Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1999 Proceedings

Americas Conference on Information Systems (AMCIS)

December 1999

Learning Spreadsheet Software in the Traditional & Synchronous Modes: A Model and A Pilot Study to Investigate End User's Performance & Satisfaction

Shailendra Palvia
Long Island University

Learning Spreadsheet Software in the Traditional & Synchronous Modes: A Model and A Pilot Study to Explore End User's Performance & Satisfaction

Shailendra C. Palvia, Long Island University, 516-299-2302, spalvia@liu.edu

Introduction

A survey of U.S. senior information systems (IS) executives found that, "organizational learning and use of IS technologies," ranked fifth out of a list of 20 critical IS management issues (Niederman, et al., 1991, p. 480). In the global context, this issue ranked sixth out of a list of 20 critical IS issues facing IS executives in India during 1988 (Palvia and Palvia, 1992); training and education of DP Personnel ranked fifth most important issue in Singapore around 1987 (Rao et al, 1987); retaining, recruiting, and training was the number one issue for Hong Kong around 1989 (Burn et al, 1993); and recruiting, training, and promoting IS staff ranked eighteenth most important issue among 20 critical issues around 1994 in Taiwan (Palvia and Wang, 1995).

Knowledge of and skills in the use of word processing (e.g., Word, WordPerfect), spreadsheet (e.g., Excel, Lotus), and database software (e.g., Access, Paradox) represent the basic computer literacy requirements for white-collar knowledge workers. A survey of knowledge and skills requirements of IS specialists and end-user personnel (Nelson, 1991) found that (a) both IS specialists and end-users are most deficient in the area of "general IS knowledge," (b) people need more "organizational knowledge," and (c) end-users need more knowledge in the use of software packages. Both academic institutions and corporate training centers are spending billions of dollars worldwide to provide the necessary training in the use of these software packages. According to Industry Report on 1997 Training Budgets (Johnston and Lou, 1997), U.S. organizations with 100 or more employees budgeted a total of $58.6 billion for formal training - a 5% increase over the previous year (not adjusting for inflation). The number of software training companies and individual consultants has been increasing exponentially to cater to the unmet demand for training. According to the 1997 Industry Report cited above, 38% of training functions are being designed and developed by outside contractors and 32% of these are delivered by some outside contractors. Given this scenario, academic institutions are increasingly expected to educate their students in these skills before they graduate. Research that can systematically and scientifically determine cost effective approaches to imparting computer software education to the diverse segments of students graduating each year has the potential of a phenomenal payoff.

Hermanutz (1991) conducted experiments to evaluate and compare the effectiveness of computer-based and traditional teacher-centered methods to teaching word-processing software. Czaza et al. (1986) evaluated the effectiveness of three training strategies -- instructor, manual, and computer -- in teaching naive computer users to use word-processing software. In another study, Bowman et al. (1995) compared a control group, taught by traditional lectures, with an experimental group using Computer Based Training (CBT). Marks (1992) found that training videos represent an alternative to traditional classroom training or interactive methods for training employees. She reported studies that indicate that people remember concepts if they are linked to images, which can be provided by video training.

It is easy to notice that nobody, to the best of author's knowledge, has evaluated alternative approaches to instructor driven computer software teaching in a classroom setting. This research is initiating investigation into this unexplored area.

A Model for Evaluating Computer Software Education Approaches

The premise of this model (Exhibit-1) is that a learning mode (or training method)[1] in either individual setting or group setting will be effective in different ways for different kinds of end-users (based on their learning styles) and for different types of tasks of learning. Furthermore, the quality of education provided with these modes can be measured in terms of user satisfaction, efficiency of the learning mode (training method), and performance.

End-User Learning Style

End-users differ in their learning styles. Importance of learning style in end-user training is the focus of an article in MIS Quarterly (Bostrom et al 1990). This study makes recommendations for software training methods based on four end-user learning styles: Converger, Assimilator, Diverger, and Accomodator. These four learning styles represent four quadrants generated by two learning modes as postulated by Kolb's theory (1971). "Level of Involvement" mode can be active

[1] This artcle, will use the term "learning mode" and "training method" interchangeably. Please note that the former term is from the perspective of a student whereas the latter term is from the perspective of an instructor.

experimentation at one extreme to reflective observation at the other extreme. "Type of conceptualization" mode can be abstract conceptualization at one extreme to concrete experiences at the other extreme. There are other studies that support our model parameter of End-User Learning Style.

