軟體再用效益演變之探討

朱文禎
高苑技術學院資訊管理系
國立政治大學資訊管理系所

楊建民
國立政治大學資訊管理系所

摘要

隨著軟體發展與演進，「軟體危機」因應而生，而軟體再用是解決「軟體危機」重要因應之道，因此探討軟體再用效益的演變將有助於產官學研在推動軟體再用相關政策之研擬參考。

以往的軟體再用效益的研究大多強調局部性財務效益，而且是橫斷面研究。本文則較廣泛的就系統、程式、應用、管理、組織與產業等七大構面 42 個關鍵效益進行縱貫性研究。經由比較 1997 年上半年和 2000 年下半年兩個時期，國內主要軟體業者和專家學者的訪談和階層程序分析法問卷，歸納並探討軟體再用關鍵效益在國內的變化趨勢。

本研究結果顯示(1) 軟體再用效益七個構面中，在重要性變化以系統屬性增加最多(0.068)，其次為產業屬性(0.021)；專案屬性的重要性減少最多(-0.054)。(2) 42 個關鍵效益中，減少系統出錯及跨平台的成本，增加元件品質等關鍵效益更受到重視。另一方面，專案中發展人數減少、開發類似軟體時間減少等效益的重要性則逐漸減少。亦即軟體再用的效益中，系統的穩定性、元件質的提昇和量的增加等關鍵效益更加重要，而且逐漸從重視個別專案效益，到組織外的跨專案效益乃至於整個產業的效益，以跨組織合作和軟體產業整體資源來面對日漸擴大和複雜軟體市場需求。

關鍵字：軟體再用效益、階層程序分析法、軟體危機、縱貫性研究

The Evolution of Software Reuse Effects

WenChen Chu
Dept. of the MIS
Kao Yuan Institute of Technology
Dept. of the MIS
National Chengchi University

JanMing Yang
Dept. of the MIS
National Chengchi University

Abstract

Software reuse plays a vital role when software crises occur along with the evolution of software development. It is expected that policy makers of software reuse in software industry will be benefited by this study.

As most researches in terms of software reuse effects focus on project and cross section effects, the objective of this paper, however, involves a longitudinal study of the evolution of software reuse effects. It compares the outcome of analytic hierarchy process drawn from

the questionnaire done with experts in 1997 and 2000. In addition, an evaluative model of software reuse effects with 7 dimensions and 42 criteria is included in this study.

The findings indicate that (1) the system (0.068) and industry (0.021) dimensions are the leading main proportions; meanwhile, the project (-0.054) dimension is one of the main proportions that are decreasing in the evolution process among 7 dimensions. (2) Low frequency of system errors, low porting costs and high components quality are more emphasized. The size of team members working on the projects and the time spent on the project are losing their importance in the evolution among 42 criteria. In other words, more stable systems, high quality and quantity components have got more emphases than ever. Furthermore, the effects of industry levels are evaluated prior to those of project levels. The corporations among the software industry are about to ally to meet the more complicated and extensive market needs.

Keywords: software reuse effects, analytic hierarchy process (AHP), software crises, longitudinal study

壹、 緒論

全球軟體產業發展和演進歷史當中，資訊系統應用層面從早期的電子資料處理(EDP)，到管理資訊系(MIS)、決策支援系統(DSS)、主管資訊系統(EIS)和知識管理系統(KMS)。軟體發展方法也從早期認為是一種手工藝術到軟體工程的提出，從結構化開發設計，然後物件導向開發設計，再演進成元件式軟體發展(component based development，CBD)。進入二十一世紀，電子商務和電子企業帶動著新的產品與服務方式，造成軟體市場規模擴大，軟體複雜度亦增加，「軟體危機」不但沒有解除，反而有日漸擴大趨勢(Laudon & Laudon, 2000；Lamela, 2000)。

軟體再用是一種能將軟體生產方式從手工轉變為工程的方法技術，能有效提高軟體生產力，是因應軟體危機的有效解決方式。軟體再用的觀念，可以追溯自 1968 年 McIlroy 在德國 NATO 研討會中，公開提出以工業化方式大量生產軟體開始，至今已經超過三十年。然而直到 70 年代末期和 80 年代初期才有一些研究團體和報告出現，如日本進行軟體工廠計畫，美國的 STARS 推動的軟體再用計畫，這一波的再用是以大型主機的程式碼再用為主（Gall et al. ,1995）。

第二波自 1991 年開始舉辦一系列的國際軟體再用工作坊(International Workshop of Software Reuse, IWSR)，將軟體再用議題蔓延成為資訊界全球化議題；而自 1993 年網際網路上 3W 瀏覽器(Mosaic)發明後，帶動上網人潮和電子商務及電子企業的興起，使得企業對軟體的需求性大為增加和迫切。為能有效解除「軟體危機」，軟體先進國家莫不積極投入於開發與推廣軟體再用技術。例如：美國 NASA 的軟工實驗室推動經驗工廠計劃，將團體的經驗互相學習後，機構化成為再用元件以利後續軟體開發和維護；美國國

家標準與技術委員會(National Institute of Standards and Technology, NIST)自 1995 年起推行先進科技計劃，投入大量經費，於協助軟體產業發展軟體元件技術；歐洲產業委員會資訊科技計劃 ESPRIT 自 1994 年起支援發展軟體工廠技術；日本在 1993 年多家公司共同組成 Intelligent Pad 協會，發展超媒體(Hypermedia)物件導向軟體再用技術(楊建民，1997)。同時，國際級資訊大廠也陸續推出元件組裝和執行環境平台、元件儲存庫、應用框架等元件式軟體技術產品。

在全球電子商務風潮下，ComponentSource 在 1995 年成立軟體元件電子市集提供軟體元件交易機制，促進軟體元件的流通，也開啟了軟體再用新紀元。其後，在 1999 年陸續有 ComponentPlanet、Flashline 等軟體元件電子市集成立。根據 IDC 分析美國軟體元件市場將從 5 億 1 千 6 百萬美元（1999 年）成長到 27 億美元（2004 年）年複合成長率達 40%。再用碼的錯誤率僅有新碼的四分之一，而且再用元件可以使最終應用產品的交貨時間加快百分之四十。Gartner Group 更預測在 2003 年將有 70%的新開發軟體將由元件組裝而成，而軟體再用技術、元件式軟體、及物件導向技術等，將成為市場的主流(朱文禎等人, 2001)。

在國內方面，自 1993 年開始經濟部委託資策會執行連續兩期的「軟五計畫」：其中第二期自 1997 年到 2001 年，這時期正值台灣電子商務引入到成長階段，軟體業者面對相關電子商務軟體和企業電子化軟體，供不應求且需求多樣化的環境下，希望引進軟體再用技術，以更有效率的方式提供更多樣化客製化和更有價值的服務 。因此，此四年計畫主要著重於軟體再用關鍵技術的研發，已經完成快速網站建置工具與元件組裝平台(NeW Platform)等關鍵技術並積極推廣軟體元件和工具供業界使用(楊仁達，2000)，對軟體再用的推廣有相當的助益。此外資訊軟體協會「軟體產業服務團」辦理「軟體工業生產力提升輔導計畫」，協助軟體業者利用元件軟體技術來提升生產力、改善經營體質並建立軟體元件資料庫。同時，透過網際網路進行元件試用的國內廠商也陸續出現。

有關軟體再用產生效益的研究不少，然而目前有關軟體再用效益衡量的研究中，主要均以局部性效益衡量為主，而且屬於橫斷面研究。然而，由於軟體元件的特性，將使得再用效益不限於組織內，在網路正向外部性下，將擴及組織外乃至整體產業。因此，本文以較全面性的軟體再用效益評估模式，縱貫性比較兩個時期主要軟體業者和專家學者的訪談和階層程序分析法問卷，歸納並探討軟體再用效益的變化程度和演變方向。

