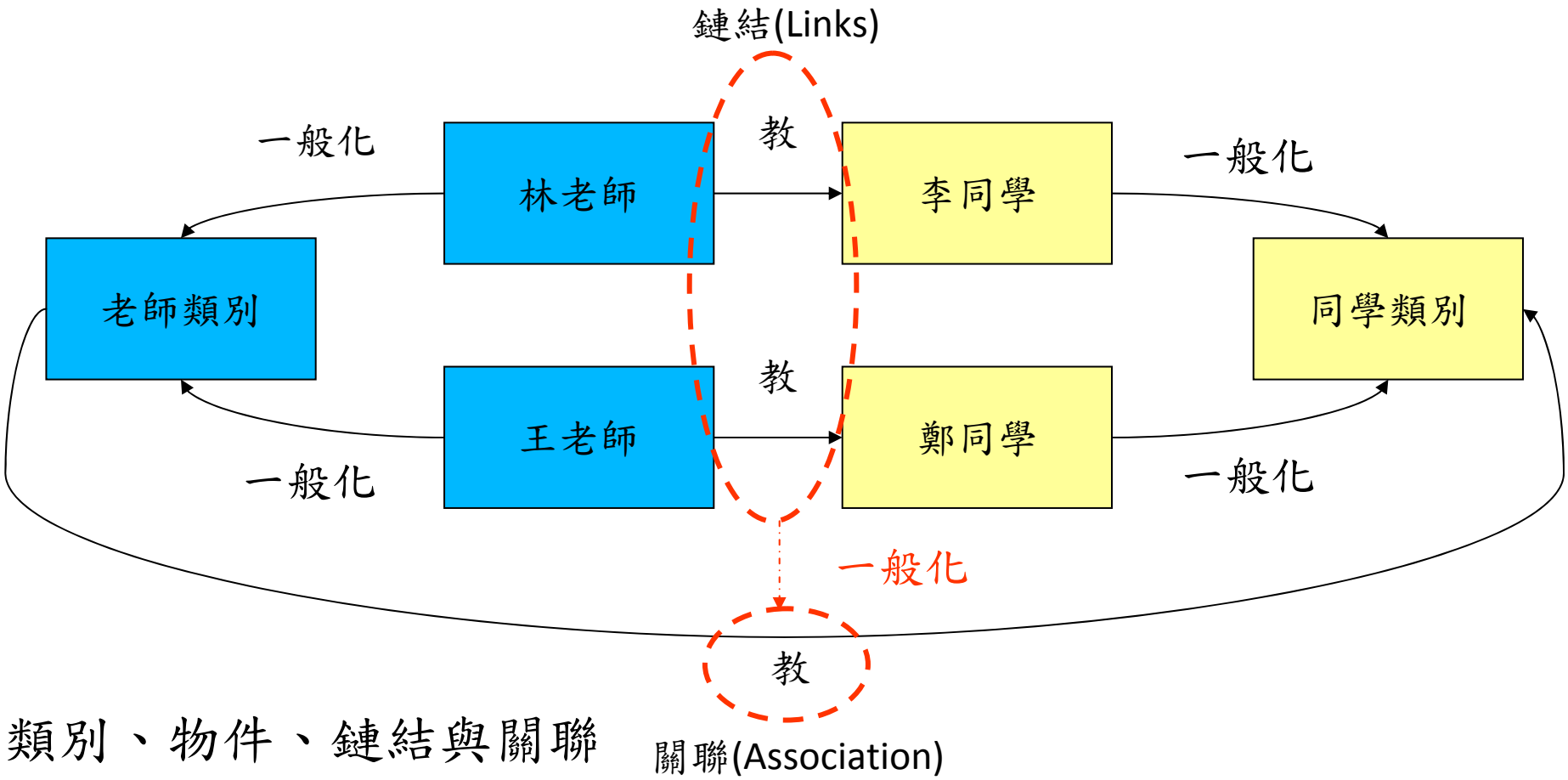
4 軟體設計策略與方法：物件導向設計3

- 物件(Object)：是與真實世界有關的事物模型，可以是真實世界中具體存在的事物，也可以是抽象的概念。
  - 例如，桌子、椅子、電腦、人、汽車等具體事物，或會議、協議等抽象概念。
- 屬性(Attribute)：物件會塑模該物件的相關屬性，以反映物件的狀態，同時也可以透過這些屬性的值，來區別不同的物件。
  - 例如，桌子物件可能包含長、寬、高、顏色等屬性。
- 義務(Responsibility)：每種物件都有其特別的行為(Behavior)，有些人稱之為義務。
  - 例如，車子物件包含啟動、前進、後退、轉彎等行為。
- 互動(Interaction)：真實世界的運作是透過車子等具體存在或是概念的事物來完成，而對應到軟體世界，則是由一堆物件之間的互動(Interaction)來完成特定功能。

