第五章 結構化分析與設計—流程塑模

內容大綱

學習目標

5.1 導論

5.2 結構化分析與設計評估準則

5.3 資料流程圖建構策略

5.4 資料流程圖建構指南

5.5 資料流程圖的評估

5.6 資料流程圖轉結構圖與模組設計

5.7 結論

學習目標

詳讀本章，你至少能瞭解：

- 系統分析與設計之評估準則。
- 資料流程圖建構策略與指南。
- 如何描述處理規格。
- 如何將資料流程圖轉成結構圖。

5.1 導論

- 結構化之分析與設計將所面對問題之流程與資料分開處理，並分別稱為流程塑模與資料塑模。本章先介紹流程塑模。

- 流程塑模主要是以資料流程圖做為塑模之工具，將流程分解成具層級結構之模組。

5.2 結構化分析與設計評估準則

- 良好的結構化設計有三個特徵：
  (1)模組間有很好的分割
  (2)階層式的系統架構
  (3)獨立的模組功能
- 要達到良好的系統設計與提升模組的品質，需考慮：
  (1)模組間的耦合力，是指一個系統內部各模組之間的相關程度。
  (2)模組的內聚力，是指一個模組內部所做事情之相關程度。
  (3)其他的因素，如功能分割等。

5.2.1 內聚力

- 內聚力是一種衡量模組內部之工作相關程度之方法。換句話說，模組的內聚力是衡量模組完成一件單一，且定義清楚之工作的程度。內聚力的種類大概可分為七種：
  - 功能內聚力(Functional Cohesion)
  - 順序內聚力(Sequential Cohesion)
  - 溝通內聚力(Communication Cohesion)
  - 暫時內聚力(Temporal Cohesion)
  - 程序內聚力(Procedural Cohesion)
  - 邏輯內聚力(Logical Cohesion)
  - 偶發內聚力(Coincidental Cohesion)等七種。

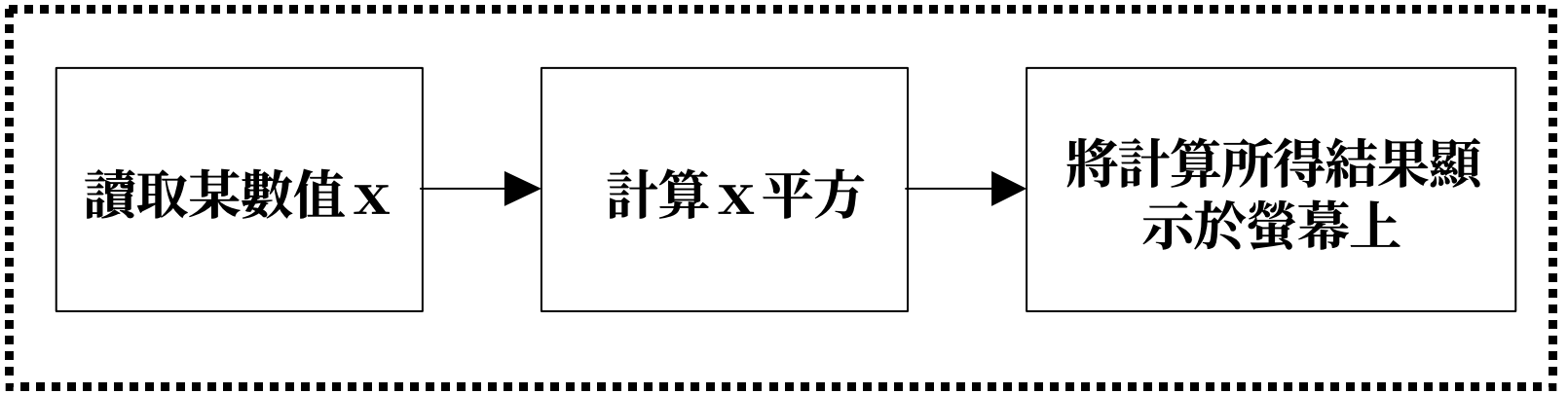
5.2.1 內聚力 (c.2)

- **功能內聚力**
  - **功能內聚力指的是當一個模組只做一件事情，亦即具有唯一的功能，是為功能型的內聚力。例如：**

檢查身分證號碼正確性	**以異動檔更新庫存主檔**	**計算營業稅**

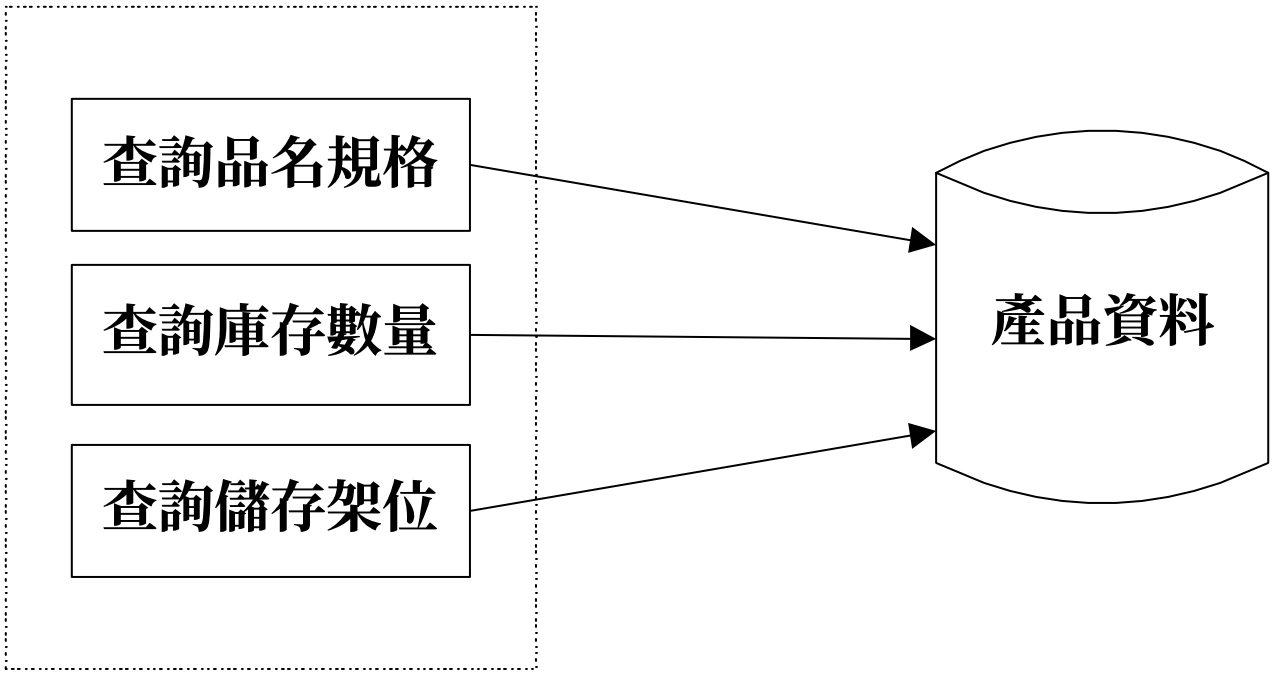
5.2.1 內聚力 (c.3)

