Effect of Software Feature Training on Beliefs, Use, and Performance: Using the Benford's Law Feature of Generalized Audit Software

Hyo-Jeong Kim
Business School, University of Colorado Denver, Denver, CO, United States., Hyo-Jeong.Kim@ucdenver.edu

Michael Mannino
Business School, University of Colorado Denver, Denver, CO, United States., michael.mannino@ucdenver.edu

Effect of Software Feature Training on Beliefs, Use, and Performance: Using the Benford's Law Feature of Generalized Audit Software

Hyo-Jeong Kim
University of Colorado Denver
Hyo-Jeong.Kim@ucdenver.edu

Michael Mannino
University of Colorado Denver
Michael.Mannino@ucdenver.edu

ABSTRACT

This experimental research investigates the effect of software training at the feature level, an area infrequently studied in technology training research. We have adopted the beliefs, use, and performance constructs from adoption and training research, and tested them with the Benford's Law feature, an advanced software feature of Generalized Audit Software (GAS). Through the analysis of 56 audit professionals, we found that software feature training increases the belief of using software features and the use of those software features. However, software feature training did not have an immediate effect on performance. Rather, auditors spent more time practicing the Benford's Law feature beyond the supplied training material and applying the Benford's Law feature to other audit tasks.

Keywords

Software features, software adoption, software training, beliefs, use, performance, software feature training, feature beliefs, software beliefs

INTRODUCTIONR

Software features, tools that are available in software packages, have been inefficiently utilized by software users. Only 10 percent of software features in MS Office were used by software users (Sun and Zhang 2008). Especially advanced software features were less accepted by internal auditors (Kim et al. 2009). Software training should be carefully developed to improve the efficiency of software feature use.

Most training research has focused on the impact of software training as a whole (Gist et al., 1989; Torkzadeh and Koufteros, 1994; Compeau and Higgins, 1995b; Gilmore, 1998; Bedard et al., 2003). Only Bolt (2001) examined the effect of software training with a software feature, solver of MS Excel. Focusing only on the use of software is not sufficient to improve task performance because the use of software is heavily affected by software features. So, software training should be more carefully investigated with software features to improve the efficiency of software use and the performance of tasks.

Software features need more theoretical and empirical support from research. Post-adoptive behaviors have been researched with technology features because the usage behaviors of technology features were quite different at the post-adoptive stage (Hiltz and Turoff, 1981; Hsieh and Robert, 2006; Jasperson et al., 2005; Kay and Thomas, 1995; Singletary et al., 2002; Sun and Zhang 2008). The modified use of technology features have been investigated while developing a better measure of system use (Barki et al., 2007; Burton-Jones and Straub, 2006; Sun and Zhang, 2008). Technology features have also affected the perception of technology (DeSanctis and Poole, 1994; Griffith, 1999; Harrison and Datta, 2007; Kim et al., 2009). Although these research efforts have been started with different interests and motivations about technology features, these efforts have found evidence that technology features impact technology use and related phenomena. Therefore, theoretical development and empirical evidence should be added to technology feature research to clarify the effect of software features.

To address the need for training improvement and gaps in research about software features, we conducted an experimental study to examine the effect of software training at the feature level. We adopted the constructs of beliefs, use and performance from technology adoption and training research and tested the model with the Benford's Law feature, an advanced software feature of GAS[1] designed to identify potential fraud in corporate data. Software feature training,

[1] GAS is a software package help auditors perform querying, sorting, summarization, stratification, and any other analytical tasks for audit processes. ACL and IDEA are most commonly used GAS.

combining practical software training and a lecture of conceptual background of Benford's Law was provided to the treatment group. Then we compared the results of training and no training group. The effects of software feature training on the beliefs of using software features, the use of software features, and the performance of audit tasks were tested in this research.

Through analysis of data from 56 audit professionals, we found that software feature training had a positive influence on feature beliefs and feature use, but had no influence on performance score and a positive influence on performance time in a short period of training time. Software feature training had no immediate effect on performance score, yet increased performance time because GAS users spent more time to use both the newly trained software feature and the software feature that they already knew. Furthermore, auditors extended the use of this new software feature to other audit tasks.

