系統分析與設計

From 線上測驗網站 http://david.bioinformatic.idv.tw/LawQuiz/

第10章. 資料庫系統設計

10.1. 資料庫的意義

10.2. 資料庫系統

10.3. 資料庫管理系統

10.4. 資料庫系統之設計

10.5. 個體導向資料庫

10.6. 資料倉儲

第11章. 系統發展

11.1. 系統發展之重要工具

11.1.1. HIPO 圖

HIPO 是 Hierarchy plus Input-Process-Output 的縮寫，HIPO 圖一種圖解的方法，可用來描述系統或程式的輸入、輸出及其處理功能。

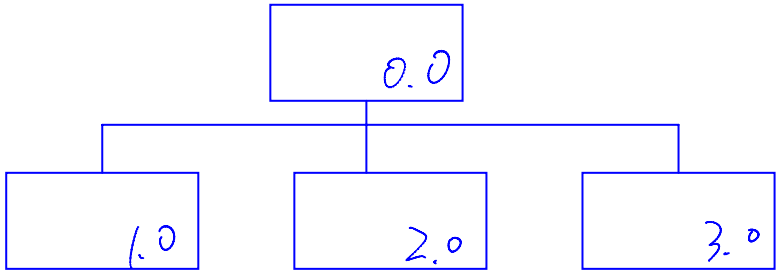
一 HIPO 的目標

- 建立一個系統的結構，其系統功能很容易理解
- 能夠描述一個 Program 的主要功能
- 提供可看的見(visual)的輸出入描述，input 如何轉換為 output
- 功能之描述是層次化的

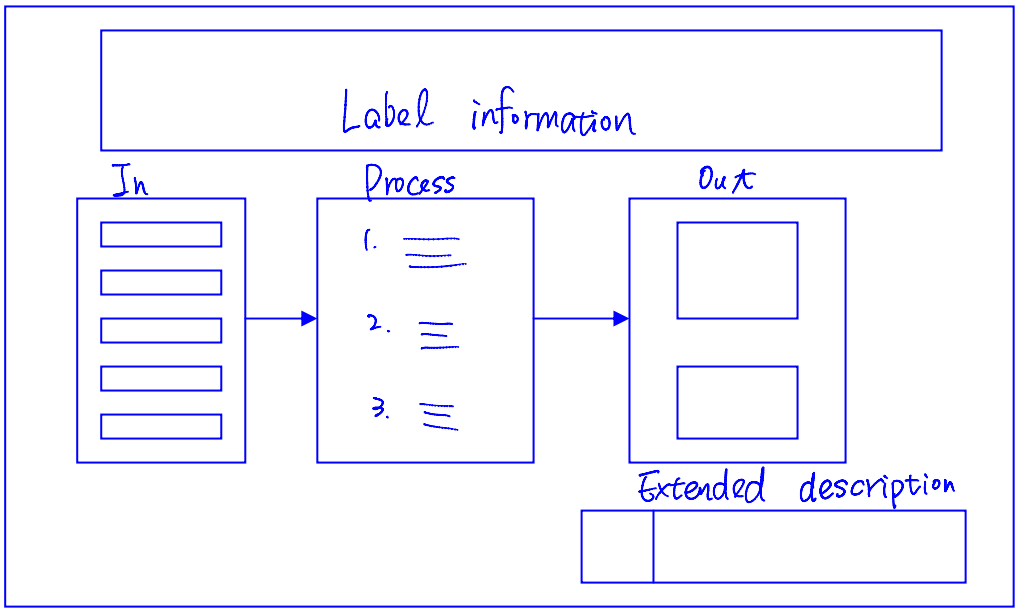
二 HIPO 圖的組成

HIPO 圖由下面三種形式的圖所組成

- **VTOC(Visual Table of Content)**



♦ **大綱圖(Overview diagram)**



♦ **細節圖(Detail diagram)**

細節圖屬於最低層次的 HIPO 圖，用來描述最 low-level 的 function。每一個細節圖也都包含有 Input、Process、Output 以及一個 exended description area。

三　HIPO 圖的優缺點

♦ **優點**

- 簡單性，易繪製
- 使用者易學
- 可作為使用者與 SA 溝通工具

♦ **缺點**

- 對大型系統很難去驗證其正確性
- Data feedback 很困難去描述
- 無法同時描述 data structure 及 program structures

