・資訊專題・

電腦軟體面面觀

榮泰生

國立政治大學企業管理研究所博士班學生

電腦是現代社會日益受重視的一項工具。
本文作者除了把電腦軟體做了概括性的介紹，
還提出企業購買軟體與自行開發的問題，
頗具參考價值。

軟體的意義與分類

軟體在建立資訊系統的過程中扮演一個關鍵性的角色。企業欲獲得軟體不外乎購買及自行開發兩種方式。如果企業決定購買軟體，則必須認明可能產生的風險並設法將此風險減到最低（minimization of the risks）；如果企業決定自行開發，則要注意所謂認知偏差（perceptual bias）的問題。

李文首將軟體種類做一概括性的介紹，

然後闡述上述兩個主題。我們了解愼重而正確購買所需軟體，或自行開發有系統且符合使用者需求的軟體可以避免無謂的爭辯、抱怨甚至訴訟；積極地說，成功的軟體系統可使軟體開發者有成就感，而使用者有滿足感。

通常軟體是指由電腦製造商提供或公司自行開發的程式，使購買者或公司本身能有效地使用電腦、解決問題。軟體包括一連串的指令或例行程式，與硬體配合來執行資料處理的功能。廣義的說，軟體包括手册（manuals）、文件編製（documentation）、甚至包括資訊部門的政策。

軟體可分爲應用軟體與系統軟體。應用軟體是解決一個特定問題（商業的、科學的問題）的程式，而系統軟體（亦稱作業系統軟體）所涉及的是中央運算單位（CPU）以及週邊設備的運作。

一、應用軟體

應用軟體對使用對象的不同可分爲套裝軟體、商業程式以及科學程式。

由於大多數的廠商均有類似的資訊需求，套裝軟體隨因應而生。用一般的決策規則，以進行統計運算、分類、檔合併（merge）以及標準化的作業，如薪資及存貨控制的程式可滿足很多組織的需求。利用套裝軟體的企業可節省自行開發的成本與時間。

軟體的選擇是一門學問

有時候套裝軟體不能滿足企業特定的需求，故企業自行或委託軟體公司撰寫符合特別需要的商業程式。1969年薩梅特（Jean Sammet）利用巴別塔（Tower of Bebel）描述商用語言的繁殖情形(1)。然而，之後的程式設計師似乎沒有理會薩梅特暗示性的警告—每年商業程式如雨後春筍般地成長。普通用在商業用途最多的語言有APL、BASIC、FORTRAN、COBOL、PL/I以及RPG等。

科學程式與商用程式具有不同的性質。前者注重精算，對輸入、輸出的要求不大，而後者的運算相對單純，比較注重輸入、輸出。由於硬體的進步，這種分野也不再明顯。例如，IBM 360系統即可同時達到上述精算與輸入、輸出速度的要求。

科學工業園區管理局表示由於未來數年內，個人電腦軟體將邁入快速成長期，年成長率亦高達百分之三十，國內電腦業者應掌握中文軟體有獨到經驗之優勢，積極開發中文套裝軟體打入國際市場。

科學園區管理局在一項「十六位元微電腦軟體未來發展方向與策略研究調查」中指出未來五年中，個人電腦軟體將進入快速成長期，並估計在一九八九年時八位元及十六位元個人電腦軟體，單單美國市場就將成長至八十四億美元，另在微電腦以上的電腦軟體將成長至兩百六十九億美元。(2)