This research substantially contributes to technology feature research by adding empirical evidence of the effect of software training at the feature level. Commonly found post adoptive behaviors, feature extension and task extension, were observed in this research, which strengthens the results of previous feature research. Moreover, this research applies feature and task centric approach for researching the effect of training on performance. Computer self-efficacy was categorized to the feature level and 6 audit tasks were used to measure the effect of training.

The remainder of this paper consists of five sections. The first section presents the hypotheses supported by theoretical backgrounds. The second section provides details about the research methodology including the subject, experimental design, experimental procedure, instrumentation, and data analysis. The third section shows the result of ANOVA and regression analysis. The fourth section discusses the result of analysis. The conclusion mentions the contributions and limitations along with directions for future study.

HYPOTHESIS DEVELOPMENT

Definition of Software Features

Technology features have been broadly defined in previous IS research. The definition of structural features includes rules, resources, and capabilities offered by a system (DeSanctis and Poole, 1994). Another definition of technology features includes the building blocks or components of a technology (Griffith, 1999). Features are also defined as "vendor-created software tools designed to complete tasks on behalf of users (Harrison and Datta, 2007, 300)", which is a most narrow and close definition of software features that we used in this research. Moreover, the attributes, characteristics, or functions of a technology are included in the definition of technology features (Kim et al., 2009). The previous research on technology features have covered a wide range of definitions, so we narrowed down the definition of technology features to software features defined as tools that are available in a software package.

Feature and Software Beliefs

Beliefs are positively affected by software training. Technological support, including training, is positively associated with perceived ease of use (Bedard et al., 2003; Chau, 1996), perceived usefulness (Agarwal and Prasad, 1999) or both (Igbaria et al., 1997). Computer self-efficacy is also positively affected by software training, but depending on situations. The effect of training on computer self-efficacy is changed by the conceptions of users' ability (Martocchio, 1994) and the type of software (Compeau and Higgins, 1995b) and the type of training (Gist et al., 1989). Nevertheless, most research has supported the positive relationship between training and computer self-efficacy (Bedard et al., 2003; Igbaria and livari, 1995; Torkzadeh and Kofteros, 1994). Thus, we hypothesize that software feature training would increase beliefs on software features and probably beliefs on software.

H1a. Software feature training has a positive influence on feature beliefs.
H1b. Software feature training has a positive influence on software beliefs.

Feature Use

Software training improves not only beliefs but also the use of software features. Organizational support has indirect effects on usage through computer self-efficacy (Igbaria and livari, 1995). Training increases the computer self-efficacy and task self-efficacy, and computer self-efficacy increases the intention to use the electronic audit work system through increased perceived ease of use (Bedard et al., 2003). Facilitating conditions significantly affect the adaptive use of a system (Sun and Zhang, 2008). However, facilitating conditions do not affect the utilization of the computer (Thompson et al., 1991). But, most research has supported the positive relationship between training and system use, so we hypothesize that software feature training would increase the use of a software feature.

H2. Software feature training has a positive influence on feature use.

Task Performance

Training improves task performance as demonstrated in prior research. Training improved the performance of ratio analysis (Bonner and Walker, 1994) and real estate valuation (Earley, 2001, 2003). Training increased the performance of analytical procedure tasks of senior auditing students (Moreno et al., 2007). Computer training had a positive relationship with successful IS implementation (Bostrom et al., 1990; Compeau et al., 1995). Computer training increased computer self-efficacy and performance (Compeau and Higgins, 1995b). IT training had a positive association with the productivity of university faculty (Gilmore, 1998). Although an inconsistent effect of training on IS implementation success has been found due to technical complexity and task interdependence (Sharma and Yetton, 2007), the positive effects were supported. So we hypothesize that software feature training would increase the performance of tasks.

H3a. Software feature training has a positive influence on performance score.
H3b. Software feature training has a negative influence on performance time.

