STUDYING THE CARRY-OVER EFFECT IN END USER COMPUTER TRAINING: PROPOSITION OF A RESEARCH AGENDA

Yuan Li
Columbia College, yli@colacoll.edu

Kuo-chung Chang
Yuan Ze University, changkc@yzu.edu.tw

Follow this and additional works at: http://aisel.aisnet.org/pacis2008

Recommended Citation
http://aisel.aisnet.org/pacis2008/249

This material is brought to you by the Pacific Asia Conference on Information Systems (PACIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in PACIS 2008 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
STUDYING THE CARRY-OVER EFFECT IN END USER COMPUTER TRAINING: PROPOSITION OF A RESEARCH AGENDA

Li, Yuan, Columbia College, 1031 Columbia College Drive, Columbia, SC 29203, USA, yli@colacoll.edu
Chang, Kuo-chung, Yuan Ze University, Chung-Li 320, Taiwan ROC, changkc@yzu.edu.tw

Abstract

The carry-over effect in end user computer training, also known as the transfer of computer skills, refers to the transfer of a person’s computer capabilities from one application to another. The importance of the carry-over effect is unquestionable, but its nature has not been well understood. In this proposal, a research agenda is developed to analyze the nature, the roles, and the antecedents of the carry-over effect. Three fundamental questions are asked: 1) what is the carry-over effect and how to measure it? 2) What roles does it play in end user computer training? And 3) what factors determine the carry-over effect? Propositions are developed to give directions in further empirical research, and some preliminary evidences are discussed. Potential implications for research and practice are discussed.

Keywords: Carry-over effect, Computer skills transfer, Computer software similarity, Computer self-efficacy, End user computer training
1 INTRODUCTION

In end user computer training field, scholars are interested in whether a person’s previously learned software applications have impact on the learning of new software applications, and how to facilitate the transfer of a person’s computer skills across the applications (Singley and Anderson 1985; Smelcer and Walker 1993; Urbaczewski and Wheeler, 2001). Earlier studies show that computer skills transfer does exist, where users can apply knowledge learned from previous software in the learning of new but similar software, such as text editing, programming, and utility programs (Polson, Muncher and Engelbeck, 1986; Singley and Anderson 1985; Urbaczewski and Wheeler, 2001). In a more recently study, Agarwal, Sambamurthy and Stair (2000) find that the impact of the general Computer Self-Efficacy (CSE) belief on the Specific CSE (SSE) belief diminished for software applications subsequent in the training sequence; instead, SSE belief of the subsequent software applications is directly determined by the SSE belief of earlier software applications. In other words, earlier learned software applications in the training sequence have more impact on the learning of new applications than general computer capability beliefs. Agarwal et al (2000) term this relationship the carry-over effect. Although different terms are used, both computer skills transfer and the carry-over effect mean that learning can be transferred in the computer training context, although the latter is based on the subjective assessment of a person’s computer capabilities.

Understanding the transfer of computer skills and the carry-over effect in computer training has great importance. Studies show that skills transfer leads to less time and effort needed to learn new applications for improved performance (Singley and Anderson 1985; Scholtz and Wiedenbeck 1990; Olfman and Shayo 1997), which is valuable for computer training. Other research shows that negative transfer may cause accident in work place (Manns and Carlson, 1992), which should be prevented when old software is upgraded to new one. Considering the fast changes in Information Technologies, the surging requirement of end user computer training in both companies and educational institutes, and the elevated reliance on computer skills to improve individual productivity and organizational performance, the analysis of computer skills transfer and the carry-over effect is still importance.

To our best knowledge, the study on computer skills transfer and the carry-over effect has not been fully developed, with two limitations exist. The first limitation is that most research is conducted from the software application’s perspective, analyzing how the similarities and sequences of the software applications may influence skills transfer between the applications. To fulfil this purpose, many studies use the software applications as the primary control variable in experiments, testing whether a group of users assigned to learn those software applications achieve better performance than other users (Polson et al, 1986; Singley and Anderson 1985). However, individual difference in the training context is seldom analyzed, i.e., how individuals differ in their skills transferred is not directly analyzed. The second limitation is the lack of a context-free construct that theorizes the extent of skills transfer or the magnitude of the carry-over effect from individual’s perspective. In fact, computer skills transfer is treated as a nominal concept, which is analyzed through other surrogate measures such as time and effort saving within specific research context. To what extent the skills are transferred, especially from the user’s perspective, is not directly measured.