Task Characteristics

Task characteristics can have a significant impact on the dependent variables in the model. In the context of learning software, the characteristics are different for word processing, spreadsheet, presentation, and database management software. Furthermore, for a specific software like Excel, characteristics differ based on the lesson being elementary or advanced (e.g., covering concepts like absolute and relative addressing.)

Based on extensive literature survey -- task size, scope, and complexity can have a significant impact on learning software. Kolb (1981) argues that because of the experiential nature of learning, different learning situations are necessarily different experiences. A subject may, therefore, prefer one style in one situation (task) and a different style in another. This points to the possibility of interaction effects of these two variables.

End-user Satisfaction Measures

The instruments used to measure user satisfaction as articulated in Bailey and Pearson (1983) and Ives et al (1983) measured impact through semantic differential scales (Melone, 1990). Galletta and Lederer (1989) have reported some problems with these instruments. Doll and Torkzadeh (1988) proposed an end-user computing satisfaction instrument. This instrument emphasizes the cognitive or belief aspects of attitudes in a short, easy-to-use, application specific instrument using Likert-type scales. The twelve questions in this instrument cover five dimensions -- content, accuracy, format, ease of use, and timeliness. While these instruments measure end-user satisfaction with the use of information systems applications, our focus in this research is in measuring end-user satisfaction with the process of providing software training.

Efficiency Measures

Efficiency measures the speed of learning a particular concept or feature pertaining to software. Sometimes, the terms -- productivity and efficiency are used interchangeably. Productivity can be measured in terms of the time taken to complete a given task. Conversely, it can also be measured in terms of the amount of task completed in a given amount of time.

Performance Measures

Instructors in schools, colleges, and universities have been struggling to determine the approaches to test computer software knowledge. For example, the Computer Science and Telecommunications Board (CSTB) of the National Research Council (NRC) is currently working on a task to explicate the various dimensions of what may be called Information Technology literacy i.e., what everyone needs to know about IT.

Instructors have typically used true or false, multiple choice, and short descriptive type questions to test the knowledge and skills about software. Students have complained that they can perform better if they are tested directly on how to use computer software. Several instructors have instituted software use "performance" tests to gauge students' software use skills. These tests may include: a) creating a professional document using a word processing software like MS-Word or WordPerfect, or b) building a worksheet using spreadsheet software like Lotus or Excel, or c) creating a database and formulating queries using a database software like Paradox or Access. Perhaps, such tests of proficiency in the use of a software can be called tests of **computing (how to use the computer?) literacy** skills. Then, what do we call the traditional paper and pencil/pen tests? Should those tests be completely done away with in light of the more popular expeditious computing literacy tests?.

A computing literacy test measures just computing skills – ability to tinker with different keyboard buttons, click the two mouse buttons on different icons and menu options, and the ability to persevere and persist by experimenting and exploring to solve problems and achieve the goals using myriad HELP routines and options. But whether one has grasped the fundamentals of software along with its important nuances is a totally different matter. That grasp of fundamentals is in the domain of knowledge as opposed to skills. That knowledge can be effectively tested with the traditional paper and pencil mode. This testing is what we can call a knowledge test or **computer literacy** test. This test can be further broken down into two components: memory recall and comprehension. Memory recall questions generally include -- true/false, fill-in-the-blanks, match answers with statements, and multiple choice type questions. Such tests lack the rigor of testing deep knowledge – knowledge that helps in critical thinking.

As students become mature in their junior and senior years, instructors have resorted to testing this deep knowledge by asking students to:

- answer complex multiple choice questions
- answer short essay type questions;
- analyze a situation (case study), provide diagnostic and prognostic analysis, and suggest solution(s);
- solve mathematical or logical problems by applying learned techniques and algorithms in unique ways.