Figure 1 summarizes the hypotheses that we formulated in our research. There were no definitions of feature and software beliefs, so we define feature beliefs as users' beliefs about software features including perceived usefulness, perceived ease of use, and self-efficacy on software features. Software beliefs are defined as users' beliefs about a software package including perceived usefulness, perceived ease of use, and self-efficacy on software. The effect of software feature training on beliefs, use, and performance were examined in this study.

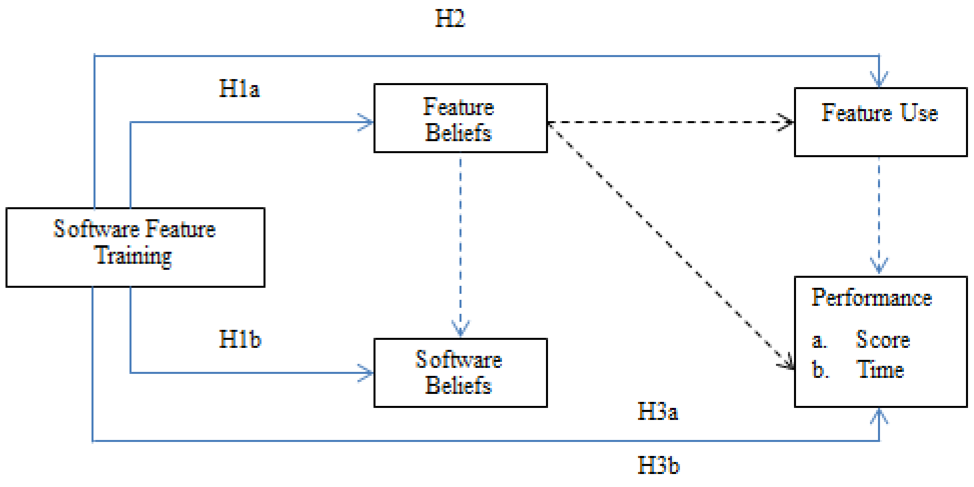


Figure 1. Software Feature Training Model

RESEARCH METHODOLOGY

We used the two-group posttest-only randomized experiment, which is a simple but effective experimental design (Christensen, L.B. 1994), with a treatment group (group A) and a comparison group (group B). Participants who did not have a basic knowledge of GAS were excluded. Once pre-screened, participants were randomly assigned to either group A or group B. Training was only given to the treatment group, followed by administration of the survey and performance test for both groups.

Subjects

We advertised free online training on Benford's Law as part of our research project through the websites or the mailing lists of GAS users[2]. The announcement explained the purpose of our experiment and provided a link to the training website. Before completing the survey and performance test, participants indicated their consent about the willingness to participate in this experimental study. A total of 161 GAS users participated in our training. Incomplete data was excluded, so finally 56 participants (28 in each group) were used for analysis (33.78% of total participants). The sample size of 56 is small but still adequate for an experimental research to compare the result of two groups. The summary of participants is available in Appendix 1.

Software Feature Training

Software feature training consisted of a lecture (approximately 15 minutes) and a self-tutorial (approximately 45 minutes) on Benford's Law. The lecture covered the conceptual backgrounds of Benford's Law. The self-tutorial covered the practical use of Benford's Law and methods to interpret the results of Benford's Law analysis. This tutorial material[3] included one guided exercise and an additional practice case.

Survey and Performance Test

The survey consists of three sections: demographic information, software beliefs, and feature beliefs. The demographic information section includes questions about age, gender, education, occupation, audit experience, software experience, and Benford's Law experience. The software beliefs section includes questions about perceived usefulness, perceived ease of use, and computer self-efficacy on GAS, and the feature beliefs section includes questions about perceived usefulness, perceived ease of use, and computer self-efficacy on advanced features.

The performance test includes six audit tasks with two kinds of questions: test questions and feature usage questions on each task. In the feature usage questions, the frequency of using eleven software features to complete each task was asked. Table 1 introduces the six audit tasks[4] that were used in the performance test. Task 4 is a most similar task to the trained task.

Tasks	**Descriptions**
Task1	Import files.
Task2	Identify unusual transactions.
Task3	Identify high payments.
Task4	Identify unauthorized employees.
Task5	Search for gaps in the check number sequence.
Task6	Test payments to unauthorized suppliers.