4 軟體設計策略與方法：物件導向設計4

- 類別(Class)：
  - 對於具有類似性質、相同的行為、意義及共同關係的物件之描述即為類別。
  - 類別是具相同性質物件的集合，物件則為類別的實體(Instance)。
  - 每一個類別的成員，都共用相同的性質。
  - 例如，銀行帳戶是一個類別，每一位客戶所開設的個別帳戶是類別的實體，即物件。
- 鏈結(Links)：物件實體之間的關係(Relationship)稱為鏈結。
  - 例如：林老師「教」李同學一門課，即為兩個實體間的鏈結關係。
- 關聯(Association)：實體物件一般化後，可得到類別(Classes)。而將鏈結抽象化、一般化後便得到類別之間的關聯。

4 軟體設計策略與方法：物件導向設計5



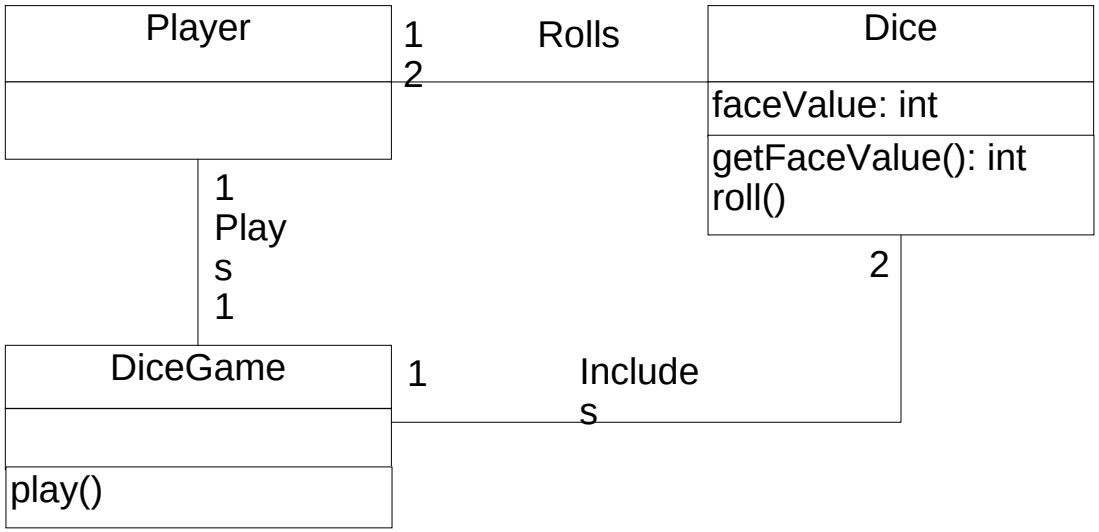
類別、物件、鏈結與關聯

4 軟體設計策略與方法：物件導向設計5

- 聚合(Aggregation)：表示物件之間的組成的關係。
  - 例如，汽車物件是由輪子物件、引擎物件、齒輪箱物件、車殼物件等所組成的集合體。
- 繼承(Inheritance)和一般化(Generalization)：
  - 一個被繼承的父類別(Parent Class)又稱為超類別(Super Class)，子代類別(Child Class)被稱為副類別(Subclass)。
  - 子代類別的行為與特性繼承自父代。
  - 子類別的實體物件也是父類別的實體物件。（「IS-A」或「A-KIND-OF」）

