<<會計資訊系統課程講義>>

系統描述工具(授課精簡版)：

DFD(資料流程圖)、SF(系統流程圖)、PM(程序圖)、UML(統一塑模語言)

周國華

國立屏東大學會計學系

初版：2007/09/23

本次修訂：2024/02/17

第一部份

資料流程圖(DFD)

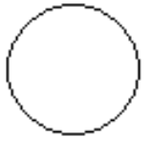
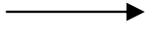
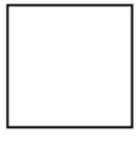
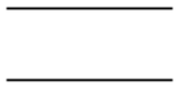
DFD：功能

- 資料流程圖(Data Flow Diagram, DFD)描述資料在系統內的子系統之間、系統與外部之間、組織內各部門之間、或組織與外部之間的流動情形，以及資料來源(source)、去向(destination)及儲存處(data store)。
- DFD是結構化系統分析及設計(SSAD)所使用的標準描述工具之一。在系統改用物件導向(OO)技術發展後，DFD仍是了解使用者需求的重要工具。

➢ DFD的F是 flow，通常翻譯成**資料流程圖**，或者**資料流向圖**，或更簡潔的**資料流圖**。

DFD：符號

- DFD用以下四種符號描述資料的流動：

	通稱為bubble，代表一個個體或程序。流入資料經此個體或程序處理後，轉換成流出資料。此圖形在DFD中代表正在描述之系統的全部或其中一部份。
	表示資料的流通路徑(資料流)，資料名稱會標示在邊上。
	表示資料的來源或去向，在DFD中代表正在描述之系統以外的其他系統或外部個體。
	代表儲存資料的地方(檔案或資料庫)。

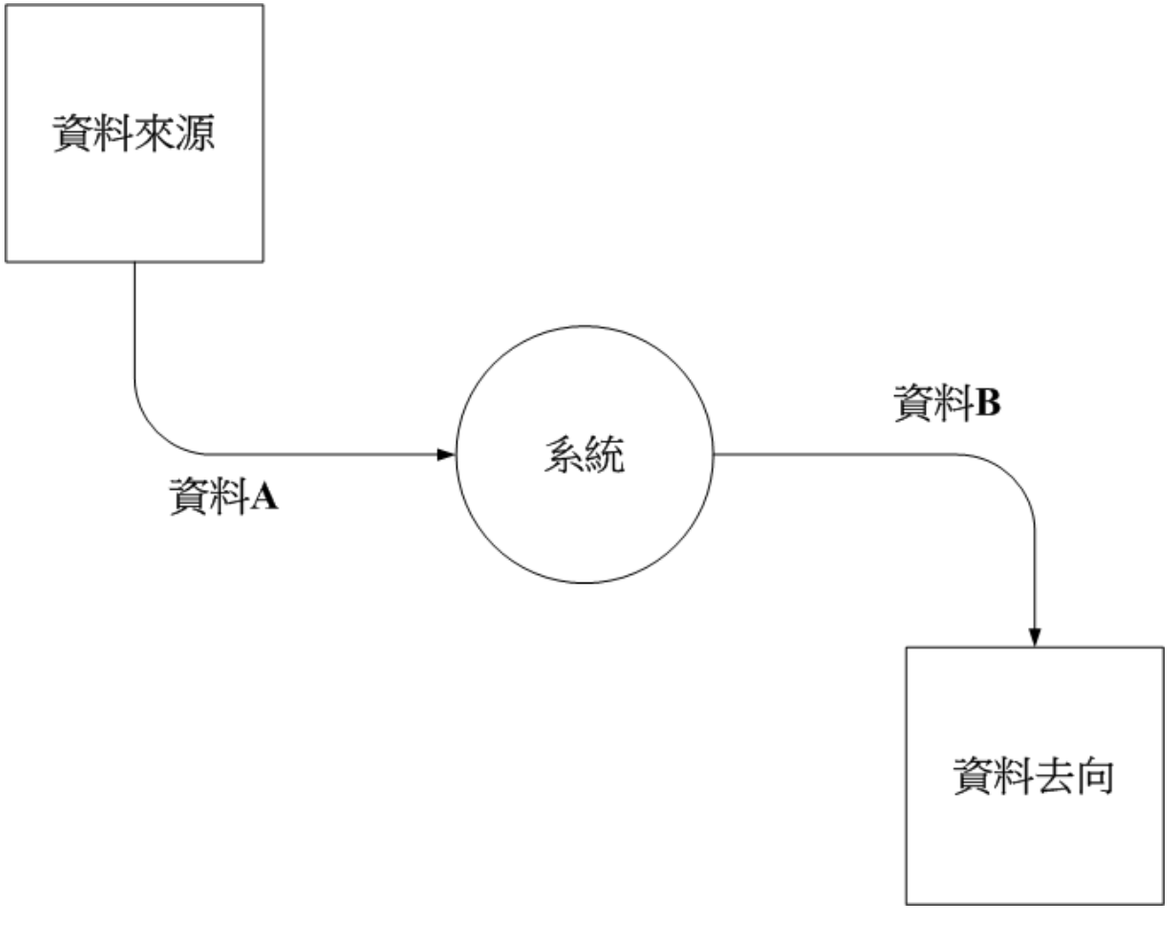
- 以上四種符號，在不同教科書或應用領域中常有不同的變異(例如: bubble用方形、矩形、六邊形或八邊形)，但基本概念則相同。

DFD：層次

- 按照描述的繁簡程度，DFD可分為以下幾個層級：
  - 背景圖(context diagram)：是DFD中最簡單、最上層、最抽象的圖，通常用一個bubble代表所描述的系統，再加上兩個方形符號表示系統之外的資料來源及去向。
  - 第0階(level 0)DFD：將上述單一bubble分解成1.0、2.0、3.0 …等數個子系統。
  - 再細分：將上述子系統進一步分解，例如系統1.0再細分為1.1、1.2、1.3...等子系統。

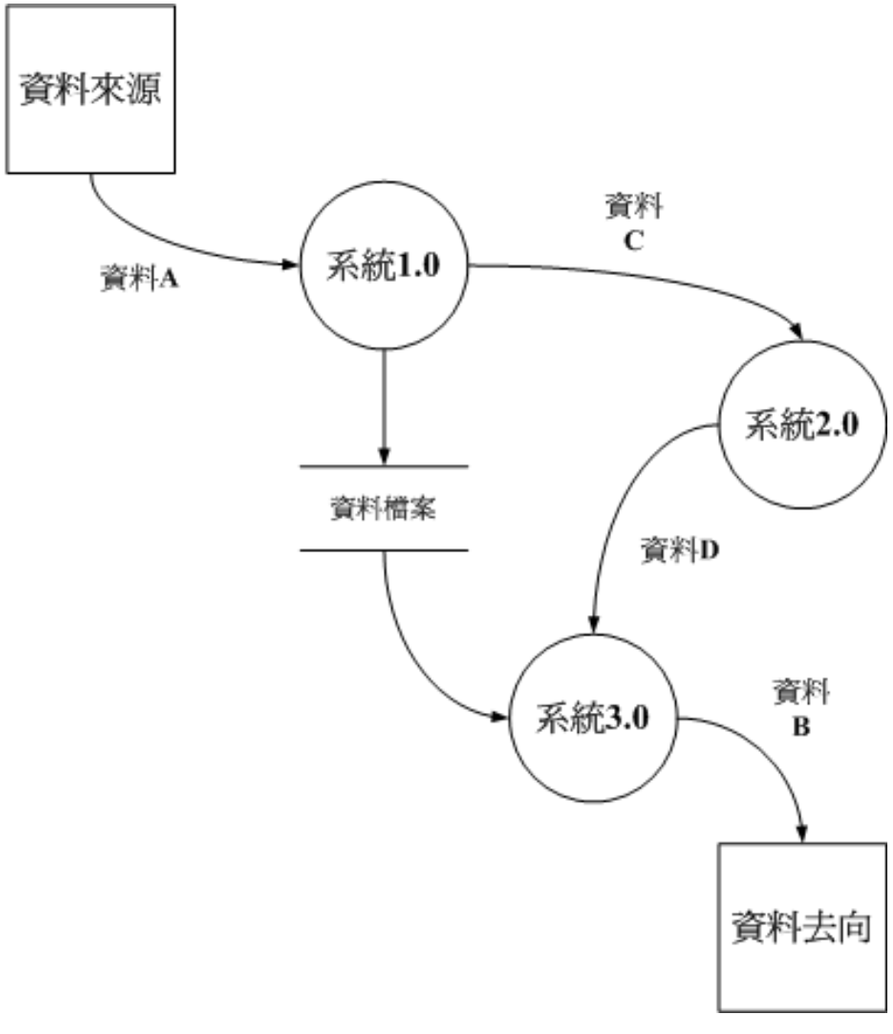
背景圖(Context Diagram)