四　HIPO 圖在文件製作上，可應用的階段

- 初步設計
- 細部設計
- 維護

11.1.2. SADT

SADT 是 Structured Analysis and Design Technique 的簡稱。

一　SADT 的主要用途

- 作為描述系統的圖形語言
- 作為產生系統描述的一種方法，可用於結構化分析與設計上。

二　SADT 所使用的符號主要有以下兩種圖

♦ **Activity diagram**

♦ **Data diagram**

Actigram 和 datagram 的描述都是由上而下的方式來逐漸劃分的

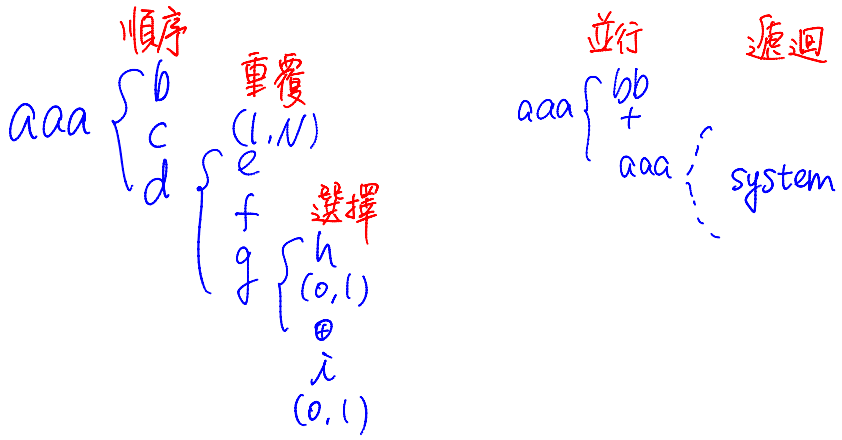
三　SADT 可應用的階段

可用於結構化分析與設計上。

11.1.3.　W/O 圖

W/O 圖是 Warnier-Orr diagram 的簡稱。W/O 圖與 HIPO 及結構圖的性質相似，主要用來描述系統或程式的輸入/輸出資料的層次結構。

一　W/O 圖的三種基本結構



二　W/O 圖可應用的階段

W/O 圖可以描述系統、程式或模組的程序結構和資料結構。故可用於系統設計和程式設計上。

三　W/O 的優缺點

- **優點**
  - 具有區段結構的特性，容易轉換成結構化程式碼
  - 提供良好的資料結構圖形，故容易製作文件
  - 可同時表達程式結構和資料結構
- **缺點**
  - 由於是層次化的結構，高層次容易閱讀，但越低層次越難閱讀
  - 不能顯示條件式的判斷邏輯

範例55.

試說明下列軟體開發技術或工具的意義，並指出它可用於資訊系統開發的哪些階段

(1)DFD

(2)W-O diagram

(3)E-R diagram

(4)structured chart

(5)context diagram

Ans:

(1)DFD: 結構化分析

(2)W-O diagram: 系統設計，程式設計

(3)E-R diagram: 結構化分析

(4)structured chart: 結構化設計

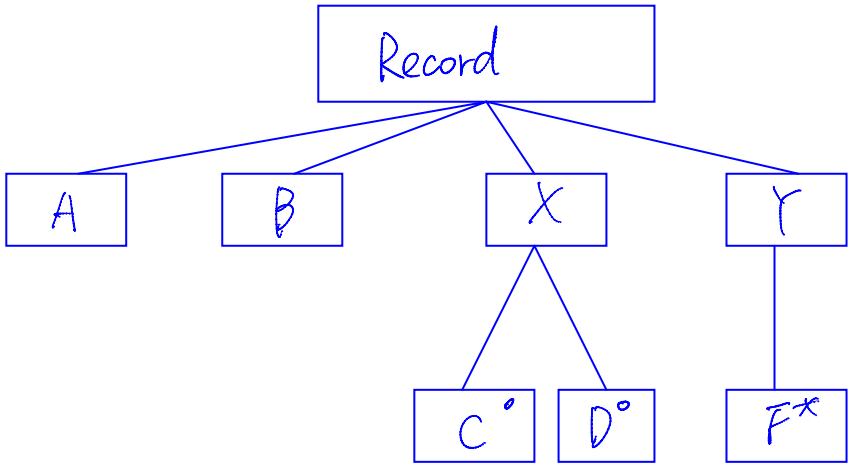
(5)context diagram: 結構化分析工具，DFD 的一部分

11.1.4. Jackson 圖

Jackson 圖是指 Michael Jackson diagram，此種圖與 W/O 圖均能夠同時顯示程式結構與資料結構。

範例56.