本文分為五部份，第二部份探討軟體演進與危機、軟體再用、軟體元件統治結構及軟體再用效益評估模式，第三部份為研究設計，第四部份說明和討論軟體再用效益演變，第五部份為結論和建議。

貳、文獻探討

一、軟體演進階段與危機

全球軟體產業已經超過五十年的歷史，其演進大致可分為以下三個階段：

第一個階段從 1950 年到 1960 年, 稱為電子資料處理(Electronic Data Processing, EDP)時代：這時期軟體是依附在大型主機上，絕大多數是量身訂作的專案軟體，最初是用於會計部門的資料處理應用程式，應用的重點在個別部門的自動化和電腦化，開發程式以組合語言和 FORTRAN 為主，軟體的開發被視為是一種手工藝術。面臨的問題是資源稀少昂貴，軟體開發時間長又難以維護，軟硬體費用比例平均是 20% 比 80%，軟體未得到應有的重視。

第二個階段從 1960 年到 1980 年稱為管理資訊系統(Management Information Systems，MIS)時代：軟體被當作是一種商品和硬體分開銷售，而且可以重複銷售。開始有跨組織的資料庫的應用，應用的重點在資訊化和合理化，開發程式以高階語言為主，軟體的開發被視為是一種工程，強調結構式開發和設計，有軟體專案團隊的管理。面臨的問題除了資源稀少昂貴, 軟體開發時間長, 軟硬體費用平均比例是 80% 比 20%，軟體維護費用增加的速度更是超過開發費用，佔軟體費用 70%。此外，達不到預期資訊化目標、無法滿足使用者需求，造成所謂的「軟體危機」。軟體危機是軟體供給面的量和質均不足，無法滿足需求，可以歸納為以下幾點：軟體開發時程總是落後，費用需要追加；開發的系統規格不符適用者需求，難以運作；系統維護困難，需要花費大量人力時間（Mizu, 1983；Paul, 1994；Boehm, 1981）。

第三個階段從 1980 年以後稱為知識化和網路化時代：大眾化軟體產品隨著個人電腦出現而普及，除了量的增加外，開始有決策支援系統和知識管理系統的應用，而網際網路出現更帶動企業電子化的加值應用，應用的重點是知識化和無所不在的服務。軟體開發採用物件導向技術和元件式軟體發展(Component Based Development, CBD)等軟體再用技術，雖然對解決「軟體危機」有幫助，但是面臨的問題是對多樣化、高品質軟體的需求持續增加，軟體元件技術變動快速，元件規格或標準尚未建立或統一，元件供不應求，亦即軟體再用仍然處於不斷演進當中(Laudon & Laudon, 2000)。

二、軟體再用

軟體再用所展現的效益包括系統發展時程縮短、成本降低、生產力提升、容易客製化及維護品質提昇，同時領域知識與技術得以累積、再用和分享等(Barnes & Bollinger，l991；Poulin et al. ，1993；Gaffney & Durek，l989；楊建民，1997；朱文禎等人，2001)。

軟體再用是指軟體發展人員，包括分析、設計、程式撰寫與測試人員，重新使用先前已開發過的軟體資源，到新系統開發程序的各階段中的一種工程方式，所謂可再用的軟體資源包括程式碼、軟體需求、分析模組、設計與測試個案等實體或電子檔形式的資訊，可以說軟體開發程序的各階段產品都可以再用(Jacobson, et al., 1997)。

軟體再用的觀念和推動在物件導向技術之前就有，但是隨著物件導向技術和元件式軟體技術演進才逐漸發展，因此，大多數仍然以物件導向分析設計配合元件軟體技術為主。採用軟體再用的軟體工程活動中包括領域工程與應用系統工程兩類。依照領域模式進行系統的分析設計，並以元件系統為出發點，尋找、修改與整合元件。這種方式與傳統方式最大的不同在於傳統專案是從事程式撰寫，而軟體再用則是尋找合適元件進行修改整合(Jacobson et al. ,1997)。

因為軟體新技術如分散式物件(如 CORBA，DCOM 等)帶來了新功能，增強了企業的競爭力，卻造成系統複雜度的大幅上升，隨著技術的發展進步，元件式軟體技術（如 EJB，J2EE，COM+，WINDOW DNA/.NET 等）使系統複雜度再度降低。元件式軟體技術使得系統發展時程縮短、成本降低和容易維護等再用效益得以呈現。

軟體元件是由物件導向原理演進而來，提供元件式軟體技術的基礎。根據 webster 字典的定義，元件是一種組成的成份(constituent part; ingredient)。而軟體元件是一種預先寫好程式碼的部份，具有高內聚力(cohesion)和低耦合力(coupling)，只能經過定義良好的界面來傳送封裝在裡面有意義的服務或功能。此外，元件標準是多個標準並存的局面，關於元件市場標準(defacto standard)或產業標準(dejure standard)，可以分為三種層次：元件界面標準，遵守 COM、 Java 或 CORBA 三種標準之一； 元件架構標準：遵守.NET/COM、J2EE/Java 二種標準之一；領域元件標準：各個產業領域的標準，如資訊電子業 Rosettanet 標準等，其他產業領域標準由 OMG(Objects Management Group)的多個標準化小組和軟體業制定當中。

元件依照應用層次分為共通性元件(common component)和領域元件(domain component)和應用系統框架(framework)。共通性元件和領域行業無關，包含使用者界面控制、文件檔案管理、訊息服務、搜尋壓縮等公用程式。領域元件是特定領域的元件，如金融財務元件，財務元件，各產業逐步訂定標準後，將會帶動領域元件使用的成長。應用系統框架可加入新元件或擴充原來元件快速產生新應用系統(Sprott, 2000；Lamela, 2000；楊仁達，2000)。

由於元件式軟體技術，使得系統發展時程縮短、成本降低、生產力提升、容易客製化及維護。然而，元件式軟體發展(CBD)過程中，需要尋找適當元件進行修改整合，如何促進擴大元件供應，以利後續發展過程中的修改和整合，就成了元件式軟體發展最基礎核心的問題。有關元件供應來源進一步探討元件統治結構(components governance

structures)。

三、軟體元件統治結構

由於軟體專屬性高、複雜和不易分割特性，在軟體元件技術推出之前，軟體專案或是軟體產品除了在組織內自行開發外(單邊統治)，就是以專案或產品委外（雙邊統治)。隨著軟體元件技術的推展和應用，元件標準的制定，除了在組織元件庫或委外合作獲得元件外，尚可透過公開市場元件電子市集如 ComponentSource、 ComponentPlanet、Flashline、ObjectTools、CodeMarket 等獲得元件再加以切割、改變和組裝。

Williamsom(1979)將商品交易的六種型態用交易的次數與投資的異質性為構面，對應到四種統治結構。透過適當契約的訂定可以發展出最適當的統治結構，可以使企業資源統治成本降低。而本研究將軟體或元件交易的六種型態用交易的次數與軟體或元件特性為構面，對應到三種統治結構（表 1)。軟體公司透過適當契約的訂定可以發展出最適當的元件統治結構，使元件統治成本降低。