面對這種擴張的市場，業者在十六位元個人電腦軟體產品的策略上可朝下列方式進

選擇軟體要了解需求

行：

• 選擇如 IBM 等大公司最新個人電腦產品爲軟體開發對象。

• 以公用軟體爲主，避開能力不足的系統軟體與具區域性色彩的應用軟體。

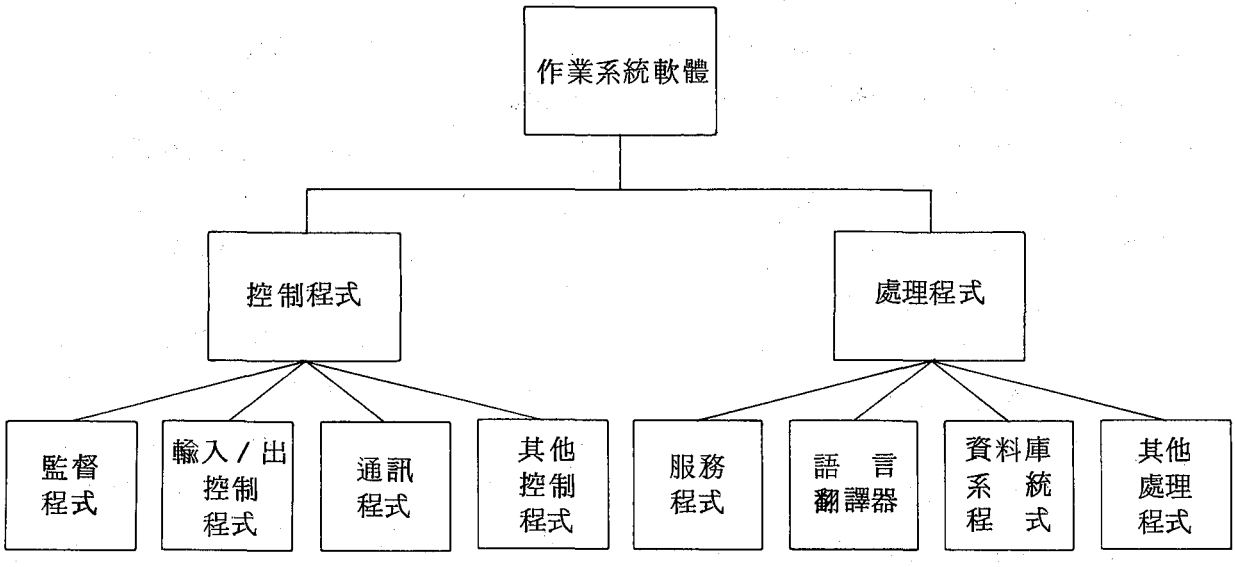
• 開發中文套裝軟體，外銷到海外華人市場。

二、系統軟體

作業系統（operating system）爲電腦的核心軟體，通稱爲系統軟體，主要功能在控制電腦資源的分配與週邊設備。一九八四年二月由工研院電子所等單位聯合開發的MITOS，是創國內電腦作業系統開發之先河(3)。由於就技術層次來看，系統軟體的設計不但需對硬體瞭若指掌，更涉及到系統科學與軟體工程，爲艱深而龐大的智慧結晶。我國電腦軟體人員多年來均以設計應用軟體爲主。這次自行發展作業系統成功，實爲我國資訊工業的一大轉捩點。

系統軟體可分爲控制程式（control programs）與處理程式（processing programs）。這兩類程式之下又有若干程式分別執行特定的功能，如圖1所示。(4)

圖1　作業系統軟體的分類



選擇軟體時風險的認明及極小化

哈佛大學派朋教授（Philip pyburn）在其著名的文章「如何突破資訊系統發展的瓶頸」中結論道：「某一企業問題及解決之道，對大多數公司而言；具有普遍性，尤其是該項問題具有高度結構化的特性，

則該公司可考慮購買套裝軟體應用程式。」(5)。本節就公司購買套裝軟體時所經歷的困難以及可能遭遇的問題加以描述，並以公司所著重的資訊系統（focus of information system）來分類，最後提出購買套裝軟體時應注意事項。

一、購買時可能遭遇的問題

軟體的適用性自然有其限制，這要看公司想要解決何種企業問題而定。蕭比利（Barry Shore）在研究數家美國公司的軟體選擇的情況之後，將所遭遇的困難彙總如下(6)：

1. 所購買的軟體並不能符合公司的需要。

2. 在購買電腦時，可供選擇的硬體有很多，然而能完全解決企業問題的軟體卻嫌不足，尤其是能等助較複雜的企業問題的軟體（如解決銷售量減少、市場佔有率減少等問題的軟體），更是缺乏。通俗而平凡的（pedestrian）軟體程式（例如，分類帳會計制度）卻充斥於市場上。

