程式語言背景知識之語法與用途對學習新程式語言之影響

蕭文峰
國立屏東商業技術學院資訊管理系

張德民
國立中山大學資訊管理系

鍾靖良
臺北市立萬芳醫院

摘要

由於易於維護與擴充的特性，近幾年物件導向已成為程式語言的主流。因此，對已有程序導向背景的學習者如何將其舊有技能遷移至物件導向程式語言的學習上是個重要的議題。過去對於影響程式語言間技能遷移的可能因素仍不是很清楚。本研究將程式語言之背景知識區分成語法及用途兩個構面，並藉由實驗問卷探討「語法的異同」與「用途的異同」會如何地影響學習者的學習。實驗結果顯示，「語法不同」或「用途不同」，皆會負向地影響受測者的表現，可知若背景程式知識與新程式間有語法或用法上的差異將會產生負遷移。另外，經由Logistic Regression分析結果顯示「題目的難易度」為一調節變項（moderator），會影響「語法的異同」或「用途的異同」對受測者答題的表現。

關鍵字：程式語言、學習遷移、正遷移、負遷移

The impact of syntax and usage in the learner's background languages on the learning of a new programming language

Wen-Feng Hsiao
Department of Information Management, National Pingtung Institute of Commerce

Te-Min Chang
Department of Information Management, National Sun Yat-sen University

Jing-Liang Chung
Municipal Wan Fang Hospital

Abstract

Owing to its easiness in maintenance and extension, object oriented programming languages has now been the main stream. Therefore, for those learners who have backgrounds in procedure-oriented programming languages, how to make use of their old skills in learning object-oriented programming language is an important issue. We still know little about the factors that influence the skill transfers between two programming languages. This study analyzes learner's background knowledge into two aspects, namely, syntax and usage, and designs the corresponding experimental questionnaire to investigate how syntax and usage of programming languages influence the learner's learning. The result shows that if the background knowledge of a learner has syntax or usage deviance with new programming language, then it will have negative skill transfer. Also, by Logistic Regression analysis, this study finds out that "difficulty of the task" is a moderator that will adjust the relation between syntax and usage of different programming languages and the subject's performance.

Key words: programming language, skill transfer, positive transfer, negative transfer

壹、緒論

近年來學習程式設計的人數逐漸地成長，而每一種程式設計（包括有網頁程式設計、視窗程式設計、Console程式設計、與網頁script程式設計等）又各自有不同的程式語言。大抵而言，資訊相關科系的學生在求學階段，不可避免地，會學習至少兩種以上的程式語言。學習者在學習許多程式語言的過程中，經常會不知不覺將其他語法混合使用，造成學習上的挫折。雖然許多整合開o發環境（Integrated Development Environment, IDE）可以協助程式開發者避免大部分由語法干擾所造成的錯誤。但由於IDE的發展嘗試兼容並蓄地包含不同語言（例如，在Visual Studio .NET中同一專案可以結合C#、VB.NET、及JavaScript等語言），許多語法混用的錯誤仍無法被找出。因此，了解在那一種情境下會受到過去所學習知識的干擾將有助於IDE的開發者及程式語言的授課教師輔助學習者快速跨越其間的障礙，達到有效學習的目的。

有學者主張學習程式語言的經驗就類似學習母語一般，第一個被深入學習的程式語言會深植在學習者心中，往後學習其它程式語言都會以第一程式語言做類比。許多證據亦顯示，造成學習者挫敗的主要原因並非程式語言的複雜度（因為任何程式語言到了進階之後都很複雜），而是學習者的動機及教授者是否能循序漸進地提供學生充份的練習。否則，簡單如視覺化程式語言（例如，Visual Basic）的學習者亦難有好的表現（參考Boulay et al. 1999建立初學者之概念機器）。

技能遷移理論認為，有些舊技能能幫助新技能的學習（正遷移），但有些則會對新技能的學習造成干擾（負遷移）。辨別假說（theory of discriminate）亦指出，受試者為了區辨新學習的事物中那些是舊的事物（或區分新舊事物間的差異），會花費更多時間學習（Schwartz & Humphreys 1973）。因此，有舊技能基礎者，一開始雖然表現較沒有基礎者好，但其進步較緩慢，甚至會在後面階段表現較無任何舊技能基礎者差（Fix & Wiedenbeck 1996）。例如，Bellin（1992）發現沒有結構化技術經驗的學生在物件導向概念上表現較佳。Miller（2000）指出，欲精通Java語言，C++的專家跟未學過C++的生手一樣至少需花兩年的時間學習 。無論如何，在學術界，對於學習新程式語言的困難度與學習者的背景知識間的關係尚不是很清楚。

同樣地，在業界，程式設計師的背景知識對學習新程式語言的難易度影響也不清楚。程式設計師的背景知識與學習新程式設計語言的影響對軟體專案管理者有重要的意義。許多專案經理被迫提供新程式語言的訓練給他們的程式設計師。而不瞭解背景知識對學習新程式語言的影響，將很難去預測移轉至新程式語言所需的時間與成本。