- 順序內聚力
  - 順序內聚力是指模組內具有多個功能或處理多件事情，且一項功能的輸出立即成為下一個功能的輸入，也就是共用相同資料，則此模組具有順序內聚力。



5.2.1 內聚力 (c.4)

- **溝通內聚力**
  - **溝通內聚力是指模組內具有多個功能或處理多件事情，且這些功能使用相同的資料（輸入），但它們的執行順序沒有相關性。**

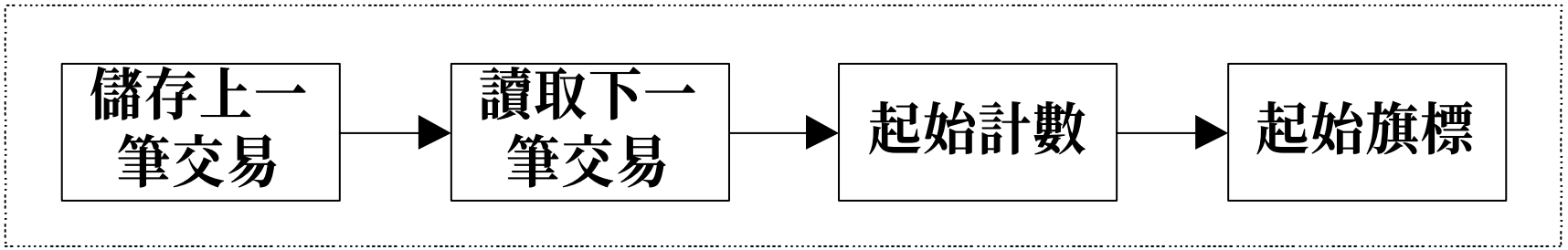


5.2.1 內聚力 (c.5)

- **暫時內聚力**
  - **模組內具有多個功能或處理多件事情，但是這些功能僅僅在時序上有所關連，也就是必須在同一時間內執行完成，所以這種模組具有暫時內聚力。**

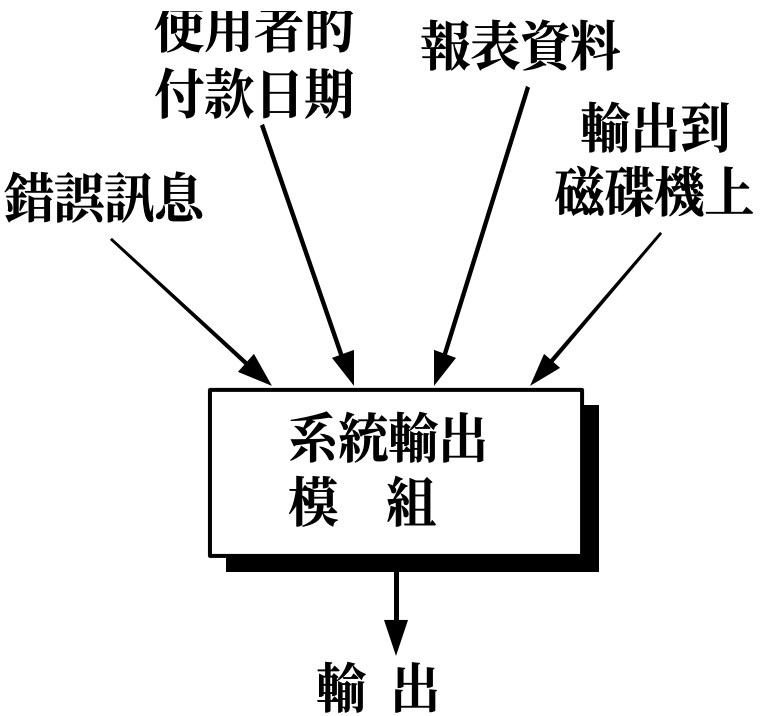
5.2.1 內聚力 (c.6)

- **程序內聚力**
  - **程序內聚力是指模組內具有多個功能或處理多件事情，這些功能必須按照一定的順序來執行，且不共用資料，這些功能群集在一個模組內僅為了確保它們的執行順序，則這模組具有程序內聚力。**



5.2.1 內聚力 (c.7)

- **邏輯內聚力**
  - **邏輯內聚力是指模組內具有多個邏輯上相關聯的功能。**



5.2.1 內聚力 (c.8)

- **偶發內聚力**
  - **若一個模組內部要做好幾件工作，且每一件工作都不相干，則該模組具有偶發內聚力。在設計時，偶發內聚力應盡量避免，例如可將個別的工作分別獨立出來自成一個模組，使各模組具有功能內聚力。**