3. 要成功地執行（或稱建制，implementation）即使是最簡單的系統，例如，分類帳會計系統，所花費的時間比原先所估計的來得多。

4. 公司所期望的系統，由於相當複雜，在建制的過程中所遭遇的困難，常常不是公司內電腦人員的能力所能解決的。

5. 公司在選購電腦時未對電腦廠商的技術支援能力做深入的分析，購內電腦之後在使用時發生了問題，而求諸電腦廠商解決時，往往得不到快速而滿意的技術支援。

二、如何解決上述問題

吾人應先了解問題的本質以及了解公司所著重的資訊系統（focus of information system）。一般而言，公司所著重的資訊系統可分四類，即交易系統（transaction system）、特定產業的交易系統（trade/transaction system）、解決問題的系統（problem-solving system）以及決策支援系統（decision support system）。現就發展這四類的軟體性質分別加以討論。

1.交易系統

此類系統在同一產業的不同公司或不同產業的各個公司之間共用性很高。例如，薪資處理系統、應收帳款系統、應付帳款系統、銷售資料的記錄以及簡單的存貨記錄系統等是。

由於此類軟體程式的共用性很高，因此大多數的軟體供應廠商均著重於此一市場區隔。有些價廉的程式也可應用在微電腦上，然而這些程式缺乏彈性，很難用來等助解決公司其它特殊的（或獨特的）問題。

2.特定產業的交易系統

符合特定產業的軟體程式的應用範圍自然比上述第一類的應用範圍狹窄，潛在使用者的數目亦相對少很多。因此軟體廠商亦不熱衷於發展這類型的軟體。即使有軟體可供使用，公司在裝置上亦有困難，同時成本亦高。這類軟體程式在設計時，通常是針對一、二型的電腦而設計的，故共用性不大，這類軟體系統包括有抵押服務軟體系統、旅館及車票訂位系統、證券報價系統等。

3.解決問題的系統

在觀念上這類系統較爲複雜，設計時所應用考慮的層面亦廣。圍繞在這問題的環境也具有不定性。例如，生產排程、存貨控制、財務計劃、車輛調度等系統均有上述的特性。以替經銷商設計一套存貨控制系統爲例，系統要決定何時訂購，訂購多少、前置時間以及最終需求等。

此類系統最複雜的要算是製造需求計畫（manufacturing requirement planning）了。在此系統要決定何時訂購零件、訂購多少、安排裝配之時程等問題。

發展解決上述問題的軟體程式比解決第一、二類的問題困難、複雜得多。每種軟體應用程式的共同性很小。也就是說，一種軟體套裝程式很難適用於每個使用者。

在裝置此類系統時，組織方面的問題亦不容忽視。抗拒改變的心理會蔓延於組織內。同時訓練直線及幕僚人員所耗費的成本也不容忽視。

4.決策支援系統

所謂決策支援系統指的是「一種交談式的系統，使得使用者可以簡便地獲取決策的模式及資料，以支援釐訂半結構化之決策工作。」(7)。現今被大型公司採用的決策支援系統的軟體套裝程式包括有：交談式的財務規劃系統（Interactive Financial Planning System）、主管資訊服務（Executive Information Service）、XSIM以及EMPIRE。近年來，中小型公司已廣泛使用 VisiCalc 及DBASE Ⅱ或DBASE Ⅲ 來協助企業主管做半結構化或非結構化的決策。(8)

與上述三類系統不同的是，決策支援系統具有一般性。它可以被應用來解決廣泛的企業問題。會計系統（指軟體程式）只能來幫助解決會計問題，然而決策支援系統可用來解決各類型的問題，例如，編製財務報表、預測現金流量、分析生產力以及控制行銷績效。

這類型的決策支援系統並不昂貴、風險亦小。現今許多公司都廣泛地利用此系統進行分析與規劃。成功的例子已履見不鮮。吾人應注意的是決策支援系統的有效實施必須與組織文化配合。