Because of these limitations, further research is restricted. For instance, most research in this area focuses on a small number of software attributes (e.g., user interface and keystrokes) or types of applications (e.g., text editing and programming) that are manageable in experimental settings. This undermines the generalizability of the research results to other software applications, including more complex Enterprise Systems. In addition, many other factors that are believed to have impact on computer skills transfer are seldom analyzed, including training methods and personal learning styles (Bostrom, Olfman and Sein, 1990). This further limits the value of the research results in end user computer training field. Compared to the potential value of skills transfer in computer training, these limitations should be properly addressed.
Since users play a central role in computer training, emphasis should be shifted from application-centred research to user-centred research in order to develop a broader view of skills transfer. A prerequisite of further research is the development of a context-free, focal construct of computer skills transfer or the carry-over effect that is able to directly measure the extent of skills transfer from user’s perspective. Due to the fact that the “carry-over effect” concept is built upon user’s self-assessment of computer capabilities, i.e., SSE, and it does not rely on particular software applications, this concept is chosen for the development of the focal construct. In this proposal, we develop a research agenda to analyze the nature of the carry-over effect and its antecedents and impact in computer training. Specifically, we aim at answering three questions:

- What is the carry-over effect and how to measure it?
- What are the roles of the carry-over effects in computer training?
- What are the antecedents of the carry-over effects and how to improve the computer skills transfer?

In answering the first question, we broaden our view on the carry-over effect and further highlight the importance of developing a context-free measurement of the construct. For question 2 and 3, we particularly discuss the consequences of the carry-over effect and its antecedents. An integrative model is then developed from the analysis. Some preliminary evidences are discussed, and the potential implications for research and practice are addressed as well.

2 DEVELOPMENT OF THE RESEARCH MODEL

2.1 Carry-over Effect: Definition and Measurement

In general, the carry-over effect is a phenomenon widely recognized in many disciplinary fields, such as psychology, ecology, statistics, pharmacology, chemistry, and business. It usually refers to something, such as a skill, a psychological status, or a business activity, transferred or extended from an earlier time or another place. In end user computer training context, the carry-over effect refers to the amount of transfer of a person’s computer skills or capabilities from one software application to another (adopted from Agarwal et al, 2000), which is also known as the transfer of computer skills (Singley and Anderson 1985). For instance, in a training program where computer users learned to use the Windows 95 operating system and then the Lotus-123 spreadsheet software, it was found that users who had a higher capability of using Windows 95 also had a higher capability of using the spreadsheet software (Agarwal et al, 2000). In another research where users were trained to use database management systems (DBMS), it was found that those who had some previous knowledge of using DBMS reported having more knowledge after the training of using a new DBMS, compared to those who had no previous DBMS knowledge (Shayo and Olfman, 1998). Similar outcomes have been reported in other computer training contexts, such as text editing (Singley and Anderson, 1985) and computer programming (Scholtz and Wiedenbeck, 1990; Urbaczewski and Wheeler, 2001).

As discussed above, a prerequisite of further theory building regarding the carry-over effect construct is the development of a scale that measures the extent of skills transfer by individuals. Several earlier efforts have been made to develop different protocols based on specific software contexts, such as reduction in learning time, total keystrokes tried, increase in typing rate in text-editing (Singley and Anderson, 1985), time needed to finish a training (Polson et al, 1987), and students’ self-reported grades in learning programming languages (Powell, 1997). Other research used alternative surrogates to measure the carry-over effect, such as the perceived ease of transfer, adopted from the perceived ease of use measures (Shayo and Olfman, 1998).