圖5-8 模組內聚力之判定決策樹

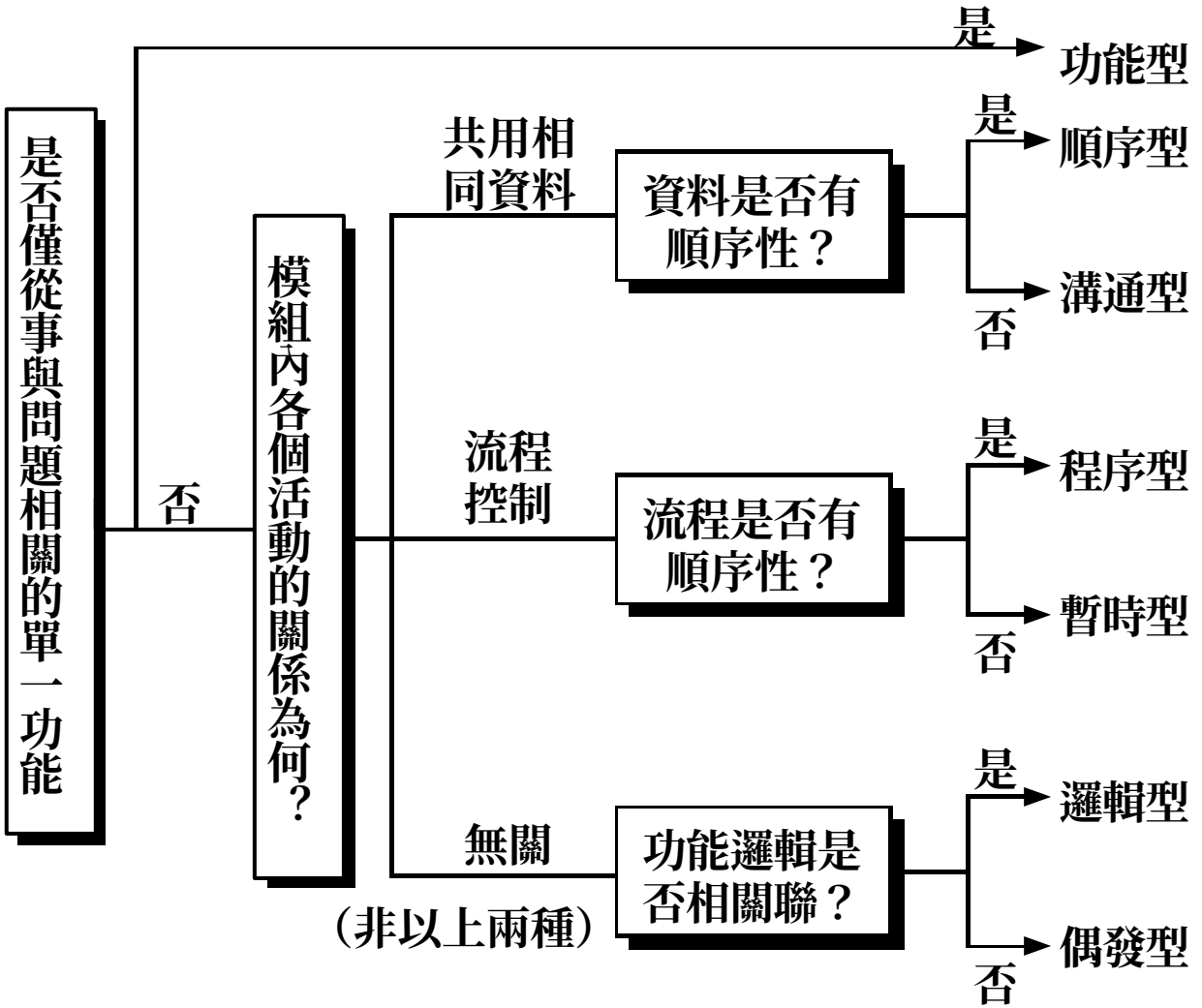


表5-1 內聚力之評比因素與結果

內聚力種類	耦合力情形	模組撰寫難易	與其他程式之共用性	維護性	瞭解性
功能型	好	好	好	好	好
順序型	好	好	中等	好	好
溝通型	中等	好	差	中等	中等
程序型	變動	中等	差	變動	變動
暫時型	差	中等	很差	中等	中等
邏輯型	很差	很差	很差	很差	差
偶發型	很差	差	很差	很差	很差

5.2.2 耦合力

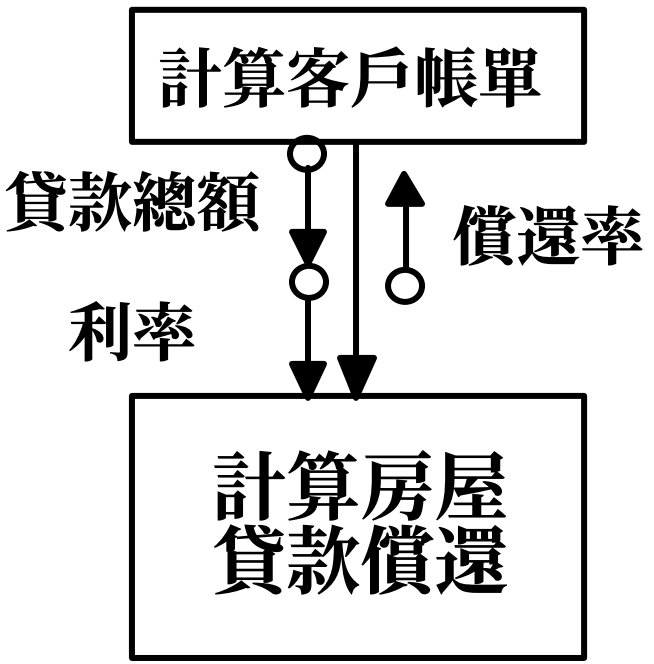
- **耦合力是一種衡量模組間相互關聯強度的方法。**
- **當解決了一模組內的錯誤狀況，而在其他的模組內引起了新的錯誤，這種現象稱為連鎖反應**(Ripple Effect)。
- **解決連鎖反應之可行方法是盡量使一個模組不與其它模組糾結在一起，即讓每個模組盡量的獨立。**

5.2.2 耦合力 (c.2)

- **耦合力可分為五類：**
  - **資料耦合力**(Data Coupling)
  - **資料結構耦合力**(Stamp Coupling)
  - **控制耦合力**(Control Coupling)
  - **共同耦合力**(Common Coupling)
  - **內容耦合力**(Content Coupling)。

5.2.2 耦合力 (c.3)

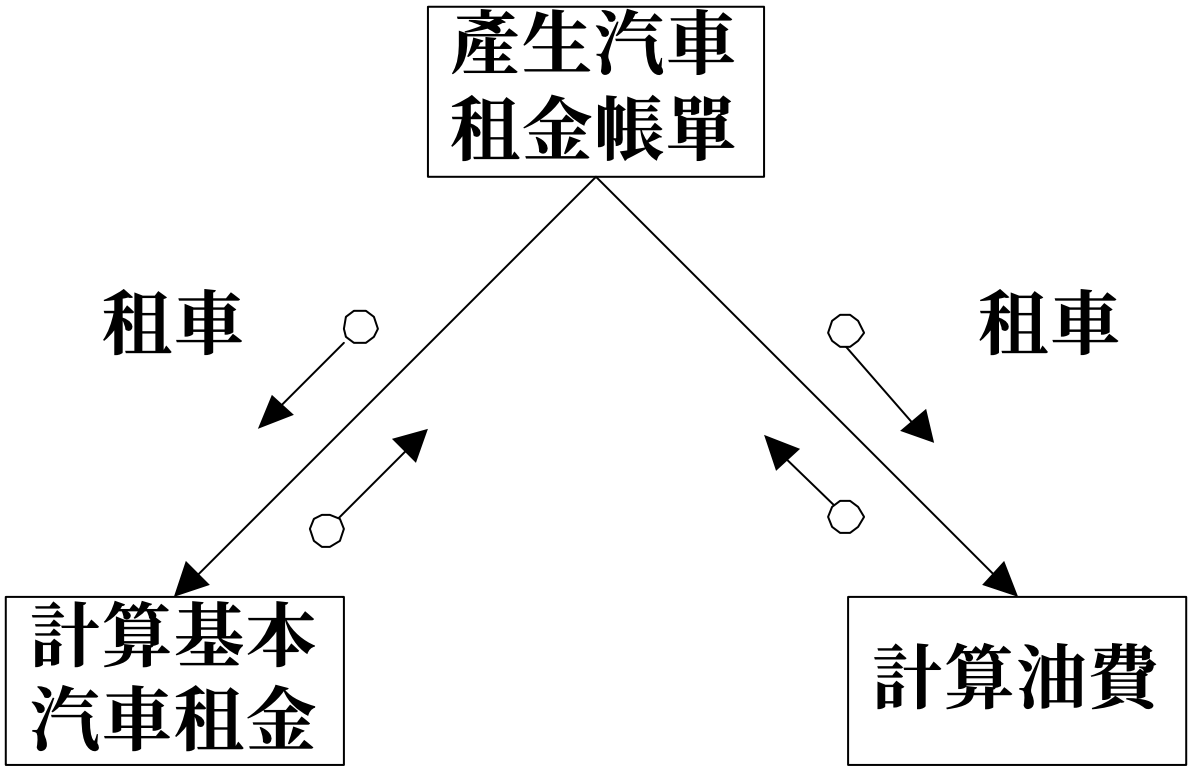
- **資料耦合力**
  - **資料耦合力是指模組間，如果使用一些簡單型別資料作為兩模組間傳遞之參數，則稱此模組間具有資料耦合力。**