從程式語言發展的角度來看，不同語言間存在著許多語法、用途上的歧義。整體而言，相同範型（paradigm）的語言，其語法雖不相同，但在用途上可能相同。例如，C和Basic的For迴圈「用途」都是連續處理相同程序n次，但其語法卻不相同（參考表1）。不同的範型則其語法及用途可能都不相同。例如，在Visual Basic中，「=」同時被當成指定運算子及測試相等的運算子 ，但學習者轉換至Java時便容易在「判斷相等」的運算出錯。

表1：「用途」相同，但「語法」不相同

C++/Java語法	Basic語法

for （int i=0; i<n; i++）
{
　重複主體;
}
	Dim i as Integer
	For i=0 to n-1
	　重複主體
	Next

因此，本研究之主要目的即在於從學習者背景程式語言與新程式語言對應之「語法」及「用途」之異同來探討學習者技能遷移的現象。

貳、理論基礎與研究模式推導

本研究擬以技能遷移理論來分析學習者的背景知識對學習程式語言績效之影響，構成本研究的主要理論基礎。分別說明如下，並同時進行研究模式之推導，提出研究假說。

一、技能遷移理論

如果兩種技能在刺激與反應間有相同或相似的元素，且對人的能力和心理特點有共同的要求時，就可能發生遷移（transfer）。然而，由於是兩種技能，如無法適切地分辨出其中的異同，則容易發生干擾（負遷移）。目前用於解釋技能遷移的主要理論是以Anderson的ACT*技能獲取理論（Singley & Andersion 1989）為基礎的「共有元素（common element）」論 。此理論認為，技能行為是由一領域的宣告性知識（事實）轉換成程序而得。這些程序只在高特定的情況下才會被觸發。如此，在一領域所學得的某一部份程序將無法被使用在此領域的其它部份，除非使用此程序的條件在兩個情況下是相同的。因此要在兩個工作間遷移技能，兩工作必須共有相同的程序或元素。據此可推得此理論的基本原則「知識的使用特定性（use specificity of knowledge）」（Anderson 1987）：由訓練及練習所獲得的不是知識，而是知識的某個特定的使用法（Anderson 1993; Singley & Anderson 1989）。此理論預測在一複雜領域中，當知識以不同的方式被使用時，子技能間的遷移將會很少甚或沒有。例如，由一子技能「寫一電腦程式」所獲取的知識將不會遷移到另一子技能「瞭解某一電腦程式」的表現，即使此兩個子技能有共同的宣告性知識庫（i.e., 程式語言指令的定義）（Pennington et al. 1995）。

當舊技能的知識無法遷移至新技能的學習時，便可能產生干擾。過去的研究發現，當程式語言學習者學習新程式語言時，容易受舊程式語言之語法干擾，影響學習的效率。例如，Stroustrup（1999）發現學過其它程式語言但初學C++的學習者常會將過去學過語言之「語法」套用至C++中，因而導致編譯錯誤。同理，在其它程式語言的學習上亦可能有相同的問題，學習者常會混淆各程式語言之語法及用途。

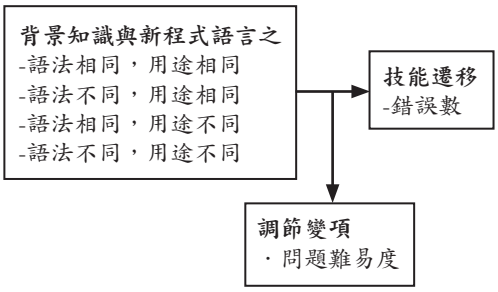


圖1：研究模式

綜合過去一個世紀的研究成果，從Thorndike（1906）以來，技能的遷移與干擾現象是一個十分普遍且穩定的現象。對此，本研究尤其有興趣於瞭解「語法」與「用法」何者之負遷移較為明顯，其研究模式如圖1所示。針對圖1之模式可推導出五個研究假說，依次說明如下：

二、正遷移

依據「共有元素」論，技能的遷移只有在十分特定的情況下才會被觸發。亦即，只有當使用兩技能的條件相同時，才可能發生技能遷移。因此，學習者的背景程式語言只有在該語言的語法及用途與新學習程式語言相同的部份才可能發生遷移的現象。例如，C/C++迴圈控制、流程條件判斷、及指定運算（assignment）等與Java的語法及用途相同者對學習Java具正遷移的影響。此可推得本研究的第一個研究假說。

H1：學習者背景知識中與新程式語言的語法相同且用途相同者，對學習新程式語言會有正遷移的現象。

三、負遷移

相反地，當知識以不同的方式被使用時，子技能間的遷移將會很少甚或沒有。心理學中探討造成學習干擾的理論主要有兩個，一為倒攝抑制（retroactive inhibition）「新學習知識干擾舊知識的記憶」，另一為順攝抑制（proactive inhibition）「舊知識的記憶干擾新學習」。然而至1960年代以後，順攝抑制成為解釋記憶干擾的主要原因。因為從長期記憶的觀點而言，儲存在長期記憶中的舊資料遠比接收新進資料多；因此，舊資料干擾新資料的機會較多也較合理。過去的研究觀察Java學習者所發生的干擾現象包括（非窮舉）：（1）在Java中沒有記錄（record）的概念，是以類別（class）代替。但對有C/C++背景的學習者並不是個好現象，因為記錄跟類別是兩種不同的概念，而且無法使學習者瞭解記錄的概念就如同靜態容器（static container）的觀念（Boszormenyi, 1998）。（2）Java的語法源自於C，允許混合敘述及單一敘述。例如if（x>1）x++; 但Java新增的控制結構則不行，如try…catch等，即使只有單一敘述也必須包覆在{ }中。例如try { a = b / c; }是正確的，但try a = b / c; 則為錯誤。這樣的改變容易讓學習者產生混淆（Thimbleby