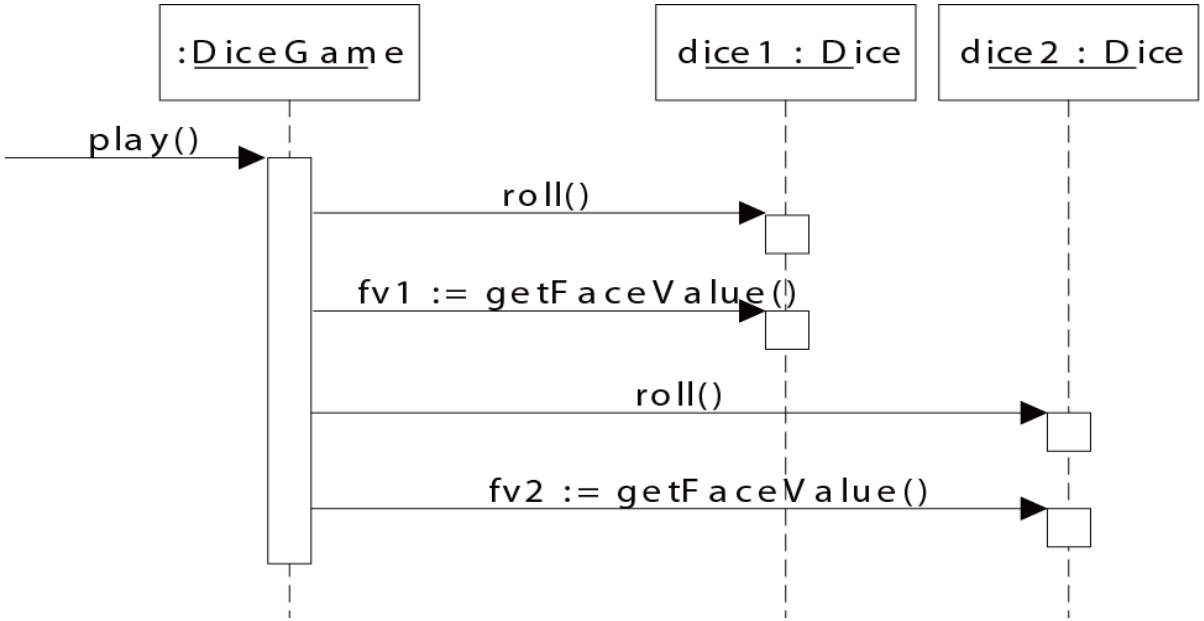
4 軟體設計策略與方法：物件導向設計6

- 物件導向設計流程可歸納為兩個步驟：
  - 物件與類別塑模(Object Class Modeling)
  - 細部流程設計(Detail Design)
- 物件與類別塑模：根據系統需求，制定軟體系統所需的類別、類別屬性、類別動作以及一些簡單的類別與類別之間的關係。



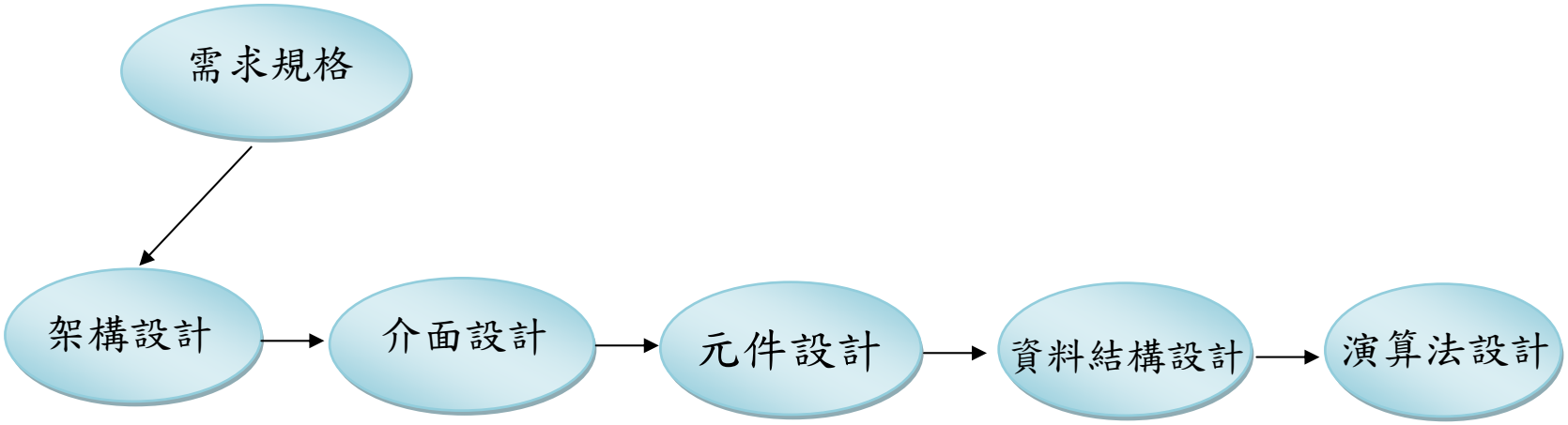
4 軟體設計策略與方法：物件導向設計7

- 細部流程設計：針對系統的使用方式或系統的行為，規範物件之間的訊息傳遞和合作的方式，以達成系統的功能任務。



4

軟體設計規劃：軟體設計步驟



軟體設計活動階段

4 軟體設計規劃：軟體設計規劃書1

- 軟體設計規劃書
  - 明確地規範系統的架構、模組、資料結構與演算法。
  - 可以提供程式撰寫人員與系統設計人員溝通的管道，同時提供未來軟體系統進行擴充與維護的參考。
- IEEE 1016軟體設計規劃書大綱[IEEE Std 1998]

1. 基本介紹(Introduction)
   1. 設計概念(Design Overview)
   2. 需求追溯表(Requirements Traceability Matrix)
2. 系統架構設計(System Architecture Design)
   1. 選擇系統架構(Choose System Architecture)
   2. 討論可能設計(Discussion of Alternative Designs)
   3. 系統介面設計(System Interface Design)

軟體設計規劃：軟體設計規劃書2

- IEEE 1016軟體設計規劃書大綱[IEEE Std 1998]（續）
  3. 元件細部描述(Detailed Description of Components)
     1. 元件1細部描述
     2. 元件2細部描述
     3. 元件n細部描述
  4. 使用者介面設計(User Interface Design)
     1. 使用者畫面設計(Screen Image)
     2. 使用者介面元件設計(Objects and Actions)
  5. 其他文件(Additional Material)

4

進階軟體設計：設計樣式1

- Erich Gamma等人對設計樣式的定義：「 一組描述類別和物件之間的溝通，然後在特定的環境下被用來解決設計上的問題。」
- 設計樣式之基本元件
  - 設計樣式名稱：一個可以代表設計樣式本質的名詞，目的是提供使用者對設計樣式有直覺上的了解，進而增加使用上的效率。
  - 設計樣式目的：設計樣式的用途、設計樣式的原理與設計樣式所強調的重點或問題。
  - 設計樣式的別名(also known as)：可代表設計樣式目的的名詞可能不只一個，因此設計樣式亦提供其他可代表的名詞供使用者利用。