5.2.2 耦合力 (c.4)

- **資料結構耦合力**
  - 資料結構耦合力是指模組間以資料結構(Data Structure)型別來做程式的介面，但並非每個模組均用到該資料結構之所有欄位。
  - 例如有一個資料結構稱為“租車”，該資料結構有六個欄位：牌照號碼、會員證號碼、使用汽油量、汽車型式、已開公里數與租借天數等。若這三個模組間是以“租車”之資料結構做為程式的介面(如圖 5-10)，則這些模組間具有資料結構耦合力。

圖5-10 資料結構耦合力



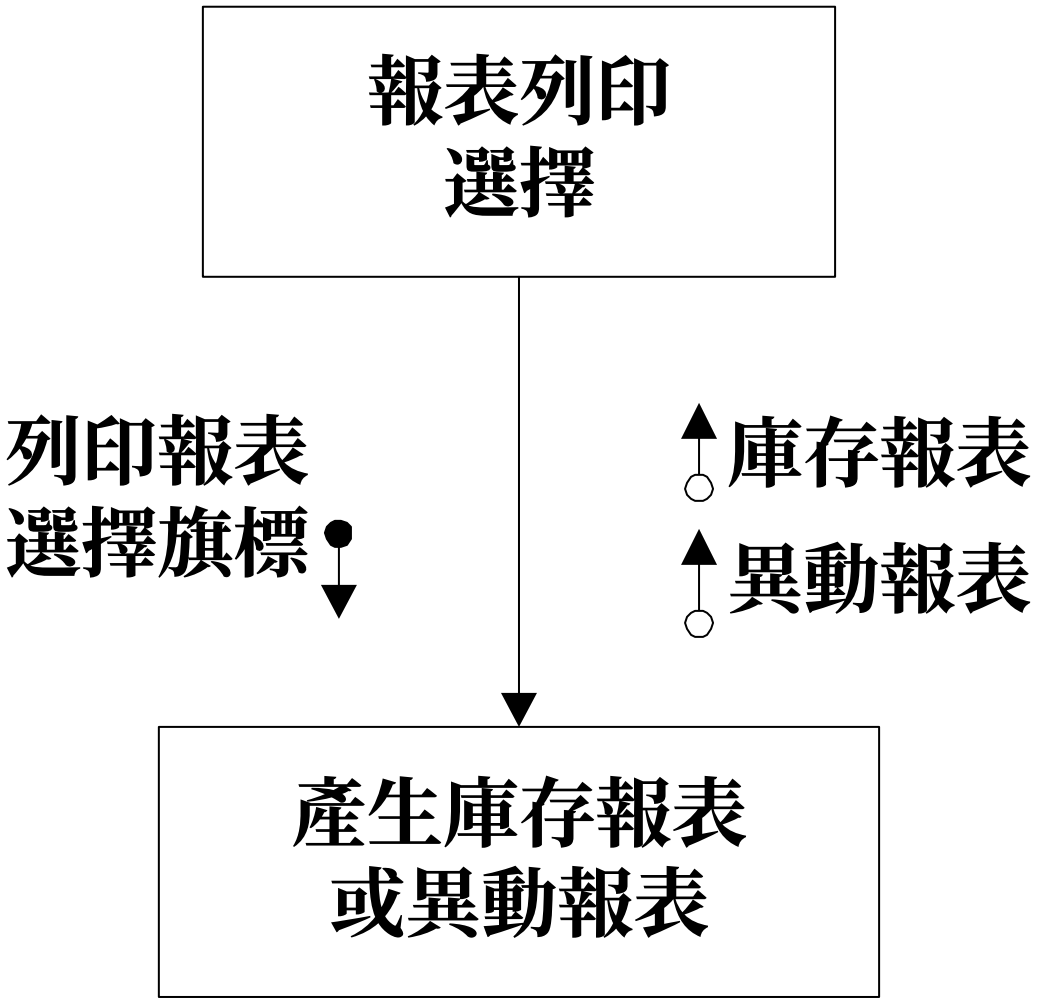
5.2.2 耦合力 (c.6)

- **資料結構耦合力可能產生以下的問題：**
  - (1)**雖然每一個模組可能只用到局部的欄位，但只要資料結構內任一個欄位修改過，則所有的相關模組均會受影響。**
  - (2)**每一個模組使用了比實際需要更多的記憶體空間。解決資料結構耦合力的方法是將所要用到的欄位傳遞過去，而不必傳整個資料結構，則資料結構耦合力就可改變成資料耦合力。**

5.2.2 耦合力 (c.7)

- **控制耦合力**
  - 控制耦合力指的是當一模組傳遞旗標去控制另一個模組內的作業（內部邏輯）時，則稱這兩模組之間具有控制耦合力。例如有兩個模組：報表列印選擇與產生庫存報表或異動報表，前一個模組傳送旗標來控制下一個模組做輸入或輸出之動作（如圖5-11），則這兩模組間具有控制耦合力。

圖5-11 控制耦合力

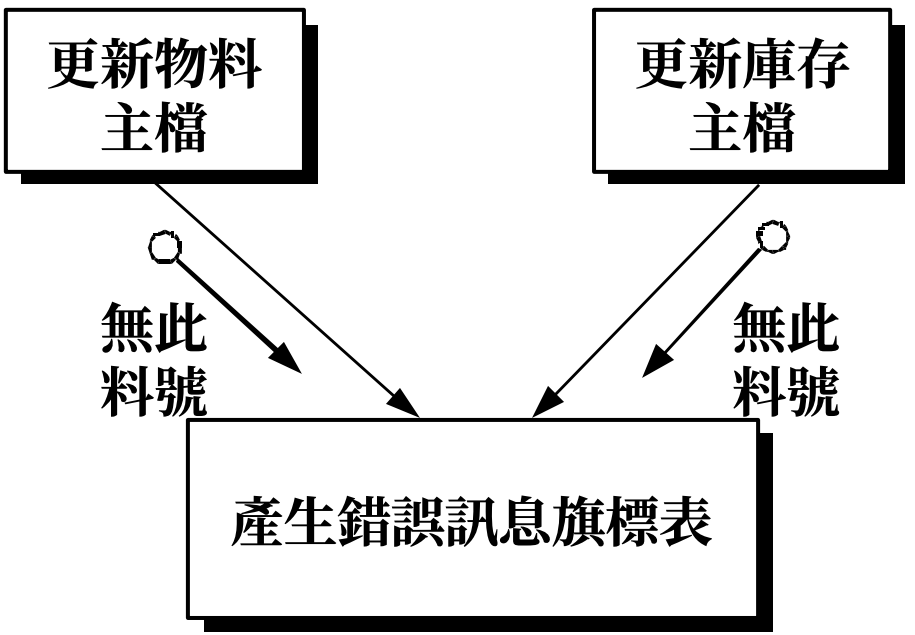


5.2.2 耦合力 (c.9)