三、選擇軟體時應注意事項

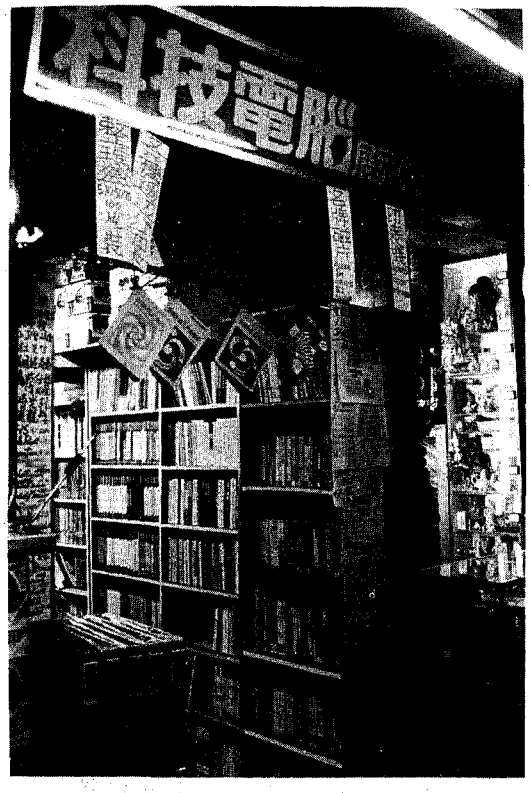
Barry Shore 根據多年經驗提出了若干建議。筆者參照其建議以及綜合近年來在系統管理期刊（Journal of Systems Management）所刊登有關選擇軟體的文章，彙總如下：

1. 管理者必須對系統設計及執行必須有適度參與，不能完全將責任推卸給經銷商。

2. 管理者必須了解公司內電腦人員或幕僚人員的能力—也就是說解決裝置軟體時所產生的問題的能力。

3. 除了交易系統及決策支援之外，在購買軟體之前要評估經銷商或賣者提供技術性服務的能力。如果仔細評估，也許可事先發現所供應的軟體系統與公司所期望的系統之間的差距。

4. 在決定引進具有解決問題能力的軟體系統之前，管理者必須詳細了解目前的資料流程（dataflow）及資訊需求。要仔細



考慮新的系統是否能滿足這項需求。

5. 購買套裝軟體兩個最大的風險在於事前未能清楚說明或認明規格（underspecification）致使成本（包括裝置成本及人員訓練等）超出了事前預估的成本（cost overruns）。

6. 組織內人員的抗拒改變會造成新系統有效實施的困難。因此，良好的規劃系統應在系統設計的早期階段，讓使用者參與。

7. 每個專案（project）都會有風險。「交易的專案」（transaction project）一般比「解決問題的系統」的風險要小。

8. 不要指望所有的軟體系統一定會大幅度的節省成本。許多系統實施所產生的利益是不太容易用金錢衡量的。

軟體評估

通常發展電腦軟體生產力的評估都是以時間及費用爲基準。一般常用的評估方式都是以多少個程式指令做爲基數，然後根據這個數字來估算所需要的人力、時間及成本。例如，科科墨模式（COCOMOL Model，爲Constructive Cost Model 之簡稱）即是先用「若干千個可交貨的原始指令」（thousands of delivered source instructions，簡稱KDSI）來計算人月（man/ month ，簡稱MM），如下列公式所示[12]：

$$MM = 2.4(KDSI)^{1.05}$$

然而，這種以程式指令爲多寡作爲依據的評估方式往往偏於主觀，很難用以比較不同案件之間之發展成本及程式生產力，同時作業使用單位也無法瞭解及審查其重要性。有鑑於此，IBM 公司的亞伯瑞奇（A. J. Alberecht）先生於一九七九年發展出一套較爲公正、客觀的軟體評估方式，彼氏稱之爲功能計點法（function point method）。[13]

功能計點法的作法是先計算某一軟體系統有多少個輸入（inputs）、輸出（outputs）、查詢（inquiries）、主檔（master files）以及有多少個不同作業項目之間的介面（application interface）需求，然後根據其各個功能需求的複雜性程度來以不同的權數（weights）及加總，最後再根據整個軟體的複雜程度加以調整而得出整個案件的功能點數。茲將評估過程概述如下：

一、軟體作業功能種類

茲將軟體作業中所牽涉到的處理功能種類及內容作一概略的說明：

1.輸入：所有在該項作業中有關的輸入資料及部份有關檔案均應計算其功能點數。例如：輸入的方式是利用終端機輸入資料（terminal screen input），亦即作業人員把某項預存在磁碟上的資料取出顯示於終端機上做進一步的資料處理。再判定其複雜性程度，賦予功能點數。