Table 1. Tested Audit Tasks

Table 2 shows eleven software features selected from GAS to examine the frequency of using software features. The training group was expected to increase the use of Benford's Law in Task 4 after training.

[2] GAS users include the members of ACL User Group, IDEA User Group, the Institute of Internal Auditors (IIA), Association of Certified Fraud Examiners (ACFE), Association of College and University Auditors (ACUA), or Information Systems Audit and Control Association (ISACA).

[3] The tutorial material is in press in *A Compendium of Classroom Cases and Tools*.

[4] The tested audit tasks were adopted from IDEA workbook version 7 and modified to meet the needs of this research.

Features	**Descriptions**[5]
Import	Imports a file into GAS.
Extraction/filter	Extracts specified data.
Summarization	Summarizes data based on a specified field.
Stratification	Counts the number of records falling within specified intervals.
Graph	Creates, formats, and edits a graph using data.
Sort	Sorts a file in ascending or descending order.
Total fields / Control total	Counts the total number of records.
Duplication key	Identifies duplicate records.
Gap detection	Identifies gaps within a specified field.
Join files/Join database	Combines two different files into a single file.
Benford's Law	Gives auditors the expected frequencies of the digits in tabulated data.

Table 2. Tested Software Features

Table 3 summarizes the expected feature use by tested audit tasks.

	Task 1	*Task 2*	*Task 3*	*Task 4*	*Task 5*	*Task 6*
Import	√					
Extraction			√			
Summarization	√					
Stratification		√				
Graph				√		
Sort						
Total fields	√		√			√
Duplication						
Gap detection					√	
Join files						√
Benford's Law				√		

Table 3. Expected Feature Use by Audit Tasks

Instrumentation

Feature and Software Beliefs

Perceived usefulness and perceived ease of use were measured using the items from Taylor and Todd (1995)'s research. Computer self-efficacy was extracted from Bedard et al. (2003) and Compeau and Higgins (1995a)'s research. Each construct was measured by 2 indicator variables with the 7-point Likert scale ranging from "strongly disagree" to "strongly agree."

[5] The feature descriptions were retrieved from GAS Review (http://www.auditsoftware.net/improving-audits.html).

Feature Use

Feature use was measured using the items from Davis et al. (1989)'s research. The frequency of 11 software features that were used in each audit task was measured using 7-point Likert scale ranging from "never" to "very frequently."

Performance

Task performance was measured using a test instrument. Performance is consists of two variable, score and time. Test score was graded after participants finished their test. Each audit task had 10 points. Test time was recorded by participants after they complete each task.

Data Analysis

Data collected from the survey and performance test were analyzed using two stages of assessments. First, the reliability and validity of the measures were examined. Then, ANOVA was applied to analyze the impact of software feature training on beliefs, use, and performance. PASW Statistics 18 was used through the whole process.

RESULTS

Group Characteristics

Group A and B were comparable prior to training except for audit experience. The ANOVA results indicated that CPA (p=.281), age (p=.082), gender (p=.755), education (p=.295), occupation (p=.139), GAS experience (p=.712), Benford's Law experience (p=.074), and software type (p=.233) were comparable in both groups, but audit experience (p=.016) was significantly higher in group A. However, audit experience did not have a significant influence on the test score of group A.

Test of Measurement

Reliability assessment was conducted using the Cronbach's alpha of belief constructs. Feature beliefs (α=.913) and software beliefs (α=.851) exceeded the generally accepted level of 0.7 (Fornell et al. 1981). The validity assessment for belief measures was conducted using the factor analysis. The rotated component matrix indicated that feature beliefs and software beliefs are distinct constructs. All feature beliefs (FPU1=.827; FPU2=.745; FPEOU1=.863; FPEOU2=.758; FSE1=.799; FSE2=.812) were highly correlated with the first factor and all software beliefs (SPU1=.739; SPU2=.680; SPEOU1=.837; SPEOU2=.720; SSE1=.640; SSE2=.711) were highly correlated with the second factor.