現有 Jackson 方法之資料結構圖如下，請用結構化分析中常用的資料字典方式定義之



Ans:

Record=A+B+X+Y

X=[C|D]

Y={F}

11.1.5. 流程圖

一　系統流程圖

二　程式流程圖

- ♦ **程式流程圖的缺點**
  - 容易導致非結構化設計
  - 不容易修改
  - 不能適切的說明程式內容

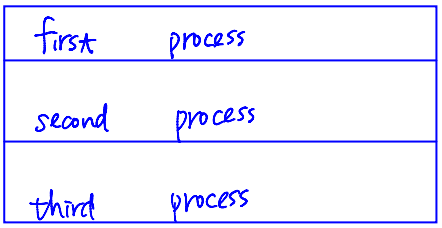
11.1.6. N-S 方塊圖

一　簡介

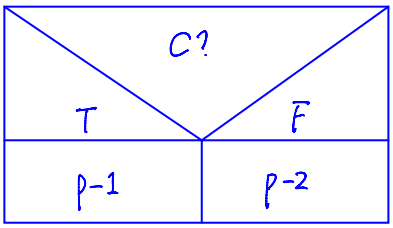
- N-S 是 Nassi-Shneiderman chart 的簡稱，所畫出來的圖一定是結構化程式。
- 使用 N-S 方塊圖主要是要解決流程圖無法用來描述結構化程式(structured programs)的問題。
- N-S 方塊圖具有可描述程式邏輯的結構及層次化關係的能力。

二　N-S 方塊圖的控制結構所使用的符號如下

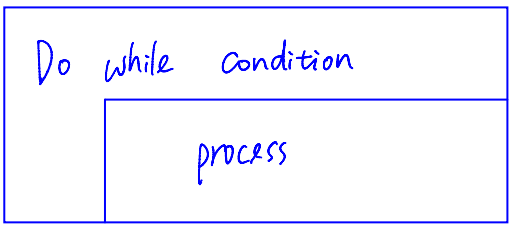
♦　**順序**



♦　**選擇**



♦　**重複**



三　N-S 方塊圖的優點

- 有明確而且醒目的重複架構
- 有明確且醒目的選擇架構
- 區域性變數和整體性變數的範圍可以區分的很清楚
- 不能任意改變控制流程，可以確保產生結構化程式。
- 因為沒有跳頁的連接符號，任何思考架構必須在一頁內表達清楚。
- 順利的表示遞迴程式

四　N-S 方塊圖的缺點

- 不易修改、不易繪製
- 只適用程式細節的描述

範例57.

系統發展工具中，常用的流程圖有哪些？其功用為何？

Ans:

♦ **系統流程圖**

主要用於系統流程的設計，即用來對整個系統的電腦處理流程進行設計與規劃。

♦ **程式流程圖**

是一種程式設計工具，他是以圖形格式來顯示程式中將被執行的敘述，以及支配他們執行的控制邏輯，或處理方塊的順序。程式流程圖可以提供兩個目的

a. 當作程式設計工具來規劃細部或複雜的程式邏輯

b. 可當作程式說明文件

♦ **N-S 方塊圖**

主要功用與程式流程圖相類似，但程式流程圖無法保證可設計出結構化程式，而 N-S 方塊圖則可以。N-S 方塊圖提供順序、選擇及重複等三種控制結構來描述程式，並且每一種組成結構都只有一個入口和一個出口。

♦ **資料流程圖**

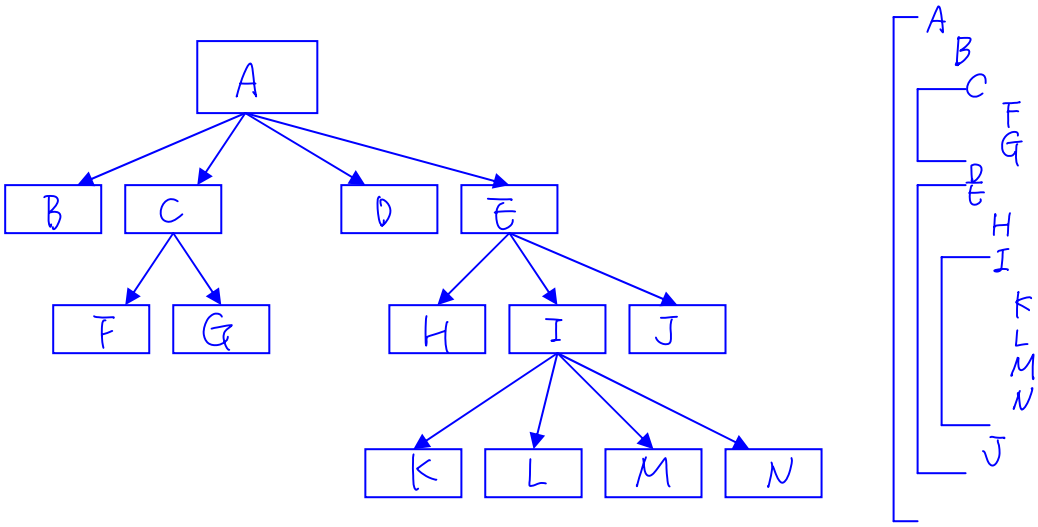
是一種結構化分析的工具，他是以圖形來描述系統中各部份之間的資料流動情況，並且可以顯示系統功能的分割結構，以及分割後各部份的介面。