2.輸出：所有關於該項作業的輸出資料均應列入計算。例如：列印的報表輸出、終端機顯像於螢幕的輸出等。

3.檔案：無論是由該項作業所產生、利用、或是負責更新的各式檔案全部要列入計算。

4.查詢：在連線作業中綜合了輸入、輸出兩項功能但是不作更新處理而立即產生線上輸出的稱之爲查詢。

5.作業項目之間之介面需求：在不同作業項目之間如果有互相傳遞資料的需求應列爲一個介面需求功能，例如：成品出庫與應收帳款之間有介面需求。

二、作業功能複雜性分析

在各別作業功能與項目確定之後，需要研判該項作業之複雜性，然後再根據其複雜性再確定是屬於簡單（simple）、一般（average）或是複雜（complex）。其計點方式如下：

這種賦予功能點數的方式見仁見智，有賴於國內主管機關訂出標準。一旦定出規範，則國內用戶及業者均可遵循使用。

圖2　作業功能與功能點數

作業功能項目	輸入			輸出			檔案			查詢			介面		
複雜性	S	A	C	S	A	C	S	A	C	S	A	C	S	A	C
功能點數	3	4	6	4	5	7	7	10	15	3	4	6	5	7	10

S：Simple　A：Average　C：Complex

三、整體軟體系統的複雜性

整體軟體系統的複雜性取決於十四個因素：

1.作業中資訊的傳遞是否透過電訊網路？

2.作業形態是否爲分散式處理？

3.在作業設計規劃中是否要考慮回覆時間（response time）及整體輸出功效？

4.在作業操作環境中有沒有特殊需要考慮的事項？

5.作業異動數量是否居高且會影響到系統設計、執行及維護？

6.作業上是否需要線上資料輸入功能？

7.如果有線上資料輸入的需要，是否需要多重操作方可輸入資料？

8.主檔是否需要線上即時更新？

9.作業中資料處理的方式是否複雜？有無牽涉到複雜的數學公式或需間歇性的人爲參與？

10.該項作業程式是否會有套裝的需要，或是部份程式會轉移不同單位或地點而通用之？

11.該項軟體是否要考慮到未來電腦系統轉換或是便於安裝使用？

12.該項作業是否要考慮到規劃周全的起動、再起動及恢復功能？

13.該項軟體系統是否要安裝在數個不同的地點而由不同的人來維護？

14.該項作業是否要考慮到提供簡便的而能應使用者要求而進行調整的功能？

等到前面所提的各項作業功能計點及整體軟體系統複雜性之「影響點數」全部計算完畢之後（整體軟體系統之複雜性取決於上述14項的考慮，每項考慮應賦予影響點數：沒有影響爲0點，偶而影響爲1點，略有影響爲2點，有一般之影響爲3點，有重大影響爲4點，對整體功效有嚴重影響的爲5點），套用下列公式求出整體軟體系統的功能點數。

總功能點數＝全部作業項目功能點數×調整數值

而調整數值＝0.65＋（0.01×影響點數）

軟體發展的困境

建立一個電腦化的資訊系統，以解決某特定的企業問題，是件複雜而困難的事。這件工件包括了人員、過程、設備以及電腦程式的發展。然而，整合這些子系統以解決實務界的問題時，亦即把實務界的眞實問題轉換成理論上的資訊系統時，不免產生設計偏差的問題。這些誤差的產生可能是因爲實務的複雜性（complexity）以及抽象性（abstractness），也可能是因爲參與設計軟體應用程式人員的認知偏差（perceptual bias）。

資訊系統的發展過程可以細分爲若干階段（phases）或步驟，通常稱之爲資訊系統發展的生命週期。我們雖可分辨這些步驟，但應了解這些步驟並不是互相獨立的。本節討論的重點是在系統發展的生命週期中認知偏差的影響。