1999）。（3）在Java中，一般變數型態採傳值呼叫（called by value）；物件則採傳參考（called by reference），這樣混用不僅容易造成混淆也使得學習者更模糊（Boszormenyi 1998）。換言之，在設計的過程中，程式本身的範型（paradigm）若有很多且很明顯的不同，將會明顯地影響到學習者的程式理解力。而程式範型將是影響初學者學習效果的重要關鍵之一。

據此，本研究推論，當學習者的舊程式語言知識無法直接套用至學習新程式語言上時，便會對學習新程式語言造成干擾。依據語法及用途上的差異，可推得H2-H5等四個假說。首先，當兩程式語言都提供相同功能（例如開啟檔案或計算陣列元素個數），但使用不同的語法，此將迫使使用者增加額外的記憶負擔，造成負遷移。例如，Visual Basic中的「select case」可對應到Java中的「switch case」，其用途是相同的，但其語法的差異常造成使用者學習上的障礙；在Java中的每一個case判斷與處理間必須以「:」區隔，且每個case間是以break來區隔，而Visual Basic則不需要。這些都將造成學習者不必要的混淆。故推得假說二。

H2：學習者背景知識中與新程式語言的語法不相同且用途相同者，對學習新程式語言會有負遷移的現象。

另外，當兩程式語言採用同一套語法來代表不同功能時，會造成學習者在理解程式上的困擾，造成負遷移。例如，Visual Basic中以「&」來代表字串與整數的串接運算，而Java中「&」代表「位元的AND運算」（bitwise AND）；Visual Basic中以「^」代表數值的指數運算，而Java中「^」代表「位元的XOR運算」（bitwise XOR）；C++中以「>>」及「<<」代表輸入及輸出串流，其左右位移的運算也是相同，只是「>>」及「<<」被overloaded成輸入及輸出運算子，而Java中「>>」及「<<」代表位元的右移及左移運算。另外在Visual Basic中測試相等是以「=」來表示；而在C++/Java中「=」代表指定運算子。本研究認為這些會造成使用者在理解上的混淆，故推得假說三。

H3：學習者背景知識中與新程式語言的語法相同但用途不相同者，對學習新程式語言會有負遷移的現象。

最後一種情形很自然是新舊程式語言彼此之語法不同且用途亦不同。符合此條件的語言構面將會非常多，即使在同一語言中也可很容易找到兩個語法不同且用途不同的構面。但與正負遷移有關的兩種情形是：（1）背景知識有的，但在新語言中沒有可對應的；（2）新語言中有的，但在背景知識中是沒有的。此兩種情形推論如下：

當學習者的背景程式語言中有適合解決特定問題的語法，但在新程式語言中無適當對應者，學習者學習新程式語言時會發生負遷移。例如C++允許重新定義運算子（operator overloading），以進行物件間的運算，在Java中並不允許，如此學習者必須將之前的思考方式重新調整，以適應新程式語言的限制，故造成負遷移。其它如C++提供自動型別轉換（implicit conversion）、多重繼承（multiple inheritance）、樣版（templates）、前處理（preprocessor）、全域變數（global variables）、記憶體管理（memory management）、指標（pointers），這些在Java中並不提供，因此在學習上會造成額外的負擔，故可推得假說四。

H4：學習者背景知識中在新程式語法中無適當對應者，對學習新程式語言語言會有負遷移的現象。

相反地，若新程式語言中的語法概念是學習者背景知識所沒有的，則不會有技能遷移發生。此類情形類似學習一全新技能，學習者背景知識既無法對新技能學習產生幫助，但也不致於發生干擾。例如，圖形界面程式設計（AWT/Swing）、Threads、Networking、文件產生器（Generators for documentation）、Reflection、界面（interface）、applet等是Java所獨有，而C++沒有的。是故，有C++背景者必須以全新的觀念來吸收這些概念。當然，這亦包括完全沒有任何程式背景的學習者，將某一個程式語言當成第一門程式設計語言所面對的情境。據此，可推得假說五。

H5：新程式語言的語法在學習者背景知識中無適當對應者，對學習新程式語言將不會有遷移的現象。

四、調節變項（moderating variables）

問題的困難度將對研究模式中的自變項與因變項的關係產生影響。問題愈困難，學習者愈需使用其深層基模來解決問題。學習者腦中的記憶與技能會很自然地浮現用以解決問題。當問題較簡單時，學習者會有較多的餘裕去思考新語言的語法及用途上的歧義。同樣地，當問題較困難時，學習者的認知風格對解題方向將會有顯著的影響。因此，「問題的難易」在本研究中視為調節變項。