- **控制耦合力有下列兩項缺點：**
  - (1)**如果被呼叫的模組將拆成兩個或兩個以上的模組時，會因資料的糾結或需瞭解呼叫模組等而不易達到目的。**
  - (2)**撰寫呼叫模組時，如不了解被呼叫的模組，便不易著手撰寫程式，同時會增加程式測試的成本。**

5.2.2 耦合力 (c.10)

- **共同耦合力**
  - **兩模組使用相同的資料區且都可讀寫資料區內之資料，則這兩模組具有共同耦合力。**
- **圖5-12 共同耦合力：**



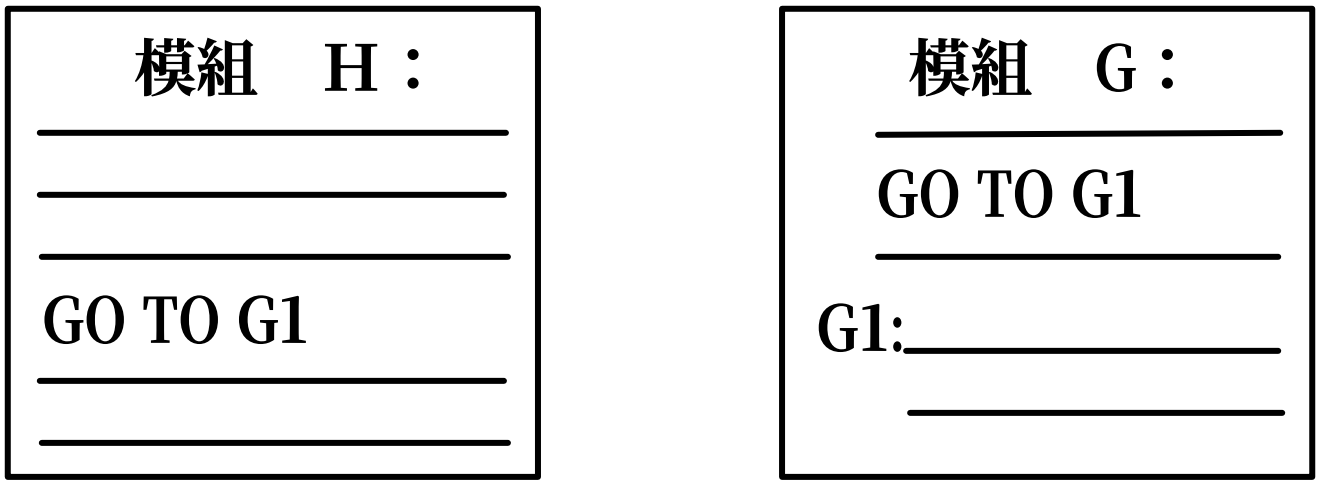
5.2.2 耦合力 (c.11)

- **共同耦合力盡量少用，主要原因為：**
  - (1)**如果共用資料產生錯誤，則所有涉及之模組均會受影響。**
  - (2)**使用共同資料區的模組名稱均模稜兩可，不易定義，經常會造成困擾。**
  - (3)**共用資料區內資料時常會被濫用，使模組的邏輯變得複雜，而不易了解。**
  - (4)**一個使用很多共用資料區的模組，在維護上相當困難。**
  - (5)**模組變動時，不知那些資料會被牽動。**

5.2.2 耦合力 (c.12)

- **內容耦合力**
  - **內容耦合力是一個模組使用另一個模組內之部份程式碼或改變其他模組內的局部變數。內容耦合力具有下列特徵：**
    (1) **一個模組以多個進入點**(Multi-entry)**的方式進入另一模組(參圖**5-13)。
    (2) **一個模組參考或改變其他模組的內部資料。**
    (3) **一個模組改變其他模組內部的執行過程。**

圖5-13 控制流程的內容耦合力



5.2.2 耦合力 (c.14)

- 一般來說，耦合力越弱越好。
- 模組間的耦合力有時可能不只是單純的一種情形，可能存在兩種以上的耦合力，此時這兩模組間的關係以較強的耦合力為準，例如兩個模組具有資料結構耦合力和共同耦合力的關係，則我們應以共同耦合力為準。

表5-2 耦合力之評比因素與結果

耦合力 種類	連鎖反應狀況	修改難度	理解性	與其他系統 之共用性
資料型	變動	好	好	好
資料結構型	變動	中等	中等	中等
控制型	中等	差	差	差
共同型	差	中等	很差	很差
內容型	很差	很差	很差	很差

- 一般而言，可以接受的內聚力包含功能內聚力、順序內聚力與溝通內聚力， 而在耦合力部份則是資料耦合力與資料結構耦合力。
- 雖然這些內聚力與耦合力是可以接受，但就系統設計而言，良好的設計希望達到模組內的內聚力為功能內聚力，即一個模組只處理單一個功能，模組間的耦合力為資料耦合力，即模組間的溝通只使用簡單型別參數來溝通。

- **一個良好的設計除了耦合力與內聚力的分析外，尚有一些值得注意的事，包括：**
  - (1)**模組功能的劃分。當模組太大為了減少功能重複的模組、管理的需求、發展可重複使用的模組或發展易撰寫的模組等情況時，都是模組功能劃分的適當時機。**
  - (2)**模組除有正規之處理外，亦須考量錯誤與輔助訊息及例外狀況之處理。**

5.3 資料流程圖建構策略

- 以資料流程圖塑模企業流程是結構化分析與設計之重點工作，也是系統模組化之重要步驟。
- 常用的資料流程圖建構方式有兩種：由上往下分割(Top-down Partitioning)與由中間往外(Middle-Out)建構方式。

5.3.1 由上往下分割

- **以由上往下分割之方式建立資料流程圖之步驟為：**

  (1) **建構環境圖。**

  (2)**由環境圖向下階層化，以分割出系統主要功能並圖示之，即第零階之資料流程圖。**

  (3)**對第零階資料流程圖中的每一個處理，再進行向下階層化，以產生更低階之資料流程圖，如此重覆進行，直到資料流程圖中所有處理不需再向下階層化為止。**

5.3.1 由上往下分割 (c.2)