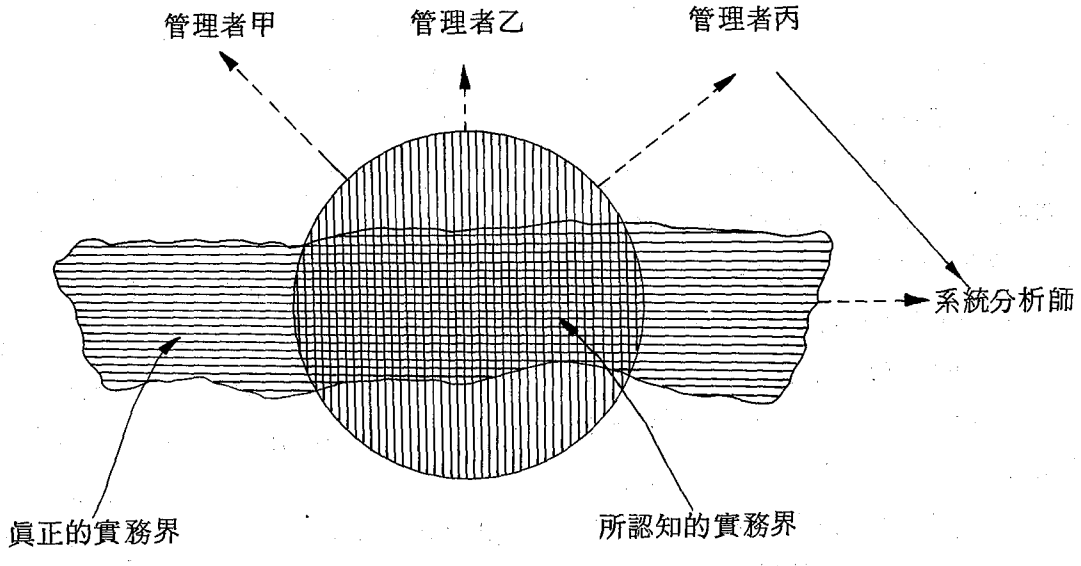
一、認知及解釋階段 (perception and interpretation phase)

組織與眞正世界（real world）第一次接觸來自於組織內管理者對外界的印象。圖3表示眞正實務界（actual real world）與所認知（perceived）的實務界之間之差別[9]。眞正的實務界並沒有邏輯上對稱（logical symmetry）的關係存在。所謂邏輯上的對稱是指：兩個變數之間具有虛僞的關係（spurious relationship）。然而，所認知的實務界比所存在的實務界有更多的對稱關係。「

所認知的系統」就是管理者認爲資訊系統正在做的，以及應該做的。這階段的轉換過程（transformation process）就是將「眞正的實務界」轉換成「所認知的實務界」。由於使用資訊系統的管理階層以及與系統分析師接觸的層面有所不同，因此這種轉換會有各種不同的方式。

爲什麼眞實的系統與所認知的系統會有所差別呢？原因在於管理者本身缺乏專業知識或只涉獵過系統的某一部份，故不易全盤掌握眞實的系統。管理者的認知也會被其過去的經驗所在左右。這就是「所需要的系統」（needed system）與最後提供的系統不能配合的原因之一(10)。

圖 3　在認知階段的偏差情形



二、分析階段 (analysis phase)

轉換過程的下一步驟牽涉到電腦系統發展人員。系統分析師與管理者的互動關係（interaction）使得電腦系統的發展更接近所認知的實務界。在第一階段所造成的若干認知與解釋的錯誤可在此分析階段加以辨識並剔除之。系統分析師也可以追溯與系統發展有關的各個管理階層，決定最原始所需求的系統。

從「提供所需系統」的觀點來看，本階段所有的活動似乎沒有正面性的效果，沒有正面性的效果，反而也許會有許多負效果。首先，系統分析師對於系統的了解經常會受管理者對眞實系統的認知所左右。系統分析師易於輕信管理者所說的。第二，系統分析師會被先前所設計的系統所左右。也就是說，如果系統分析師在最近有系統設計成功的經驗，他很可能以同樣的方式再設計此系統。第三，由於系統提供新的（或未能預期的）機會有很多，系統分析師會有使系統弄得更複雜的傾向。也就是說，系統分析師會使

系統更複雜，反而沒有簡化該系統。第四，系統分析師常將系統分成若干子系統（sub-unit ），目的在於求作業上的方便（operationalize ）。然而，當這些子系統再加以整合時，可能會產生「失眞」的現象(11)。最後，對於眞實系統各階段的定義經由不同階層的分析師的解說之後，會有所偏差。