參、研究方法

要觀察技能遷移的現象理想上應從學習者學習第一個程式語言開始測量與記錄，如此才能瞭解那些技能發生了正遷移，那些發生了負遷移。然而此類「縱貫性的研究（longitudinal study）」有其執行面的困難。首先，我們無法知道學習者何時學習第一個程式語言。再者，若從大一開始記錄，則由於某些大專院校，C++與Java程式語言是二選一的選修，因此人數不會很多，而且C++與Java間之技能遷移便無從瞭解。故欲達到合理的樣本數勢必在許多學校同時施行此研究，這在人力物力與時間上是不可行的。因此，本研究退而求其次，採用「橫斷面（cross-sectional）」的研究方法，以實驗方法並設計問卷來了解學習者技能遷移的情形。

我們以目前佔有率最高之程式語言（VB、C++、與Java）進行假說探討；因為佔有率最高，因此學習人口會相對較多，也因而同時學過此三個程式語言中任兩個（含）以上的機率會較高（注意，技能遷移的發生是在新舊程式之間。故學過兩種或以上的程式語言是必要的）。為了瞭解研究變項間可能的因果關係，於問卷的設計上我們同時考量語法與用途在三種語言上的異同、以及題目的難異度，並反覆修正我們的問卷，以期能確實測量到受測者的技能遷移的影響變因。

一、研究設計

本問卷題目主要是以檢定第二節之假說為目的來進行設計（附錄中列出各程式語言之第一題）。為了得到較高的可靠性，問卷中的每一假說至少會有2小題來進行衡量。且為了得到較普遍的結果（亦即，非專業的程式設計人員亦能參與），題目的選擇上儘量以基本語法為主，而非刻意要求受測者死背出來的語法。例如，很多學過Java的學生都知道在Java中欲讀取鍵盤的輸入相當的複雜，其語法為String s= new BufferedReader(new InputStreamReader (System.in)).readLine();。在我們的題目中不會有類似此類語法的填答。另外為了使三份問卷題目具有一致性，故設計的題數與題型盡量相同。同時，為了避免受測者的負擔，我們將所有題目以填空方式進行測驗；不以選擇題方式進行的主要原因是受測者在有限的選項中較容易選取到正確答案，且程式設計本身本來就不是選擇題。最後，我們採以填空方式而不讓受測者依題目要求自由發揮的原因是我們的研究目的在於瞭解不同程式語言間是否具有正負遷移的現象，而非在測驗他們的程式撰寫與組織能力。以填空的方式進行可以得到比較一致的答案，而且也可大幅縮減施測及分析所需的時間。

為獲得遷移的大小與方向，受測者在問卷的基本資料部份必須回答學習過的程式語言及順序；且為了使問題與答案較為完備與可行，我們進行了兩次的前測，分別由三位程式能力較好的學生進行預試（pilot test），並根據他們的答題情況及建議對問卷進行修改。

二、受測者獲取

為召募自願的受測者我們在台灣最大的BBS站（PTT）及各大專院校系所的BBS站張貼本研究實驗訊息（五月二十一日陸續公告實驗訊息）。最後共募集141位受測者，分別來自中山大學、屏東商業技術學院、屏東科技大學、高雄第一科技大學，高雄應用科技大學，成功大學，崑山科技大學、靜宜大學、台灣師大、長庚大學、世新大學等學校以及幾位業界人士。本實驗從五月十八日在屏東商業技術學院資管大三開始進行施測（實驗訊息公告之前，共31名受測者），直到七月三日在高雄應用科技大學最後一次進行實驗。

三、實驗流程

由於受測者散居各地，為使實驗能嚴謹的進行，我們會儘量到受測者的學校進行施測。實驗一開始會由實驗者說明實驗的內容與注意事項，並提醒受測者我們會依照受測者答題正確率與完成率進行排名，前五名可獲得獎金：第一名4,000元、第二名3,000元、第三名1,500元、第四名1,000元及第五名500元，以藉此鼓勵他們認真回答。接著進入實驗，本問卷分為二大部份：第一部份是程式語法填空，第二部份是基本資料。每份問卷第一部份（不管是選擇填答VB, C++, 或Java）皆為十一題，每一題有多個空格讓受測者填寫基本語法。而每位參與實驗填寫的受測者必須修過或學過VB.NET（VB）、C++（C）或Java程式語言至少兩種以上。在填寫問卷時，受測者只需選擇自認為最熟悉的語言進行填寫。為了獲得可靠的結果，我們要求受測者在填答的過程中不可查閱書本或網