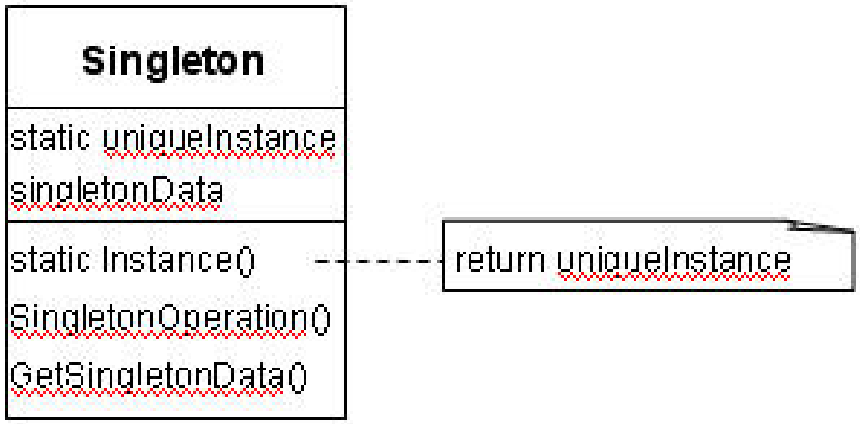
4

進階軟體設計：設計樣式2

- 設計樣式之基本元件（續）
  - 設計樣式動機：以劇本的方式來描述設計樣式所要解決的問題，以類別和物件的結構來表示解決的方法，此劇本將幫助使用者了解設計樣式所依循的原則。
  - 設計樣式之應用性：主要表達(1)適合此設計樣式之狀態；(2)此設計樣式要解決何種不足之設計；與(3)指出這些合適的狀態之不同處。
  - 設計樣式結構：以類別圖形來表示設計樣式，使用的符號是根據UML類別圖和互動式圖(Interaction Diagram)表示物件之間靜態結構和互動關係。
  - 設計樣式的參與者：描述所有參與設計樣式的類別和這些類別所要完成的責任。
  - 參與者之間的合作：描述設計樣式成員如何合作以達成彼此之間的責任。