- **應用由上往下分割之方式建構資料流程圖可能會碰到以下的問題：**
  - (1)**第零階之資料流程圖不容易產生，因為分析師不容易從環境圖直接分出系統的主要功能。**
  - (2)**對於一個大的系統而言，常依分析師來分割，但這種方式可能不是最佳的系統分割。**
  - (3)**倘若於舊系統上建立新系統，則舊系統的主要功能分割方式可能變成新系統的主要功能分割方式。**

5.3.2 由中間往外

- Yourdon (1988;1989) 基於由上往下分割方式可能遭遇的問題，因此建議採用由中間往外的方式建構資料流程圖，其建構步驟為：

  (1)建立環境圖。

  (2)建立事件列。

  (3)建立初步的資料流程圖，也就是將前述的事件列利用事件分割(Event partitioning)方法，以獲得初步資料流程圖。

  (4)對初步資料流進行向上及向下階層化，直到獲得完整的資料流程圖為止。

5.3.2 由中間往外 (c.2)

- **建立環境圖**
  - 因環境圖已於需求分析階段建立，因此在系統分析階段可以直接應用已建立之環境圖或只對環境圖進行修正即可。
- **建立事件列**
  - 可由環境圖中之外部實體逐一檢討其與系統之互動關係，以建立事件列，並以文句之方式命名。

5.3.2 由中間往外 (c.3)

- **建立初步的資料流程圖**
  - **將事件列利用事件分割的方法，建立初步資料流程圖**(First-cut DFD)，**步驟如下：**
    (1)**確定每個事件系統所做的回應**
    (2)**連接每個事件所對應的資料流程圖並建立初步資料流程圖**

5.3.2 由中間往外 (c.4)

- 對初步資料流程圖進行向上及向下階層化，直到獲得完整的資料流程圖為止
  - 因為每個事件對應一個處理(Process)，若事件列中有50個事件，那麼初步資料流程圖中亦將有50個處理，因此必須進行向上階層化，以降低初步DFD的處理個數，但亦額外增加上層DFD。另外部分處理可能需進行向下階層化，因此也會增加額外的下層DFD。

5.4 資料流程圖建構指南

- **以由中間往外之策略建立資料流程圖時，需做小幅修正，因為：**
  - (1)**目前應用系統之設計大多以資料庫為中心，也就是說大部分之處理所需之資料輸入與輸出都直接經由資料庫，而非處理間之直接傳遞。**
  - (2)**第3章建議用流程圖配合處理描述、藍圖與資料辭彙以表達使用者之巨觀需求。上述概念已包含環境圖與事件列之資訊，故可捨棄此兩步驟。**

5.4 資料流程圖建構指南 (c.2)

- **資料流程圖之階層數最多不要超過四層，也就是至多到第三階之資料流程圖，因為層級越多表示以後系統結構之縱深越長，系統也越不易維護。**
- **修正後之由中間往外策略的實施程序之詳細的步驟如下：**

5.4 資料流程圖建構指南 (c.3)

- **步驟一：找出初步DFD元素**
  - **首先，從需求分析之結果（流程圖及其處理描述、藍圖與資料辭彙），找出初步資料流程圖之：**
    (1) **外部實體**
    (2) **處理**
    (3) **資料儲存**
    (4) **資料流等**

5.4 資料流程圖建構指南 (c.4)

(1) 找出外部實體

- 外部實體可由所有流程圖中之外部實體得到。也就是找出在電腦化時與系統有互動關係之外部實體。

(2) 找出處理

- 初步資料流程圖之處理可由所有流程圖上之處理得到，每個處理皆有其輸入與輸出格式(Input and Output Form)、所涉及之主要與次要外部實體等。

5.4 資料流程圖建構指南 (c.5)

(3) 找出資料儲存

- 資料儲存可由需求分析中之藍圖（包括輸入與輸出格式）尋找。
- 一般來說，一個原始藍圖至少可產生一個資料儲存，但經常是可以產生好幾個。

(4) 找出資料流

- 找出外部實體、處理與資料儲存後，便可逐一檢查每一處理所需之資料來自何方及輸出到何處。
- 此外，每一處理之主要行為者其資料流均為雙向。

5.4 資料流程圖建構指南 (c.6)

- 以夢幻系統之銷售流程圖為例，至少可整理出客戶、業務部與生產部三個外部實體，訂貨與送貨兩個處理及五個資料儲存。
- 對訂單處理而言，它由客戶傳送資料所引發，主要行為者為業務部，業務部分別從客戶、產品與訂單中擷取資料，經處理後之結果存於訂單資料儲存，並傳送一份給業務部（同單位）或生產部等外部實體（參表5-4）。

表5-4 資料流表達範例

	客戶	產品	訂單	生產需求	送貨單	*客戶*	*業務部*	*生產部*
訂單處理	↓	↓	↓ ↑	↑		↓	↓ ↑	↑
送貨處理	↓	↓	↓		↓ ↑	↑	↓ ↑	

5.4 資料流程圖建構指南 (c.8)

- **步驟二：向上整合以建立資料流程圖**
  - 若處理的數目很少且很單純，可不需向上整合而直接劃出最終之資料流程圖，但在大部份的情況需將處理分群，以向上整合成較高層次之處理，且需對每一新產生之處理命名。

5.4 資料流程圖建構指南 (c.9)

- 對上層資料流程圖而言，其處理及資料流是下層資料流程圖之處理及其資料流之匯總，且外部實體與資料儲存均不變。
- 以表5-4為例，訂單處理與送貨可整合成銷售管理，而其資料為兩者資料流之聯集（如表5-5）。