路上的任何資料。為避免受測者的投機行為，實驗者會於實驗過程中監看受測者的填答情形。受測者問卷的完成時間約為四十五分鐘至五十五分鐘。

肆、結果與分析

為瞭解影響技能遷移的主要因素，我們在問卷題型中抽取11格的相關假說題型進行分析（H1、H4、與H5各為1格；H2為5格；H3為3格），其餘格數則供另一相關研究所使用，在此不列入分析。這些題項的選取方式是依照每一假說在問卷所佔的比例來選取，以期較能代表整份問卷（在C++與Java問卷中屬於H1的題型雖然不少，但在這三分問卷中，其語法與用途皆為相同的題型相對較少，所以在此我們僅抽取1格代表H1）。這些題型分別為判斷數值小於、常數宣告、產生亂數、字串宣告、case、繼承、判斷數值相等、求算某數n次方、位元AND運算、字串與整數串接與轉換建構子、樣板（template）介面（interface）及視窗元等11個題項。這11個題項在VB、C++與Java語言的差異如表2所示。

其中，H4所關心的是學習者背景知識中在新程式語法中無適當對應的語法構面，其技能遷移必須考慮到學習的次序，因此所選擇的題項是（1）如果受測者有學過VB或Java但填答時選C++問卷者（表示他/她最熟悉的語言是C++，但之前有學過VB或Java），則會以字串與整數串接那題來檢驗；（2）如果受測者學過C++，但以Java或VB填答時，則會以物件建構子那題來檢驗，看受測者是否會寫成C++的語法（詳見表3）。 而H5所探討的是新程式語言在學習者背景知識中無適當對應者的影響，故在三份問卷（程式語言）各挑出一題項是該語言所特有的語法構面（e.g. Java的Interface, C++的template, 及VB的GUI元件）。

表2：三種語言（三份問卷）在選定題項之對應語法

假說	任務	VB	C++	Java
H1	判斷數值小於	<	<	<
H2	常數宣告	Const PI As Double=3.1415926	const double PI=3.1415926	final static double PI=3.1415926
	產生亂數	Int（Rnd（）*29）	rand（）%29	（int）（Math.random（）*29）
	宣告字串型態	Dim a As String	string a	String a
	case	Case 8 To 9	case 8:case 9:	case 8:case 9:
	繼承	inherits	:	extends
H3	判斷數值相等	If a=b Then	if（a==b）	if（a==b）
	計算某數n次方	a^10	pow（a,10）	Math.pow（a,10）
	位元AND運算	i=i AND x	i=i & x;	i=i & x;
H4	字串與整數串接	s3=s1 & age	s3=s1 + itoa（age,c,10）;	s3=s1 + age;
	轉換建構子	rational a=new rational（3,1）	rational a=3;	rational a=new rational（3,1）;
H5	樣版		template<class T>	
	介面			interface temperatureConversion
	視窗元件	b（i）.Text=i+1		

表3：H4分析題型

	任務	語法
（VB/Java有） 但C++沒有	字串與整數串接	C++語法為s3=s1 + itoa（age,c,10） （VB與Java都可以運算子串接）
C++有 但Java沒有	轉換建構子	Java語法為rational a=new rational（3,1）; （C++可寫成rational a=3;）
C++有 但VB沒有	轉換建構子	VB語法為rational a=new rational（3,1） （C++可寫成rational a=3;）

一、正負遷移判斷準則

所謂的「正遷移」指的是新技能的學習會因與舊技能的知識相同而受益；所謂的「負遷移」指的是舊技能知識干擾新技能的學習。因此，正遷移的分析只針對受測者答對的題目或空格進行分析，但並不是所有答對的空格都是正遷移；而負遷移的分析只針對受測者答錯的題目或空格進行分析，但並不是所有答錯的空格都是負遷移。如何判斷某答對的空格是否是由正遷移造成的；某答錯的空格是否是由負遷移造成（而不是亂答）並不容易。所以在分析之前我們必須瞭解受測者過去所學習的程式語言（由基本資料可獲得），才能判斷是否有正負遷移的現象。因此會以兩種語言進行評閱分析（一為填寫問卷語言，另一為過去所學的語言），我們的準則如下。

若兩種程式語言的語法與用途皆相同時，可視為正遷移，其餘的情形皆不屬於正遷移。例如，在Java問卷所填寫的for迴圈其語法為for（int i=1;i<=10;i++），這與C++的語法for（int i=1;i<=10;i++）完全相同且用途亦同，因此可視為正遷移。

當受測者將答案填寫為另一種程式語言的語法時，即是一種負遷移。例如，在C++或Java問卷中的分支判斷語法寫成VB的case 8 to 9，也是一種負遷移（屬H2，語法不相同但用途相同；因為兩者語法很類似，但卻不相容）。或者在C++或Java問卷填寫求算某變數的10次方時，若填寫為a^10表示受到VB的干擾，因此視為負遷移（屬H3語法相同用途不同；因為^運算子在C++或Java代表的是邏輯XOR運算）。最後，若C++問卷中的「字串與數值串接」受測者將之寫成s1+age亦是一種負遷移（屬H4，背景知識在新程式語法中無適當對應者；因為C++中沒有字串與數值串接的直接運算子）。

當受測者填答的問題為該程式語言所特有的語法時（屬H5，新程式語言的語法在背景知識中無適當對應者），則會觀察所填答的語法是否有寫成其它語言的語法以及是否正確，以判斷是否受遷移影響。例如，Java語言中的interface是C++或VB所沒有的，我們會觀察受測者填答時的語法是否正確，或是寫成其它語言的語法。