Effect of Software Feature Training on Beliefs

Software feature training had no detectable impact on software beliefs but had an impact on feature beliefs. The perceived usefulness (FU1=.000; FU2 = .000), perceived ease of use (FEOU1=.000; FEOU2 = .000), and self-efficacy (FSE1=.002; FSE2=.000) on software features of group A were significantly higher than those of group B. Consistent with H1a, software feature training has a positive influence on feature beliefs. However, the perceived usefulness (SU1=.281; SU2=.240), perceived ease of use (SEOU1=.308; SEOU2=.098), and self-efficacy (SSE1=.603; SSE2 = .160) on GAS of group A were not significantly higher than those of group B. Inconsistent with H1b, software feature training has no influence on software beliefs.

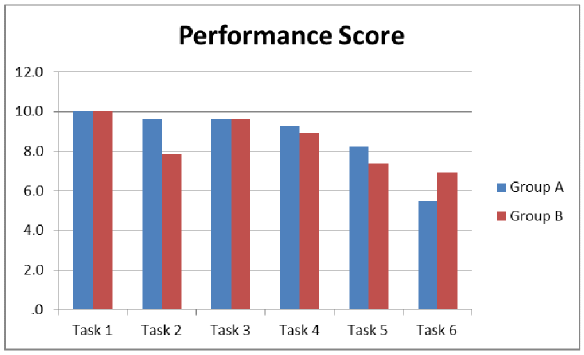
Effect of Software Feature Training on Feature Use

Software feature training had significant influences on the use of Benford's Law not only in Task 4 but also in Task 2. The use of Stratification (p=.004), Graph (p=.016), and Benford's Law (p=.000) was significantly different in Task 2 because group B had more used Stratification while group A had more used Benford's Law and Graph to complete Task 2. The use of Graph (p=.002) and Benford's Law (p=.000) was significantly different in Task 4 because group A had more used Graph and Benford's Law after training. Consistent with H2, software feature training had a positive influence on feature use.

Effect of Software Feature Training on Performance

Software feature training had no influence on test score and a positive influence on test time for Task 4. The test score of group A was higher than that of group B but it was not statistically significant (p=.647) and the test time of group A was significantly higher than that of group B for Task4 (p=.031). Group A had a little bit better score, but took more time to complete the task than group B. Training had increased scores and time but did not solely attribute the increase to the training (test time of Task 4, η 2 =.085; test score of Task 2, η 2 =.073). Inconsistent with H3a and H3b, software feature training had no influence on performance score and a positive influence on performance time.

However, the average score and time of group A and group B were different by tasks (Figure 2). The average test score of group A was higher in Task 2, Task 3, and Task 4; same in Task 1 and Task 5; lower in Task 6. We assumed that task is another factor that affected the performance score of tasks. The test time was also affected by tasks. The average test time of group A was higher in Task 1, Task 3, Task 4, and Task 6; lower in Task 2 and Task 5. In addition to the effect of training on performance, task is probably another factor that impacts performance score and time.



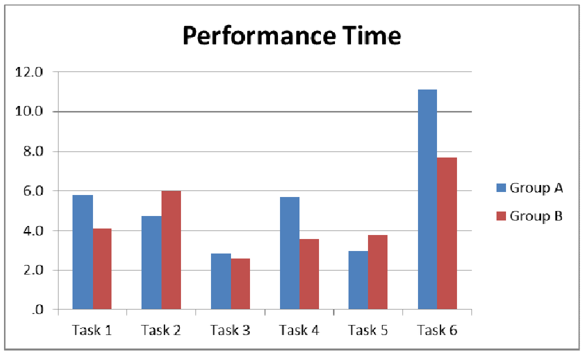


Figure 2. Average Score and Time by Tasks

DISCUSSION

Table 4 summarizes the hypothesis testing results. Although the majority of hypotheses have been contradicted, previous research has explained similar phenomena.