如特殊軟體專案或專屬規格軟體元件專屬性很高，且交易頻次也很大的情況下，則購買過程中所需的搜尋、議價與監督成本會很大，則向外部購買所產生的交易成本，將遠高於其因內部專業生產而獲得的優勢與利益，亦即交易成本與生產成本加總的結果並非最有效率，因此宜採單邊統治（內部化）的型態。如一般軟體專案或一般規格非標準化軟體元件為混合性（半標準品)，且交易頻次很大的情況下宜委託外人生產，以發揮專業效果，但為降低交易成本，應和對方維持長期的合作關係，增加彼此的信任感，以降低投機行為，因此採雙邊統治結構方式。若交易頻次很低時，向外界購買將面臨高昂的交易成本，但自行生產可能未到達規模經濟，因此會有第三方中介團體的出現，以其專業的能力協助雙方進行交易，降低彼此的交易成本，形成三邊統治的型態。

最後，當交易商品為標準化軟體元件時，透過市場競爭可使廠商的生產成本降至最低，同時交易過程中產生的交易成本也甚低，故採市場統治結構。而軟體元件市集是由中立第三者以其專業的能力協助雙方進行交易，可以視為市場統治或三邊統治(Williamson，1979；吳思華，1998)。

統治結構可以透過不同資訊系統加以展現，由於軟體元件可以經過網路遞送，使得以資訊系統來對應統治結構時，完整的包含交易資訊服務和元件遞送服務。

表 1 元件交易統治結構類型

		投資特性（資產專屬性/軟體元件特性）		
		非特定性	混合性	專屬性
交易頻次	偶而交易	市場統治	三邊統治 存在中介團體 (新古典契約)	
	經常交易	(古典契約)	雙邊統治 合作網路 (關係	單邊統治 內部組織 契約)

[資料來源：修改自 Williamson(1979)]

跨組織資訊系統可以分為電子市集系統、跨組織資訊鏈結和電子階層。其中電子階層從組織內部獲取元件是一種垂直整合的方式；跨組織資訊鏈結(合作網路)是一種雙邊資訊共享的投資，其供需關係是供需雙方事先確定的，是一種連接供應商和採購者的行銷或後勤跨組織資訊系統；電子市場是一種多邊資訊共享的投資，其供需關係是透過市場系統運作後才建立的，可能是產業內平台(一個或少數買方或賣方主導)，虛擬系統(無明顯主導者)和電子市場(第三者主導)。表 2 列出交易成本的四種統治結構、相關學者和本研究提出的各種資訊系統形式間關係。（Williamson，1979；Bakos，1991；Choudhury et al. ，1998；Rindfleisch & Heide，1997；Malone et al.，1987，1989）。

表 2 交易成本理論的統治結構和資訊系統形式

學者	統治結構和資訊系統形式		
Williamsom(1979)	單邊	雙邊	三邊或市場
Malone 等人(1987,1989)	電子階層	電子市場	
Bakos(1991)	N/A	跨組織資訊鏈結	電子市集系統
Choudhury 等人(1998)	N/A	行銷或後勤跨組織資訊系統	產業內平台、虛擬系統、電子市場
Rindfleisch & Heide(1997)	垂直整合	垂直跨組織關係或水平跨組織關係	
本研究	電子階層	跨組織資訊鏈 (合作網路)	電子市場

(資料來源：本研究整理)

在有限理性下，組織是效率的追尋者，降低交易成本和效率的追尋是一致的。電子市場和合作網路相較於電子層級有更低的交易成本，由於電子網路相連，造成通訊、 經紀和整合的效益，而且具有實體和虛擬的網路外部性、學習效率和規模效率，可以整合供應和需求雙方流程，擴大買方和賣方交易，使得願意使用元件式軟體發展的人愈來愈多，在報酬遞增下產生正回饋，又吸引更多的開發者/使用者加入，電子層級的垂直整合將會被合作網路和電子市場所取代。因此隨著元件標準化和產業分工體系建立，元件統治結構，逐漸從單邊統治到合作網路的雙邊統治和市場統治結構(Williamson, 1992；

Malone et al. , 1987 & 1989)。由以上探討可知不同的軟體元件統治結構將影響公司成本和效率，亦即軟體再用的效益應考慮組織內、跨組織到產業不同層次。

四、軟體再用效益

目前有關軟體再用效益衡量的研究中，主要均以局部財務性的成本效益分析和再用率的量測為主。Gaffney & Durek(l989)提出軟體再用的成本效益分析模式，以再用率和再用成本來提供軟體開發成本及組織生產力的估算。Barnes & Bollinger(l991)提出再用投資的品質評估模式，係以使用軟體再用技術的利潤和軟體再用技術的投資的比值來表示投資的品質。Poulin、Caruso 及 Hancock(1993)所提出的企業軟體再用量測(Business Reuse Metrics)，係透過軟體再用比例、軟體再用規避的成本、附加價值、額外的開發成本來評估投資報酬率。

根據軟體再用的效益研究結果指出軟體再用可以降低專案成本、減少開發所需時間。影響專案成本的程式因素中，除了軟體的形式與程式撰寫所使用的語言外，其他因素為軟體複雜度、重複使用性和可靠度，這些因素可以因可再用軟體具備複雜度低、可靠性高及其可重複使用的特性而克服，同時品質也得以提昇（Boehm，1981；Khairuddin & Key，1995)。

除此之外，影響專案成本的因素包括專案與顧客的介面、需求定義的程度、需求更動的程度等，這些屬於溝通的問題影響專案的規劃與設計，間接影響專案成效。軟體再用可以加速雛形系統的發展，增進溝通成效，並由現有元件的功能說明文件之輔助，有助於需求定義的明確化，進而減少需求變動的程度。另外由於再用元件庫的建立，組織知識可以累積再用，使得因組織架構或是人員變動的衝擊可以降低(Boehm，1981；Gaffney & Durek，1989；Poulin，1997)。

資訊系統對組織運作績效的影響很大，尤其是企業組織採用企業資源規劃整合所有的運作之後，資訊系統對組織的影響變得更大。相對的組織外在環境的變動也越來越大，組織為了因應環境的快速變化，實在有必要加速資訊系統發展的速度。軟體再用是資訊系統的基礎建設，透過軟體再用可以加快資訊系統開發，提昇資訊系統品質，將進一步提昇企業產品品質，降低產品成本，增進營業利潤（McFarlan，1984；Lozinsky，1998；Stinchcombe，1990)。

軟體再用牽涉跨程式語言、技術平台、跨組織的複雜性技術，藉由元件架構標準與界面標準的建立可以有效克服上述問題，並且降低產業進入門檻，進而擴大可再用元件供應來源與分擔元件開發成本，並促進產業分工體系的建立。技術與元件供應者為了增進成本效益，有必要降低技術移轉的困難度來促進使用量。而整體產業共同發展軟體再用會使相關技術與管理方式更加成熟，使得企業組織願意提昇再用技術的層次，使得軟體人員素質獲得普遍的提昇，形成一種正向的回饋。

因此，軟體再用效益不僅是在局部財務面效益，更會影響人員、組織及產業等層面。楊建民（1997）和朱文禎等（2001）採用包含系統、程式、專案、應用、人員、組織績效和產業屬性，共七大構面的軟體再用關鍵效益架構，配合階層程序分析法來建立軟體再用效益評估模式。本文即引用此較為全面性的軟體再用效益評估模式，來探討軟體再用效益的演變。

參、研究設計和方法

一、研究架構和研究對象

軟體再用關鍵效益架構，分為系統屬性、程式屬性、專案屬性、應用屬性、人員屬性、組織績效屬性及產業屬性，共七個主要構面所組成(圖 1)（楊建民，1997；朱文禎等人，2001)。依據此構面設計成 AHP 問卷的第一層級，四十二個關鍵效益為第二層級(附錄一)。評估軟體再用效益是屬於較專業的議題，因此研究對象以軟體產業資深從業人員、專家和學者為主。