表4為回答各份問卷（VB、C++、及Java）其所學過的對應背景程式語言所佔人數，因此C++/Java對VB技能遷移的問卷有23份、VB /Java對C++技能遷移的問卷有67份、VB/C++對Java技能遷移的問卷有51份。

表4：填答語言及背景語言

填答語言	背景語言	人數
VB	C++	15
	Java	8
C++	VB	31
	Java	36
Java	VB	14
	C++	37

二、資料編碼

我們針對每一位受測者的問卷進行編碼。首先在問卷的評估上，由於問卷的每一題題目有相當高的獨立性，因此我們是以題為單位進行評估，並藉以過濾掉空格數太多的題目。以題為單位進行分析的好處是我們可以獲得較多的樣本數（相較於以整份問卷為單位）。移除未填寫的記錄後，各題項之有效填答人數如表5所示。

表5：每題項回答的有效人數

題項	1	2	3	4	5	6	7	8	9	10	11
人數	135	137	122	133	101	133	100	136	56	104	94

然而，即使我們的問卷採填空方式進行，而非開放性問卷，但同樣的問題其解決的語法有時並非唯一。例如，在Java中欲串接字串與整數可直接以+號進行，亦可複雜地使用StringBulider物件來達成（參考表6）。因此，欲評估某段語法是否受技能遷移影響必須依據受測者的填答來個別評估，這工作是相當花體力與精神。

表6：Java複雜字串與整數串接

```
String stringadd（String s1,int age）
{
  StringBuilder sb=new StringBuilder（）;
  sb.append（s1）.append（age）;
  return sb.toString（）;
  /*較簡單的語法是
  return s1+age;
  一行即可*/
}
```

最後，我們將問卷空格填答的可能情況分別編碼如下：當空格填寫正確編碼為0；空格填寫錯誤，且是受遷移而錯誤編碼為1；空格填寫錯誤，但非受遷移而錯誤編碼為2。

三、信度分析

為檢驗本研究問卷之信度，我們進行Cronbach's分析。我們依據上節所述規則將每份問卷的空格進行編碼，再依不同程式語言問卷（VB、C++、與Java）進行Cronbach's α 分析。分析結果如表7所示，可知VB、C++、與Java問卷的Cronbach's α 皆為可信。顯示我們的問卷具有可靠度、一致性與穩定性。

表7：問卷信度分析

	VB	C++	Java
Cronbach's α	0.945	0.929	0.938

四、結果分析

我們依所得編碼（去除編碼2）結果計算分別在H1到H5受技能遷移而答錯的比例，並使用t檢定，檢定每一個假說（H1的虛無假說為H2 $p \leq 0.5$；H2至H4的虛無假說設為$p \geq 0.5$；H5的虛無假說設為H0: $p=0.5$）是否成立。由表8可知H1、H2、H3、H4檢定皆達顯著水準（$\alpha=0.05$）。顯示當語法相同與用途相同時，會有正遷移的影響；當語法不同而用途相同時，對學習新程式語言會受所學過的程式語言所干擾；當語法相同而用途不同時，對學習新程式語言會受所學過的程式語言所干擾；當學習者背景知識中在新程式語法中無適當對應者，對學習新程式語言會受所學過的程式語言所干擾。另外，H5未達顯著水準，表示正確與錯誤的比例大約是各半，亦即受測者在回答H5的問題時並不受遷移影響。此結果呼應我們的假說五，若新程式語言的語法在學習者背景知識中無適當對應者，對學習新程式語言將不會有遷移的現象。

表8：背景知識假說檢定

	proportion	t Value	Pr > |t|
H1	0.0580	38.47	<.0001*
H2	0.7460	22.22	<.0001*
H3	0.9855	59.67	<.0001*
H4	0.9903	5.72	<.0001*
H5	0.4560	-1.14	0.2557

註：*為顯著（在α=0.05之下）

為進一步瞭解「語法」與「用途」是否會受題目的「難易度」調節，以及「語法」與「用途」何者對技能遷移有較重要的影響，我們進行迴歸分析，將「語法」、「用途」與「難易度」當作自變數，受測者填答的表現當作應變數。為決定題目的難易度，我們分別請兩位研究生對各題進行獨立難易度評分（他們對VB、C++、及Java都有相當的了解）。結果發現，兩位研究生的評分是相當一致的，因此我們依據其評分將簡單的題目編碼為0，困難的題目編碼為1。在背景知識方面，我們以語法與用途異同進行編

碼。其分別代表語法相同編碼為1，不同設為0；用途相同設為1，不同設為0。最後的資料結果列於表9。由於資料的輸出變項是二元的（答錯或答對），故本研究以Logistic Regression進行分析。

表9：進一步探索的分析資料

語法	用途	難易度	y*	n
0	1	1	119	135
1	0	0	23	137
0	1	1	117	122
0	1	0	33	133
1	0	1	68	101
1	1	0	16	133
1	0	0	60	100
0	1	0	120	136
0	1	1	24	56
0	0	1	87	104
0	0	1	60	94