- 背景圖的通用樣式如下：



第0階DFD圖

- 第0階DFD圖的通用樣式如下：



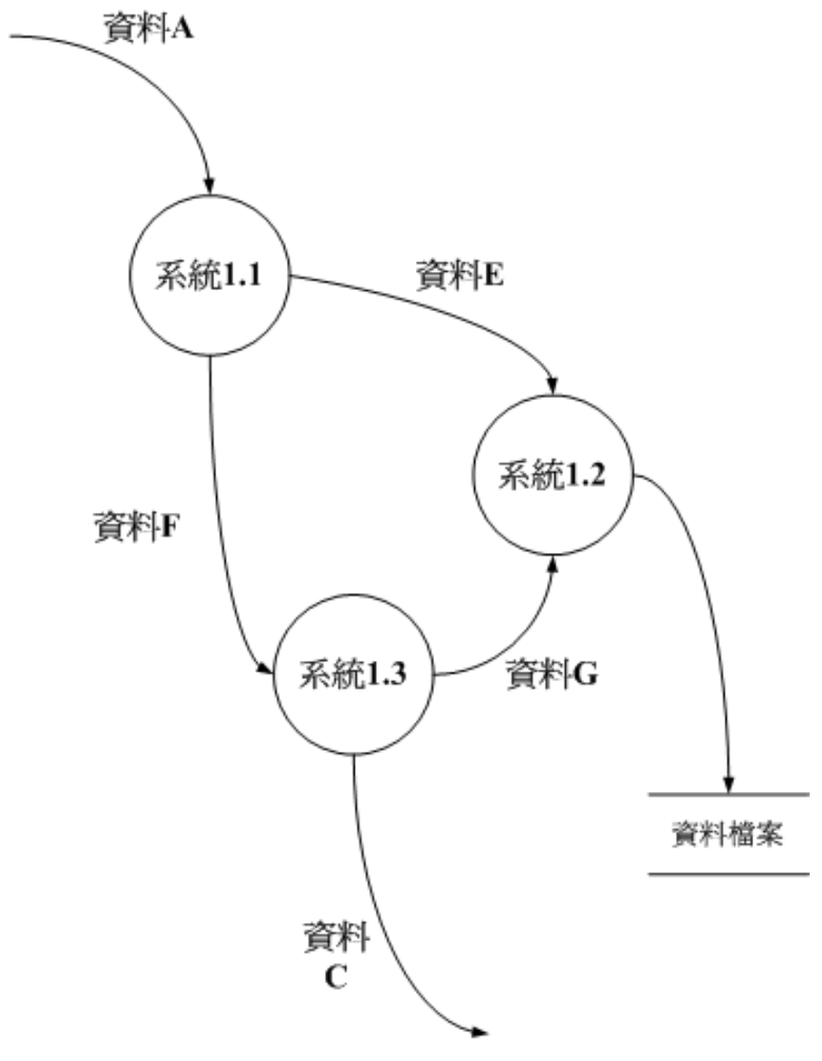
請注意：左圖的內容僅為例示，實際上的系統內容可能千變萬化。

再細分準則：流入流出一致

- 無論是將背景圖分解成第0階圖、或是將第0階圖做進一步細分，必須遵守「流入與流出上下層一致」原則。英文稱之為a set of balanced DFDs。
- 以前述二圖為例，背景圖所描述的系統有資料A流入、資料B流出；第0階圖也必須遵守資料A流入(圖例流入系統1.0)、資料B流出(圖例從系統3.0流出)。
- 若要進一步細分，則(依照圖例)系統1.0的子系統必須「一進二出」、系統2.0的子系統必須「一進一出」、系統3.0的子系統必須「二進一出」，且流入流出的資料名稱必須與上一層相同。

再細分範例

- 右圖為前述系統1.0再細分後之樣式：



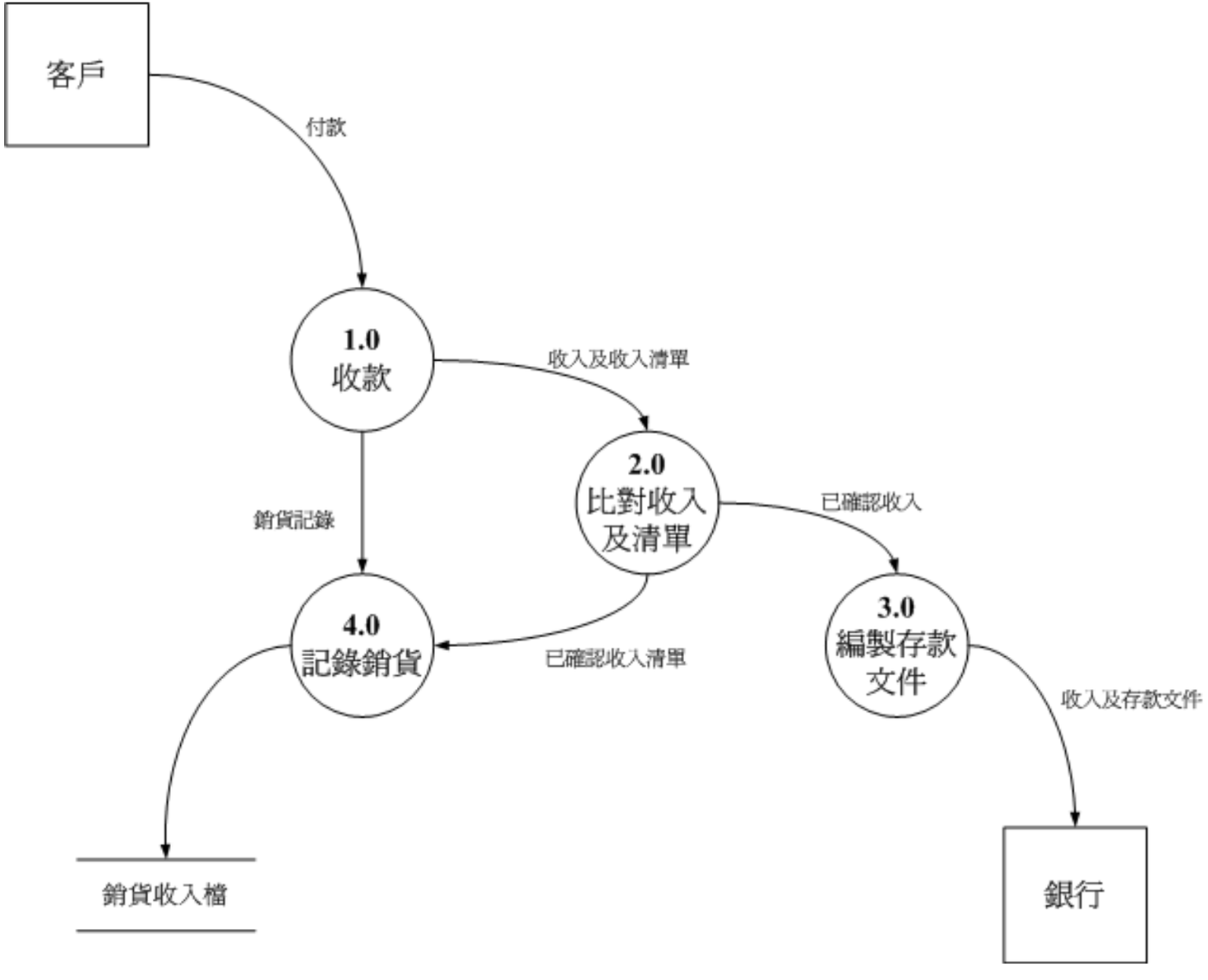
DFD：類型

- DFD可按資料及程序的描述方式分成兩種類型：
  - 實體資料流程圖(physical DFD)：此圖中，資料有具體的名稱；bubble是處理資料的人、地、物等個體(entity)，以名詞表示。
    - 實體DFD描述系統的基礎架構，可回答如何做(how, 物)、在哪做(where, 地)、誰來做(by whom, 人)等問題。
  - 邏輯資料流程圖(logical DFD)：此圖中，資料是泛稱；bubble代表處理資料的程序(process)，以動詞表示。
    - 邏輯DFD描述系統的各項作業，可回答做什麼(what)這項問題。