本研究分別在 1997 年上半年和 2000 年下半年對代表性專家和學者做訪談和問卷調查。在 1997 年上半年經由資策會建議 15 位專家學者，包含 6 位業界專家和部門主管及 9 位資訊相關系所教授，以 AHP 問卷評估各項關鍵效益的優先順序與權重，並進行階層程序分析法。在 2000 年下半年，再由軟體發展協會提供的軟體元件發展業界學界名單中，挑選 15 位專家學者，包含 8 位業界專家和部門主管及 7 位資訊相關系所教授評估各項關鍵效益，並進行階層程序分析法後，比較出軟體再用效益關鍵效益的重要性變化方向和程度。

二、階層程序分析法

階層程序分析法（Analytic Hierarchy Process，AHP）是由 Thomas L. Satty 在 1970 年代提出來的(Satty, 1980)。乃是將欲研究的複雜系統，分解成簡明的層級結構系統，如將目標分解成諸評估要素，再分解成許多解決方案；接著透過成偶比對（pairwise comparison）而求得各層級要素或方案的優先順序，最後再經綜合（synthesis）及比較等步驟，以決定最佳方案的一套理論。曾有用於投資計畫的評估（翁俊興，1983)、資訊服務供應商評選決策（施穎偉，2000）和資源分配（張紹文，1983)。本研究用七大構面的軟體再用關鍵效益架構來建立軟體再用效益評估模式，屬於決定優先次序和績效衡量的問題適合用階層程序分析法。

將軟體再用關鍵效益，進行成偶比對評估。其評估尺度劃分為同等重要、稍重要、頗重要、極重要、及絕對重要，分別賦與 1，3，5，7，9 的衡量值，兩個方向共九個等級。 在優先向量之解法方面，以 NGM(Normalization of the Geometric Mean of the Rows)法，將各列元素相乘，取其幾何平均數，再予以常化而得。

一致性比率 C.R.的計算公式

C.R.＝C.I./R.I.，其中：一致性指標 C.I.＝(λ_{max} -n)/(n-1)，

若 C.I. ≤ 0.1，則一致性程度視為可接受。

而 λ_{max} 為最大特徵值（Eigen Value）， n 為矩陣維度

R.I.（Random Index）為隨機指標，由隨機產生的倒值矩陣之一致性指標，其值隨矩陣階數的增加而增加。

本研究用 Expert Choice 軟體是以 Inconsistency Ratio 來表示一致性比率，所以在此 C.R.即為 Inconsistency Ratio 值。

肆、 軟體再用效益演變和討論

一、1997 年軟體再用效益

影響軟體再用關鍵效益七大構面。經由 15 位專家學者問卷分析，再由 Expert Choice 執行運算的結果計算的結果，獲得成對比較矩陣的優先向量及該矩陣的 Inconsistency Ratio 量測值（表 3, 1997 年部分），其中 Inconsistency Ratio 的值為 0.02，並未超過 AHP 方法的臨界值 0.1；亦即本份問卷關於軟體再用關鍵效益方面，所蒐集到的專家學者意見尚屬一致。

七大構面中以應用屬性對軟體再用效益評估最為重要、其次為系統屬性，而程式屬性之重要性最低。由軟體再用關鍵效益架構圖(圖 1)中，各七大構面中關鍵效益，由 Expert Choice 執行運算的結果，可獲得七個成對比較矩陣的優先向量及各矩陣的 Inconsistency Ratio 量測值（均小於 0.1）。再換算成整個樹狀節構的最底層之 42 個關鍵效益絕對權重(附錄二，1997 年部份)。

在 1997 年部分，前十個最重要的關鍵效益中（表 4），可發現軟體再用效益主要著重在應用屬性、系統屬性、組織績效屬性上。就十大關鍵效益中，應用屬性與系統屬性各佔三個，組織績效屬性佔了二個，而人員屬性及產業屬性各佔一個；至於專案屬性與程式屬性則未包含在其中。

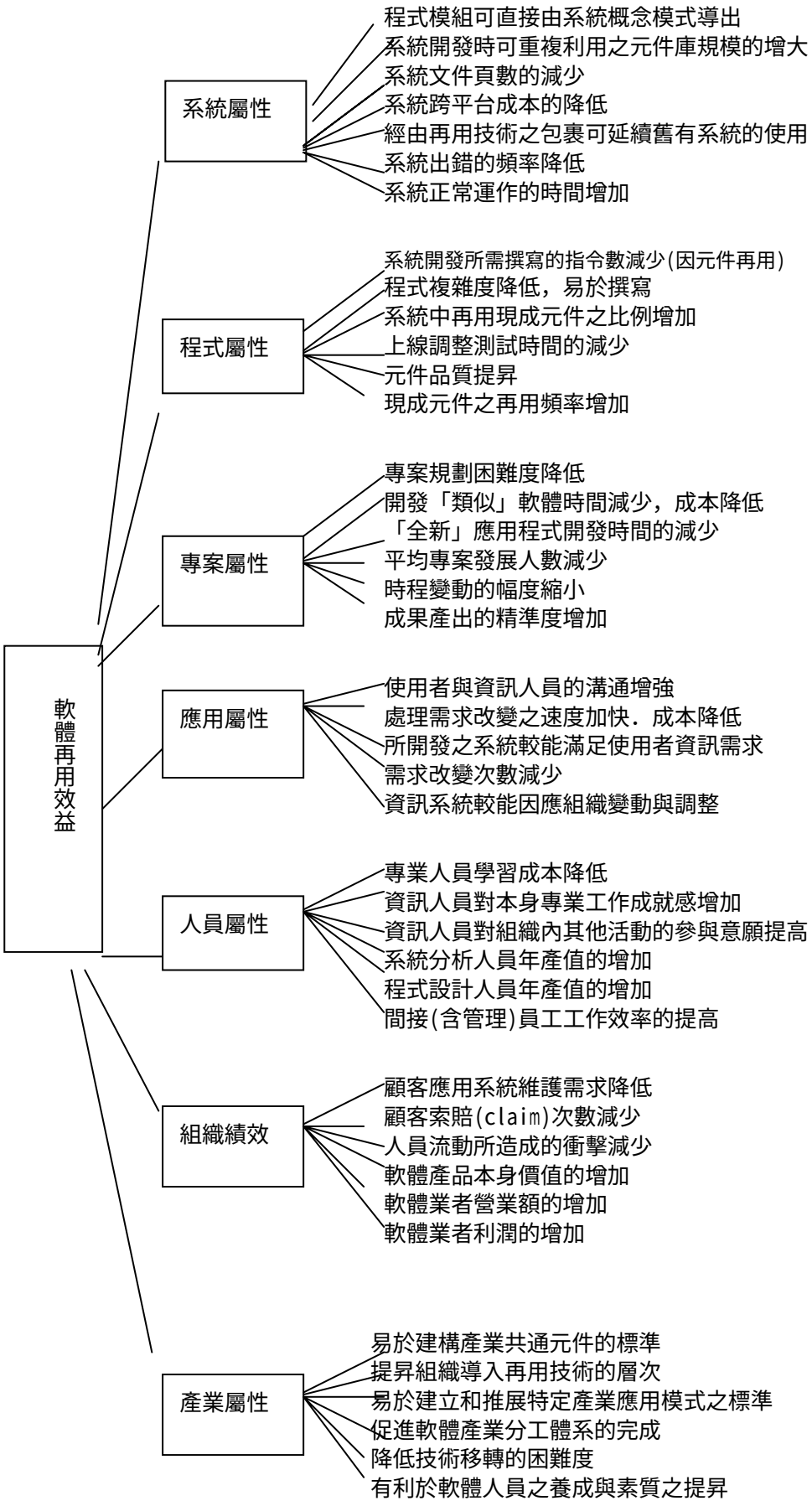

圖 1: 軟體再用關鍵效益架構圖（楊建民，1997；朱文禎等人，2001）

表 3 軟體再用效益七大構面順序表

構面	系統屬性	程式屬性	專案屬性	應用屬性	人員屬性	組織績效屬性	產業屬性
權重(1997 年)	0.173	0.11	0.137	0.181	0.134	0.152	0.114
順序(1997 年)	2	7	4	1	5	3	6
權重(2000 年)	0.241	0.13	0.083	0.151	0.099	0.160	0.135
順序(2000 年)	1	5	7	3	6	2	4
權重變化	0.068	0.02	-0.054	-0.03	-0.035	0.008	0.021