The above two types of measurement approaches have their limitations. The first approach, due to its reliance on specific software context and task, lacks generalizability to other software contexts. For instance, the measures of the text-editing skills based on keystrokes are not directly applicable to
programming languages or database software, and time needed to finish a task is not suitable for complex tasks. More importantly, relying on the performance indicator of using the second software application poses direct conflict with the definition of the carry-over effect, as the carry-over effect deals with the portion of the skills transferred from one application to another, which does not count the new skills learned in the second applications. New skills may be learned in the training process, which also influences the performance. In addition, other factors such as task characteristics, level of persistence, amount of effort, and environmental motivation all have impact on training performance (Marakas, Yi and Johnson, 1998). These suggest that the performance of using the second software is not an accurate indicator of the carry-over effect.

The second approach, which was adopted from the measurement of perceived ease of use, shows problems in construct validity. According to the study by Agarwal et al (2000), perceived ease of use is an indicator of training performance; therefore it should not be confused with the skills transfer measures. In sum, available approaches, although having merits in understanding the carryover effects during the early years, do not perform a satisfying job in measurement; a new instrument is needed.

The new scale of measuring the carry-over effect should give direct assessment of the proportion of skills transferred. It should also be context independent and applicable to different software applications. A dominant approach to measuring personal computer capability constructs in the IS field, such as CSE (Compeau and Higgins, 1995), SSE (Johnson and Marakas, 2000), and Personal Innovativeness in IT (Agarwal and Prasad, 1998), is based on individuals’ self-assessment of their capabilities. This approach is proved to be valid across research contexts. Since the carry-over effect stands for the transfer of computer skills and capabilities of individuals, it can be treated as a latent construct and measured via computer users’ self-assessment. Further research is needed to develop such a scale.

2.2 The Roles of the Carry-over Effect

Observations from existing literature suggest that the carry-over effect exists when there is an association between a person’s previously learned computer skills and newly developed skills. This association is, however, not constant since not only positive transfer but also negative transfer (Manns and Carlson, 1992) and even non-transfers (Shayo and Olfman, 1994) have been observed. Therefore, in addition to the association between the computer skills, a better understanding of the carry-over effect can be achieved via the analysis of the underlying process through which the computer skills are transferred across applications. To interpret this process, we draw upon the common element theory, which was first proposed by psychologist Thorndykein. This theory suggests that training in one kind of activity will transfer to another only if the activities share common elements (Singley and Anderson, 1985, p.404). In the computer training field, it means that elements such as keystrokes, command lines, menu items, and interface, etc., that are common across applications, enables the reuse of previously learned computer skills in new applications. This theory has been widely adopted in skills transfer research. For instance, Singley and Anderson (1985) applied the theory in text-editing research and analyzed the common editing skills such as inserting, deleting, and replacing. Polson et al (1986) applied this theory in the computer utility tasks, examining other common elements such as disk formatting, duplicating, and file name changing. These studies confirmed the roles of the common elements in skills transfer and made the conclusion that the correlation between the skill sets of two software applications is determined by the amount of skills transferred. Figure 1 shows this relationship.
Figure 1. Impact of the carry-over effects in computer training.

Figure 1 further shows that although skills can be transferred between applications, the amount of transfer, or the carry-over effect, is not constant. Strong carry-over effect happens when large amount of skills can be transferred, such as using a newer version of the software; weak carry-over effect happens when innovative software is adopted by users with no prior experience. Putting these together, we can say that the carry-over effect is a factor that moderates the association between the computer capabilities, and this construct is itself determined by other factors. A positive carry-over effect suggests a positive association between the computer capabilities, a negative carry-over effect suggests a negative transfer, and a zero carry-over effect implies no association between the computer skills.

To support the moderating role of the carry-over effect, we need certain measures of a person’s computer skills both before and after the training program; the software specific Computer Self-Efficacy (SSE) is selected. SSE captures a person’s self-assessment of computer skills, which can be readily adjusted to fit a specific application context, such as database, graphics, or spreadsheet. In addition, it is found that SSE assessment of a software application was influenced by the SSE assessment of an earlier software package (Agarwal et al., 2000; Yi and Davis, 2003). Given the analysis above, we propose that:

**Proposition 1**: The carry-over effect (COE) moderates the relationship between a user’s perceived SSE beliefs regarding the two software applications in the learning sequence (i.e., SSE1 and SSE2).

Figure 2 illustrates the role of the carry-over effect, given the selected computer capability measures (i.e., SSE).