*y為受遷移而答錯該題項的人數

為了檢定調節變項（問題難易度）的影響，首先我們針對表9進行檢定調節變項（問題難易度）與自變項之間是否有交互作用的存在。其結果如表10與表11所示。

表10：Model無交互作用項

	Intercept	語法	用途	難易度
coeff.	0.4054	-1.2232	-0.2450	1.2095
P-value	0.0451*	<.0001*	0.1493	<.0001*

註：*為顯著（在α=0.05之下）

表11：Model有交互作用項

	Intercept	語法	用途	難易度	語法*難易度	用途*難易度
coeff.	1.6483	-2.2664	-1.3715	-0.5897	1.9308	1.9033
P-value	<.0001*	<.0001*	<.0001*	0.1035	<.0001*	<.0001*

註：*為顯著（在α=0.05之下）

由表10可知「語法」、「用途」、及「問題難易度」的單純主效果都是顯著的。進一步加入「問題難易度」與「語法」及「用途」的交互作用項發現，「問題難易度」的主效果消失，而交互作用項是顯著的（表11），此說明了「問題難易度」的確為調節變項。

由於「語法」與「用途」都呈現負相關的趨勢，亦即「語法」或「用途」由0（不同）變成1（相同）時，其錯誤數會由多變到少，此與我們的假說是一致的。而且語法影響的程度要高於用途與難易度（其係數為 |-2.2664|，大於 |-1.3715| 及 |-0.5897|），顯示「語法」異同的混淆所造成遷移的影響要遠高於「用途」的影響。

伍、結論與未來研究方向

程式設計是資訊相關科系的必修課程，大抵而言，資訊相關科系的學生在求學階段，不可避免地，會學習至少兩種以上的程式語言。在學習的過程中，當在已有某一程式語言背景的情況下學習另一程式語言時，學習者經常會不知不覺地混用背景程式語言，造成學習上的挫折。因此找出造成學習者混淆的因素，以避免學習者的挫折，為提昇國內學生資訊素養的第一步。本研究將程式語言之背景知識區分成「語法」及「用途」兩個構面，以技能遷移理論為主要理論基礎，藉由問卷探討「語法的異同」與「用途的異同」（自變項）對學習者表現（因變項）的影響。

由實驗結果發現學習者在「語法的異同」與「用途的異同」兩個維度上各有不同程度的遷移影響：

一、若舊程式語言與新程式語言的語法及用途皆相同時，對學習者會有正遷移的影響；

二、若舊程式語言與新程式語言的語法不同但用途相同時（例如，VB/C++/Java三種語言用於產生亂數的語法不同），學習者會受所學過的程式語言的語法所干擾；

三、若舊程式語言與新程式語言有相同的語法但卻有不同的用途時（例如，指定運算和相等判斷的混淆，在VB「=」既是指定也是相等判斷，但在Java或C++則「=」是指定運算，「==」才是相等判斷），學習者也會誤將舊語法用在新語言的不同用途上；

四、若舊程式語言的語法在新學習的程式語言中無適當對應者（例如，C++中的轉換建構子是VB/Java所沒有的），學習者會由於此知識的落差，而造成干擾；

五、相反地，若新程式語言的語法在學習者背景知識中無適當對應者（例如，Java中的界面），對學習新程式語言將不會有遷移的現象。

最後在進一步調節變數的探討上，分析結果顯示「題目的難易度」確實會調節自變項的「語法」或「用途」對學習程式語言的表現的關係，而且「語法」異同的混淆所造成遷移的影響要遠高於「用途」的影響。

本研究對實務的建議可從三方面來說明：首先，對程式語言的發展者[1]而言，應考慮所發展的語言語法是否容易跟其它語言混淆或抵觸（例如，以「=」當判斷運算子），以避免語言的學習挫折，進而達到推廣的目的。其次，對程式設計的教授者而言，除了單純地教授語言的語法本身外，更應協助學習者釐清語法與用途的異同（例如，CS1教Java，CS2教C++），以協助建立他/她們的自信。因為學生在程式設計課程上的挫敗，一開始很大部份是由於語法及用途上的混淆，而非解題邏輯上的錯誤。最後，對整合開發環境的發展者而言，在兼容並蓄地包含不同語言的同時，亦應考量對語法及用途易混淆處加以提示。例如，在asp.net中撰寫JavaScript敘述，當JavaScript敘述少了分號或語法大

[1] 隨時都會有新語言被提出來，例如用於開發Youtube的程式語言Python是1990年中才被提出。

小寫拼錯，程式並不會出錯，只是執行不出想要的結果，學習者通常找了半天才知道確實原因；同樣地，在C++中條件判斷寫成 if（x=10）程式亦不會出錯，只是x變成了10。本研究認為整合開發環境工具若能從語法及用途易混淆處著手，將提高這些工具的有用性。