(資料來源：本研究整理)

表 4 軟體再用關鍵效益（1997 年前十大）

順序		權重		權重變化	關鍵效益	構面
1997	2000	1997	2000			
1	1	0.059	0.055	-0.004	所開發之系統較能滿足使用者資訊需求	應用屬性
2	11	0.041	0.031	-0.010	處理需求改變之速度加快，成本降低	應用屬性
3	7	0.038	0.035	-0.003	系統開發時可重複利用之元件庫規模的增大	系統屬性
4	13	0.036	0.029	-0.007	程式模組可直接由系統概念模式導出	系統屬性
5	2	0.034	0.055	0.021	系統出錯的頻率降低	系統屬性
6	24	0.034	0.020	-0.014	使用者與資訊人員的溝通增強	應用屬性
7	4	0.034	0.036	0.002	軟體業者利潤的增加	組織績效屬性
8	5	0.033	0.035	0.002	易於建構產業共通元件的標準	產業屬性
9	15	0.033	0.027	-0.006	人員流動所造成的衝擊減少	組織績效屬性
10	23	0.033	0.020	-0.013	間接人員（含管理）作業效率的提高	人員屬性

(資料來源：本研究整理)

由此一結果可發現軟體再用在 1997 年，在應用屬性及系統屬性上有比較明顯的助益，而其組成關鍵效益為：所開發之系統較能滿足使用者資訊需求、處理需求改變之速度加快成本降低、使用者與資訊人員的溝通增強、系統開發時可重複利用之元件庫規模的增大、程式模組可直接由系統概念模式導出、系統出錯的頻率降低，從這些效益內容中可發現能為軟體開發人員帶來快速修改應用系統以及時因應環境變動的需求。

而從組織績效屬性的關鍵效益：軟體業者利潤的增加、人員流動所造成的衝擊減少、可發現其能為軟體業者帶來較高的營業效益及減少因人員流動所帶來的困擾。

二、1997 年到 2000 年軟體再用效益之演變

2000 年再經 8 位業界專家和部門主管及 7 位資訊相關系所教授問卷和訪談後，經 AHP 計算後扣除未能通過層級一致性檢定者，得到 13 位專家學者的整體優先向量其七個構面的權重、順序和權重變化(表 3，2000 年部份)。七個構面中前三個和 1997 年一樣但次序稍有不同，主要著重系統屬性、組織績效屬性和應用屬性。在權重變化以系統屬性增加最多(0.068)，其次為產業屬性(0.021)、程式屬性(0.02)和組織績效屬性(0.008)。而專案屬性的權重減少最多(-0.054)，其次為人員屬性(-0.035)和應用屬性(-0.03)(圖 2)。

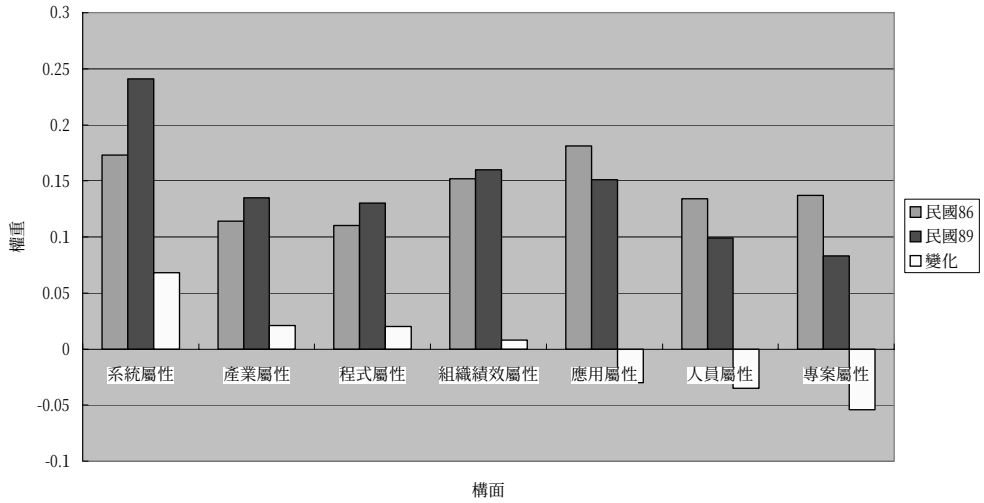

圖 2 軟體再用效益構面權重變化

2000 年部分，前十大最重要的關鍵效益中(表 5)有五個和 1997 年前十大關鍵效益一樣，只是次序稍有不同，包括所開發之系統較能滿足使用者資訊需求；系統出錯的頻率降低；系統開發時可重複利用元件庫規模的增大；軟體業者利潤的增加；易於建構產業共通元件的標準。新進入前十大的五個關鍵效益為：系統跨平台成本的降低、可延續舊有系統的使用、系統正常運作的時間增加、元件品質提昇、軟體業者營業額的增加。同時這五個關鍵效益也是屬於權重變化方面重要性增加的前十大關鍵效益，其他重要性增加的前十大關鍵效益包含為：系統出錯的頻率降低、現成元件之再用頻率增加、降低技術移轉的困難度、促進軟體產業分工體系的完成、軟體產品本身價值的增加(圖 3)。

表 5 軟體再用關鍵效益（2000 年前十大）

順序		權重		權重變化	關鍵效益	構面
2000	1997	2000	1997			
1	1	0.055	0.059	-0.004	所開發之系統較能滿足使用者資訊需求	應用屬性
2	5	0.055	0.034	0.021	系統出錯的頻率降低	系統屬性
3	18	0.036	0.023	0.013	元件品質提昇	程式屬性
4	7	0.036	0.034	0.002	軟體業者利潤的增加	組織績效屬性
5	8	0.035	0.033	0.002	易於建構產業共通元件的標準	產業屬性
6	19	0.035	0.023	0.012	系統正常運作的時間增加	系統屬性
7	3	0.035	0.038	-0.003	系統開發時可重複利用之元件庫規模的增大	系統屬性
8	34	0.034	0.015	0.019	系統跨平台成本的降低	系統屬性
9	35	0.033	0.015	0.018	可延續舊有系統的使用	系統屬性
10	26	0.032	0.019	0.013	軟體業者營業額的增加	組織績效屬性

(資料來源：本研究整理)