**Figure 2. The impact of the carry-over effect in end user computer training.**
2.3 Exploring the Antecedents of the Carry-over Effect

The above analysis illustrates the roles of the carry-over effect in computer training. It also shows potential factors that have impact on the carry-over effect. Once the impact of the carry-over effect is confirmed, the next issue is to recognize the factors that maximize the carry-over effect. This would help scholars and practitioners to develop training programs to facilitate skills transfer.

Up to now, many factors are thought to have impact on the carry-over effect. In addition to the software similarity discussed above, scholars have also recognized training methods, the assumptions people have about information systems, the organizational context of information systems usage, and the nature of individual characteristics and goals (Bostrom et al, 1990; Shayo and Olfman, 1994; Olfman and Shayo, 1997). Although it is challenging to incorporate all these factors in a single research, important ones should be included in the research framework to provide guidance to researchers and practitioners. Primarily drawing upon Bostrom et al’s (1990) framework, we discuss the impact of several factors next.

Similarity of software attributes. Similarity of software attributes is a factor that has strong impact on skills transfer, according to the common element theory and previous research. In addition, Anderson’s ACT theory (ACT stands for Adaptive Control of Thought) provides a better view on the impact of this factor (Singley and Anderson, 1985). The ACT theory suggests that production rules exist in human cognition and behaviour, which are the elements implied in the common element theory. The extent to which skills can be transferred between two tasks depends on the degree of overlapping between the production-rule sets of those two tasks. Using this theory, Singley and Anderson (1985) proposed a two-component model of transfer, stating that the set of production rules of a task consists of two components: a general component that intersects with other production sets, and a specific component unique for a task. The general component, which stands for the common elements between the two tasks, is the basis of skills transfer: the larger the sets overlap, the larger the general component and the greater the amount of transfer. At the extreme where the production set of a task is nearly a subset of another, as Singley and Anderson illustrate, the skills can be maximally carried over.

Although the ACT theory makes progress in specifying what the common elements are in the carry-over effect, the understanding of the common software attributes is still limited, with studies focusing on a small number of software attributes from a short range of applications. Based on studies that focus on high-level software attributes in more recent studies (e.g., Besnard and Cacitti, 2005; Harvey and Rousseau, 1995; Scholtz and Wiedenbeck, 1990), we recognize the following dimensions of similarities between software applications:

- Function: a feature in an application that is constructed to perform a specific task or action.
- Interface: the appearance and structure of how the functionalities in an application are organized.
- Process: a procedure involved in performing a series of functions or steps in an application to complete a task.
- Syntax: the set of rules that constitute the statements or structure of the programming language in an application by combining the elements or characters into permitted construction.
- User document: user files that are created in an application with certain features, structures, appearance, and components.

We suggest that these software attributes together will influence the carry-over effect, and propose that:

**Proposition 2**: Similarities between software applications have positive impact on the carry-over effect.

Software training methods. Training methods and others factors related to curriculum design may have
impact on skills transfer, too. For instance, the number of software packages taught simultaneously and their sequence are expected to have potential impact on skills transfer (Urbaczewski and Wheeler, 2001); the result is, however, mixed. The mixed result can be interpreted by other factors overlooked in the research, such as software attributes and individual difference. Another factor discussed by Bostrom et al (1990), which is analogy-based training, is noteworthy. Analogy-based training refers to the mapping of knowledge from similar systems that are familiar to the users. For instance, in introducing Microsoft Excel, an instructor may compare its menu system to that of Microsoft Word, which enables the students to quickly master the basic features of the Excel application. While other training methods may also influence skills transfer, in general it suggests that training methods is an important factor in determining the carry-over effect. We propose that

**Proposition 3**: Analogy-based training has positive impact on the carry-over effect.

*Individual characteristics*. Another factor often included in computer training research refers to individual difference in learning. Several constructs have been analyzed in previous studies, including, such as Personal Innovativeness in IT (PIIT), previous experiences (Agarwal et al, 2000), and personal learning style (Bostrom et al, 1990). The first two factors were included in Agarwal et al’s (2000) study, where the direct impact of these factor on the general CSE were analyzed, but not the correlation between SSE1 and SSE2. We argue that PIIT and previous experience both have direct impact on skills transfer, given that a person with higher PIIT and more previous experience will find it easier to map knowledge from a previous application to a new one (Bostrom et al, 1990). Similarly, Bostrom et al (1990) discuss the role of individual learning styles such as mapping by analogy in computer training, and suggest that this learning is crucial to skills transfer as well. Putting these all together, we propose that

**Proposition 4**: Individual factors, including PIIT, previous experiences, and learning style, have positive impact on the carry-over effect.