11.1.7.　動作圖

動作圖是一種用來繪出高層次分解、程式結構以及程式控制結構細節(條件、CASE、DO WHILE 及 DO UNTIL 之結構)的簡易方法。

大多數的圖解方法均無法同時會出程式的結構大綱以及控制結構細節，但動作圖可以滿足以上的要求。

模組階層圖對應的動作圖



動作圖的優點

- 很容易繪製及更改
- 對於人工描繪及電腦編輯都可以用
- 是唯一能從最高整體概觀擴展成細部編碼階段的方法(最終的分解)
- 可繪製各種傳統結構化程式的架構，而且比虛擬碼更為生動
- 很容易教導終端使用者；也鼓勵終端使用者將他們的功能擴展成審查或設計細部的程序邏輯
- 可列印在壹班寬度的紙張上，不必用大張的壁紙，這使得他們適合採用個人電腦來設計。
- 可與資料模式相結合
- 能與第四代語言相配合的很好，且能適應特殊語言的語法
- 能以電腦交互核對複雜規格上所能使用的資料

11.1.8. 赫思圖(HOS chart)

HOS 是功能分解圖(一種樹狀結構)的控制形式，用來顯示每一方塊的輸入與輸出的資料類型。HOS(High Order Software)最大優點是能夠建立複雜系統的規格說明，證明內容的一致性且無潛在錯誤存在，同時這些規格說明可以用來產生零錯誤的程式碼。

HOS 的創始者認為所有的功能分解都能劃分成為二元的分解方式，供有三種型態：先 A 後 B、AB 無先後與 AB 二選一。

11.1.9. 因果圖

因果圖(Cause-effect graph)可應用在功能測試中

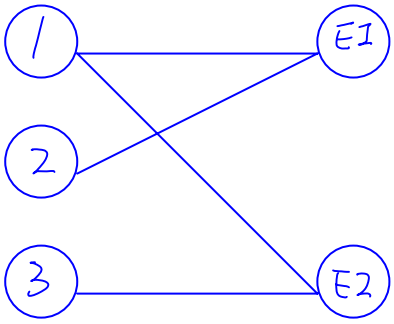
- 功能測試是檢查系統的實際功能是否滿足系統的功能性要求
- 系統的每一項功能均能與執行該項功能的系統組件有密切關聯。執行某項功能的一組模組被稱作 thread，所以功能測試有時也被稱作 thread testing。
- 因果圖可以提供一種藉分析需求來建立測試個案的方法。

畫成因果圖之後，接下來是把因果圖轉換成為決策表，決策表上應該標上

- **因的部份**
  - 真 I(invoked)
  - 偽 S(suppressed)
  - 無關緊要 X(don't care)
- **果的部份**
  - 出現 P(present)
  - 未出現 A(absent)

因: 我很乖 1
爸爸心情好 2
媽媽心情好 3

果: 得十元 E1
得兩元 E2



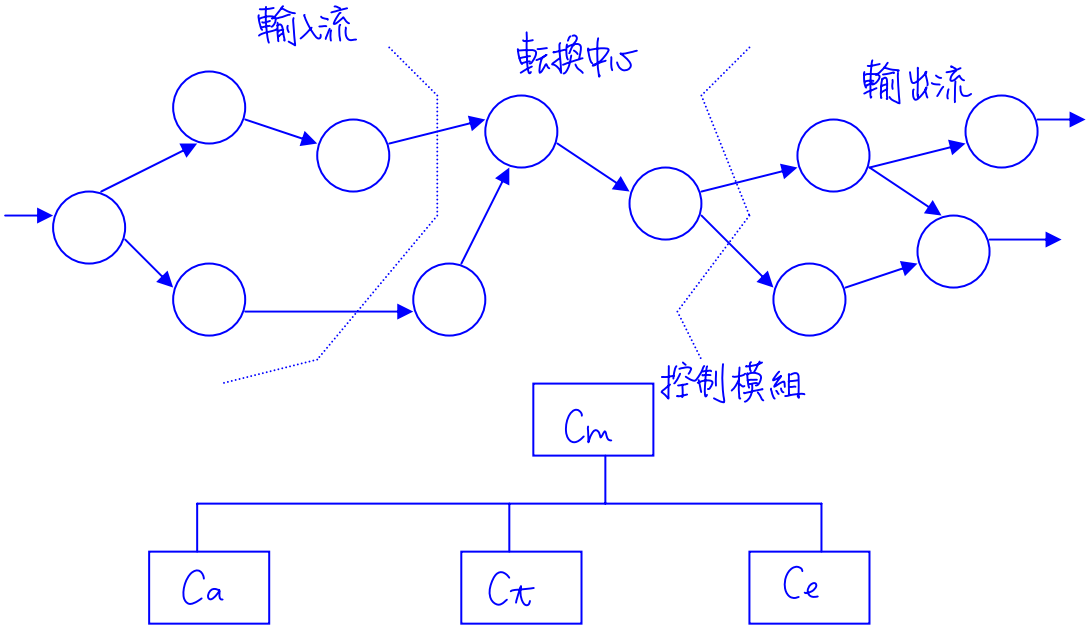
	Test case 1	**Test case 2**
1	I	I
2	S	I
3	I	S
E1	A	P
E2	P	A