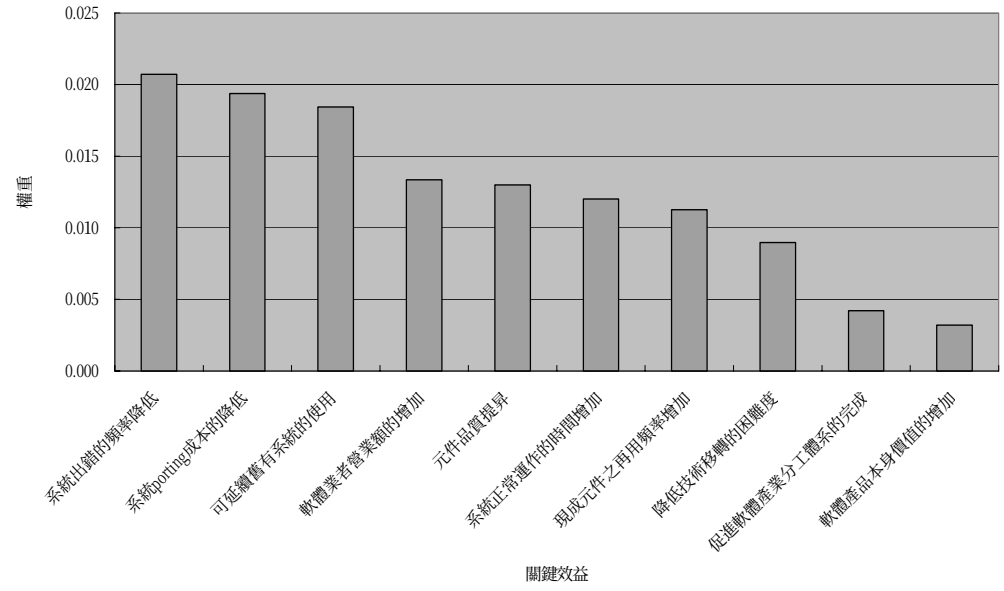

圖 3　權重增加前十大關鍵效益

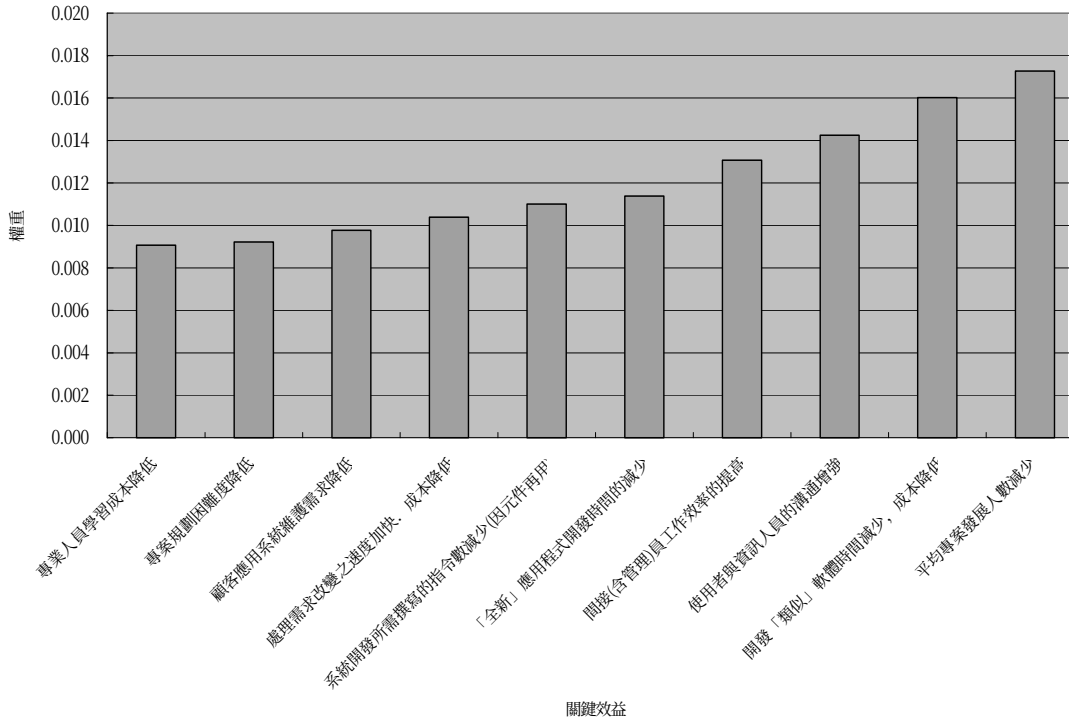

圖 4 權重減少前十大關鍵效益

退出前十大的五個關鍵效益其中有兩個重要性減少較多，分別是使用者與資訊人員的溝通增強、間接人員（含管理）作業效率的提高。同時這兩個關鍵效益也是屬於權重變化方面重要性減少的前十大關鍵效益(圖 4)，其他重要性減少的前十大關鍵效益包含：平均專案發展人數減少、開發「類似」軟體時間減少、「全新」應用程式開發時間的減少、專案規劃困難度降低、專業人員學習成本降低、系統開發所需撰寫的指令數減少、顧客應用系統維護需求降低、處理需求改變之速度加快成本降低(重要性降低但仍然排名十一)。

三、效益演變討論

從 1997 年到 2000 年正是推動軟五計畫第二期的階段，隨著領域工程和元件軟體技術的引入、相關軟體再用整合開發工具和管理平台的轉移、相關組織的輔導訓練皆有助於軟體再用供給面的強化。另一方面，在需求面上以 web 為基礎資訊系統(WIS)和跨組織資訊系統如 ERP、SCM 需求隨著電子商務快速成長而大為增加。由推動計畫初期和末期兩個年度再用效益的比較可得知：延續舊有系統的使用、降低系統跨平台成本、增加元件品質和元件再用頻率、促進分工體系和建立共通元件標準、降低技術移轉的困難、增加軟體價值和系統正常運作的時間等關鍵效益的重要性逐漸增加。另一方面，個別專案中發展人數減少、開發「類似」軟體和「全新」應用程式時間減少、規劃困難度降低、人員學習成本降低、使用者與資訊人員的溝通增強和間接人員（含管理）作業效率的提高等效益重要性則降低。

由權重增加和減少的關鍵效益（圖 3 和 4），顯示隨著元件技術引入和元件標準的遵循，再用效益著重在系統的穩定性（出錯的頻率降低、正常運作時間增加）、系統的相容性（跨平台成本降低、延續舊有系統的使用）。由於採用元件式開發方式，需要在元件庫或元件市集中尋找元件以進行切割、改變和組裝的工作。元件標準的遵循，有助於降低技術移轉的困難，在產業形成上中下游分工體系，各自提供擅長專精的元件，將使得元件品質提昇和供應量的增加，進一步促成現成元件再用頻率增加。亦即軟體再用關鍵效益更加重視經由產業分工合作來加強系統的穩定性、相容性和提升元件質與量。

另一方面，以專案導向式的效益，如專案人數（平均專案發展人數減少）、專案時間（開發類似軟體時間減少、全新應用程式開發時間減少）、專案困難度降低和專案人員學習成本降低的重要性下降。由於元件界面標準、元件架構標準和領域元件標準的遵循，有助於需求定義的明確化和雛形系統的發展，促進溝通成效（使用者和資訊人員的溝通增強）。同時領域知識可以累積學習，組裝速度加快且系統較能滿足顧客需求（處理需求改變速度加快、顧客應用系統維護需求降低）。亦即軟體再用關鍵效益從重視個別專案效益，逐漸到元件標準的遵循和經由產業分工合作來提升元件質和量等再用的基礎建設，以加強系統的穩定性和相容性，個別專案的效益雖然未刻意強調，但是由於再用基礎建設的強化，因而較容易達成。

軟體元件逐漸成為業界標準時，軟體再用可因正向網路外部性進而產生報酬遞增現象，亦即軟體再用採用者的數量與其價值形成一自我增強的正回饋效應，使市場鎖定該技術成為主流。此時透過市場競爭可使廠商的生產成本降至最低，同時交易過程中產生的交易成本也甚低。同時由產業合作網路和軟體元件市場來提升元件質與量等再用基礎建設，其學習效率和規模效率明顯，組裝成的系統穩定性和相容性提高，個別專案開發人力和時間自然就會減少。而且處理需求改變速度自然加快、應用系統維護需求自然就會降低。