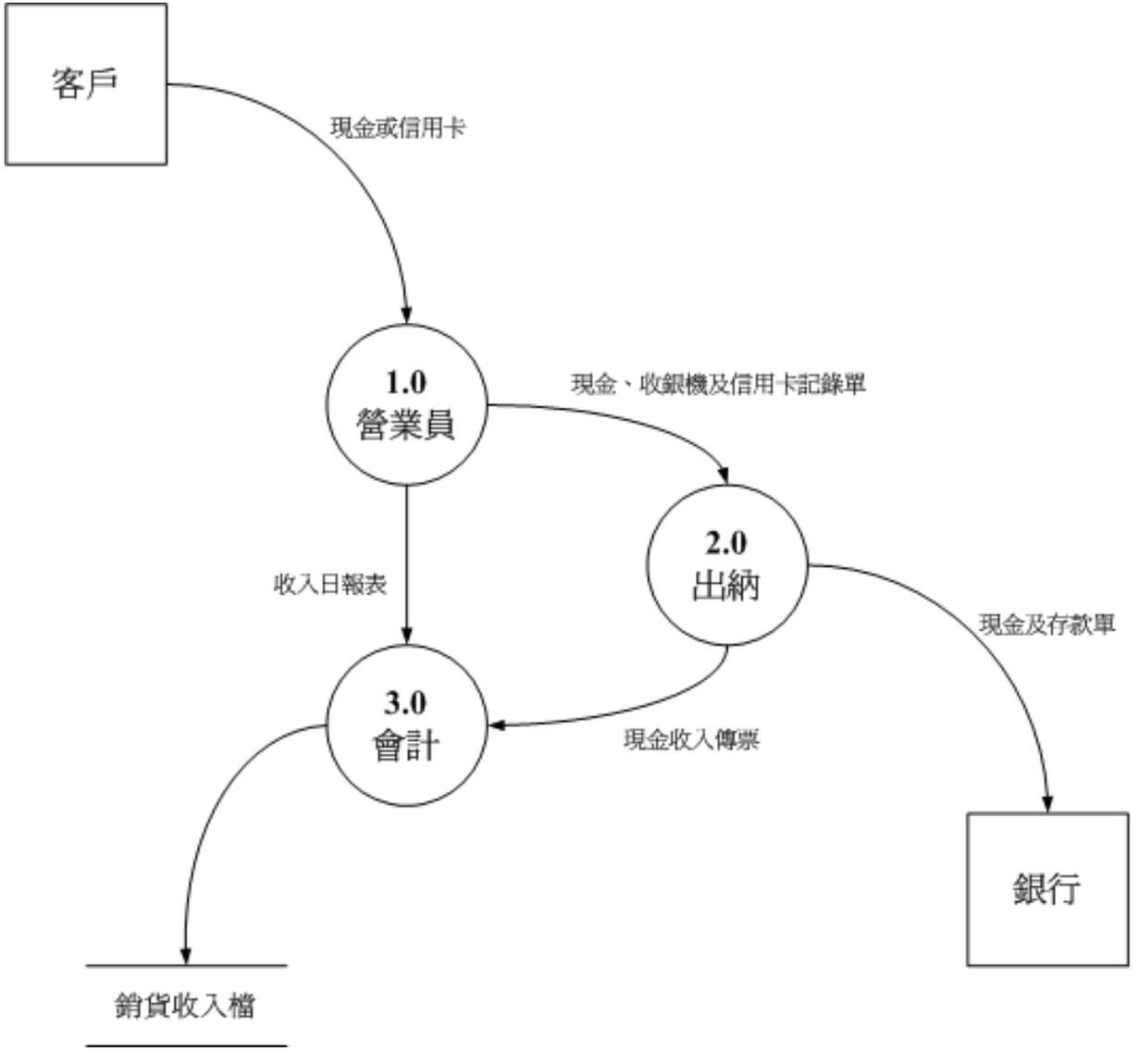
DFD：Logical vs. physical

- 長期而言，系統做什麼(what)的答案比較穩定，但系統如何做(how)、在哪做(where)、誰來做(by whom)的答案則會隨著時間及技術而改變。
- 在建置新系統時，通常會先繪製現有系統及新系統的logical DFD，以提供使用者新舊系統的比較資訊；然後再根據新系統的logical DFD，繪製physical DFD。

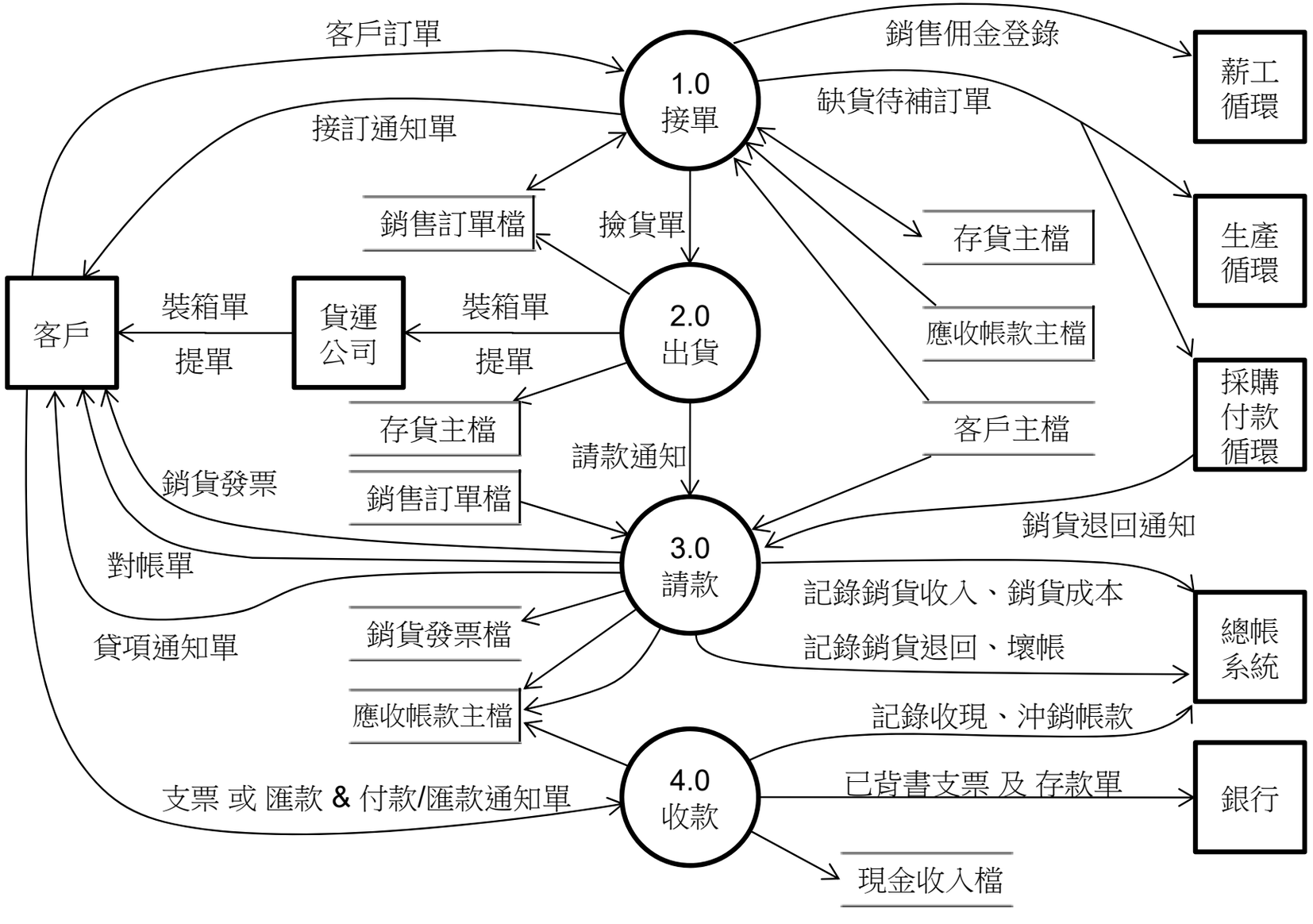
Logical DFD 範例：現銷系統



Physical DFD 範例：現銷系統



DFD範例 : 銷售及收款循環(第0階邏輯DFD)