Hypothesis		Results
H1a	Software feature training has a positive influence on feature beliefs.	O
H1b	Software feature training has a positive influence on software beliefs.	X
H2	Software feature training has a positive influence on feature use.	O
H3a	Software feature training has a positive influence on performance score.	X
H3b	Software feature training has a negative influence on performance time.	X

Table 4. Hypothesis Testing Results

O: Consistent with Hypothesis | X: Inconsistent with Hypothesis

Contrary to our expectations, software beliefs were not affected by software feature training (H1b) while feature beliefs were significantly affected by software feature training (H1a). As Harrison and Datta (2007) mentioned, it seems like beliefs on software features are different from beliefs on a software application. Although software is a sum of software features, beliefs on software are different from beliefs on software features. So only the beliefs on software features are affected by software feature training.

As anticipated, the use of software features was increased by software feature training (H2). The extended use of features (Saga and Zmud, 1994) and feature extension to other tasks that designers had not intended (Jasperson et al. 2005) were observed after training. Auditors had increased the number of software features to complete the task and extended the use of software features to a new task after training. IT users created new knowledge by merging, categorizing, reclassifying, and synthesizing existing knowledge (Alavi and Leidner, 2001). Software feature training may amplify the knowledge of GAS users as integrating newly acquired software skills with existing task knowledge, thus resulted in feature and task extension.

Software feature training had no influence on performance score (H3a) and a positive influence on performance time (H3b). Different from our expectation, the test score was not significantly increased or the test time was not significantly decreased for Task 4, which was similar to the trained task, after training. As Hiltz and Turoff (1981) and Kay and Thomas (1995) argued, the time gap between training and test may affect score and time. We conducted the test right after training, before

GAS users became accustomed to using the new software feature. Thus, users may have spent more time to practice this feature.

The effects of task on software training were found through additional analysis. The test scores of most tasks (Task 2, 4, and 5) were positively affected by training, but there were still other tasks that were not affected (Task 1 and 3) or negatively affected (Task 6) by training. The test time of most tasks (Task 1, 3, 4 and 6) was increased while that of other tasks (Task 2 and 5) was decreased after training. Task is a factor that modifies system use (As Sun and Zhang, 2008) and task interdependence is important on IS implementation success (Sharma and Yetton, 2007). We assumed that task might mediate the effect of training on performance.

We found that training had more effect on Task 2 than Task 4, which was different from our expectation. We expected that the test score of Task 4 would be most affected by training, but training was more effective on Task 2. These results indicated some potential lessons about training materials. The task design from the training materials was not that effective. The training materials indicated that stratification should be used for Task 2, but users found a new way to efficiently use Benford's Law for a stratification task.

SUMMARY

This research provides empirical evidence about technology features showing the impact of software features on software training. Software feature training had changed the GAS users' beliefs on software features. Consequently, users had more frequently used the new software feature with software features already known. They had also applied this new software feature not only to the already known task but also to a new task. However, training had no immediate effect on performance score and time because GAS users spent more time to practice the new software feature and did not know exactly when they could apply this feature to new tasks.

Furthermore, this study substantially contributes to technology training research by adding new perspectives to training research. Applying the feature centric approach of technology, the classification of computer self-efficacy was extended to the feature level. Computer self-efficacy was categorized into software self-efficacy and feature self-efficacy. For task perspective, we tested 6 audit tasks to examine the effect of software feature training. Training had no effect on the performance of the trained task in a short period of time, but the effect of training on performance was affected by the type of tasks. Although the analysis results were contradicted in some tasks, this research provided opportunities on future research to understand the effect of tasks.

We are interested in a number of extensions to this research. First, this study can be extended to the effect of software types such as ACL and IDEA. Second, this research can be applied to other professionals to compare the influences of software features among different professional groups. Third, other training constructs, such as outcome expectation and prior experiences, can be added to this training research. Fourth, other advanced features such as data mining can be added to training. Fifth, an important extension is to design studies to improve understanding about the relationship between training, use and performance. Lastly, the limitation of this experiment can be improved with onsite training, objective measures, and different types of tasks.