伍、結論與建議

一、結論

本研究經由比較 1997 年和 2000 年軟體再用關鍵效益，歸納軟體再用效益在的變化程度和方向。雖然在兩個時期效益構面次序稍有不同，但是系統屬性、組織績效屬性和應用屬性仍是前三大最重要構面。在構面權重變化以系統屬性增加最多(0.068)，其次為產業屬性(0.021)、程式屬性(0.02)。另一方面，專案屬性的權重減少最多(-0.054)，其次為人員屬性(-0.035)。

關鍵效益方面，2000 年前十大個最重要的關鍵效益中有五個和 1997 年前十大關鍵效益一樣，只是次序稍有不同。由權重變化增加的前十大關鍵效益和權重變化減少的前十大關鍵效益，顯示業者更重視可延續舊有系統的使用、降低系統跨平台成本、增加元件品質和元件再用頻率、促進分工體系和建立共通元件標準、降低技術移轉的困難、增加軟體價值和系統正常運作的時間等關鍵效益。另一方面，在元件庫和領域知識的再用機制下，對使用者與資訊人員的溝通增強、個別專案中發展人數減少、規劃困難度降低、人員學習成本降低、開發「類似」軟體和「全新」應用程式時間減少等關鍵效益重要性則有減少趨勢。

由於標準化元件架構和元件界面的遵循，使得軟體再用正向網路外部性明顯，進而產生報酬遞增現象。其效益逐漸從重視個別專案效益，到組織外的跨專案效益乃至於整個產業的效益。也就是經由產業合作網路和軟體元件市場來提升元件質與量等再用基礎建設，使得組裝成的系統穩定性和相容性提高，個別專案開發和維護等效益自然較容易達成。此再用效益演變的趨勢和相關學者提出的看法一致，認為在電子網路相連下，電子階層會逐漸演變成跨組織合作和電子市集(Williamson, 1992；Malone et al. , 1987 & 1989)。對於目前元件統治結構仍然以電子階層為主的國內軟體業者相當有啟示作用。

總之 , 本文用較為全面性效益評估模式，縱貫面比較軟體再用效益的演變，這方面的議題為專家學者所重視，但研究較為稀少。在實務上，提供對軟體再用效益較為全面性的衡量模式和效益變化程度與方向，使軟體業者和相關單位在推動和管理軟體再用相關政策時，能充分掌握軟體再用效益，將有限資源集中在關鍵效益上。

二、建議

建議業者改變傳統上只重視個別專案可以節省多少時間和人力的專案導向做法，從再用制度、流程和管理方式著手，培養再用習慣和文化，遵循軟體元件架構與界面標準，致力提昇元件品質，累積擴大再用元件庫和提高再用率等跨專案、跨組織的軟體再用基

礎建設做起。同時，相關單位掌握網路外部性的正回饋特性，促進軟體產業分工體系形成，並加速推動國內元件交易市場的建立，以產業整體資源來面對規模日益複雜和擴大的軟體市場。

參考文獻

1. 朱文禎、鄭景華、楊建民，”軟體再用效益評估模式之建構”，第三屆永續發展管理研討會，屏東，頁 95-107(2001)。
2. 吳思華，策略九說，頁 173-175，台北，臉譜文化(1998)。
3. 施穎偉，電子商務時代供應練互動模式之研究，國立政治大學資管研究所博士論文，民國 89 年。
4. 翁俊興，分析層級程序法應用在投資計畫評估之研究，國立政治大學企管研究所碩士論文，民國 72 年。
5. 張紹文，分析層級程序法應用在行銷資源分配之應用，國立政治大學企管研究所碩士論文，民國 72 年。
6. 楊仁達，“ 企業 e 化之系統基石-軟體元件”，資策會(2000)。
7. 楊建民，軟體再用技術效益評估模式之研究，資訊工業策進會委託學術機構研究計畫報告(1997)。
8. Bakos, J. Y., ”A Strategic Analysis of Electronic Marketplaces,” *MIS Quarterly,* 1991, Vol. 15, No. 3, pp. 295-310.
9. Barnes, B. and T. Bollinger., ” Making Software Reuse Cost Effective,” *IEEE Software***,** 1991, Vol. 8, No. 1, pp. 13-24.
10. Boehm, B. W., *Software Engineering Economics***,** Prentice Hall, N.J. (1981).
11. Choudhury, V., K. S. Hartzel, & B. R. Konsynski, ”Uses and Consequences of Electronic Markets: An Empirical Investigation in the Aircraft Parts Industry,” *MIS Quarterly,* 1998, Vol. 22, No. 4, pp. 471-508.
12. Eisenhardt, K. M., “Agency Theory: An Assessment and Review,” *Academy of Management Review,* 1989, Vol. 14, No. 1, pp. 57-74.
13. Frakes, W. and C. Terry, “Software Reuse: Metrics and Models,” *ACM Computing Surveys***,** 1996, Vol. 28, No. 2, pp. 415-435.
14. Gaffney, J. E. and T. A. Durek, ” Software Reuse-key to Enhanced Productivity: Some Quantitative Models,” *Information and Software Technology*, 1989, Vol. 31, No. 5, pp. 258-267.
15. Gall, H., M. Jazayeri, & R. Klosch, ” Research Directions in Software Reuse: Where to go from here?” Proceedings of the 17th International Conference on Software Engineering, pp. 225-228(1995).
16. Jacobson, I., M. Griss, and P. Jonsson, *Software Reuse: Architecture, Process and Organization for Business Success*, Addison-Wesley, N.Y. (1997).
17. Khairuddin, H. and, E. Key, "A Software Reusability Attributes Model," *International*

Journal of Computer Applied Technology, 1995, Vol. 8, No. 1-2, pp.69-77.

18. Lamela, C., Breaking down the barriers to software component technology, Doc 03-012-04 001027, IntellectMarket, Inc. (2000).
19. Laudon, K. C. & J. P. Laudon, *Management Information Systems: Organization and Technology in the Networked Enterprise* (6th ed.), Prentice-Hall, N.J. (2000).
20. Lozinsky, S., *Enterprise-wide Software solutions*, Addison-Wesley, N.Y. (1998).
21. Malone, T. W., J. Yates, & R. I. Benjamin, "Electronic Markets and Electronic Hierarchies, " *Communications of the ACM*, 1987, Vol. 30, No. 6, pp. 484-497.
22. Malone, T. W., J. Yates, & R .I. Benjamin, "The Logic of Electronic Markets," *Harvard Business Review*, 1989, Vol. 67, No. 3, pp.166-172.
23. McFarlan, F., "Information Technology Changes the Way You Compete," *Harvard Business Review*, 1984, Vol. 62, No. 3, pp.77-88.
24. Mizuno, Y., "Software Quality Improvement," *IEEE Computer*, 1983, Mar. pp.66-72.
25. Paul, R. J.,"Why Users Cannot 'Get What They Want'," *International Journal of Manufacturing Systems Design*, 1994, Vol. 1, No. 4, pp. 389-394.
26. Poulin, J. S., J. M. Caruso, & D. R. Hancock, "The Business Case for Software Reuse," *IBM System Journal*, 1993, Vol. 4, No. 32, pp.567-594.
27. Poulin, J. S., *Measuring Software Reuse- Principles, Practices and Economic Models*, Addison-Wesley, N.Y. (1997).
28. Rindfleisch, A. and J. B. Heide, "Transaction Cost Analysis: Past, Present, and Future Applications," *Journal of Marketing*, 1997 , Vol. 61, No. 3, pp.30-54.
29. Satty, T. L., *The Analytic Hierarchy Process*, MacDraw Hill, N.Y. (1980).
30. Sprott, D., "Open market components, A CBDi Forum Report," http://www.compoentsource.com/BuildCo…/cbdiopen_market.asp (2000)
31. Stinchcombe, A., *Information and Organizations*, University of California Press, C.A. (1990).
32. Williamson, O. E., "Transaction-Cost Economics: The Governance of Contractual Relations," *Journal of Law and Economics*, 1979, Vol. 22, pp.233-261.
33. Williamson, O. E., "Market, Hierarchies, and the Modern Corporation: an Unfolding Perspective," *Journal of Economic Behavior and Organization*, 1992, Vol. 17, No. 3, pp.335-352.