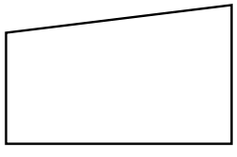
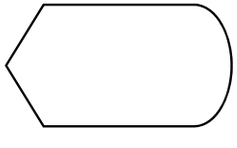
第二部分

系統流程圖

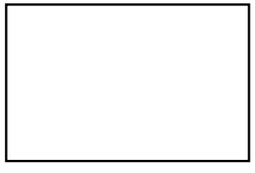
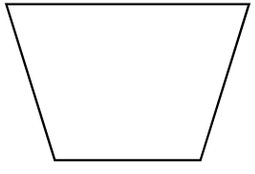
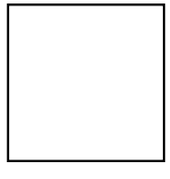
Flowchart： 三種類型

- 系統流程圖(system flowchart)：將特定企業程序(business process)的流程作完整表達，包含該程序的資訊程序(information process, 即 資料輸入、處理、儲存及輸出)及相關的營運程序(operations process, 即 人員、設備、組織及活動)。
- 程式流程圖(program flowchart)：表達資訊程序中的電腦程式邏輯。
- 文件流程圖(document flowchart)：將文件在人工化系統(manual system)內的流轉程序作完整表達。

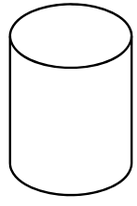
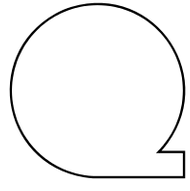
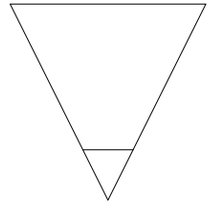
Flowchart 符號：輸入輸出

	表示輸入或輸出的文件或報告。
	人工輸入。
	打孔卡片，由讀卡機讀取。考試用答案卡是常見實例。
	通用輸入輸出符號，亦可代表**會計帳簿**。
	電腦螢幕，通常做為輸出符號；但具有觸控式功能的螢幕亦做為輸入符號。

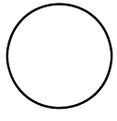
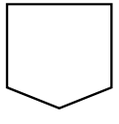
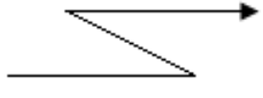
Flowchart 符號：處理

	代表電腦處理，包含資料查詢及檔案更新。
	代表人工處理。例如 編製文件、在文件上簽章等。
	透過電腦周邊設備所做的附帶性處理，例如 文件掃瞄、條碼及RFID讀取等。

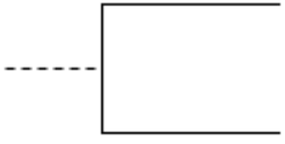
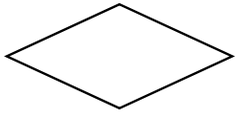
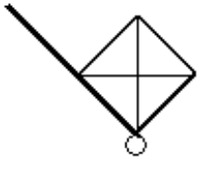
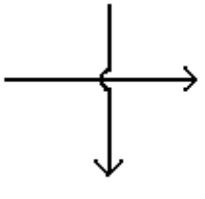
Flowchart 符號：儲存

	表示資料儲存在可直接存取資料(direct access)的媒體上，包含磁碟、光碟等。
	表示資料儲存在須循序(sequential)存取資料的媒體上，例如 磁帶。
	書面文件歸檔符號，其內上方註明文件或檔案名稱，下方三角形可書寫不同字母：N表示按編號歸檔，A表示按字母順序歸檔，C或D表示按日期歸檔。

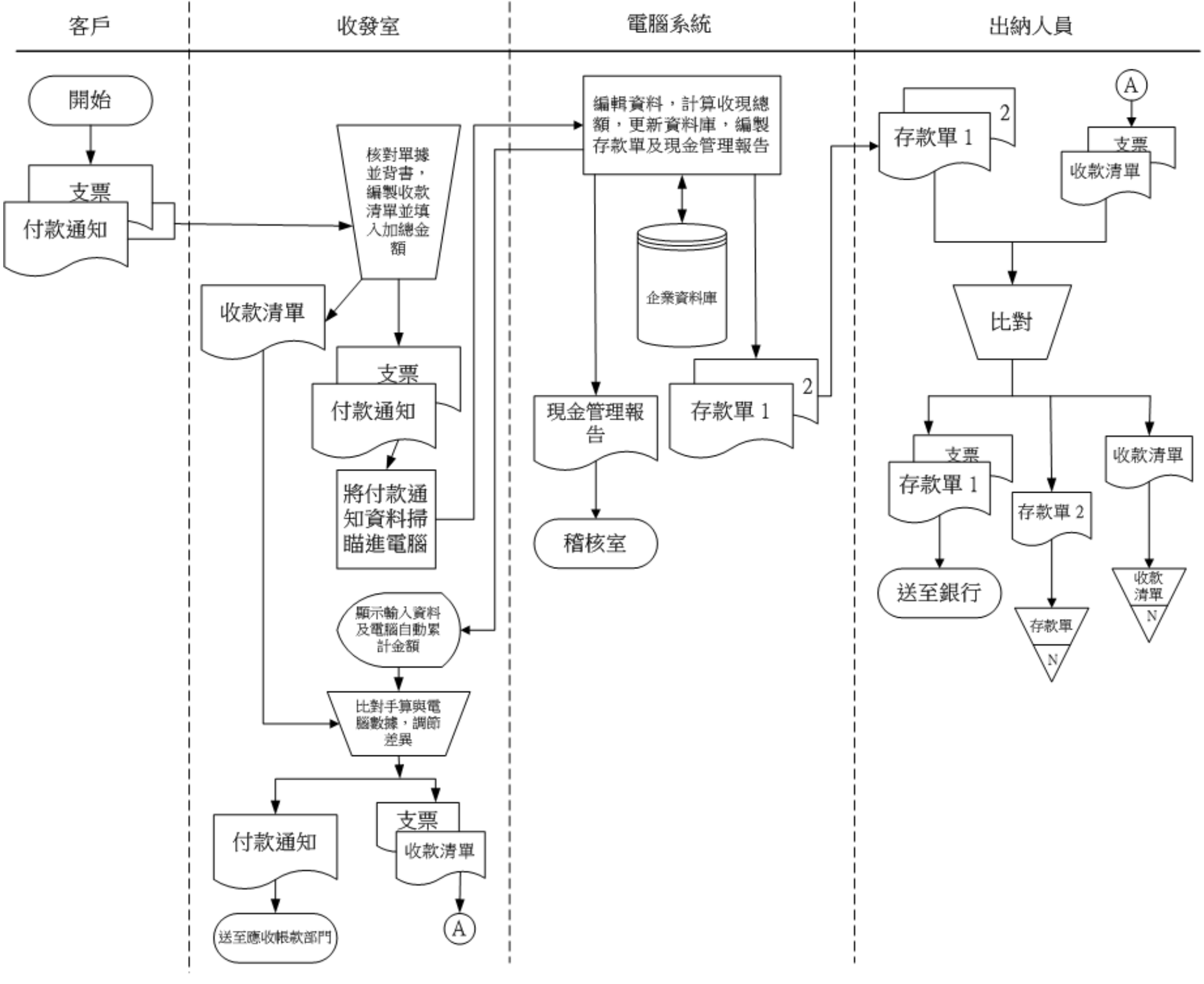
Flowchart 符號：連結

	代表流程的起點或終點，亦可代表外部個體。
	同頁連結，其內可書寫A、B、C…等字母。
	跨頁連結，其內可書寫頁碼及A、B、C…等字母。
	代表文件、實物或處理程序的流向。通常流向為向右、向下。
	通訊連結，代表以電話或電腦網路進行資料或訊息傳輸。