11.2. 程式設計方法論

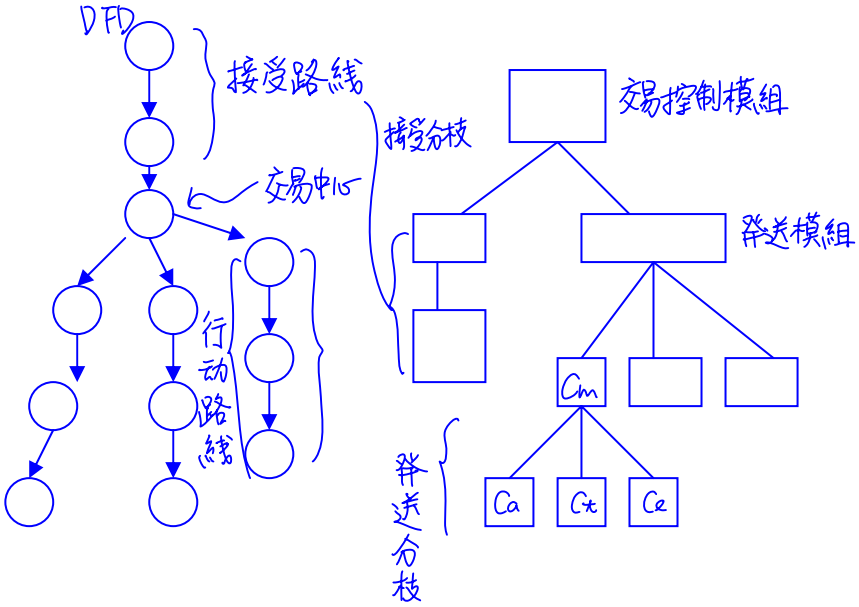
11.2.1. 資料流導向設計法-Yourdon 法

Yourdon 法的目的就是把資料流程圖，轉換成為結構圖。

♦ 轉換分析



♦ 交易分析



11.2.2. 資料結構導向設計法-Jackson 法

資料結構導向設計法的目的，和資料流導向設計法一樣，仍然是將軟體系統的規格轉換成軟體結構的過程，提供一套系統化的技術。所不同的是資料結構導向設計法是以資料的結構為設計的藍圖，在設計過程中所使用的工具並不是資料流程圖及結構圖，而是階層化的資料結構圖以及程式結構圖。

11.2.3. 資料流與資料結構導向技術的比較

一 共同點

資料流導向與資料結構導向設計技術，最重要的共同點，就是兩者都是

建築在「程式結構可以由資料的特性導出，因此軟體設計的藍圖應該是問題所欲處理資料的特性，而不是控制程序」的理論上。

二　相異處

資料流導向與資料結構導向設計，可從下列四點區分

- **資料特性**
  - 資料流
  - 資料結構
- **工具**
  - 資料流程圖及結構圖
  - Jackson 圖及虛擬碼
- **產生結果**
  - 結構圖(初步設計的結果)
  - 虛擬碼的演算程序
- **所欲達成的目標**
  - 產生的結構圖虛符合最大內聚，最小耦合
  - 產生優良的演算程序

範例58.

何未結構化分析？有哪些不同的分析方法？列舉三種

Ans:

a. 結構化分析是指將系統的架構、功能與處理程序，以各種系統邏輯模型(即各種結構化分析工具)予以表示的分析法，其產品稱為結構化規格書。

b. 三種結構化分析方法

- **資料流程法**

太簡單懶的打

- **結構化分析和設計技術(Structured Analysis and Design Technique, SADT)**

此種方法提供了活動圖(actigram)和資料圖(datagram)這兩種圖形工具，提供系統分析師對軟體系統的功能進行逐步分解，並描述資訊(資料與控制)在整個系統彼此間的關係。活動圖與資料圖的功用如下

- 活動圖用來顯示系統中的活動、活動之間的關係、以及活動拆解的情形
- 資料圖用來顯示系統中所處理的資料以及資料拆解的情形

- **Jackson 系統開發(Jackson System Development, JSD)**

JSD 屬於資料結構導向的分析方法，主要是以 Jackson 圖作為描述工具。Jackson 圖可用相同的符號來表示資料結構和程序結構，共有順序、選擇及重複三種成員結構。