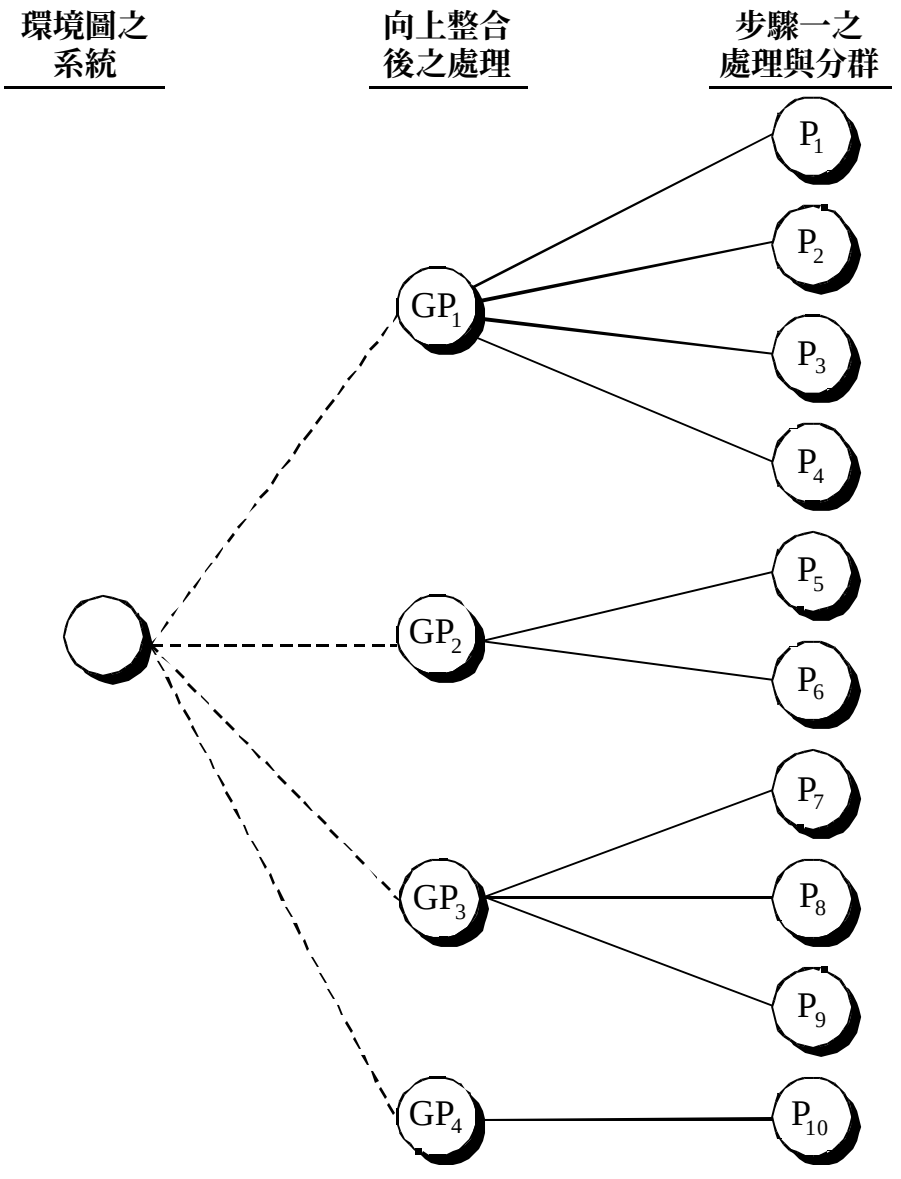
表5-5 整合後之資料流表達範例

	客戶	產品	訂單	生產需求	送貨單	客戶	業務部	生產部
訂單處理	↓	↓	↓↑	↑		↓	↓↑	↑
送貨處理	↓	↓	↓		↓↑	↑	↓↑	

⬇ 整合

	客戶	產品	訂單	生產需求	送貨單	客戶	業務部	生產部
銷售管理	↓	↓	↓↑	↑	↓↑	↓↑	↓↑	↑

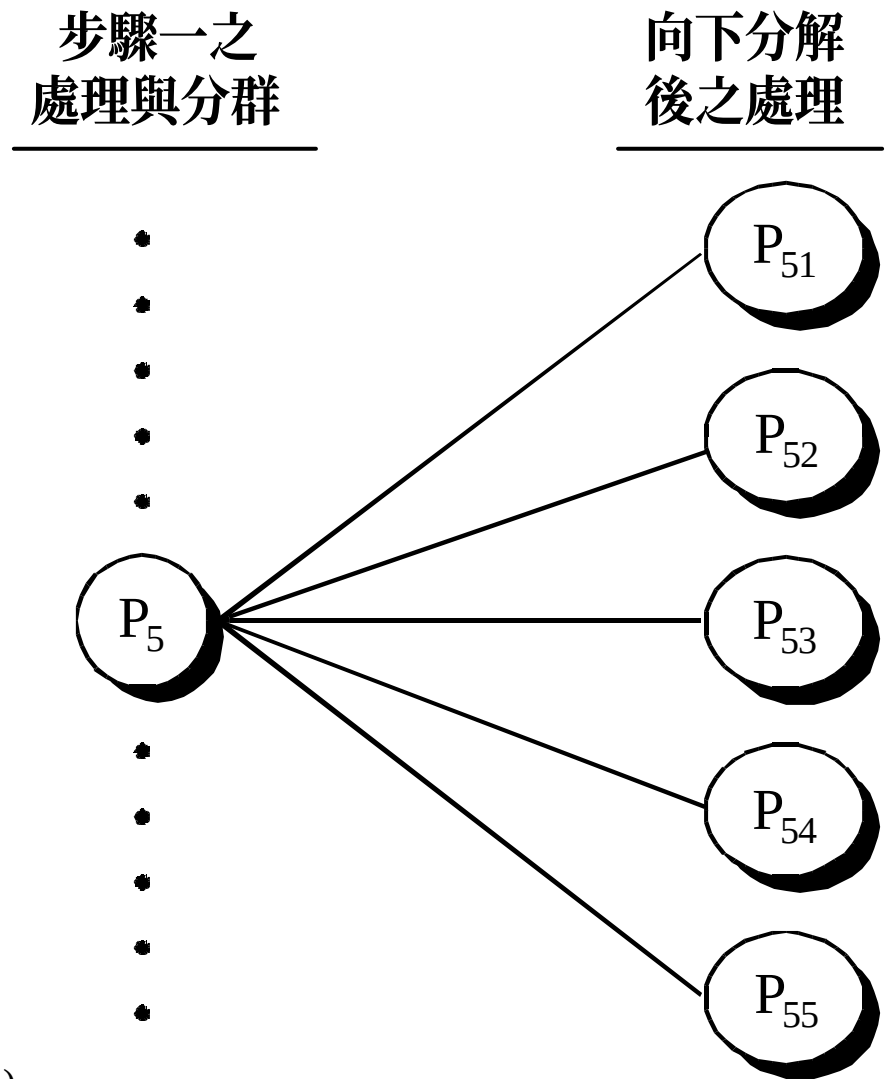
圖5-14 處理之向上整合



5.4 資料流程圖建構指南 (c.12)

- **步驟三：向下分解以建立低層資料流程圖**
  - 若步驟一產生之處理已很單純，則可不必再向下分解，但在某些情況下，例如一個處理包含太多的工作或操作，可能需將處理向下分解成多個較單純之低層處理。
  - 向下分解之原則可依內聚力或程式碼之多寡，例如不要超過200行，來判定。

圖5-15 處理之向下分解



5.5 資料流程圖的評估

- **資料流程圖的產生是經由一連串重覆階層化動作，以獲得最後之資料流程圖。一般來說，整個流程圖之製作常無法一次作好，需要反覆修改才能愈趨實用。但是如何確認最後的資料流程圖為最佳的呢？正確性(Correctness)與有用性(Usefulness)是兩個可評估的準則。**

5.5 資料流程圖的評估 (c.2)

- **測試資料流程圖的正確性**
  - (1) **對資料流程圖做外部一致性檢查**
  - (2) **對資料流程圖進行內部一致性檢查** (Consistency Checking)
  - (3) **資料維持**(Data Conservation)
  - (4) **排演** (Walkthrough)

5.5 資料流程圖的評估 (c.3)