附錄一　　系統屬性

1. 就您個人的看法而言，下列影響**系統屬性**的關鍵因素，其相對重要性如何？

說明：

影響**系統屬性**的關鍵因素如下：

概念與實作一致------　程式模組可直接由系統概念模式導出

元件庫規模增加------　系統開發時可重複利用之元件庫規模的增大

系統文件減少---------　系統文件頁數的減少

跨平台成本減少------　系統跨平台成本的降低

延續舊有系統---------　經由再用技術之包裹可延續舊有系統的使用

錯誤率降低----------　系統出錯的頻率降低

正常運作時間增加----- 系統正常運作的時間增加

		絕對重要 9	極為重要 7	頗為重要 5	稍為重要 3	同等重要 1	稍為重要 3	頗為重要 5	極為重要 7	絕對重要 9	
1	概念與實作一致	□	□	□	□	☐	□	□	□	□	元件庫規模增加
2	概念與實作一致	□	□	□	□	☐	□	□	□	□	系統文件減少
3	概念與實作一致	□	□	□	□	☐	□	□	□	□	跨平台成本減少
4	概念與實作一致	□	□	□	□	☐	□	□	□	□	延續舊有系統
5	概念與實作一致	□	□	□	□	☐	□	□	□	□	錯誤率降低
6	概念與實作一致	□	□	□	□	☐	□	□	□	□	正常運作時間增加
7	元件庫規模增加	□	□	□	□	☐	□	□	□	□	系統文件減少
8	元件庫規模增加	□	□	□	□	☐	□	□	□	□	跨平台成本減少
9	元件庫規模增加	□	□	□	□	☐	□	□	□	□	延續舊有系統
10	元件庫規模增加	□	□	□	□	☐	□	□	□	□	錯誤率降低
11	元件庫規模增加	□	□	□	□	☐	□	□	□	□	正常運作時間增加
12	系統文件減少	□	□	□	□	☐	□	□	□	□	跨平台成本減少
13	系統文件減少	□	□	□	□	☐	□	□	□	□	延續舊有系統
14	系統文件減少	□	□	□	□	☐	□	□	□	□	錯誤率降低
15	系統文件減少	□	□	□	□	☐	□	□	□	□	正常運作時間增加
16	跨平台成本減少	□	□	□	□	☐	□	□	□	□	延續舊有系統
17	跨平台成本減少	□	□	□	□	☐	□	□	□	□	錯誤率降低
18	跨平台成本減少	□	□	□	□	☐	□	□	□	□	正常運作時間增加
19	延續舊有系統	□	□	□	□	☐	□	□	□	□	錯誤率降低
20	延續舊有系統	□	□	□	□	☐	□	□	□	□	正常運作時間增加
21	錯誤率降低	□	□	□	□	☐	□	□	□	□	正常運作時間增加

(由於篇幅所限，僅附部份問卷，完整問卷歡迎索取)

附錄二

軟體再用效益絕對權重總列表

順序		權重		權重變化	關鍵效益	構面
1997	2000	1997	2000			
1	1	0.059	0.055	-0.004	所開發之系統較能滿足使用者資訊需求	應用屬性
2	11	0.041	0.031	-0.010	處理需求改變之速度加快，成本降低	應用屬性
3	7	0.038	0.035	-0.003	系統開發時可重複利用之元件庫規模的增大	系統屬性
4	13	0.036	0.029	-0.007	程式模組可直接由系統概念模式導出	系統屬性
5	2	0.034	0.055	0.021	系統出錯的頻率降低	系統屬性
6	24	0.034	0.020	-0.014	使用者與資訊人員的溝通增強	應用屬性
7	4	0.034	0.036	0.002	軟體業者利潤的增加	組織績效屬性
8	5	0.033	0.035	0.002	易於建構產業共通元件的標準	產業屬性
9	15	0.033	0.027	-0.006	人員流動所造成的衝擊減少	組織績效屬性
10	23	0.033	0.020	-0.013	間接人員（含管理）作業效率的提高	人員屬性
11	16	0.029	0.026	-0.003	系統分析人員年產值的增加	人員屬性
12	21	0.029	0.021	-0.008	資訊系統較能因應組織變動與調整	應用屬性
13	17	0.028	0.025	-0.003	成果產出的精準度增加	專案屬性
14	12	0.027	0.030	0.003	軟體產品本身價值的增加	組織績效屬性
15	38	0.026	0.010	-0.016	開發「類似」軟體時間減少，成本降低	專案屬性
16	18	0.025	0.024	-0.001	程式設計人員年產值的增加	人員屬性
17	32	0.024	0.014	-0.010	顧客應用系統維護需求降低	組織績效屬性
18	3	0.023	0.036	0.013	元件品質提昇	程式屬性
19	6	0.023	0.035	0.012	系統正常運作的時間增加	系統屬性
20	36	0.023	0.012	-0.011	「全新」應用程式開發時間的減少	專案屬性
21	41	0.023	0.006	-0.017	平均專案發展人數減少	專案屬性
22	35	0.022	0.013	-0.009	專案規劃困難度降低	專案屬性
23	19	0.02	0.022	0.002	上線調整測試時間的減少	程式屬性
24	20	0.02	0.021	0.001	易於建立、推展特定產業應用模式之標準	產業屬性
25	27	0.02	0.018	-0.002	提昇組織導入再用技術的層次	產業屬性
26	10	0.019	0.032	0.013	軟體業者營業額的增加	組織績效屬性
27	29	0.019	0.018	-0.001	程式複雜度降低，易於撰寫	程式屬性
28	37	0.019	0.011	-0.008	資訊人員對本身專業工作成就感增加	人員屬性
29	25	0.018	0.019	0.001	需求改變次數減少	應用屬性
30	14	0.016	0.027	0.011	現成元件之再用頻率增加	程式屬性
31	28	0.016	0.018	0.002	系統中再用現成元件之比例增加	程式屬性
32	40	0.016	0.007	-0.009	專業人員學習成本降低	人員屬性
33	42	0.016	0.005	-0.011	系統開發所需撰寫的指令數減少(因元件再用)	程式屬性
34	8	0.015	0.034	0.019	系統跨平台成本的降低	系統屬性
35	9	0.015	0.033	0.018	可延續舊有系統的使用	系統屬性
36	30	0.015	0.016	0.001	有利於軟體人員之養成與素質之提昇	產業屬性
37	33	0.015	0.014	-0.001	顧客索賠(claim)次數減少	組織績效屬性
38	26	0.014	0.018	0.004	促進軟體產業分工體系的完成	產業屬性
39	31	0.014	0.015	0.001	時程變動的幅度縮小	專案屬性
40	34	0.012	0.013	0.001	系統文件頁數的減少	系統屬性
41	39	0.012	0.008	-0.004	資訊人員對組織內其他活動的參與意願提高	人員屬性
42	22	0.011	0.020	0.009	降低技術移轉的困難度	產業屬性

(資料來源：本研究整理)