JSD 法乃是一種循著系統化的方法，做出問題之資料結構所對應的軟體結構，使問題獲得解決的分析與設計技術，重點包括

- 幫助分析人員確定關鍵資訊物件和操作
- 使用 Jackson 圖表示輸入與輸出資料的順序、選擇和重複特性
- 找出輸入與輸出資料結構的共同點，將兩者合而為一，再用直接對應的方式做出軟體結構圖

11.3. 測試

11.3.1. 軟體錯誤

11.3.2. 測試工作

一　測試的實施階段

- ♦ **單元測試**
- ♦ **整合測試**
- ♦ **功能測試**
- ♦ **效能測試**
- ♦ **驗收測試**

二　測試的類型

- ♦ **黑箱測試**
- ♦ **白箱測試**

11.3.3. 程式測試

一　單元測試

二　整合測試

- ♦ **由下而上測試法**
- ♦ **由上而下測試法**

11.3.4. 系統測試

以黑箱測試為主

- ♦ **功能測試**
- ♦ **效能測試**
  - Stress test
  - Volumn test
  - Configuration test
  - Compatibility test
  - 回歸測試
  - 安全測試
  - 時序測試
  - 環境測試
  - 品質測試
  - 復原測試
  - 維護測試
  - 文件測試
  - 人工因素測試
- ♦ **驗收測試**
- ♦ **安裝測試**

11.3.5. 白箱測試

11.3.6. 黑箱測試

範例59.

試述測試的目標。請說明好的測試案例應具備哪些條件？

Ans:

a. 依照 Myers 對測試的目標所下的定義為

- 測試是為了從一可執行的程式找出錯誤來(有效性)
- 一個好的測試案件，其具有較高的機率來發現尚未被發現的錯誤(機率性)
- 一個成功的測試，是發現了尚未被發現的錯誤(預知性)

以上目的總之為抓錯、找錯，進而希望能找出沒有效率的程式碼

b. 一個好的測試案件應具備的條件

- 有效的找出錯誤
- 儘可能的測試所有路徑
- 以最小量的資料完成可接受的測試
- 成本最低的測試
- 有一個測試的標準

範例60.

解釋名詞

(1)錯誤測試(Error Checking Routine)

(2)正常測試(Normal Path Testing)

(3)white box testing

(4)驗證與確認(Verification & Validation; V&V)

(5)stress testing

(6)Beta Test

(7)Unit Test

(8)迴轉時間(Turnaround time)

(9)產生(Throughput)

Ans:

(1)錯誤測試(Error Checking Routine)

在資料通訊的過程中，當資料傳送到接收端時，用來檢測資料是否發生錯誤的常式

(2)正常測試(Normal Path Testing)

對經常會發生的事件，作一個正常性的測試，以檢查所建立的邏輯模式是否正確。

(3)white box testing

將受測對象視為「開放箱」，俗稱白箱。其內部結構與邏輯皆已得知，可執行模組內所有敘述以及所有控制路徑，以檢查受測對象能否順利運作。

(4)驗證與確認(Verification & Validation; V&V)

Verification: Are we building the product right?

Validation: Are we building the right product

(5)stress testing

滿載測試主要用來評估系統在滿載情況的運作情形。當需求文件指定系統必須處理某種數量的裝置獲使用者的要求時，滿載測試可以評估這些裝置都開機或使用者同時使用時，系統的功能是否合乎預設的要求標準。對於平時負載量不大而尖峰需求時期又嚴重負載的系統，尤須進行滿載的測試。

(6)Beta Test

在驗收測試階段，假定系統已經實際安裝於工作場所，由使用者來評估系統的性能，此種測試方法稱為實驗測驗。若由軟體公司本身的成員擔任使用者，先進行實驗測試(稱為 alpha test)，之後再將系統交給顧客進行真正的實驗測試，此時顧客所進行的實驗測試就稱為 Beta Test。

(7)Unit Test

在程式設計階段，由程式設計師自行對所撰寫的程式單元(或模組)進行測試，以便及早發現問題，此種測試工作稱為單元測試。

(8)迴轉時間(Turnaround time)

a. 從程式設計完成、上線測試到執行，最後將執行結果交回給程式員手中所需的時間稱之。

b. 在通訊中，將傳輸訊號改變方向所需的時間稱之。

(9)產生(Throughput)

單位時間內系統可以處理完成的資料量

範例61.