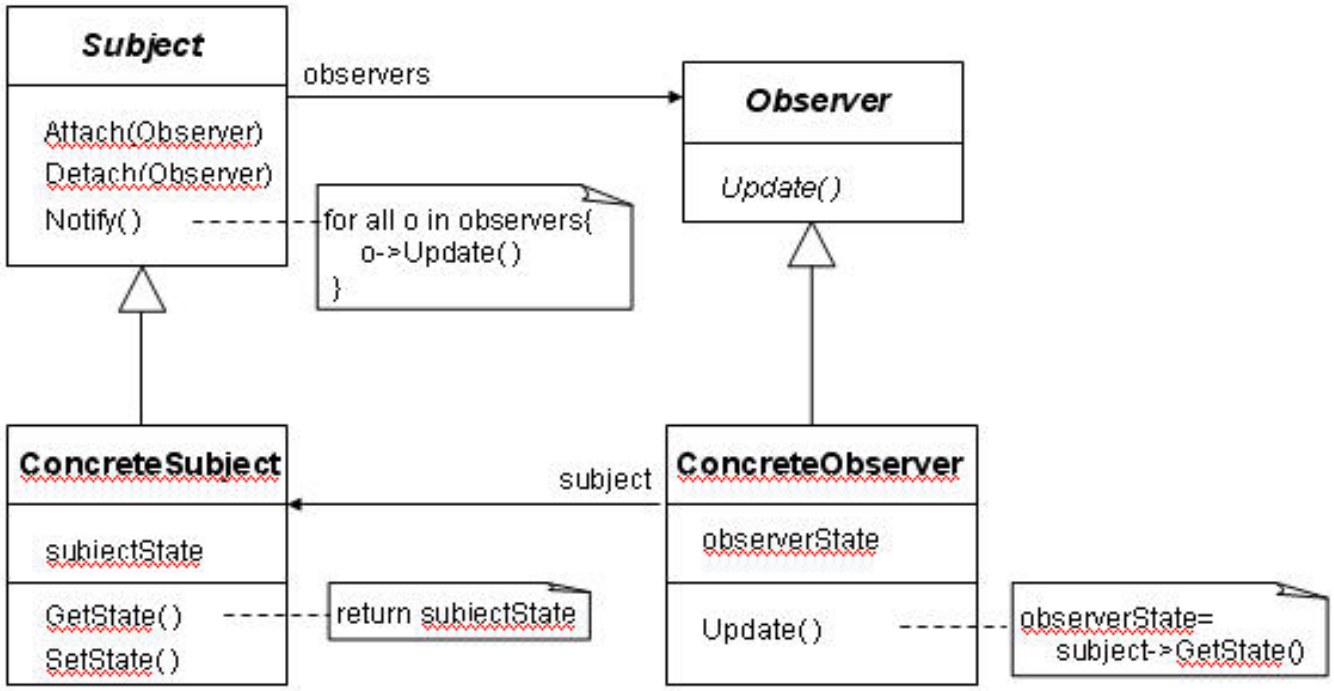
4 進階軟體設計：設計樣式3

- Singleton設計樣式：單一個體(Singleton)樣式確保類別只會有一個物件實體存在，並提供單一存取窗口。



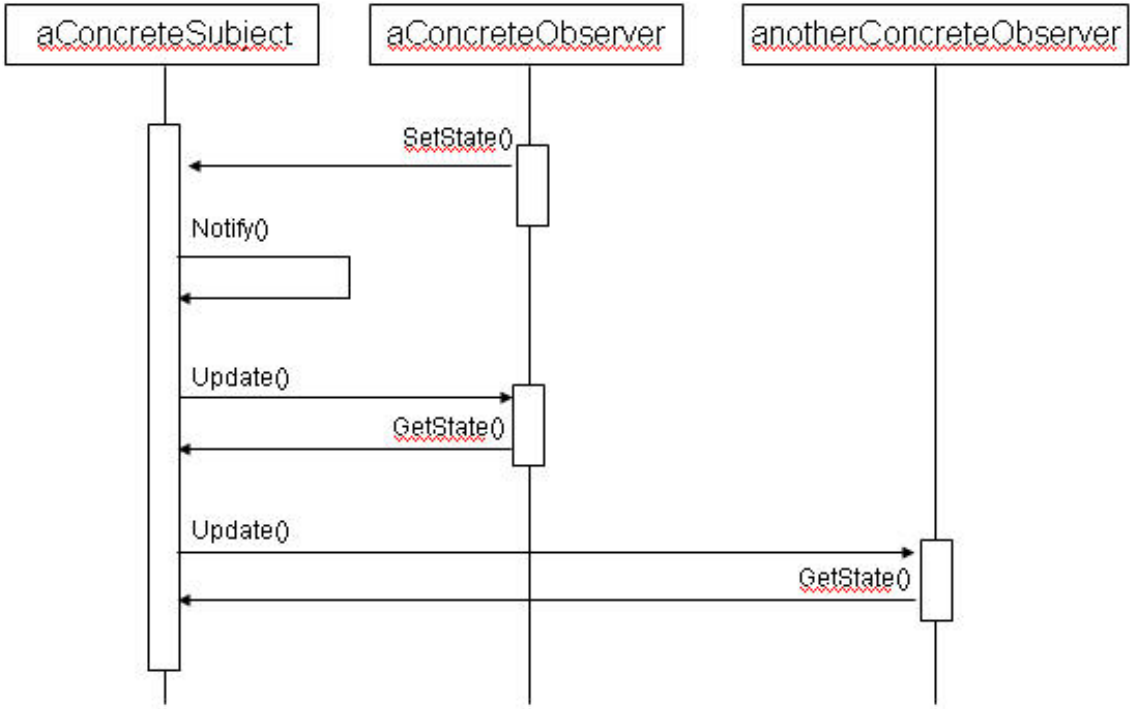
4 進階軟體設計：設計樣式4

- Observer設計樣式：觀察員(Observer)樣式在物件之間定義一組一對多的相依關係，讓一個物件的狀態改變，立刻能通知到所有關心此狀態改變的所有物件。