三、設計階段 (design phase)

在此階段，系統分析師的工作是：發展出一套系統設計，並研究組織的各個層面，發展並詳細研究新系統與現有系統之相互影響（interactions）。結果是：設計出系統分析師所認知的系統（與眞實系統最相似的系統設計）。從分析階段或設計階段的變化很大。系統發展的重心從眞實世界（不論是眞正的或認知的）轉換成能描述系統活動的「過程」—這裏所謂的過程指的是爲要達成所設定的目標，在一時間序列上所從事的一連串活動（algorithm）。在這階段所發生的轉換現象是把連續的過程變成是不連續的或離散的；解決問題的程序從動態的（dynamic）變成靜態的；從重視事件或目標變成重視步驟和功能。此階段最後的結果是把所有的子系統整合成一個完整的子系統。

值得一提的是在此階段程式設計師的功能也開始發揮。系統分析師與程式設計師的職掌因公司而異。這兩種人士的良好溝通有助於減少設計的不一致現象。

四、程式撰寫及測驗 (programming and testing)

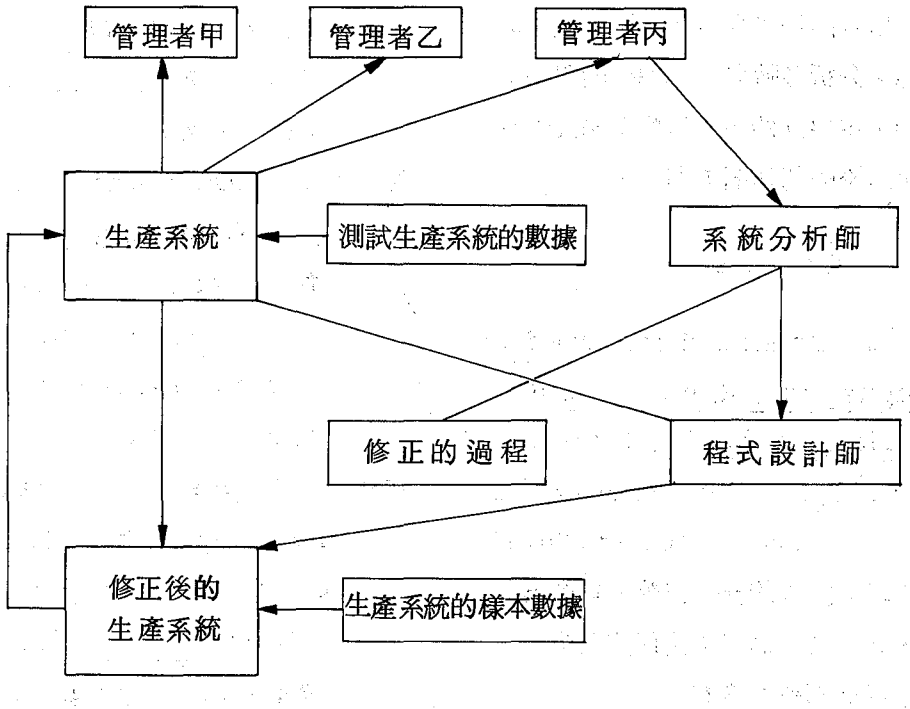
解決問題的程序加以界說之後，就可將這些程序轉換成程式。程式撰寫完成之後，程式設計師應以algorithm 爲基礎來評估程式之正確性。評估之內容包括利用由程式設計師發展的測試數據（test data）看看是否適合某些狀況。撰寫程式—測試—評估—比較的過程是週而復始的。用測試數據加以測試之後，結果將與預期的結果做比較。如果結果令人滿意，則吾人可認定此程式符合algorithm 之要求。

在發展生管系統的過程中，程式的撰寫似乎是一個相對簡單的步驟，然而以撰寫程式及程式本身之性質而論，產生錯誤的可能性仍然是很大的。也就是說，程式越大，無法發現隱含的錯誤（latent error）的可能性越大。要注意：如果改正了程式某處的錯誤，並不能保證其它的部份無誤，程式的結構具有達蹟性或相互關達性。在這一階段，將「程序」轉換成「程式」的過程中，產生認知偏差的可能性很小，但是不一致（或錯誤）的可能性仍然是很大的。