(1)結構化程式設計

(2)因果圖

(3)資料流程圖

(4)STS 分解

(5)雛型

(6)由上而下測試

(7)交易分解

以上所列的各種技術各用在哪個階段？

Ans:

(1)結構化程式設計	實做
(2)因果圖	測試
(3)資料流程圖	分析
(4)STS 分解	分析
(5)雛型	需求分析
(6)由上而下測試	測試
(7)交易分解	設計

11.4. 撰寫系統文件

11.4.1. 系統文件

系統文件可分成以下四種

一　系統說明書

二　程式說明書

三　操作說明書

四　使用者說明書

11.4.2.　其他與資訊系統管理相關的文件

一　機房管理文件

二　行政管理文件

範例62.

文件通常分為兩類：一為系統發展文件；另一為計畫控制文件。試說明此兩類文件。並說明計畫控制文件常被忽略的原因及後果。

Ans:

a.系統發展文件通常包括

- 系統說明書
- 程式說明書
- 操作說明書
- 使用者說明書

c.　計畫控制文件通常包括

- 進度控制文件：如甘特圖 PERT 圖
- 預算控制文件
- 其他管理文件

計畫控制文件常被忽略是因為在計畫進行期間，重點工作都集中在資訊系統的發展上，對於各種控制文件的整理與撰寫都不注重，如此容易造成專案計畫的管理出問題，如進度落後、預算超支，或輕易的改變計畫內容等等。

11.5. 系統轉換

系統轉換(System conversion)是指由某一系統轉換為另一個系統的意思。當新系統發展完成並辦妥各種訓練之後，舊系統即可以停用，由新系統取而代之。

常用的系統轉化方式有以下五種，必要時可以兩種或三種並用

一　平行轉換

二　分批轉換

三　分段轉換

四　立即轉換

五　試驗轉換/前導轉換

範例63.

解釋名詞：Pilot Conversion

Ans:

前導轉換是指在新系統全面施行之前，先選擇少數的使用者作為系統轉換的前導，利用並行或直接或漸次轉換的模式先行轉換，直到這些使用者運作正常之後，再全體使用者一起進行系統轉換。

第12章. 資料通信

12.1. 資料通訊簡介

12.2. 資料傳輸之分類

12.3. 區域網路技術

12.4. 廣域網路

12.5. 網路設計的各項考慮

12.6. 通信協定

12.7. 錯誤控制

12.8. 網路新發展與應用

第13章. 專案管理、維護及品質保證

13.1. 專案管理

13.1.1. 專案計畫

一　專案計畫

爲了完成某一個特定的工作任務，由不同專長的人員共同組成一個跨越部門組織的專案小組，共同完成此依工作，等待任務完成之後，專案小組可以立刻解散，成員各自回到原來部門。

二　專案管理

在特定的工作目標之下，實施專案計畫的管理程序如計畫、組織、領導、控制與協調，以最經濟有效的方法達成目標者，稱之為專案管理。

13.1.2. 專案組織

13.1.3. 專案管理的問題與特性

範例64.

CoCoMo 模型

Ans:

CoCoMo(Constructive Cost Model)模型是一種非線性的軟體成本估算模式，可分為三種不同的詳細程度

a. 基本模式：只考慮軟體的規模

b. 中級模式：考慮軟體規模和十五項成本影響因子

c. 詳細模式：進一步將成本依照開發階段來劃分

上述每一種模式又將軟體分為簡單型、中間型及複雜型等三種複雜程度。

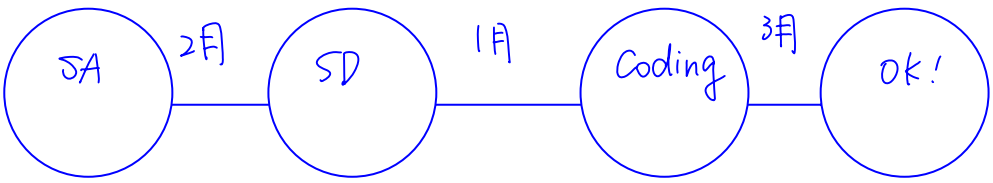
13.1.4. 專案管理的工具

一　進度控制

- **甘特圖**
- **要徑法(CPM：Critical Path Method)**

由於甘特圖相當簡單，因此很難將工作細分，在工作內容的預計和進度的推算上都不是很精確。對於較複雜的專案計畫，較適合採用要徑法來管制其進度。

要徑法常用的工具為 PERT Chart，圓形代表事件，直線代表事件的執行。



計算每一個事件的最早與最晚開始時間

- **最早開始時間**

先期事件的最早開始時間，加上先期事件的期望完成時間

- **最晚開始時間**

後續事件的最晚開始時間，減去本事件的期望完成時間

13.2. 系統維護

13.2.1. 系統維護

系統維護工作可以分成以下四類

- 更正性維護(修改)
- 適應性維護(新增)
- 完善性維護(新增)
- 預防性維護(新增)

13.2.2. 組態管理

一個大型資訊系統的維護工作中，最重要的就是軟體的「組態管理」(configuration management)。組態管理的任務乃是在一個大型資訊系統的生命週期之間，儘可能的近低整個系統的維護成本並減少更改所造成的衝擊。組態管理的主要工作項目，包括下列四項工作