4

進階軟體設計：設計樣式5

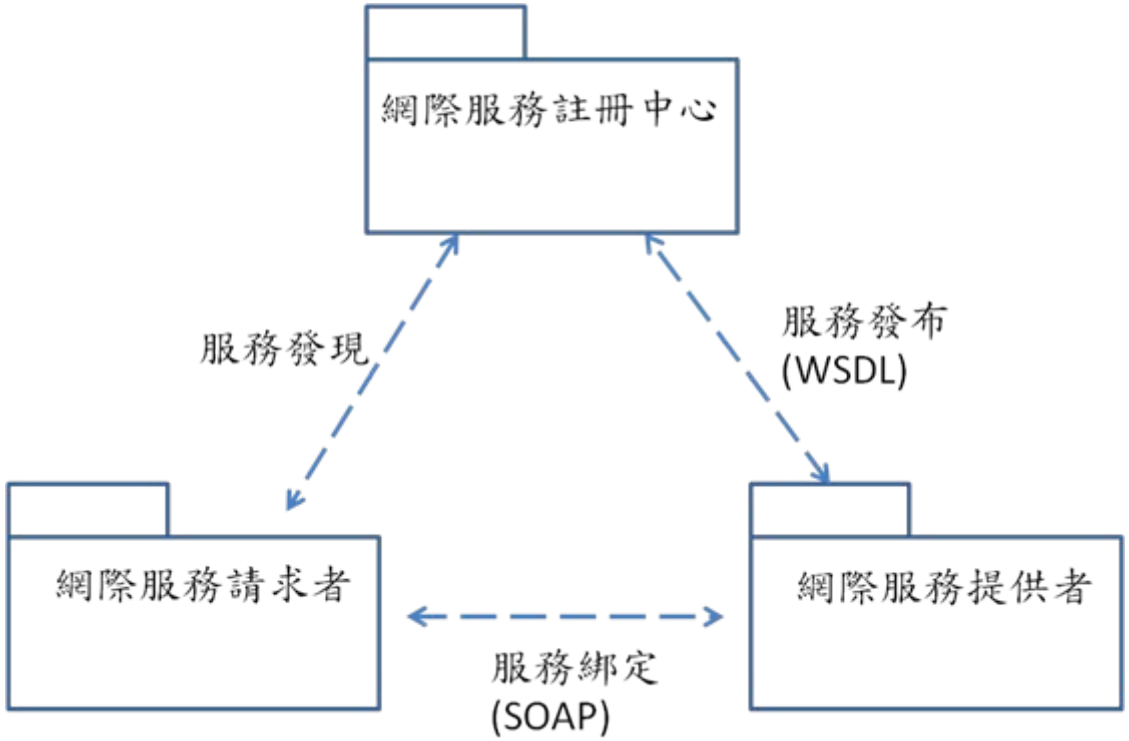


觀察員(Observer)設計樣式互動圖

當實體主題對象物件的資訊受到更動後，便開始主動通知其所屬的所有實體觀察員要執行資料更新，然後，各個實體觀察員再向主題對象物件取得目前最新的資料。

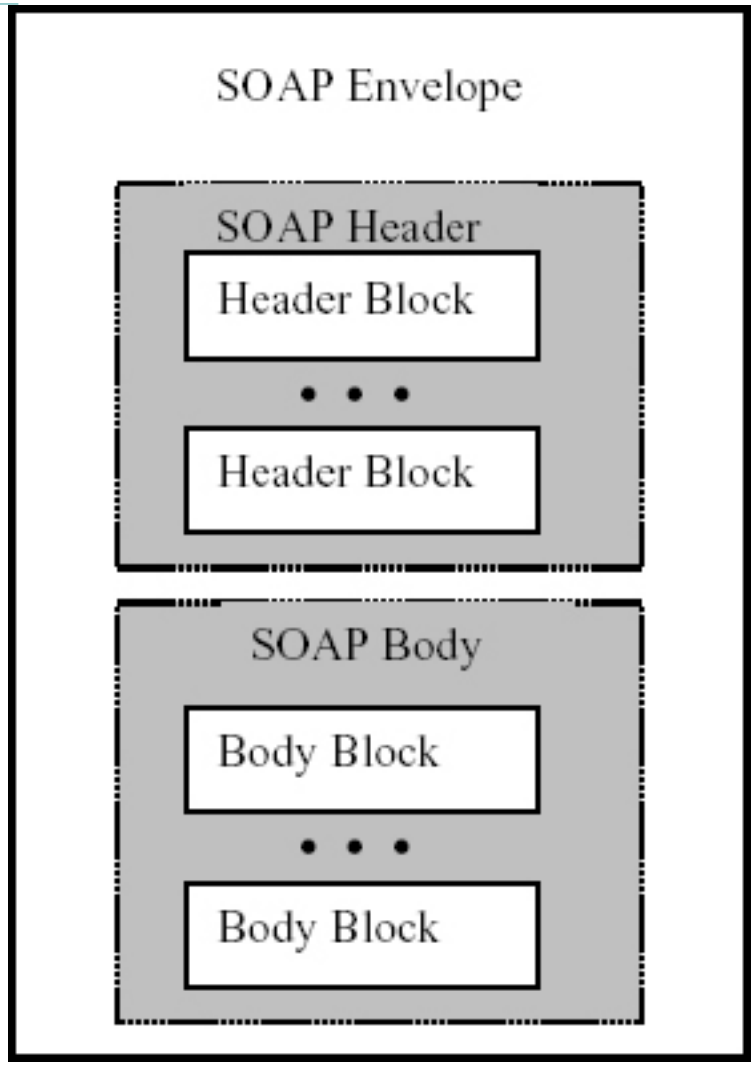
4

進階軟體設計：服務導向架構