Although other factors may exist, software attributes, training methods, and individual factors build a strong basis of understanding the carry-over effect. Their relationship is illustrated in Figure 3.

![Figure 3. The antecedents to the carry-over effect.](image)

**2.4 Towards an Integrative Research Framework**

In the above we analyzed the nature of COE, its impact on skills transfer, and its antecedents. Aggregating these relationships together, we develop the following research model (Figure 4) to show an integrative view of the carry-over effect. It should be mentioned that the antecedents of the COE construct is not comprehensive, as many other factors may also have impact on the extent of COE.
3 PROPOSITION OF THE RESEARCH AGENDA

We suggest a two-step approach (Fornell and Larcker, 1981) for the verification of the research model, given the factor that a new construct (COE) is to be measured. A complete research agenda would seem as follows:

- Scale development: The first step is to develop a measurement scale of the COE construct and test it within a nomological network with some of antecedents or the dependent variables;
- Justifying the impact of the COE construct: Rigorous tests should be conducted to examine the validity of the COE construct and its measurement. These tests involve different combinations of software applications from various application areas, such as word processing, spreadsheets, graphics, databases, web development, and even programming for IS majors.
- Exploring the impact of the antecedents: the impact of the antecedents should be tested once the moderating effect of the COE is confirmed. Both surveys and experiments can be conducted to analyze their relationship.
- Towards integration: the last effort would be to test the integrative model in Figure 3.

A prerequisite of this line of research is the development of a measurement scale for the COE construct. An earlier effort was made by the authors to fulfill the task. Following the typical procedure of scale development, the authors developed an 8-item measurement scale, as shown in Table 1. The data collected shows that the COE construct has high reliability based on these items, and is a unidimensional construct. These items measure a person’s computer skills transfer from one application (a) to another (b) in different aspects.

1. My skills learned from (a) are helpful in learning how to use (b).
2. My knowledge of (a) saves the efforts in learning (b).
3. I can apply the skills I learned from (a) to the use of (b).
4. The skills I learned from (a) can be reused in (b).
5. My knowledge of (a) can be applied to the understanding of the features of (b).
6. The knowledge I learned from (a) enables me to spend less time to learn (b).
7. My skills of (a) are helpful in recognizing the features of (b).
8. My skills of (a) enable me to find the needed features of (b) to solve a problem.
Note: “a” refers to the first software learned, and “b” refers to the second software.

Table 1. Measurement Items of the COE Construct

To further test the validity of the COE construct in its nomological network, we select one of the antecedents, the similarity between software applications, and test its relationship with COE. The result shows that these two constructs are highly related ($\beta = 0.439, t = 6.33$). In other words, higher level of software similarity results in higher extent of COE. These evidences are however still preliminary and further analysis should be conducted to provide more systematic test of the research model.

4 POTENTIAL IMPLICATIONS FOR RESEARCH AND PRACTICE

The current research has potential implications for research and practice. The development of a context-free measurement of the COE construct addresses the issues of conflicting measurements in previous studies, making it possible to conduct research across contexts and compare the results directly. The analysis of the moderating effect helps to clarify the conflicting outcomes regarding the positive, negative and no transfer between software applications. More importantly, understanding the antecedents of COE helps to design research that may generate consistent results by incorporating various factors. All these would promote out understanding of computer skills transfer.

For practitioners, the results of this study may provide a basis to develop better training programs in facilitating the learning process and enhancing learning performance, e.g., to manage the specific content to be covered and the sequence in the training programs (Agarwal et al, 2000). For the end users who need to learn new software throughout their career life, a better understanding of COE provides guidelines for comparison across applications and for rationalizing their personal training programs to achieve better training performance. This would make it easier to master new applications. Lastly, understanding COE helps software developers to manage the features of software so as to design a system that is easy to learn and to use.

References