本研究肴於受測者取得不易，所得研究結果是否穩健仍待更多的資料進行驗證，例如，以業界的程式設計師為受測樣本。另外，不同的程式語言的學習先後次序、學習的時間長短、以及學習者的能力都可能直接或間接地影響遷移的方向與強度，因此未來的研究方向可考慮這些因素對對學習程式語言的影響。

致謝

本研究為國科會NSC 95-2416-H-251-011 補助之成果，特此致謝；作者並感謝兩位匿名審查委員的寶貴意見及建議。

參考文獻

1. Anderson, J.R.“Skill acquisition: Compilation of weak-method problem solutions,” *Psychological Review* (94:2), 1987, pp. 192-210.
2. Anderson, J.R. *Rules of the mind. Hillsdale*, Erlbaum, NJ, 1993.
3. Bellin, D.“A seminar course in object oriented programming,” *SIGCSE Bulletin* (24:1), 1992, pp. 134-137.
4. Boszormenyi, L.“Why Java is not my favorite first course language,” *Software-Concepts & Tools* (19:3), 1998, pp. 141-145.
5. Boulay, B.D., O'Shea, T., and Monk, J.“The black box inside the glass box: presenting computing to novices,” *International Journal Human-Computer Studies* (51:2), 1999, pp. 265-277.
6. Fix, V., and Wiedenbeck, S. (1996).“An Intelligent Tool to Aid Students in Learning Second and Subsequent Programming Languages,” *Computers & Education* (27:2), 1996, pp 71-83.
7. Miller, R.“Migrating from C++ to Java,” *Chips, the Department of the Navy Information Technology Magazine*, 2000 (available online at http://www.chips.navy.mil/newarchives.html).
8. Pennington, N., Nicolich, R., and Rahm, J.“Transfer of Training between Cognitive Subskills: Is Knowledge use specific?” *Cognitive Psychology* (28:2), 1995 , pp 175-224.
9. Schwartz, R. M., and Humphreys, M. S.“Similarity judgements and free recall of unrelated words,” *Journal of Experimental Psychology* (101), 1973, pp. 10–13.
10. Singley, M.K., and Anderson, J.R. *The transfer of cognitive skill*, Harvard Univ. Press, MA, 1989.

11. Stroustrup, B.“Learning Standard C++ as a New Language,”*The C/C++ Users Journal*, 1999, pp. 1-11.
12. Thimbleby, H.“A Critique of Java,”*Software-Practice and Experience* (29:5), 1999, pp. 457–478.
13. Thorndike, E.L. *Principles of teaching based on psychology*, Mason-Henry Press, New York, 1906.

附錄

由於研究問卷題目相當冗長（受測者要填答的內容並不多，但題目本身卻相當佔版面，若以1題一個頁面的配置，三份問卷共需33頁，並不適合置於本論文中），因此在此僅列出各份問卷的第一題供讀者參考。

一、Java題

1. 設計一函式，此函式可接受主程式所傳入的一個字串及一整數，並且傳回合併後的結果。例如，依序輸入"How old are you?"及20，經過函式的運算後會輸出"How old are you? 20"。

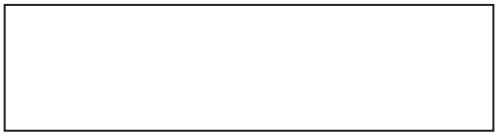
```
public class Main
_____________
        _____________stringadd (String s1,int age)
        _____________

        [          ] //請完成字串串接功能，並回傳字
串接結果

_____________

public static void main(String[] args)
_____________
        _____________="How old are you?";
        _____________ =20;
        System.out.println("String 1;"_____________s1);
        System.out.println("age:"_____________age);
```

s3=stringadd(s1,age);
System.out.println("Result after adding String 1 and age:"_____s3);
_____
_____

二、C++題

1. 設計一函式，此函式可接受主程式所傳入的一個字串及一整數，並且傳回合併後的結果。例如，依序輸入"How old are you?"及20，經過函式的運算後會輸出"How old are you? 20"。

_____ stringadd(string s1, int age)
_____


//請完成字串串接功能，並回傳字串接結果
_____

void main(int argc, char* argv[])
_____
_____="How old are you?";
_____=20;
_____
cout <<"string 1=" << s1<<endl;
cout <<"age=" << age <<endl;
s3=stringadd(s1,age);
cout << "Result after adding string 1 and age:"<<s3<<endl;
getch();
_____

三、VB.NET題

1. 設計一函式，此函式可接受主程式所傳入的一個字串及一整數，並且傳回合併後的結果。例如，依序輸入"How old are you?"及20，經過函式的運算後會輸出"How old are you? 20"。

```
Module Module1
_____
    _____ stringadd(ByVal s1 As String, ByVal age As Integer)

    
請完成字串串接功能，並回傳字串串接結果

_____

Sub Main ( )
_____
    _____ = "How old are you?"  '宣告s1字串
    _____ = 20  '宣告變數age為整數
                '宣告s3字串
    Console.WriteLine("String 1:"_____ s1)
    Console.WriteLine("age="_____ age)
    s3 = stringadd(s1, age)
    Console.WriteLine("Result after adding String 1 and age:"_____ s3)
    Console.ReadLine()
    _____
    _____
```