REFERENCES

1. Agarwal, R. and Prasad, J. (1999) Are individual differences germane to the acceptance of new information technologies? *Decision Sciences* 30, 2, 361-391.
2. Alavi, M., and Leidner, D. E. (2001) Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues, *MIS Quarterly* 25, 1, 107-136.
3. Barki, H., Titah, R., and Boffo, C. (2007) Information system use-related activity: An expanded behavioral conceptualization of information system use, *Information System Research* 18, 2, 173-192.
4. Bedard, J. C., Jackson, C., Ettredge, M. L, and Johnstone, K. M. (2003) The effect of training on auditors' acceptance of an electronic work system, *International Journal of Accounting Information Systems* 4, 4, 227-250.
5. Bolt, M. A., Killough, L. A., and Koh, H. C. (2001) Testing the interaction effects of task complexity in computer training using the social cognitive model, *Decision Sciences* 32, 1, 1-20.
6. Bonner, S. E. and Walker, P. L. (1994) The effects of instruction and experience on the acquisition of auditing knowledge, *Accounting Review* 69, 1, 157-178.
7. Bostrom, R. P., Olfman, L., and Sein, M. (1990) The importance of learning style in end-user training, *MIS Quarterly* 14, 1, 101-119.
8. Burton-Jones, A and Straub, D.W. (2006) Reconceptualizing system usage: An approach and empirical test, *Information Systems Research* 17, 3, 228-246.
9. Chau, P.Y.K. (1996) An empirical assessment of a modified technology acceptance model, *Journal of Management Information Systems* 13, 2, 185-204.
10. Christensen, L.B. (1994) Experimental methodology, Sixth ed., Needham Heights, Mass.: Allyn and Bacon.
11. Compeau, D. R. and Higgins, C. A. (1995) Computer self-efficacy: development of a measure and initial test, *MIS Quarterly* 19, 2, 189–211.
12. Compeau, D. R. and Higgins, C. A. (1995) Application of social cognitive theory to training for computer skills, *Information Systems Research* 6, 2, 118-143.
13. Compeau, D. R., Olfman, L., Sein, M., and Webster, J. (1995) End-user training and learning, *Communications of the ACM* 38, 7, 25-26.
14. Davis F. D., Bagozzi R. P., and Warshaw P. R. (1989) User acceptance of computer technology: A comparison of two theoretical models, *Management Science* 35, 8, 982-1002.
15. DeSanctis, G., and Poole, M.S. (1994) Capturing the complexity in advanced technology use: Adaptive structuration theory, *Organization Science* 5, 2, 121-147.
16. Earley, C. E. (2001) Knowledge acquisition in auditing: Training novice auditors to recognize cue relationship in real estate valuation, *The Accounting Review* 76, 1, 81-97.
17. Earley, C. E. (2003) A note on self-explanation as a training tool for novice auditors: The effects of outcome feedback timing and level of reasoning on performance, *Behavioral Research in Accounting* 15, 111-124.
18. Fornell, C. and Larcker, D. (1981) Evaluating structural equation models with unobservable variables and measurement error, *Journal of Marketing Research* 18, 39-50.
19. Gilmore, E. (1998) Impact of training on the information technology attitudes of university faculty, Doctoral dissertation, University of North Texas, Denton.
20. Gist, M. E., Schwoerer, C., and Rosen, B. (1989) Effects of alternative training methods on self-efficacy and performance in computer software training, *Journal of Applied Psychology* 74, 6, 884-891.
21. Griffith, T. L. (1999) Technology features as triggers for sensemaking, *Academy of Management Review* 24, 3, 472–488.
22. Harrison, M. J. and Datta, P. (2007) An empirical assessment of user perceptions of feature versus application level usage, *Communications of the Association for Information Systems* 20, 300-321.