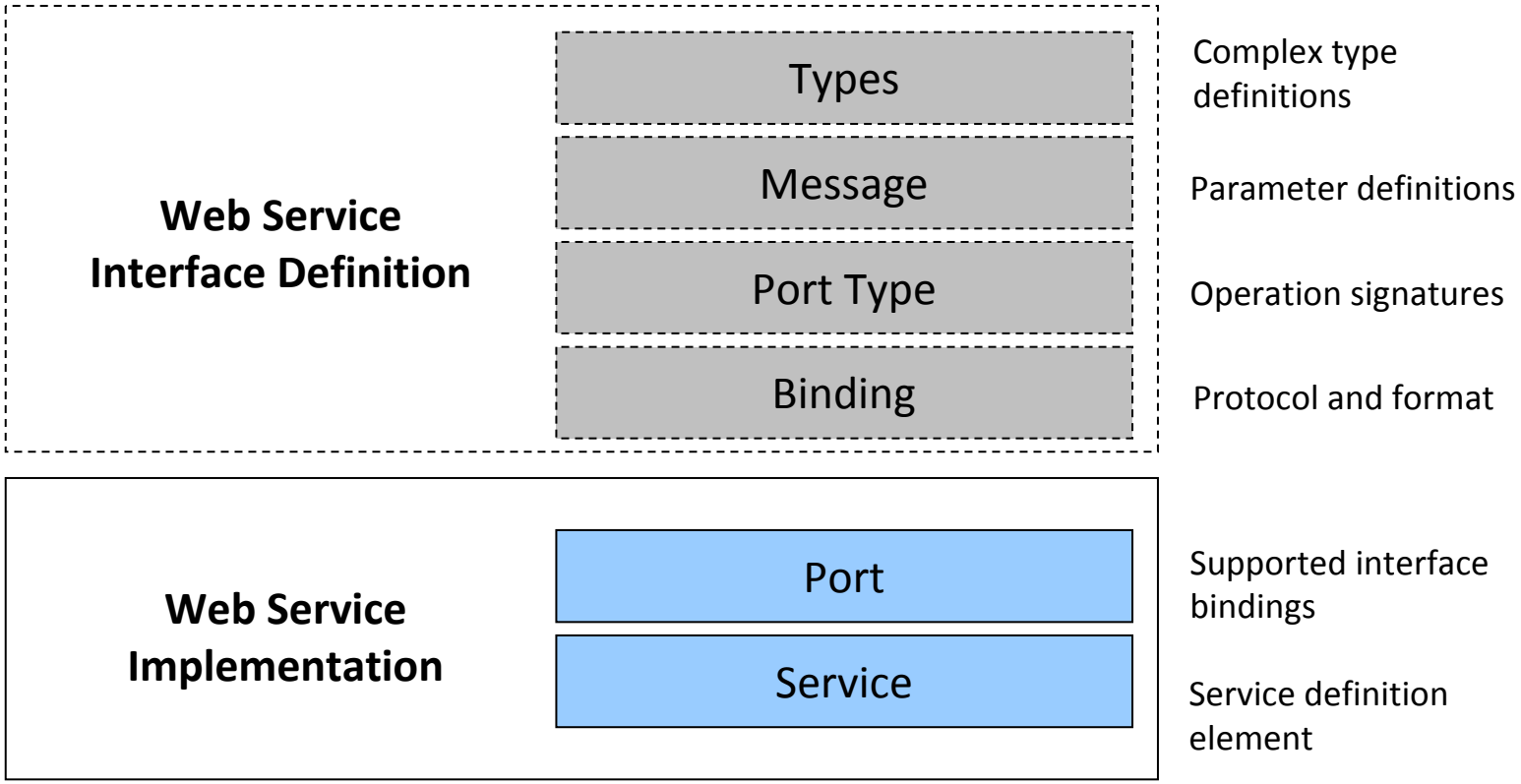
SOAP (Simple Object Access Protocol)

- SOAP提供了一種簡單的方式讓分散式架構中的個體，透過XML之標準格式來達成文件訊息交換與遠端方法呼叫(Remote Procedure Call, RPC)。
- SOAP主要包含三個部分：
  - SOAP envelope
  - SOAP encoding rules
  - SOAP RPC representation