Flowchart 符號：其他

	文字註解，以虛線連結。
	批次總數、控制總數。
	決策點。
	貨物。
	流程穿越。

系統流程圖範例：收款程序



第三部分

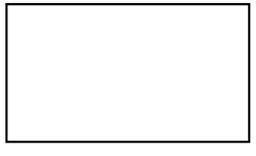
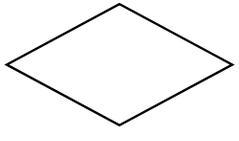
程序圖

程序圖：功能

- 程序圖(process map)能以相對簡單的符號完整表達企業程序(business process)的內涵。
- 在企業程序改變前後，最適合用程序圖以對照方式表達企業程序的現狀(as is)及改變後(could be)的新狀態。
- 常見應用領域：
  - 企業程序設計與分析(e.g., 會計師分析企業的內控程序)。
  - ERP系統導入前的企業流程再造(BPR)。
  - 六標準差(six sigma)管理模式的施行。

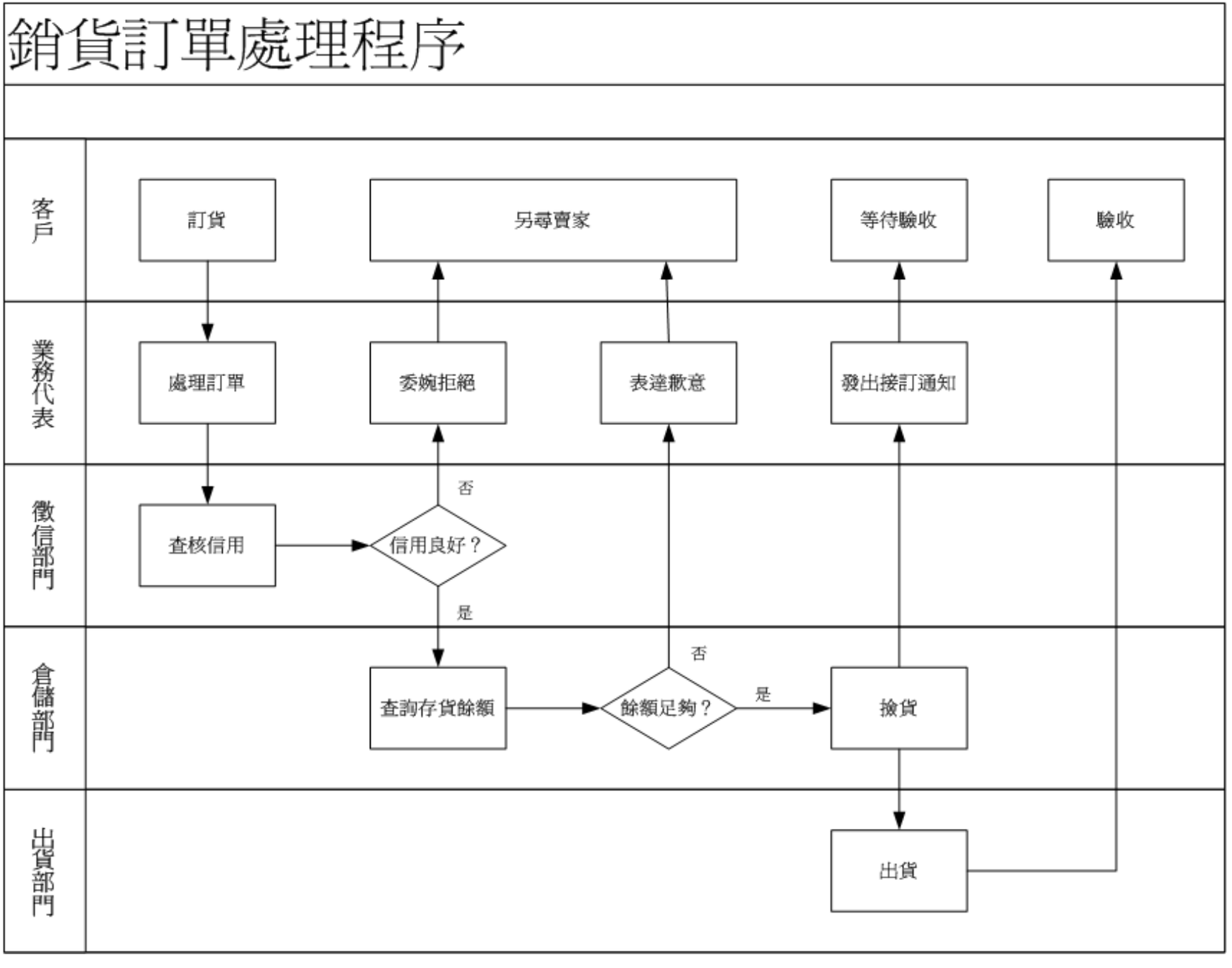
程序圖：符號

- 標準的程序圖，僅使用以下三種符號：

	代表特定程序。
	代表決策點。
	代表流程方向。

- 除了上述三種符號，許多使用單位會另外增添幾項常用符號，因此程序圖在實務上有相當大的變異性。

程序圖範例：接訂程序


銷貨訂單處理程序
客戶
業務代表
徵信部門
倉儲部門
出貨部門
訂貨
另尋賣家
等待驗收
驗收
處理訂單
委婉拒絕
表達歉意
發出接訂通知
查核信用
信用良好？
否
是
查詢存貨餘額
餘額足夠？
否
是
撿貨
出貨

第四部分

UML (統一塑模語言)

物件導向(OO)

- Object-oriented (OO)：應用程式(application)由可重複使用的軟體物件(object)或元件(component)組合而成。
  - 軟體物件可用來描述實體物件以及抽象概念。
  - 元件是由功能相關的物件組合而成。
- OOAD：物件導向分析(Analysis)與設計(Design)。
- OOP：物件導向程式設計(Programming)。
- OO語言：
  - 最早具有OO重要特色的語言：Simula (1967)
  - OO理論據以發展的語言：SmallTalk (1972~1980)
  - 目前主流OO語言：Java，Python，C++，C#，VB.NET，Ruby
  - iOS及Mac OS平台使用的OO語言：Swift，Objective-C
  - 極具親和力的3D教學用OO語言：Alice 2.2, 3.0

※ 中國大陸把OO翻譯為「面向對象」，OOP為「面向對象編程」。

UML與OO

- Unified Modeling Language (UML)：統一塑模語言，是物件導向分析與設計的標準工具語言，亦可用來描述企業程序。
  - UML 2.0並未完整支援data modeling，但class diagram可提供類似ERD的資料塑模功能。
- 塑模(modeling，或稱建模)：意指開發資訊系統時，必須先確認使用者需求，並將此需求以通用的圖形及語法建立成視覺化模型(visual model)，以便有效傳達給程式設計師。

UML 2.0的圖形名稱 2-1

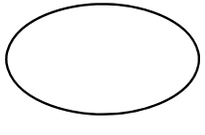
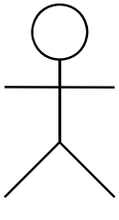
- UML 2.0將圖形分成三大類，共14種圖：
  - 結構圖形：**類別圖**，物件圖，元件圖，剖析圖，複合結構圖，佈署圖，套件圖。
  - 行為圖形：**使用案例圖**，**活動圖**，狀態機器圖。
  - 互動圖形：**順序圖**，溝通圖，計時圖，互動觀點圖。