23. Hiltz, S. R. and Turoff, M. (1981) The evolution of user behavior in a computerized conferencing center, *Communication of the ACM* 24, 11, 739-751.
24. Hsieh, J. P. and Robert, J. P. (2006) Understanding post-adoptive usage behaviors: A two-dimensional view, *DIGIT 2006 Proceedings* (Paper 3).
25. Igbaria, M. and Iivary, J. (1995) The effects of self-efficacy on computer usage, *Omega-International Journal of Management Science* 23, 6, 587-605.
26. Igbaria, M., Zinatelli, N., Cragg, P., and Cavaye, A. (1997) Personal computing acceptance factors in small firms: A structural equation model, *MIS Quarterly* 21, 3, 279–302.
27. Jasperson, J. S., Carter, P. E., and Zmud, R. W. (2005) A comprehensive conceptualization of post-adoptive behaviors associated with information technology enabled work systems, *MIS Quarterly* 29, 3, 525-557.
28. Kay, J. and Thomas, R. C. (1995) Studying long-term system use, *Communications of the ACM* 38, 7, 61-69.
29. Kim, H.J., Mannino, M., and Nieschwietz, R.J. (2009) Information technology acceptance in the internal audit profession: Impact of technology features and complexity, *International Journal of Accounting Information System* 10, 4, 214-228.
30. Martocchio, J. J. (1994) Effects of conceptions of ability on anxiety, self-efficacy, and learning in training, *The Journal of applied psychology* 79, 6, 819-825.
31. Moreno, K. K., Bhattacharjee, S., and Brandon, D.M. (2007) The effectiveness of alternative training techniques on analytical procedures performance, *Contemporary Accounting Research* 24, Fall, 983-1014.
32. Saga, V. L., and Zmud, R. W. (1994) The nature and determinants of IT acceptance, routinization, and infusion. in Diffusion Transfer and Implementation of Information Technology, North-Holland, Amsterdam, 67-86.
33. Sharma, R. and Yetton, P. (2007) The contingent effects of training, technical complexity, and task interdependence on successful information systems implementation, *MIS Quarterly* 31, 2, 219-238.
34. Sun, H. and Zhang, P. (2008) Adaptive System Use; An investigation at the system feature level. *ICIS 2008 Proceedings*, Paper 170.
35. Taylor, S. and Todd, P. (1995) Understanding information technology usage: A test of competing models, *Information Systems Research* 6, 2, 144–76.
36. Thompson, R. L., Higgins, C. A., and Howell, J. M. (1991) Personal computing: Toward a conceptual model of utilization, *MIS Quarterly* 15, 1, 125-143.
37. Torkzadeh, G. and Koufteros, X. (1994) Factorial validity of a computer self-efficacy scale and the impact of computer training, *Educational and Psychological Measurement* 54, 3, 813-821.

APPENDIX 1. SUMMARY OF PARTICIPANTS

Demographic variable		Percent	Valid %[6]
CPA	Yes	30.4%	34.0%
	No	58.9%	66.0%
Age	18-29	16.1%	18.4%
	30-39	37.5%	42.9%
	40-49	14.3%	16.3%
	50-59	16.1%	18.4%
	60 and over	3.6%	4.1%
Gender	Male	39.3%	44.0%
	Female	50.0%	56.0%
Education	High school	0%	0%
	Some College	1.8%	2.0%
	Associate Degree	0%	0%
	Bacheror's Degree	57.1%	64.0%
	Master's Degree	28.6%	32.0%
	Doctorate/Ph.D.	1.8%	2.0%
Job	Audit staff	39.3%	44.0%
	Audit manager	5.4%	6.0%
	Audit director	5.4%	6.0%
	IT audit staff	7.1%	8.0%
	IT audit manager	7.1%	8.0%
	IT audit director	1.8%	2.0%
	IT professional	5.4%	6.0%
	Student	1.8%	2.0%
	Others	16.1%	18.0%
Audit Experiences	Less than 1	8.9%	10.0%
	1-2	10.7%	12.0%
	3-5	21.4%	24.0%
	6-10	19.6%	22.0%
	11-15	16.1%	18.0%
	More th15	12.5%	14.0%
GAS Experiences	Less than 1	10.7%	12.2%
	1-2	28.6%	32.7%
	3-5	30.4%	34.7%
	6-10	8.9%	10.2%
	11-15	5.4%	6.1%
	More than 15	3.6%	4.1%
Benford's Law Experiences	Less than 1	62.5%	70.0%
	1-2	14.3%	16.0%
	3-5	10.7%	12.0%
	6-10	1.8%	2.0%
	11-15	0%	0%
	More than 15	0%	0%
GAS Type	ACL	87.5%	87.5%
	IDEA	12.5%	12.5%

[6] Valid Percent shows the percentage excluded missing values.