4 WSDL (Web Services Description Language)

- WSDL是一種基於XML格式之描述語言，主要用來描述網際服務為有哪些方法(Operations)，而這些方法是用以文件為導向或是以程序為導向的方式交換訊息。

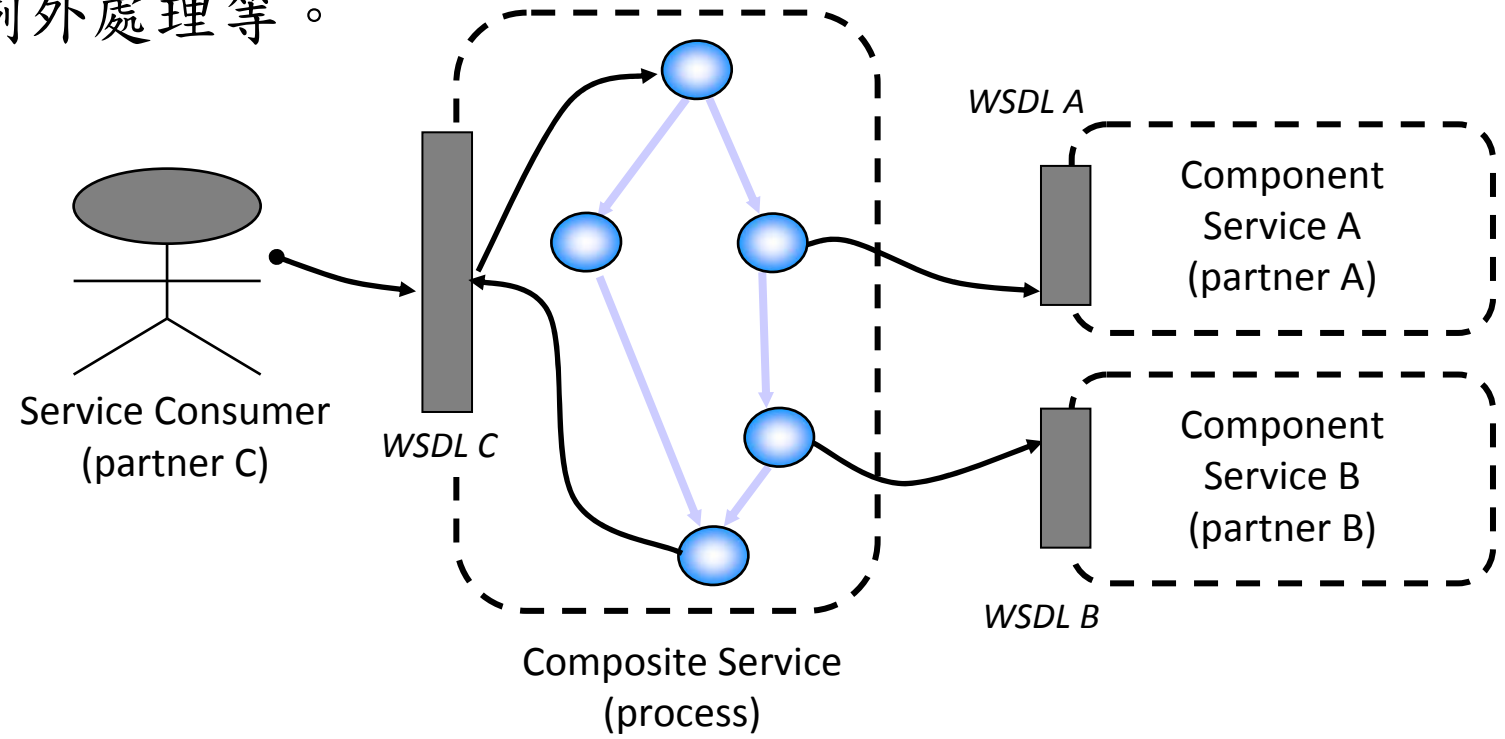


UDDI (Universal Description, Discovery and Integration)

- UDDI提供了實作一個服務仲介者(Service Broker)所需的應用程式介面(API)。
- 藉由UDDI服務仲介者，服務提供者能夠發布(Publish)他們提供的網際服務至登錄資料庫(Registry)上，而服務要求者可以經由服務仲介者的登錄資料庫尋找得到所需的服務。

BPEL4WS (Business Process Execution Language for Web Services)

- BPEL是專為服務組合機制所訂定的一個標準，透過流程的方式來串接多個外部服務。
- BPEL可描述服務流程內部的運作，包括流程控制、活動(Activity)執行、對外部服務的訊息交換、內部資料存取以及例外處理等。



本章總結

- 傳統上採用結構化分析設計方法，是以功能為導向的分析設計方式，包含控制流程圖與結構圖等作法。
- 目前普遍採用的物件導向設計方法，其以類別與物件為軟體元件的設計方式。特別的是，在這方法上有多位有豐富經驗的程式設計者及學者專家，將其設計經驗淬鍊、整理、包裝成設計樣式，提供軟體設計師一個迅速且有效率的解決方案
- 進階的服務導向架構(SOA)設計方法，可促進企業組織內、外部(如：內部應用程式、客戶、與供應商等)相關單位順利的溝通，以有效地達到企業目標。

作業練習 (Exercises)1

1. 請簡要說明軟體設計的幾個重要原則；討論在軟體設計的四個部分，如何運用這些軟體設計原則。
2. 請寫出三種架構樣式的特性，比較這些架構樣式的使用時機以及可能的優、缺點。
3. 「電腦販賣網站系統」中，顧客向一家賣電腦的網站購買電腦，顧客可以登入系統，瀏覽網站系統目前販賣各類型的電腦以及相關配備；顧客可以選擇標準配備或是客製化的電腦，配備自選。在確定購買時，顧客下訂單必須填寫送貨以及付款資訊，並可以利用網路查閱訂單的狀態，所購買的電腦連同收據發票會一起運送給顧客。請根據以上的需求，使用架構樣式設計軟體系統架構。

作業練習 (Exercises)2

4. 「簡易錄影帶出租系統」中，有顧客、出租店、錄影帶、影帶目錄、影片描述、租借合約、會員、租借金額等，功能有錄影帶出租、歸還錄影帶、根據合約付款、取得收據等，請使用功能導向設計方法設計此系統。
5. 若顧客有兩種，一類是公司客戶，另外一類為個人客戶，兩者可以一般化成為一個父類別——客戶。客戶有名稱、住址。個人客戶有信用卡帳號；公司客戶有信用狀態；客戶會下訂單買產品。請使用物件導向的類別圖設計以上的需求敘述。
6. 請以前述作業「電腦販賣網站系統」、和「簡易錄影帶出租系統」為例，撰寫出IEEE 1016軟體設計規劃書。

作業練習 (Exercises)3

7. 找一個設計樣式，使用設計樣式基本元素，描述此設計樣式。
8. 針對「電腦販賣網站系統」，使用目標導向軟體架構方法，設計軟體系統架構。
9. 請找出三個目前市面上已經應用的服務導向架構系統，描述這些系統的特性與優點，並說明他們為何可以稱為服務導向架構系統。