* 本課程將介紹四種與需求分析及系統分析相關的圖形：類別圖、使用案例圖、活動圖及順序圖。

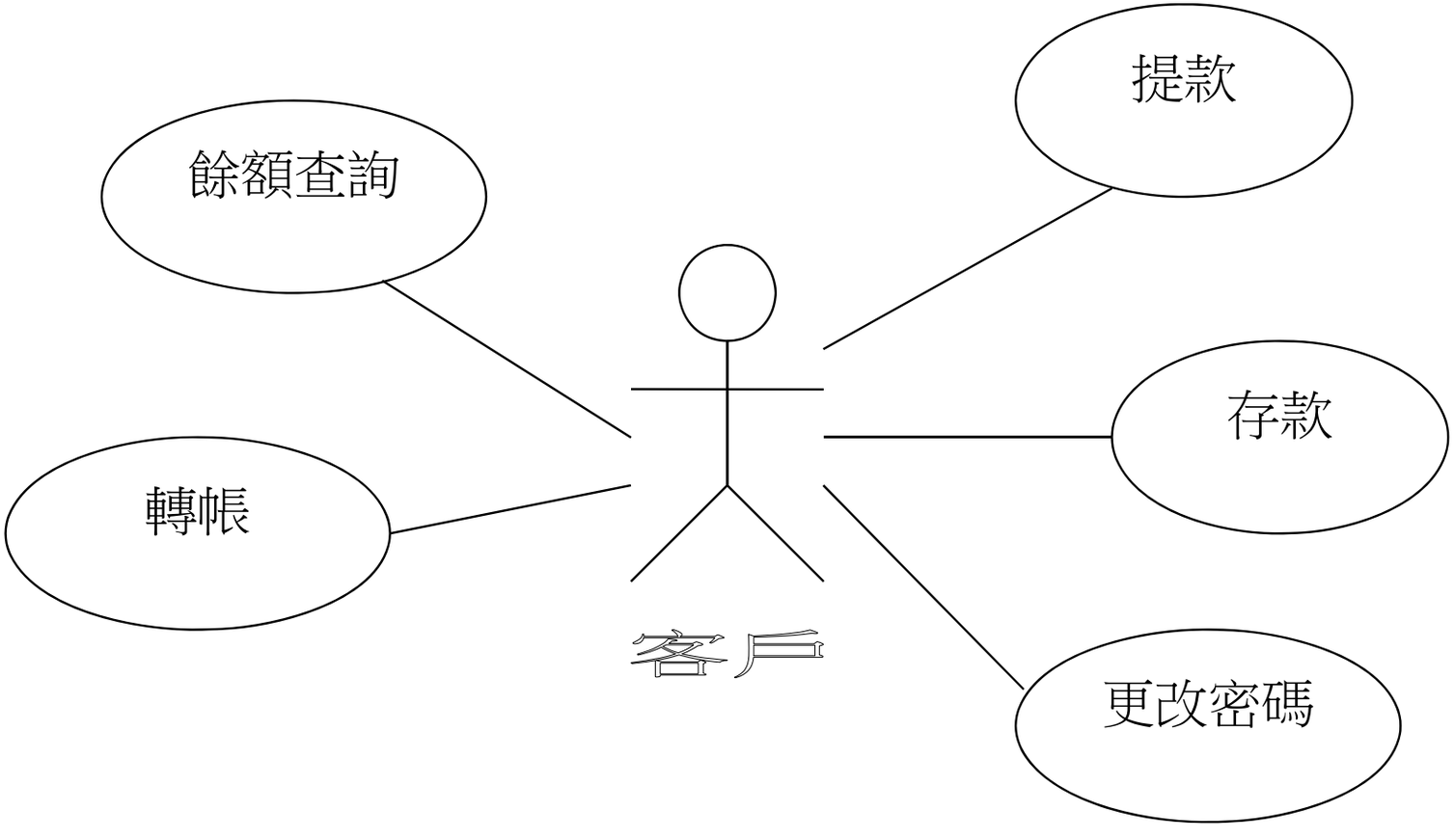
系統開發程序

1. 確認需求：Use Case Model
   - 從使用者取得完整的需求資料。
     - UML圖形工具：使用案例圖，活動圖。
2. 系統分析(OOA)：Conceptual Model or Analysis Model
   - 將需求資料轉成開發者觀點。
     - UML圖形工具：概念類別圖，系統順序圖。
3. 系統設計(OOD)
   - 將概念模型轉成可供特定程式語言實作的觀點。
     - UML圖形工具：設計類別圖，物件順序圖，溝通圖……
4. 程式設計(OOP)

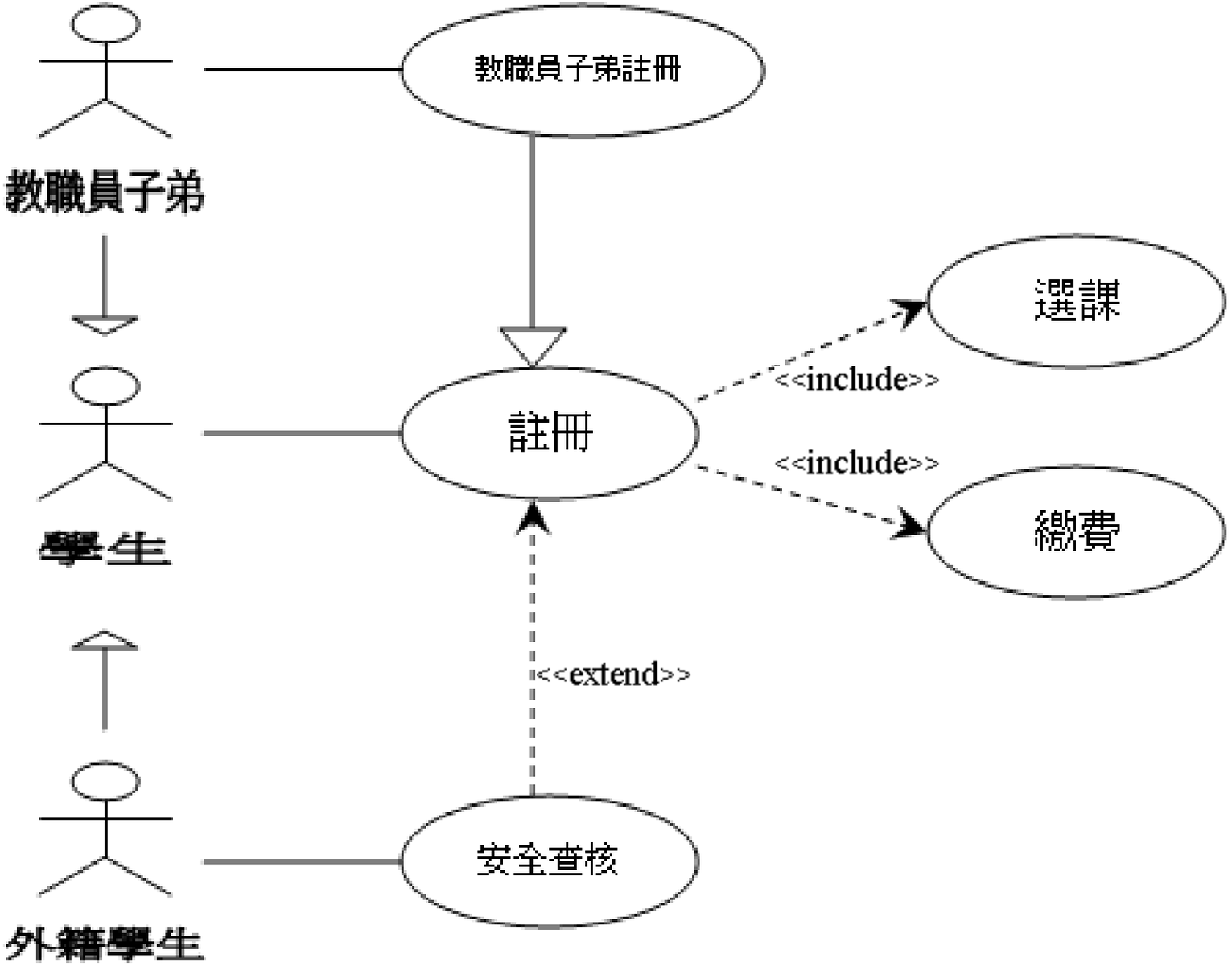
使用案例圖

- 使用案例圖(use case diagram, ucd)：此圖可表達使用者對系統功能的期待，每個use case代表使用者認定系統應提供的某項功能。與該系統互動的人(使用者、維護者)或其他系統，在ucd中稱為角色(actor)。
- 符號：
  - 使用案例： 
  - 角色： 