五、建制與維護階段 (implementation and maintenance phase)

完成了生產軟體系統之後，經營者將驗收系統，如果系統（不論眞正的或認知的系統）與所發展出來的生產系統有不一致的現象，則要再一次經過系統發展生命週期的各階段。圖4說明了這階段的各細節。

圖4　建制及維護的細節



結論

廣義的說，軟體包括程式、手册、文件編製，甚至包括資訊部門的政策。軟體可分爲應用軟體及系統軟體。前者的目的在解決商業上的或科學上的問題，而後者是在控制電腦資源的分配及週邊設備。由於未來五年內，個人電腦軟體將進入快速成長期，軟體業者的發展大有可爲。

企業在選擇軟體的方式不外乎購買與自行開發。在購買軟體時管理者要適度的參與，要了解經銷商提供技術性服務的能力，要清楚目前的資料流程以及軟體規格、裝是成本以及人員訓練成本。同時不要太指望軟體系統一定會大幅度的節省成本。

一九七九年ＩＢＭ公司亞伯瑞奇發展的功能計點法可做爲評估軟體的參考。該方法先計算某一軟體系統有多少個輸入、輸出、查詢、主檔以及介面。然後根據其各個功能需求的複雜性程度乘以不同的權數及加總，最後再根據整個軟體的複雜性程度加以調整而得出整個案件的功能點數。

如果企業自行開發軟體系統要注意開發出來的系統是否合乎管理者或使用者的需求。管理者、資訊系統分析師以及設式設計師的認知偏差是造成「所設計出來的軟體並非是所預期的」的主要原因。

管理者必須加强有關資訊的事業知識，因爲認知偏差對資訊系統發展生命週期各階段的影響是很大的。以認知偏差的程度而言，在系統發展的早期階段，認知偏差的可能性很大。將作業程序（algorithm）轉換成程式（programs）之後，認知偏差的可

能性變小，但是不一致性仍然很大。本文對軟體發展的困境的研究是以資訊系統發展各階段爲經，以各階段管理者與資訊人員之認知偏差爲緯，分析各階段認知偏差的現象，程度及原因，主要目的在於提醒有關人士正視這個易受忽略的認知偏差問題。

註　釋

註1：J. E. Sammet, Programming Languages：History and Fundamentals (Englewood Cliffs, N. J. Prentice-Hall, Inc.,) front cooer。Babel，塔名，據舊約聖經記載，築於巴比倫，企圖使其高與天齊而終告失敗，意指好高騖遠的空想。

註2：74年1月27日中央日報。

註3：73年2月7日聯日報。

註4：D. Hussain and K. M. Hussain, Information Processing Systems for Management, (Homewood, Il,: Richard D. Irwin, Inc.,) P. 50。

註5：Philip Pyburn, "Breaking the Systems Development Bottleneck," Harvard Business Review, (March-April, 1983), P. 136。

註6：Barry Shore, "Identifying and Minimizing Risks in Software Selection," Journal of Systems Management, (Aug. 1984), PP. 26～32。

註7：R. H. Sprague, "Development of Decision Support Systems," MIS Quarterly, Vol. 6, No. 4, (Dec. 1980), PP. 1～26。

註8：R. J. Mann and H. J. Watson, "A Countingency Model for User Involvement in DSS Devecopment," MIS Quacterly, Vol. 8 No. 1, (Mar, 1984), P. 27。

註9：J. W. Spence, "Software Development Dilemma: A Case of Perceptual Bias, "Journal of Systems Management, (Aug. 1984). P. 34。

註10：R. Ackoff, "Management MIS-information Systems," Management Science Quarterly, (Dec. 1967). PP. 147～156。

註11：R. Bostrom and S. J. Heinen, "MIS Problems and Failures: A Socio-Technical Perspective—Part I, "MIS Quarterly, (Sep. 1977), PP. 17～32。

註12：Boem, "The Basic COMOCO Model, "Software Engineering Economics, Chap. 5。

註13：詳見文北崗，「資訊週刊」，第78期，（台北：工商時報，73年1月）。