There is considerable literature support for this dichotomy (computing or computer literacy) or really a taxonomy of three levels (computing, memory recall, and critical thinking) of learning and testing knowledge accordingly.

Bloom's well known hierarchical taxonomy (Bloom, 1956/1984) of six cognitive learning objectives has a bearing on the classification scheme for software learning. Bloom's taxonomy was developed to be used in the context of existing educational units and programs, to be logical and internally consistent, to be consistent with current understanding of psychological phenomena, and to be neutral and free from value judgements. These six objectives arranged in the sequence of lower-to--higher types of learning are (Bloom, 1956/1984):

a. **Knowledge**: learners have knowledge of and ability to recall or recognize information.
b. **Comprehension**: learners understand and can explain the knowledge in their own words.
c. **Application**: learners are able to use knowledge in real situations.
d. **Analysis**: learners are able to break down complex concepts or information into simpler, related parts.
e. **Synthesis**: learners are able to combine elements to form a new, original entity.
f. **Evaluation**: learners are able to make judgements.

Boom's taxonomy has been a widely accepted logical explanation of learning levels. How do we go from "no knowledge" about a domain to "eureka" feeling about that domain? While learning a new concept, most of us go through several stages of learning:: the first stage involves some kind of cramming or rote memorization, while the last stage (that may happen in weeks, months, or years) may suddenly create an awareness of the understanding of the fundamentals of that topic (so called "eureka" feeling) freeing the person of the need to memorize. Bloom's taxonomy essentially provides us with a six-layers or levels of learning between these two extremes of "no knowledge" and "eureka knowledge."

Testing the Model for Computer Software Education in Group Settings

This research focuses on learning in group settings. There are two dimensions on which one can classify teaching computer software in group settings. Dimension One is students having or not having computers at the time of instruction for immediate practice. Dimension Two is instructor having or not having computer (for immediate demonstration) at the time of instruction. The classification below (Exhibit-3) can be described as follows:

- **Traditional:** Instructor teaching in a traditional mode lecturing and using chalkboard (or easel board) as necessary and students listening and taking notes so that they can practice immediately after teaching.
- **Delayed:** Instructor teaching with computers and students listening and taking notes so that they can practice immediately after teaching.
- **Asynchronous:** Instructor teaching in a traditional mode lecturing and using chalkboard (or easel board) as necessary and students practicing concurrently using computers.
- **Synchronous:** Instructor teaching with computers and students practicing concurrently using computers.

Research Hypotheses

This paper explores if the "synchronous" learning mode is superior in terms of efficiency, performance, and satisfaction compared to the "traditional" learning mode. The synchronous learning mode allows both the instructor and students to have computers in front of them for online demonstration and concurrent practice respectively. In contrast, in the "traditional" learning mode, neither the teacher nor the students have access to computers during the delivery of the lesson.

The synchronous mode will be used as an anchor for hypotheses generation. According to the conventional wisdom among IT educators and professionals, online demonstration and presentation of a computer software with students practicing what is being taught concurrently is the best way to impart computer software education. Given this context, the hypotheses to be tested are:

1. Improvement in Memory Recall score using the synchronous mode is significantly greater than that using the traditional mode.

Two *additional implicit hypotheses* are:

a. The traditional mode of instruction and learning improves Memory Recall score significantly.
b. The synchronous mode of instruction and learning improves Memory Recall score significantly.

2. Improvement in Computer Software Critical Thinking (Comprehension) using the synchronous mode is significantly greater than that using the traditional Mode.

Two *additional implicit hypotheses* are:

a. The traditional mode of instruction and learning improves Software Critical Thinking score significantly.
b. The synchronous mode of instruction and learning improves Software Critical Thinking score significantly.

3. Improvement in Software Computing Literacy using the synchronous mode is significantly greater than that using the traditional Mode(alternative hypothesis).

Two *additional implicit hypotheses* are:

a. The traditional mode of instruction and learning improves Computing Literacy score significantly.
b. The synchronous mode of instruction and learning improves Computing Literacy score significantly.

4. Use of synchronous mode of software education is significantly more satisfying to the students than the use of traditional mode.

Research Methodology, data analysis and conclusions will be provided by the author to those requesting.