使用案例圖範例：ATM



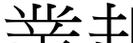
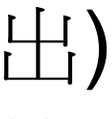
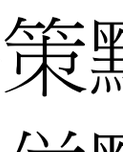
使用案例圖範例：註冊

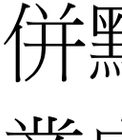


活動圖

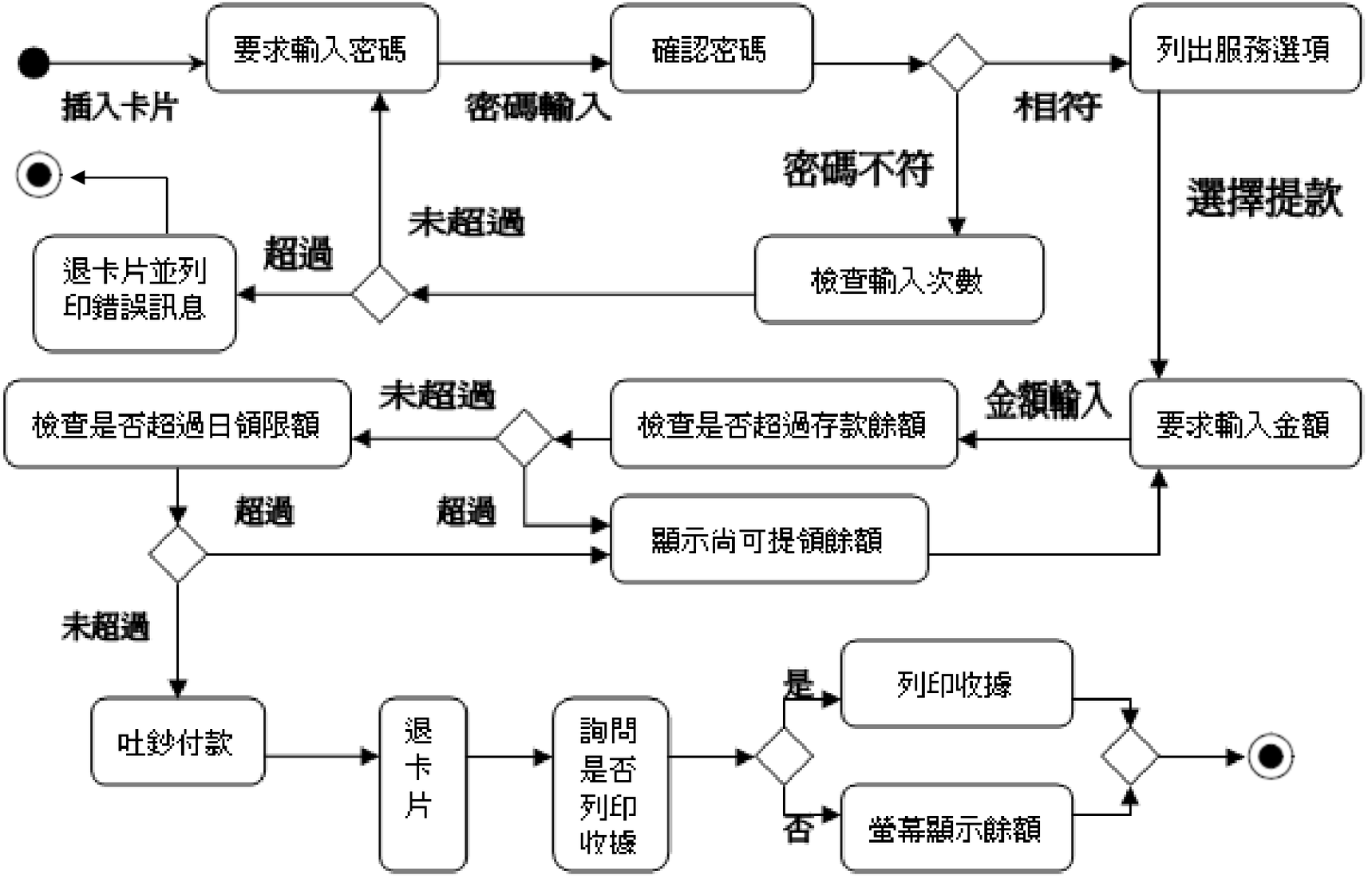
- 活動圖(activity diagram，或稱作業圖)：此圖可用來描述
  - 個別使用案例內的詳細作業流程。
  - 企業程序。
  - 企業規則的細節。
- 在傳統結構化分析中所使用的DFD及system flowchart，在OOAD中可用活動圖代替。

活動圖：符號

- 符號：
  - 作業起點： 
  - 作業終點(可有多個)： 
  - 分岔點(fork，一作業進、多作業同時出)： 
  - 會合點(join，上述多個平行作業同時進、一作業出)： 
  - 決策點(decision，一進、擇一出)： 

  - 合併點(merge，多進一出)： 
  - 作業內容： 

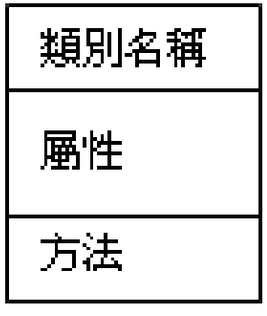
活動圖範例：提款

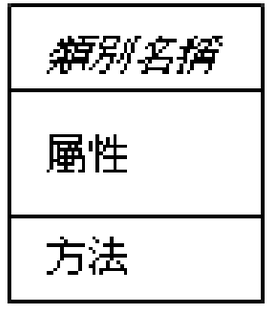

要求輸入密碼
確認密碼
列出服務選項
插入卡片
密碼輸入
相符
密碼不符
選擇提款
未超過
超過
退卡片並列印錯誤訊息
檢查輸入次數
檢查是否超過日領限額
未超過
檢查是否超過存款餘額
金額輸入
要求輸入金額
超過
超過
顯示尚可提領餘額
未超過
吐鈔付款
退卡片
詢問是否列印收據
是
列印收據
否
螢幕顯示餘額

類別圖

- 類別圖(class diagram)：此圖藉由描述系統內的各個類別以及類別之間的關係，以呈現系統結構。OO技術的核心是類別及其物件，故此圖是OOAD中最核心的圖形。
- OO內的每個類別包含名稱、屬性(即資料，包含變數及常數)及方法(即行為)，與傳統結構化系統將資料交由資料庫、行為交由應用程式處理的模式大不相同。
- 分類：
  - 概念類別圖：在系統分析(OOA)階段繪製的類別圖，不必考量特定技術內涵(e.g., Java or C++)，也不必考慮技術細節(可忽略屬性及方法)。此圖可用來塑模企業程序，形成概念模型(conceptual model)。
  - 設計類別圖：在系統設計(OOD)階段繪製的類別圖，必須將選定技術之細節包含在圖形內。