- **組態規劃**

所有的文件都算是軟體的組態。組態管理的規劃，第一項工作就是要決定哪些文件要列為正式組態管理的範圍之內，也就是所謂的正規文件。再決定了正規文件的範圍之後，第二項工作就是要確定

- 正規文件的命名策略
- 正規文件之間的各種關係
- 由誰負責某一個正規文件的品質
- 由誰負責成交某一個正規文件

最後，組態管理的規劃必須建立一個組態管理資料庫。在這個資料庫裡面，所有組態管理的紀錄都有系統的組織起來，並提供一個友善的查詢介面讓組態管理人員得以容易的查詢有關系統更改、系統建構，與系統版本的諸項問題。

- **更改控制**

當一個系統被認定為基準線時，即表示該系統已經發行且正被使用中。

- **系統建構**
- **版本控制**

13.3. 軟體品質保證

13.3.1. 影響軟體品質的因素

根據 McCall 的定義

♦ **在產品操作方面**

- 正確性
- 可靠性
- 效率性
- 安全性
- 可用性

♦ **在產品修改方面**

- 可維護性
- 更改彈性
- 可測試性

♦ **在產品移植方面**

- 可攜性
- 可再利用性
- 互動性

13.3.2. 軟體品質保證的工作項目

- 使用技術性的方法與工具
- 正式技術覆核
- 軟體測試
- 標準化
- 更改控制
- 測量
- 紀錄與報告

13.3.3. 軟體可靠度

大部分軟體可靠度的研究重點，都放在如何估計軟體可能發生「異常行為」的機率問題上。

軟體可靠度的測量標準

- 失效率：也就是當機
- 出錯率
- 平均失效期率：系統兩次失效之間的平均時間
- 待用率：系統隨時待用的機率，這種可靠度適合用來測量那些修復與重新啟動需要大量時間的系統。

範例65.

試說明資訊系統開發的每一個階段中，有哪些做法，可以確保所開發系統的品質？

Ans:

在分析階段應做需求審查的工作，包括

- 審查系統的目的和目標
- 將需求與系統的目的和目標作比較，以確認是否所有的需求都是必要的
- 檢查這個系統與其他系統之間的介面，並確保需求能正確的反應客戶的心意和目的
- 評估系統發展或實際運作時的風險，討論和比較各種選擇方案，並且確定客戶同意我們所選用的方法
- 審查驗收測試的標準

在世紀階段應對系統設計規格書進行覆核，覆核的檢查工作包括

- 是否能解決客戶的問題
- 使否為模組化結構？其結構品質是否良好？
- 能否改進現有的模組化結構？
- 是否具有可攜性
- 是否有高度的可讀性
- 演算法的選用是否適切
- 如果是階段是發展的一部分，其階段介面的設計是否正確？
- 是否具備品質良好的文件說明？
- 是否具備各模組與需求規格的對照參考表？
- 是否具備各資料與需求規格的對照參考表？

在製作階段應該進行各種測試工作

- 單元測試
- 整合測試
- 功能測試
- 效能測試
- 驗收測試
- 安裝測試

第14章. 電腦輔助軟體工程

14.1. CASE 的概念

利用電腦科技來輔助軟體開發的過程、方法、技術和工具，都可以稱作「電腦輔助軟體工程」。

14.2. CASE 的分類

14.2.1. 依照支援的範圍分類

- 工具：CASE tools
- 工作平台：CASE workbenches
- 環境：CASE enviroments

14.3. 整合式 CASE

14.4. CASE 的生命週期

14.5. CASE 工作平台

14.6. CASE 軟體發展環境

14.7. 考古題

範例66.

在系統發展中，使用 CASE Tools 有哪些限制與缺點？

Ans:

a.並非所有的電腦系統都適合用 CASE tools 來發展，使用 CASE tools 有以下的限制，後者較適合 CASE

- 技術性/觀念性
- 效率/彈性
- 作業上/管理上
- 專業人員/專端使用者

c. CASE tools 使用上的缺點

- 它能有效處理軟體發展的問題，但無法提出現行基本程式不足之處
- 採用此種工具會改變程式寫作方式，容易造成程式設計師之抗拒心理
- 採用此種工具需經過一段長時間的訓練方能操作
- 目前系統發展的各階段之 CASE tools 整合不足(????虎爛 看見鬼)

範例67.

解釋名詞 第四代技術(4th Generation Technique, 4GT)

Ans:

第四代技術包含了廣泛的一群軟體工具，且他們都有一個共同點：即每個工具都能讓軟體開發者在較高的層次上描述軟體的特性，然後再依照其描述自動的產生原始程式碼。