(1) 對資料流程圖做外部一致性檢查

a. 確認資料流程圖中的每個資料流、處理及檔案皆有名稱，且均有資料字典定義之。

b. 確認每個處理是否有一個低層次資料流程圖與它對應，否則該處理便是最低層處理，應有一處理規格描述以描述該處理(系統)之行為。

c. 確認每個資料儲存是否在實體關係圖中至少存在一個實體與之對應。

5.5 資料流程圖的評估 (c.4)

(2)對資料流程圖進行內部一致性檢查，例如：

a.確認資料流程圖是否平衡，例如檢查其上下層間之資料流、資料儲存與外部實體是否皆一致。

b.檢查資料流程圖是否存在重覆或多餘的處理。

c.檢查資料流程圖中是否存在Output-only或Input-only的處理。

d.檢查資料流程圖中是否存在Output-only或Input-only的檔案。

e.檢查資料流程圖的編號是否正確，詳細準則請參第４章。

5.5 資料流程圖的評估 (c.5)

(3) 資料維持(Data Conservation)

- 藉由對處理的輸出及輸入資料流之觀察，判斷是否存在有多餘的或缺少的資料流。

(4) 排演 (Walkthrough)

- 有關資料流程圖中可能的概念性錯誤(Conceptual Error)，例如使用者作業需求方面之錯誤，若只由技術人員進行檢查很難發現，因此可以透過使用者及系統發展人員共同排演與開會討論，對資料流程圖做總檢查以找出概念性錯誤。

5.5 資料流程圖的評估 (c.6)

- **測試資料流程圖的有用性**
  - **測試資料流程圖的有用性即評估資料流程圖是否過於複雜，不容易閱讀等。通常測試資料流程圖的有用性，必須評估以下事情：**
    - (1)**處理的名稱是否有意義與唯一。**
    - (2)**最低層資料流程圖中是否存在內聚力太弱的處理，若有，則需進行向下階層化。**
    - (3)**任何一張資料流程圖中，是否存在某個處理之介面複雜度太高，即輸出/入資料流數目大多，若是，則需進行再分割。**
    - (4)**任何一張資料流程圖中，處理個數是否太多，若是，則需進行向上階層化。**

5.6 資料流程圖轉結構圖與模組設計

- **資料流程圖轉結構圖之步驟有四：**
  (1)**設立總裁**(President)**與副總裁**(Vice Presidents)
  (2)**設立較低層模組**
  (3)**修改結構圖**
  (4)**進行評鑑**

5.6 資料流程圖轉結構圖與模組設計 (c.2)

- **步驟一：設立總裁與副總裁**
  - 在結構圖中，設立一總裁，而在其下擺多位副總裁。環境圖上之系統可視為總裁，而第零階資料流程圖上之處理視為副總裁，資料流程圖上之資料流變成模組間必要的聯繫。處理聯繫時，暫時先忽略所有錯誤之發生情況、資料庫及其資料流等。

5.6 資料流程圖轉結構圖與模組設計 (c.3)

- **步驟二：設立較低層模組**
  - **把第一階及其更低階資料流程圖上之處理依序懸掛在結構圖上的副總裁底下，例如某第零階之資料流程圖下有更低階之資料流程圖，則須把第一階之處理掛在其第零階處理之下，同樣的，第二階之處理應掛在其所屬第一階處理之下。**

5.6 資料流程圖轉結構圖與模組設計 (c.4)

- **步驟三：模組設計與結構圖修改**
  - 完成第一版之結構圖後，應先對結構圖中之每一模組進行模組設計，再進一步修改結構圖使之更完美。這些工作包括：
    (1)需加入資料流程圖中所沒有的例外狀況處理，出現錯誤時之錯誤訊息處理及操作時可能之輔助訊息處理等。
    (2)將結構圖上較弱的地方再分解且加以重新組織。

5.6 資料流程圖轉結構圖與模組設計 (c.5)

- 原則上，完成資料流程圖建構後，每一個最底層的處理至少都將是一個模組。
- 經上述修改後之結構圖不一定是很好，應用內聚力與耦合力之設計評估準則可幫助我們進一步的加以改善：
  - 檢查內聚力
  - 檢查耦合力

5.6 資料流程圖轉結構圖與模組設計 (c.6)

- **檢查內聚力**
  - DeMarco (1979)**將七種內聚力分為可接受與不可接受的內聚力，摘述如下：**

內聚力種類	型　　式	代　表　意　義
可接受的內聚力	功能型、順序型、溝通型	爲可行之模組設計
不可接受的內聚力	程序型、暫時型、邏輯型、偶發型	應盡量避免使用

5.6 資料流程圖轉結構圖與模組設計 (c.7)

- **檢查耦合力**
  - **模組間的耦合力不是單一的情形，可能存在兩種以上的耦合力，這時候要以較高的耦合力為準。例如如果兩個模組間同時具有資料結構型及共同型之耦合關係，則應以共同型之耦合力為準。**
  - **應檢查結構圖上是否有不可接受的耦合力，若發現，則應加以修正，因為較強之耦合力將導致較弱的內聚力，而使得系統不易維護，應盡量避免。**

5.6 資料流程圖轉結構圖與模組設計 (c.8)

- **步驟四：進行評鑑**
  - 完成模組設計與結構圖修改後，接下來應確定結構圖的運作功能。也就是該結構圖應能正確的描述系統的行為，以完成流程圖上所描述之企業流程與規則。
  - 進行評鑑之目的是希望能儘早找出錯誤並及早修正，而不希望等到系統完成或在運作時發生錯誤再去修改它。

5.7 結論

- 本書建議資料流程圖之建立採由中間往外之策略，該策略之概念與執行已經過修改與擴充，主要概念有二：
  (1)處理間之資訊輸入與輸出以資料庫為中心，也就是說大部分之處理所需之資料輸入與輸出都直接由資料庫，而非處理間之直接傳遞。
  (2)以需求分析之流程圖，配合處理描述、藍圖與資料辭彙以表達使用者之巨觀需求，並將這些資訊直接轉成資料流程圖之元素，以簡化資料流程圖之製作。

5.7 結論 (c.2)

- **此外，流程圖之處理描述、資料流程圖之處理描述和模組設計中之描述間有一些差異，摘述如下：**

不同階段之處理描述	不同概念層級之描述重點
流程圖之處理描述	主要描述企業流程與規則
資料流程圖之處理描述	主要描述系統行為
模組設計中之描述	主要描述程式邏輯

5.7 結論 (c.3)

- **雖然資料流程圖已廣泛的被應用於企業流程塑模上，但基本上資料流程圖之應用仍有其不足的地方，例如：**
  - (1)**資料流程圖之應用是功能導向的結構化分析，一旦流程或功能有所改變，將會導致資料流程圖產生一連串改變。**
  - (2)**資料流程圖缺乏時間狀態表示，在記載流程順序時，並未提供和時間有關的資訊與控制。**
  - (3)**有關資料流程圖在製作上及使用者之學習方面仍須再改進。**