類別圖：符號

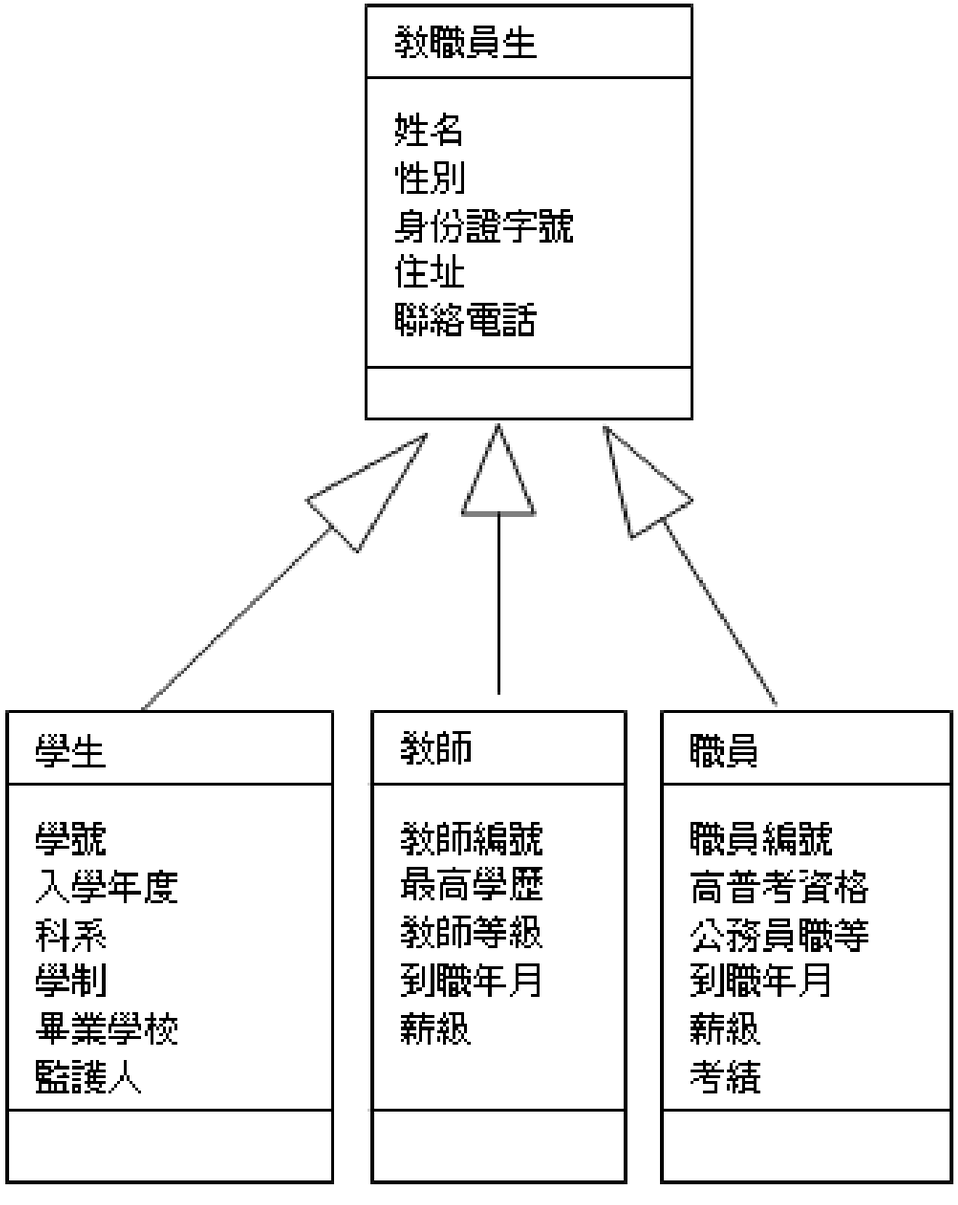
- 類別型態及符號：
  - 類別： 

抽象類別： 

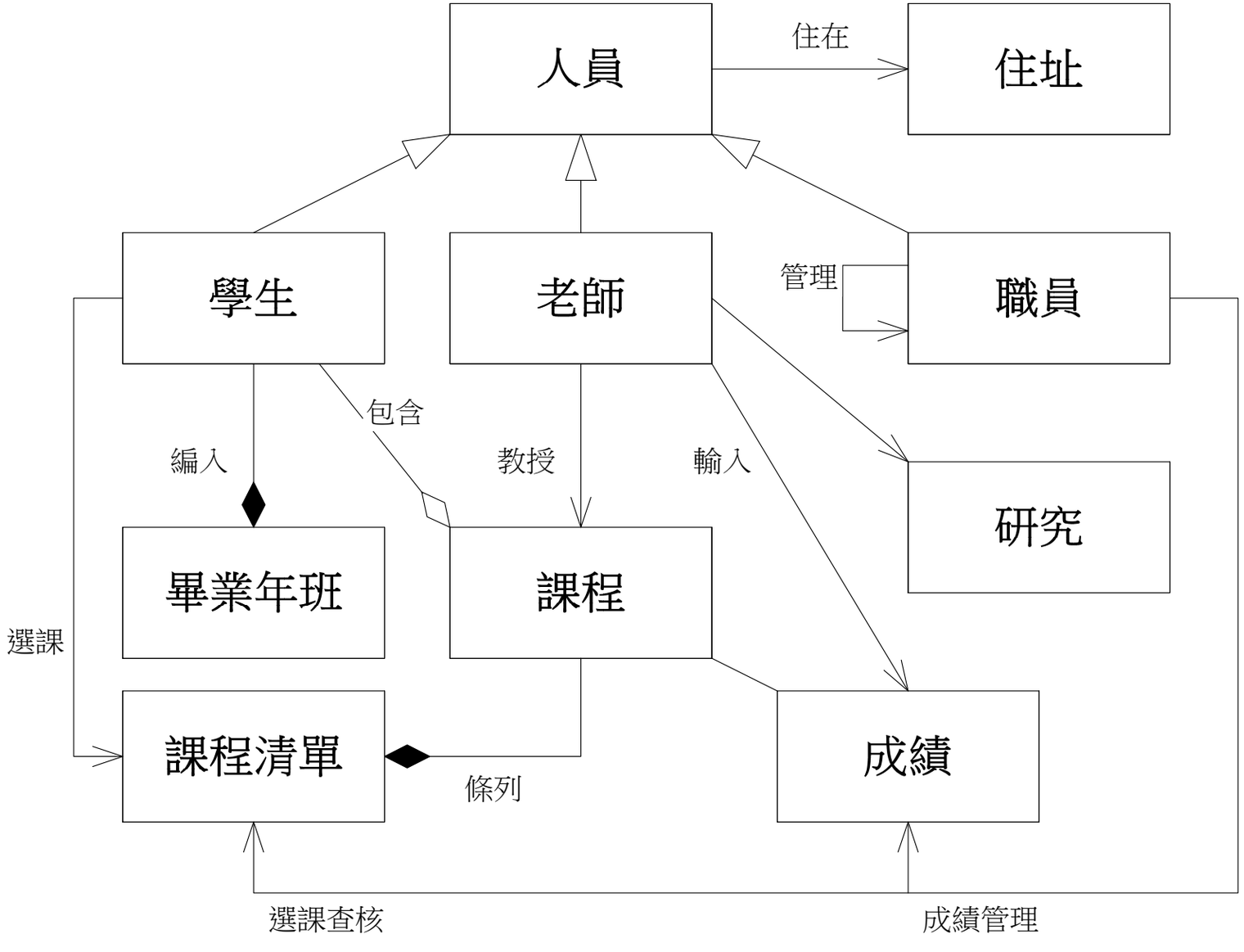
介面： 

    - 抽象類別意指其多個方法中至少有一個是抽象方法，介面的方法則都是抽象方法。(抽象方法：只有方法名稱而無實作內涵。)
  - 有時為了讓概念模型更簡潔，可將類別的屬性或方法省略。

類別圖範例：繼承


教職員生
姓名
性別
身份證字號
住址
聯絡電話
學生
學號
入學年度
科系
學制
畢業學校
監護人
教師
教師編號
最高學歷
教師等級
到職年月
薪級
職員
職員編號
高普考資格
公務員職等
到職年月
薪級
考績

類別圖範例：教學管理

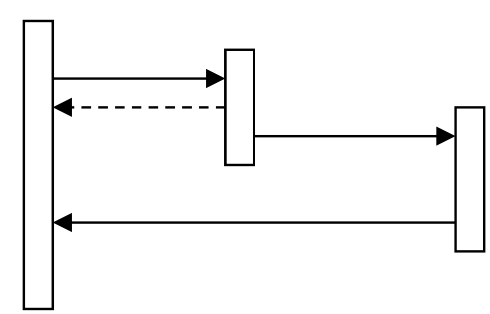


順序圖

- 順序圖(sequence diagram)：依時間順序描述系統內部各成員之間的互動，通常可分成兩大類：
  - 描述使用情境：主要用於系統分析(OOA)階段，著重在角色(actor)與系統之間的互動，將系統當成一個黑盒子，通稱為系統順序圖。
  - 描述方法邏輯：主要用於系統設計(OOD)階段，著重在物件之間的訊息傳遞，通稱為物件順序圖。

順序圖：符號

- 符號：
  - 物件：【<u>物件名稱：類別名稱</u>】，或【<u>：類別名稱</u>】
  - 生命線(lifeline)：由上到下的虛線，代表物件、角色或系統的存活時間。
  - 訊息線：——→ 或 ←——
  - 回應線：- - - -→ 或 ←- - - -
  - 促動盒：位於物件生命線上的長條矩形，代表訊息已發出， 並由接收物件進行處理中。



系統順序圖範例：新生報到